

# Bachelorarbeit

# HSR Videowall

Abteilung Informatik  
Hochschule für Technik Rapperswil

Frühjahrssemester 2012



<b>Autoren</b>	Lukas Elmer, Christina Heidt, Delia Treichler
<b>Betreuer</b>	Prof Dr. Markus Stolze
<b>Projektpartner</b>	HSR, Rapperswil, SG
<b>Experte</b>	Markus Flückiger, Zühlke
<b>Gegenleser</b>	Prof. Dr. Peter Heinzmann

# I. Inhaltsverzeichnis

I.	Inhaltsverzeichnis .....	1
II.	Abstract .....	2
III.	Danksagung .....	3
IV.	Management Summary .....	4
V.	Technischer Bericht .....	11
VI.	Projekt Retrospektive.....	197
VII.	Verzeichnisse .....	234
VIII.	Anhang.....	239

## II. Abstract

Grosse Monitorkonstellationen bieten die Möglichkeit, Inhalte auf attraktive und imposante Weise zu präsentieren. Mittels Microsoft Kinect ergibt sich eine neue Art der Steuerung: Eine Anwendung kann mit Körperbewegungen anstatt Tastatur, Maus oder Touch bedient werden. Die Vereinigung von einer Monitorwand und Kinect – nachfolgend als Videowall bezeichnet – bietet eine neuartige Präsentations- und Interaktionsmöglichkeit.

Die HSR wollte mit der Bachelorarbeit „HSR Videowall“ die technische Machbarkeit einer solchen Videowall und deren Nutzen für die Hochschule abklären. Diese Arbeit beinhaltet daher Abklärungen in drei Bereichen:

- **Technologie Grafikkarten/Auflösung**  
Zur Ermittlung der optimalen Auflösungen wurden Tests mit den eigens für die Bachelorarbeit gekauften Grafikkarten durchgeführt.
- **Nutzerbedürfnisse und Interaktion**  
Um die Bedürfnisse der zukünftigen Nutzer zu untersuchen, wurden Fragebögen verteilt und ausgewertet. Mit Kinect wurden verschiedene Benutzerstudien durchgeführt.
- **Softwaretechnologie**  
Die Inhalte der Videowall sollen aktuell und interaktiv sein. Um zu demonstrieren, wie Softwarekomponenten dynamisch in die Applikation eingebracht werden können, wurde ein Plug-in System aufbauend auf C# mit MEF und Unity entwickelt.

# III. Danksagung

**Prof. Dr. Markus Stolze** für die kompetente und partnerschaftliche Betreuung und sein wertvolles und konstruktives Feedback.

**Markus Flückiger** für die Unterstützung und die tollen Ideen und die Sicht über den Tellerrand hinaus.

**Michael Gfeller und Silvan Gehrig** für die Code Reviews und die positive Kritik am Code und an der Architektur.

**Kevin Gaunt** für die Ideen und die tatkräftige Unterstützung beim Imagine Cup.

**Marion Schleifer** für das Korrekturlesen der Bachelorarbeit.

**Allen an den Usability Tests beteiligten Personen** für die Teilnahme an den Usability Tests und die wertvollen Inputs.

**Allen an der Umfrage beteiligten Personen** für die Teilnahme an der Befragung.

# IV. Management Summary

## IV.1 Ausgangslage

Neue Technologien führen zu neuen Präsentationsmöglichkeiten. Durch den Einsatz dieser Technologien werden Innovation und Wissen über den neusten Stand der Technik demonstriert. Beide Faktoren spielen eine wichtige Rolle für eine technische Hochschule. Wer würde sein Studium an einer Schule beginnen, welche über keine Beamer in den Hörsälen, sondern lediglich Hellraumprojektoren verfügt? Oder an einer, an welcher alle Übungsräume mit Röhrenbildschirmen ausgestattet sind?

Eine moderne Hochschule soll zum einen bei den Besuchern einen positiven Eindruck hinterlassen, zum anderen aber auch bei den Studenten und Angestellten. Eine Möglichkeit, sich als moderne Hochschule zu profilieren, ist die Nutzung von innovativen Präsentationstechniken. Durch ihre ständige Anwesenheit stellen Studenten und Angestellte die Hauptzielgruppe für Präsentationen dar. Sinnvolle Präsentationsinhalte wären einerseits Informationen aus den verschiedenen Studiengängen. Andererseits sind auch Inhalte denkbar, welche den Alltag vereinfachen oder erheitern.

Um die Nutzung innovativer Präsentationsmöglichkeiten zu ermöglichen, plant die HSR eine interaktive Videowall im Eingangsbereich des Verwaltungsgebäudes der HSR. Dieses Gebäude ist ein attraktiver Standort, da sich dort die Mensa, der Empfang und die Aula befinden. Um zu den erwähnten Räumen zu gelangen, muss der Eingangsbereich, welcher ein relativ breiter Gang ist, passiert werden. Dieser Bereich stellt den idealen Ort dar, um die Videowall aufzustellen.

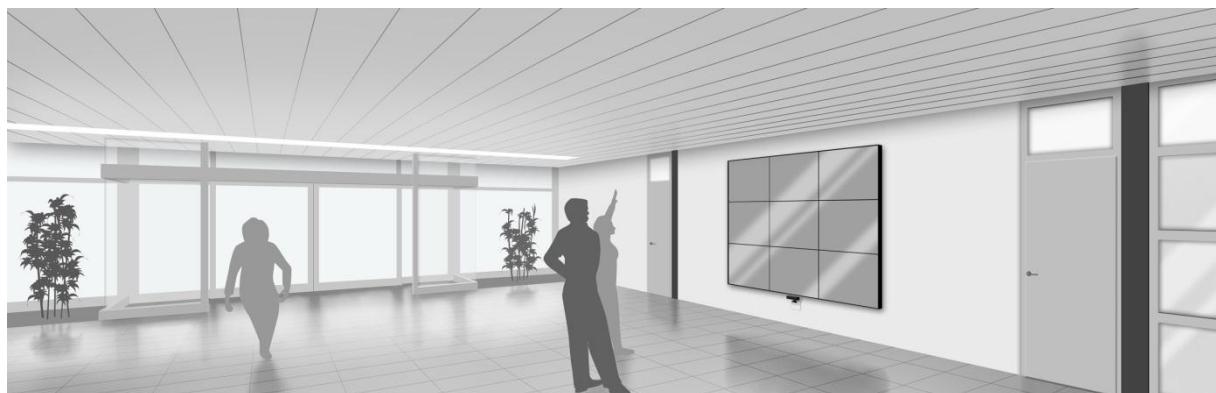


Abbildung 1 - Videowall im Eingangsbereich des Verwaltungsgebäudes

## IV.2 Vorgehen, Technologien

Diese Arbeit evaluiert das Potential von Kinect als Steuerungsgerät für die Videowall. Kinect ist ein Gerät, mit dessen Hilfe Applikationen durch Körperbewegungen und Sprache gesteuert werden können. Da es sich hierbei um eine Microsoft-Technologie handelt, wurde zur Entwicklung der Applikation WPF und .NET gewählt.



Abbildung 2 - Kinect, Bildquelle: [www.wikipedia.org](http://www.wikipedia.org)

Die Videowall soll, in Blickrichtung Mensa, an der linken Wand des Eingangsbereichs des Verwaltungsgebäudes der HSR installiert werden. Damit die Passanten von Kinect erkannt werden und so die Videowall durch Körperbewegungen steuern können, müssen die sich im Erkennungsbereich des Gerätes aufhalten. Aus diesem Grund wurde zu Beginn beobachtet, wie sich die Personen, die sich im Raum aufhalten, verhalten. Es wurden der rechtwinklige Abstand, welchen die Passanten zur Wand haben, und die Gruppengrößen, in denen Personen den Raum passieren, analysiert. Weiter wurde der Skelett-Erkennungsbereich des Kinect Sensors ausgemessen. Dies ist der Bereich, in dem Personen erkannt werden und deren Körper als Skelett interpretiert wird.

Als initiale Anforderung an die Videowall wurde vom Auftraggeber die Präsentation der Bachelorposter definiert. Es war daher abzuklären, wie gross das Interesse der Studenten an den Postern ist. Des Weiteren stellte sich auch die Frage, ob Videos sich nicht wesentlich besser zur Präsentation der Arbeiten auf der Videowall eignen würden. Die durchgeführte Befragung sollte auch klären, ob Studenten dazu bereit wären, Videos über ihre Arbeiten zu erstellen. Aus den Antworten der vom Team verteilten Fragebögen an Studenten der HSR wurde ersichtlich, dass sich nur etwa die Hälfte der befragten Studenten für die Poster interessieren und dass der Wille, ein Video zu erstellen, gering ist. Es wurde auch festgestellt, dass für die Poster eines Studienganges sehr kleine Schriftgrößen verwendet werden und so das Lesen des Textes erschwert bis gar nicht möglich ist. Trotz diesen Resultaten wurde die Idee der Präsentation der Bachelorposter auf der Videowall weiter ausgearbeitet, da sich mit dieser Applikation alle Studiengänge der HSR auf der Videowall präsentieren können.

Im Zuge des Projekts wurden weitere Ideen für Inhalte für die Wall erarbeitet. Da sich die Videowall im gleichen Gebäude wie die Mensa befindet, erschien es sinnvoll, auf der Videowall, zusätzlich zu den Bachelorpostern, das Mittagsmenu anzuzeigen.

Die Inhalte der Videowall sollen aktuell und interaktiv sein. Daher soll die Videowall-Applikation einfach um weitere Inhalte erweiterbar sein. Institute und auch interessierte Studenten hätten so die Möglichkeit, selbst entwickelte Applikationen mit Hilfe der Videowall einem grösseren Publikum zu präsentieren. Aus diesem Grund wurde ein Plug-in System erarbeitet. Wenn die entwickelte Applikation ein bestimmtes Interface implementiert und mit bestimmten Schlüsselwörtern ausgestattet ist, kann sie automatisch zur Videowall-Applikation hinzugefügt werden.

Für die Videowall-Monitore wurde die ideale Grösse und Konstellation gesucht. Mit einem Hellraumprojektor wurden verschiedene Varianten von Konstellationen an die Wand des Eingangsbereichs projiziert (siehe Abbildung 3 - Projektion der 3 x 3 55" Monitorkonstellation im Eingangsbereich des Verwaltungsgebäudes). Somit konnte besser abgeschätzt werden, wie sich die Videowall später in den Raum eingeben würde.



Abbildung 3 - Projektion der 3 x 3 55" Monitorkonstellation im Eingangsbereich des Verwaltungsgebäudes

Um die ideale Konfiguration für die ausgewählten Grafikkarten und Monitore zu eruieren, baute das Team eine Test-Videowall in seinem Bachelor-Arbeitszimmer auf (siehe Abbildung 4 - Testhardware). Durch Tests mit verschiedenen Treibereinstellungen und Auflösungen wurde nach der idealen Hardwarekonfiguration gesucht und mehrere Lösungsvorschläge erarbeitet.



Abbildung 4 - Testhardware

Bei der Videowall-Anwendung steht der Nutzer im Zentrum. Die Bedienung soll für ihn einfach verständlich sein. Auch die Inhalte sollen für ihn interessant sein und auf eine spannende Weise dargeboten werden, damit die Videowall immer wieder genutzt wird. Ein Demomodus soll Personen zur Videowall locken. Zur Prüfung der Einfachheit und Verständlichkeit der Steuerung und der Wirkung des Demomodus wurden Usability Tests durchgeführt.

Die Inhalte der Videowall müssen verwaltet werden. Das Sekretariat der HSR arbeitet bereits mit einem Typo3 CMS. Aus diesem Grund wurden die verschiedenen Varianten der Integration der Videowall-Administration in das vorhandene System beschrieben.

## IV.3 Ergebnisse

Die Teammitglieder arbeiteten bereits in ihrem 5. Semester mit WPF und .NET und konnten die dort gesammelten Erfahrungen für dieses Projekt nutzen. Der Kinect Sensor sowie die zu erarbeitende Hardwarekonfiguration der Videowall stellten aber neue Herausforderungen an das Team.

Durch die Passantenanalyse konnte bestätigt werden, dass etwa die Hälfte der Personen im Eingangsbereich den Skelett-Erkennungsbereich des Kinects passieren.

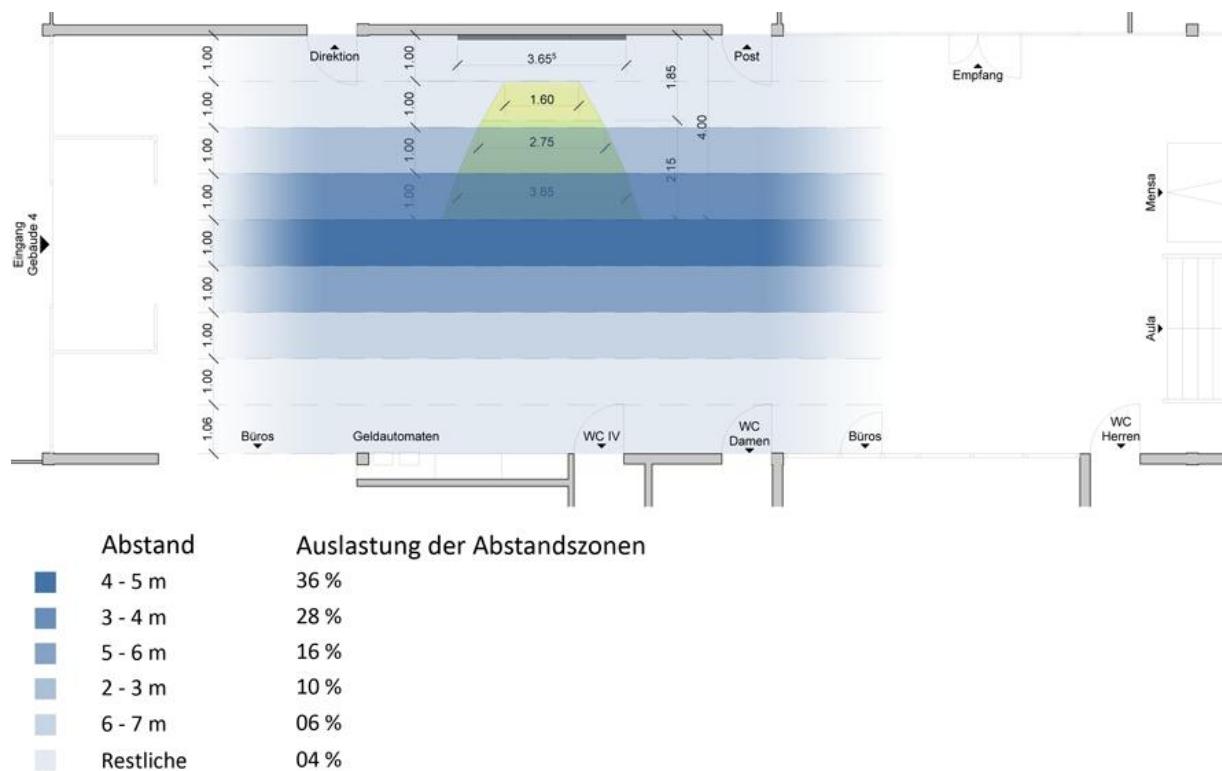


Abbildung 5 - Auslastung der Abstandszonen (aus Passantenanalyse) und Kinect Skelett-Erkennungsbereich

In der Bachelorarbeit wurde ein funktionstüchtiger Prototyp erarbeitet. Mit diesem sollte die technische Machbarkeit aufgezeigt werden. Der Prototyp wurde so aufgebaut, dass er im Falle eines positiven Entscheids für die Videowall durch das Institut für Software (IFS) einfach weiterentwickelt werden kann.

Auf dem Prototypen können Bachelorposter angesehen werden und man kann sich über das aktuelle Mittagsmenü der Mensa informieren. Ein Demomodus ist aktiv wenn keine Personen von Kinect erkannt werden. Er dient dem Anlocken der Passanten, damit diese mit der Videowall interagieren.

Das zusätzlich erarbeitete, einfach einsetzbare Plug-in System bietet anderen Entwicklern die Möglichkeit, ihre Inhalte auf unkomplizierte Weise zur Videowall hinzuzufügen und zu präsentieren.

Als ideale Monitorkonstellation wird eine 3 x 3 55"-Monitorkonstellation vorgeschlagen. Sie bringt sich einerseits gut in den Raum ein, andererseits werden damit klassische Formate, wie beispielsweise Video, gut unterstützt. Durch das Arbeiten mit der Test-Videowall wurde festgestellt, dass eine hohe Auflösung der Monitore und gleichzeitig eine hohe Performance der Applikation schwierig in Einklang zu bringen sind. Es konnten dennoch zwei Varianten erarbeitet werden, die je nach Bedürfnis eingesetzt werden können. Die eine bietet eine hohe Auflösung, Animationen funktionieren jedoch nur beschränkt. Bei der zweiten Variante ist die Auflösung beschränkt, Animationen sind dank guter Performance aber problemlos möglich.



**Abbildung 6 - Usability Test**

Durch die durchgeführten Usability Tests konnte bestätigt werden, dass die Steuerung mittels Kinect einfach verständlich ist. Auch die positive Auswirkung des Demomodus auf das Interesse der Passanten an der Videowall wurde mit einem solchen Test validiert.

Für die Bachelorarbeit wurden verschiedenste Analysen durchgeführt. Aufgrund des beschränkten Zeitrahmens war es erforderlich, diese zu priorisieren, was oftmals schwierig war. Trotz dieser Herausforderung ist es gelungen, viele neue Erkenntnisse zu schaffen und einen funktionstüchtigen Prototyp zu erstellen. Der Prototyp bietet eine dynamische Erweiterbarkeit in Form eines Plug-in Frameworks. Dazu bestehen zwei Plug-in Applikationen, mit der einen können die Bachelorposter angeschaut werden, in der anderen kann man sich über Mittagsmenü der Mensa informieren.

## IV.4 Ausblick

Diese Bachelorarbeit ist eine Machbarkeitsstudie. Mit ihr wurde eruiert, ob eine Anschaffung einer Videowall für die HSR sinnvoll ist, was im Laufe der Arbeit erwiesen werden konnte. Die Machbarkeitsstudie ist die Grundlage für eine mögliche Weiterentwicklung des Projektes durch das Institut für Software (IFS).

Bei einer Weiterführung der Videowall muss primär ein Content Management zur Administration der Inhalte der Videowall entwickelt werden. Zudem ist bei den Hardwarekomponenten eine definitive Entscheidung für eine bestimmte Konfiguration zu treffen. Die Videowall verfügt derzeit über zwei Inhalte, die Poster-Applikation und das Mittagsmenu der Mensa. Abzuklären wäre hierbei, ob weitere Applikationen zum Grundumfang der Videowall-Anwendung gehören sollen. Sollen Studenten eine Applikation für die Wall erstellen können, müssen klare Regeln für den Ablauf der Erstellung, der Abnahme und den Inhalt der Anwendung aufgestellt werden.

Soll die Poster-Applikation weiter betrieben werden, so sind zwei Herausforderungen zu meistern. Mit der in der Machbarkeitsstudie erarbeiteten Hardware-Lösung sind nicht alle Poster lesbar. Es muss daher eine Möglichkeit erarbeitet werden, diese Poster lesbar zu machen. Dies könnte einerseits über eine Zoom-Möglichkeit gelöst werden oder über einen vordefinierten Pfad, über den der Benutzer durch das vergrößert angezeigte Poster geführt wird.

Die Bachelorposter sind möglicherweise in ihrer statischen Form nicht attraktiv genug. Interaktive Elemente auf einem Plakat könnten diese Attraktivität wesentlich steigern.

Mit diesen Erweiterungen wird der Prototyp zum fertigen Produkt.

# V. Technischer Bericht

<b>V.1 Einleitung .....</b>	<b>13</b>
V.1.1 Änderungsgeschichte .....	13
V.1.2 Einleitung .....	14
<b>V.2 Projektmanagement .....</b>	<b>15</b>
V.2.1 Änderungsgeschichte .....	15
V.2.2 Projektplan .....	16
V.2.3 Projektorganisation .....	18
V.2.4 Risiken .....	18
V.2.5 Vorgehensmodelle .....	19
<b>V.3 Vorstudie .....</b>	<b>21</b>
V.3.1 Änderungsgeschichte .....	23
V.3.2 Vision .....	24
V.3.3 Gebäude der HSR .....	25
V.3.4 Initiale Stakeholderanalyse .....	26
V.3.5 Konkurrenzanalyse .....	28
V.3.6 Passantenanalyse .....	35
V.3.7 Interaktionsbereich des Kinect Sensors .....	39
V.3.8 Befragung .....	40
V.3.9 Rollen & Personas .....	45
V.3.10 Sofortiges Erfolgserlebnis .....	50
V.3.11 Motivation zur wiederholten Nutzung der Videowall .....	50
V.3.12 Microsoft Imagine Cup .....	50
<b>V.4 Anforderungen .....</b>	<b>51</b>
V.4.1 Änderungsgeschichte .....	51
V.4.2 Tools .....	52
V.4.3 Funktionale Anforderungen .....	53
V.4.4 Nicht-funktionale Anforderungen .....	60
V.4.5 Design Constraints .....	62
V.4.6 Zugänglichkeit (Accessibility) .....	62
<b>V.5 Domain Analyse .....</b>	<b>63</b>
V.5.1 Änderungsgeschichte .....	64
V.5.2 Systemübersicht .....	65
V.5.3 Daten .....	67
V.5.4 Graphical User Interface (GUI) .....	71
<b>V.6 Entwurf .....</b>	<b>88</b>
V.6.1 Änderungsgeschichte .....	89
V.6.2 Design Entscheide .....	90
V.6.3 Betriebskonzept der Applikation .....	93
V.6.4 Lebenszyklus der Applikation .....	94

V.6.5	Architektur .....	97
V.6.6	Plug-in Framework .....	101
V.6.7	Design des Demomodus .....	108
V.6.8	Interaktion durch Handtracking .....	111
<b>V.7</b>	<b>HSR Videowall Evaluation .....</b>	<b>113</b>
V.7.1	Änderungsgeschichte .....	114
V.7.2	Software Evaluation.....	115
V.7.3	Hardware Evaluation.....	115
V.7.4	Evaluation Mitsubishi Display Wall .....	131
V.7.5	Beschaffungsanalyse .....	132
V.7.6	Lesbarkeit der Poster.....	133
<b>V.8</b>	<b>Realisierung &amp; Test .....</b>	<b>135</b>
V.8.1	Änderungsgeschichte .....	138
V.8.2	Usability Tests .....	139
V.8.3	Unit Tests .....	143
V.8.4	Systemtests .....	145
V.8.5	Stabilitätstest.....	154
V.8.6	Dokumentation der Realisierung .....	156
V.8.7	Beschreibung der Applikationen .....	172
V.8.8	Code Reviews .....	175
<b>V.9</b>	<b>Betriebskonzept .....</b>	<b>182</b>
V.9.1	Änderungsgeschichte .....	183
V.9.2	Betrieb und Administration der Videowall .....	184
V.9.3	Installationsdokumentation .....	188
<b>V.10</b>	<b>Ausblick.....</b>	<b>193</b>
V.10.1	Änderungsgeschichte .....	193
V.10.2	Ausblick.....	194
V.10.3	Weiterentwicklung.....	194
V.10.4	Zeitplan .....	196

## V.1 Einleitung

V.1.1	Änderungsgeschichte .....	13
V.1.2	Einleitung .....	14

---

### V.1.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
04.06.2012	1.0	Erste Version des Dokuments	CH
05.06.2012	1.1	Review	DT
09.06.2012	1.2	Review und Korrekturen	LE
14.06.2012	1.3	Review und Korrekturen	DT

---

## V.1.2 Einleitung

Die Bachelorarbeit ist eine Machbarkeitsstudie mit Prototyp. In der Machbarkeitsstudie werden vor allem Risikothemen abgeklärt, welche technischer und benutzerspezifischer Natur sein können.

Der technische Bericht beginnt mit dem Projektmanagement (siehe V.2 Projektmanagement), in welchem der Projektplan dargelegt und die Projektorganisation umschrieben wird. Des Weiteren werden die Risiken und das gewählte Vorgehensmodell aufgezeigt.

Das nächste Kapitel enthält die Vorstudie (siehe V.3 Vorstudie). Diese zeigt die Vision auf und enthält die Stakeholder- und die Konkurrenzanalyse. Danach widmet sich die Vorstudie der Passantenanalyse, dem Interaktionsbereich des Kinect Sensors und der Benutzerbefragung. Rollen und Personas mit den dazugehörigen Szenarien werden daraufhin vorgestellt. Abschliessend werden noch das sofortige Erfolgserlebnis und die Motivation zur wiederholten Nutzung der Videowall umschrieben und auf den Microsoft Imagine Cup eingegangen.

Das Kapitel Anforderungen (siehe V.4 Anforderungen) beschreibt zu Beginn die im Projekt verwendeten Tools. Danach sind die funktionalen und nichtfunktionalen Anforderungen festgehalten. Weiter beschreibt es die Design Constraints und Accessibility im Zusammenhang mit der Nutzung von Kinect und der Monitorwand.

Im Kapitel Domain Analyse (siehe V.5 Domain Analyse) gibt einen Überblick über das gewünschte System. Weiter werden die Daten und das zugehörige Domain Model beschrieben. Auch wird der Prozess zur Entstehung der Poster erklärt. Danach wird auf das Graphical User Interface (GUI) eingegangen. Für das GUI wurden Ideen gesammelt, welche in den Unterkapiteln vorgestellt werden. Die Screen Map zeigt auf, wie die Elemente des Domain Models grafisch eingebunden werden. Im Anschluss daran werden die Design Entscheide für das GUI und das externe Design vorgestellt. Schliesslich werden die Kinect Guidelines beschrieben und es ist festgehalten, wie diese in der Arbeit umgesetzt wurden.

Das Kapitel Entwurf (siehe V.6 Entwurf) widmet sich einleitend den Design Entscheiden. Weiter sind das Betriebskonzept und der Lebenszyklus der Applikation festgehalten. Daraufhin wird die gewählte Architektur umschrieben, die verwendeten Patterns vorgestellt und eine Erklärung zu Prozessen und Threads gegeben. Anschliessend wird die Funktion des Plug-in Frameworks aufgezeigt. Zum Schluss widmet sich das Kapitel dem Design des Demomodus und der Interaktion durch das Handtracking.

Im Kapitel HSR Videowall Evaluation (siehe V.7 HSR Videowall Evaluation) wird die Software Evaluation und die Hardware Evaluation für die verschiedenen Videowall-Komponenten, wie die Monitore und die Grafikkarten, beschrieben. Auch die für die Arbeit aufgebaute Testhardware und die damit durchgeföhrten Tests werden erläutert. Daraufhin wird die Mitsubishi Display Wall vorgestellt und es folgt eine Beschaffungsanalyse. Zum Schluss wird noch auf die Lesbarkeit der Poster auf der Videowall eingegangen.

Im nachfolgenden Kapitel Realisierung und Test (siehe V.8 Realisierung & Test) sind die durchgeföhrten Usability Tests und deren Auswertungen dokumentiert. Danach werden die Unit Tests, System Tests und ein Stabilitätstest aufgeführt. Weiter ist die Realisierung dokumentiert, es wird auf die Übereinstimmung mit der Architektur eingegangen und die Code Statistik und Qualität festgehalten. Im Verlauf des Projektes wurden neben der Applikation auch kleine Prototypen erstellt, welche im Abschnitt Beschreibung der Applikationen erläutert werden. Schliesslich sind die Code Reviews dokumentiert.

Das vorletzte Kapitel Betriebskonzept (siehe V.9 Betriebskonzept) hält Betrieb und die Administration der Videowall und die Installationsdokumentation fest.

Das letzte Kapitel Ausblick (siehe V.10 Ausblick) widmet sich dem Ausblick für die Weiterentwicklung der Videowall.

## V.2 Projektmanagement

<b>V.2.1 Änderungsgeschichte .....</b>	<b>15</b>
<b>V.2.2 Projektplan .....</b>	<b>16</b>
V.2.2.1 Releases .....	16
V.2.2.2 Milestones / Sprints.....	16
V.2.2.3 Zeitplan und Zeiterfassung.....	17
V.2.2.4 Aufwandschätzung.....	17
<b>V.2.3 Projektorganisation .....</b>	<b>18</b>
V.2.3.1 Team und Verantwortlichkeiten.....	18
V.2.3.1.1 Lukas Elmer (Abk. LE).....	18
V.2.3.1.2 Christina Heidt (Abk. CH) .....	18
V.2.3.1.3 Delia Treichler (Abk. DT) .....	18
V.2.3.2 Aufgabenteilung Programmierung .....	18
<b>V.2.4 Risiken .....</b>	<b>18</b>
<b>V.2.5 Vorgehensmodelle .....</b>	<b>19</b>
V.2.5.1 Scrum.....	19
V.2.5.2 RUP .....	19
V.2.5.3 UCD .....	20

---

### V.2.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
23.02.2012	1.0	Erste Version des Dokuments	DT
24.02.2012	1.1	Review	CH
13.04.2012	1.2	Änderung Daten DailyScrum	DT
09.06.2012	1.3	Review und Korrekturen	LE
12.06.2012	1.4	Review	DT
14.06.2012	1.5	Review	LE

## V.2.2 Projektplan

Das Management des Projektes Videowall wird im Redmine<sup>1</sup> durchgeführt.

### V.2.2.1 Releases

Das Projekt nach dem Vorgehensmodell Scrum (siehe Unterkapitel V.2.5 Vorgehensmodelle) durchgeführt. Es ist während des Projekts immer am Ende eines Sprints ein voll funktionsfähiger Prototyp verfügbar. Was dieser Prototyp kann, ist der Tabelle 2 - Die geplanten Sprints zu entnehmen.

Folgender Release ist vorgesehen:

Version	Typ	Beschreibung	Datum
1.0	Finalversion	Fertiger Prototyp, Bugs gefixt oder dokumentiert, Code Reviews durchgeführt, Dokumentation abgeschlossen	15.06.2012

Tabelle 1 - Releases

### V.2.2.2 Milestones / Sprints

Ein Milestone ereignet sich jeweils am Ende eines Sprints. Die Sprints, wie auch die Milestones, sind im Redmine dokumentiert.

Auf der Website <https://redmine.elmermx.ch/> steht der detaillierte Projektplan zur Verfügung. Die detaillierte Planung der jeweiligen Sprints erfolgt schrittweise nach den Vorgehensmodellen Scrum, RUP und UCD (siehe Unterkapitel V.2.5 Vorgehensmodelle).

Sprint	Datum	Grobe Beschreibung
1	27.02.2012	Versionskontrolle eingerichtet, Technologien studiert, einfache kleine Experimente entwickelt
2	05.03.2012	Microsoft SDK getestet, Benutzerbefragung und -beobachtung durchgeführt, Mini-Prototyp mit Tracking von Personen erstellt, Testsetup evaluiert
3	12.03.2012	Vision erstellt, Bachelorposter organisiert und konvertiert, Microsoft Imagine Cup Projekt-Plan erstellt, Entscheidung für Kinect Framework getroffen
4	19.03.2012	Prototyp getestet (formativer empirischer Test), Personas und Szenarien erarbeitet, Kinect Record/Replay-Möglichkeit gefunden und entwickelt, Skeleton Aufnahmen gemacht, Backlog mit User Stories erstellt
5	26.03.2012	Prototyp getestet (formativer empirischer Test), Demonstration Prototyp vorbereitet, WPF-Applikation implementiert und auf Testhardware getestet, Grobarchitektur erarbeitet, implementiert und Review mit Silvan Gehrig
6	02.04.2012	Prototyp getestet (formativer empirischer Test), tiefere Auflösungen auf der Videowall getestet, Handtracking-Prototyp erstellt, Prototyp mit Mittagsmenu erstellt
7	16.04.2012	Architekturprototyp: Poster-Applikation implementiert, mit Kinect „klicken“ implementiert, Webarchitektur von Microsoft studiert
8	23.04.2012	Usability Test für Prototyp organisiert, Prototyp auf Test-Setup getestet, DirectX auf Testhardware getestet
9	30.04.2012	Ideen für Teaser gesammelt, Applikation mit linker Hand bedienbar gemacht, WPF-Applikation mit Video getestet, Microsoft Managed Extensibility Framework (MEF) studiert
10	07.05.2012	Plug-in Framework entworfen und implementiert, Mitsubishi Display Wall angesehen, Code Review, Anpassungen gemäß Code Review durchgeführt
11	14.05.2012	Plug-ins aus Hauptapplikation extrahiert, Demomodus entworfen und implementiert

<sup>1</sup> <https://redmine.elmermx.ch/>

Sprint	Datum	Grobe Beschreibung
12	22.05.2012	Plug-in Schnittstelle definiert und dokumentiert, Accessibility dokumentiert, Demomodus weiterentwickelt, 2x4 Videowall-Setup getestet, Mittagsmenu automatisch aktualisiert, externes Design definiert und dokumentiert
13	28.05.2012	Usability Test durchgeführt mit Prototyp, Dokumentation überarbeitet, Unit Tests erstellt, erste Version des Bachelorposters erarbeitet
14	04.06.2012	Abstract und Management Summary geschrieben, Screen Map erstellt, externes Design validiert, Anpassungen gemäss Usability Test und Code Reviews durchgeführt, Code dokumentiert
15	11.06.2012	Video/Wiki Seite erstellt, Dokumentation erweitert, Software-Architektur beschrieben, Stabilitätstest durchgeführt, Programmierung abgeschlossen
16	15.06.2012	Dokumentation abgeschlossen, CD gebrannt, Poster gedruckt, Test-Setup abgebaut, HSR-Forum-Stand vorbereitet, betreut und abgebaut

Tabelle 2 - Die geplanten Sprints

### V.2.2.3 Zeitplan und Zeiterfassung

Die einzelnen Arbeitspakete (Tickets) sind den jeweiligen Sprints zugeordnet. Das Projekt ist in 16 Sprints unterteilt. Das Ende eines Sprints entspricht jeweils einem Milestone.

Die komplette Zeitplanung und die Zeiterfassung werden auf dem Redmine-Server durchgeführt. Für jedes Arbeitspaket wird der Zeitaufwand geschätzt (siehe Kapitel V.2.2.4 Aufwandschätzung) und ein Ticket erstellt. Diese Tickets werden dann den jeweiligen Sprints zugeordnet. Wurde an einem Ticket gearbeitet, wird die dafür aufgewendete Zeit auf das Ticket gebucht. Die Erfassung der Zeit für die jeweiligen bearbeiteten Tickets wird jeweils sofort nach Abschluss der Arbeiten vorgenommen. Somit ist die Zeiterfassung stets aktuell.

Die Reportfunktion bietet einen Überblick über den geplanten und den tatsächlichen Zeitaufwand. Zudem ist es möglich, den Arbeitsaufwand mittels einer cvs-Datei zu exportieren und z.B. in Excel anschaulich darzustellen.

### V.2.2.4 Aufwandschätzung

Die Aufwandschätzung ergibt sich durch den geschätzten Aufwand pro Ticket im Redmine.

Der Zeitaufwand, welcher das Abarbeiten eines Tickets benötigt, wird mittels Planning Poker geschätzt. Dazu überlegt sich jedes Teammitglied alleine, wie viel Zeit für die Abarbeitung eines bestimmten Tickets benötigt wird. Laufen die Schätzungen für ein Ticket bei der Besprechung auseinander, kann eine Diskussion versteckte Anforderungen aufdecken. Schliesslich einigt sich das Team auf eine Schätzung.

---

## V.2.3 Projektorganisation

---

### V.2.3.1 Team und Verantwortlichkeiten

#### V.2.3.1.1 Lukas Elmer (Abk. LE)

Kenntnissein: Ruby on Rails, PHP, Python / Django, Typo3, Wordpress, Java, XHTML, JavaScript, C++, C#, Ubuntu Server  
Rolle/Verantwortlichkeiten: Architektur, Serverunterhalt von Redmine, Konfigurationsmanagement  
Mailadresse: lelmer@hsr.ch / lukas.elmer@gmail.com  
Skype Adresse: lukas.elmer



#### V.2.3.1.2 Christina Heidt (Abk. CH)

Kenntnissein: Java, HTML/CSS, C++, C#, Photoshop  
Rolle/Verantwortlichkeiten: Grafisches Design, Risikomanagement, Anforderungen, Sitzungsprotokollierung  
Mailadresse: cheidt@hsr.ch  
Skype Adresse: christina\_heidt



#### V.2.3.1.3 Delia Treichler (Abk. DT)

Kenntnissein: Java, HTML/CSS, C++, C#  
Rolle/Verantwortlichkeiten: Überwachung und Erstellung Projektplan, Teamsitzungen, Usability-Tests  
Mailadresse: dtreichler@hsr.ch  
Skype Adresse: de-lia



---

### V.2.3.2 Aufgabenteilung Programmierung

Es ist geplant, jedes wichtige Feature der zu implementierenden Applikation zuerst als kleine Anwendung zu programmieren, welche dann nur eine bestimmte Funktionalität beinhaltet. An der Entwicklung dieser Mini-Applikationen (für weitere Informationen siehe Kapitel V.8.7.3 Mini-Applikationen), deren Funktionalität nach der Fertigstellung in die Hauptapplikation übernommen wird, sind alle Teammitglieder beteiligt.

Gegen Ende des Projektes soll nochmals Refactoring betrieben werden, damit der Code für die Assistenten, die ihn übernehmen, möglichst sauber strukturiert ist. Dadurch werden die originalen Codeteile der Mini-Applikationen nicht mehr in ihrer alten Form in der Hauptapplikation vorhanden sein. Es wurde entschieden, dass das Refactoring aus einer Hand gemacht werden wird, damit die Applikation in einem Fluss und korrekt strukturiert ist.

Die zuvor erstellten Mini-Applikationen sind genauso bedeutungsvoll wie der Code der Hauptapplikation.

---

## V.2.4 Risiken

Das Risikomanagement befindet sich im Anhang (siehe VIII Anhang).

## V.2.5 Vorgehensmodelle

Um alle Kriterien an diese Bachelorarbeiterfüllen zu können, wurden Punkte aus den folgenden drei Vorgehensmodellen angewendet: Scrum, RUP und UCD. Was genau aus diesen Modellen angewendet wurde und wie wird nachfolgend beschrieben.

### V.2.5.1 Scrum

Für das Projekt Videowall wird hauptsächlich der Ansatz von Scrum verfolgt, weil dieses Vorgehensmodell auf die Eigenorganisation der einzelnen Teammitglieder ausgerichtet und äußerst produktiv ist, da Overhead so weit wie möglich reduziert wird.

Die nachfolgende Tabelle zeigt auf, welche Elemente (Rollen, Meetings und Artefakte) von Scrum wie gehandhabt werden.

Scrum-Element	Umsetzung
<b>Rollen</b>	
Product Owner, Scrum Master	Diese Rollen können personalbedingt nicht besetzt werden. Die Aufgaben des Product Owners und des Scrum Masters werden vom Entwicklungsteam übernommen.
<b>Meetings</b>	
Sprint Planung	Die Planung des Sprints wird zu Beginn des jeweiligen Sprints durchgeführt. Termin: Montag, 13.00-13.30: Sprintplanung des nächsten Sprints
Daily Scrum	Das Meeting wird zu folgenden Zeiten durchgeführt: Dienstag, 9.50-10.10: Daily Scrum Donnerstag, 10.20-10.40: Daily Scrum Weitere Daily Scrum Meetings nach Bedarf.
Sprint Review	Das Review Meeting findet jeweils am vorletzten Tag (Freitag) des aktuellen Sprints statt. (Freitag, 10.10-10.30: Vorbereitung, Vorbesprechung Meeting)
<b>Artefakte</b>	
Product Backlog	Die Anforderungen an das Produkt sind als Tickets im Redmine erfasst. Die Schätzung des Aufwands wird für jede Anforderung auf dem entsprechenden Ticket nach dem Modell Planning Poker vorgenommen.
Sprint Backlog	Die für einen Sprint geplanten Aufgaben existieren im Redmine als Ticket, welche dem jeweiligen Sprint zugeordnet sind. Der Restaufwand, der für eine einzelne Aufgabe noch benötigt wird, ist über die Differenz der geschätzten und bisher gebuchten Zeit ersichtlich. Gleichzeitig dient das Redmine als Ersatz für das Taskboard.
Burndown-Charts	Das Gantt-Diagramm, welches sich im Redmine anzeigen lässt, erübrigte einen Burndown-Chart.
Impediment Backlog	Im Redmine Wiki besteht eine Seite zur Eintragung von Hindernissen und Problemen des Projektes.
Releaseplan	Die Tickets im Redmine bieten einen Überblick über den Zeitplan und die Termine/Meetings. Die erwartete Anzahl Sprints ist unter Roadmap ersichtlich. Für die Behandlung von Risiken siehe Unterkapitel V.2.4 Risiken.

Tabelle 3 - Scrum Elemente

### V.2.5.2 RUP

Da für die Bachelorarbeit viele Dokumente erarbeitet werden müssen, ist die Strukturierung der Dokumente an RUP angelehnt (Vorstudie, Anforderungen, Domain Analyse, Entwurf, Realisierung und Test).

Zusätzlich wird zu Beginn der Arbeit ein einfacher Architekturprototyp erstellt, der alle Layer und Tiers abdeckt, um die größten technischen und architektonischen Risiken abzudecken.

---

### V.2.5.3 UCD

Eine hohe Usability ist für die Videowall aus verschiedenen Gründen besonders wichtig. Einerseits muss die Videowall ohne die Hilfe eines Benutzerhandbuchs, also einfach, zu bedienen sein. Andererseits soll die Videowall die HSR präsentieren und einen möglichst positiven Eindruck bei den Besuchern hinterlassen.

Um eine hohe Usability zu erreichen werden für die Videowall auch Aspekte des User Centered Design (UCD) beachtet. Dazu zählen:

- Analyse des Nutzungskontextes durch Benutzerbeobachtung und Erstellen von Personas und Szenarien
- Benutzerumfragen durchführen und auswerten
- Anforderungen an die Applikation gemäss den Personas und Szenarien definieren
- Aufstellen von Thesen (z.B. „Meine Hand ist die Maus“) und diese Thesen durch Usability Tests validieren
- Implementationen oder Arbeiten mithilfe von Usability Tests validieren

Durch das Vorgehen nach UCD ist es möglich, Benutzer früh miteinzubeziehen und die Anforderungen so zu definieren, dass die Nutzer ihre Ziele auf möglichst einfache Weise erreichen können. Es ist auch möglich, durch Usability Tests die Qualität einer Lösung zu beurteilen und zu verbessern.

## V.3 Vorstudie

<b>V.3.1 Änderungsgeschichte .....</b>	<b>23</b>
<b>V.3.2 Vision .....</b>	<b>24</b>
<b>V.3.3 Gebäude der HSR .....</b>	<b>25</b>
<b>V.3.4 Initiale Stakeholderanalyse.....</b>	<b>26</b>
<b>V.3.5 Konkurrenzanalyse .....</b>	<b>28</b>
V.3.5.1 Präsentationsmöglichkeiten für Poster.....	28
V.3.5.2 Bestehende Videowalls.....	28
V.3.5.3 Der Videowall ähnliche Produkte .....	30
V.3.5.3.1 Interaktion mit Videowall.....	30
V.3.5.3.1.1 HoloWall.....	30
V.3.5.3.1.2 It's Mine, Don't Touch!: Interactions at a Large Multi-Touch Display in a City Centre .....	30
V.3.5.3.1.3 Extending Touch: Towards Interaction with Large-Scale Surfaces .....	31
V.3.5.3.2 Interaktion ohne Videowall .....	31
V.3.5.3.2.1 Microsoft Kinect Sensor and Its Effect.....	32
V.3.5.3.2.2 Leap Motion .....	32
V.3.5.3.2.3 Panasonic D-Imager .....	32
V.3.5.3.2.4 Dance Dance Revolution .....	33
V.3.5.3.3 Fazit.....	34
<b>V.3.6 Passantenanalyse .....</b>	<b>35</b>
V.3.6.1 Abstandszonen .....	36
V.3.6.2 Gruppengrößen .....	38
<b>V.3.7 Interaktionsbereich des Kinect Sensors .....</b>	<b>39</b>
<b>V.3.8 Befragung.....</b>	<b>40</b>
V.3.8.1 Fragebogen .....	41
V.3.8.2 Auswertung .....	42
V.3.8.3 Fazit.....	44
<b>V.3.9 Rollen &amp; Personas .....</b>	<b>45</b>
V.3.9.1 Rollen.....	45
V.3.9.2 Personas.....	45
V.3.9.2.1 Persona Peter Posterleser .....	46
V.3.9.2.1.1 Ist-Szenario-1.....	46
V.3.9.2.1.2 Soll-Szenario-1.....	47
V.3.9.2.1.3 Ist-Szenario-2.....	47
V.3.9.2.1.4 Soll-Szenario-2.....	47
V.3.9.2.2 Persona Noemi Nichtinteressiert.....	47
V.3.9.2.2.1 Ist-Szenario 1.....	48
V.3.9.2.2.2 Soll-Szenario 1.....	48
V.3.9.2.2.3 Ist-Szenario 2.....	48
V.3.9.2.2.4 Soll-Szenario 2.....	48

V.3.9.2.3	Persona Erich Eventbesucher .....	49
V.3.9.2.3.1	Ist-Szenario-1.....	49
V.3.9.2.3.2	Soll-Szenario-1.....	49
<b>V.3.10</b>	<b>Sofortiges Erfolgserlebnis.....</b>	<b>50</b>
<b>V.3.11</b>	<b>Motivation zur wiederholten Nutzung der Videowall .....</b>	<b>50</b>
<b>V.3.12</b>	<b>Microsoft Imagine Cup .....</b>	<b>50</b>

### V.3.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
24.02.2012	1.0	Erste Version des Dokuments	CH
28.02.2012	1.1	Fragebogen, Passantenanalyse	CH
02.03.2012	1.2	Passantenanalyse	CH
08.03.2012	1.3	Vision	CH
08.03.2012	1.4	Befragung, Review Testsetup, Review Passantenanalyse	DT
09.03.2012	1.5	Vision, Hardware-Setup, Review Befragung	CH
13.03.2012	1.6	Review Vision und Hardware-Setup	DT
13.03.2012	1.7	Personas und Szenarien Peter Posterleser und Erich Eventbesucher	CH
13.03.2012	1.8	Persona Noemi Nichtinteressiert, Szenarien Noemi Nichtinteressiert	DT
14.03.2012	1.9	Review	CH
14.03.2012	1.10	Review Personas und Szenarien	DT
19.03.2012	1.11	Ergänzung der Einführung zu den Personas mit Rollen	DT
19.03.2012	1.12	Kapitel 4: Interaktionsbereich des Kinects Sensors	DT
20.03.2012	1.13	Review Interaktionsbereich, Einfügen Abbildung	CH
23.03.2012	1.14	Korrekturen aus dem Review M. Stolze	CH
26.03.2012	1.15	Review, Nutzwertanalyse	DT
27.03.2012	1.16	Review Nutzwertanalyse, Testhardware dokumentiert	CH
03.04.2012	1.17	Review, Anpassungen Kapitel I.6 Befragung gemäss Sitzung vom 2.4.12	DT
13.04.2012	1.18	Review, Testhardware in anderes Dokument verschoben	CH
18.04.2012	1.19	Sofortiges Erfolgserlebnis	DT
22.04.2012	1.20	Anpassung Vision	CH
23.04.2012	1.21	Konkurrenzanalyse und ähnliche Arbeiten	LE
24.04.2012	1.21	Review Vision	DT
25.04.2012	1.22	Ergänzung Befragung: Varianz	DT
04.05.2012	1.23	Korrekturen von Herrn Heinzmann	CH
06.05.2012	1.24	Review Korrekturen	DT
24.05.2012	1.25	Stakeholderanalyse	DT
24.05.2012	1.26	Review Korrekturen Markus Stolze	DT
28.05.2012	1.28	Konkurrenzanalyse und ähnliche Arbeiten	LE
29.05.2012	1.29	Review Stakeholderanalyse	CH
29.05.2012	1.30	Review Konkurrenzanalyse	DT
04.06.2012	1.31	Ergänzung Kapitel Wiederholte Nutzung	DT
08.06.2012	1.32	Konkurrenzanalyse	DT
09.06.2012	1.33	Review und Korrekturen	LE
10.06.2012	1.32	Konkurrenzanalyse	DT
11.06.2012	1.33	Review Konkurrenzanalyse	CH
12.06.2012	1.34	Review	DT
14.06.2012	1.35	Imagine Cup	CH
14.06.2012	1.36	Review und Korrekturen	DT

---

### V.3.2 Vision

Neue Technologien führen zu neuen Präsentationsmöglichkeiten. Durch den Einsatz dieser Technologien werden Innovation und Wissen über den neusten Stand der Technik demonstriert. Beide Faktoren spielen eine wichtige Rolle für eine technische Hochschule. Wer würde sein Studium an einer Schule beginnen, welche über keine Beamer in den Hörsälen sondern lediglich Hellraumprojektoren verfügt? Oder an einer, an welcher alle Übungsräume mit Röhrenbildschirmen ausgestattet sind?

Eine moderne Hochschule soll zumeist bei den Besuchern einen positiven Eindruck hinterlassen, zum anderen aber auch bei den Studenten und Angestellten. Eine Möglichkeit, sich als moderne Hochschule zu profilieren, ist die Nutzung von innovativen Präsentationstechniken. Durch ihre ständige Anwesenheit stellen Studenten und Angestellte die Hauptzielgruppe für Präsentationen dar. Sinnvolle Präsentationsinhalte wären einerseits Informationen aus den verschiedenen Studiengängen. Andererseits sind auch Inhalte denkbar, welche den Alltag vereinfachen oder erheitern.

Um die Nutzung innovativer Präsentationsmöglichkeiten zu ermöglichen, plant die HSR eine interaktive Videowall im Eingangsbereich des Verwaltungsgebäudes (Gebäude 4, siehe V.3.3 Gebäude der HSR). Dieses Gebäude ist ein attraktiver Standort, da sich dort die Mensa, der Empfang und die Aula befinden. Um zu den erwähnten Räumen zu gelangen, muss der Eingangsbereich, welcher ein relativ breiter Gang ist, passiert werden. Dieser Bereich stellt den idealen Ort dar, um die Videowall aufzustellen.

Die imposante Größe der Videowall soll diese für die Passanten unübersehbar machen und deren Neugier wecken. Der Nutzer kann mittels Gesten mit der Videowall interagieren, dadurch entfallen Eingabegeräte wie Tastatur oder Maus komplett. Zu Beginn werden auf der Wall die Bachelorposter präsentiert. Diese Anwendung wurde durch den Auftraggeber als Startpunkt für die Nutzung der Videowall definiert. Die Nachforschungen im Rahmen dieser Arbeit haben mögliche Problempunkte dieser Anwendung zutage gefördert. So muss sich im Betrieb zeigen, ob mit diesen Inhalten allein die gewünschte wiederholte Nutzung der Videowall erreicht werden kann. Auch lässt sich mittlerweile abschätzen, dass die ungezoomte Darstellung auf der Videowall für manche Bachelorposter nicht genügend hochauflösend ist. Die Analyse zeigt aber auch, dass die leichte Verfügbarkeit dieser attraktiven Inhalte über alle Studiengänge dafür spricht, dass die Darstellung dieser Poster eine sinnvolle Pilotanwendung darstellt. Mittelfristig kann dies dazu führen, dass Studenten durch die Aussicht auf eine Publikation auf der Videowall motiviert werden, ihre Arbeiten auf interaktiver Art (beispielsweise als Video) zu dokumentieren, wodurch sich die Attraktivität der Inhalte für die Nutzer erhöht.

Welche Inhalte könnten nun zusätzlich auf der Wall angeboten werden? Wie wäre es nun beispielsweise mit Informationen, die man täglich benötigt? Dies könnten Informationen zu Veranstaltungen, das Mittagsmenü der Mensa, die Wetterlage oder sogar ein Spiel sein. Stellen Sie sich vor, Sie haben den ganzen Tag mit dem Besuchen von Vorlesungen verbracht. Wäre es jetzt nicht eine tolle Abwechslung, ein paar Minispiele auf der Videowall zu spielen? Solche und viele andere Anwendungen sind für die Wall denkbar.

Das System der Videowall bietet die Möglichkeit, Studenten der Informatik eine neue Plattform zur Verfügung zu stellen. Durch die Nutzung der Plattform könnten Projektarbeiten einem grossen Publikum präsentiert werden.

Eine zusätzliche Anforderung ist, dass die Inhalte der Wall verwaltet werden können. Daher ist es wichtig, dass das System gut wartbar und über eine Administrationsoberfläche einfach bedienbar ist.

Das Projekt HSR Videowall sollte alle diese Themen abdecken.

### V.3.3 Gebäude der HSR

Es ist geplant, die Videowall im Eingangsbereich des Verwaltungsgebäudes (Gebäude 4) aufzustellen. Um eine bessere Übersicht über die verschiedenen Gebäude der HSR zu erhalten, wurden diese hier aufgelistet.



- 1 Schulgebäude Mitte
- 2 Laborgebäude
- 3 Hörsaalgebäude
- 4 Verwaltungsgebäude / Aula / Hochschuldienste
- 5 Foyergebäude
- 6 Schulgebäude See
- 7 Pavillons

Abbildung 7 - Gebäude der HSR, Bildquelle: [www.hsr.ch](http://www.hsr.ch)

### V.3.4 Initiale Stakeholderanalyse

Die Stakeholderanalyse dient der Ermittlung der Interessenträger dieser Arbeit. Weiter hilft die Analyse, den Prototypen sinnvoll, mit den wichtigen Interessenten im Fokus, umsetzen zu können.

In den nachfolgenden zwei Tabellen sind alle Stakeholder, unterteilt nach Projekt- und Produkt-Stakeholder, aufgelistet. Zudem ist festgehalten, worin das Interesse der Stakeholder am Gelingen des Projektes besteht. Die Unterteilung in die Kategorien „sehr wichtig“, „wichtig“ und „weniger wichtig“ zeigt, welche Stakeholder wie wichtig für den Erfolg des Projektes sind.

Kategorie	Projekt-Stakeholder	Interesse(n) des Stakeholders
<b>sehr wichtig</b>		
	Markus Stolze (Auftraggeber)	- Umsetzung und Erfolg des Projektes - technische Machbarkeit
	Entscheidungsgremium	- neue Technologie/Fortschritt/Wissen an der HSR demonstrieren - technisch innovativen Eindruck der HSR bei Besuchern (z.B. von Institutionen) wie auch Mitarbeitern und Studenten hinterlassen
	Institut für Software (IFS)	- Nutzung der neuen Technologie für Arbeiten/Projekte - Weiterentwicklung des bestehenden Projektes

Tabelle 4 - Initiale Stakeholderanalyse, Projekt-Stakeholder

Die Einteilung der Projekt-Stakeholder in die verschiedenen Kategorien in Tabelle 4 - Initiale Stakeholderanalyse, Projekt-Stakeholder lässt sich wie folgt begründen:

- Einstufung als sehr wichtig:
  - Stakeholder Markus Stolze: Er ist der Auftraggeber.
  - Stakeholder Entscheidungsgremium: Von dessen Entscheidung hängt die Realisierung der Videowall ab.
  - Stakeholder Institut für Software: Das Team wird durch die Assistenten Silvan Gehrig und Michael Gfeller des IFS in der Durchführung des technischen Teils des Projektes unterstützt. Die Arbeit bietet dem Institut für Software Weiterentwicklungsmöglichkeiten.

Kategorie	Produkt-Stakeholder	Interesse(n) des Stakeholders
<b>sehr wichtig</b>		
	Studenten	- Präsentation der Arbeit ist modern und interaktiv - Wissensaustausch zwischen Studiengängen durch Präsentation von Arbeiten aller Studiengänge - Informationen über die Themen der Projekte an den verschiedenen Instituten der HSR - Eigener Beitrag für die Videowall in Form einer Applikation, welche eingebunden werden kann. - Unterhaltung, Interaktivität, Spiel und Spass
	Eventbesucher (fachliche Konferenzen)	- Eventinformationen zentral und auf einen Blick verfügbar - Unterhaltung, Interaktivität, Spiel und Spass
<b>wichtig</b>		
	Studiengangleiter	- neue Technologie/Fortschritt/Wissen aus den verschiedenen Abteilungen an der HSR demonstrieren
	Institute	- Nutzung der neuen Technologie für Arbeiten/Projekte
	Eventbesucher (Bachelorfeiern o.Ä.)	- Eventinformationen zentral und auf einen Blick verfügbar - Betrachten aller Bachelorposter an einem zentralen Ort - Unterhaltung, Interaktivität, Spiel und Spass
<b>weniger wichtig</b>		
	Allgemeine Besucher (z.B. auch Informatik-Schulklassen)	- Unterhaltung, Interaktivität, Spiel und Spass - Beispiel für das, was an der HSR gelehrt wird (Abteilung Informatik)

Tabelle 5 - Initiale Stakeholderanalyse, Produkt-Stakeholder

Die Einteilung der Produkt-Stakeholder in die verschiedenen Kategorien in Tabelle 5 - Initiale Stakeholderanalyse, Produkt-Stakeholder lässt sich wie folgt begründen:

- Einstufung als sehr wichtig:
  - Stakeholder Studenten: Sie stellen eine der zwei Hauptzielpersonengruppen für die Arbeit dar.
  - Stakeholder Eventbesucher: Die Eventbesucher sind eine der zwei Hauptzielpersonengruppen für die Arbeit.
- Einstufung als wichtig:
  - Stakeholder Studiengangleiter: Die Arbeit bietet ihnen die Möglichkeit, die Arbeiten aus ihren Studiengängen attraktiv zu präsentieren.
  - Stakeholder Institute: Die Arbeit bietet eine neue Plattform und somit eine neue Präsentationsmöglichkeit.
- Einstufung als weniger wichtig:
  - Stakeholder Allgemeine Besucher: Die Arbeit bietet die Plattform, auf welcher Informationen an die Besucher überbracht und ihr Interesse geweckt werden kann.

### V.3.5 Konkurrenzanalyse

In der Konkurrenzanalyse werden folgende Konkurrenten genauer untersucht:

- Präsentationsmöglichkeiten für Poster
- bestehende Videowalls
- der Videowall ähnliche Produkte

#### V.3.5.1 Präsentationsmöglichkeiten für Poster

Wie der Aufgabenstellung (VIII Anhang) entnommen werden kann, ist das Präsentieren der Bachelorposter eine Anforderung an die Arbeit, die auch umgesetzt werden soll. Daher ist eine Analyse von konkurrierenden Präsentationsarten sinnvoll.

Es gibt verschiedene Arten, wie die Bachelorposter präsentiert werden können:

- In der bestehenden Papierform an der Bachelorausstellung
- Über das Web:  
Für die Abteilung Informatik besteht eine Lösung zur Präsentation der Bachelorarbeiten mit der Eprints-Website<sup>2</sup>. Dieselbe oder eine ähnliche Lösung wäre für die Präsentation der Bachelorposter aller Abteilungen der HSR denkbar.
- Über einen Newsletter:  
Der Newsletter soll halbjährlich auf die Bachelorposter, welche für die im vergangenen Semester durchgeführten Arbeiten erstellen wurden, aufmerksam machen. Dazu sollen diese Poster im Web abrufbar sein.
- Über die HSR Videowall

Nachfolgende Tabelle zeigt, welche Vor- und Nachteile mit den einzelnen Möglichkeiten verbunden sind.

Präsentationsart	Vorteile	Nachteile
Papierform	- Visuelle Anziehung	- Statisch - Ausstellungszeit vorgegeben und beschränkt - Nur aktuelle Poster - An verschiedenen Standorten ausgestellt
Web	- Alle bisher erstellten Poster jederzeit verfügbar	- Das Interesse an den Postern muss explizit vorhanden sein
Newsletter	- Explizite Aufforderung weckt Interesse der Leserschaft	- Nur aktuelle Poster - Erreicht nur Newsletterabonnenten
HSR Videowall	- Interaktiv, weckt Spieltrieb, Spassfaktor - Alle bisher erstellten Poster jederzeit verfügbar - Poster sind an zentralem, viel besuchten Ort „ausgestellt“ - Man stösst eher zufällig auf Poster, kann verborgenes Interesse wecken	

Tabelle 6 - Vor- und Nachteile der Posterpräsentationsarten

Die Videowall ist interaktiv. Der Nutzer trifft eher spontan auf die Poster, welche zentral und vollständig vorhanden sind, was vermutlich sein Interesse für die Poster weckt. Die HSR Videowall bietet für die Präsentation der Bachelorposter nur Vorteile und eignet sich daher ideal für diese Aufgabe.

#### V.3.5.2 Bestehende Videowalls

In diesem Kapitel wird untersucht, was für Videowalls bestehen und was diese bieten. Dazu soll festgehalten werden, was eine Videowall ist: Eine für die Öffentlichkeit zugängliche Anzeige mit dynamischen Inhalten wie

<sup>2</sup> <http://eprints.hsr.ch/>

Bilder oder Videos. Die Steuerung erfolgt durch Körperbewegungen mithilfe des Microsoft Kinect Sensors<sup>3</sup> (weitere Informationen im Kapitel V.3.7 Interaktionsbereich des Kinect Sensors).

Um herauszufinden, was für Videowalls bestehen, wurde Recherche im Internet betrieben. Es konnten vornehmlich Videoaufnahmen von solchen Installationen gefunden werden. Alle nachfolgend aufgelisteten, mit einer kurzen Beschreibung versehenen Videowalls werden durch Körperbewegung, also ohne Berührungen, gesteuert.

Videowall	Kurzbeschreibung
<b>Repetto Paris</b> <sup>4</sup>	Mittels Swipe-Gesten können verschiedene Videoausschnitte angezeigt werden.
<b>Advanced Interface Design</b> <sup>5</sup>	Der vorbeilaufende Passant steuert das Abspielen des Videos. Geht man rückwärts, spult das Video zurück.
<b>Diesel</b> <sup>6</sup>	Mit wilden Hand- und Armbewegungen kann im Schaufenster mit dem Wetter gespielt und ein Unwetter mit Wind und Blitzen erzeugt werden.
<b>Chanel Paris</b> <sup>7</sup>	Die über einen Hintergrund unregelmässig verteilten Chanel-Logos können durch Körperbewegungen weggewischt werden. Nach kurzer Wartezeit nehmen die Logos wieder ihren alten Platz ein.
<b>Interactive Mirror</b> <sup>8</sup>	Läuft ein Passant am Spiegel vorbei, erzeugt das farbige „Flammen“, welche durch zusätzliche Armbewegungen verstärkt werden können.
<b>Swivel 3D Virtual Dressing Room</b> <sup>9</sup>	In einem virtuellen Ankleidezimmer kann man durch Auswahl der Kleider und Accessoires mit der Hand diese anprobieren und schliesslich vom Outfit ein Foto machen lassen.
<b>Interaktive Vitrine NSE</b> <sup>10</sup>	In einer interaktiven Vitrine befinden sich zwei angekleidete Figuren, unterteilt in Kopf, Oberkörper und Unterleib. Mit Handbewegungen kann jeder Teil mit den anderen kombiniert/ausgetauscht werden.
<b>Bank of Moscow</b> <sup>11</sup>	Mittels Swipe-Gesten kann durch den Zeitstrahl der Bank gescrollt werden. Durch längeres Stillhalten der Hand kann eine 1-Jahres-Periode genauer angeschaut werden. Weiter können auch Bilder angeschaut werden.
<b>Kiwibank</b> <sup>12</sup>	Das Nachahmen einer bestimmten Körperposition startet ein Spiel, bei dem während dem Fliegen Coins gesammelt werden können. Zwei Personen können gleichzeitig je mit einer Bewegung der linken und rechten Hand bunte, vorbeiziehende Sprechblasen öffnen und den darin versteckten Text hervorholen. In einer dritten Applikation kann durch Kurzbeschreibungen von „New Zealander of the Year & Local finalists“ gescrollt werden, das Bild kann durch Anklicken mit der Hand vergrössert werden.

Tabelle 7 - Bestehende Videowalls mit Kurzbeschreibung

Videowalls und ähnliche Produkte werden an Showcases wie beispielsweise dem NEC Showcase 2012<sup>13</sup> präsentiert. Der Swivel 3D Virtual Dressing Room wurde z.B. an ebendiesem Showcase präsentiert.

Zu keiner in den Videos der Tabelle 7 - Bestehende Videowalls mit Kurzbeschreibung vorgestellten Videowalls existierte eine technische Beschreibung über die verwendete Hard- und Software und wie die Videowall umgesetzt wurde.

Videowalls mit Kinect sind am Entstehen und werden im Moment noch nicht serienmäßig hergestellt. Sie werden im Rahmen von Technology-Showcases gezeigt.

<sup>3</sup> <http://www.xbox.com/de-DE/kinect>

<sup>4</sup> Repetto Paris : <http://www.youtube.com/watch?v=La2xIJ-SzwQ&feature=related>

<sup>5</sup> Advanced Interface Design : <http://www.youtube.com/watch?v=xFgvNMN2DiQ&feature=related>

<sup>6</sup> Diesel: <http://www.youtube.com/watch?v=wT2zyT5eJIU&feature=related>

<sup>7</sup> Chanel Paris: <http://www.youtube.com/watch?v=CLD1wVbcD8w&feature=related>

<sup>8</sup> Interactive Mirror: <http://www.youtube.com/watch?v=4F3rnV3-6VM&feature=related>

<sup>9</sup> Swivel 3D Virtual Dressing Room: <http://www.youtube.com/watch?v=y0TSw15aYyk>

<sup>10</sup> Interaktive Vitrine NSE : <http://www.youtube.com/watch?v=lmSoV2Mb8gE&feature=related>

<sup>11</sup> Bank of Moscow : <http://www.youtube.com/watch?v=KBHgRcMPaYI&feature=related>

<sup>12</sup> Kiwibank : <http://www.youtube.com/watch?v=Yk6PLmUY3tw&feature=related>

<sup>13</sup> NEC Showcase 2012 : <http://www.showcase-nec.com/index.php/showcase2012/zones/ret/>

### V.3.5.3 Der Videowall ähnliche Produkte

Um in Erfahrung zu bringen, welche alternativen Arbeiten es gibt, die der Videowall ähnlich sind, werden in diesem Unterkapitel ähnliche kommerzielle und wissenschaftliche Projekte betrachtet.

Die nachfolgend kurz beschriebenen Projekte sind unterteilt nach Projekten, welche eine Interaktion mit einer Videowall (siehe nachfolgendes Unterkapitel V.3.5.3.1 Interaktion mit Videowall) und nach Projekten, bei welchen die Interaktion mit Kinect oder einer ähnlichen Kamera funktioniert (siehe Unterkapitel V.3.5.3.2 Interaktion ohne Videowall).

#### V.3.5.3.1 Interaktion mit Videowall

Da die Videowall ohne zusätzliche Hilfsmittel wie Tastatur oder Maus bedienbar sein soll, müssen andere Möglichkeiten gefunden werden, um mit der Wall zu interagieren.

##### V.3.5.3.1.1 HoloWall

Beim Projekt HoloWall [matsushita03] wurde untersucht, wie eine Wall, welche Körperteile oder Objekte mittels Infrarot erkennen kann, funktionieren könnte. Die Infraroterkennung findet aber erst statt, wenn beispielsweise ein Finger genug nahe an der Bildschirmoberfläche ist.

Die Infrarot-Technik [han05] ist bekannt und wird z.B. beim Microsoft Surface<sup>14</sup> angewendet. Es stellen sich aber zwei Probleme:

- Bei der Anwendung mit einer Wall mit grossen Abmessungen ergibt sich das Problem, dass nicht die ganze Bildschirmfläche und nicht alle darauf befindlichen Elemente für den Nutzer erreichbar sind.
- Die bauliche Situation lässt den Einbau eines Back Projectors nicht zu.

Deshalb kann diese Technologie nicht für die Bachelorarbeit verwendet werden.

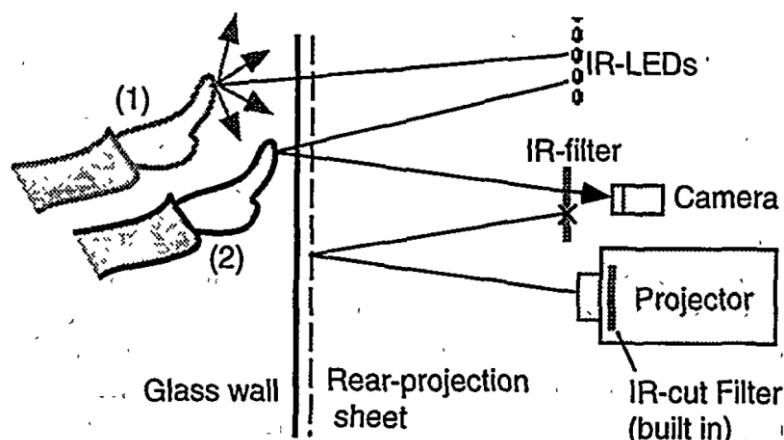


Abbildung 8 - Konfiguration der HoloWall, Bildquelle [matsushita03]

##### V.3.5.3.1.2 It's Mine, Don't Touch!: Interactions at a Large Multi-Touch Display in a City Centre

Die Touch Wall des Artikels It's Mine, Don't Touch!: Interactions at a Large Multi-Touch Display in a City Centre [peltonen08] kann ausschliesslich mit Touch-Gesten bedient werden. Eine Besonderheit dieser Applikation ist, dass eine Interaktion mit mehreren Personen gleichzeitig möglich ist.

Wie im obenstehenden Projekt (V.3.5.3.1.1 HoloWall) besteht hier eine ähnliche Problematik: Nicht alle Elemente sind auf dem grossen Bildschirm erreichbar, somit kann nicht überall eine Touch-Geste ausgeführt werden.

Deshalb eignet sich diese Technologie für dieses Bachelorprojekt nicht.

<sup>14</sup> <http://www.microsoft.com/surface/en/us/default.aspx>

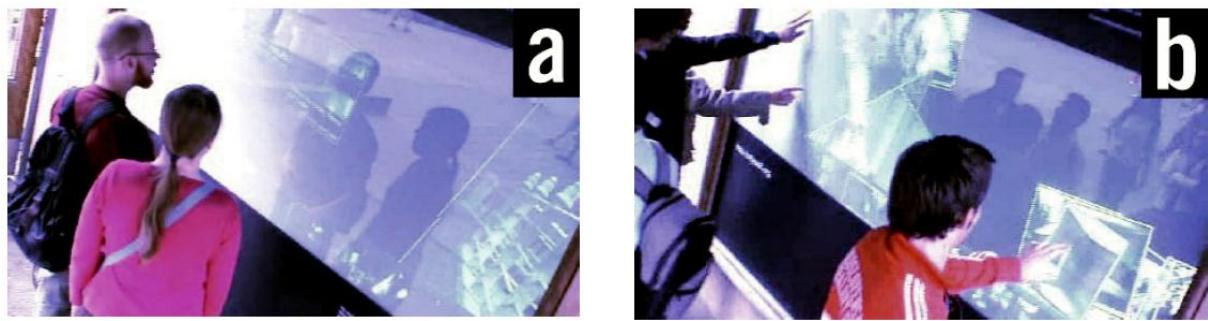


Abbildung 9 - Touch Wall Setup: It's Mine, Dont't Touch!, Bildquelle: [peltonen08]

#### V.3.5.3.1.3 Extending Touch: Towards Interaction with Large-Scale Surfaces

In der Arbeit Extending Touch: Towards Interaction with Large-Scale Surfaces [schick09] wurde untersucht, wie eine grosse Videowall bedient werden kann. Hierbei wurde festgestellt, dass Touch für eine solche Wall nicht ausreicht, weshalb zusätzlich zu Touch eine „Pointer Interaktion“ entwickelt wurde. Der Benutzer kann so auf ein Objekt zeigen und dieses bewegen.

Für dieses Projekt kamen spezielle Kameras zum Einsatz, die 3D Erkennung musste selbst entwickelt werden. Deshalb kommt diese Technologie ebenfalls nicht in Frage für diese Bachelorarbeit. In dieser Bachelorarbeit wird für die 3D Erkennung die vergleichsweise kostengünstige Kinect inklusive Framework verwendet.



Abbildung 10 - Extending Touch, Bildquelle [schick09]

---

#### V.3.5.3.2 Interaktion ohne Videowall

Im Bereich der Kameraerkennung wird heute noch stark geforscht, da sich durch schnellere Rechner und parallele Berechnungen auf der Grafikkarte neue Möglichkeiten ergeben, Kamera-Input zu erkennen und zu verarbeiten.

### V.3.5.3.2.1 Microsoft Kinect Sensor and Its Effect

In dem Paper Microsoft Kinect Sensor and Its Effect [zhang12] ist beschrieben, was für Funktionen der Kinect 3D Sensor bietet und was für Möglichkeiten daraus entstehen. Im Speziellen wird auf das Kinect Skeletal Tracking [microsoft10] genauer eingegangen.

Da die HSR Videowall auf der Kinect-Technologie basiert, ist dieses Paper besonders interessant.

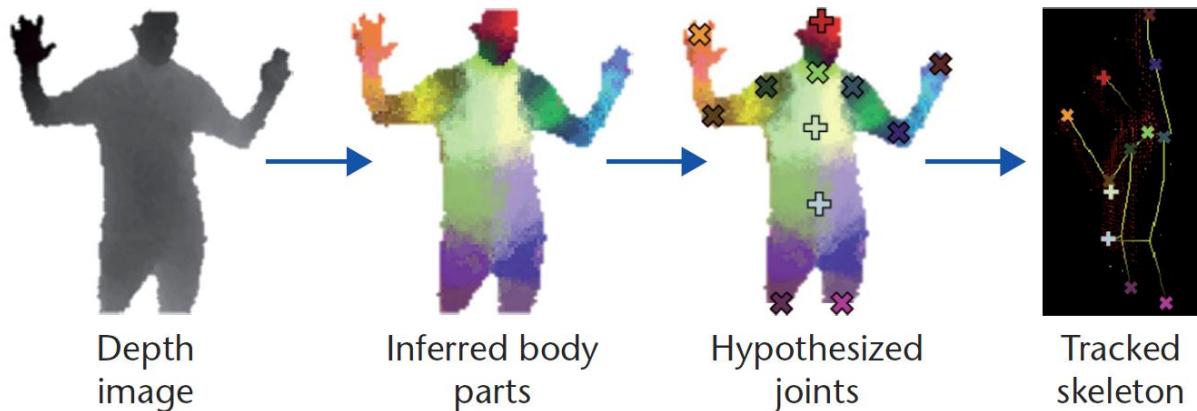


Abbildung 11 - Skeletal Tracking, Bildquelle [zhang12]

### V.3.5.3.2.2 Leap Motion

Leap Motion<sup>15</sup> ist eine Art 3D-Kamera, ähnlich wie der Kinect Sensor. Anders als Kinect fokussiert dieses Gerät die Interaktion mit den Händen.

Möchte man Leap Motion für die HSR Videowall einsetzen, so müsste abgeklärt werden, wo der Sensor für eine optimale Erkennung platziert werden müsste. Möglicherweise würde dieser Punkt mitten im Raum liegen, da der interaktive 3D-Raum, der vom Sensor generiert wird, beschränkt ist. Zusätzlich wird die Hardware gemäss Hersteller [leapmotion12] erst im Dezember 2012 oder Januar 2013 verfügbar sein. Aus diesen Gründen ist Leap Motion für diese Bachelorarbeit nicht geeignet.



Abbildung 12 - Leap Motion Sensor, Bildquelle: [www.technobuffalo.com](http://www.technobuffalo.com)

### V.3.5.3.2.3 Panasonic D-IMager

D-IMager<sup>16</sup> ist, wie der Kinect Sensor, eine 3D Kamera. Die Auflösung des 3D Bildes ist mit 160x120px im Vergleich zur Kinect mit 640x480px deutlich kleiner. Zusätzlich ist es schwierig, an weitere Informationen oder an Codebeispiele heranzukommen.

Aus diesen Gründen wird der Kinect Sensor dem D-IMager Sensor vorgezogen.

<sup>15</sup> <http://www.leapmotion.com/>

<sup>16</sup> <http://pewa.panasonic.com/components/built-in-sensors/3d-image-sensors/d-imager/>



Abbildung 13 - Panasonic D-IMager, Bildquelle: [www.panasonic.biz](http://www.panasonic.biz)

#### V.3.5.3.2.4 Dance Dance Revolution

Das Spiel Dance Dance Revolution<sup>17</sup> wird mit einem Controller, der mit den Füßen zu bedienen ist, gesteuert. So ein Controller könnte auch für die Videowall genutzt werden. Allerdings ergeben sich dabei folgende Probleme:

- Die Steuerung ist eingeschränkt auf die Bedienung mit den Füßen.
- Man muss sich zur Bedienung an einen vorbestimmten Ort stellen. Durch den benötigten Aufbau geht der Wow-Effekt, den Kinect mit der Erkennung ohne sichtbare Installation bietet, verloren.

Aus diesen Gründen ist diese Art von Bedienung für die HSR Videowall nicht geeignet.

Interessanterweise wurde das Spielprinzip im Spiel DanceEvolution<sup>18</sup> für die Xbox übernommen, wobei die Steuerung des Spiels mit Kinect funktioniert.



Abbildung 14 - Dance Dance Revolution Game, Bildquelle : [www.wikipedia.org](http://www.wikipedia.org)

<sup>17</sup> [http://en.wikipedia.org/wiki/Dance\\_Dance\\_Revolution](http://en.wikipedia.org/wiki/Dance_Dance_Revolution) (Anmerkung: Die Wikipedia-Seite wurde verwendet, da die Original-Seite <http://www.konami.jp/bemani/ddr/jp/> lediglich auf Japanisch verfügbar ist.)

<sup>18</sup> <http://www.konami-dmstar.com/danceevolution/>

### V.3.5.3.3 Fazit

Die HSR Videowall soll eine Monitorwand sein, auf der dynamisch Inhalte wie Text, Bilder, Videos usw. angezeigt werden können. Die Steuerung ist im Begriff miteingeschlossen und erfolgt mittels Kinect.

Wie in der Aufgabenstellung (siehe VIII Anhang) beschrieben ist, ist eine Aufgabe der Bachelorarbeit, Bachelorposter auf der Videowall auszustellen. Damit die Texte der Poster angenehm gelesen werden können, ist eine möglichst hohe Auflösung für eine gute Lesbarkeit wichtig.

Eine weitere Anforderung ist, dass die mit dem Kinect Sensor interagierende Person mit optimal drei bis vier Metern Abstand zur HSR Videowall steht. Durch diese Restriktion ist eine Projektion mit einem Beamer aufgrund der Minimaldistanz, welche dieser für die Projektion benötigt, nicht möglich. Auch Kurzdistanzbeamer können keine zufriedenstellende Lösung bieten, da deren Auflösung zurzeit mit WXGA (1280×800) noch zu gering ist.

Der Stand der Technik ist noch nicht ausgereift für die optimale Videowall:

- Recherchen zeigen aber, dass Monitore mit schmaler Rahmenbreite grösser und billiger werden. Eingeholte Offerten über für die Videowall geeignete Monitore können im Anhang (VIII Anhang) eingesehen werden.
- Die Steuerungssoftware ist noch unausgereift. Die Herausforderung ist es, eine möglichst gute Auflösung und eine zugleich eine hohe Performance zu erhalten. Es gibt kein Angebot einer oder mehrere Grafikkarten, die eine hochauflöste Darstellung und fließende Animationen gleichzeitig bietet. Das Kapitel V.7.3.2 Grafikkarten bietet weitere Informationen über die Matrox-Grafikkarten, die für die HSR Videowall angeschafft wurden.
- Wie aus dem Unterkapitel V.3.5.3 Der Videowall ähnliche Produkte ersichtlich ist, werden für die Interaktion zwischen Passanten und einer Videowall verschiedene Lösungen angeboten. Es gibt aber noch keine Standardlösung.

Wie aus der Analyse bestehender Videowalls hervorgeht (Unterkapitel V.3.5.2 Bestehende Videowalls), sind diese erst am Entstehen. So hat beispielsweise die bekannte Firma Yahoo<sup>19</sup> für ihren Hauptsitz in Kalifornien in Zusammenarbeit mit Tronic<sup>20</sup> eine Interactive Video Wall<sup>21</sup> geschaffen.

Die Arbeit hat zum Ziel, eine Videowall, die aus mehreren grossen Monitoren besteht, hervorzubringen. Die einzelnen Bildschirme sollen mit einer möglichst hohen Auflösung betrieben werden.

Die Konkurrenzanalyse ergab, dass es noch kein Produkt oder Projekt gibt, welches die Anforderungen aus der Aufgabenstellung (siehe VIII Anhang) vollständig erfüllt. Gewisse Teilprobleme wurden aber in Projekten bereits untersucht. Aus diesem Grund wird in dieser Bachelorarbeit ein Prototyp erarbeitet, welcher die geforderten Ansprüche erfüllen soll.

<sup>19</sup> <http://www.yahoo.com/>

<sup>20</sup> <http://tronicstudio.com/>

<sup>21</sup> <http://www.youtube.com/watch?v=VnL03JyLVcA>

### V.3.6 Passantenanalyse

Um festzustellen, mit welchem Abstand zu der Wand, an welcher die Videowall befestigt werden soll, sich die Passanten bewegen, wurde am 28. Februar eine Benutzeranalyse durchgeführt. Dabei wurde beobachtet wie viele Personen innerhalb von zwei Minuten den Eingangsbereich passieren. Zudem konnten dadurch die verschiedenen Gruppengrößen, in denen sich die Passanten im Verwaltungsgebäude bewegen, analysiert werden. Um das Verhalten möglichst vieler Personen erfassen zu können, wurden für die Beobachtung die zwei Hauptaktivitätszeiten eines normalen Wochentages ausgesucht. Dies sind die Zehn-Uhr- und die Mittagspause. An folgenden Daten wurden zu folgenden Zeiten Beobachtungen durchgeführt:

Datum	Beginn	Ende
28.02.2012	9:28	10:16
28.02.2012	11:23	13:10

Tabelle 8 - Beobachtungszeitabschnitte

Während diesen Zeiten wurde das Verhalten von insgesamt 1512 Personen festgehalten. Diese haben sich über die Zeit wie folgt verteilt:

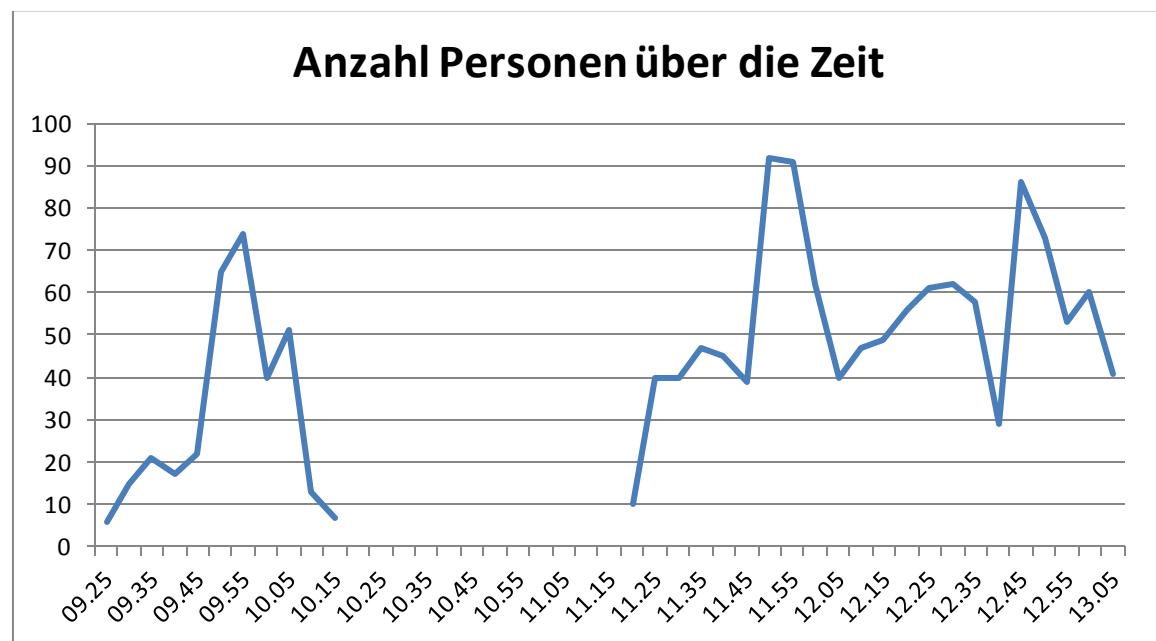


Abbildung 15 - Anzahl Personen über die Zeit

### V.3.6.1 Abstandszonen

Durch die Beobachtung konnte die Auslastung der verschiedenen Abstandszonen ausgewertet werden.

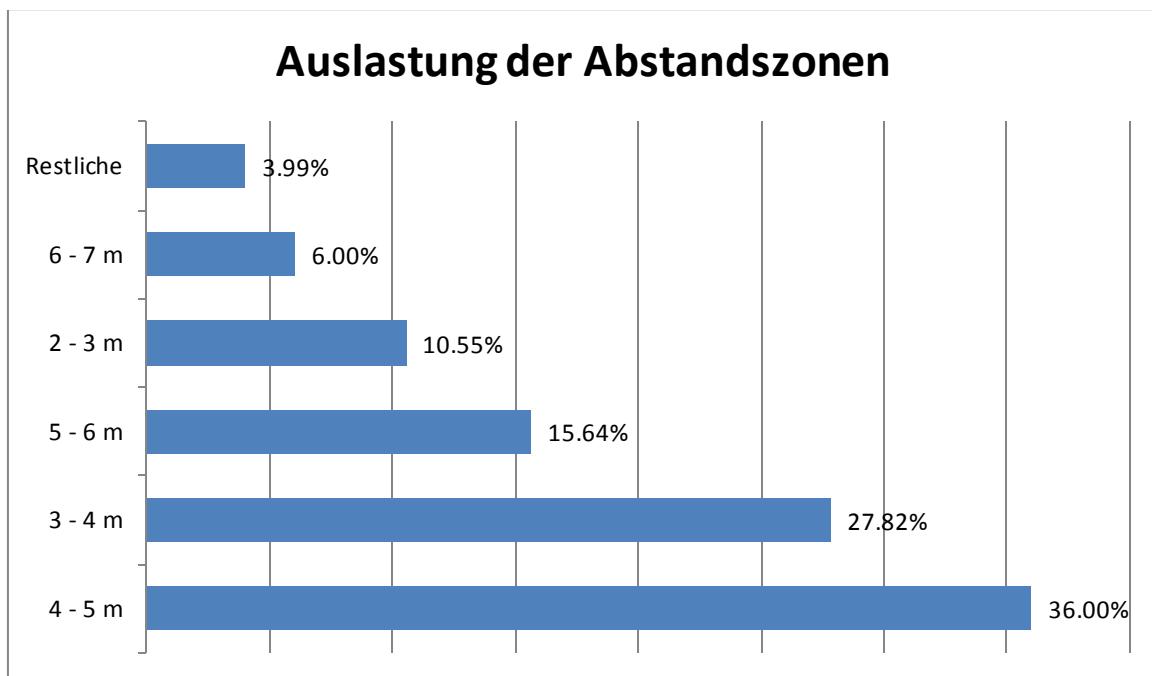


Abbildung 16 - Auslastung der Abstandszonen

Im Grundriss sind diese Werte auf der folgenden Abbildung 17 - Auslastung der Abstandszonen, Grundriss Verwaltungsgebäude ersichtlich.

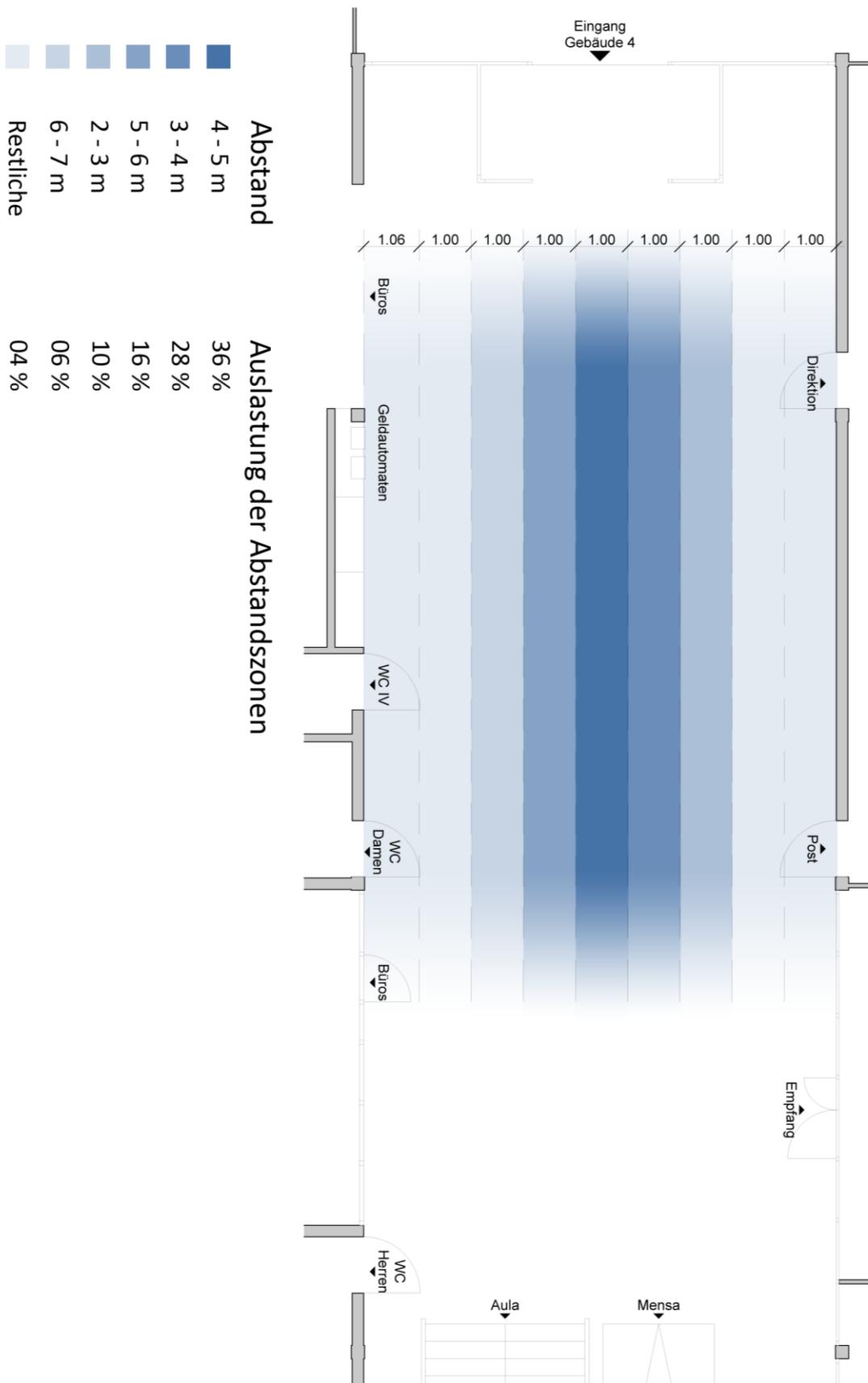


Abbildung 17 - Auslastung der Abstandszonen, Grundriss Verwaltungsgebäude

### V.3.6.2 Gruppengrößen

Folgende Gruppengrößen wurden beobachtet und im folgenden Diagramm prozentual ausgewertet:

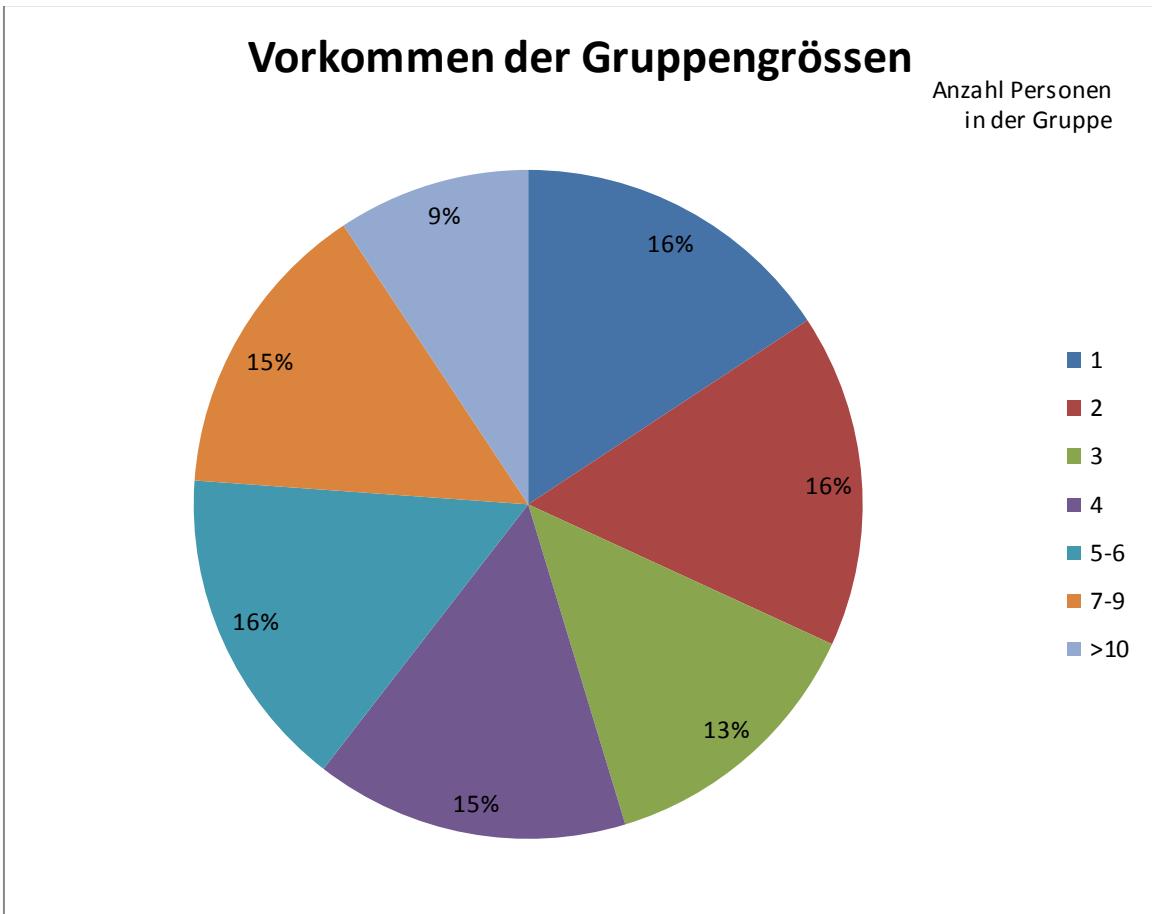


Abbildung 18 - Vorkommen der Gruppengrößen

Wichtig für dieses Projekt ist jedoch vor allem, wie viele Personen sich einzeln oder allgemein in Gruppen bewegen. Dies kann folgendem Diagramm entnommen werden:

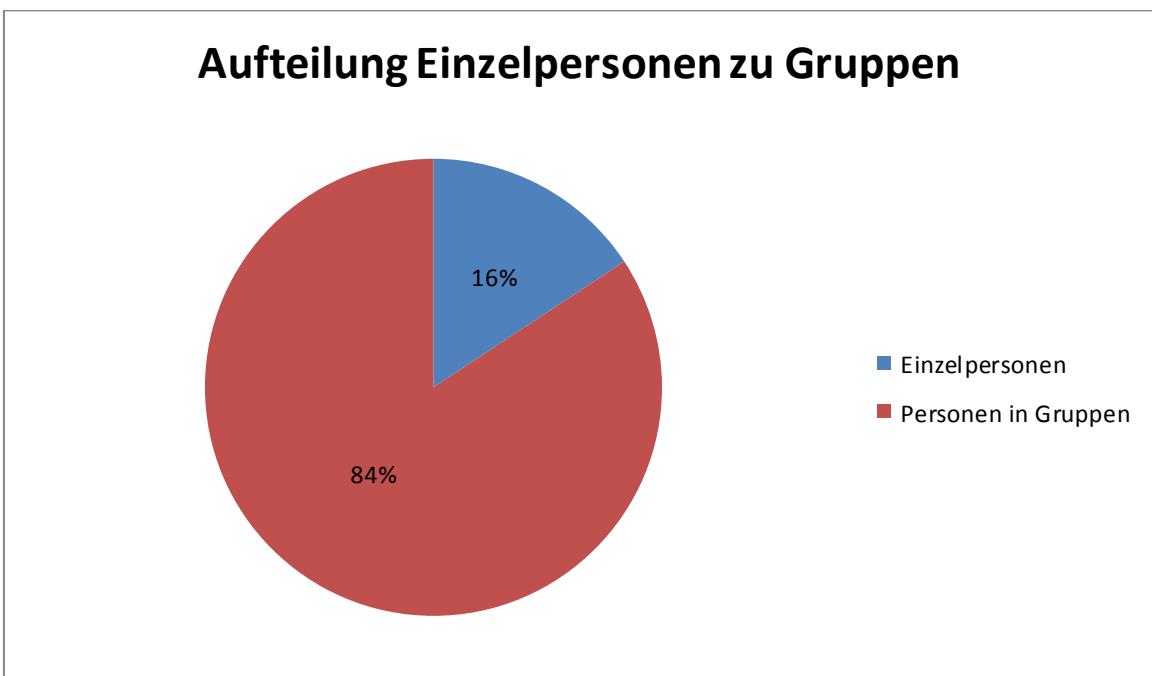


Abbildung 19 - Aufteilung Einzelpersonen zu Gruppen

### V.3.7 Interaktionsbereich des Kinect Sensors

Kinect bietet eine Skelett-Erkennung. Dabei handelt es sich um 20 Punkte des Körpers (zum Beispiel linke Hand, linker Ellbogen etc.), welche erkannt und zu einem Skelett verbunden werden. Mit dieser Skelett-Erkennung können die Bewegungen von bis zu zwei Personen erkannt und verfolgt werden. Der Kinect Sensor hat aber nur einen beschränkten Bereich, in dem er Personen erkennen kann.

Um ein ungefähres Bild über die Grösse dieses Bereichs zu erhalten, sind im Verwaltungsgebäude Tests mit Kinect durchgeführt worden. Hierfür wurde die Kinect Explorer - Applikation, welche Bestandteil des Kinect for Windows Developer Toolkits<sup>22</sup> ist verwendet. Mit Klebeband wurden die Abstände von einem, zwei, drei, vier und fünf Metern zur Wand gekennzeichnet. Eine Person des Teams, welches die Ausmessungen durchführte, stellte sich auf die Abstandsmarkierung und bewegte sich langsam und zum Sensor gedreht entlang der Markierung nach links und später nach rechts. Die zweite Person prüfte mit der Applikation, bis zu welcher Position das Skelett der Person noch vom Gerät erkannt wurde. Sobald das Skelett der Person in der Testapplikation nicht mehr sichtbar war, wurde die Position mit Klebeband am Boden gekennzeichnet.

Zusätzlich wurden Tests durchgeführt, um zu bestimmen, wie gut Kinect Personen erkennt, die den Erkennungsbereich des Sensors durchlaufen. Dazu lief eine Person mehrere Male mit normaler Geschwindigkeit parallel zur Wand ausgerichtet mit verschiedenen Abständen durch den Erkennungsbereich von Kinect. Sie wurde meistens erkannt. Die Reaktion auf eine vorbeigehende Person, also das Erkennen und Anzeigen des Skeletts dieser Person, ist jedoch langsam. Da die Erkennung aber gewährleistet ist, ist das Risiko 3: „Kinect: Erkennung von der Seite“ des Risikomanagements (siehe dazu VIII Anhang) bereinigt.

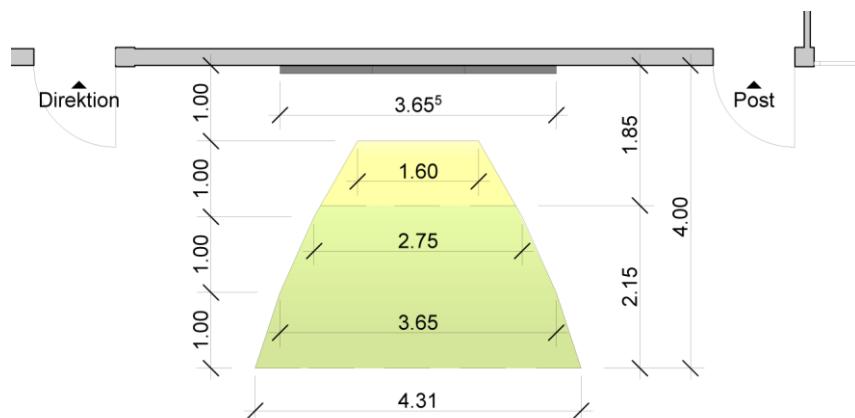


Abbildung 20 - Kinect Skelett-Erkennungsbereich, Grundriss Verwaltungsgebäude

Grün = Fläche in der Skelett von Kinect erkannt wird.  
Gelb = Fläche in der Skelett verfolgt werden kann, nachdem es erkannt wurde.  
Weiss = Fläche in der keine Erkennung durch Kinect stattfinden kann.

Zur Durchführung der Messungen wurde der Sensor auf einer Höhe von 39 cm aufgestellt, mit einem Winkel von 10°. Abbildung 20 - Kinect Skelett-Erkennungsbereich, Grundriss Verwaltungsgebäude zeigt den mit Hilfe der Aufnahmen ausgemessenen Bereich, innerhalb wessen Kinect Personen erkennen und deren Skelett darstellen kann. Dieser liegt zwischen 185 und 400 cm, im rechten Winkel zur Wand gemessen. Der Bereich wird in der Grafik grün dargestellt. Ist man bereits von Kinect in der grünen Zone erkannt worden, kann man sich auch weiter nach vorne in den gelben Bereich (weniger als 185 cm Abstand zum Sensor) bewegen. Dort wird man bis zu einem Meter Abstand noch erkannt, Füsse und Kopf sind jedoch nicht mehr sichtbar. Daher eignet sich dieser gelbe Bereich nur noch bedingt für die Interaktion.

<sup>22</sup> <http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx>

### V.3.8 Befragung

Die Poster der Bachelorarbeiten aller Abteilungen werden als Inhalt einer ersten Anwendung eingesetzt. Im Meeting vom 20.02.2012 wurde die Möglichkeit, zusätzlich zu den Postern interaktive Inhalte wie Videos aufzuschalten, diskutiert. Es stand auch die Frage im Raum, ob in Zukunft nur noch Videos zur Präsentation der Bachelorarbeiten auf der Videowall gezeigt würden. Der Vorteil von Videos besteht darin, dass der Betrachter keine Anstrengungen unternehmen muss, um zu den gewünschten Informationen zu kommen. Ein Video vermittelt dem Zuschauer in kurzer Zeit alle relevanten Informationen über die Arbeit, welche er ansonsten selbst aus dem Poster erfassen müsste. Zudem wird durch den Einsatz von visuellen Effekten schnell die Aufmerksamkeit des Zuschauers erlangt.

Um herauszufinden, wie intensiv sich die aktuell an der HSR immatrikulierten Studenten bisher allgemein für die über die Bachelorarbeiten veröffentlichten Informationen interessierten und ob sie bereit wären, für ihre Arbeit ein Video zu erstellen, wurde eine Befragung durchgeführt. Der dazu erstellte Fragebogen ist im Unterkapitel V.3.8.1 Fragebogen zu finden.

Es wurden total 203 Studenten der HSR befragt. Hierbei wurden die Studenten direkt angefragt, ob sie den Fragebogen ausfüllen möchten. Im Vergleich zu einer Online-Umfrage konnte auf diese Weise in sehr kurzer Zeit eine hohe Anzahl an Personen befragt werden. Um ein repräsentatives Umfrageergebnis zu erhalten, wurde dabei beachtet, dass pro Abteilung ein Minimum von 20 Meinungen eingeholt wurde. Weiter wurden pro Studiengang Studenten aus unterschiedlichen Semestern befragt. Die Tabelle 9 - Anzahl Fragebögen pro Abteilung zeigt, aus welcher Abteilung wie viele Studenten einen Fragebogen ausfüllten.

Abteilung	Anzahl Fragebögen
Bauingenieurwesen	25
Elektrotechnik	29
Erneuerbare Energien und Umweltechnik	20
Informatik	37
Landschaftsarchitektur	23
Maschinentechnik	48
Raumplanung	21

Tabelle 9 - Anzahl Fragebögen pro Abteilung

---

### V.3.8.1 Fragebogen

Name: \_\_\_\_\_

Studienrichtung: \_\_\_\_\_

Semester: \_\_\_\_\_

**Q: Ich sehe mir die Ausstellung der Bachelorarbeiten an und lese die Poster aufmerksam durch.**

- trifft zu       trifft eher zu       trifft eher nicht zu       trifft nicht zu

**Q: Ich habe mich schon in der Bachelorarbeitsbroschüre über interessante Bachelorarbeiten informiert.**

- trifft zu       trifft eher zu       trifft eher nicht zu       trifft nicht zu

**Q: Ich empfinde die Präsentation der Bachelorarbeiten als wertvoll und interessant.**

- trifft zu       trifft eher zu       trifft eher nicht zu       trifft nicht zu

**Q: Das Lesen der Poster oder der Broschüre ist mir zu zeitaufwändig.**

- trifft zu       trifft eher zu       trifft eher nicht zu       trifft nicht zu

**Q: Durch das Lesen der Poster oder der Broschüre erhalte ich einen guten Eindruck über den Umfang der Arbeiten.**

- trifft zu       trifft eher zu       trifft eher nicht zu       trifft nicht zu

**Q: Ein kurzes Video (2 Minuten) würde die gleichen Informationen, welche auf einem Poster vorhanden sein würden, in ansprechenderer Weise vermitteln.**

- trifft zu       trifft eher zu       trifft eher nicht zu       trifft nicht zu

**Q: Ich würde für meine Bachelorarbeit anstelle eines Posters lieber ein kurzes Video erstellen (2 Minuten).**

- trifft zu       trifft eher zu       trifft eher nicht zu       trifft nicht zu

**Q: Wenn das Video auf dem HSR YouTube Channel publiziert würde, macht mir das nichts aus.**

- trifft zu       trifft eher zu       trifft eher nicht zu       trifft nicht zu

### V.3.8.2 Auswertung

Die Abbildung 21 - Total aller Studiengänge zeigt die Auswertung der Antworten aller 203 befragten HSR-Studenten.

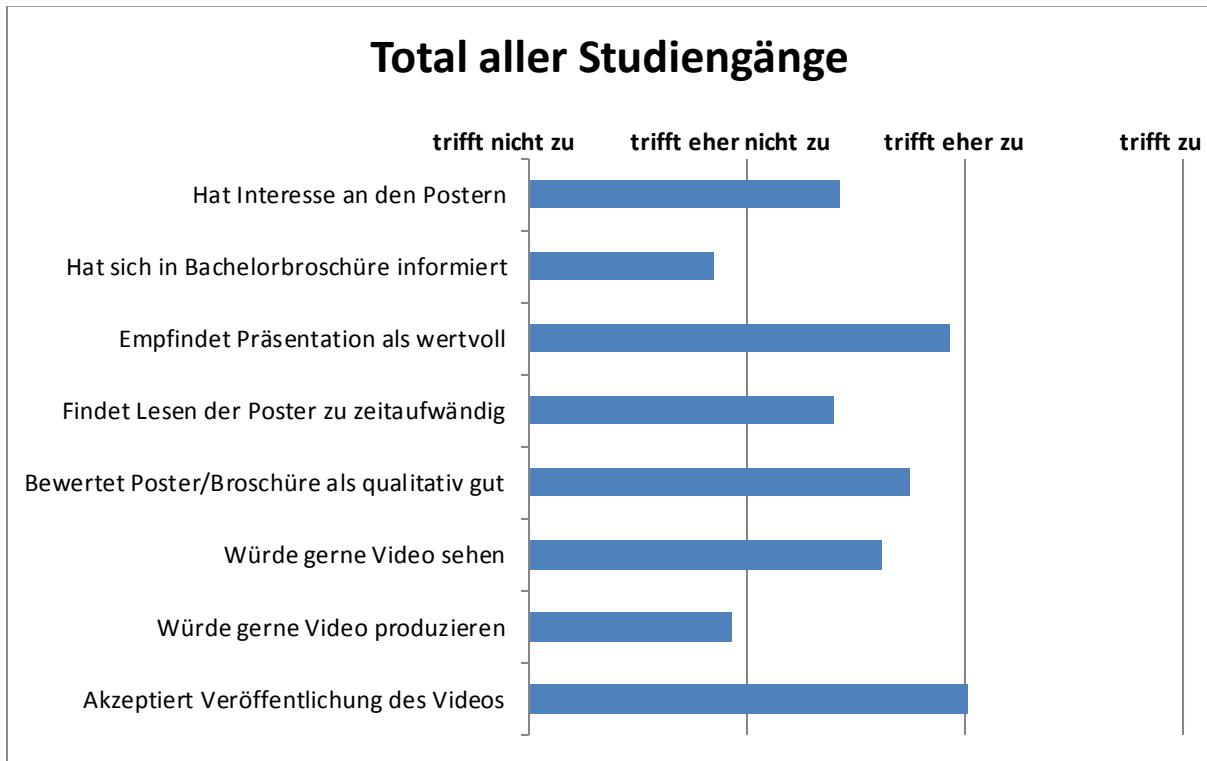


Abbildung 21 - Total aller Studiengänge

Wie aus der obigen Abbildung ersichtlich ist, zeigt in etwa die Hälfte aller befragten Studenten Interesse an den in der Bachelorausstellung präsentierten Postern und empfinden die darauf enthaltenen Informationen als eher wertvoll. Die Bereitschaft, ein Video anstelle eines Posters vorzulegen, ist gering. Das Konsumieren von Videos erhält hingegen mehr Zuspruch.

In der nachfolgenden Abbildung 22 - Vergleich der Studiengänge zeigt sich, dass die Antworten in den verschiedenen Abteilungen für gewisse Fragen merklich unterschiedlich ausgefallen sind.

Da der Zweck der durchgeföhrten Befragung war, Antworten zu „Würde gerne Video sehen“ zu erhalten, können als Beispiel die Antworten zu dieser Frage ein wenig genauer analysiert werden. 59% aller Studenten lehnen die Video-Erstellung ab. Studenten der Abteilungen Informatik, Erneuerbare Energien und Umwelttechnik antworteten am wenigsten abweisend. Daraus ist zu schliessen, dass sich ein Video zur Präsentation der in ihrem Studiengang zu erstellenden Arbeiten wohl besser eignet. Studierende des Bauingenieurwesens hingegen lehnen den Vorschlag, das Poster mit einem Video zu ersetzen, am eindeutigsten ab.

Zu dieser Frage wurden auf den Fragebögen vermehrt Bemerkungen hinterlassen, welche einerseits zum Ausdruck bringen, dass sich ein Video für gewisse Arbeiten nicht eignet. Weiter wird bemerkt, dass die Gestaltung des Posters ein wichtiger Teil der Arbeit sei. Auch wurde festgehalten, dass ein Video als Ergänzung zum Poster wohl eher auf Akzeptanz stossen würde. Andererseits äusserte sich ein Student der Raumplanung im Gespräch positiv gegenüber der Erstellung eines Videos, da es sich bestens für die Präsentation der 3D-Darstellungen eignen würde.

Im Dialog mit den Studenten, welche einen Fragebogen ausfüllten, bekundeten einige auch Bedenken über das Produzieren eines Videos, da sie kein Wissen über und keine Erfahrung mit den Erstellungswerkzeugen hätten. Weiter bestehe auch Unsicherheit darüber, ob die Qualität des Inhalts des Videos genügen würde.

## Vergleich der Studiengänge

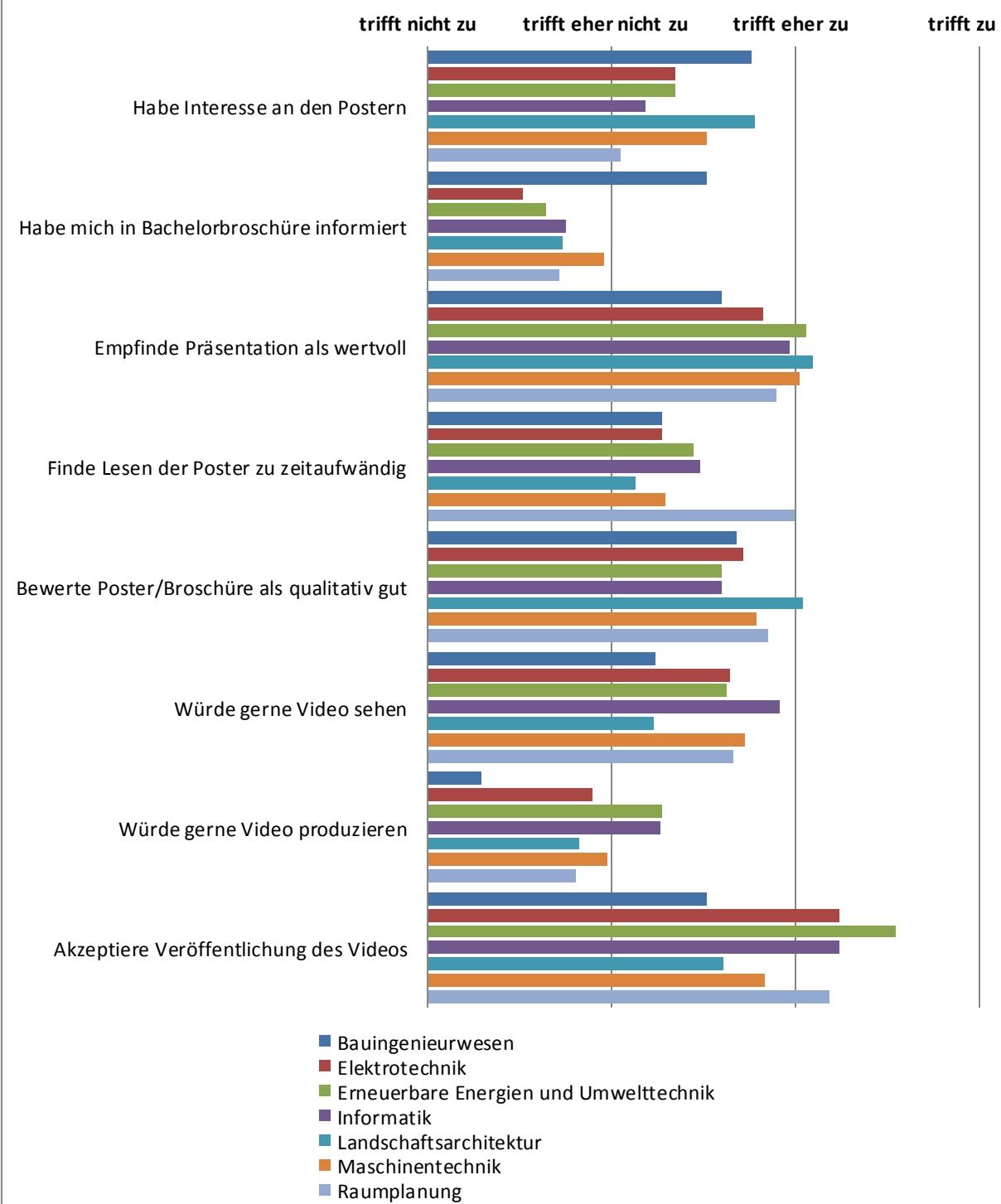


Abbildung 22 - Vergleich der Studiengänge

Die obigen zwei Abbildungen zeigen je nur den Durchschnitt der Meinungen aller Studenten (Abbildung 21 - Total aller Studiengänge) oder pro Abteilung, unterteilt nach Semester (Abbildung 22 - Vergleich der Studiengänge). Die untenstehende Abbildung 23 - Auswertung nach Quartilen zeigt die Verteilung der Antworten, welche dazu in Viertel unterteilt angezeigt wird. Zum besseren Vergleich ist in dieser Grafik zusätzlich der Durchschnitt aller Antworten (die gleichen Werte, welche in Abbildung 21 - Total aller Studiengänge zu sehen sind) auf eine bestimmte Frage eingezeichnet.

## Auswertung nach Quartilen, alle Studiengänge

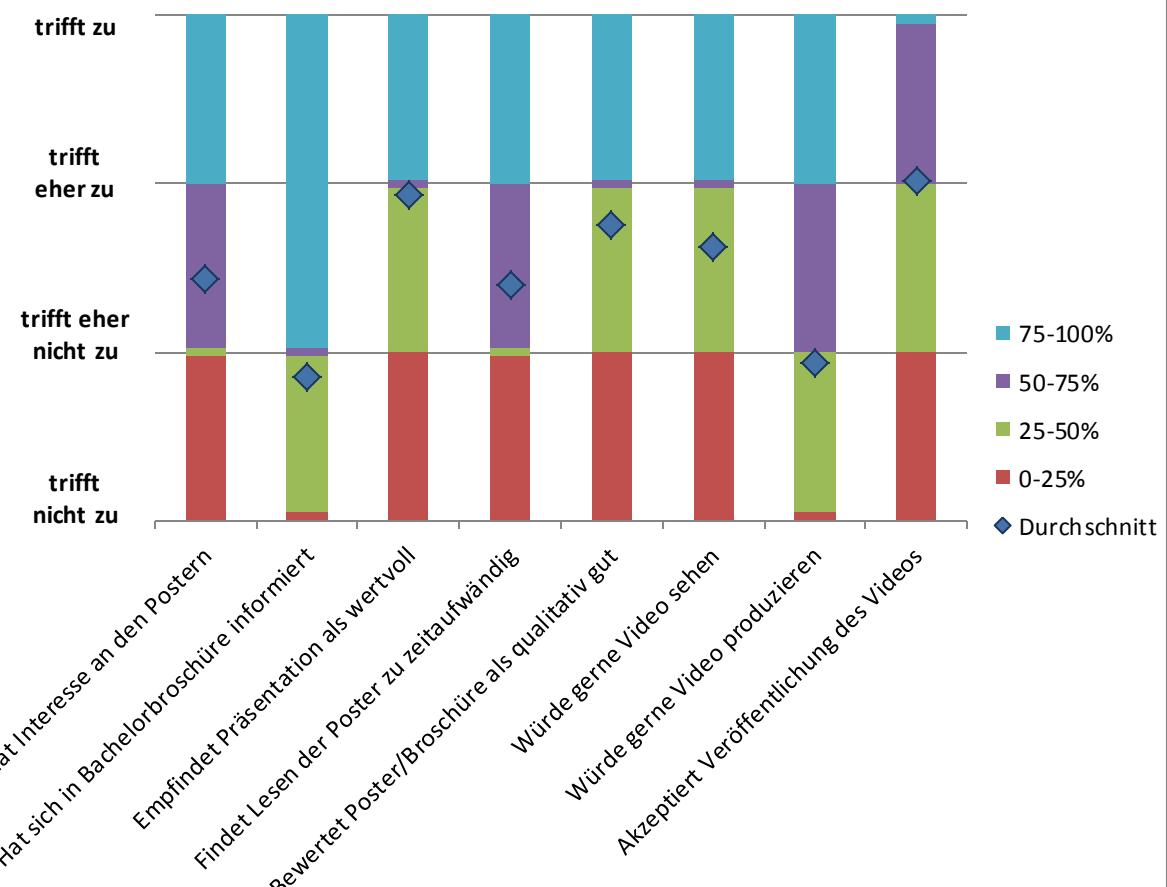


Abbildung 23 - Auswertung nach Quartilen

Alle ausgefüllten Fragebögen sind im Anhang (VIII Anhang) zu finden. Im selben Kapitel befindet sich auch die ausführliche Auswertung der Bögen pro Abteilung mit Unterscheidung der Antworten nach Semester.

### V.3.8.3 Fazit

Gerade durch Abbildung 23 - Auswertung nach Quartilen ist schnell ersichtlich, dass sich das Interesse der Studenten an den Postern in Grenzen hält. Dieser Wert kann auch mit der Präsentation der statischen Poster auf der Videowall kaum geändert werden. Jedoch wäre eine Steigerung der Attraktivität durch ein Poster mit dynamischen Elementen denkbar. Dabei bleibt das Grundbild das Poster. Die Studiengänge, für welche es Sinn macht, sollen aber die Möglichkeit haben, das Poster mit Videos zu erweitern. Das Drücken einer Schaltfläche, welche auf dem Poster positioniert ist, löst dann zum Beispiel die Produktdemo aus.

Auch sind zwei verschiedene Ideen zur Steigerung der Akzeptanz der Videos denkbar. Zum einen könnte aus einem speziellen Formular per Knopfdruck ein Video generiert werden. Dies würde eine Vereinfachung der Produktion von Videos für alle Studiengänge darstellen. Zum anderen könnte ein Beispielvideo zur Verfügung gestellt werden, damit sich die Studenten das Endprodukt bereits bis zu einem gewissen Grad vorstellen können. Damit auch Studenten, welche technisch weniger bewandert sind, mit möglichst geringem Aufwand zu einem akzeptablen Ergebnis kommen, soll in einem eigens für die Videoerstellung reservierten Raum ein Betreuer für Fragen und zur Unterstützung zur Verfügung stehen.

Zum jetzigen Zeitpunkt konzentriert sich die Arbeit jedoch nur auf die Poster, jedoch ohne Video oder dynamische Elemente. Die Poster-Applikation soll eine Beispielapplikation für die Wall aufzeigen.

Der Fragebogen zeigt klar auf, dass die meisten Personen durch andere Inhalte als Poster von der Wall angelockt werden müssen.

## V.3.9 Rollen & Personas

### V.3.9.1 Rollen

Für das Projekt ergeben sich insgesamt vier Rollen:

Dies ist zum Ersten der HSR Student, welcher sich Poster ansieht.

Die zweite Rolle ist daher die externe Person, die eine Veranstaltung an der HSR besucht: Neben dem Schulunterricht finden an der HSR auch immer wieder Veranstaltungen für externe Personen statt. Diese Besucher sind ebenfalls potenzielle Videowall Nutzer. Bei den Veranstaltungen gibt es immer Pausen. Diese Zeit kann genutzt werden, um Aussenstehenden die Arbeiten der HSR näher zu bringen und im Idealfall eine Zusammenarbeit zwischen externen Instanzen und der HSR zu fördern.

Weiter gibt es die Rolle des Studenten, der gerne eine Applikation programmieren möchte, um diese dann auf der Videowall verfügbar machen zu können.

Ferner besteht die Rolle der Sekretärin, die die Bachelorposter und andere Inhalte der Videowall verwaltet.

### V.3.9.2 Personas

Durch die Befragungen (siehe V.3.8 Befragung) lassen sich folgende Punkte als Verhaltensvariablen für die Evaluierung der verschiedenen Personas für das zu entwickelnde System übernehmen:

- Interesse an den Postern
- Wert der Präsentation
- Zeitaufwand für das Lesen
- Qualität der Poster/Broschüre

Die Verteilung der Antworten auf die oben genannten Punkte aus den Befragungen sieht wie folgt aus:

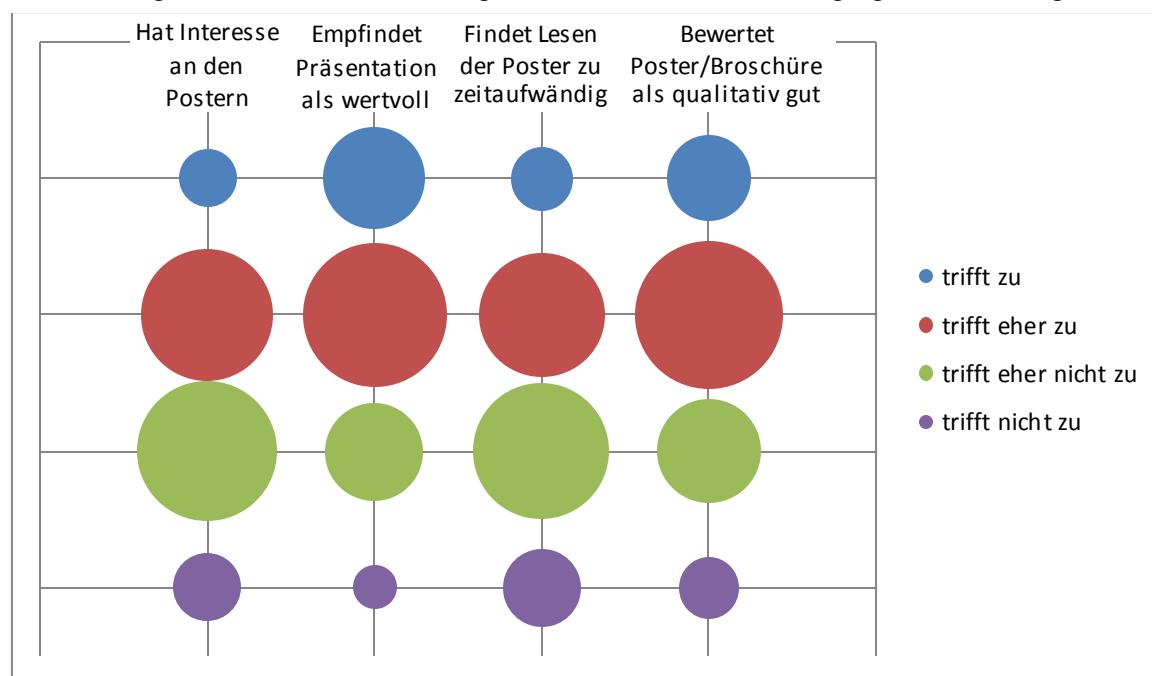
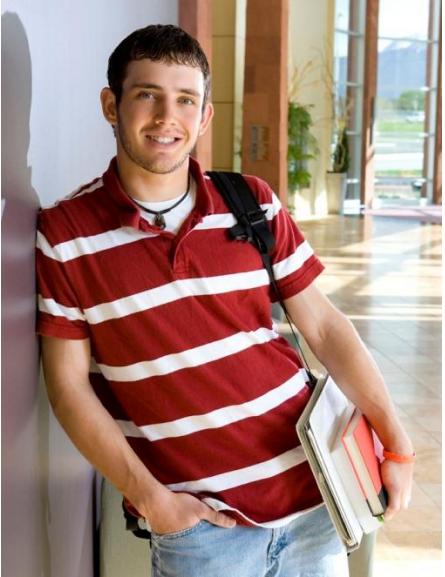


Abbildung 24 - Meinungsverteilung

Wie aus der Abbildung 24 - Meinungsverteilung ersichtlich ist, bilden sich bei jedem der vier aufgeführten Punkte zwei grosse Gruppen. Eine Gruppe bei der Antwort „trifft eher zu“ und eine andere bei „trifft eher nicht zu“. Aus diesem Grund wurden für die Rolle des Studenten, welcher Poster liest, zwei Personas ausgearbeitet. Die eine Persona interessiert sich für die Bachelorarbeiten und liest den Inhalt der Poster auch aufmerksam. Die zweite Persona schliesst Studenten ein, die sich für die auf den Postern vorgestellten Arbeiten nicht besonders begeistern können und auch den Zeitaufwand, um die Poster zu lesen, meist als zu gross empfinden.

Auch die Rolle des Eventbesuchers wurde zu einer Persona ausgearbeitet. Die drei Personas werden nachfolgend beschrieben.

#### V.3.9.2.1 Persona Peter Posterleser

Peter Posterleser		<b>Kurzprofil</b> HSR Student im 6. Semester, Studiengang Maschinenbau 25 Jahre
		<b>Abbildung 25 - Peter Posterleser, Bildquelle:</b> <a href="http://www.office.com">www.office.com</a>
Arbeitskontext (Lärm, Unterbrüche, Regeln)		Durch die für sein Semester gewählten Module befindet sich Peter mehrheitlich in den Gebäuden 1, 2, 3 und 5. Er ist ein regelmässiger Mensa-Besucher und daher auch fast jeden Tag im Gebäude 4 anzutreffen. Zu Beginn eines Semesters nimmt sich Peter in den Unterrichtspausen oder Zwischenstunden manchmal Zeit die ausgestellten Bachelorposter zu lesen.
Persönlichkeit & Vorlieben		Peter ist daran interessiert zu erfahren, was andere Personen für Arbeiten durchgeführt haben. Besonders interessieren ihn jene aus seinem eigenen Studiengang. Er liest aber auch Poster von Arbeiten aus anderen Studiengängen.
Vorkenntnisse & Lernen (Computer, Domain)		Durch sein Studium kennt sich Peter zwar gut mit Computern aus, jedoch nicht mit Kinect. Er ist aber neuen Technologien gegenüber offen und würde diese auch gerne ausprobieren.
Eigenschaften / Verhaltensvariablen		<ul style="list-style-type: none"> <li>• Hat Interesse an den Postern: Trifft zu</li> <li>• Empfindet Präsentation als wertvoll: Trifft zu</li> <li>• Bewertet Poster/Broschüre als qualitativ gut: Trifft zu</li> <li>• Findet Lesen der Poster zu zeitaufwändig: Trifft nicht zu</li> </ul>
Ziele		<ul style="list-style-type: none"> <li>• Sich über andere Arbeiten informieren</li> </ul>

#### V.3.9.2.1.1 Ist-Szenario-1

Das neue Semester hat gerade erst begonnen und Peter besucht an diesem Tag die Kunststofftechnik-Vorlesung. In der Pause geht er mit einem Freund in das Gebäude 4, um sich in der Mensa ein Brötchen zu kaufen. Zurück im ursprünglichen Gebäude bleibt Peter immer noch ein wenig Zeit bis zum Ende der Pause. Er bemerkt, dass momentan die Poster seines Studienganges in diesen Räumlichkeiten ausgestellt sind. Er nutzt daher die verbleibende Zeit um diese zu betrachten. Er entdeckt dabei ein Poster, welches er besonders spannend findet und beginnt, dieses genauer zu lesen. Kurz darauf ertönt der Pausengong und Peter geht wieder zurück in die Vorlesung.

### V.3.9.2.1.2 Soll-Szenario-1

Das neue Semester hat gerade erst begonnen und Peter besucht an diesem Tag die Kunststofftechnik-Vorlesung. In der Pause geht er mit einem Freund in das Gebäude 4, um sich in der Mensa ein Brötchen zu kaufen. Dabei fällt den beiden die grosse Monitorwand auf, welche im Eingangsbereich des Gebäudes 4 steht. Bald schon bemerken sie, dass diese auf sie reagiert, wenn sie an ihr vorbeilaufen. Interessiert kommen die beiden näher und stellen fest, dass über diese Monitorwand die Poster der Arbeiten der HSR angeschaut werden können. Die zwei interagieren mit der Wall und entdecken bald ein Poster, welches sie besonders spannend finden und beginnen, dieses zu lesen. Kurz darauf ertönt der Pausengong. Peter würde sich aber gerne noch weiter über das eben angesehene Poster informieren und fotografiert deshalb mit seinem Smartphone den abgebildeten QR-Code. Danach kehren die beiden schnell in die Vorlesung zurück.

### V.3.9.2.1.3 Ist-Szenario-2

Am Dienstag der dritten Semesterwoche, kurz vor Mittag, möchten sich Peter und eine Studienkollegin noch Poster aus anderen Studiengängen ansehen. Sie haben sich für diejenigen der Elektrotechnik entschieden. Sie gehen eigens dafür in das Gebäude 5. Dabei müssen die beiden jedoch feststellen, dass die Ausstellung der Bachelorarbeiten bereits vorüber ist und keine Poster mehr ausgestellt sind. Etwas enttäuscht kehren sie ins Gebäude 1 zurück.

### V.3.9.2.1.4 Soll-Szenario-2

Am Dienstag der dritten Semesterwoche, kurz vor Mittag, möchten sich Peter und eine Studienkollegin noch Poster aus anderen Studiengängen ansehen. Sie haben sich für diejenigen der Elektrotechnik entschieden. Peter erinnert sich an die Videowall in Gebäude 4 und sie begeben sich gemeinsam dorthin. Die beiden stellen schnell fest, dass man die Auswahl der Poster auf einzelne Studiengänge beschränken kann. Auf diese Art können sie nur in den Postern der Elektrotechnik stöbern. Die zwei stoßen nach kurzer Zeit auf einige spannende Projekte, welchen sie mehr Zeit widmen. Nach gut 20 Minuten werden sie von ihren Studienkollegen abgeholt um in der Mensa zu Essen.

### V.3.9.2.2 Persona Noemi Nichtinteressiert

Noemi Nichtinteressiert	 <p><b>Kurzprofil</b> HSR Studentin im 4. Semester, Studiengang Raumplanung 23 Jahre</p>
<b>Abbildung 26 - Noemi Nichtinteressiert,</b> <b>Bildquelle: www.office.com</b>	

Arbeitskontext (Lärm, Unterbrüche, Regeln)	Die Vorlesungen von Noemi finden alle im Gebäude 1 statt. Ihr Arbeitsraum für Projektarbeiten befindet sich ebenfalls in diesem Gebäude. Auch die Mittagszeit verbringt sie grösstenteils an diesem Ort, da sie ihr Mittagessen meistens von zu Hause mitnimmt. Nur zwischendurch besucht sie das Gebäude 4 um die Mensa zu nutzen, den Badge aufzuladen oder sich am Empfang zu informieren.
Persönlichkeit & Vorlieben	Noemi ist nicht besonders interessiert an den ausgestellten Bachelorarbeiten. Zudem ist ihr auch der Aufwand, um die Poster aufzusuchen und sie zu lesen, zu gross. Da die Poster der Abteilung Raumplanung aber im Gebäude 1 ausgestellt werden, kann sie sich dann zwischendurch aber doch durchringen, ein paar wenige Poster ihres eigenen Studiengangs zu betrachten.
Vorkenntnisse & Lernen (Computer, Domain)	Noemi besitzt grundlegende Computerkenntnisse, hat aber noch nie von Kinect gehört.
Eigenschaften / Verhaltensvariablen	<ul style="list-style-type: none"> <li>• Hat Interesse an den Postern: Trifft nicht zu</li> <li>• Empfindet Präsentation als wertvoll: Trifft nicht zu</li> <li>• Bewertet Poster/Broschüre als qualitativ gut: Trifft nicht zu</li> <li>• Findet Lesen der Poster zu zeitaufwändig: Trifft zu</li> </ul>

### V.3.9.2.2.1 Ist-Szenario 1

Noemi hält sich bei den Tischen im 1. Stock des Gebäudes 1 auf. Das neue Semester hat erst begonnen und bereits ist eine Übung ausgefallen. Da sie noch kein Projekt hat, an dem sie in den gewonnenen zwei Stunden arbeiten könnte, überlegt sie, wie sie sich die Zeit vertreiben könnten. Ihre Studienkollegen beschliessen, in der Mensa eine Kaffeepause zu machen. Auf dem Weg dorthin passieren sie das Foyer, in welchem die Bachelorarbeiten des vergangenen Semesters ausgestellt sind. Zusammen mit zwei Kolleginnen bleibt Noemi zurück und sie schauen sich zusammen mit anderen interessierten Besuchern die Poster an. Entgegen ihrer Begleiterinnen hat Noemi aber keine grosse Lust, sich über eine Arbeit genauer zu informieren und verliert bald das Interesse an der Ausstellung. So schlendert sie in Richtung Mensa und gesellt sich dort zu ihren Studienkollegen.

### V.3.9.2.2.2 Soll-Szenario 1

Noemi hält sich bei den Tischen im 1. Stock des Gebäudes 1 auf. Das neue Semester hat erst begonnen und bereits ist eine Übung ausgefallen. Da sie noch kein Projekt hat, an dem sie in den gewonnenen zwei Stunden arbeiten könnte, überlegt sie, wie sie sich die Zeit vertreiben könnte. Ihre Studienkollegen beschliessen, in der Mensa eine Kaffeepause zu machen. Im Eingang des Gebäudes 4 fällt ihnen sofort die Videowall auf. Zwei Besucher haben gerade die Benutzung der Wall beendet und verlassen diese. Sogleich übernehmen die zwei Kolleginnen von Noemi die Steuerung. Noemi bleibt ebenfalls gespannt stehen und entdeckt, dass auf der Monitorwand die Bachelorposter abgebildet werden. Angespornt durch ihre Kolleginnen bleibt sie für eine Weile dort und schaut sich die Poster mit ihnen zusammen an.

### V.3.9.2.2.3 Ist-Szenario 2

Zur Mittagszeit begibt sich Noemi zur Mensa im Gebäude 4, da sie es versäumt hat, etwas von zu Hause mitzunehmen. Im Eingangsbereich des Gebäudes lädt sie ihren Badge auf. Heute ist viel Betrieb und vor allem die rechte Warteschlange für die Standardmenus ist besonders lang. Noemi stellt sich daher in die linke Reihe, in welcher man für das Tagesmenu ansteht. Die Infokarten, auf welchen die heute angebotenen Menüs aufgelistet sind, befinden sich erst weiter vorne bei den Tablets und dem Besteck. Dort angekommen stellt sie fest, dass das Tagesmenu sogar nicht ihrem Geschmack entspricht. Daher quetscht sie sich, nicht gerade zur Freude ihrer Mitstudenten, in die rechte Warteschlange.

### V.3.9.2.2.4 Soll-Szenario 2

Zur Mittagszeit begibt sich Noemi zur Mensa im Gebäude 4, da sie es versäumt hat, etwas von Zuhause mitzunehmen. Im Eingangsbereich des Gebäudes lädt sie ihren Badge auf. Heute ist viel Betrieb und vor allem die rechte Warteschlange für die Standardmenus ist besonders lang. Während Noemi darauf wartet, dass auch ihre Studienkollegen ihren Badge aufgeladen haben, entdeckt sie, dass auf der Videowall die Menus der Mensa

angezeigt werden. Noemi stellt fest, dass das Tagesmenü so gar nicht ihrem Geschmack entspricht. Sie stellt sich daher in die rechte Warteschlange. Um sich die Zeit ein wenig zu vertreiben, schaut sie den Mitstudierenden zu, wie diese mit der Videowall interagieren.

### V.3.9.2.3 Persona Erich Eventbesucher

<b>Erich Eventbesucher</b>  <b>Abbildung 27 - Erich Eventbesucher,</b> Bildquelle: <a href="http://www.office.com">www.office.com</a>	<b>Kurzprofil</b> Mitarbeiter aus dem privaten Sektor 31 Jahre
Arbeitskontext (Lärm, Unterbrüche, Regeln)	Erich befindet sich bei Eventbesuchen an der HSR jeweils den ganzen Tag im Gebäude 4. Denn für die Veranstaltungen an der HSR wird meist die Aula genutzt, welche sich in ebendiesem Gebäude befindet. Auch das Mittagessen wird dort serviert.
Persönlichkeit & Vorlieben	Erich ist an neuen Technologien und Entdeckungen grundsätzlich interessiert.
Vorkenntnisse & Lernen (Computer, Domain)	Erich verfügt über gute Computerkenntnisse und hat schon von Kinect gehört, dies aber bis jetzt noch nicht ausprobieren können.
Eigenschaften / Verhaltensvariablen	<ul style="list-style-type: none"> <li>• Hat Interesse an den Postern: Trifft zu</li> <li>• Empfindet Präsentation als wertvoll: Trifft zu</li> <li>• Bewertet Poster/Broschüre als qualitativ gut: Trifft zu</li> <li>• Findet Lesen der Poster zu zeitaufwändig: Trifft nicht zu</li> </ul>
Ziele	<ul style="list-style-type: none"> <li>• Zeit in den Pausen überbrücken</li> </ul>

#### V.3.9.2.3.1 Ist-Szenario-1

Erich besucht zusammen mit seinen Firmenkollegen eine Veranstaltung an der HSR. Diese findet in der Aula im Gebäude 4 statt. Nach einer Einführung erfolgt die erste Pause. Erich und seine Kollegen nutzen diese Zeit, um etwas nach draussen zu gehen und frische Luft zu schnappen. Auf dem Weg dorthin haben sie noch genug Zeit, um sich im Gebäude etwas genauer umzusehen und betrachten für kurze Zeit den Informationsstand. Ihr Interesse verfliegt jedoch recht schnell, da es sich hierbei vor allem um Informationen für zukünftige Studenten oder Angebote für Studierende handelt. Die Gruppe begibt sich nach draussen und kehrt erst zurück, als sie in den Saal gerufen wird.

#### V.3.9.2.3.2 Soll-Szenario-1

Erich besucht zusammen mit seinen Firmenkollegen eine Veranstaltung an der HSR. Diese findet in der Aula im Gebäude 4 statt. Schon beim Betreten des Gebäudes fällt ihm die Videowall an der Wand im Eingangsbereich auf. Jedoch hat er keine Zeit, sich genauer damit auseinanderzusetzen, da die Veranstaltung gleich beginnt und er sich unverzüglich in die Aula begeben muss. Nach einer Einführung erfolgt die erste Pause. Erich und seine Kollegen nutzen diese Zeit, um etwas nach draussen zu gehen und frische Luft zu schnappen. Auf dem Weg dorthin fällt ihm wieder die Videowall auf. Interessiert nähert er sich dieser und bemerkt, dass diese auf seine Bewegungen reagiert. Erich stellt bei näherer Betrachtung fest, dass es sich bei den angezeigten Elementen um

Poster zu Arbeiten, die an der Hochschule durchgeführt wurden, handelt. Erich sieht sich einige der Poster an und möchte eines davon genauer studieren. In diesem Moment werden er und seine Gruppe jedoch wieder in den Saal gerufen. Erich nimmt sich jedoch fest vor, die Videowall am Mittag noch einmal über längere Zeit zu nutzen.

---

### V.3.10 Sofortiges Erfolgserlebnis

Gerätesoftware und Internetapplikationen, welche dem Benutzer ein sofortiges Erfolgserlebnis bieten, haben eine positive Wirkung auf den Nutzer. Das Erfolgserlebnis vermittelt ihm das Gefühl, dass er fähig ist, die Applikation zu bedienen: Eine Swipe-Geste auf dem Smartphone bestätigt das eigene Tun, indem unverzüglich der nächste Screen angezeigt wird. Ein Tastendruck in der Suchleiste erzeugt eine unmittelbare Antwort, indem eine Liste von möglichen Suchbegriffen angezeigt wird.

Auch die Videowall bietet ein sofortiges Erfolgserlebnis. Sobald der Benutzer von Kinect erkannt worden ist, wird das Skelett des Benutzers auf der Wall dargestellt. Bewegt sich die Person, so macht das Skelett die Bewegungen zeitgleich nach. Sofort realisiert der Nutzer, dass das Skelett ihn selbst darstellt und weiß somit, dass er die Steuerung der Applikation in der Hand hat.

---

### V.3.11 Motivation zur wiederholten Nutzung der Videowall

Ein Demomodus soll die Videowall attraktiv machen und die Aufmerksamkeit der Passanten wecken. Interessante Inhalte aus verschiedenen Themenbereichen sollen die Nutzer motivieren, mit der Wall zu interagieren.

Auf der Wall sollen daher Informationen, die für die unterschiedlichen Benutzergruppen von Interesse sind, präsentiert werden:

- Für Studenten, welche sich für Bachelorarbeiten interessieren, eignet sich das Browsen der Poster. Jedes Semester kommen hier neue Arbeiten hinzu, die Auswahl vergrößert sich also stets. Es ist denkbar, auf der Videowall Videos über die Arbeiten zu präsentieren. Das macht die Wall interaktiver und somit interessanter für eine wiederholte Nutzung.
- Für Eventbesucher ist es sicher von Interesse, an einem zentralen Ort einen Überblick über den Event erhalten zu können. Gerade technisch affine Benutzer können sich auch für das Browsen der Poster begeistern.
- Für Studenten, welche sich nicht so sehr für die Bachelorarbeiten erwärmen können, und für Besucher besteht die Möglichkeit, sich über die Videowall über das Mittagsmenu der öffentlichen Mensa der Hochschule zu informieren. Eine Erheiterung durch ein einfaches Spiel soll die Motivation der Passanten allgemein zur wiederholten Nutzung der Wall aufrecht zu erhalten.

---

### V.3.12 Microsoft Imagine Cup

Das Team entschied sich kurzfristig am Microsoft Imagine Cup<sup>23</sup> teilzunehmen. Bei diesem Wettbewerb können Teams zu verschiedenen Themen einen Projektplan und später eine Applikation präsentieren. Das Thema Kinect Fun Lab Challenge schien gut zu dem Projekt zu passen und daher wurde dafür ein Projektplan erstellt, welcher die Idee der Videowall weiter ausbaute. Der Projektplan wurde zusammen mit Kevin Gaunt vom Institut für Software (INS) erstellt und kann im Anhang eingesehen werden (VIII Anhang). Leider konnte sich das Team mit diesen Plan nicht gegen die Konkurrenz durchsetzen.

---

<sup>23</sup> <http://www.imaginecup.com>

## V.4 Anforderungen

<b>V.4.1 Änderungsgeschichte .....</b>	<b>51</b>
<b>V.4.2 Tools.....</b>	<b>52</b>
<b>V.4.3 Funktionale Anforderungen.....</b>	<b>53</b>
<b>V.4.4 Nicht-funktionale Anforderungen .....</b>	<b>60</b>
V.4.4.1 Funktionalität.....	60
V.4.4.1.1 Angemessenheit.....	60
V.4.4.2 Zuverlässigkeit.....	60
V.4.4.2.1 Reife.....	60
V.4.4.3 Benutzbarkeit.....	60
V.4.4.3.1 Verständlichkeit & Erlernbarkeit.....	60
V.4.4.3.2 Bedienbarkeit.....	60
V.4.4.3.3 Attraktivität.....	60
V.4.4.4 Effizienz.....	61
V.4.4.4.1 Zeitverhalten.....	61
V.4.4.5 Änderbarkeit & Wartbarkeit.....	61
V.4.4.6 Übertragbarkeit.....	61
V.4.4.6.1 Austauschbarkeit.....	61
V.4.4.6.2 Installierbarkeit.....	61
<b>V.4.5 Design Constraints .....</b>	<b>62</b>
<b>V.4.6 Zugänglichkeit (Accessibility).....</b>	<b>62</b>

---

### V.4.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
18.05.2012	1.0	Erste Version des Dokuments	CH
19.05.2012	1.1	Review Accessibility	DT
29.05.2012	1.2	Funktionale & Nicht-funktionale Anforderungen	CH
29.05.2012	1.3	Review Funktionale Anforderungen	DT
04.06.2012	1.4	Design Constraints	CH
04.06.2012	1.5	Nicht-funktionale Anforderungen	DT
04.06.2012	1.6	Tools	DT
05.06.2012	1.7	Tools	CH
09.06.2012	1.8	Review und Korrekturen	LE
11.06.2012	1.9	Korrekturen	CH
12.06.2012	1.10	Review	DT
14.06.2012	1.11	Review	DT

## V.4.2 Tools

Zur Durchführung der Arbeit und Entwicklung der Videowall wurden die nachfolgend aufgelisteten Werkzeuge verwendet.

Tool	Version	Link
Windows 7	SP 1	<a href="http://windows.microsoft.com/de-CH/windows/home">http://windows.microsoft.com/de-CH/windows/home</a>
Tortoise SVN	1.7.4	<a href="http://tortoisesvn.net/">http://tortoisesvn.net/</a>
Adobe Reader	X	<a href="http://get.adobe.com/de/reader/">http://get.adobe.com/de/reader/</a>
.NET	4.0.30319	<a href="http://www.microsoft.com/net">http://www.microsoft.com/net</a>
Kinect SDK	1.5	<a href="http://www.microsoft.com/en-us/kinectforwindows/">http://www.microsoft.com/en-us/kinectforwindows/</a>
Visual Studio 2010 Ultimate mit Power-Tools	10.0.40219.1 SP1Rel	<a href="http://www.microsoft.com/visualstudio/11/de-de">http://www.microsoft.com/visualstudio/11/de-de</a>
ReSharper	6.1	<a href="http://www.jetbrains.com">http://www.jetbrains.com</a>
dotCover	1.2	<a href="http://www.jetbrains.com">http://www.jetbrains.com</a>
GhostDoc	3.0	<a href="http://submain.com/download/ghostdoc/">http://submain.com/download/ghostdoc/</a>
Expression Blend	4.0.20525.0	<a href="http://www.microsoft.com/expression/products/Blend_Overview.aspx">http://www.microsoft.com/expression/products/Blend_Overview.aspx</a>
WPF Inspector	0.9.9	<a href="http://www.wpftutorial.net/Inspector.html">http://www.wpftutorial.net/Inspector.html</a>
NDepend Trial	v4.0	<a href="http://www.ndepend.com/NDependDownload.aspx">http://www.ndepend.com/NDependDownload.aspx</a>
Adobe Photoshop	CS4 Extended	<a href="http://www.adobe.com/de/products/photoshop.html">http://www.adobe.com/de/products/photoshop.html</a>
Adobe InDesign	CS4 Extended	<a href="http://www.adobe.com/de/products/indesign.html">http://www.adobe.com/de/products/indesign.html</a>
Adobe Illustrator	CS4 Extended	<a href="http://www.adobe.com/de/products/illustrator.html">http://www.adobe.com/de/products/illustrator.html</a>
Microsoft Office 2010	14.0.6112.5000	<a href="http://office.microsoft.com/de-ch/">http://office.microsoft.com/de-ch/</a>
Redmine	2.0.1	<a href="http://redmine.org">http://redmine.org</a>
Matrox PowerDesk	1.14.183.508	<a href="http://www.matrox.com/graphics/de/products/multi_display_software/powerdesk/">http://www.matrox.com/graphics/de/products/multi_display_software/powerdesk/</a>

Tabelle 10 - Tools

### V.4.3 Funktionale Anforderungen

Um die funktionalen Anforderungen möglichst effizient und trotzdem exakt zu definieren, wurden User Stories als Teil von Scrum verwendet. Nachfolgend sind die User Stories nach Sprint gruppiert. Die User Stories sind mit dem jeweiligen Sprint, in welchem sie umgesetzt wurden, gekennzeichnet. Nicht umgesetzte User Stories sind mit „U“ markiert. Weitere Details sind in der Tabelle User Stories im Anhang (VIII Anhang) zu entnehmen.

Nachfolgend eine Übersicht über die User Stories:

Legende: U -> Unplanned

Titel	User Story	Definition of Done	Sprint
<b>Poster werden angezeigt</b>	Als HSR-Besucher möchte ich die Poster über die Bachelorarbeiten lesen können, um Eindrücke über die Schule und die Möglichkeiten, die man an dieser Schule als Student hat, zu erhalten.	Ein Poster wird angezeigt und kann gelesen werden.	Bis SP7
<b>Poster browsen</b>	Als Videowall Benutzer möchte ich durch alle Poster browsen können, damit ich einen Eindruck von den Arbeiten erhalte.	Durch Drücken einer Vor- und Rückwärtstaste kann durch die Liste der Poster navigiert werden.	Bis SP7
<b>Handcursor wird dargestellt</b>	Als Videowall Benutzer möchte ich sehen, wo sich meine Hand auf der Wall befindet, damit ich fähig bin, die Wall zu bedienen.	Ein Cursor wird dargestellt und bewegt sich analog zur Hand des Benutzers.	Bis SP7
<b>Eigenes Skelett wird dargestellt</b>	Als Videowall Benutzer möchte ich in Form eines Skeletts sehen, dass die Videowall mich erkannt hat, damit ich nachverfolgen kann, wie das System auf meine Bewegungen reagiert.	In der Applikation wird das Skelett des Benutzers dargestellt, das sich zu den Bewegungen des Benutzers dynamisch anpasst.	Bis SP7
<b>Sofortiges Erfolgserlebnis für Einstieg sichergestellt</b>	Als neuer Videowall Benutzer möchte ich sofort ein Erfolgserlebnis erleben, weil ich einen positiven ersten Eindruck haben möchte.	Das sofortige Erfolgserlebnis (Skelett wird dargestellt) ist dokumentiert.	Bis SP7
<b>Handcursor schön dargestellt</b>	Als Videowall Benutzer möchte ich als Cursor eine Hand sehen, damit mir sofort klar ist, dass ich die Applikation mit der Hand steuern kann.	Der Cursor wird als kleine Hand dargestellt (analog XBox-Spiele).	SP8
<b>Skelett schön dargestellt</b>	Als Videowall Benutzer möchte ich ein Skelett mit Verbindungslien der einzelnen Punkte (Hand, Ellbogen, Hüfte etc.) sehen können, damit ich meine Bewegungen besser nachvollziehen kann.	Vom Benutzer wird ein Skelett mit Verbindungslien dargestellt.	SP8
<b>Handcursor ruckelt weniger 1</b>	Als Videowall Benutzer möchte ich, dass der Cursor weniger ruckelt, damit ich besser auf die Buttons drücken kann.	Da der Input des Skeletal Trackings ungenau ist und über einen etwas hohen Jitter verfügt, wird für das Hand Tracking jeweils der Mittelwert der letzten paar Aufzeichnungen verwendet. Somit ruckelt die Hand dann weniger.	SP8
<b>Video wird dargestellt</b>	Als Entwickler möchte ich testen, dass Videos auf der Videowall laufen, damit ich weiß, dass dies problemlos funktioniert und später bei Bedarf Videos eingebunden werden können.	Ein Video kann mittels einer WPF-Applikation abgespielt werden und auf der Test Hardware/Mitsubishi Wall getestet werden.	SP9

Titel	User Story	Definition of Done	Sprint
<b>Applikation ist mit linker Hand bedienbar</b>	Als Videowall Benutzer möchte ich die Applikation auch mit der linken Hand bedienen können, damit mir die Bedienung leichter fällt.	Die Applikation soll automatisch erkennen, wenn die linke Hand höher als die rechte Hand gehalten wird, und dann die Kontrolle an die linke Hand übergeben. Wird die rechte Hand dann wieder merklich höher als die linke Hand gehalten, so soll die Applikation wieder mit der rechten Hand bedient werden können.	SP9
<b>Ideen gesammelt wie Personen werden von Videowall angezogen werden</b>	Als Passant möchte ich beim Vorbeigehen an der Videowall etwas Interessantes sehen, damit mein Interesse geweckt wird und ich auf die Videowall zugehe und mit ihr interagiere.	Es werden mindestens zwei verschiedene Ideen zusammengetragen, diskutiert und ausgewertet. Am Schluss wird die beste Idee ausgewählt und dokumentiert.	SP10
<b>Plug-in Möglichkeit</b>	Als Student möchte ich eine App, das mit den vorgegebenen Schnittstellen funktioniert, für die Videowall schreiben, damit ich es später auf die Videowall hochladen kann.	In der Videowall-Applikation ist es möglich, dynamisch ein Plug-in zu laden und anzuzeigen. Für jedes geladene Plug-in wird ein Menubutton angezeigt. Dynamisch laden heisst, dass ein Plug-in (.dll) in den Ordner, in dem die Applikation gerade läuft, hineinkopiert werden kann und das Plug-in dann von der Applikation automatisch geladen wird.	SP10
<b>Mittagsmenu App in Plug-in umgewandelt</b>	Als Plug-in Entwickler möchte ich ein Beispiel für ein Plug-in ansehen können, damit ich weiß, wie ich vorgehen muss, wenn ich ein Plug-in entwickeln möchte.	Das Mittagsmenu Plug-in ist in einem eigenen Projekt und die Videowall-Applikation hat keine Referenz zu diesem Projekt. Das Plug-in (Mittagsmenu) wird von der Videowall App aber automatisch hineingeladen, siehe "Plug-in Möglichkeit".	SP11
<b>Poster App in Plug-in umgewandelt</b>	Als Plug-in Entwickler möchte ich ein Beispiel für ein Plug-in ansehen können, damit ich weiß, wie ich vorgehen muss, wenn ich ein Plug-in entwickeln möchte.	Das Poster Plug-in ist in einem eigenen Projekt und die Videowall-Applikation hat keine Referenz zu diesem Projekt. Das Plug-in (Poster) wird von der Videowall App aber automatisch hineingeladen, siehe "Plug-in Möglichkeit".	SP11
<b>Demomodus (Verfolgung von Passanten) Kraftfeld besprochen und dokumentiert</b>	Als Videowall Benutzer möchte ich einen Demomodus sehen, der meine Aufmerksamkeit auf die Wall zieht und ich so beginne, mit ihr zu interagieren.	Die ausgesuchte Idee mit dem Kraftfeld wurde im Team besprochen und dokumentiert.	SP11
<b>Demomodus: Vom Demomodus wird in den Interaktionsmodus gewechselt</b>	Als Videowall Benutzer möchte ich, dass die Applikation vom Demomodus in den Interaktionsmodus wechselt, nachdem ich eine Zeit lang (z.B. 5 Sekunden) gewartet habe, damit ich mit der Applikation interagieren kann.	Die Applikation wechselt, nachdem für 5 Sekunden ein Skelett erkannt wurde, in den Interaktionsmodus. Es darf kein "Flackern" auftreten.	SP11

<b>Titel</b>	<b>User Story</b>	<b>Definition of Done</b>	<b>Sprint</b>
<b>Demomodus: Vom Interaktionsmodus wird in den Demomodus gewechselt</b>	Als Videowall Benutzer möchte ich, dass die Applikation vom Interaktionsmodus in den Demomodus wechselt, nachdem eine Zeit lang (z.B. 10 Sekunden) kein Skelett erkannt wurde, damit ich durch den Demomodus von der Videowall angezogen werde.	Die Applikation wechselt, nachdem für 10 Sekunden kein Skelett erkannt wurde, automatisch in den Demomodus. Es darf kein "Flackern" auftreten.	SP11
<b>Demomodus: externes Design erstellt</b>	Als Videowall Benutzer möchte ich einen ansprechenden Demomodus, damit mich die Videowall optisch anspricht und ich mit dieser interagieren will.	Das in WPF umgesetzte externe Design wurde vom Auftraggeber nach einer Demonstration abgenommen.	SP11
<b>Bild der Hand ist auf die rechte bzw. linke Hand abgestimmt</b>	Als Videowall Benutzer möchte ich für meine Hand das passende Handsymbol sehen, damit das Symbol die Hand zeigt, mit der die Bedienung gerade stattfindet.	Ist die rechte Hand aktiv, so wird das Symbol der rechten Hand angezeigt. Sobald die Bedienung zur anderen Hand wechselt, so wechselt auch das Symbol.	SP11
<b>Deployment Entwickler PC</b>	Als Entwickler möchte ich die entwickelte Applikation auf dem lokalen PC öffnen können, damit die Applikation weiterentwickelt werden kann.	Der Entwickler kann die Solution öffnen und die Applikation ausführen, ohne dass diese abstürzt.	SP12
<b>Plug-in Schnittstelle definiert</b>	Als Entwickler möchte ich eine Plug-in Schnittstelle definieren, damit Interessierte Apps für meine Applikation erstellen können.	Das Interface oder die Interfaces für das Plug-in System sind implementiert.	SP12
<b>Demomodus: Demotext zu aktiver App wird angezeigt</b>	Als Videowall Benutzer möchte ich beim Demomodus einen ansprechenden und interessanten Text sehen, damit dieser mein Interesse weckt und ich mit der Videowall interagieren will.	Für jede App existiert ein Demotext (durch das Interface IApp), der jeweils im Demomodus durch ein Binding auf die aktuelle App angezeigt wird.	SP12
<b>Demomodus: Apps werden automatisch gewechselt</b>	Als Videowall Benutzer möchte ich Texte von verschiedenen Apps sehen, damit ich die App wählen kann, die mich am meisten anspricht.	Während dem Demomodus wird alle 20 Sekunden die aktive Applikation gewechselt, wodurch auch automatisch der Demotext gewechselt wird.	SP12
<b>Navigation mit schönen "Tabs" dargestellt</b>	Als Entwickler möchte ich dem Benutzer die Navigation zwischen den Ansichten mit Tabs ermöglichen, damit auf einen Blick ersichtlich ist, welche Ansichten/Informationen die Wall zur Verfügung stellt und wie von der einen in die andere gewechselt werden kann.	Externes Design ist implementiert.	SP12
<b>Das Mittagsmenu wird angezeigt</b>	Als Mensabesucher möchte ich gerne das aktuelle Mittagsmenu sehen, damit ich dies nicht erst in der Mensa am kleinen Tisch tun muss.	Das Mittagsmenu wird in einem gut lesbaren Format angezeigt.	SP12
<b>Mittagsmenu App automatisch aktualisiert</b>	Das Mittagsmenu soll automatisch aktualisiert werden.	Das aktuelle Menu wird von <a href="http://hochschule-rapperswil.sv-group.ch/de/menuplan.html">http://hochschule-rapperswil.sv-group.ch/de/menuplan.html</a> geladen, sobald die Videowall-Applikation gestartet wird.	SP12
<b>Administration der Videowall Inhalte definiert</b>	Als Betreiber des Systems möchte ich, dass ein möglicher Ablauf für die Administration (Erstellung, Löschung, Mutationen) definiert ist, damit ich weiß, was für Systeme dazu noch entwickelt werden müssen.	Es liegt eine Dokumentation vor, wie die Daten der Videowall bearbeitet werden können.	SP13

<b>Titel</b>	<b>User Story</b>	<b>Definition of Done</b>	<b>Sprint</b>
<b>Externes Design festgelegt und validiert</b>	Als Entwickler möchte ich für die Design User Stories eine "Definition of Done" festlegen können, damit der Abschluss der User Stories validiert werden kann.	Externes Design ist festgelegt und validiert.	SP14
<b>Deployment Videowall Server</b>	Als Entwickler möchte ich die entwickelte Applikation auf den Videowall Server deployen können, damit die Applikation dann darauf läuft.		U
<b>Deployment Web</b>	Als Entwickler möchte ich die entwickelte Applikation auf den Web Server deployen können, damit die Applikation dann darauf läuft.		U
<b>Sekretärin kann Poster verwalten</b>	Als Sekretärin möchte ich die Poster verwalten können, damit die dargestellten Poster auf der Wall immer auf dem aktuellsten Stand sind.		U
<b>About View</b>	Als Bewunderer der Applikation möchte ich sehen, wer diese Applikation unter der Leitung von wem entwickelt hat, weil mich interessiert, bei welchem Dozent eine so coole Arbeit realisiert werden kann.		U
<b>Kinect Hand hervorgehoben</b>	Als Videowall Benutzer möchte ich, dass ich meine aktive Hand sehe, damit ich weiss, dass ich die Applikation mit dieser Hand bedienen kann.		U
<b>Handcursor dreht nicht bei aktivem Menu</b>	Als Videowall Benutzer möchte ich, dass keine Handcursor Animation bei dem zurzeit aktiven Menu abgespielt wird, damit ich nicht meine, dass noch etwas passiert.		U
<b>Am Handcursor soll erkennbar sein, ob man auf einem interaktiven Objekt ist</b>	Als Videowall Benutzer möchte ich, dass der Handcursor, je nachdem, ob er sich über einem interaktiven Objekt (z.B. Button) befindet oder nicht, anders gekennzeichnet wird, damit ich weiss, wann ein 'Klicken' möglich ist.		U
<b>Poster filtern nach Studiengang</b>	Als Videowall Benutzer möchte ich die Poster nach Studiengang filtern, damit ich nur die Poster meiner Abteilung browsen kann.		U
<b>News anzeigen</b>	Als Videowall Benutzer möchte ich News/Headlines über das Thema meines Studiengangs sehen, da mich dies interessiert.		U
<b>Sekretärin kann News Feeds der Wall verwalten</b>	Als Sekretärin möchte ich die News Feeds der Wall verwalten können, damit die News Feeds auf dem aktuellsten Stand sind.		U

<b>Titel</b>	<b>User Story</b>	<b>Definition of Done</b>	<b>Sprint</b>
<b>News mit QR-Code</b>	Als Videowall Benutzer möchte ich neben den News für meinen Studiengang einen QR-Code sehen, den ich abfotografieren kann, damit ich später auf dem Computer nochmals die gleichen Headlines sehen und mich mittels Internetrecherche in die Themen vertiefen kann.		U
<b>Studenten können App hochladen</b>	Als Student möchte ich meine selbst entwickelte App hochladen können, damit sie dann auf der Videowall verfügbar ist.		U
<b>Plug-in Dokumentation geschrieben</b>	Als App Entwickler möchte ich eine Dokumentation zur Verfügung haben, damit ich nachschauen kann, wie ich beim Programmieren einer App vorgehen muss.		U
<b>Informationen zu aktuellen Events werden auf der Videowall angezeigt</b>	Als Eventbesucher möchte ich auf der Videowall aktuelle Informationen zum Event angezeigt haben, damit ich mich dort informieren kann.		U
<b>Sekretärin kann Informationen zu Events verwalten</b>	Als Sekretärin möchte ich Informationen zu aktuellen Events hochladen, damit sich die Eventbesucher auf der Wall informieren können.		U
<b>Browsing Modus schön dargestellt</b>	Als Videowall Benutzer möchte ich eine ansprechende Darstellung des Browsing Modus, damit mir die Applikation besser gefällt und mir die Bedienung mehr Spass macht.		U
<b>Demomodus schön dargestellt</b>	Als Videowall Benutzer möchte ich eine ansprechende Darstellung des Demomodus, damit mich dieser eher anspricht und ich die Applikation verwenden möchte.		U
<b>Animationen bei Browsing Modus</b>	Als Videowall Benutzer möchte ich eine Animation sehen können, wenn ich zum nächsten Poster navigiere, damit es für mich besser ersichtlich ist, dass das Poster gewechselt hat.		U
<b>Animation bei Tab/Ansichten-Wechsel</b>	Als Videowall Benutzer möchte ich eine Animation sehen können, wenn ich ein anderes Tab wähle, damit es für mich besser ersichtlich ist, dass die Ansicht gewechselt hat.		U
<b>Avatar wird dargestellt</b>	Als Videowall Benutzer möchte ich anstelle eines Skeletts einen Avatar sehen, damit ich sofort begreife, dass ich als Mensch erkannt bin.		U
<b>Algorithmus zur Poster Sortierung</b>	Als Videowall Benutzer möchte ich, dass aktuelle Poster öfter vorkommen, da mich diese mehr interessieren.		U

<b>Titel</b>	<b>User Story</b>	<b>Definition of Done</b>	<b>Sprint</b>
<b>QR-Code wird pro Poster dargestellt</b>	Als Videowall Benutzer möchte ich einen QR Code fotografieren, damit ich die Informationen von der Videowall auf meinem Mobiltelefon mitnehmen kann.		U
<b>L Poster Problematik</b>	Als Videowall Benutzer möchte ich die L-Poster lesen können, damit ich mich auch über die Arbeiten dieser Studienabteilung informieren kann.		U
<b>Easteregg</b>	Als Entwickler möchte ich beim Präsentieren der Wall ein Easteregg ausführen, damit meine Besucher und ich Spass haben.		U
<b>Poster und Informationen sind auf einer Website verfügbar</b>	Als Videowall Benutzer möchte ich Zusatzinformationen zu den Postern auf einer Website ansehen können, damit ich mithilfe eines fotografierten QR Codes (User Story: QR-Code wird pro Poster dargestellt) weitere Informationen zum Poster erhalte.		U
<b>Sekretärin kann Zeitschlüsse für Inhalte definieren</b>	Als Sekretärin möchte ich zu verschiedenen Zeiten verschiedene Informationen auf der Videowall anzeigen.		U
<b>Handcursor ruckelt weniger 2</b>	Als Videowall Benutzer möchte ich, dass der Cursor noch weniger ruckelt, damit ich besser auf die Buttons drücken kann.		U
<b>Die Seetemperatur wird angezeigt</b>	Als Student sehe ich die Seetemperatur, weil ich wissen möchte, wie angenehm der Sprung in den See sein wird.		U
<b>Das Wetter wird angezeigt</b>	Als Besucher sehe ich das aktuelle Wetter und das der nächsten zwei Tage, weil mich das derzeitige und zukünftige Wetter interessiert.		U
<b>Sonneneinstrahlung wird angezeigt</b>	Als gesundheitsbewusste Person möchte ich die Sonneneinstrahlung sehen, damit ich weiß, ob ich einen Hut anziehen muss.		U
<b>Gesten können erkannt werden</b>	Als Benutzer der Videowall möchte ich die Applikation mit Gesten bedienen können, damit so eine alternative Steuerung zum Handcursor existiert.		U
<b>Zur Entspannung kann ein Spiel gespielt werden</b>	Als HSR-Besucher möchte ich auch mal keine Informationen erhalten, sondern ein Spiel spielen, damit ich abschalten kann.		U
<b>Kinect Winkel wird automatisch ausgerichtet</b>	Als Betreiber des Systems möchte ich, dass die Kinect automatisch auf einen festen Winkel eingestellt wird, den ich in einer Konfigurationsdatei verändern kann, damit die Personen gut von der Kinect erkannt werden können.		U
<b>Steuerung mittels Spracherkennung ist möglich</b>	Als fauler Video-Wall-Nutzer möchte ich mittels Sprachbefehlen navigieren, weil ich zu träge bin, um mich zu bewegen.		U

<b>Titel</b>	<b>User Story</b>	<b>Definition of Done</b>	<b>Sprint</b>
<b>Lesemodus Poster</b>	Als Videowall Benutzer möchte ich nach der Auswahl eines Postertitels in den Lesemodus wechseln, damit ich das Poster besser lesen kann.		U
<b>Rating</b>	Als Videowall Benutzer möchte ich die gelesenen Poster raten, damit zukünftige Leser sehen, welche Poster ich interessant fand.		U
<b>Kommunikation mit anderen Videowall Interagierenden</b>	Als Videowall Benutzer möchte ich direkt mit anderen Videowall Benutzern (geografisch getrennt) kommunizieren können.		U
<b>Zweiter Mitspieler hat Pointer</b>	Als zweiter Mitspieler möchte ich, dass meine Hand als Pointer dargestellt wird, damit ich meine Mitspieler auf etwas hinweisen kann.		U
<b>Interaktive Hilfe wird angezeigt</b>	Als Videowall Benutzer möchte ich eine interaktive Hilfe sehen, die mir zeigen kann, was für Gesten ich in bestimmten Teilen der Applikation nutzen kann, damit ich nicht frustriert von der Wall davonlaufen muss.		U
<b>Skelett ruckelt nicht</b>	Als Videowall Benutzer möchte ich, dass das Skelett nicht ruckelt, damit ich vom Ruckeln nicht irritiert/abgelenkt werde.		U
<b>Navigationshilfe</b>	Als Videowall Benutzer möchte ich, nachdem ich mich für längere Zeit unschlüssig mit dem Cursor am selben Ort befinde, die Objekte, mit welchen interagiert werden kann, aufleuchten sehen.		U

Tabelle 11 - User Stories

---

## V.4.4 Nicht-funktionale Anforderungen

Die nichtfunktionalen Anforderungen lassen sich zum Teil aus den User Stories ableiten. Einige Anforderungen, wie z.B. die Wartbarkeit, können jedoch nicht daraus abgeleitet werden. Deshalb ist es notwendig, diese festzuhalten.

---

### V.4.4.1 Funktionalität

#### V.4.4.1.1 Angemessenheit

Die Videowall soll für alle Passanten einfach bedienbar sein. Die Inhalte sollen interessant sein und für jede Benutzergruppe etwas bieten. Die Wall soll die Aufmerksamkeit der Passanten wecken und die Nutzer durch attraktive und aktuelle Inhalte zur erneuten Nutzung der Videowall animieren.

Mit Usability Tests soll geprüft werden, ob die Applikation die Aufmerksamkeit der Passanten erlangen kann.

---

### V.4.4.2 Zuverlässigkeit

#### V.4.4.2.1 Reife

Obwohl es sich die Videowall-Applikation um einen Prototyp handelt, werden gewisse Stabilitätsanforderungen an die Wall gestellt. Die Videowall soll 24 Stunden am Stück in Betrieb sein können, ohne dass die Applikation terminiert oder gravierende Memory Leaks entstehen, die zu einem Absturz führen könnten.

Dies ist durch einen Stabilitätstest zu belegen.

---

### V.4.4.3 Benutzbarkeit

#### V.4.4.3.1 Verständlichkeit & Erlernbarkeit

Die Applikation muss beim ersten Kontakt sogleich verständlich sein. Ansonsten verliert der Nutzer schnell das Interesse an der Videowall. Er wird diese verlassen und sie auch zu einem späteren Zeitpunkt nicht erneut nutzen wollen. Die Bedienung muss intuitiv sein, damit der Nutzer nicht zuerst ein Handbuch lesen muss.

Die schnelle Verständlichkeit soll mit Usability Tests validiert werden.

---

#### V.4.4.3.2 Bedienbarkeit

Die Bedienung soll einfach und intuitiv sein und über die Bewegung der Hand geschehen. Die einzelnen Komponenten sollen über eine genügend grosse Fläche verfügen, sodass sie mit dem Handcursor treffsicher ausgewählt werden können.

Die Bedienung mit der Hand soll durch einen Usability Test verifiziert werden.

---

#### V.4.4.3.3 Attraktivität

Der Entwicklungsprozess ist so gestaltet, dass neben den Risiken auch die Benutzerbedürfnisse im Fokus stehen. Diese Bedürfnisse sollen mit Hilfe von Befragungen und Usability Tests erkannt werden.

Durch einen Demomodus soll der Nutzer auf die Applikation aufmerksam gemacht und angelockt werden. Die Darstellung des eigenen Skeletts soll auf den Nutzer ansprechend wirken und ihn dazu animieren, herauszufinden, wie er die Applikation mit seinem eigenen Körper steuern kann.

Das Skelett soll das sofortige Erfolgserlebnis für den Nutzer sicherstellen. Dies sowie die Attraktivität des Demomodus werden durch Usability Tests geprüft.

---

#### V.4.4.4 Effizienz

##### V.4.4.4.1 Zeitverhalten

Die Applikation soll innerhalb fünf Minuten aufgestartet sein. Das Handtracking soll innerhalb einer Sekunde auf Benutzereingaben reagieren.

Wurde ein Nutzer erkannt und bewegt dieser sich vor der Videowall, so soll die Reaktion des dargestellten Skeletts und des Handursors auf diese Bewegungen so schnell erfolgen, dass der Nutzer das Gefühl der direkten Manipulation erhält.

Dies ist mit Usability Tests zu verifizieren.

---

#### V.4.4.5 Änderbarkeit & Wartbarkeit

Die Software wird zukünftig vom Institut für Software (IFS) weiterentwickelt. Damit dies einfach geschehen kann, soll auf die Codequalität geachtet werden, wobei ReSharper genutzt wird, um die Qualität zu prüfen.

Auch die Code-Metriken sollen beachtet werden. Ziel ist es, einen „Maintainability Index“ von mindestens 50% zu erreichen, dies auf Ebene Projekt.

Es soll eine einfache Möglichkeit geben, die Videowall dynamisch mit Inhalten zu erweitern. Um dies realisieren zu können, soll ein Plug-in System entwickelt werden.

---

#### V.4.4.6 Übertragbarkeit

##### V.4.4.6.1 Austauschbarkeit

Indem mit dem vorgegebenen Interface gearbeitet wird, können Applikationen für die Videowall unabhängig entwickelt werden. Das Plug-in System ermöglicht das dynamische Hinzufügen von Inhalten.

---

##### V.4.4.6.2 Installierbarkeit

Um das Projekt weiterentwickeln zu können, soll es möglich sein, die Applikation auf dem lokalen Computer zu öffnen und auszuführen, ohne dass diese abstürzt.

#### V.4.5 Design Constraints

Kinect ist durch die Aufgabenstellung als Inputgerät festgelegt. Es ist auch festgelegt, dass eine Monitorwand zu verwenden ist, und kein Beamer. Das Corporate Design der HSR gibt Richtlinien für das externe Design vor.

#### V.4.6 Zugänglichkeit (Accessibility)

Der eigene Körper dient für die Videowall als Steuerelement. Für Personen mit einer körperlichen Behinderung ist die Applikation daher bedingt geeignet. Durch das Wizard of Oz - Experiment (siehe V.8.2.1 Test 1: Wizard of Oz) wurde das Konzept „Meine Hand ist die Maus“ bestätigt. Sofern der Nutzer einen Arm hat, ist die Bedienung daher gewährleistet. Es kann hierbei jedoch passieren, dass gewisse Punkte des Skeletts, welche für die Komplettierung des Skeletts fehlen, fehlinterpretiert werden. Hält man sich beispielsweise einen Arm hinter den Rücken, so können die Punkte des Ellbogens und der Hand nicht mehr erkannt werden und werden hier im Beispiel entlang der Hüfte angezeigt (siehe gelbe Punkte in Abbildung 28 - Handerkennung bei Arm hinter dem Rücken).

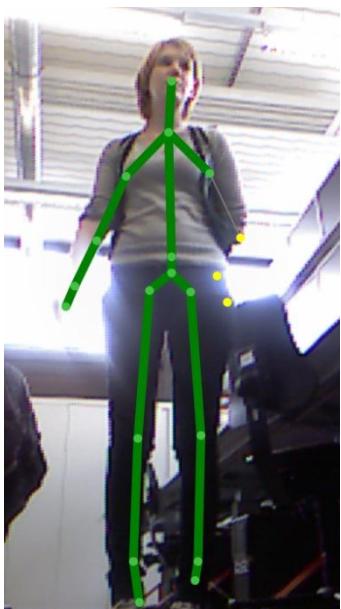


Abbildung 28 - Handerkennung bei Arm hinter dem Rücken

Die Applikation ist nicht auf Personen mit einer Sehbehinderung ausgelegt. Um diesen Personen eine optimale Bedienung der Videowall bieten zu können, müssten die Interaktionen mit akustischen Signalen beantwortet werden. Zu diesem Stand der Entwicklung sind Musik oder andere akustische Signale ausgeschlossen. Diese würden für Mitarbeiter, welche im Gebäude 4 arbeiten, störend wirken. Weiter kann die Applikation mit beliebig gestalteten, von interessierten Programmierern erstellten Applikationen erweitert werden. Es ist nicht gewährleistet, dass diese Erweiterungen für Personen mit partiell-funktionalen Sehbehinderungen optimiert sind.

## V.5 Domain Analyse

<b>V.5.1 Änderungsgeschichte .....</b>	<b>64</b>
<b>V.5.2 Systemübersicht .....</b>	<b>65</b>
V.5.2.1 Videowall mit Kinect.....	65
V.5.2.2 Service Server mit Datenbank.....	65
V.5.2.3 Webserver .....	66
V.5.2.4 Mobiltelefon.....	66
V.5.2.5 Sekretariats-Computer .....	66
<b>V.5.3 Daten.....</b>	<b>67</b>
V.5.3.1 Prozessdiagramm.....	67
V.5.3.1.1 Ist-Prozess.....	67
V.5.3.1.2 Soll-Prozess.....	67
V.5.3.2 Domain Models .....	68
V.5.3.2.1 Framework VideoWall.....	68
V.5.3.2.2 PosterApplication.....	69
V.5.3.2.3 LunchMenuApplication.....	69
V.5.3.3 Verfügbarkeit der Daten .....	70
V.5.3.3.1 Poster .....	70
V.5.3.3.2 Mittagsmenu .....	70
<b>V.5.4 Graphical User Interface (GUI) .....</b>	<b>71</b>
V.5.4.1 Empirischer formativer Test zur Eruierung der Navigationsart.....	71
V.5.4.1.1 Ideensammlung.....	71
V.5.4.1.2 Ausarbeitung.....	73
V.5.4.1.3 Umsetzung .....	74
V.5.4.1.4 Durchführung & Fazit.....	75
V.5.4.2 Demomodus .....	76
V.5.4.2.1 Ideensammlung und Besprechung.....	76
V.5.4.2.2 Auswahl der besten Idee für den Demomodus .....	80
V.5.4.3 Screen Map .....	81
V.5.4.4 Design Entscheide.....	81
V.5.4.4.1 Steuerung mit der Hand.....	81
V.5.4.4.2 Demomodus .....	81
V.5.4.4.3 Menu mit Tabs .....	82
V.5.4.4.4 Corporate Design HSR .....	82
V.5.4.5 Externes Design .....	82
V.5.4.5.1 Videowall-Applikation .....	82
V.5.4.5.2 Poster-Applikation (Plug-in).....	83
V.5.4.5.3 Mittagsmenu-Applikation (Plug-in).....	84

V.5.4.5.4	Demomodus .....	84
V.5.4.6	Guidelines.....	86
V.5.4.6.1	Human Interface Guidelines.....	86
V.5.4.6.1.1	Best Practices for Designing Interactions.....	86
V.5.4.6.1.2	Basic Interactions .....	86
V.5.4.6.1.3	Distance-Dependent Interactions.....	87
V.5.4.6.1.4	Multimodality.....	87
V.5.4.6.1.5	Multiple Users.....	87

---

## V.5.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
19.03.2012	1.0	Erste Version des Dokuments	DT
20.03.2012	1.1	Review	CH
26.03.2012	1.2	Durchführung und Resultathinzugefügt	LE
27.03.2012	1.3	Review Test-Durchführung	CH
27.03.2012	1.4	Review, Test-Resultat	DT
02.04.2012	1.5	Review	CH
04.05.2012	1.6	Ideensammlung Demomodus	DT
07.05.2012	1.7	Review Demomodus	CH
09.05.2012	1.8	Umsetzung Demomodus Kraftfeld	DT
15.05.2012	1.9	Externes Design	CH
15.05.2012	1.10	Review Externes Design	DT
15.05.2012	1.11	Umsetzung Demomodus Teaser	DT
22.05.2012	1.12	Review Umsetzung Demomodus Teaser	CH
22.05.2012	1.13	Domain Models	CH
23.05.2012	1.14	Verfügbarkeit Daten	CH
24.05.2012	1.15	Review Korrektur Markus Stolze	DT
25.05.2012	1.16	Administration der Videowall hinzugefügt	LE
29.05.2012	1.17	Screen Map	CH
04.06.2012	1.18	Review Screen Map	DT
04.06.2012	1.19	Guidelines, Design Entscheide	CH
04.06.2012	1.20	Ergänzung Kinect Bedienung	DT
04.06.2012	1.21	Demomodus Zustandsdiagramm angepasst und beschrieben	LE
05.06.2012	1.22	Review Guidelines	DT
06.06.2012	1.23	Prozessdiagramm	CH
08.06.2012	1.24	Review Design Entscheide, CI/CD Absprache mit HSR	DT
09.06.2012	1.25	Review und Korrekturen	LE
10.06.2012	1.26	Review Systemübersicht	CH
10.06.2012	1.27	Review Poster-Prozess	DT
11.06.2012	1.28	Korrekturen Markus Stolze übernommen	DT
13.06.2012	1.29	Review	DT

## V.5.2 Systemübersicht

Das System ist in mehrere Teile unterteilt, es handelt sich dabei um die folgenden:

- HSR Videowall mit Kinect
- Service Server mit Datenbank
- Webserver
- Mobiltelefon
- Sekretariats-Computer

Dieser Aufbau entspricht dem gewünschten Endsystem. Der in dieser Bachelorarbeit effektiv umgesetzte Teil findet sich im Kapitel V.6.5.1 Physische Sicht.

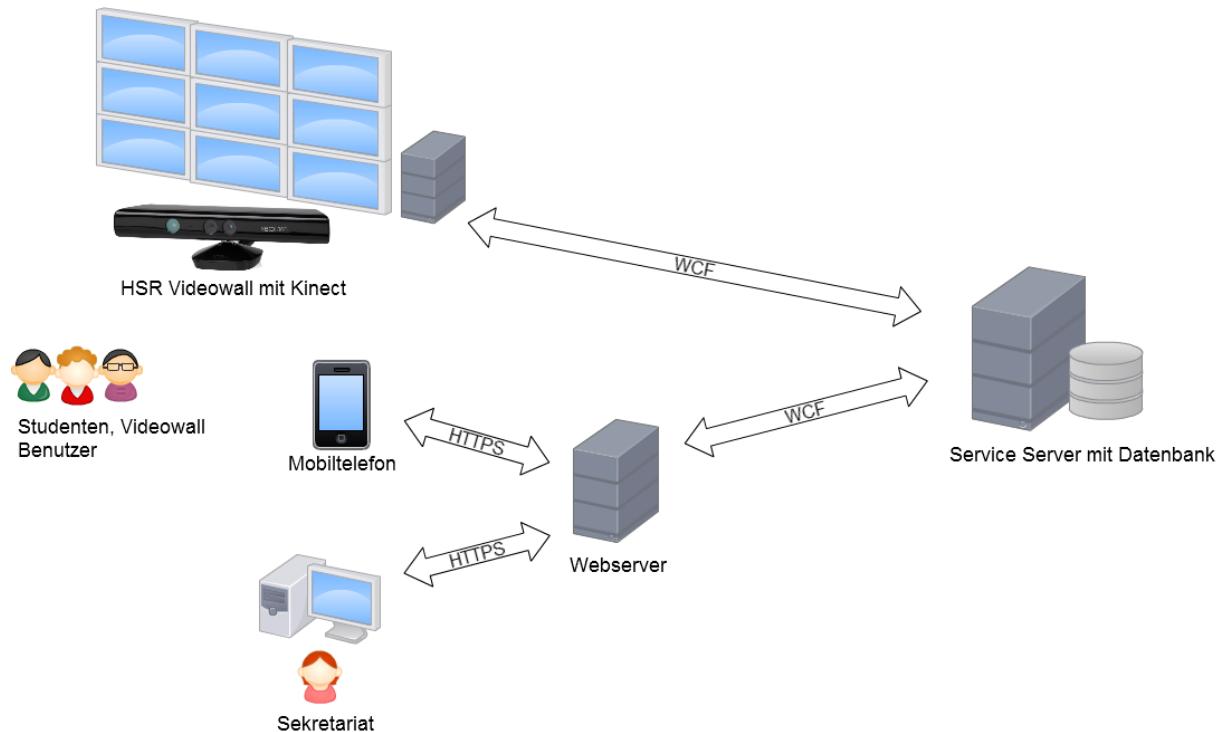


Abbildung 29 - Systemübersicht, gewünschtes Endsystem

### V.5.2.1 Videowall mit Kinect

Über die Videowall können sich Nutzer über verschiedene Themen (beispielsweise Poster von Bachelorarbeiten, das Mittagsmenü der Mensa oder das Wetter) informieren. In Zukunft sind auch Optionen wie z.B. Minispiele denkbar. Die Wall wird mittels Kinect gesteuert. Die dafür benötigten Daten werden durch WCF vom Service Server geladen.

### V.5.2.2 Service Server mit Datenbank

Auf dem Service Server werden die verschiedenen Daten, welche die Videowall benötigt, abgelegt. Diese können mittels WCF über den Webserver verwaltet oder auf der Videowall angezeigt werden.

Dieser Server könnte auch in die Cloud (Windows Azure<sup>24</sup>) ausgelagert werden, falls für die Daten kein eigener Server betrieben werden soll.

<sup>24</sup> <https://www.windowsazure.com/>

### V.5.2.3 Webserver

Der Webserver bietet einerseits eine Administrationsoberfläche für das Sekretariat an, um die Daten verwalten zu können. Dies aber nur, sofern die Daten nicht direkt in der Typo3 Datenbank des HSR Webauftritts<sup>25</sup> verwaltet werden (siehe auch V.9.2.3.1.1 Administration über Typo3 CMS).

Andererseits können auf dem Webserver per Mobiltelefon spezifische Informationen zu den auf der Wall dargestellten Daten abgerufen werden.

Beide Anforderungen sind einfach über einen Webserver realisierbar, es muss so keine zusätzliche Applikation auf den Zielgeräten installiert werden. Beide Verbindungen basieren auf dem Protokoll HTTPS. Die durch das Sekretariat getätigten Änderungen werden vom Webserver aus mittels WCF an den Service Server weitergeleitet.

### V.5.2.4 Mobiltelefon

Über den Browser des Mobiltelefons können spezifische Informationen zu den visualisierten Daten der Videowall abgerufen werden.

### V.5.2.5 Sekretariats-Computer

Die Administrationsoberfläche kann über den Browser eines Sekretariats-Computers aufgerufen werden. Über diese können die Daten der Videowall verwaltet werden.

---

<sup>25</sup> <http://www.hsr.ch/>

### V.5.3 Daten

#### V.5.3.1 Prozessdiagramm

##### V.5.3.1.1 Ist-Prozess

Die Poster durchlaufen, von ihrer Entstehung bis zu ihrer Ausstellung, den nachfolgenden Prozess:

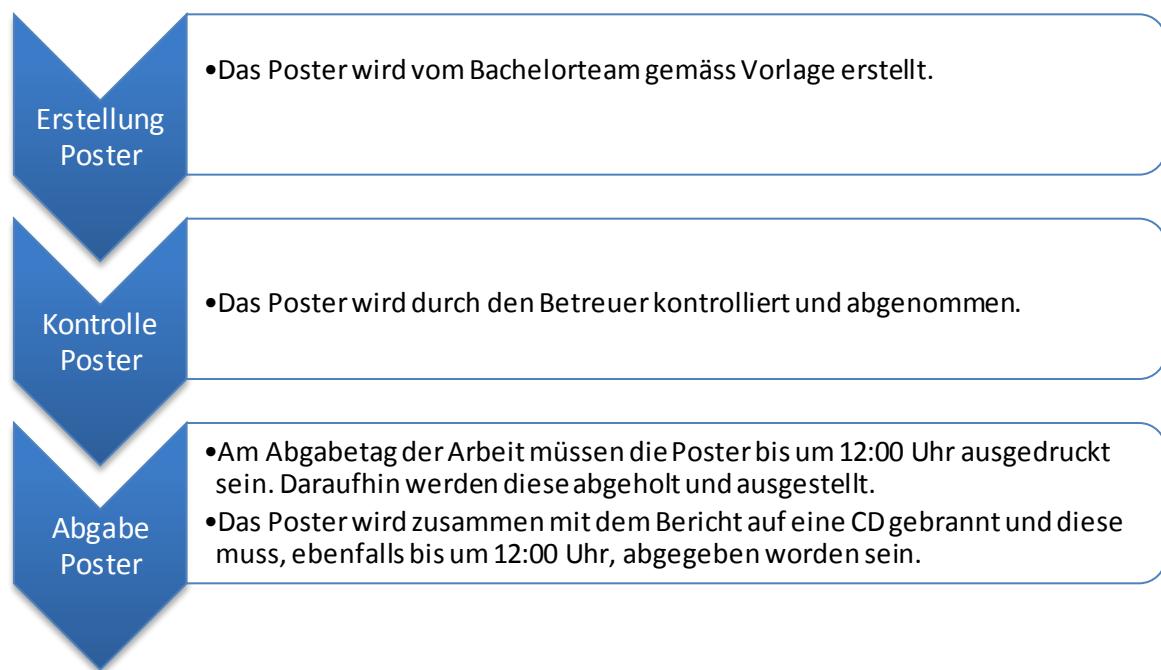


Abbildung 30 - Poster Ist-Prozess

##### V.5.3.1.2 Soll-Prozess

Auf der in Zukunft an der Hochschule installierten Videowall sollen die Bachelorposter aller Studiengänge verfügbar sein. Daher soll ein Poster in Zukunft den folgenden Prozess zu durchlaufen haben:

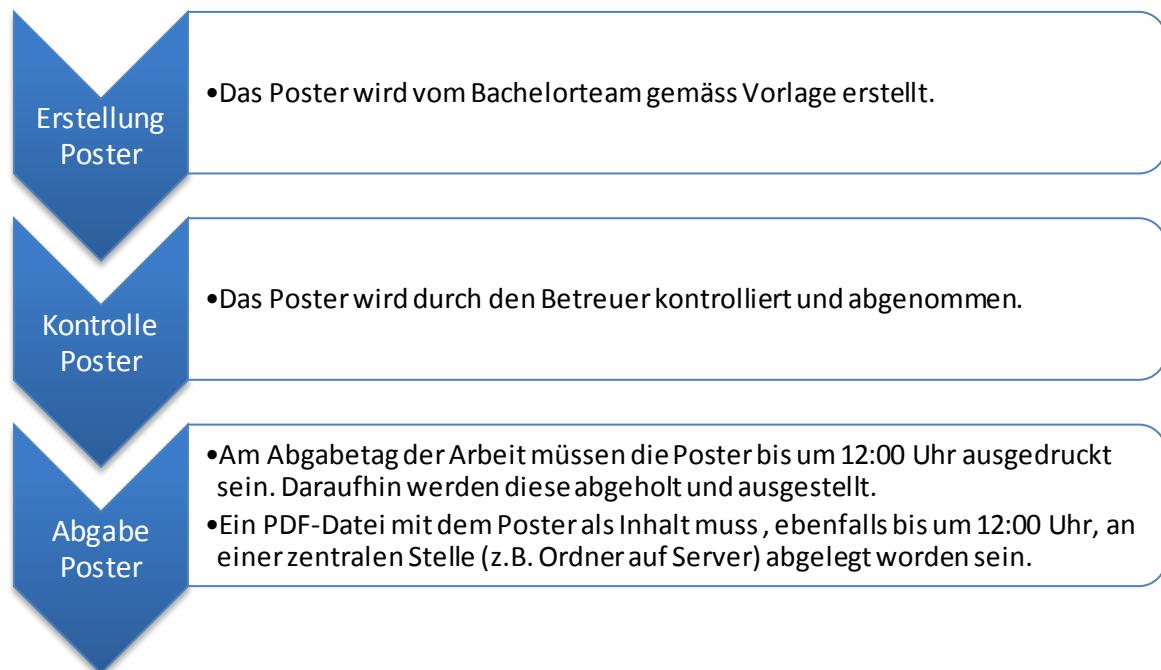


Abbildung 31 - Poster Soll-Prozess

### V.5.3.2 Domain Models

Auf der Videowall sollen verschiedene Inhalte präsentiert werden. Die Präsentation der Poster oder das Mittagsmenu sind Beispiele für solche Inhalte. Eine Anforderung an das Framework ist es daher, dass diese Inhalte hinzugefügt werden können.

#### V.5.3.2.1 Framework VideoWall

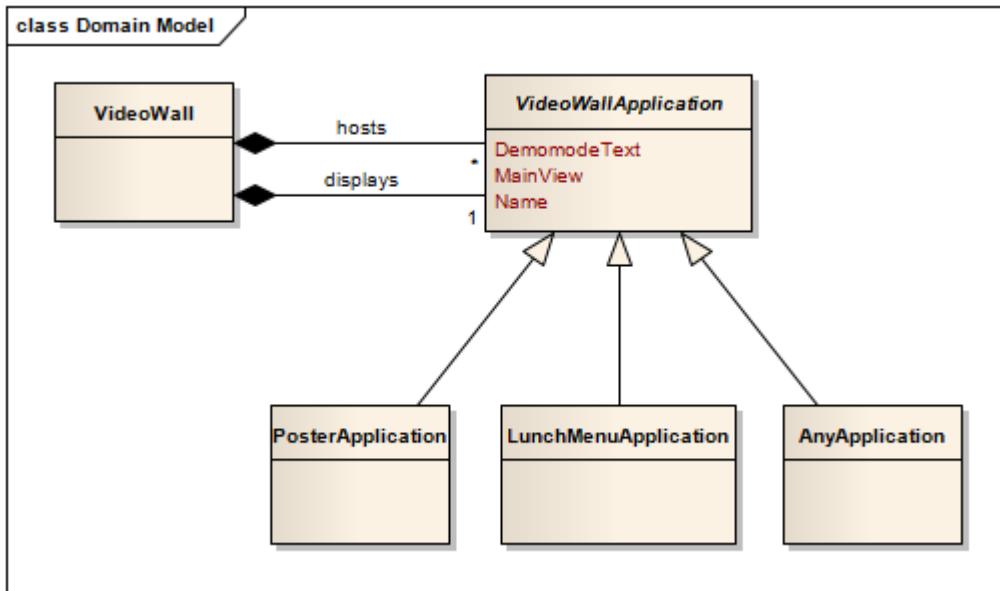


Abbildung 32 - Domain Model VideoWall

Wie in Abbildung 32 - Domain Model VideoWall ersichtlich ist, verwaltet die *VideoWall* mehrere *VideoWallApplications*. Zudem wird immer eine *VideoWallApplication* von der *VideoWall* angezeigt. Subklassen der *VideoWallApplication* sind die *PosterApplication*, die *LunchMenuApplication* oder jeglicher denkbarer Inhalt, welcher auf der *VideoWall* präsentiert werden soll. Diese Inhalte werden durch die Subklasse *AnyApplication* veranschaulicht. Eine Subklasse der *VideoWallApplication* verfügt über folgende Attribute:

Attribut	Beschreibung	Beispiel
<b>DemomodeText</b>	Der Teaser-Text, welcher im Demomodus angezeigt wird (siehe hierzu V.6.7.2 Design des Demomodus „Teaser“).	Hunger?
<b>MainView</b>	Die View, welche als Einstiegspunkt in die Applikation dient.	-
<b>Name</b>	Der Name der Applikation.	Mittagsmenu

Tabelle 12 - Attribute *VideoWallApplication*

Wie die Attribute auf GUI-Ebene eingesetzt werden, kann im Kapitel V.5.4.3 Screen Map nachgelesen werden.

### V.5.3.2.2 PosterApplication

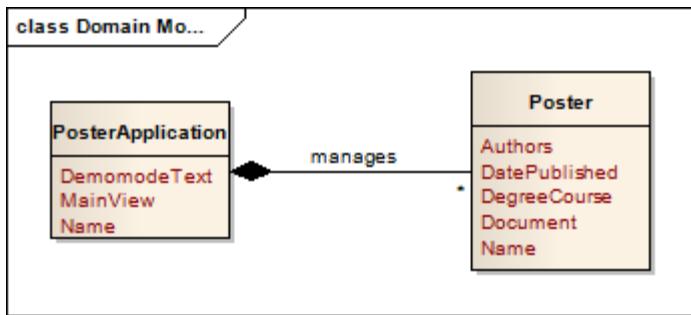


Abbildung 33 - Domain Model PosterApplication

Die *PosterApplication* verwaltet ihrerseits *Posters*. Ein *Poster* verfügt über folgende Attribute:

Attribut	Beschreibung	Beispiel
<b>Authors</b>	Die Namen der Autoren, welche das Poster erstellt haben.	Lukas Elmer, Christina Heidt, Delia Treichler
<b>DatePublished</b>	Das Datum der Publikation des Posters.	24.05.2012
<b>DegreeCourse</b>	Der Studiengang, für welchen das Poster erstellt wurde.	Informatik
<b>Document</b>	Das Poster selbst.	-
<b>Name</b>	Der Name der Arbeit.	HSR Videowall

Tabelle 13 - Attribute PosterApplication

### V.5.3.2.3 LunchMenuApplication

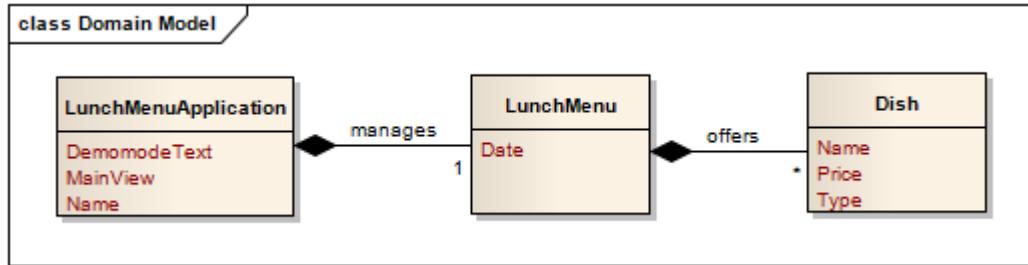


Abbildung 34 - Domain Model LunchMenuApplication

Die *LunchMenuApplication* verwaltet das *Lunchmenu*. Dieses verfügt über folgende Attribute:

Attribut	Beschreibung	Beispiel
<b>Date</b>	Das Datum des Lunchmenus. Dabei handelt es sich immer um das Datum des aktuellen Tags.	22.05.2012

Tabelle 14 - Attribute LunchMenu

Das *LunchMenu* selber bietet verschiedene *Dishes* an. Ein *Dish* hat folgende Attribute:

Attribut	Beschreibung	Beispiel
<b>Name</b>	Der Name des Dishes.	Poulet im Kokosnussmantel auf Karottenstiften mit Salbei Thai-Currysauce Basmatireis Fleisch aus der Schweiz
<b>Price</b>	Der Preis des Dishes.	INT 8.00 EXT 10.60
<b>Type</b>	Der Typ des Dishes.	Tagesteller

Tabelle 15 - Attribute Dish

---

### V.5.3.3 Verfügbarkeit der Daten

#### V.5.3.3.1 Poster

Die Poster sind bei den jeweiligen Sekretariaten in digitaler Form vorhanden. Sie sind zu einem grossen Teil als PDF abgespeichert, einige wenige Poster sind PowerPoint-Präsentationen. Dies kommt daher, dass die Vorlage für das Poster eine PowerPoint-Präsentation ist.

Die Poster, welche nicht im PDF-Format vorliegen, müssen manuell als PDF abgespeichert werden.

---

#### V.5.3.3.2 Mittagsmenu

Die Daten des Mittagsmenus werden auf der Internetseite der Mensa HSR Hochschule Rapperswil der SVGroup<sup>26</sup> abgerufen. Die benötigten Informationen zum Mittagsmenu werden aus dem HTML-Dokument herausgelesen und in der Applikation dargestellt.

---

<sup>26</sup> <http://hochschule-rapperswil.sv-group.ch/de.html>

## V.5.4 Graphical User Interface (GUI)

### V.5.4.1 Empirischer formativer Test zur Eruierung der Navigationsart

Die Videowall wird mittels Gesten gesteuert. Um herauszufinden, welche Gesten Nutzer intuitiv benutzen würden, wurde ein Test durchgeführt. Dieser wurde als ein Wizard of Oz - Experiment durchgeführt (siehe V.8.2.1 Test 1: Wizard of Oz). Der Test sollte auch zeigen, ob das erarbeitete GUI für den Benutzer einfach verständlich ist.

Es wurde folgendermassen vorgegangen:

- Am 14.03.2012 wurden erste Ideen zum GUI gesammelt.
- Nach dem Meeting vom 16.03.2012 wurden die am Meeting erhaltenen Inputs in die Test-Erarbeitung miteinbezogen.
- Am 27. März 2012 wurde der Test durchgeführt.

#### V.5.4.1.1 Ideensammlung

Am 14.03.2012 wurden Ideen zum GUI der Videowall gesammelt und Skizzen erstellt. Parallel dazu wurde überlegt, wie der Test ablaufen könnte.

Bei der Sammlung von Ideen für die Applikation selbst wurden auch erste Vorschläge für einen Demomodus festgehalten (siehe Abbildung 35 - Anforderungen an den Test). Dieser wurde aber erst später ausgearbeitet (siehe Kapitel V.5.4.2 Demomodus), da er für den Wizard of Oz - Test zur Eruierung der Navigationsart noch nicht benötigt wurde.

Folgende Überlegungen wurden gemacht:

Der Test wird als PowerPoint-Präsentation vorbereitet und mit einem Beamer projiziert. Je nach dem, wohin in der Applikation die Testperson navigiert, wird eine andere Folie der Präsentation eingeblendet. Dazu bestehen keine fliessenden Übergänge, damit der Aufwand zur Erstellung des Tests kleingehalten werden kann. Der Testperson soll zusätzlich ein Laserpointer zur Verfügung stehen, mit welchem sie Schaltflächen anwählen kann, da dies nicht über Gesten allein möglich ist. In der Applikation würde dies später ähnlich gelöst werden, indem der Nutzer seine Hand als Pointer verwenden kann, um Schaltflächen zu aktivieren.

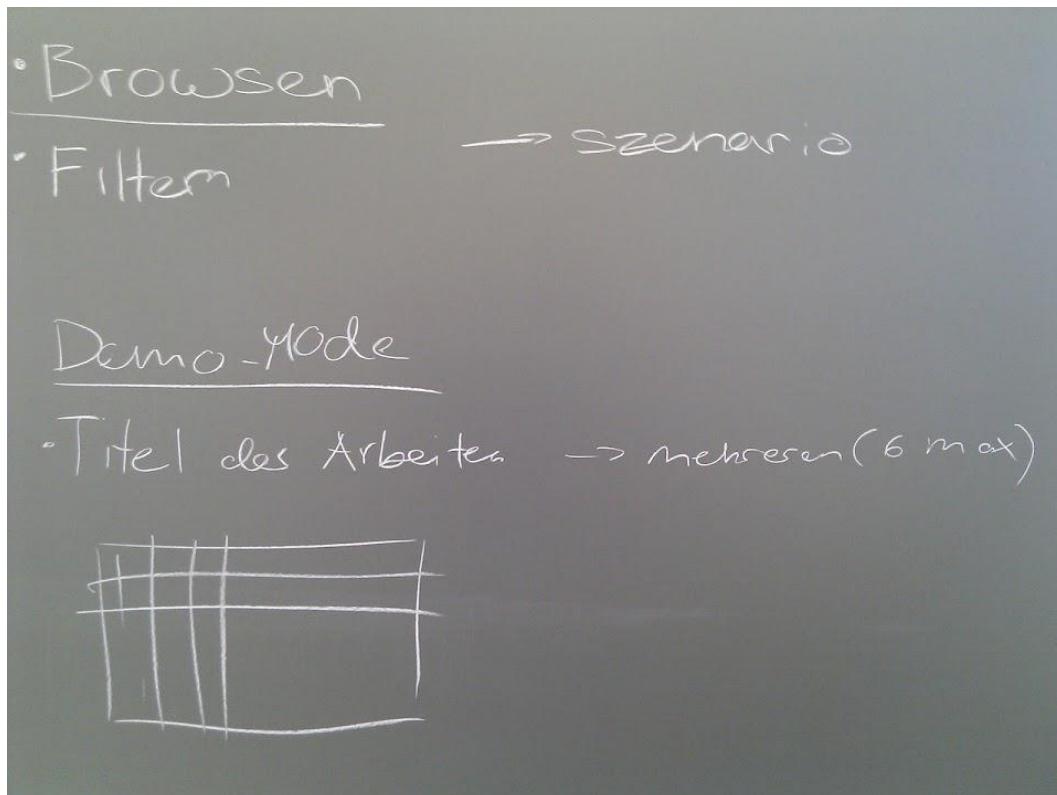
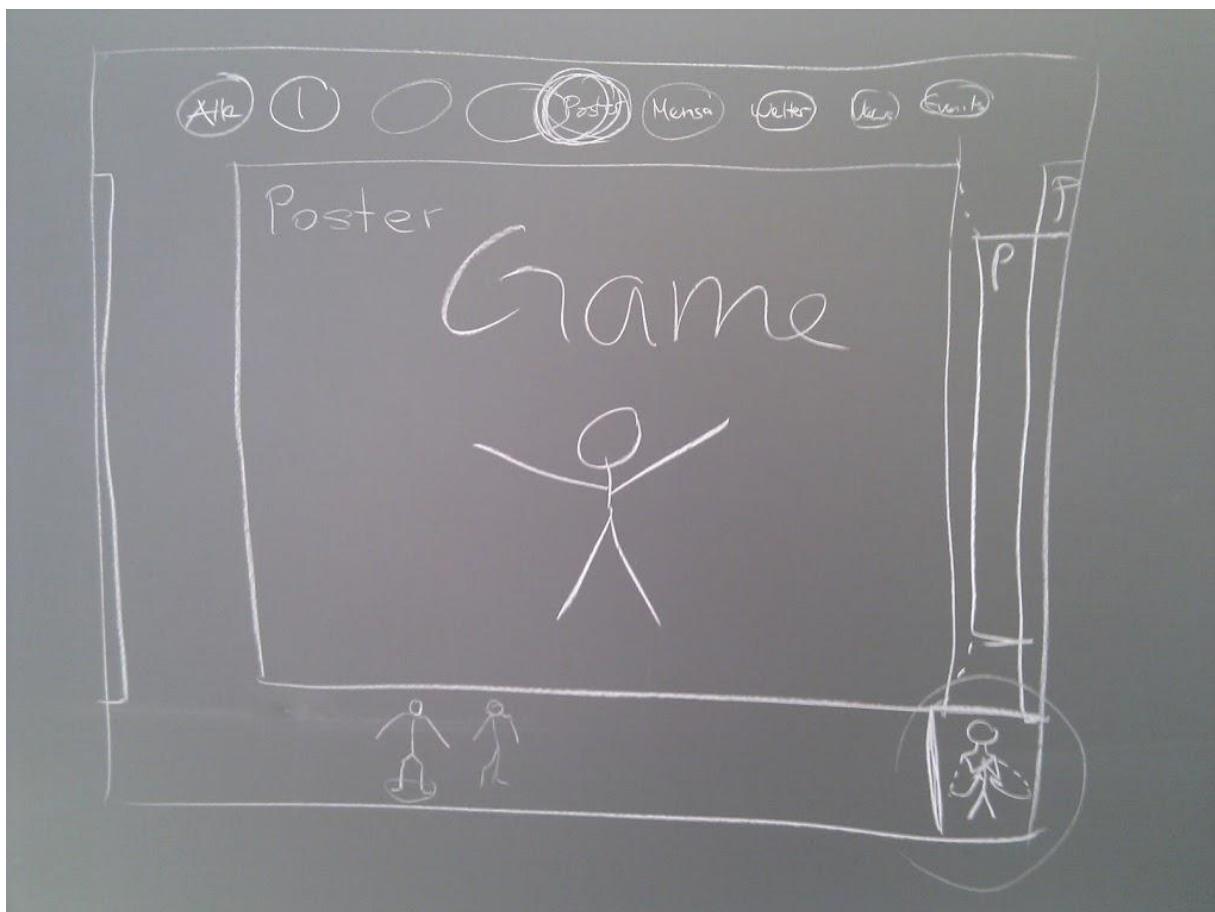


Abbildung 35 - Anforderungen an den Test

Die Abbildung 35 - Anforderungen an den Test zeigt, welche Anforderungen mit dem Test abgedeckt werden sollen. Der Test prüft das Browsen der Poster und die Navigation zwischen den verschiedenen Ansichten. In diesen werden beispielsweise die Poster, das Mittagsmenu der Mensa, das Wetter oder Informationen zu Veranstaltungen an der HSR dargestellt. Zwischen diesen soll einfach gewechselt werden können. Bei den Postern soll es zudem möglich sein, die Auswahl auf eine bestimmte Abteilung einzuschränken. Diese Anforderungen sollen getestet werden, indem die Testpersonen ein oder mehrere für den Test erarbeitete Szenarien durchlaufen wird.

Die Nutzer sollen zu Beginn von der Videowall angelockt werden. Dies soll über einen Demomodus geschehen. Eine Idee dazu ist, dass dem Nutzer ein Titel eines Posters als Schriftzug folgt, sobald dieser den Bereich betritt, in dem er von Kinect erkannt wird. Es können maximal sechs verschiedene Schriftzüge zur gleichen Zeit sechs Personen folgen, da dies die maximale Anzahl an Personen ist, die Kinect gleichzeitig erkennen kann. Ein anderer Vorschlag ist, das Poster in Stücke zerschnitten darzustellen, wobei die einzelnen Stücke ungeordnet auf der Wall angezeigt werden und sich bewegen. Sobald nun jemand erkannt wird und sich diese Person zur Videowall hindeht, so vereinigen sich die Teile zu einem Poster. Im Test wird der Demomodus weggelassen, da es nicht möglich ist, ihn im Wizard of Oz - Experiment umzusetzen. Es ist bekannt, dass je nach Teaser die Interaktion mit der Videowall variieren kann.



**Abbildung 36 - Posteransicht**

Die Abbildung 36 - Posteransicht stellt die Ansicht der Poster dar. Links und rechts des aktuell angezeigten Posters sind Teile des vorangehenden und des nachfolgenden Posters sichtbar. Dies veranschaulicht dem Benutzer, dass noch mehr Poster existieren und es möglich ist, zwischen ihnen zu navigieren.

Am oberen Rand befinden sich Schaltflächen, über welche zwischen den Ansichten gewechselt werden kann. In der Mitte des unteren Randes werden die Skelette der Personen, die von Kinect erkannt worden sind, angezeigt. Die Person, welche die Applikation steuern kann, wird gekennzeichnet.

Die Interaktive Hilfe in der rechten unteren Ecke wird in der Testpräsentation nicht vorkommen. Mit dem Test soll auch validiert werden, ob die Steuerung genug intuitiv ist, dass eine Hilfe überflüssig ist.

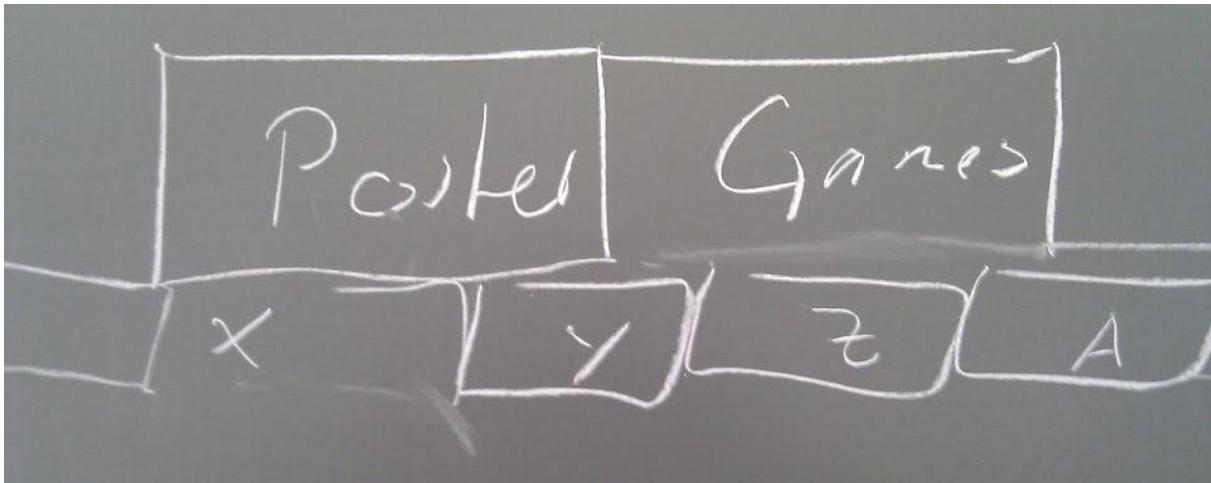


Abbildung 37 - Unterteilung in Tabs

Die obenstehende Abbildung 37 - Unterteilung in Tabs zeigt auf, dass die einzelnen Ansichten (obere Tab-Reihe) weiter unterteilt werden können (untere Tab-Reihe). Befindet man sich nun bei in der Ansicht der Poster, dient die zweite Reihe Tabs dazu, dass die Auswahl an Postern auf eine bestimmte Abteilung (z.B. Informatik) eingeschränkt werden kann.

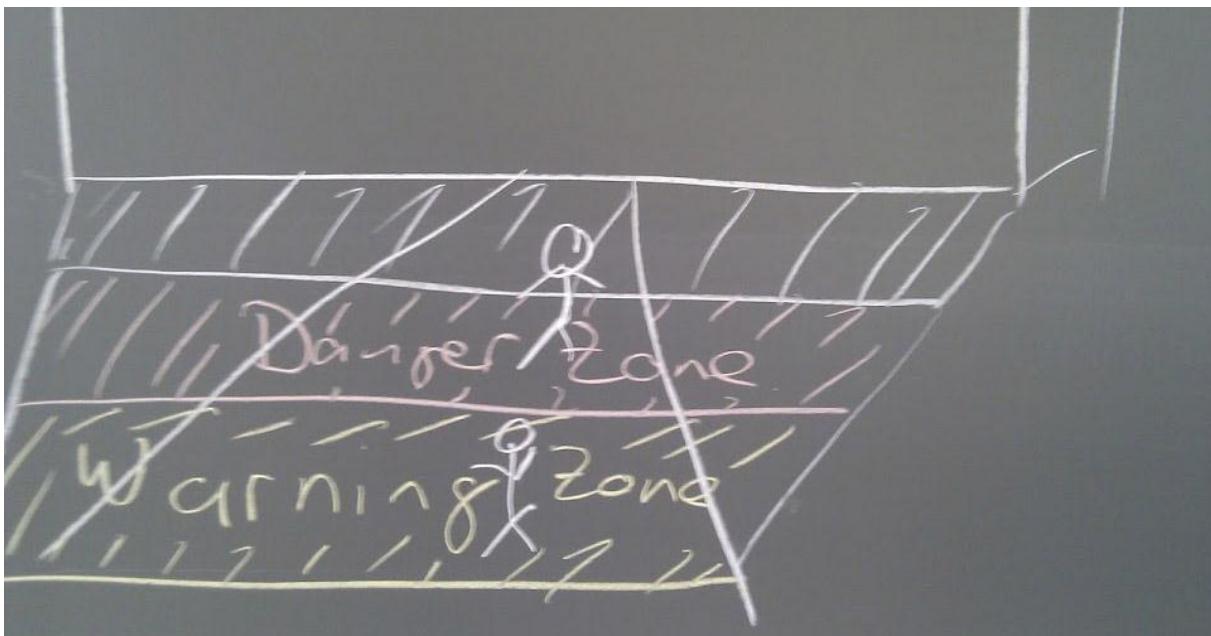


Abbildung 38 - Zonenmarkierung

Des Weiteren ist es für den Nutzer wichtig zu wissen, in welchem Abstand zur Videowall er stehen muss, um erkannt zu werden. Es sind zwei Zonen denkbar, eine Browsing- und eine Lese-/Interaktionszone. Wie in Abbildung 38 - Zonenmarkierung ersichtlich ist, könnten diese direkt am Boden vor der Videowall gekennzeichnet werden. Diese Markierungen würden zusätzlich auf die Videowall aufmerksam machen und der Nutzer würde immer, ob er im richtigen Bereich steht. Dieser Teil würde im Test mit Klebstreifen am Boden umgesetzt werden.

#### V.5.4.1.2 Ausarbeitung

Im Meeting vom 16.03.2012 wurde die Alternative, die Applikation nur mit der Hand als Zeiger (also ohne Gesten) zu bedienen, vorgeschlagen. Es gibt zwei Auffassungen der Steuerung von Kinect, die getestet werden können:

1. Konzept: Meine Hand ist die Maus
2. Konzept: Mein Körper ist die Maus (Steuerung mit Gesten)

Beim ersten Konzept könnte, wie bereits im Unterkapitel V.5.4.1.1 Ideensammlung beschrieben, ein Laserpointer genutzt werden, um den Pointer des Nutzers zu repräsentieren.

Bei der Diskussion über das zweite Konzept entstand die Idee, neben der Identifikation der Gesten zusätzlich die rechte Hand des Benutzers zu identifizieren, da gewisse Elemente nicht mit Gesten angesteuert werden können. Durch die Projizierung mit dem Beamer entsteht durch die Testperson ein Schatten an der Leinwand. Dieser könnte genutzt werden, um das Skelett zu simulieren. An der Stelle, wo nun der Schatten der Hand ist, könnte zusätzlich ein „Bällchen“ angezeigt werden. Dadurch wissen die Personen, wo ihre Hand ist und können diese an die von ihnen gewünschte Stelle bewegen. Problematisch ist hierbei, dass der Schatten wohl über die gesamte Applikation reichen müsste, damit der Nutzer alle aufgezeigten Elemente erreichen kann.

Die Markierungen am Boden, welche die Zonen umschreiben (siehe Abbildung 38 - Zonenmarkierung), könnten genauso gut in der Applikation selbst ersichtlich sein. Diese könnten unten in der Mitte zusammen mit dem Skelett angezeigt werden. So sieht der Nutzer immer, ob er sich in der richtigen Zone befindet.

Um die erarbeiteten Ideen sinnvoll zu testen, soll die Testapplikation interaktiv sein und daher sogleich als WPF-Applikation umzusetzen. Diese soll in etwa wie folgt aussehen:

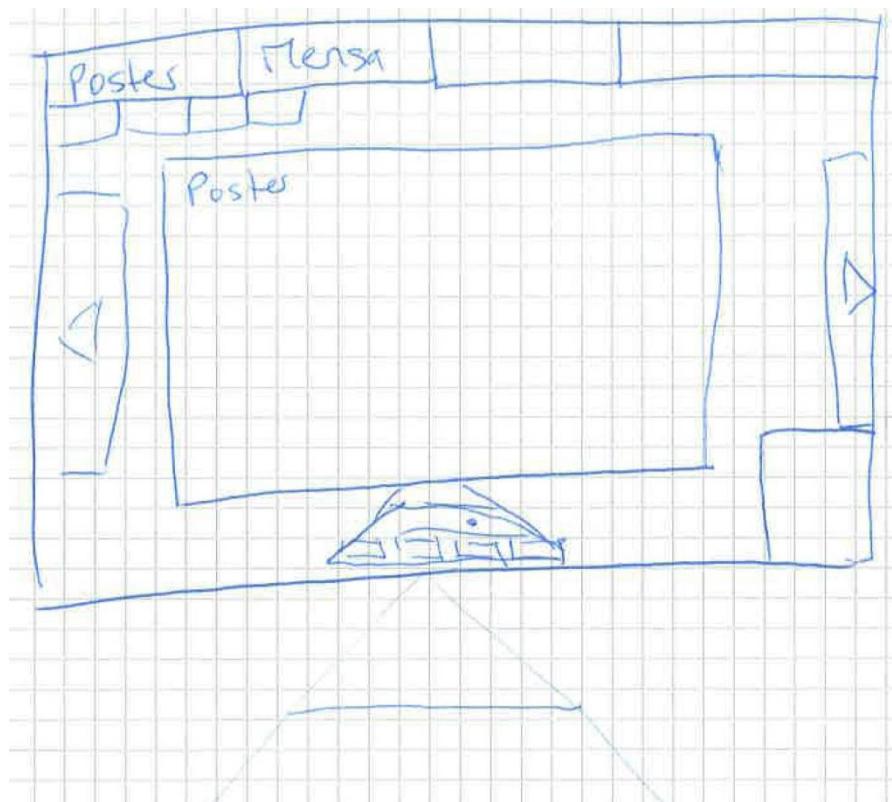


Abbildung 39 - Skizze Testapplikation

Durch die oberen Tabs kann zwischen den verschiedenen Ansichten (Poster, Mittagsmenu) gewechselt werden. Befindet man sich in der Browsing-Zone, so wird das Poster etwas kleiner dargestellt, dafür hat das Menu oben mehr Platz. Wechselt man in die Lese-Zone, so vergrößert sich das Poster und das Menu wird dafür kleiner. Die Zonen werden unten in der Mitte angezeigt. Dort befindet sich auch das Skelett, welches die Bewegungen der Person imitiert und angibt, in welcher Zone sie steht.

#### V.5.4.1.3 Umsetzung

Nicht alle in den Kapiteln V.5.4.1.1 Ideensammlung und V.5.4.1.2 Ausarbeitung beschriebenen Ideen konnten für die Testapplikation umgesetzt werden. Daher nachfolgend eine Zusammenfassung, was und was nicht für den Test umgesetzt wurde:

- Der Test wird als WPF-Applikation, nicht als PowerPoint-Präsentation, realisiert.
- Es wird kein Laserpointer, dafür werden die Handbewegungen des Benutzers mit der Maus simuliert.
- Es wird nur das Skelett des Benutzers angezeigt, auf Markierungen am Boden wird verzichtet.
- Für die Anzeige des Skeletts wird Kinect verwendet, jedoch nicht für die Steuerung.

- Der Demomodus wird nicht in diesen Test miteinbezogen und soll zu einem späteren Zeitpunkt umgesetzt werden.

Nachfolgend ist die Applikation, welche beim Test eingesetzt werden wird, genauer beschrieben und es wird gezeigt, wie sie gesteuert wird.

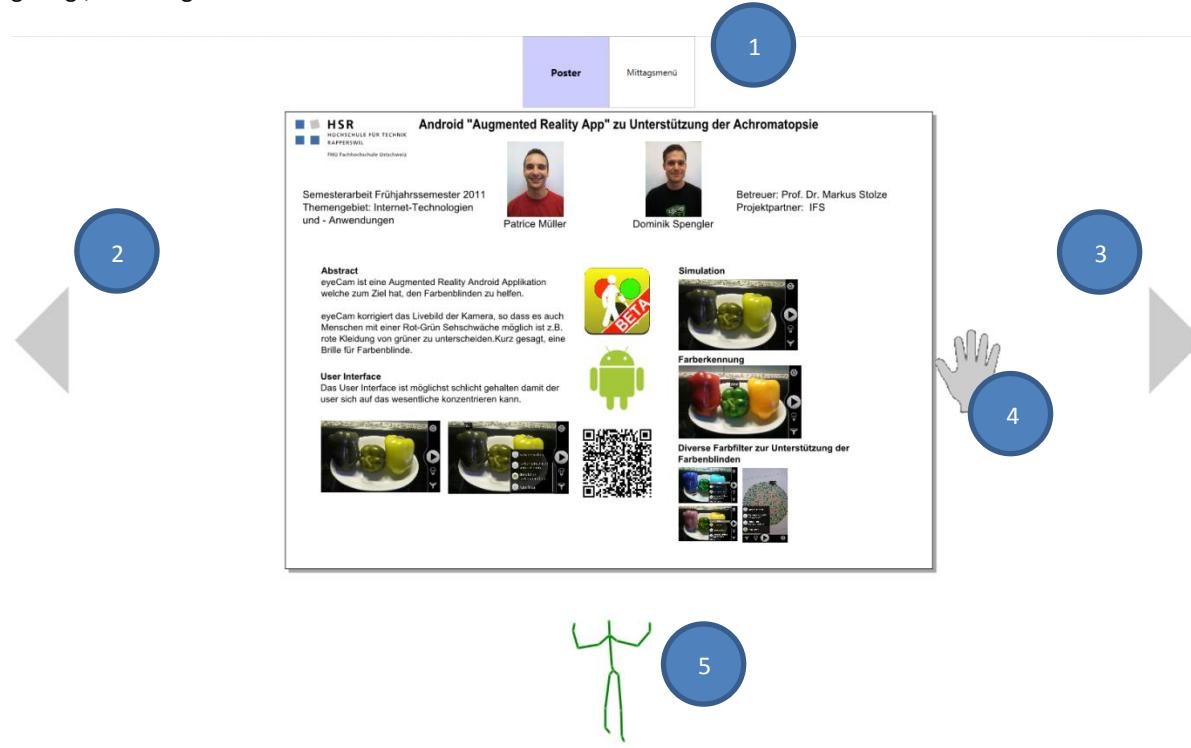


Abbildung 40 - Testapplikation

Die Testapplikation besteht aus zwei Ansichten. In der einen können Poster gelesen werden, in der anderen Ansicht wird das Mittagsmenu der Mensa angezeigt.

Die blauen Punkte in der Abbildung 40 - Testapplikation dienen der Beschriftung der einzelnen Komponenten in der Poster-Ansicht:

- Das Menu. Hier kann zwischen den Ansichten (hier Poster und Mittagsmenu) gewechselt werden.
- Der Navigationspfeil nach links. Er wird dazu benutzt, um nach links zum vorhergehenden Poster zu navigieren.
- Der Navigationspfeil nach rechts. Er wird dazu benutzt, um nach rechts zum nachfolgenden Poster zu navigieren.
- Die Hand. Sie symbolisiert die Hand der Testperson und befindet sich dort, wo die Testperson hinzeigt. Die Mauszeiger-Hand wird am Computer von den Testüberwachern bewegt (manuelle Steuerung), und zwar synchron zu den Bewegungen der Hand der Testperson.
- Das Skelett der Testperson. Es dient dazu, der Testperson zu zeigen, dass sie erkannt wird und merkt, dass sie durch Körperbewegungen die Applikation steuern kann. Das Skelett wird mithilfe von Kinect angezeigt.

Damit eine Schaltfläche effektiv gedrückt wird, muss die Testperson ihre Hand eine Weile darüber halten. Dabei wird über der Mauszeiger-Hand ein Uhr-Symbol angezeigt. Dies dient der Testperson als Feedback, damit diese weiß, dass die Applikation die Geste erkannt hat.

#### V.5.4.1.4 Durchführung & Fazit

Am 27. März 2012 wurde der Test als Wizard of Oz - Experiment durchgeführt. Für weitere Informationen siehe Kapitel V.8.2 Usability Tests.

## V.5.4.2 Demomodus

Damit Personen, welche das Verwaltungsgebäude der HSR passieren, mit der Videowall interagieren, müssen sie zuerst einmal auf diese aufmerksam und auch von ihr angezogen werden. Zu diesem Zweck wird ein Demomodus, der die Aufmerksamkeit und das Interesse der Passanten auf sich lenkt, erstellt.

Bei der Erarbeitung des Wizard of Oz - Experiments (Unterkapitel V.5.4.1 Empirischer formativer Test zur Erfüllung der Navigationsart) wurden bereits erste Ideen für einen Demomodus festgehalten. Trotzdem wurden nochmals Ideen gesammelt um eine möglichst breite Auswahl an Ideen für einen Demomodus zu erhalten.

### V.5.4.2.1 Ideensammlung und Besprechung

Im Sprint 9, im Zeitraum vom 23. bis am 30. April 2012, überlegte jedes Teammitglied für sich allein, wie der Demomodus umgesetzt werden könnte und hielt die Ideen fest. Am 01.05.12 wurden die unterschiedlichen Ideen im Team diskutiert. Die Ideen werden nachfolgend erläutert.

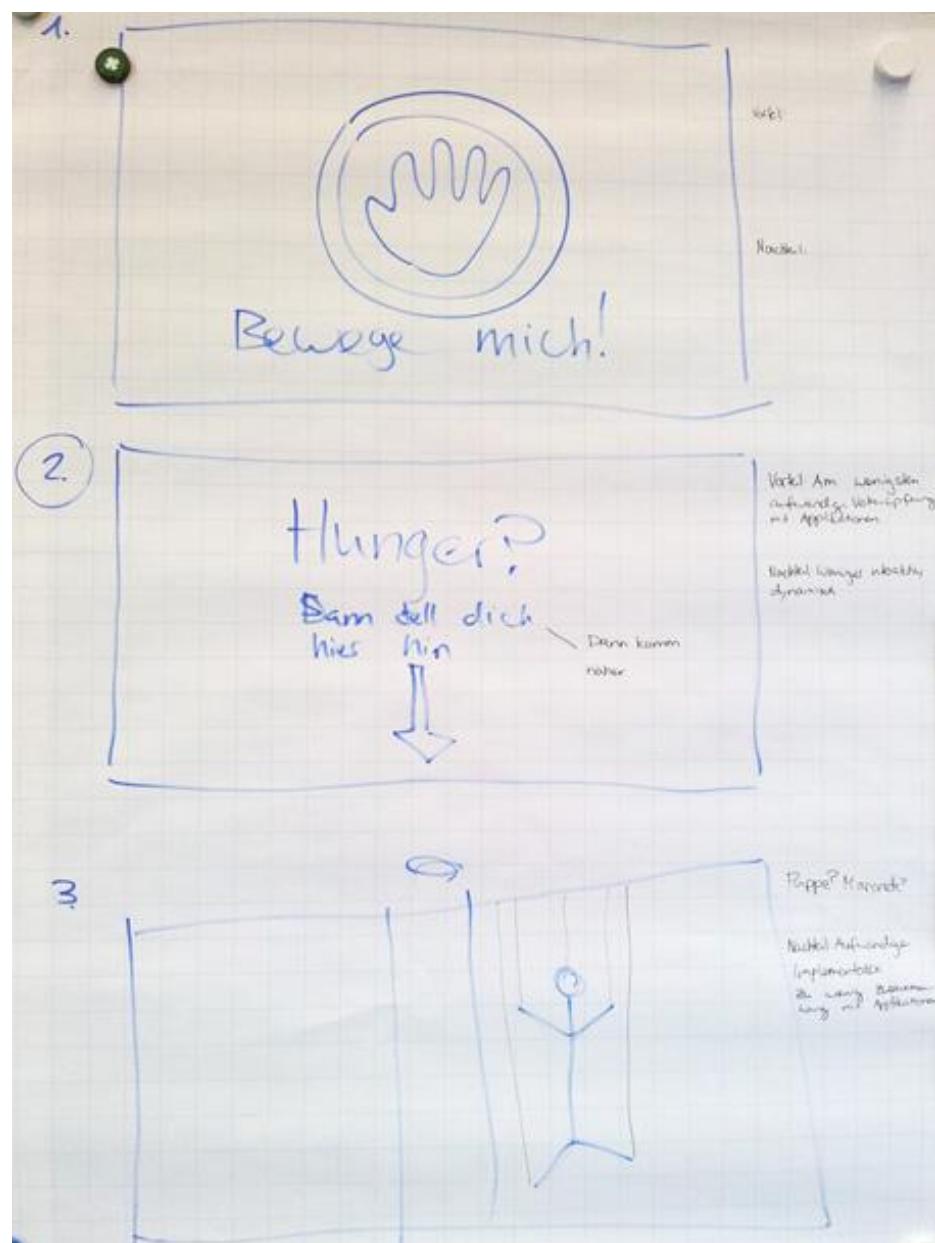


Abbildung 41 - Demomodus, Ideen 1-3

Der Demomodus der Idee 1 zeigt eine Hand. Der Schriftzug fordert die Passanten auf, die Hand zu bewegen. Eine Schwierigkeit ist, dass der Benutzer nicht weiß, wie er die Hand bewegen kann. Zudem ist dieser Demomodus zu wenig attraktiv, da er statisch ist.

Bei der Idee 2 wird der Benutzer direkt angesprochen. Um herauszufinden, was die Videowall für Informationen zur Frage oder Aussage auf der Wall bietet, stellt sich der Nutzer auf die am Boden markierte Stelle. Somit kann er von Kinect erkannt werden und gelangt dann in die Applikation mit der Ansicht, welche im Zusammenhang mit dem zu Beginn gezeigten Begriff oder Spruch steht. Im Falle der Skizze (siehe Idee 3, Abbildung 41 - Demomodus, Ideen 1-3) fragt die Wall: „Hunger?“. Stellt sich nun eine Person an die bezeichnete Stelle, so wird das Mittagsmenü angezeigt.

Idee 3 zeigt ein Skelett, welches Passanten mit Winken oder anderen Gesten dazu auffordert, näher zu kommen. Das auf der Wall angezeigte Skelett befindet sich immer auf gleicher Höhe mit dem Benutzer. Bewegt sich der Nutzer also beispielsweise nach links, so bewegt sich das Skelett ebenfalls nach links. Sobald der Benutzer in einen bestimmten Bereich des Sensors eingetreten ist (in Abbildung 41 - Demomodus, Ideen 1-3 durch die zwei senkrechten Striche in der Mitte markiert), so übernimmt er das Skelett und die Applikation startet. Bei dieser Lösung besteht die Problematik, dass der Demomodus wenig Zusammenhang mit der Applikation selbst hat.

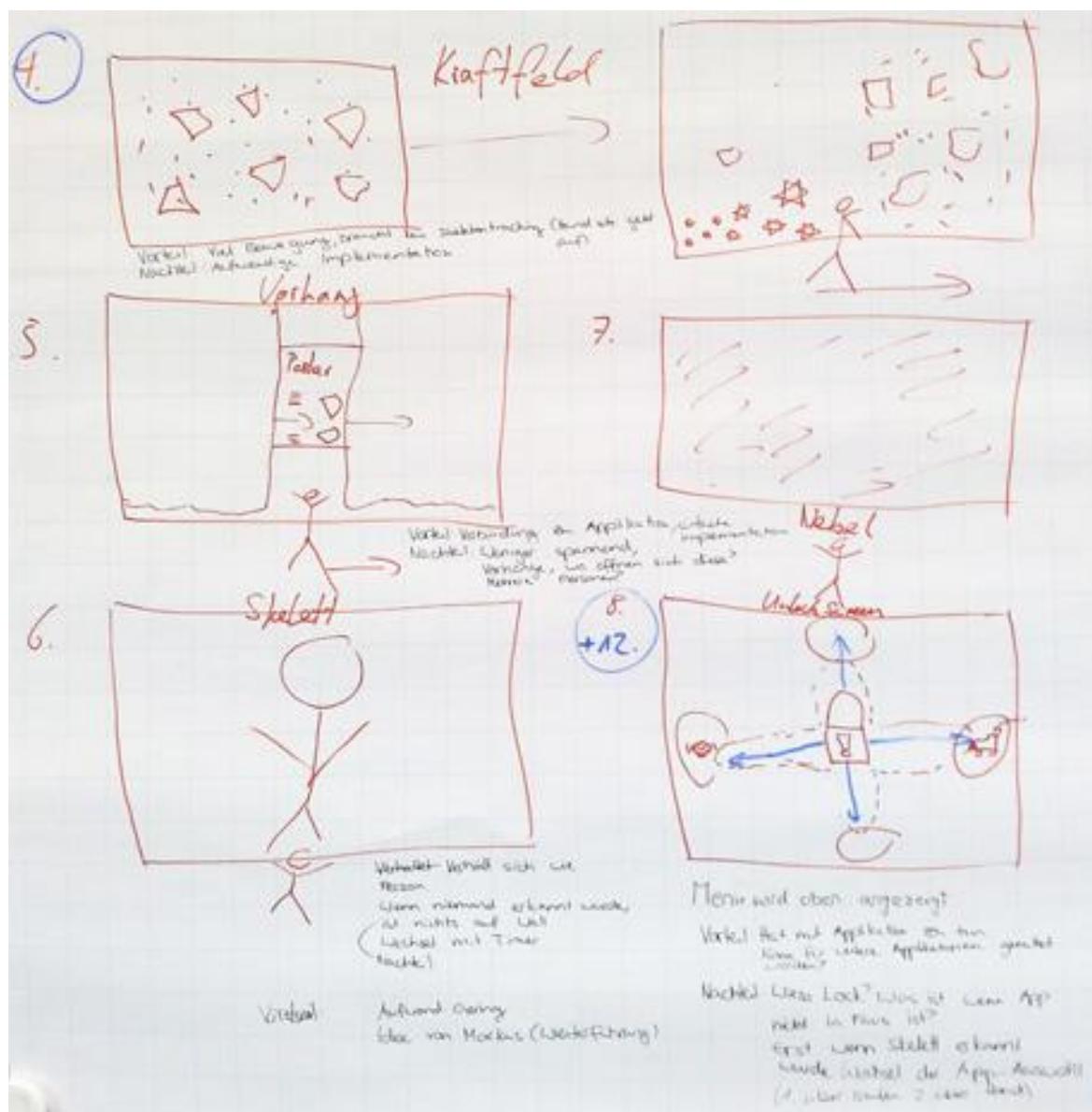


Abbildung 42 - Demomodus, Ideen 4-8

Bei der Idee Nummer 4 werden Objekte, verteilt über alle Monitore, dargestellt. Diese Idee ist an das Video über die interaktive Projektion einer Chanel Boutique in Paris<sup>27</sup>, welches bereits in der Konkurrenzanalyse (siehe V.3.5 Konkurrenzanalyse) referenziert wurde, angelehnt. Als Objekte sind hier beispielsweise Dreiecke, Puzzleteile oder kleine Stücke eines Posters denkbar. Wird nun ein Passant, welcher an der Wall vorbeilaufen, mit der Tiefenkamera erkannt, verschieben sie durch seine Bewegungen die Objekte auf der Wall. Werden mehrere Personen erkannt, welche beispielweise auch noch aus zwei verschiedenen Richtungen kommen, werden die Objekte von beiden Seiten verdrängt und bewegen sich in alle Richtungen. Bleibt der Benutzer über eine gewisse Zeitspanne vor der Wall stehen, so setzen sich die Teilchen zu einem Ganzen zusammen. Danach wird vom Demomodus in den Interaktionsmodus gewechselt.

Der Demomodus Idee 5 ist ein Vorhang, durch dessen schmale Öffnung ein Teil eines Posters sichtbar ist (diese Idee stammt vom Betreuer Markus Stolze). Die Öffnung des Vorhangs bewegt sich synchron mit der Position des Nutzers vor der Wall. Die Breite der Vorhangöffnung ist bestimmt durch den waagrechten Abstand der Hände des Benutzers. Die Vorzüge dieser Variante sind die einfache Implementation und die deutliche Verbindung zur Applikation selbst. Allerdings ist der Demomodus nicht sehr spannend und wird das Interesse der Passanten nur für kurze Zeit wecken können.

Idee Nummer 6 stellt das Skelett des Passanten, welcher bereits erkannt wurde, ganz gross auf der Wall dar. Kann kein Benutzer erkannt werden, wird nichts auf der Wall dargestellt, was der Nachteil dieser Lösung ist. Das grosse Skelett imitiert alle Bewegungen des Nutzers. Es wird so unmissverständlich klar, wie die Steuerung der Applikation vor sich geht. Nach Ablauf eines Timers wechselt die Anzeige zur eigentlichen Applikation.

Bei der Idee 7 wird die Applikation im Hintergrund schwach angezeigt, davor befindet sich Nebel. Durch Wischbewegungen des Benutzers kann der Nebel entfernt werden und die eigentliche Applikation kommt dahinter zum Vorschein.

Die Idee 8 für den Demomodus zeigt einen Lock-Screen mit einem Vorhängeschloss, analog zu dem eines Smartphones [chaudhri09]. Durch das Vorbeilaufen an der Wall, durch näher kommen oder weiter weg gehen kann die Applikation entsperrt werden. Dabei öffnet sich dann der Screen, auf dessen Symbol das Schloss beim Entsperrnen geschoben wurde. Läuft ein Passant nun von rechts nach links an der Videowall vorbei, so schiebt er das Vorhängeschloss auf das Symbol mit dem Teller und die Wall zeigt das Mittagsmenü der Mensa an.

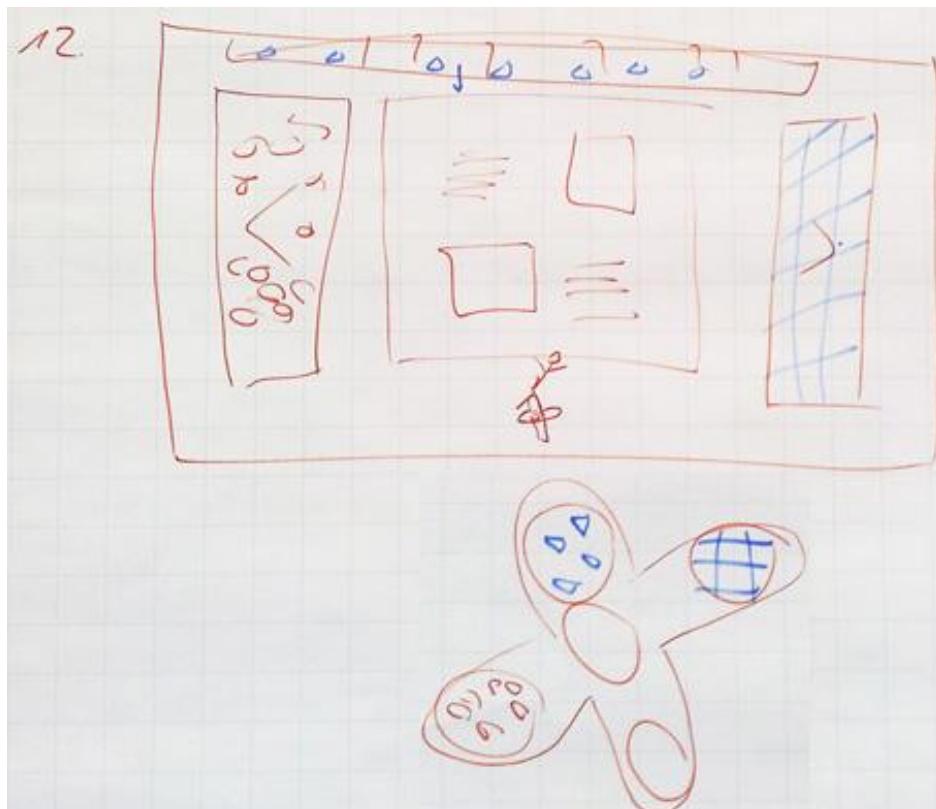


Abbildung 43 - Demomodus, Idee 12, Erweiterung zu Idee 8

<sup>27</sup> <http://www.youtube.com/watch?v=CLD1wVbcD8w&feature=related>

Die Idee 12 aus obiger Abbildung zeigt eine Weiterentwicklung der Idee 8. Am Boden wird eine Markierung angebracht, welche einem besser verständlich macht, dass man die Wall steuern kann, indem man seine Position ändert. Dieses Kreuz am Boden bietet nun eine zusätzliche Navigationsmöglichkeit zum simplen Steuern durch die Hand (siehe dazu Kapitel V.5.4.1 Empirischer formativer Test zur Eruierung der Navigationsart). Durch farbliche Kennzeichnung der Schaltflächen der Applikation können diese auch durch Positionsänderungen analog der Markierung am Boden betätigt werden, und nicht mehr nur alleine durch das Nutzen der Hand als Pointer.

Die Erweiterung der Steuerung durch die Nutzung der Markierungen am Boden kann auch für andere Ansichten der Applikation verwendet werden.

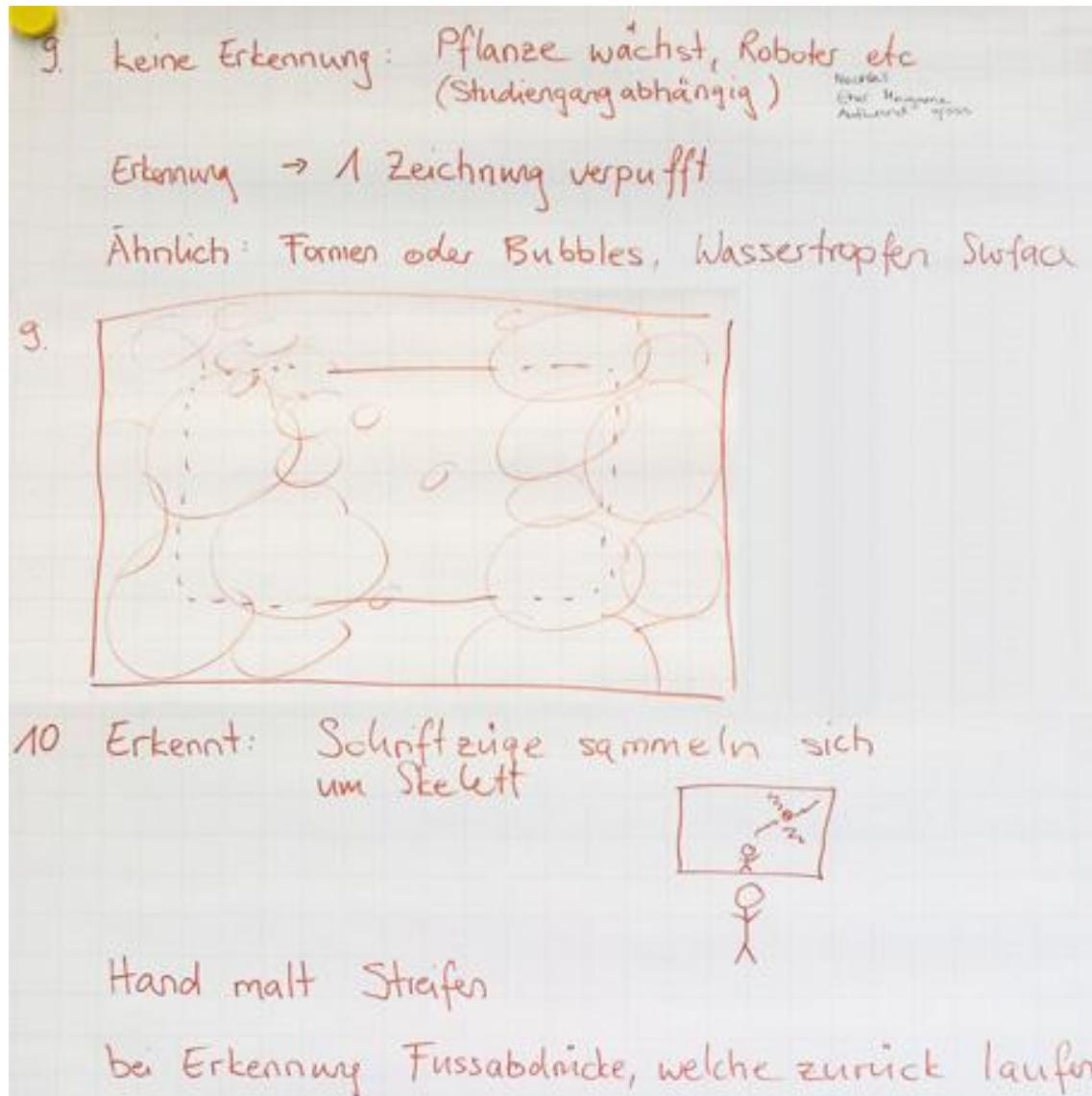


Abbildung 44 - Demomodus, Ideen 9 und 10

Bei der Idee 9 wachsen oder erscheinen im Demomodus Pflanzen, Roboter und andere Gegenstände, die thematisch mit den Studiengängen an der HSR zu tun haben. Wird ein Passanterkannt, so reagiert einer der Gegenstände auf ihn. Bleibt ein Passant nach der Erkennung stehen, so verschwinden alle Gegenstände und die Applikation kommt zum Vorschein. Als Alternative könnten auch Seifenblasen auf der Wall dargestellt werden. Wird eine Person erkannt, so kann diese durch Bewegungen diese Seifenblasen zerplatzen lassen. Der Aufwand für die Implementierung einer dieser Ideen ist jedoch gross, und zudem würden sich die Ideen auch eher als Mini-Game anstatt als Demomodus eignen.

Der Demomodus der Idee 10 zeigt verschiedene Schriftzüge, beispielsweise die Titel von zufällig ausgewählten Postern. Sobald ein Passanterkannt wurde, sammeln sich diese Schriftzüge um die Hand des Benutzers.

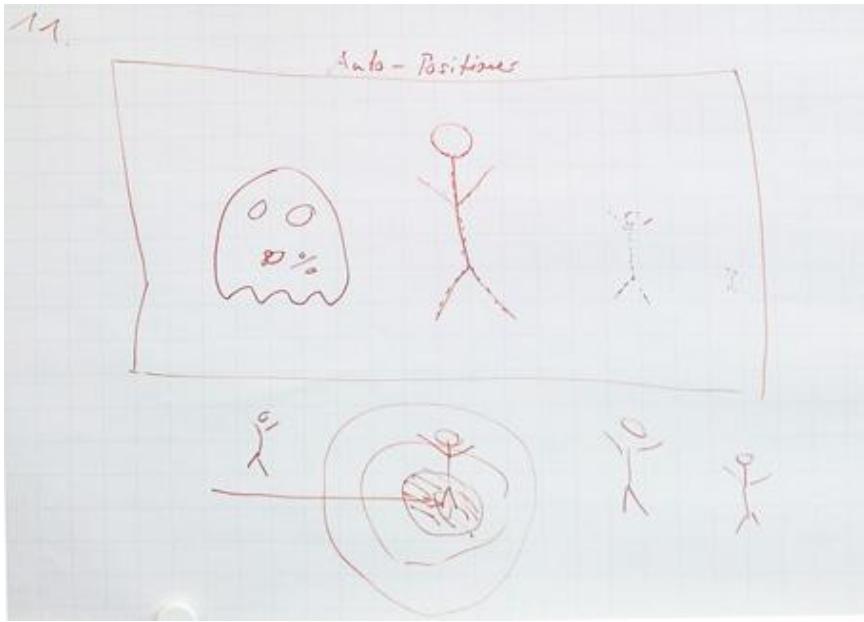


Abbildung 45 - Demomodus, Idee 11

Die Idee 11 dient vor allem der korrekten Positionierung des Benutzers vor der Wall. Passanten, welche weit entfernt von der idealen Position stehen, werden auf der Wall als kleine, durchsichtige Skelette (hier sind auch alternative Anzeigen denkbar) repräsentiert. Je näher man zur Mitte steht, desto grösser und deutlicher zeigt die Wall das Skelett an. Eine Markierung am Boden soll dem Benutzer helfen, sich ideal zu positionieren. Ein Nachteil dieser Idee ist der fehlende Zusammenhang zwischen dem Demomodus und der Applikation.

#### V.5.4.2.2 Auswahl der besten Idee für den Demomodus

Nach dem Zusammentragen und Besprechen aller Ideen am 01.05.12, wählte das Team die folgenden drei Ideen zur weiteren Vertiefung aus: Nummer 2, 4 und 8 resp. 12.

Am 02.05.12 wurden diese drei Ideen auch noch mit Markus Stolze besprochen. Dabei kam zur Sprache, dass sich die Idee 8 resp. 12 nicht eignet, da durch den positiv ausgefallenen Usability Test (V.5.4.1 Empirischer formativer Test zur Eruierung der Navigationsart) die Steuerung auf „Meine Hand ist die Maus“ festgelegt wurde.

Der Demomodus aus den Ideen 8 und 12 bringt mehrere Schwierigkeiten mit sich. Das Angebot von zwei Navigationsmöglichkeiten (Hand und Markierung am Boden) kann verwirrend sein. Die Sperrung der Wall mit einem symbolischen Vorhängeschloss wirkt sich eher negativ auf das mögliche Interesse der Benutzer aus, da das Schloss als Interaktions-Verbot aufgefasst werden könnte. Bei einem Smartphone macht solch eine Sperrung durchaus Sinn, damit nicht unabsichtlich irgendwelche Schaltflächen betätigt werden. Bei der Wall ist dies hingegen nicht nötig. Zudem ist die Anzahl der Favoriten-Programme in dieser Ansicht mit dem Lock-Kreuz auf vier Stück beschränkt.

Als Favorit zur Umsetzung wurde die Idee 4 gewählt:

Die Idee 4 mit den Objekten, die durch Bewegungen von Passanten durcheinander gewirbelt werden, bringt viel Bewegung und hat daher eine grosse Anziehungskraft. Des Weiteren ist kein Skeletal Tracking nötig, zur Umsetzung wird der Tiefensor genutzt. Die Problematik der verzögerten Erkennung des Skeletts eines Passanten besteht hier also nicht. Erschwerend ist hier nur die eher aufwändige Implementation der Idee. Trotzdem soll diese Idee als Demomodus umgesetzt werden.

Die Idee 2 ist die Alternative, falls die Umsetzung der Idee 4 aus Zeitgründen nicht klappen sollte:

Die Idee 2 mit dem Anzeigen eines Begriffes oder Spruches hat den Vorteil, dass sie ohne grossen Aufwand implementiert werden kann. Zudem hat der Demomodus konkret etwas mit der Applikation selbst zu tun. Nachteilig erweist sich, dass dieser Modus wenig Dynamik hat. Diese Idee wird daher als Alternative zur Idee 4 beibehalten.

Die anderen Design-Vorschläge zu den Demomodi (siehe V.5.4.2.1 Ideensammlung und Besprechung) wurden hier so ausführlich dokumentiert, damit allfällige weitere Arbeiten hierauf aufsetzen können.

Das Design des Demomodus ist im Kapitel V.6.7 Design des Demomodus festgehalten.

### V.5.4.3 Screen Map

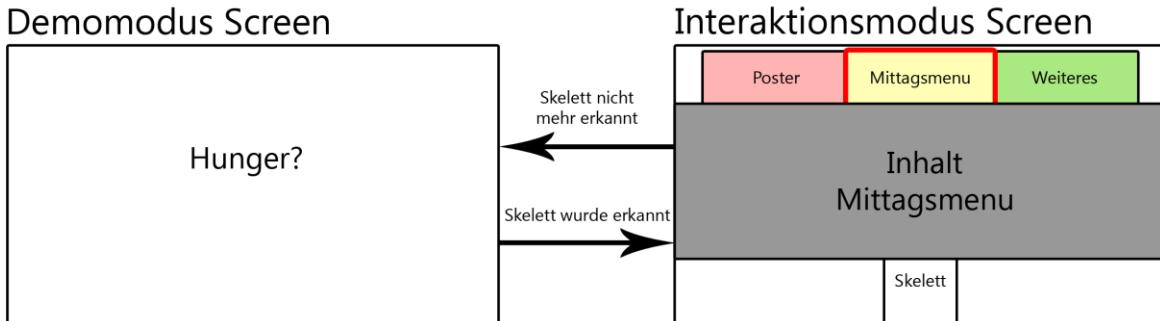


Abbildung 46 - Screen Map

Die Screen Map bezieht sich auf die im Kapitel V.5.3.2 Domain Models vorgestellten Klassen und Attribute und beschreibt die Applikation im aktuellen Zustand.

Solange kein Skelett erkannt wird, befindet sich die Applikation im Demomodus (mehr Informationen hier zu im Kapitel V.5.4.2 Demomodus) und der Demomodus Screen wird angezeigt. Auf diesem wird der *DemomodeText* einer zufällig ausgewählten *VideoWallApplication* dargestellt, diese Applikation und somit der Text wechseln periodisch. Im Fall der Abbildung 46 - Screen Map handelt es sich bei der *VideoWallApplication* um die *LunchMenuApp*.

Sobald ein Skelett erkannt wird, wechselt die Anzeige auf der Videowall vom Demomodus in den Interaktionsmodus. Im Interaktionsmodus Screen können über das Menu die verschiedenen *VideoWallApplications* angezeigt werden. Für jede *VideoWallApplication* wird ein „Knopf“ erstellt, mit dessen Betätigung die jeweilige *VideoWallApplication* ausgewählt werden kann. Der Inhalt der momentan ausgewählten *VideoWallApplication* wird in der Mitte des Screens dargestellt, im Beispiel der Abbildung 46 - Screen Map ist das der Inhalt der *LunchMenuApp*. Bei einem Ansichtenwechsel auf die *PosterApp* würde in der Screen-Mitte der Inhalt der *PosterApp* visualisiert werden.

### V.5.4.4 Design Entscheide

Die Design Entscheide für die verschiedenen Applikationen werden nachfolgend aufgelistet.

#### V.5.4.4.1 Steuerung mit der Hand

Das externe Design der Videowall-Applikation und der Poster-Applikation (Plug-in) wurde mehrheitlich aus dem ersten Usability Test (siehe V.8.2.1 Test 1: Wizard of Oz) übernommen, da die Nutzer das dort verwendete Design gut verstanden. Um eine Schaltfläche auswählen zu können, soll der Nutzer eine Zeitlang mit dem Handcursor darauf verweilen. Damit das vom Nutzer verstanden wird, wird zusätzlich eine Animation gestartet, sobald der Nutzer sich auf der Schaltfläche befindet.

Abbildung 47 - Handcursor Animation

Aus dem Test hat sich ergeben, dass die Applikation nicht nur mit der rechten, sondern auch mit der linken Hand bedienbar sein soll. Deshalb wurde ein zweiter Handcursor für die linke Hand erstellt, welcher angezeigt wird, sobald die Applikation mit der linken Hand bedient wird.

#### V.5.4.4.2 Demomodus

In der Ideensammlung für den Demomodus (V.5.4.2.1 Ideensammlung) wurde das Design des Demomodus grob definiert. Der Hintergrund sollte eine auffällige Farbe haben. Auf dem Hintergrund soll ein Teaser-Text angezeigt werden und darunter die Aufforderung „Komm näher“. Die Farbe und der Text dienen dem Anlocken der Passanten, damit diese darauf mit der Videowall interagieren. Sobald der Nutzer von Kinect erkannt wurde, beginnt ein Timer herunterzuzählen. Nach dessen Ablauf wechselt die Applikation in den Interaktionsmodus und der Benutzer kann die Applikation steuern.

Der Demomodus wurde nach dessen Implementation mit einem Usability Test geprüft (siehe V.8.2.3 Test 3: Reaktion auf Demomodus). Aus dem Test konnte folgender Schluss gezogen werden: Wenn ein Passant im

Demomodus von Kinect erkannt wird, soll beim Anzeigen des Timers gleichzeitig das Skelett des Nutzers angezeigt werden, damit neugierig wird und stehen bleibt. Er wird sofort erkennen, dass er das Skelett ist und somit nach dem Wechseln in den Interaktionsmodus die Applikation problemlos bedienen kann. Dieser Redesign Entscheid wurde sogleich nach der Durchführung des Tests implementiert.

---

#### *V.5.4.4.3 Menu mit Tabs*

Das Menu der Applikation besteht aus Tabs. Diese Tabs stellen Elemente dar, welche die meisten Nutzer von Browserapplikationen kennen und daher verstehen sollten. Dieser Sachverhalt wurde beim Test V.8.2.4 Test 4: Grafisches Design validiert. Nach dem Test wurden noch farbliche Anpassungen vorgenommen, welche dem Nutzer verdeutlichen, welches Tab gerade aktiv ist.

Zwei weitere Punkte, welche aber im Rahmen der Bachelorarbeit nicht umgesetzt werden können, ergaben sich aus dem Usability Test. Der Handcursor dreht sich über jedem anwählbaren Element, was für den Nutzer verwirrend sein kann. Im Menu dreht sich der Handcursor also beispielsweise auch, wenn er sich über dem bereits aktiven Tab befindet. Auch soll der Handcursor, wenn er sich über einem nicht anwählbaren Element befindet, abgeschwächt oder durchgestrichen dargestellt werden. Diese zwei Anforderungen wurden in Form von User Stories in den Backlog (siehe V.4.3 Funktionale Anforderungen) aufgenommen.

---

#### *V.5.4.4.4 Corporate Design HSR*

Das Corporate Design der HSR [hsr11] legt verschiedene Farben, Schriftarten sowie die Verwendung des HSR Logos fest. Diese Vorgaben wurden für die Videowall so gut als möglich eingehalten.

Am 05.06.12 wurde das Design mit Michael Lieberherr und Oliver Kirchhofer von der Kommunikationsstelle der HSR besprochen. Aus dem Gespräch ergaben sich folgende zwei Änderungen, die sogleich umgesetzt wurden:

- Verwendung der Schriftart Arial ist in Ordnung
- Pfeile in der Poster-Applikation kleiner machen (Iceberg Buttons)

Das daraus resultierende Externe Design wird im Kapitel V.5.4.5 Externes Design aufgezeigt.

Es wurden noch weitere Design Vorschläge gemacht, welche bei einer Weiterentwicklung der Applikation umgesetzt werden könnten:

- Das HSR Logo kommt oben links hin. Der Hintergrund des Logos muss weiss sein, dies bedarf Anpassungen im Menu.
- Der Abstand zwischen den Tabs im Menu soll grösser sein.
- Die Iceberg-Buttons sollen Grow-Buttons sein, welche in der Vertikale grösser werden, wenn mit der Maus darüber gefahren wird. Dadurch erkennt der Benutzer, dass der Button grösser ist als er angezeigt wird.
- Es sollen Gesten verwendet werden können, damit für das Browsen der Poster keine Pfeile mehr angezeigt werden müssen.
- Das Menu soll einem Home-Screen, auf welchem das HSR Logo platziert wird, weichen. Von da aus können die Inhalte der Videowall durch das Anwählen der Buttons angeschaut werden. Über einen Home-Button im unteren Bereich neben dem Skelett kommt man zurück zum Home-Screen.

---

#### *V.5.4.5 Externes Design*

Nach den in Kapitel V.5.4.4 Design Entscheide aufgelisteten Design Entscheiden wurde das Externe Design für die Hauptapplikation sowie für die Poster- und die Mittagsmenu-Applikation erarbeitet.

---

#### *V.5.4.5.1 Videowall-Applikation*

Oben befinden sich die Tabs. Aktiviert man eines davon, so wird in die entsprechende Applikation gewechselt. Das aktive Tab wird weiss dargestellt, die nicht aktiven Tabs sind Blau hinterlegt. In der Mitte des Bildschirmes wird die aktive Applikation angezeigt. Unten links wird zu jeder Zeit das HSR Logo angezeigt. In der Mitte des unteren Bereichs wird das Skelett des Nutzers in blauer Farbe angezeigt. Der Cursor ist als Hand dargestellt. Wird mit der rechten Hand bedient, so wird ein Icon mit einer rechten Hand angezeigt. Wird mit der linken Hand bedient, so hat der Cursor ein Icon mit einer linken Hand.



### Mensa Menu Donnerstag, 07.06.2012

Tagesteller	Vegetarisch	Täglich im Angebot	Wochen-Hit
Bis zum Start des neuen Semester ist unser Angebot um den Tagesteller reduziert INT 8.00 EXT 10.60	Bis zum Start des neuen Semester ist unser vegetarisches Gericht als Komponente erhältlich. INT 8.00 EXT 10.60	Dauerbrenner  Kalbsbratwurst vom Grill (Schweiz) mit Zwiebelsauce INT 8.00 EXT 10.60	Räucherlachsteller (GB/ Zucht) mit Sahne Meerrettich auf Kartoffelreibeplätzchen Nüsslisalat  Vorbereitungszeit ca. 8 min INT 14.50 EXT 15.50



Abbildung 48 - Externes Design, Videowall-Applikation

#### V.5.4.5.2 Poster-Applikation (Plug-in)



### Biodiversitätskalender Linthebene

Analyse: [www.biodiversitaet-linthebene.ch](#)

#### Lage des Perimeters



Perimeter

#### Die Linthebene im Wandel



Die Linthebene im Wandel

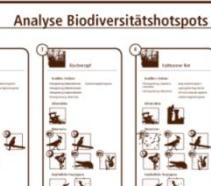


Abbildung 49 - Externes Design, Poster-Applikation (Plug-in)

Die Poster-Applikation zeigt ein Poster an. Mit den Pfeilen, welche sich links und rechts vom Poster befinden, kann zum vorherigen oder zum nachfolgenden Poster navigiert werden. Die Pfeile sind in Blau gehalten.

---

#### V.5.4.5.3 Mittagsmenu-Applikation (Plug-in)

Die Abbildung 48 - Externes Design, Videowall-Applikation zeigt die Mittagsmenu-Applikation. Diese wurde sehr schlicht gehalten. Besonders relevante Texte, wie das Datum und die Kategorie, wurden im Schriftbild hervorgehoben.

---

#### V.5.4.5.4 Demomodus

Für den Demomodus „Teaser“ (siehe V.6.7 Design des Demomodus) wurde ein externes Design erarbeitet. Sobald der Demomodus aktiv ist, wird auf der Videowall eine aus einer vorgegebenen Farbpalette zufällig ausgewählte Farbe gezeigt.

In der Mitte wird jeweils der Teaser-Text der im Hintergrund aktiven Applikation angezeigt. Dies könnte beispielsweise die Mittagsmenu-Applikation sein, was dann dazu führt, dass der entsprechend passende Text „Hunger?“ angezeigt wird. Unterhalb dieses Textes befindet sich noch ein Zusatztext, welcher die Passanten dazu animieren soll, sich der Wall zu nähern.



Abbildung 50 - Externes Design, Demomodus Teaser-Text

Sobald sich ein Passant der Wall genähert hat und dessen Skelett erkannt wurde, beginnt ein Countdown und das erkannte Skelett wird angezeigt.

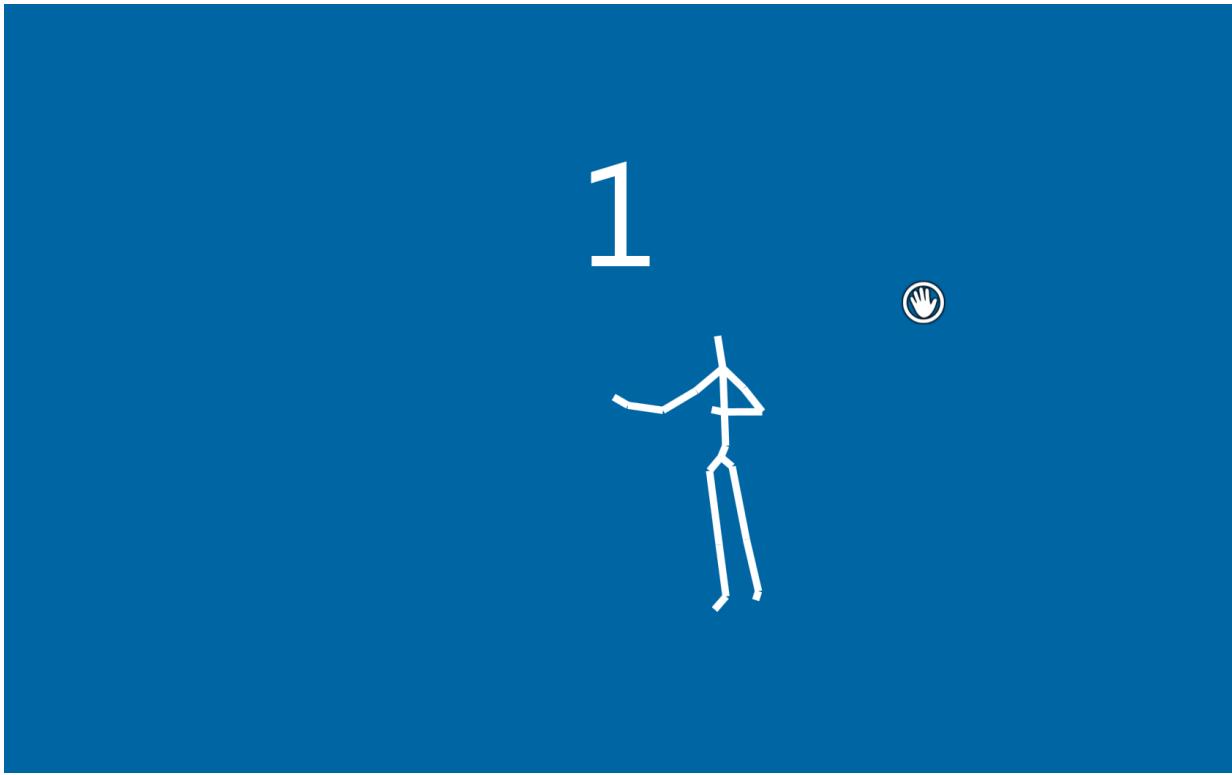


Abbildung 51 - Externes Design, Demomodus Countdown

Der Countdown ist dazu da, dass der Nutzer einerseits eine Rückmeldung auf sein Näherkommen erhält und andererseits darüber informiert wird, wie lange er noch warten muss, bis es weiter geht. Die Skelett-Anzeige wurde nach einem Usability Test (siehe V.8.2.3 Test 3: Reaktion auf Demomodus) implementiert, sie macht dem Nutzer klar, dass er von der Applikation erkannt worden ist.

Ist der Countdown bei 0 angekommen, wird vom Demomodus in den Interaktionsmodus gewechselt. Auch wird dem Nutzer schon zu diesem Zeitpunkt der Handcursor angezeigt, damit dessen Verwendung in der Wartezeit erlernt werden kann.

Entfernt sich ein Nutzer vor Ablauf des Countdowns von der Wall, so wird wieder der Teaser-Text angezeigt.

## V.5.4.6 Guidelines

Die „Microsoft for Kinect Human Interface Guidelines“<sup>28</sup> definieren eine Reihe von Prinzipien. Dieses Dokument war jedoch erst ab dem 21.05.2012 online verfügbar. Die Bachelorarbeit startete aber bereits am 20.02.2012, daher war bis zu diesem Zeitpunkt schon ein Grossteil der Applikation umgesetzt. Trotz dem Fehlen der Guidelines zu Beginn der Bachelorarbeit konnten die Anforderungen und Empfehlungen aus den Guidelines erfüllt werden, was auf die zahlreich durchgeführten Usability Tests (siehe V.8.2 Usability Tests) zurückzuführen ist.

Die Guidelines sind in den nachfolgenden Kapiteln kurz zusammengefasst und es ist beschrieben, wie diese im Projekt umgesetzt wurden.

### V.5.4.6.1 Human Interface Guidelines

#### V.5.4.6.1.1 Best Practices for Designing Interactions

Das Kapitel beschreibt, wie Stimme und Gesten für die Steuerung von Kinect verwendet werden. Unter dem Wort „Gesten“ versteht das Team bestimmte Abläufe von Bewegungen. Die Guidelines bezeichnen aber auch eine simple Handbewegung (nicht bestimmt in welche Richtung) als Geste: „*Basic Gesture Types Gestures can take many forms, from using your hand to target something on the screen, to specific, learned patterns of movement, to long stretches of continuous movement.*“<sup>28</sup>

Im Falle der Videowall werden lediglich „*Innate Gestures*“<sup>28</sup> verwendet, welche wie folgt beschrieben werden: „*Innate gestures are ones that the user intuitively knows or that make sense based on the users' understanding of the world.*“<sup>28</sup>. Dies trifft auf das Konzept „Meine Hand ist die Maus“ (siehe V.8.2.1 Test 1: Wizard of Oz) zu. Weiter ist festgehalten, dass ein visuelles Feedback vorhanden sein soll. Der Nutzer soll unter anderem wissen, ob er von Kinect erkannt wurde, ob er die Kontrolle hat und ob er am richtigen Ort steht. Diese Anforderungen werden alle durch die Darstellung des Skeletts und der Hand gewährleistet.

Auch soll die Verwendung von Kinect aus einer gewissen Distanz geschehen. Die Videowall hat eine imposante Grösse. Damit diese überschaubar ist, muss der Nutzer automatisch ein paar Meter Abstand nehmen.

Gesten sollen immer mit beiden Händen (ob rechts oder links) ausführbar sein. Dies ist mit dem Handcursor möglich.

„*Whether using gesture, or voice, or both, providing good feedback is critical to making users feel in control and helping them understand what's happening.*“<sup>28</sup> – Sobald sich der Handcursor auf einem auswählbaren Element befindet, wird ein Fortschrittskreis um die Hand angezeigt. Auswählbare Elemente sind zudem immer in der Farbe Blau (siehe V.5.4.5 Externes Design) gehalten.

#### V.5.4.6.1.2 Basic Interactions

In diesem Kapitel wird beschrieben, wie Interaktionen gestaltet werden sollen.

Es besagt, dass Interaktionen abbrechbar sein sollen. Bewegt man den Handcursor von einer auswählbaren Schaltfläche weg, bevor der Fortschrittskreis vollständig ausgefüllt ist, verschwindet die Fortschrittsanzeige wieder. Auch das Wechseln vom Demomodus in den Interaktionsmodus (siehe V.6.7 Design des Demomodus) kann abgebrochen werden, indem sich der Nutzer aus dem von Kinect erkennbaren Bereich entfernt.

Im Kapitel steht weiter, dass die Inputmethode möglichst praktisch sein soll für die gestellten Aufgaben. Die Schaltflächen sind gross genug, damit sie problemlos mit dem Handcursor angewählt werden können.

Weiter wird die Zone für das Handtracking angesprochen: „*The most common way of targeting, and the way targeting is done for Kinect on Xbox360 is with a cursor visual that is controlled by hand movement. The simplest way of implementing a cursor is to define a Physical Interaction Zone and just do a direct mapping of the horizontal and vertical position of the hand in the Zone to the screen. This is how the Xbox360 cursor is implemented.*“<sup>28</sup> – dies beschreibt die genau gleiche Idee, wie sie auch für dieses Projekt für das Handtracking (siehe V.6.8 Interaktion durch Handtracking) umgesetzt wurde.

Wie in den Guidelines empfohlen, wurde der Jitter beim Handcursor, also das Ruckeln der Hand auf dem Monitor auch ohne Bewegung des Nutzers, verringert.

Auch wurde für das Konzept „Meine Hand ist die Maus“ ein Cursor in Form einer Hand genommen. Verwendet

<sup>28</sup> [microsoft12.2] Microsoft Corporation, „Kinect for Windows Human Interface Guidelines“, <http://www.microsoft.com/en-us/kinectforwindows/develop/learn.aspx>  
letzter Zugriff 04.06.2012

der Nutzer die linke Hand, so wird der Handcursor als linke Hand dargestellt und umgekehrt. Auch dies wird in den Guidelines vorgeschlagen.

#### V.5.4.6.1.3 Distance-Dependent Interactions

Das Kapitel beschreibt verschiedene Interaktionszonen und was für Interaktionen sich in den jeweiligen Zonen am besten eignen. Für die Videowall wurden nur der „Far(2.0-4.0 Meters)“<sup>28</sup>- und der „Out of Range (>4 Meters)“<sup>28</sup>-Bereich verwendet. Im Bereich mit mehr als 4 Metern Abstand vom Sensor soll der Nutzer animiert werden, näher zu kommen: „Your UI should focus on informing users that there is an interesting interaction available and enticing them to move closer. Visuals must be very large and simple“<sup>28</sup> – Der Demomodus verfügt über eine Hintergrundfarbe und einen grossen Teaser-Text (siehe V.5.4.5 Externes Design) und erfüllt daher dieses Kriterium.

#### V.5.4.6.1.4 Multimodality

Dieses Kapitel beschäftigt sich mit dem Einsatz von mehreren Inputmethoden – einerseits die Stimme, andererseits Gesten. Da für die Steuerung der Videowall-Applikation lediglich Gesten verwendet werden, wird auf dieses Kapitel nicht weiter eingegangen.

#### V.5.4.6.1.5 Multiple Users

Das Kapitel „Multiple Users“ beschäftigt sich mit der Möglichkeit mehrere Nutzer zu tracken. Für diese Arbeit wurde das Single „Driver Model“ übernommen: „This model assigns one of the users as the “driver” at any given time and only registers actions taken by that user. The driver role can be selected or transferred in a number of ways, including choosing the first user to engage, or the user that is closer to the sensor. This is one way to avoid conflicting inputs. This model is usually indicated by having visuals that show which person is being tracked and only having one cursor on the screen at any time.“<sup>28</sup>. Durch das dargestellte Skelett ist klar, welche Person die Applikation gerade bedient. Zudem gibt es nur einen Handcursor.

## V.6 Entwurf

<b>V.6.1 Änderungsgeschichte .....</b>	<b>89</b>
<b>V.6.2 Design Entscheide .....</b>	<b>90</b>
V.6.2.1 Frameworks.....	90
V.6.2.1.1 Framework 1: Kinect for Windows SDK .....	90
V.6.2.1.2 Framework 2: OpenNI .....	90
V.6.2.1.3 Framework 3: OpenKinect / libfreenect.....	90
V.6.2.1.4 Nutzwertanalyse.....	90
V.6.2.1.5 Sensitivitätsanalyse.....	91
V.6.2.1.6 Weiteres.....	92
V.6.2.2 PDF-Darstellung.....	92
V.6.2.2.1 Varianten .....	92
V.6.2.2.1.1 Variante 1: PDF direkt darstellen.....	92
V.6.2.2.1.2 Variante 2: Umwandlung zu XPS.....	92
V.6.2.2.1.3 Variante 3: Umwandlung zu Bild .....	92
V.6.2.2.2 Nutzwertanalyse.....	93
<b>V.6.3 Betriebskonzept der Applikation .....</b>	<b>93</b>
<b>V.6.4 Lebenszyklus der Applikation .....</b>	<b>94</b>
V.6.4.1 Übersicht.....	94
V.6.4.2 Startup.....	95
V.6.4.3 Laufen.....	96
V.6.4.4 Beenden .....	96
<b>V.6.5 Architektur.....</b>	<b>97</b>
V.6.5.1 Physische Sicht.....	97
V.6.5.2 Logische Sicht.....	97
V.6.5.2.1 Common .....	98
V.6.5.2.2 ResourceLoader .....	98
V.6.5.2.3 Interfaces.....	98
V.6.5.2.4 Data.....	98
V.6.5.2.5 ServiceModels.....	98
V.6.5.2.6 ViewModels.....	98
V.6.5.2.7 Views .....	98
V.6.5.3 Patterns.....	99
V.6.5.3.1 MVVM .....	99
V.6.5.3.2 Inversion of Control / Dependency Injection mit Unity .....	99
V.6.5.3.3 Extension Interface.....	99
V.6.5.4 Prozesse und Threads.....	99
V.6.5.4.1 Allgemein .....	99

V.6.5.4.2	Kinect.....	100
V.6.5.4.3	Plug-ins.....	100
V.6.5.4.4	Echte Parallel Verarbeitung.....	100
<b>V.6.6</b>	<b>Plug-in Framework.....</b>	<b>101</b>
V.6.6.1	Grundlagen.....	101
V.6.6.2	Probleme .....	104
V.6.6.3	Lösung .....	105
V.6.6.3.1	Dynamische Sicht.....	107
<b>V.6.7</b>	<b>Design des Demomodus .....</b>	<b>108</b>
V.6.7.1	Besprechung des Demomodus „Kraftfeld“.....	108
V.6.7.1.1	Fazit.....	109
V.6.7.2	Design des Demomodus „Teaser“.....	109
<b>V.6.8</b>	<b>Interaktion durch Handtracking.....</b>	<b>111</b>
V.6.8.1	Kinect Daten.....	111
V.6.8.2	Handtracking.....	111
V.6.8.3	Anklickbare Elemente.....	112

---

## V.6.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
08.03.2012	1.0	Erste Version des Dokuments	LE
09.03.2012	1.1	Korrekturen und Review Frameworks	DT
02.04.2012	1.2	Interaktion durch Handtracking hinzugefügt	LE
13.04.2012	1.3	Architektur und PDF-Darstellung	CH
13.04.2012	1.4	Review, Konvertierung mit Image Magick	DT
16.04.2012	1.5	Physische und logische Sicht, Auflösung	CH
16.04.2012	1.6	Review Auflösung	DT
24.04.2012	1.7	Review Auflösung, Performance	LE
07.05.2012	1.8	Anpassung aus Code Review 03.05.2012	DT
19.05.2012	1.9	Review Begründung Nutzwertanalyse	DT
29.05.2012	1.10	Miniapps	CH
29.05.2012	1.11	Review Miniapps	DT
30.05.2012	1.12	Architektur	CH
08.06.2012	1.13	Architektur ergänzt, Lebenszyklus der Applikation hinzugefügt	LE
09.06.2012	1.14	Plug-in Möglichkeit, Prozesse und Threads, anklickbare Elemente hinzugefügt, kleine Korrekturen	LE
10.06.2012	1.15	Review Architektur	CH
10.06.2012	1.16	Review und Korrekturen	LE
11.06.2012	1.17	Applikationen	CH
12.06.2012	1.18	Review	DT
14.06.2012	1.19	Review, kleine Korrekturen	LE
14.06.2012	1.20	Korrekturen	CH

## V.6.2 Design Entscheide

### V.6.2.1 Frameworks

Um eine Applikation mit Microsoft Kinect zu entwickeln, stehen die folgenden drei Frameworks zur Verfügung:

- Kinect for Windows SDK
- OpenNI
- OpenKinect

Nachfolgend einige Anmerkungen zu diesen Frameworks.

#### V.6.2.1.1 Framework 1: Kinect for Windows SDK<sup>29</sup>

Das offizielle Kinect Framework von Microsoft für Windows wurde kurz vor Beginn dieser Arbeit, im Februar 2012, in der Version 1.0 herausgegeben. Wenn beachtet wird, dass andere Frameworks schon eher, als Beispiel OpenNI Ende 2010, veröffentlicht wurden, ist dies relativ spät. Entsprechend sind für dieses Framework viel weniger Beispiele und Bibliotheken im Internet zu finden, wobei deren Qualität hoch ist.

Dieses Framework geht durch die Nutzwertanalyse (siehe Unterkapitel V.6.2.1.4 Nutzwertanalyse) klar als Sieger hervor.

#### V.6.2.1.2 Framework 2: OpenNI<sup>30</sup>

Dieses Framework wurde in der Version 1.0.0.23 im Dezember 2010 erstmals freigegeben und konzentriert sich, im Gegensatz zum Microsoft Kinect SDK, nicht nur auf Kinect als Eingabemöglichkeit, sondern allgemein auf Natural User Interfaces (NUI).

Um weitere Geräte anzusprechen und gerätespezifische Funktionen zu implementieren, lässt sich im Framework zusätzliche Middleware einsetzen. So wird mit NiTE<sup>31</sup> von PrimeSense<sup>32</sup> entwickelt, um das Skeletal Tracking durchzuführen.

#### V.6.2.1.3 Framework 3: OpenKinect<sup>33</sup> / libfreenect<sup>34</sup>

OpenKinect ist eine Community, die den libfreenect Treiber entwickelt. Leider gibt es dafür aber keine erweiterten Funktionen wie Gestenerkennung oder Skeletal Tracking.

#### V.6.2.1.4 Nutzwertanalyse

Um herauszufinden, welches dieser drei Framework das passende für die Entwicklung der Videowall - Applikation ist, wurde am 8. März 2012 eine Nutzwertanalyse durchgeführt.

Die Gewichtung der verschiedenen Kriterien lässt sich wie folgt begründen:

- Das Kriterium „Cooperative Support, Weiterentwicklung, Community“ ist sehr wichtig, da bei der Weiterentwicklung der Applikation sich auch die Libraries oder SDKs weiterentwickeln sollen oder dass neue Features zu Verfügung stehen.
- „Windows Integration und Installation“ ist bedingt wichtig, weil es zwar wünschenswert ist, die Integration in Windows und die Installation ohne grossen Aufwand durchführen zu können, jedoch andere Kriterienpunkte entscheidender für die Wahl des Frameworks sind.
- Das Kriterium „Linux / Mac Kompatibilität“ wurde als nicht wichtig eingestuft, da Kinect selbst schon von der Microsoft Corporation ist und es daher sinnvoll ist, mit Microsoft-Technologien zu arbeiten.
- Das Kriterium „Skeletal Tracking Qualität“ ist essentiell für die Interaktion des Nutzers mit der Applikation und erhält daher eine hohe Wichtigkeit.

<sup>29</sup> <http://www.microsoft.com/en-us/kinectforwindows/>

<sup>30</sup> <http://openni.org/>

<sup>31</sup> <http://www.primesense.com/Nite/>

<sup>32</sup> <http://www.primesense.com/>

<sup>33</sup> [http://openkinect.org/wiki/Main\\_Page](http://openkinect.org/wiki/Main_Page)

<sup>34</sup> <https://github.com/OpenKinect/libfreenect>

- Die „Libraries für Gestenerkennung“ sind bedingt wichtig, da Gesten zum jetzigen Zeitpunkt nicht verwendet werden.
- Die „Record / Replay Funktionalität“ ist wichtig, da damit ein Nutzer simuliert werden kann und so die Applikation vereinfacht getestet werden kann.
- Das Kriterium „Dokumentation“ ist wichtig, um die Features Framework zu kennen und zu verstehen.
- Zu den Punkten „Mit Framework realisierte Beispiele und Libraries (Quantität)“ und „Mit Framework realisierte Beispiele und Libraries (Qualität)“ ist zu erläutern, dass Beispiele dem Verständnis helfen, nicht aber notwendig sind und daher nur bedingt wichtig sind. Bei den Beispielen bedingt es nicht nur einer hohen Anzahl sondern auch einer guten Qualität.

Die Evaluation wurde manuell durchgeführt. Die Bewertung der einzelnen Kriterien mittels wenig wichtig (1), bedingt wichtig (3) und sehr wichtig (5) ist selbsterklärend und wird daher nicht begründet.

Nutzwertanalyse: Auswahl Kinect Framework									
Kriterium	Gewichtung	Framework 1		Framework 2		Framework 3			
		Kinect for Windows SDK	OpenNI	Bewertung	Total	Bewertung	Total		
<b>Cooperate Support, Weiterentwicklung, Community</b>	5			5	25	3	15	3	15
<b>Windows Integration und Installation</b>	3			5	15	3	9	3	9
<b>Linux / Mac Kompatibilität</b>	1			1	1	5	5	5	5
<b>C# / .NET Framework / Visual Studio Integration</b>	5			5	25	3	15	1	5
<b>Skeletal Tracking Qualität</b>	5			5	25	3	15	1	5
<b>Libraries für Gestenerkennung</b>	3			3	9	5	15	1	3
<b>Record / Replay Funktionalität</b>	5			3	15	5	25	1	5
<b>Dokumentation</b>	5			5	25	3	15	1	5
<b>Mit Framework realisierte Beispiele und Libraries (Quantität)</b>	3			1	3	5	15	5	15
<b>Mit Framework realisierte Beispiele und Libraries (Qualität)</b>	3			5	15	1	3	1	3
<b>Total Punkte</b>			<b>158</b>			<b>132</b>		<b>70</b>	
<b>Rang</b>				<b>1</b>		<b>2</b>		<b>3</b>	

Bemerkung: Die Gewichtungs- / Bewertungsskala geht von wenig (1), bedingt (3) bis zu sehr wichtig (5).

Tabelle 16 - Nutzwertanalyse: Auswahl Kinect Framework

Aus der Analyse (siehe Tabelle 16 - Nutzwertanalyse: Auswahl Kinect Framework) geht das Framework 1: als Sieger vor dem Framework 2: OpenNI hervor.

#### V.6.2.1.5 Sensitivitätsanalyse

In der Sensitivitätsanalyse wird untersucht, wie stark sich eine Änderung auf das Gesamtergebnis auswirken würde.

Das Framework 3 wird auch bei Änderungen der Bewertung nicht als Sieger hervorgehen.

Zwischen dem Framework 1 und 2 ist der Bewertungsunterschied einiges kleiner. Da das Framework 1 von Microsoft aber über ein ausgeklügeltes, vorhersehendes Skeletal Tracking System<sup>35</sup> verfügt, eine bessere Dokumentation besitzt und perfekte Windows, Visual Studio, C# und .NET Integration bietet, würde sich dieses

<sup>35</sup> <http://www.cs.dartmouth.edu/~cs104/BodyPartRecognition.pdf>

Framework trotz Anpassungen an einzelnen Gewichtungen oder Bewertungen gegenüber dem Framework 2 durchsetzen. Dementsprechend ist diese Nutzwertanalyse nicht sensitiv gegenüber Änderungen.

---

#### V.6.2.1.6 Weiteres

Bei der Nutzwertanalyse wurden zwar möglichst viele nummerisch bewertbare Kriterien untersucht, es fehlt aber noch der persönliche Eindruck. Für das Projekt Videowall fällt die Entscheidung gefühlsmässig auf das Microsoft Framework, da auch die übrigen für die Arbeit eingesetzten Technologien von Microsoft sind und damit gerechnet werden muss, dass andere Frameworks nicht ohne Probleme mit den für das Projekt bereits festgesetzten Microsoft-Technologien zusammenarbeiten können.

Weiter ist in der Bachelorarbeit „Kinect Bodyscanner“ von Felix Egli und Michael Schnyder [egli11] im Kapitel 3.3.1 Kinect Framework auf Seite 30 beschrieben, dass OpenNI für die Arbeit nur eine temporäre Lösung ist. Sie geben an, dass es geplant ist, auf das offizielle Framework von Microsoft zu wechseln, sobald dieses verfügbar ist. Auch diese Aussage spricht klar für das „Kinect for Windows SDK“.

---

#### V.6.2.2 PDF-Darstellung

Die Poster liegen alle im PDF-Format vor. Die unterschiedlichen Möglichkeiten, wie diese Dokumente in der Applikation dargestellt werden können, sind nachfolgend beschrieben. Anschliessend folgt eine Nutzwertanalyse zur Eruierung der am besten geeigneten Darstellungsvariante.

---

##### V.6.2.2.1 Varianten

###### V.6.2.2.1.1 Variante 1: PDF direkt darstellen

Wird diese Variante gewählt, so können die PDF-Dokumente ohne zusätzlichen Umwandlungsaufwand verwendet werden. Jedoch bietet das WPF-Framework keine Komponente an, welche ein PDF-Dokument direkt darstellen kann. Es besteht aber die Möglichkeit, einen Browser einzubinden, welcher zur Darstellung der PDFs den auf dem System installierten PDF-Reader nutzt. Dabei sollten aber die Steuerelemente (Drucken, Verschicken, Zoom, Suche etc.) sowie der Standardhintergrund des PDF-Readers nicht sichtbar sein. Dies kann jedoch nicht über WPF gesteuert werden. Des Weiteren sind die PDF-Dokumente von unterschiedlicher Qualität, was sich zeigt, wenn in der Applikation von einem zum nächsten Dokument navigiert wird. Einige Dokumente benötigen sehr viel Zeit, bis sie geladen sind, andere wiederum haben nur eine kurze Ladezeit.

###### V.6.2.2.1.2 Variante 2: Umwandlung zu XPS

In WPF können XPS Dokumente mittels der DocumentViewer Klasse angezeigt werden. Das Layout des Viewers kann angepasst werden, so können beispielsweise die Steuerelemente ausblendet werden. Bei einem XPS Dokument handelt es sich, wie bei einem PDF-Dokument, um eine Vektorgrafik. Ein XPS Dokument benötigt spürbar Zeit, um geladen zu werden, wenn von einem zum nächsten Dokument navigiert wird. Es ist schlecht möglich, die XPS-Dokumente im Voraus zu laden, da das Laden immer über den GUI-Thread geschieht, dieser aber gleichzeitig auch für Animationen und Ähnliches verwendet wird.

###### V.6.2.2.1.3 Variante 3: Umwandlung zu Bild

Die einfachste Möglichkeit der Darstellung der PDFs besteht darin, die Dokumente in Rastergrafiken umzuwandeln. Diese können mit minimalem Aufwand in eine WPF-Applikation eingebunden werden. Mit Hilfe verschiedenster Open Source Libraries ist die Umwandlung von einem PDF zu einem Bild problemlos möglich. Getestet wurde dies mit ImageMagick<sup>36</sup>, einer frei verfügbaren Software. Sie bietet die Umwandlung von Dokumenten zu Bildern. Eine Rastergrafik benötigt auch deutlich weniger Ladezeit als die Dokumente bei den beiden anderen Varianten. Nachteilig an dieser Lösung ist jedoch, dass das Dokument durch die Umwandlung auf eine Maximalgrösse beschränkt wird und dementsprechend bei einer Änderung der Auflösung der Bildschirme die Bilder neu umgewandelt werden müssen.

---

<sup>36</sup> [www.imagemagick.org](http://www.imagemagick.org)

#### V.6.2.2.2 Nutzwertanalyse

Nachfolgende Nutzwertanalyse, welche am 13. April 2012 durchgeführt wurde, lässt die Variante 3: „Umwandlung zu Bild“ als Sieger hervorgehen.

Die Gewichtung der verschiedenen Kriterien lässt sich wie folgt begründen:

- Das Kriterium „Programmieraufwand“ ist nur bedingt wichtig, da der Aufwand für die Programmierung für eine der drei Varianten nur gering ist im Vergleich zum Aufwand, welcher für das Ausprogrammieren der gesamten Applikation betrieben wird.
- Die „Darstellungsqualität“ ist bedingt wichtig. Das Poster muss lesbar sein. Dazu bedarf es aber keiner überaus hohen Qualität.
- Die „Ladezeit bei Navigation“ darf nicht zu lange dauern, ansonsten würde die User Experience wesentlich darunter leiden. Das könnte dazu führen, dass die Nutzer nicht mehr mit der Wall interagieren wollen. Darum ist dieses Kriterium wichtig.
- Das Kriterium „Abhängigkeit von externen Komponenten“ ist ebenfalls wichtig. Sind Komponenten über eine Zeit lang nicht verfügbar, übt sich dies negativ auf den Betrieb aus.

Die Evaluation wurde manuell durchgeführt. Die Bewertung der einzelnen Kriterien mittels wenig wichtig (1), bedingt wichtig (3) und sehr wichtig (5) ist selbstsprechend und wird daher nicht begründet.

Nutzwertanalyse: PDF-Darstellung		Variante 1: PDF direkt darstellen		Variante 2: Umwandlung zu XPS		Variante 3: Umwandlung zu Bild	
Kriterium	Gewichtung	Bewertung	Total	Bewertung	Total	Bewertung	Total
Programmieraufwand	3	1	3	3	9	5	15
Darstellungsqualität	3	5	15	5	15	3	9
Ladezeit bei Navigation	5	3	15	1	5	5	25
Abhängigkeit von externen Komponenten	5	1	5	3	15	5	25
<b>Total Punkte</b>			<b>33</b>			<b>44</b>	
<b>Rang</b>			<b>3</b>			<b>2</b>	
Bemerkung: Die Gewichtungs- / Bewertungsskala geht von wenig (1), bedingt (3) bis zu sehr wichtig (5).							

Tabelle 17 - Nutzwertanalyse: PDF-Darstellung

#### V.6.3 Betriebskonzept der Applikation

Die Ausführungen zum Betriebskonzept der HSR Videowall sind im Kapitel V.9 Betriebskonzept zu finden.

## V.6.4 Lebenszyklus der Applikation

In diesem Abschnitt wird der Lebenszyklus der Applikation dargestellt. Zu Beginn wird eine Übersicht gezeigt, daraufhin werden die einzelnen Schritte noch genauer beschrieben.

### V.6.4.1 Übersicht

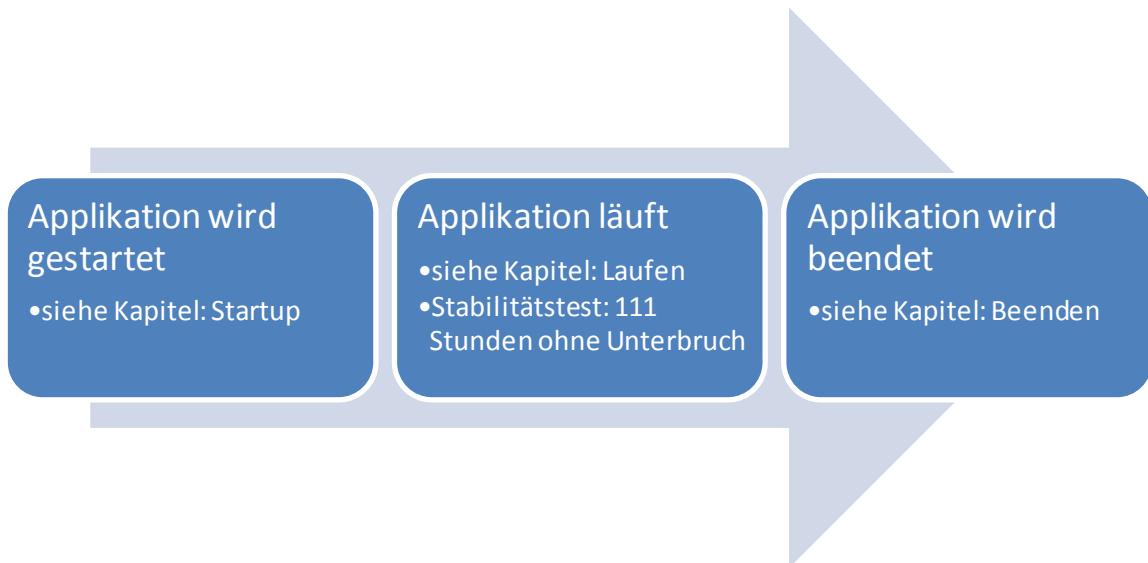


Abbildung 52 - Übersicht des Lebenszyklus

#### V.6.4.2 Startup

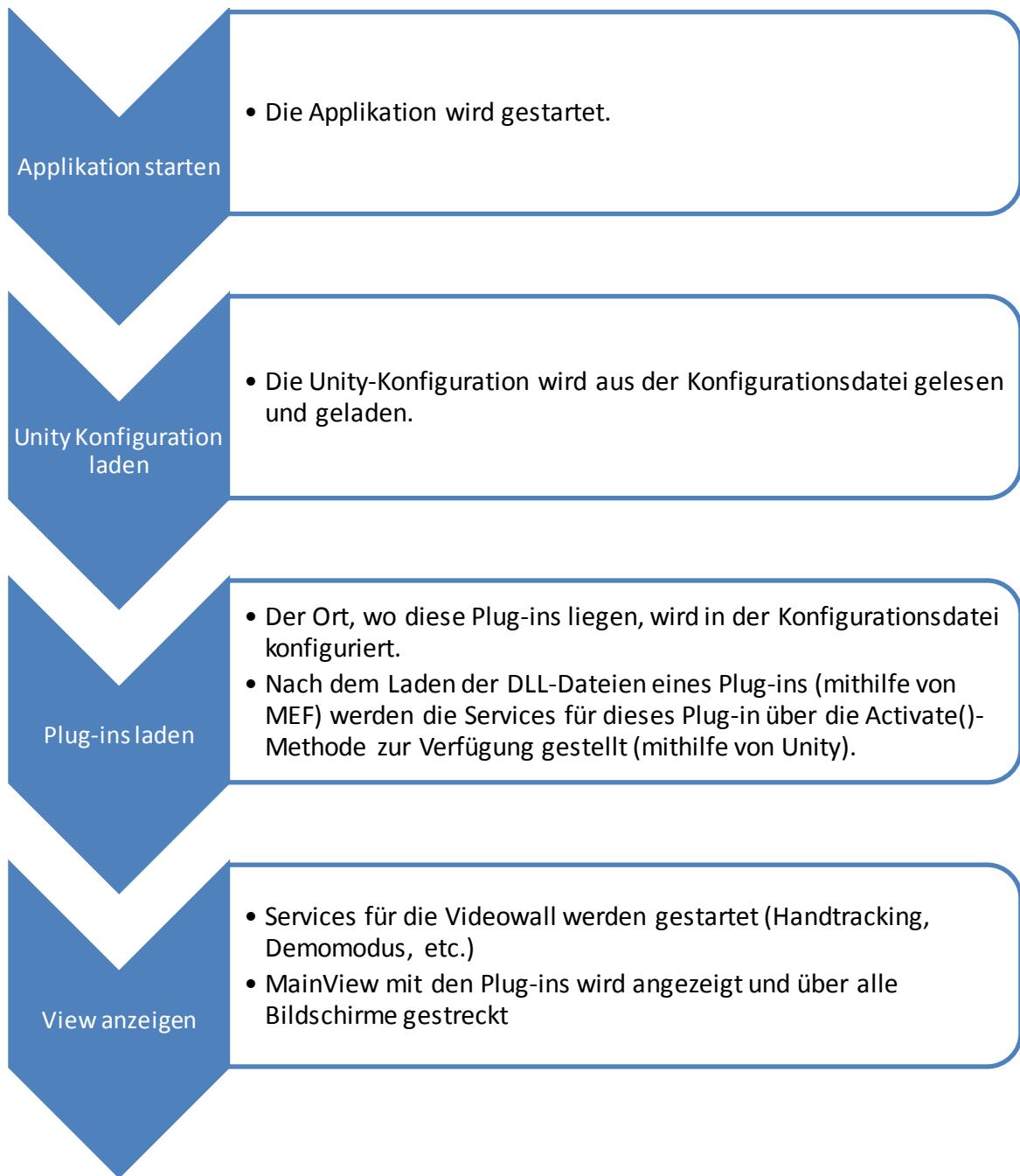


Abbildung 53 - Der Startup Prozess

#### V.6.4.3 Laufen

Der Demomodus ist unter V.6.7.2 Design des Demomodus „Teaser“ genau beschrieben. Dieser funktioniert folgendermassen:



Abbildung 54 - Demomodus Ablauf

Die Plug-ins sind unter V.8.7.2 Plug-ins beschrieben. Die nachfolgende Abbildung zeigt, wie dass die Navigation zwischen den einzelnen Plug-ins möglich ist. Dieser Wechsel ist im Interaktionsmodus durch den Benutzer möglich und wird im Demomodus während der Teaser-Text angezeigt wird automatisch gemacht.



Abbildung 55 - Navigation zwischen einzelnen Plug-in Applikationen

#### V.6.4.4 Beenden

Das Beenden der Applikation wird, wie im letzten Code Review mit Michael Gfeller und Silvan Gehrig (VIII Anhang) besprochen, über den Kill-Command für den aktuell laufenden Prozess realisiert. Somit ist garantiert, dass alle Threads beendet werden.

In Zukunft wäre es schön, wenn diese Prozedur statt mit dem Kill-Command über die Methode Dispose() des Interfaces IDisposable gelöst werden könnte. Dann wäre es möglich, auf dem MainViewModel eine Shutdown-Methode einzurichten, die dann alle Ressourcen freigibt. Dies ist in der aktuellen Version allerdings nicht nötig, da es keine parallel laufenden Threads gibt, die nebenläufig Daten verarbeiten müssen.

## V.6.5 Architektur

### V.6.5.1 Physische Sicht

In der Domain Analyses sind die Tiers der gewünschten Architektur grob beschrieben (siehe V.5.2 Systemübersicht). Da für die Bachelorarbeit ein begrenzter Zeitrahmen zur Verfügung stand, wurde das Hauptaugenmerk auf die „HSR Videowall mit Kinect“ gelegt. Die weiteren Tiers konnten aus zeitlichen Gründen nicht implementiert werden. Abbildung 56 - Systemübersicht zeigt das in der Arbeit entwickelte System.



Abbildung 56 - Systemübersicht

Der Prototyp der Machbarkeitsstudie bietet folgende Funktionen:

- Die wichtigste Funktion des Prototyps ist die dynamische Erweiterbarkeit, welche in Form eines Plug-in Frameworks (siehe V.6.6 Plug-in Framework) umgesetzt wurde.
- Für das Framework wurden zwei Plug-in Applikationen erstellt. Mit der einen Applikation können die Bachelorposter angeschaut werden, in der anderen Applikation kann man sich über Mittagsmenu der Mensa informieren.

### V.6.5.2 Logische Sicht

Die grundlegende Architektur wurde im Team erarbeitet und durch Silvan Gehrig am 02.04.2012 validiert. Die verschiedenen Schichten sind in den nachfolgenden Unterkapiteln beschrieben.

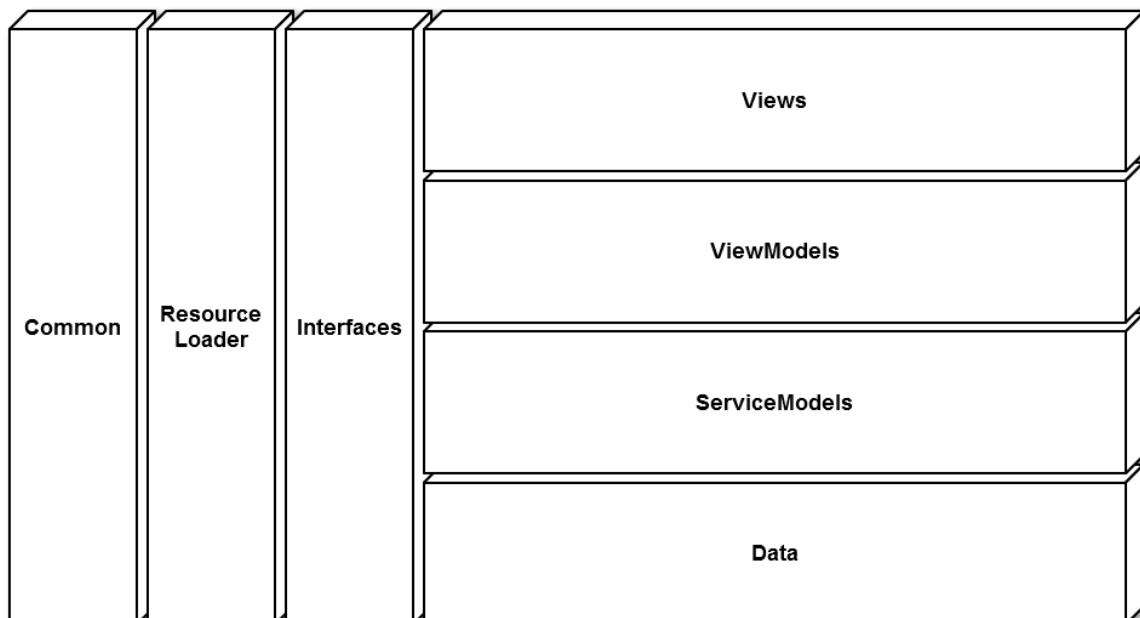


Abbildung 57 - Architektur Diagramm

Es wurde diskutiert, ob zwischen den Schichten ViewModels und Services (nicht im Diagramm ersichtlich) zusätzlich ein Business Layer eingefügt werden sollte. Da die bereitgestellten Daten jedoch nicht manipuliert sondern lediglich angezeigt werden, ist ein Business Layer überflüssig. Dieser würde nur das Service Interface kopieren und dadurch zu einem Durchlauferhitzer werden. Deshalb wurde der Business und Services Layer zu einem gemeinsamen ServiceModels Layer zusammengefasst.

---

#### *V.6.5.2.1 Common*

Im Common Layer befinden sich Klassen, welche von Klassen aus den meisten anderen Schichten verwendet werden.

---

#### *V.6.5.2.2 ResourceLoader*

In der Schicht ResourceLoader werden Ressourcen, welche für die Videowall benötigt werden, geladen.

---

#### *V.6.5.2.3 Interfaces*

Die Interfaces werden von Klassen eines Plug-ins implementiert. Die Interfaces definieren Elemente, welche jedes Plug-in zu Verfügung stellen muss und noch weitere Interfaces, welche das Plug-in für die Nutzung weiterer Funktionalitäten (z.B. Zugriff auf Skelett-Daten des Kinect Sensors) nutzen kann.

---

#### *V.6.5.2.4 Data*

Der Data Layer regelt die Datenbankanbindung, lädt die benötigten Ressourcen und greift auf die Daten von Kinect zu (beispielsweise Skelett- oder Tiefendaten).

---

#### *V.6.5.2.5 ServiceModels*

Im ServiceModels Layer werden die vom Data Layer erhaltenen Daten in Models gespeichert. Diese werden dann über verschiedene Services den ViewModels zur Verfügung gestellt.

---

#### *V.6.5.2.6 ViewModels*

Die ViewModels stellen die von den Services erhaltenen Daten der View zur Verfügung. Auf diesem Layer befindet sich auch die Implementation des ICommand Interfaces. Diese Funktionen können somit von ViewModels und Views verwendet werden.

---

#### *V.6.5.2.7 Views*

Die Views stellen die Elemente aus den ViewModels grafisch dar.

### V.6.5.3 Patterns

Nachfolgend sind die verwendeten Patterns beschrieben.

#### V.6.5.3.1 MVVM

Das MVVM („Model“, „View“, „ViewModel“) Pattern [microsoft09] wird benötigt, um die View vom Model zu entkoppeln. Deshalb wird als Zwischenglied ein ViewModel erzeugt, das die Commands des GUIs abarbeitet und die verfügbaren Elemente dem GUI zur Verfügung stellt.

Die grundlegende Idee dahinter ist, dass sich das GUI schneller ändert als die Businesslogik und deshalb die zwei Komponenten möglichst stark abtrennen sind. Zusätzlich kann das ViewModel mit Unit Tests geprüft werden.

#### V.6.5.3.2 Inversion of Control / Dependency Injection mit Unity

Damit die Komponenten jederzeit und einfach ausgetauscht werden können, wurde mit Unity Containern<sup>37</sup> gearbeitet, um Inversion of Control durch Dependency Injection zu ermöglichen. So können beispielsweise auf eine einfache Art und Weise Komponenten einer Software ausgetauscht werden, indem die Container ausgetauscht werden. Bei der Videowall wird dies benutzt, um bei der Entwicklung zwischen dem echten Kinect Sensor und einem simulierten Kinect Sensor zu wechseln. Dependency Injection kann auch beim Testen helfen, indem die Mock Objekte beim Unity Container registriert werden.

#### V.6.5.3.3 Extension Interface

Das Extension Interface zeigt auf, wie eine Architektur aufgebaut werden kann, damit Erweiterungen an der Applikation einfach vorzunehmen sind.

Wie das Pattern für die Videowall verwendet wird, kann im nachfolgenden Unterkapitel (V.6.6 Plug-in Framework) nachgelesen werden.

### V.6.5.4 Prozesse und Threads

#### V.6.5.4.1 Allgemein

Grundsätzlich machen Multithreading und Multiprocessing dann Sinn, wenn die Performance einer Applikation erhöht werden soll. Da dies in der jetzigen Version noch nicht elementar ist, laufen alle Verarbeitungen in genau einem Prozess in genau einem Thread, dem GUI-Thread. Quasiparallele Verarbeitungen laufen über die Dispatcher Queue [microsoft12.3]:

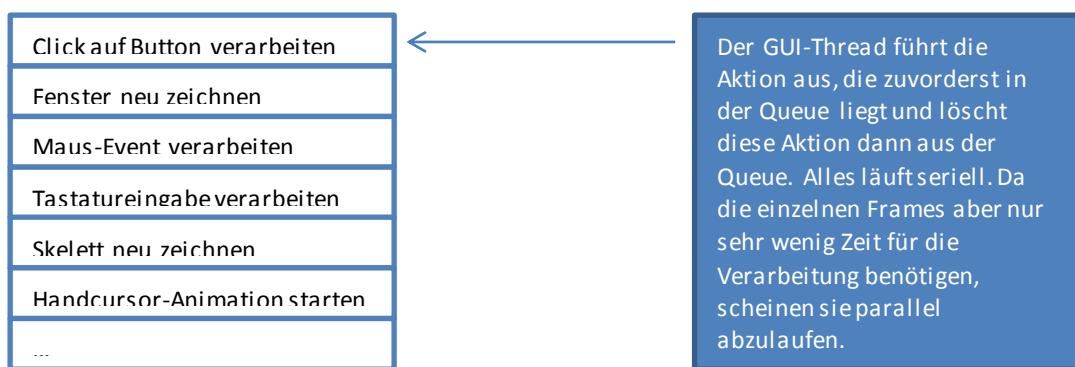


Abbildung 58 - Dispatcher Queue

Eine Ausnahme stellt die Klasse `Kinect.Toolbox.Record.SkeletonReplay` dar, die sich in einer Library befindet und in der Klasse `VideoWall.Data.Kinect.Implementation.AutoPlayFileSkeletonReader` benutzt wird. Da dieses

<sup>37</sup> <http://unity.codeplex.com/>

SkeletonReplay in einem anderen Thread läuft, wird beim Feuern des Events die Verarbeitung sofort in die Dispatcher Queue des GUI-Threads verschoben, damit die Verarbeitung weiter seriell ablaufen kann.

Weiter ist es möglich, dass das .NET Framework oder das Kinect Framework von Microsoft im Hintergrund weitere Threads laufen lässt, diese Events aber durch die Frameworks in die Dispatcher Queue des GUI-Threads verschoben werden.

---

#### V.6.5.4.2 Kinect

Das Kinect SDK von Microsoft bietet auf der Klasse *KinectSensor* einen Event *SkeletonFrameReady*. Dieser wird ausgelöst sobald neue Skelett-Daten bereit sind. Dieser Event tritt bereits im GUI-Thread auf und muss dementsprechend nicht in die Dispatcher Queue verschoben werden.

---

#### V.6.5.4.3 Plug-ins

Da zurzeit auch die Plug-ins im GUI-Thread ablaufen ist es wichtig, dass die Plug-ins vor dem produktiven Einsatz einem Code Review unterzogen werden. So kann sichergestellt werden, dass die Plug-ins das GUI nicht blockieren und dass die Plug-ins keine eigenen Threads oder Prozesse starten.

Die Events, die an die Plug-ins gesendet werden, laufen auch im GUI-Thread ab.

---

#### V.6.5.4.4 Echte Parallele Verarbeitung

Sollte es in Zukunft nötig sein, mehrere Threads oder Prozesse einzusetzen, wird empfohlen, die Events nach der parallelen Verarbeitung sofort wieder in die Dispatcher Queue des GUI-Threads zu verschieben. Mit dieser Massnahme ist Programmieren ohne (Dead)Locks weiterhin möglich.

## V.6.6 Plug-in Framework

### V.6.6.1 Grundlagen

Die erste Frage, die sich bei einem Framework stellt, ist, wie ein Plug-in in das Framework geladen wird (der Extension Point). Microsoft bietet für diesen Zweck das Managed Extensibility Framework (MEF)<sup>38</sup> an.

Technische Details dazu können in der MEF-Dokumentation<sup>39</sup> nachgelesen werden.

Die wichtigste Funktionalität von MEF, die für die HSR Videowall gebraucht wird, ist die folgende:

- Das Schlüsselwort Export markiert eine Klasse (Einstiegspunkt), die ein von einem Framework (Videowall-Applikation) definierten Interface (IApp) implementiert, für den Export aus.
- Das Framework (Videowall-Applikation) importiert alle Klassen, die einen bestimmten Pfad haben (Ordner Extensions) und das Interface IApp implementieren.

```
namespace PosterApp.Main
{
    [Export(typeof(IApp))]
    public class PosterApp : IApp
    {
        public UserControl MainView { get; private set; }
        public string Name { get; private set; }
        public string DemomodeText { get; private set; }

        public PosterApp()
        {
            Name = "Posters";
            DemomodeText = "Neugierig?";
        }

        public void Activate(IVideoWallServiceProvider serviceProvider)
        {
            var unityContainer = new UnityContainer();
            unityContainer.RegisterInstance(serviceProvider);
            MainView = unityContainer.Resolve<PosterView>();
        }
    }
}
```

Abbildung 59 - Poster-Applikation (Extension) wird über [Export(typeof(IApp))] als IApp exportiert

Die obenstehende Abbildung zeigt die Klasse PosterApp, welche das Interface IApp implementiert. Der Ausdruck [Export(typeof(IApp))] markiert die Klasse für den Export.

<sup>38</sup> <http://mef.codeplex.com/>

<sup>39</sup> [microsoft12.1] Microsoft Corporation, Documentation for MEF,  
<http://mef.codeplex.com/documentation>  
letzter Zugriff: 22.05.2012

```

namespace VideoWall.ServiceModel.Apps.Implementation
{
    /// <summary>
    /// Controls the apps.
    /// </summary>
    /// <remarks>
    /// Reviewed by Lukas Elmer, 05.06.2012
    /// </remarks>
    // ReSharper disable UnusedMember.Global
    // Created by unity, so ReSharper thinks this class is unused, which is wrong.
    internal class AppController : IAppController
    // ReSharper restore UnusedMember.Global
    {
        #region Declarations

        #region Properties

        /// <summary>
        /// The video wall applications.
        /// </summary>
        public IEnumerable<IApp> Apps { get { return _extensionFolders.
            Where(ef => ef.IsLoaded()).Select(ef => ef.App); } }

        #endregion

        #region Constructors / Destructor

        #region Methods

        /// <summary>
        /// Loads the apps.
        /// </summary>
        private void LoadApps()
        {
            foreach (var extensionFolder in _extensionFolders)
            {
                _extensionManager.Init(extensionFolder);
                extensionFolder.App.Activate(
                    new ProductionVideoWallServiceProvider(extensionFolder, _player));
            }
        }

        #endregion

    }
}

```

Abbildung 60 - AppController koordiniert den Import der Apps

Der Import der Apps wird vom AppController koordiniert.

```

    ...internal class ExtensionFolder
    ...
    {
        #region Properties

        /// <summary>
        /// Gets the directory where the extension should be found.
        /// </summary>
        public DirectoryInfo Directory { get; private set; }

        /// <summary>
        /// The video wall applications.
        /// </summary>
        [Import]
        public IApp App { get; private set; }

        #endregion

        #region Constructor / Destructor

        /// <summary>
        /// Initializes a new instance of the <see cref="ExtensionFolder" /> class.
        /// </summary>
        /// <param name="directory">The directory.</param>
        public ExtensionFolder(DirectoryInfo directory)
        {
            Directory = directory;
        }

        #endregion

        #region Methods

        /// <summary>
        /// Determines whether this instance is loaded.
        /// </summary>
        /// <returns><c>true</c> if this instance is loaded; otherwise, <c>false</c>.</returns>
        public bool IsLoaded()
        {
            return App != null;
        }

        #endregion
    }
}

```

Abbildung 61 - Der ExtensionFolder (Videowall-Applikation) importiert über das Attribut [Import] die Klassen, die das Interface IApp implementieren und sich in einem bestimmten Ordner (Directory) befinden.

Der Ausdruck [Import] im Framework (Videowall-Applikation) importiert die Klasse, welche das Interface IApp implementieren und sich in einem bestimmten Ordner befindet.

```

namespace VideoWall.ServiceModels.Apps.Implementation
{
    /// <summary>
    /// The extension manager is responsible to load a extension from a specific folder
    /// </summary>
    /// <remarks>
    /// Reviewed by Lukas Elmer, 05.06.2012
    /// </remarks>
    internal class ExtensionManager
    {
        #region Methods

        /// <summary>
        /// Inits the specified application with extension.
        /// </summary>
        /// <param name="extensionFolder">The application with extension.</param>
        public void Init(ExtensionFolder extensionFolder)
        {
            PreOrPostCondition.AssertNotNull(extensionFolder, "extensionFolder");
            var container = new CompositionContainer(
                new DirectoryCatalog(extensionFolder.Directory.FullName));
            container.ComposeParts(extensionFolder);
        }

        #endregion
    }
}

```

Abbildung 62 - Der ExtensionManager führt den Import schliesslich mithilfe von MEF aus

Der ExtensionManager führt den Import des Plug-ins schliesslich mithilfe von MEF<sup>39</sup> aus.

### V.6.6.2 Probleme

Beim Entwickeln eines Frameworks ist oftmals nicht vorhersehbar, wie dieses in der Zukunft aussehen wird, da sich die Anforderungen an das Framework stetig ändern. Würde nur ein einziges Interface (IApp), über das die Services des Frameworks angesprochen werden können, zur Verfügung gestellt, so müsste sich dieses ständig ändern. Folglich müssten die Plug-ins, zum Beispiel die PosterApp (siehe Unterkapitel V.6.6.1 Grundlagen), nach jeder Änderung am Interface (IApp) neu kompiliert werden. Mit nur einem Interface ist es also schwierig, den Plug-ins neue Funktionalität zur Verfügung zu stellen.

Ein weiteres Problem eines einzigen Interfaces ist, dass dieses beliebig gross werden kann und dadurch die Kopplung steigt und die Kohäsion sinkt, was sehr unschön ist.

Das anfängliche IApp Interface wurde folgendermassen implementiert:

```

namespace Interfaces
{
    public interface IApp
    {
        UserControl.MainView { get; }
        string Name { get; }
        string DemomodeText { get; }

        // Directory to store files
        string ResourceDirectory { get; }

        // Kinect based events
        event KinectChangedEventArgs KinectChanged;
        event SkeletonChangedEventArgs SkeletonChanged;
        event DepthImageChangedEventArgs DepthImageChanged;

        // Mouse position changed event
        event MousePositionChangedEvent MousePositionChanged;

        // Called after construction
        void ActivateDatabase(IAppDatabase appDatabase);

        // .... in future, this WILL change!
    }
}

```

Abbildung 63 - Anfängliche Implementation des Interfaces IApp

Wie in Abbildung 63 - Anfängliche Implementation des Interfaces IApp erkennbar ist, ist das Interface relativ gross und stellt verschiedenste Services zur Verfügung, die nichts miteinander zu tun haben. Beispielsweise das ResourceDirectory Property, welches die Plug-in-Dateien zur Verfügung stellt oder der SkeletonChangedEvent, der vom Framework aufgerufen werden soll, sobald sich das Skelett verändert hat.

Ändert sich dieses Interface, beispielsweise durch Hinzufügen neuer Funktionalität, müssen auch immer alle Plug-ins neu kompiliert werden. Dies ist suboptimal, speziell dann, wenn die Plug-ins von verschiedenen Personen gewartet werden.

---

### V.6.6.3 Lösung

Die Lösung ist an das Extension Interface [schmidt00] angelehnt. Es bietet einen Ansatz, das Problem des ständig ändernden Interfaces zu lösen. In der Videowall-Applikation wurde das Extension Interface in abgeänderter Form angewendet, ohne die Vererbung des Root Interfaces. Zusätzlich wurde Unity<sup>37</sup> verwendet um die Factory aus dem Extension Interface zu ersetzen. Die gegenwärtige Implementation des Interfaces sieht folgendermassen aus:

```

namespace VideoWall.Interfaces
{
    /// <summary>
    /// This is the entry point for the framework.
    /// Every application must implement that interface.
    /// </summary>
    public interface IApp
    {
        /// <summary>
        /// Gets the main view.
        /// </summary>
        UserControl MainView { get; }

        /// <summary>
        /// Gets the name.
        /// </summary>
        string Name { get; }

        /// <summary>
        /// Gets the demomode text.
        /// </summary>
        string DememodeText { get; }

        /// <summary>
        /// Loads the app. At this place, the app can load application specific services.
        /// </summary>
        /// <param name="videoWallServiceProvider"> The app info. </param>
        void Activate(IVideoWallServiceProvider videoWallServiceProvider);
    }
}

```

Abbildung 64 - Das IApp Interface

Das IApp Interface (siehe Abbildung 64 - Das IApp Interface) bietet einen Einstiegspunkt. Da jede Applikation dieses Interface implementiert, sind hier nur die Anforderungen beschrieben, die jede Applikation anbieten muss. Speziell ist die Methode Activate, die auf jeder vom Framework zu ladenden Extension genau einmal aufgerufen wird (siehe auch Dependency Injection, [eilbrecht07]). In dieser Methode kann das Plug-in über das IVideoWallServiceProvider-Objekt weitere Services anfordern:

```

namespace VideoWall.Interfaces
{
    /// <summary>
    /// Provides services for the video wall, like hand tracking,
    /// skeleton tracking or a file service
    /// </summary>
    public interface IVideoWallServiceProvider
    {
        /// <summary>
        /// Gets an implementation of the interface
        /// <typeparam name="T"> </typeparam>
        /// which is provided by the video wall
        /// </summary>
        /// <typeparam name="T"> </typeparam>
        /// <returns> </returns>
        T GetExtension<T>() where T : IVideoWallService;
    }
}

```

Abbildung 65 - Durch den IVideoWallServiceProvider können weitere Extensions geladen werden

Über die Methode GetExtension des IVideoWallServiceProviders aus obiger Abbildung kann das Plug-in weitere Services (IVideoWallService) anfordern.

Das IVideoWallServiceInterface ist ein Marker-Interface. Es ist nicht vorgesehen, dass Applikationen weitere Plug-ins registrieren können. Dies stellt den Hauptunterschied zum Extension Interface Pattern dar.

### V.6.6.3.1 Dynamische Sicht

Nachfolgend ein Sequenzdiagramm, welches den Ablauf des Ladens und Aktivierens der Applikationen durch das Framework (Videowall-Applikation) veranschaulicht.

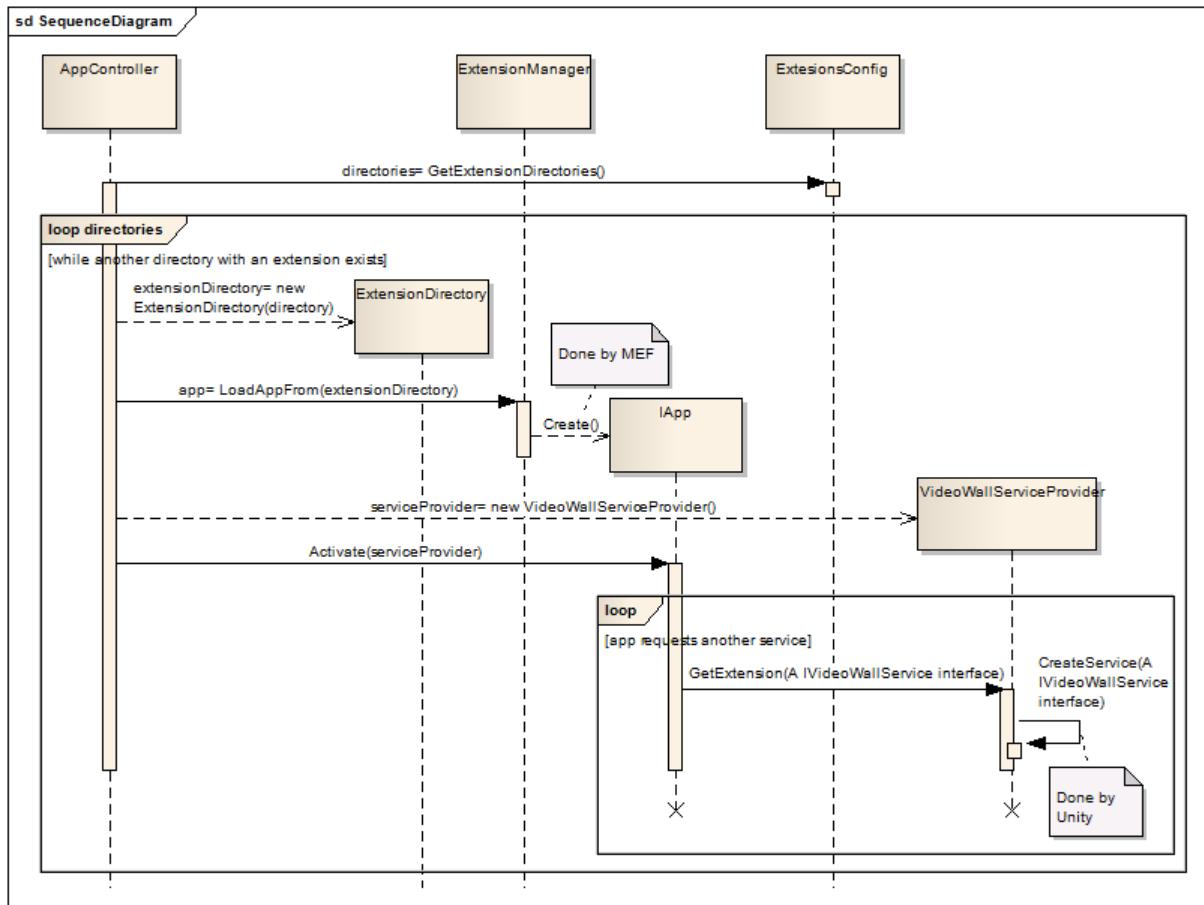


Abbildung 66 - Sequenzdiagramm, Ablauf des Ladens und Aktivierens von Applikationen durch das Framework

## V.6.7 Design des Demomodus

Die für den Demomodus gesammelten Ideen sind im Kapitel (siehe V.5.4.2 Demomodus Link Domain Analyse Demomodus) zu finden.

### V.6.7.1 Besprechung des Demomodus „Kraftfeld“

Am 07.05.2012 besprach das Team, wie bei der Umsetzung des ausgewählten Demomodus „Kraftfeld“ vorgegangen werden soll. Es handelt sich hierbei um den Demomodus, bei dem durch Vorbeilaufen die über alle Monitore verteilten Objekte (z.B. kleine Stücke eines Posters) bewegt werden können (siehe Domain Analyse, Demomodus, Sammlung der Ideen, Beschreibung zu Idee Nummer 4).

Die Abbildung 67 - Teilaufgaben des Demomodus "Kraftfeld" zeigt, dass die Applikation aus sechs Teilaufgaben bestehen müsste. Der erste Punkt ist das Generieren von Screenshots (1), welche dann in Teilchen zerschnitten werden. Weiter müssen diese Teilchen über den ganzen Bildschirm verteilt angezeigt werden(2). Damit bereits hier Dynamik im Spiel ist, benötigt jedes einzelne Teilchen eine Grundanimation (z.B. eine leichte Hin- und Herbewegung). Der dritte Punkt ist das Zusammenfügen der Teilchen (3) zu einem Ganzen, dem Ursprungsbild. Als Nächstes müssen die Bewegungen der Teilchen (4), die durch das Passieren der Videowall ausgelöst wird, festgelegt und implementiert werden. Dazu mehr im nachfolgenden Abschnitt, der die Abbildung 68 - Ideen zur Bewegungsart der Teilchen beschreibt. Abschliessend folgt das Wechseln vom Demomodus in den Interaktionsmodus (5) und umgekehrt (6).

Ein Usability Test und das Umsetzen der allfällig dadurch entstandenen Verbesserungsansätze runden die Implementation ab.

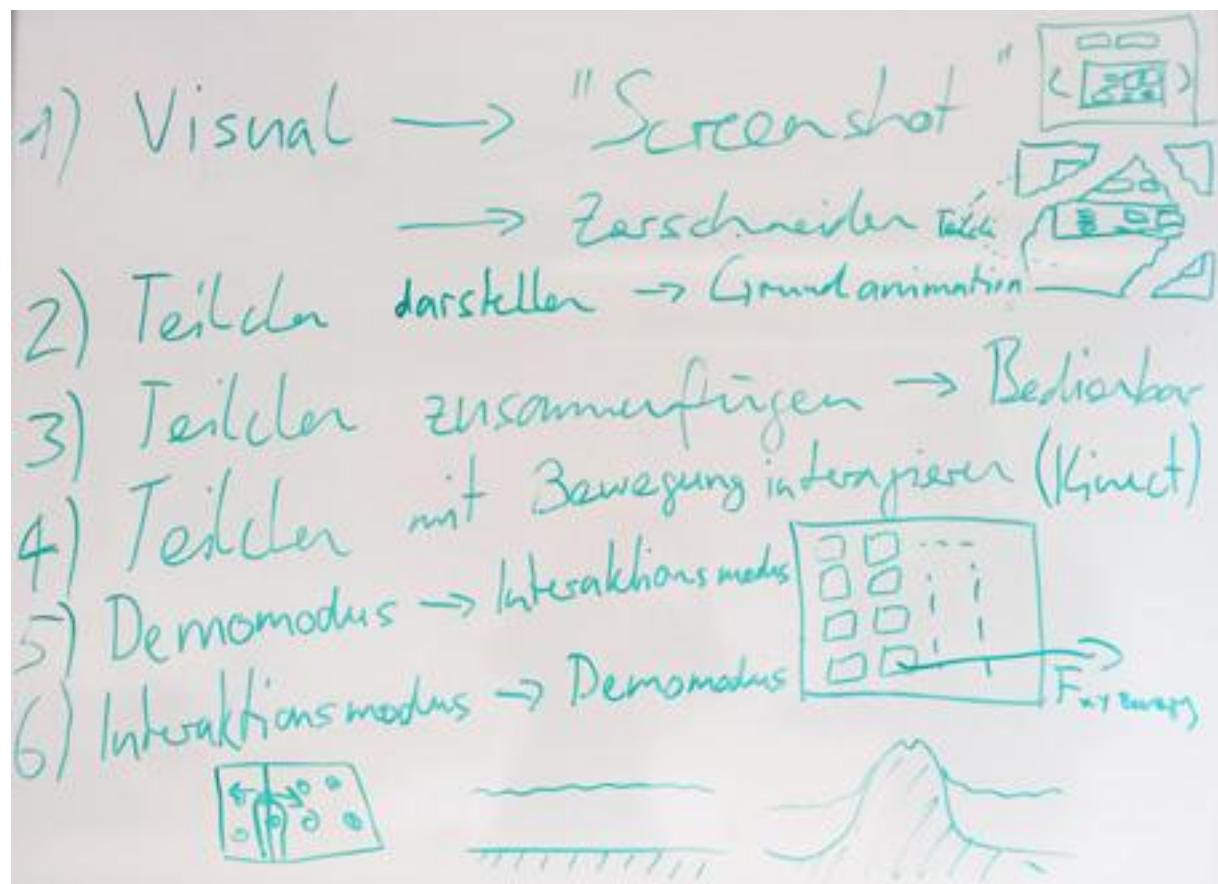
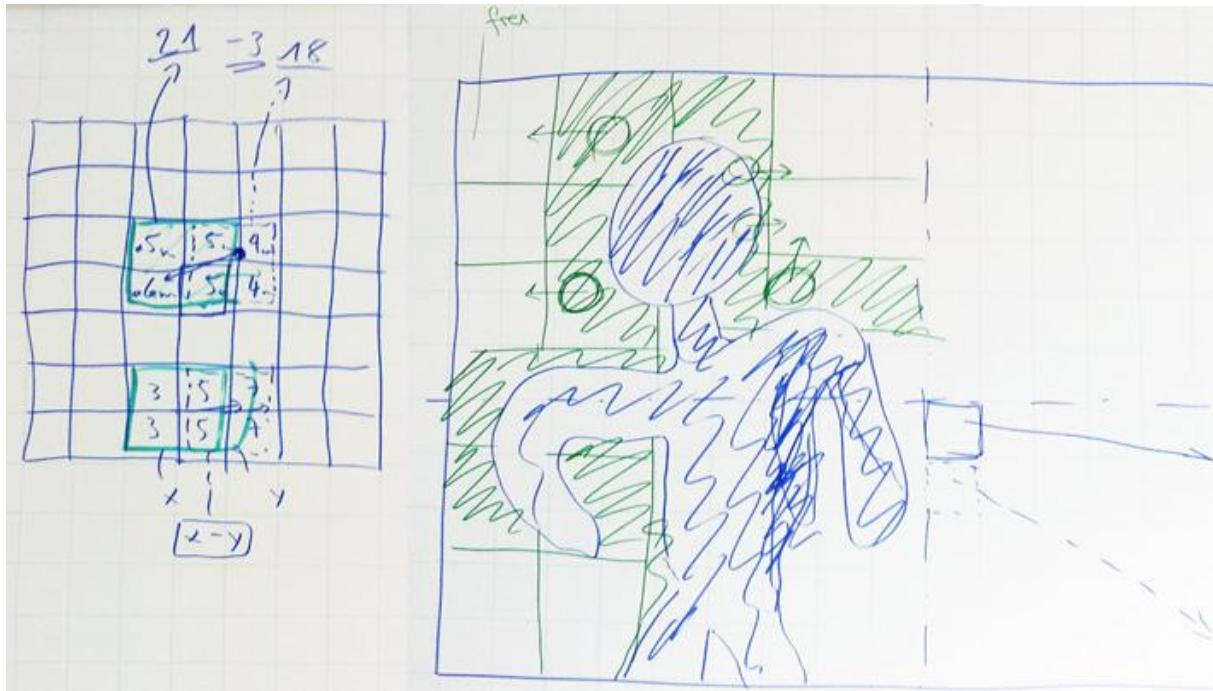


Abbildung 67 - Teilaufgaben des Demomodus "Kraftfeld"

Die Umsetzung des im obigen Abschnitt aufgelisteten Punktes Nummer 4 benötigt mathematische Vorarbeit. Die Abbildung 68 - Ideen zur Bewegungsart der Teilchen zeigt Ansätze, wie die Bewegungen der Teilchen berechnet werden könnte.



**Abbildung 68 - Ideen zur Bewegungsart der Teilchen**

Im oberen Teil der Abbildung soll mit den Abstandsangaben, die vom Tiefensensor der Kinect erfasst werden, gearbeitet werden. Der Tiefensensor misst für jedes Pixel, wie weit der darauf zu sehende Mensch oder Gegenstand vom Sensor entfernt ist. Mit Hilfe der daraus gewonnenen Zahlwerte könnten nun Geradensteigungen und Vektorrichtungen für die Bewegung der Teilchen, welche auf den Wall verteilt dargestellt werden, ausgerechnet werden.

Im unteren Teil der Abbildung wird im Hintergrund ein feines Raster über die Monitore gelegt. Ein Quadrat dieses Rasters beinhaltet mehrere Pixel. Wird nun das Skelett des Benutzers erkannt, so werden die Rasterquadrate, von denen ein oder mehrere Pixel im Bereich des Skeletts sind, als besetzt markiert (grün schraffierte Fläche). Teilchen, die sich auf diesen besetzten Rasterquadraten befinden, suchen sich nun den kürzesten Weg auf ein freies Quadrat. Teilchen, welche bereits auf einem freien Quadrat dargestellt werden, bewegen sich nicht.

#### V.6.7.1.1 Fazit

Beim Notieren der Teilaufgaben, welche alle erledigt werden müssen, um den Demomodus umsetzen zu können, wurde dem Team bewusst, dass die verfügbare Zeit nur für die Implementation des einfacheren Demomodus (Idee 2, siehe Domain Analyse Auswahl der besten Idee für den Demomodus) reicht. Das Team kam daher zum Schluss, den Demomodus „Kraftfeld“ aus zeitlichen Gründen nicht umzusetzen. Erklärungen zum alternativ umgesetzten Demomodus „Teaser“ sind im nachfolgenden Kapitel (V.6.7.2 Design des Demomodus „Teaser“) zu finden.

#### V.6.7.2 Design des Demomodus „Teaser“

Auch dieser Demomodus wurde in einzelne Teilaufgaben unterteilt. Wie bereits im Unterkapitel V.6.7.1 Besprechung des Demomodus „Kraftfeld“ erwähnt ist, muss es möglich sein, zwischen dem Interaktions- und Demomodus zu wechseln. Sobald der Demomodus angezeigt wird, soll der Hintergrund auf eine zufällig ausgewählte Farbe gesetzt werden. Zudem soll auch ein Teaser-Text zur jeweilig im Hintergrund aktiven App angezeigt werden. Dabei könnte es sich, wie in der ersichtlich, um einen Text wie „Hunger? – Dann stell dich hier hin“ handeln.

Die nachfolgende Abbildung 69 - Zustandsdiagramm Interaktions- und Demomodus zeigt das Zustandsdiagramm, welches den Wechsel vom Interaktionsmodus (Active) in den Demomodus (Teaser) und zurück aufzeigt.

Zu Beginn befindet sich die Applikation im Interaktionsmodus (Active). Solange ein Skelett erkannt wird, bleibt die Applikation in diesem Status. Wird über eine bestimmte Zeit (beispielsweise 10 Sekunden) kein Skelett mehr erkannt, wird in den Demomodus (Teaser) gewechselt. Die Applikation bleibt so lange im Demomodus, bis wieder ein Skelett erkannt wurde. Darauf folgt der Wechsel in den Countdown. Wird hier immer ein Skelett erkannt, so läuft Zähler von 5 Sekunden rückwärts bis auf 0 Sekunden und die Applikation wechselt in den Interaktionsmodus (Active). Falls im Countdown kein Skelett mehr erkannt werden sollte, so wird zurück in den Demomodus (Teaser) gewechselt.

Während dem sich die Applikation im Demomodus befindet, werden im Hintergrund nach Ablauf einer bestimmten Zeit (zum Beispiel 20 Sekunden) die aktuelle Applikation und die Farbe des Demomodus - Hintergrunds gewechselt.

Während dem ganzen Ablauf speichert die Applikation jeweils die Zeit ab, bei der zuletzt ein Skelett erkannt wurde. Somit ist es möglich, die Zeitspanne, während der kein Skelett erkannt wurde, zu messen.

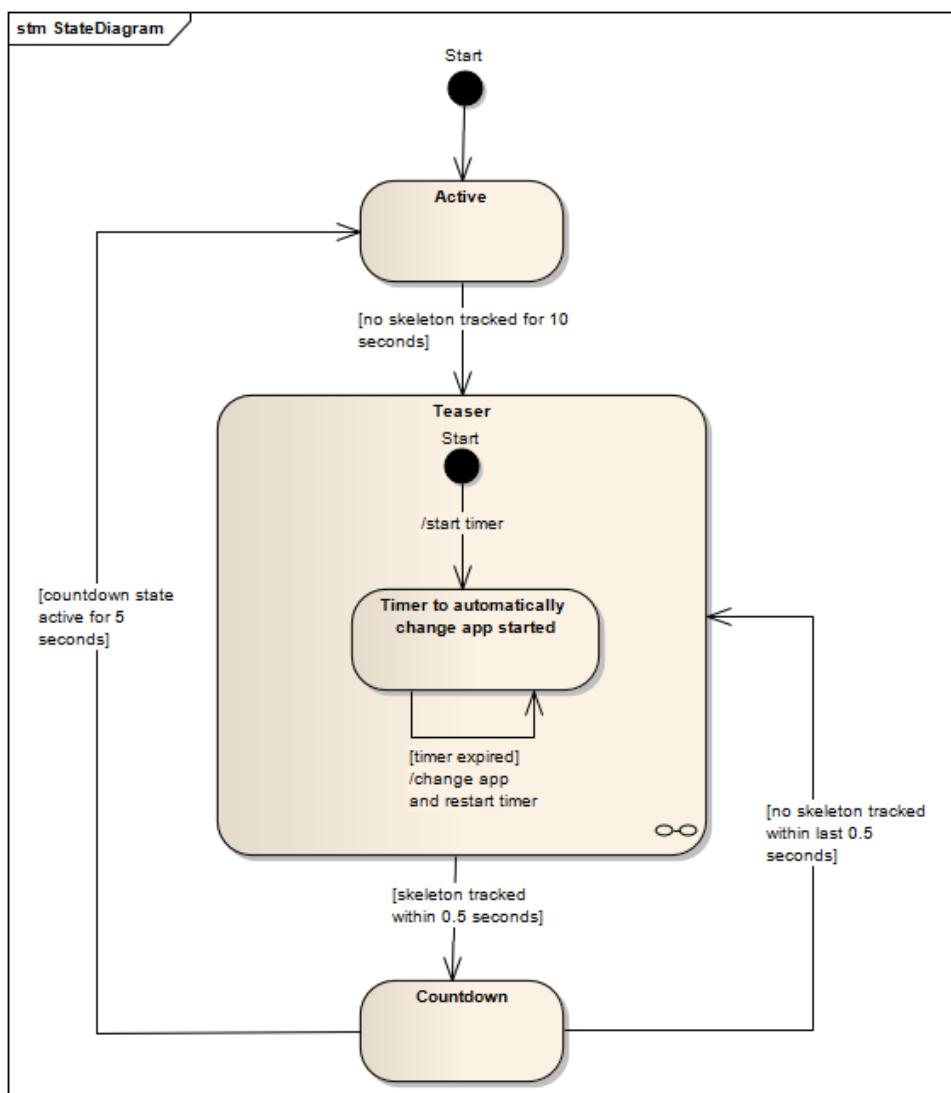


Abbildung 69 - Zustandsdiagramm Interaktions- und Demomodus

## V.6.8 Interaktion durch Handtracking

Mittels eines Usability Tests wurde evaluiert, dass der Benutzer mithilfe der Hand die Applikation bedienen kann („Meine Hand ist die Maus“) (siehe). Wie das genau funktioniert, wird in diesem Kapitel erläutert.

### V.6.8.1 Kinect Daten

Eines der wichtigsten Features des Kinect SDK ist das sogenannte Skeletal Tracking. Hierbei wird mit Hilfe der Sensoren (Tiefensensor, Bildsensor, Infrarotsensor) versucht, ein menschliches Skelett zu erkennen, und zwar in Echtzeit. Es ist möglich, gleichzeitig von zwei Personen das Skelett anzuzeigen. Für das Handtracking auf der Videowall ist aber nur das Tracken eines Skeletts vorgesehen.

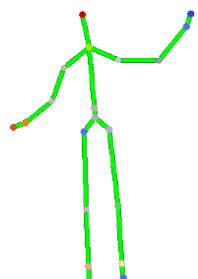


Abbildung 70 - Beispiel eines Skeletts

### V.6.8.2 Handtracking

Wie aus der obigen Abbildung (Abbildung 70 - Beispiel eines Skeletts) ersichtlich ist, besteht das Skelett aus einzelnen Punkten, welche die Gelenke wie Schultern oder Knie der verfolgten Person darstellen. Es kann daher die rechte Hand eruiert und dargestellt werden.

Die Position der Hand des Benutzers muss auf dem Bildschirm zeitgleich nachgestellt werden. Damit sich der Benutzer der Applikation nicht zu viel und weit bewegen muss, wird eine Grenze für das Tracken der Hand festgelegt. Das sieht schematisch folgendermassen aus:

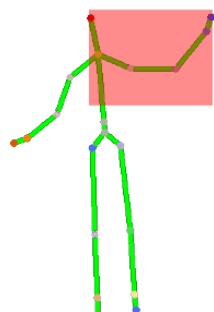


Abbildung 71 - Skelett mit Zone (rot) für das Handtracking

Der rote Bereich stellt den Bildschirm dar. Wenn nun der Benutzer seine Hand in der oberen rechten Ecke der roten Zone bewegt, so wird diese oben rechts auf dem Bildschirm angezeigt, wie Abbildung 72 - Beispiel Monitor mit Handtracking zeigt. Dort, wo sich die Hand im roten Bereich befindet, wird sie folglich auf dem Bildschirms angezeigt. Befindet sich die Hand ausserhalb des roten Bereichs, so wird sie (analog zur Maus auf dem Bildschirm) am Rand des Bildschirms oder gar nicht angezeigt.

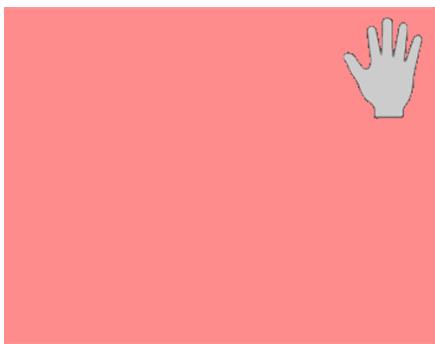


Abbildung 72 - Beispiel Monitor mit Handtracking

Wie die konkreten Masse des Bereichs für das Handtracking lauten und wo sich der Bereich genau befindet, ist in der Entwicklungsphase noch detailliert zu definieren und kann direkt dem Quellcode entnommen werden. Grundsätzlich ist klar, dass sich der Bereich über der Hüfte des Skelettes befinden und etwa bis zur Körpermitte gehen wird. Ebenfalls wird der Bereich nicht weit über die Position des Kopfes hinausragen.

#### V.6.8.3 Anklickbare Elemente

Für eine erste Version der Videowall, welche noch nicht mit Gesten gesteuert werden kann, ist es notwendig, dass gewisse Elemente angeklickt werden können, wie zum Beispiel das Menu zum Navigieren oder die Pfeile zum Browsen der Poster. Da diese Elemente alle mit Buttons realisiert wurden, müssen diese somit anklickbar sein. Falls neue Buttons hinzugefügt werden, sollen diese Buttons ebenfalls anklickbar sein.

Um also diese anklickbaren Elemente zu suchen, wird nach dem Starten der Applikation und den Plug-ins der gesamte Visual Tree<sup>40</sup> nach Buttons durchsucht und in einer Liste gespeichert. Sobald sich dann der Handcursor bewegt wird, wird durch die Liste der Buttons iteriert und untersucht, ob sich der Handcursor über einem Button befindet. Falls ja, wird ein Timer gestartet, der auf diesen Button klickt, sobald eine bestimmte Dauer (ca. 1.5 Sekunden) abgelaufen ist.



Abbildung 73 - Handcursor auf nicht anklickbarem Element

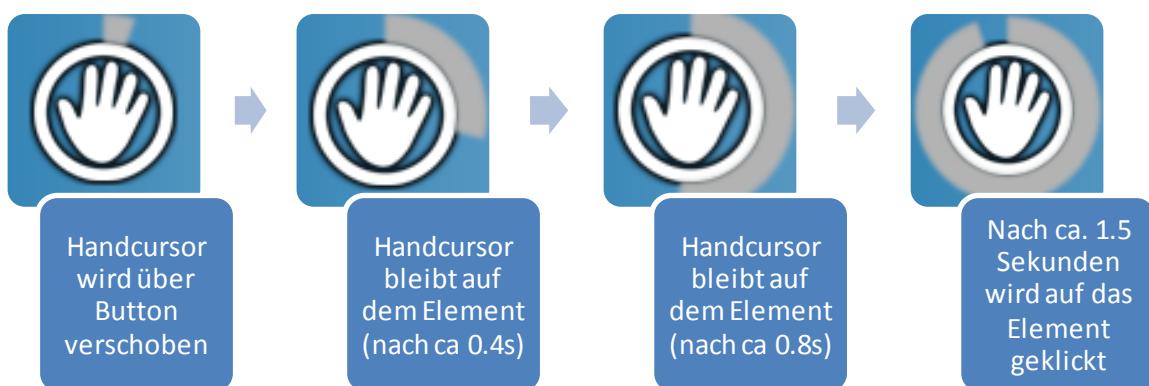


Abbildung 74 - Ablauf eines Klicks auf einen Button

<sup>40</sup> [http://msdn.microsoft.com/en-us/library/ms753391.aspx#two\\_trees](http://msdn.microsoft.com/en-us/library/ms753391.aspx#two_trees)

## V.7 HSR Videowall Evaluation

<b>V.7.1 Änderungsgeschichte .....</b>	<b>114</b>
<b>V.7.2 Software Evaluation .....</b>	<b>115</b>
<b>V.7.3 Hardware Evaluation .....</b>	<b>115</b>
V.7.3.1 Monitoranzahl und -anordnung .....	115
V.7.3.1.1 Variante A: 3 x 3 55“ Monitore .....	116
V.7.3.1.2 Variante B: 2 x 2 55“ Monitore .....	117
V.7.3.1.3 Variante C: 1 x 6 55“ Monitore .....	118
V.7.3.1.4 Variante D: 2 x 4 55“ Monitore .....	119
V.7.3.1.5 Fazit Monitorkonstellationen .....	120
V.7.3.2 Grafikkarten .....	121
V.7.3.3 Testhardware .....	121
V.7.3.3.1 Performance Tests mit WPF-Applikationen .....	122
V.7.3.3.1.1 Übersicht .....	122
V.7.3.3.1.2 WDDM .....	122
V.7.3.3.1.3 XDDM .....	123
V.7.3.3.1.4 Darstellungsoptionen Poster / PDF .....	123
V.7.3.3.2 Test mit Direct-Applikationen .....	123
V.7.3.3.2.1 Fazit .....	124
V.7.3.3.3 Tests auf abgeänderter Testhardware mit einer Grafikkarte und acht Monitoren .....	125
V.7.3.3.4 Tests mit verkleinertem Video .....	125
V.7.3.3.5 Fazit der durchgeföhrten Tests mit unterschiedlicher Hardwarekonstellation .....	131
<b>V.7.4 Evaluation Mitsubishi Display Wall .....</b>	<b>131</b>
<b>V.7.5 Beschaffungsanalyse .....</b>	<b>132</b>
V.7.5.1 Videowall mit 3 x 3 55“ Monitoren .....	132
V.7.5.1.1 Verwendung von Daisy Chain Board .....	132
V.7.5.2 Videowall mit 3 x 3 46“ Monitoren .....	132
<b>V.7.6 Lesbarkeit der Poster .....</b>	<b>133</b>
V.7.6.1 Prozentuale Lesbarkeit .....	133

---

### V.7.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
04.05.2012	1.0	Erste Version des Dokuments, kopieren der Hardware Texte in dieses Dokument	CH
06.05.2012	1.1	Mitsubishi Display Wall, Review DirectX	CH
06.05.2012	1.2	Review DirectX, Review Mitsubishi Display Wall	DT
18.05.2012	1.3	Review Variante 4, 2 x 4 55" Monitore	DT
18.05.2012	1.4	Dokumentation 2 x 4 Monitore Performance Test	DT
19.05.2012	1.5	Begründung Gewichtung Nutzwertanalyse	DT
21.05.2012	1.6	Review Monitore Performance Test, Begründung Nutzwertanalyse	CH
24.05.2012	1.7	WPF Video Performance Tests hinzugefügt	LE
24.05.2012	1.8	Review Korrekturen Markus Stolze	DT
24.05.2012	1.9	WPF Video Performance Tests ausgearbeitet	LE
25.05.2012	1.10	Software Kapitel hinzugefügt	LE
27.05.2012	1.11	Lesbarkeit L-Poster	CH
27.05.2012	1.12	Review Verkleinertes Video abspielbar	DT
28.05.2012	1.13	Review Lesbarkeit L-Poster	DT
04.06.2012	1.14	Ergänzung Lesbarkeit	DT
05.06.2012	1.15	Beschaffungsanalyse	CH
10.06.2012	1.16	Review und Korrekturen	LE
10.06.2012	1.17	Todos abgearbeitet	LE
12.06.2012	1.18	Review	DT
14.06.2012	1.19	Fazit	CH
14.06.2012	1.20	Review	DT

---

## V.7.2 Software Evaluation

Um die Hardware zu evaluieren wurde folgende Software verwendet:

- Windows 7 64 Bit
- Microsoft .NET 4.0
- Matrox PowerDesk (Display Manager)

Eine genaue Beschreibung des Testsystems ist im Anhang zu finden (VIII Anhang). Weitere Informationen zu den Tools sind im Kapitel V.4.2 Tools zu finden.

---

## V.7.3 Hardware Evaluation

Ein wichtiger Teil dieser Arbeit war die Evaluierung der Zielhardware. Zu Beginn war unklar, ob die Wall aus 3 x 3 55“ Monitoren bestehen soll oder ob sich andere Formate besser eignen würden. Für ein angenehmes Lesen der Poster ist eine möglichst hohe Auflösung wünschenswert. Diese könnte jedoch zu Performance-Problemen führen. Diese wiederum würden sich negativ auf das Nutzererlebnis auswirken. Aus diesem Grund galten auch abzuklären, welche technischen Möglichkeiten es gibt, um mehrere Monitore zusammenzuschliessen und was für eine Auflösung und Performance damit erreicht werden kann.

---

### V.7.3.1 Monitoranzahl und -anordnung

Es ist geplant, die Videowall im Verwaltungsgebäude an der Wand zwischen dem Rektorat und dem Eingang für die Post zu montieren. Die Raumhöhe dieses Gebäudes ist im Vergleich zu anderen Räumen an der HSR eher tief, sie beträgt 2.81 Meter. Daher war es fraglich, ob sich eine grosse Videowall gut in diesen Raum einbringen kann.

Neben dem Finden der passenden Räumlichkeiten ist auch die optimale Anzahl der Bildschirme und deren Anordnung ein wichtiges Thema. Folgende drei Varianten standen zur Diskussion:

- 3 x 3 55“ Monitore
- 2 x 2 55“ Monitore
- 1 x 6 55“ Monitore

An der wöchentlichen Sitzung mit Markus Stolze vom 14.05.2012 wurde diskutiert, ob sich die Performance-Probleme, welche im Kapitel V.7.3.3 Testhardware festgehalten sind, durch die Eliminierung eines Bildschirmes lösen lassen. Werden nur acht Monitore genutzt, so wird nur eine Grafikkarte (die Matrox M9188 mit 8 Anschlüssen, siehe Kapitel V.7.3.2 Grafikkarten) benötigt. Das Ergebnis des Tests ist im Unterkapitel V.7.3.3 Tests auf abgeänderter Testhardware mit einer Grafikkarte und acht Monitoren nachzulesen. Daher stand noch eine vierte Variante zur Diskussion:

- 2 x 4 55“ Monitore

Um eine realistische Einschätzung machen zu können, wie die unterschiedlichen Monitorkonstellationen im für die Videowall vorgesehenen Raum wirken, wurde eine Visualisierung mithilfe eines Hellraumprojektors durchgeführt. Dazu wurden die Seitenverhältnisse der verschiedenen Konstellationen aufgezeichnet und auf eine A4 Folie gedruckt. Zusätzlich wurde ein gewünschtes Anzeigemedium, in diesem Fall ein Poster, ebenfalls auf der Folie platziert. Da sich an der Wand, an welcher die Videowall installiert werden soll, zurzeit noch ein Infostand (Möbel-Elemente mit Broschüren, eine Pinnwand mit Plakaten und ein öffentlich zugänglicher PC) befindet, wurden die Montagevarianten an die dem Infostand gegenüberliegenden Wand projiziert. Der Hellraumprojektor wurde so im Raum platziert, dass die Projektion jeweils soweit vergrößert wurde, dass sie den echten Massen der Monitore entsprach.

#### V.7.3.1.1 Variante A: 3 x 3 55" Monitore

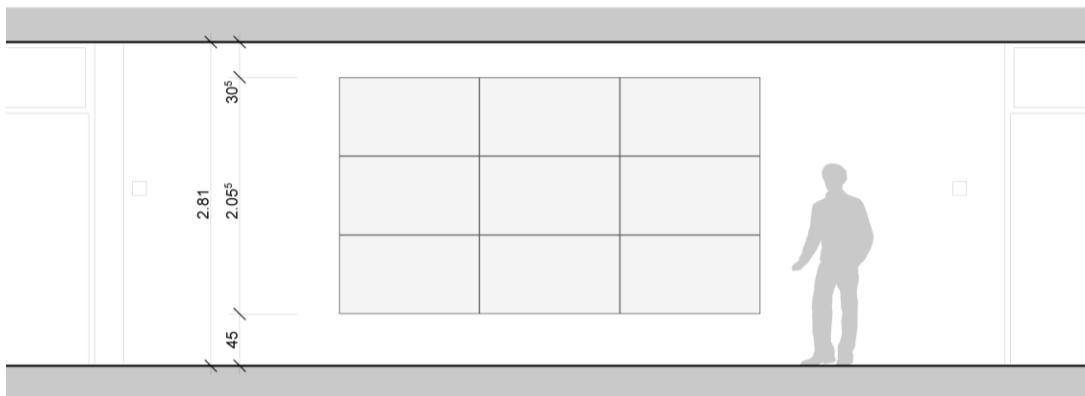


Abbildung 75 - Variante A: 3 x 3 55" Monitore, Ansicht

Wie aus der Aufgabenstellung ersichtlich ist (VIII Anhang), wurde eine Monitorwand mit 3 x 3 55" Monitoren vorgeschlagen. Zu Beginn wurde befürchtet, dass diese durch ihre Abmessungen übermäßig gross in dem Raum erscheinen würde. Auch wurde davon ausgegangen, dass die auf der Videowall dargestellten Elemente nicht auf einen Blick erfasst werden können.

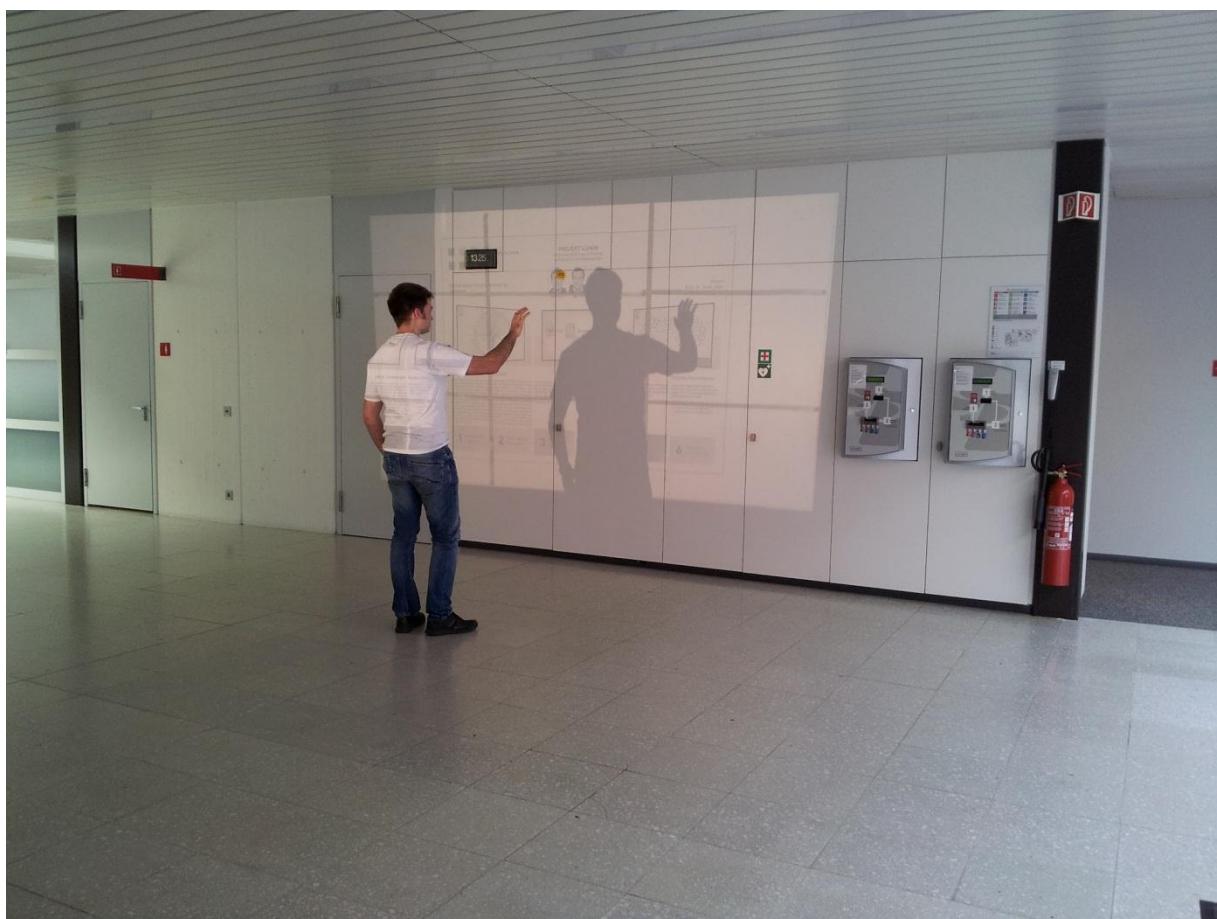


Abbildung 76 - Variante A: 3 x 3 55" Monitore, Hellraumprojektor Test

Durch die anschauliche Projektion konnte sich das Team jedoch vom Gegenteil überzeugen. Das auf der Videowall dargestellte Poster besitzt in dieser Variante eine angenehme Grösse, um die darauf platzierten Texte zu lesen und die Bilder zu betrachten. Auch die Wall wirkt nicht zu massiv, dafür sehr eindrücklich. Das klassische Format mit dem Seitenverhältnis 16:9 eignet sich auch gut für Spiele und die Darstellung von Videos.

#### V.7.3.1.2 Variante B: 2 x 2 55" Monitore

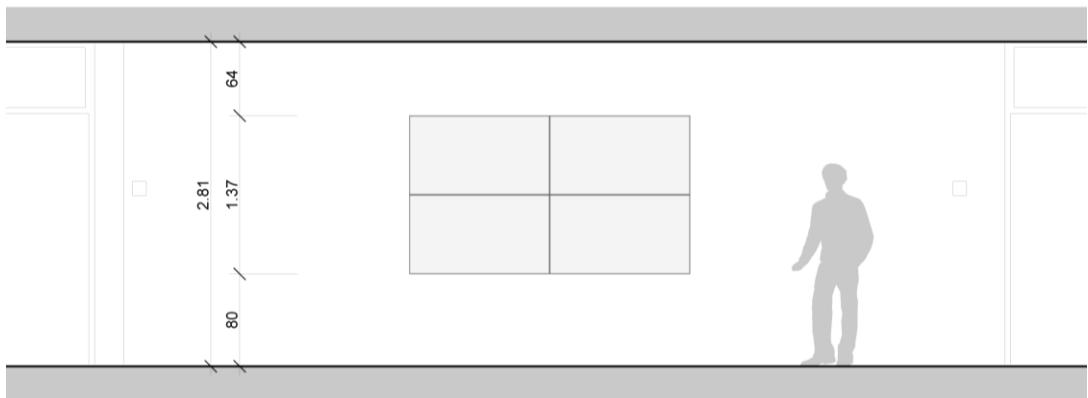


Abbildung 77 - Variante B: 2 x 2 55" Monitore, Ansicht

Wie im Unterkapitel V.7.3.1.1 Variante A: 3 x 3 55" Monitore erwähnt, wurde bei der Variante A davon ausgegangen, dass die Monitorwand im Gebäude 4 an der vorgesehenen Wand zu gross wirken würde. Daher wurde eine kleinere Variante mit 2 x 2 55" Monitoren ebenfalls getestet.

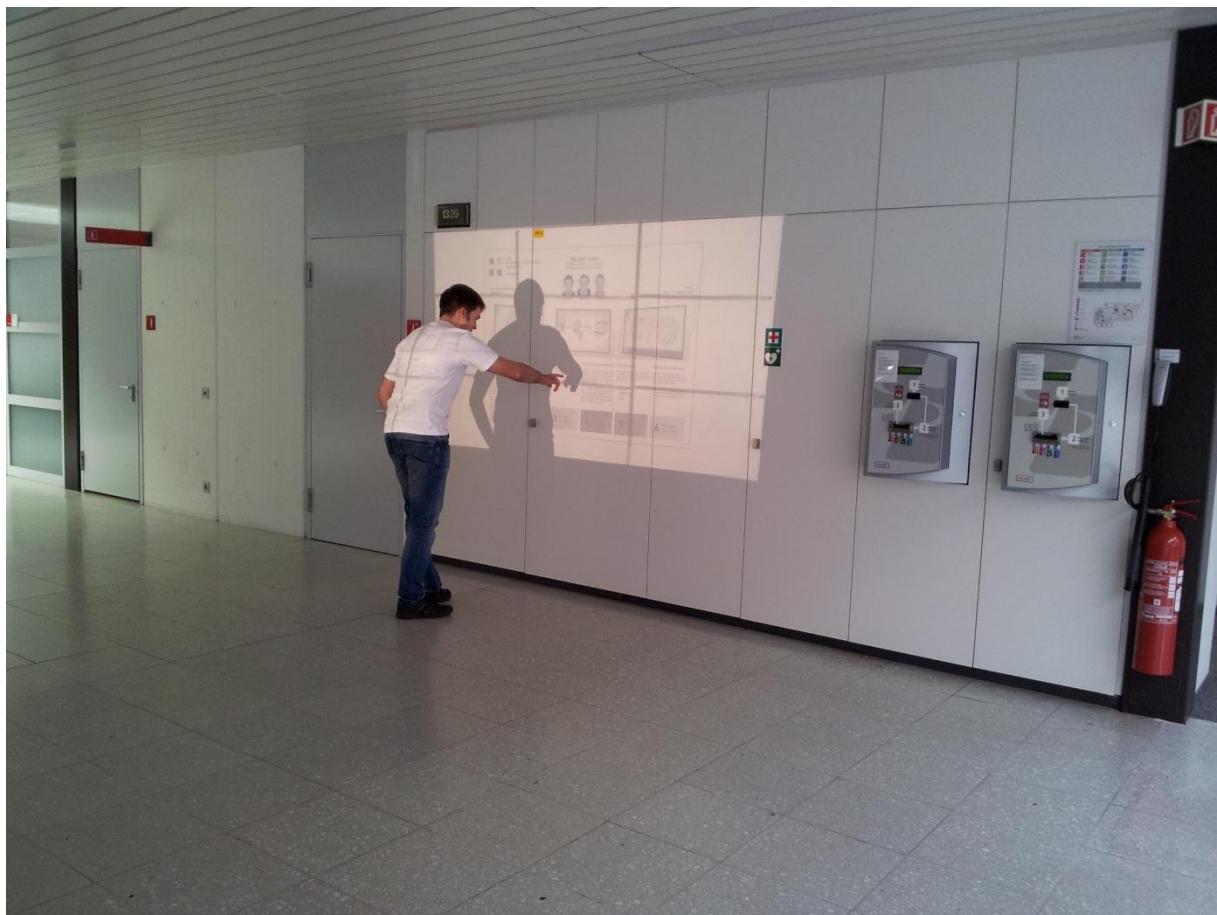


Abbildung 78 - Variante B: 2 x 2 55" Monitore, Hellraumprojektor Test

Aus Abbildung 78 - Variante B: 2 x 2 55" Monitore, Hellraumprojektor Test (auf dem Hellraumprojektor sind immer noch 3 x 3 Monitore sichtbar, da diese auf der Folie fest eingezeichnet wurden) ist jedoch schnell ersichtlich, dass diese Konstellation klein und verloren wirkt im Raum. Die Eindrücklichkeit, welche Variante A (siehe V.7.3.1.1 Variante A: 3 x 3 55" Monitore) vermittelt, entfällt hier.

#### V.7.3.1.3 Variante C: 1 x 6 55" Monitore

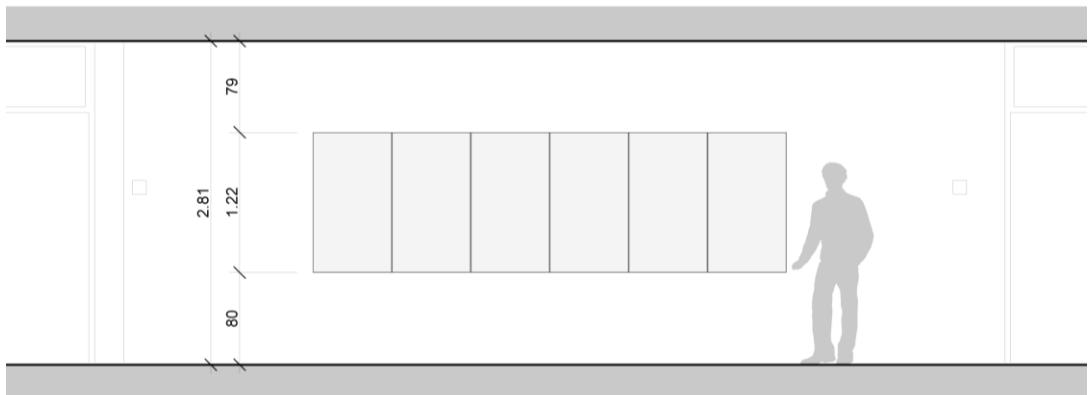


Abbildung 79 - Variante C: 1 x 6 55" Monitore, Ansicht

Diese Variante fügt sich von den Abmessungen her perfekt in den Gang des Verwaltungsgebäudes ein. Die Personen, die an der Videowall vorbei gehen, müssen bei dieser Variante eine längere Strecke bewältigen, bis sie das andere Ende der Wall erreichen. Daher ist die Zeit, in der sich die Passanten vor der Videowall bewegen, bei dieser Monitorkonstellation grösser. Die längere Zeitspanne bietet noch bessere Gelegenheit, die vorbeilaufende Person zu animieren, die Videowall zu benutzen. Denkbar ist bei dieser Lösung, dass die Möbel-Elemente des Infostandes (siehe Kapitel V.7.3.1 Monitoranzahl und -anordnung) ihren Platz behalten und die sechs Bildschirme darüber montiert werden. Das 1 x 6 Format ist jedoch für klassische Anwendungen wie Videos oder Spiele unvorteilhaft. Auf den Seiten der Bildschirmfläche würde zu viel Platz ungenutzt bleiben.



Abbildung 80 - Variante C: 1 x 6 55" Monitore, Hellraumprojektor Test

In der Abbildung 80 - Variante C: 1 x 6 55" Monitore, Hellraumprojektor Test ist ersichtlich, dass sich das auf der Videowall angezeigte Poster nur über die mittleren beiden Bildschirme erstreckt. Ein weiterer negativer Punkt ist, dass die Konstellation mit ihrer geringen Höhe im Raum verloren wirkt, obwohl dieser selbst auch

über keine grosse Höhe verfügt. Der Hauptnachteil ist jedoch, dass für diese Länge der Monitorkonstellation mehrere Kinects benötigt werden würden, um den gesamten Bereich mit Sensoren abdecken zu können. Dies würde die Entwicklung verkomplizieren.

#### V.7.3.1.4 Variante D: 2 x 4 55" Monitore

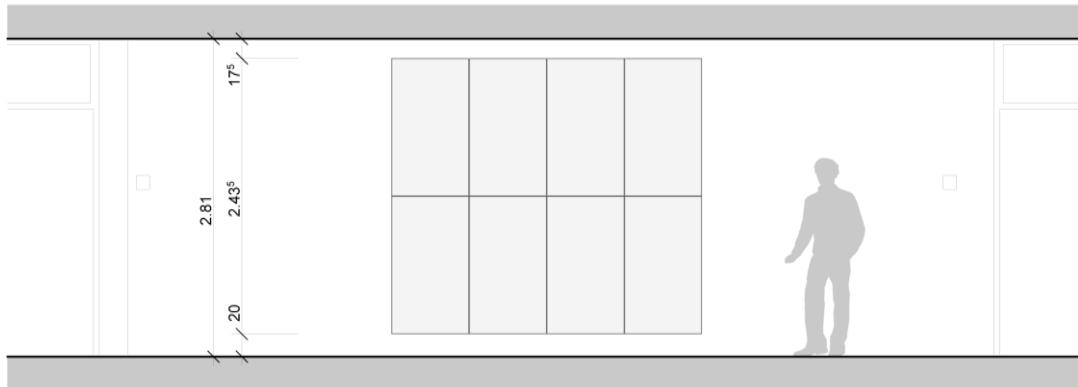


Abbildung 81 - Variante D: 2 x 4 55" Monitore, Ansicht

Im Unterkapitel V.7.3.3.3 Tests auf abgeänderter Testhardware mit einer Grafikkarte und acht Monitoren wurde getestet, ob die Performance-Probleme durch die zweite Grafikkarte (Matrox M9128 mit 2 Anschlüssen, siehe Kapitel V.7.3.2 Grafikkarten) hervorgerufen wurde. Aber würde sich eine geeignete Konstellation mit 8 Monitoren finden lassen? Um diese Frage zu beantworten, entschied sich das Team am 14.05.2012 dazu, eine 2 x 4 55" Monitor Variante zu prüfen.



Abbildung 82 - Variante D: 2 x 4 55" Monitore, Hellraumprojektor Test

Wie durch Abbildung 82 - Variante D: 2 x 4 55" Monitore, Hellraumprojektor Test ersichtlich ist, findet auch diese Monitorkonstellation gut im Raum Platz, obwohl der Abstand vom Boden und der Decke zur Videowall wesentlich knapper ist als bei V.7.3.1.1 Variante A: 3 x 3 55" Monitore. Bei der Variante mit den 2 x 4

Monitoren kann das Poster optimal platziert werden. Ein Vorteil ist, dass in der Applikation nun mehr Platz für die Anzeige des Skeletts am unteren Rand und des Menus am oberen Rand bleibt.

#### V.7.3.1.5 Fazit Monitorkonstellationen

Die Nutzwertanalyse bietet eine Auswertung der Kriterien, die in diesem Kapitel für die einzelnen Monitorkonstellationen diskutiert wurden. Die Nutzwertanalyse beinhaltet nur die Varianten 1 bis 3, da die Variante 4 erst gegen Ende des Projektes (Mai 2012) zur Sprache kam und zum Zeitpunkt der Analyse am 26. März 2012 noch nicht bekannt war.

Die Gewichtung der verschiedenen Kriterien lässt sich wie folgt begründen:

- Das Kriterium „Eignung für Raumhöhe (Raumgefühl)“ ist wichtig. Die Videowall soll in den bestehenden Raum passen, damit man sich gerne darin aufhält.
- Der Punkt „Bildschirmfläche überblickbar“ ist bedingt wichtig, da man als Nutzer nicht an einem Punkt stillstehen muss sondern sich vor der Wall bewegen kann.
- Die „Eignung des Formats (Seitenverhältnis)“ ist weniger wichtig. Die Benutzeroberfläche kann auf das Format angepasst werden.
- Das Kriterium „Darstellungsqualität/-grösse zum Lesen“ wird als wichtig angeschaut, da die Videowall mit ihren Grundapplikationen Poster und Mittagsmenu (siehe V.6.5.1 Physische Sicht) viel Text enthält, der angenehm zu lesen sein soll.
- Die „Kosten“ sollen der optimalen Videowall-Monitorkonstellation nicht im Wege stehen und wurden daher als wenig wichtig eingestuft.

Die Evaluation wurde manuell durchgeführt. Die Bewertung der einzelnen Kriterien mit wenig wichtig (1), bedingt wichtig (3) und sehr wichtig (5) ist selbsterklärend und wird daher nicht begründet.

Nutzwertanalyse: Auswahl Monitorkonstellation für Videowall								
Kriterium	Gewichtung	Variante 1		Variante 2		Variante 3		
		3 x 3 55" Monitore		2 x 2 55" Monitore		1 x 6 55" Monitore		
Bewertung	Total	Bewertung	Total	Bewertung	Total	Bewertung	Total	
Eignung für Raumhöhe (Raumgefühl)	5	3	15	1	5	5	25	
Bildschirmfläche überblickbar	3	5	15	5	15	3	9	
Eignung des Formats (Seitenverhältnis)	1	5	5	5	5	3	3	
Darstellungsqualität/-grösse zum Lesen	5	5	25	3	15	1	5	
Kosten	1	1	1	5	5	3	3	
<b>Total Punkte</b>		<b>61</b>		<b>49</b>		<b>45</b>		
<b>Rang</b>			<b>1</b>		<b>2</b>		<b>2</b>	

Bemerkung: Die Gewichtungs- / Bewertungsskala geht von wenig (1), bedingt (3) bis zu sehr wichtig (5).

Tabelle 18 - Nutzwertanalyse: Auswahl Monitorkonstellation für Videowall

Aus der Analyse (siehe Tabelle 18 - Nutzwertanalyse: Auswahl Monitorkonstellation für Videowall) geht hervor, dass sich die ursprünglich vorgeschlagene Variante (siehe Unterkapitel V.7.3.1.1 Variante A: 3 x 3 55" Monitore), so wie sie auch in der Aufgabenstellung (VIII Anhang) festgehalten ist, am besten für den vorgesehenen Raum eignet.

Die nachträglich dokumentierte Variante 4 mit den 8 Monitoren (siehe V.7.3.1.4 Variante D: 2 x 4 55" Monitore) ist der Variante 1: 3 x 3 55" Monitore sehr ähnlich. Sie benötigt in der Vertikale noch mehr Platz, ist aber etwas schmäler. Würde die Nutzung von lediglich einer Grafikkarte grosse Vorteile mit sich bringen (siehe hierzu V.7.3.3.3 Tests auf abgeänderter Testhardware mit einer Grafikkarte und acht Monitoren), so würde diese Variante derjenigen mit 3 x 3 Monitoren vorgezogen werden.

---

### V.7.3.2 Grafikkarten

Zu Beginn lag der HSR eine Offerte für eine Videowall vor, bei welcher die Bildschirme mithilfe eines Daisy Chain Boards zusammengeschlossen werden (für detaillierte Informationen siehe Anhang). Die Auflösung der Wall ist durch dieses Board aber auf ein Maximum von 1920 x 1200 beschränkt. Bei einer solch niedrigen Auflösung sind aber nicht alle Poster aller Abteilungen, namentlich die der Landschaftsarchitektur, lesbar. Daher wurde nach einer Möglichkeit gesucht, eine höhere Auflösung, idealerweise 3 x Full HD (5760 x 3240), zu erzielen. Poster mit kleinen Texten können bei einer solchen Auflösung gut gelesen werden. Es war aber abzuklären, ob eine 3 x Full HD-Auflösung überhaupt erreicht werden kann. Das Team beschloss daher, eine Grafikkartenlösung zu finden, mit welcher die neun Bildschirme der gewünschten Monitorkonstellation (siehe Unterkapitel V.7.3.1.5 Fazit Monitorkonstellationen) angesteuert werden können. Für die Lösung wurden Kartenhersteller oder Drittanbieter bezüglich einer Offerte angefragt. Die erhaltenen Offerten können im Anhang (VIII Anhang) eingesehen werden. Die Offerte der Firma Matrox konnte eine zufriedenstellende Lösung anbieten. Folgende zwei Karten wurden angeschafft:

1. Matrox M9188 mit 8 Anschläßen



Abbildung 83 - Matrox M9188

2. Matrox M9128 mit 2 Anschläßen



Abbildung 84 - Matrox M9128

---

### V.7.3.3 Testhardware

Am 15.03.2012 wurde die Testhardware aufgebaut. Diese Massnahme wurde ergriffen, um herauszufinden, ob die gewünschte Auflösung von 3 x Full HD möglich ist.

Für die Testhardware wurden die im Kapitel V.7.3.2 Grafikkarten beschriebenen Karten in einen Schulcomputer eingebaut. An diesen wurden neun Monitore (Fujitsu P22W-5 ECO IPS, 22 Zoll) mit je einer maximalen Auflösung von 1680 x 1050 angeschlossen. Die maximale Auflösung von 5040 x 3150 über alle neun Bildschirme entspricht nicht ganz dem vorgesehenen Setup von 3 x 3 Monitoren mit 3 x Full HD (5760 x 3240) Auflösung, ist aber für ein Testsetup ausreichend.



Abbildung 85 - Testhardware

#### V.7.3.3.1 Performance Tests mit WPF-Applikationen

##### V.7.3.3.1.1 Übersicht

Um zu testen, wie flüssig verschiedene WPF-Applikationen auf der Test-Wall laufen, wurde einerseits die Studienarbeit Project Flip 2.0<sup>41</sup>, welche das Team im Herbstsemester 2011 erarbeitet hatte (Applikation, mit welcher Projekte durchstöbert, gefiltert und gelesen werden können), und zum anderen die Testapplikation für den empirisch formativen Test (siehe V.5.4.1 Empirischer formativer Test zur Eruierung der Navigationsart) verwendet.

Für die Steuerung der neu eingebauten Hardware standen zwei Treiber zu Verfügung, einer basiert auf dem Windows Display Driver Model [microsoft06] (WDDM, neu seit Windows Vista) und der andere auf dem Windows 2000 Display Driver Model [microsoft12] (XDDM).

##### V.7.3.3.1.2 WDDM

Zu Beginn wurde der WDDM-Treiber verwendet. Mit diesem lief jedoch keine der Applikationen flüssig, schon nur das Maximieren einer Applikation über alle neun Bildschirme dauerte ein paar Sekunden. Applikationen mit einem aufwändigen GUI und Animationen, wie bei Project Flip 2.0<sup>41</sup>, verursachten grosse Performance-Probleme. Die Applikation lief sehr langsam und stockend, die verschiedenen Bildschirme liefen nicht immer synchron. Bei einer herabgesetzten Auflösung (1280 x 800 - 640 x 480 pro Bildschirm) liefen die Monitore wieder ohne Probleme synchron, aber auch mit diesen Einstellungen war die Applikation nicht flüssig und reagierte nur langsam.

Die Testapplikation lief ebenfalls langsam. Die Poster werden der Testapplikation für den empirisch formativen

<sup>41</sup> [elmer11] Lukas Elmer, Christina Heidt, Delia Treichler, „Project Flip 2.0“, <http://eprints3.hsr.ch/220/>  
letzter Zugriff: 13.04.2012

Test als XPS-Dokumente zur Verfügung gestellt. Der Wechsel vom einen zum nächsten Poster brauchte spürbar Zeit.

#### V.7.3.3.1.3 XDDM

Da die Tests mit dem WDDM-Treiber kein zufriedenstellendes Resultat lieferte, wurde der XDDM Treiber installiert um herauszufinden, ob mit der Verwendung dieses Treibers eine Verbesserung der Performance festgestellt werden kann. Zusätzlich wurden bei der Project Flip 2.0<sup>41</sup> Applikation alle Effekte (Schlagschatten- oder Unschärfeeffekte) des GUIs entfernt. Nach diesen Anpassungen konnte zumindest einigermassen flüssig durch die Projekteübersicht der Applikation gescrollt werden.

Trotz allem waren aufwändigere Animationen bei einer hohen Auflösung nicht flüssig. Die dargestellten XPS-Dokumente der Testapplikation brauchten bei einer hohen Auflösung immer noch einige Zeit, um angezeigt zu werden. Diese Zeit war auch immer noch spürbar, als die Auflösung weit heruntergesetzt wurde (640 x 480 pro Bildschirm).

#### V.7.3.3.1.4 Darstellungsoptionen Poster / PDF

Im Zuge der Evaluation der Darstellungsoptionen der Poster im PDF-Format (siehe V.6.2.2 PDF-Darstellung) wurde die Testapplikation für den empirisch formativen Test leicht abgeändert um weitere PDF-Darstellungsoptionen zu prüfen. Die Variante 1 (siehe V.6.2.2.1.1 Variante 1: PDF direkt darstellen) konnte nach kurzer Testphase und Auswertung ausgeschlossen werden und wurde daher nicht mit verschiedenen Auflösungen getestet. Die Variante 3 (siehe V.6.2.2.1.3 Variante 3: Umwandlung zu Bild) hingegen wurde auf der Wall ausführlicher untersucht. Das Anzeigen der Bilder benötigte bei den verschiedenen Auflösungen erwartungsgemäß immer etwa gleich lange. Mit dieser Variante könnte auf der Videowall die volle Auflösung (3 x Full HD) genutzt werden.

---

### V.7.3.3.2 Test mit Direct-Applikationen

Das Hardware-Setup mit den zwei Matrox Grafikkarten (siehe Unterkapitel V.7.3.2 Grafikkarten) ist bezüglich Performance nicht zufriedenstellend. So kann zum Beispiel beim Abspielen eines Videos oder einer einfachen Animation ein leichtes "Ruckeln" festgestellt werden. Da in der Videowall-Applikation Animationen und später auch Videos eingesetzt werden sollen, stellt dieser Fakt ein erhebliches Problem dar. Die Videowall-Applikation soll eine neue Technologie demonstrieren und sofort einen positiven Eindruck beim Benutzer hinterlassen. Dies kann jedoch nicht gewährleistet werden, wenn die Applikation bei Animationen ruckelt.

Die Performance-Problematik bezüglich Grafikkarten allgemein wurde auch am Meeting vom 12.04.2012 mit Markus Flückiger von der Zühlke Engineering AG besprochen. Seine erste Vermutung war, dass das Problem mit WPF zusammenhängt, da diese Technologie sich nicht für grafisch aufwändige Applikationen eignet. Nach Absprache mit Spezialisten der Zühlke Engineering AG schickte Markus Flückiger am 16.04.2012 eine E-Mail (VIII Anhang) mit dem Vorschlag, den GUI-Layer der Applikation mit DirectX zu entwickeln, um eine bessere Performance erzielen zu können.

Da eine Ersetzung von WPF durch DirectX einen grossen Aufwand mit sich bringen würde, wurde zuerst abgeklärt, ob mit DirectX entwickelte Applikationen denn tatsächlich schneller und vor allem flüssiger laufen würden. Es wurden Tests mit verschiedenen Programmen (Hardware Acceleration Stress Test<sup>42</sup>), speziell mit 3D Computer Games (Sanctuary Demo 2.3<sup>43</sup>, Unreal Tournament 2004<sup>44</sup>), durchgeführt. Bei diesen Tests wurde sehr schnell festgestellt, dass die Matrox Grafikkarten nicht den kompletten Befehlssatz von DirectX implementieren (entsprechende Fehlermeldungen wurden bei den Tests angezeigt). Nachdem die Prüfung mit den Videospiele fehlgeschlagen, wurden weitere Tests mit zwei Video Playern (VLC<sup>45</sup> und Windows Media Player) und einer selbst programmierten WPF-Applikation durchgeführt. Für alle Prüfungen wurde das Windows 7-Beispielvideo (C:\Users\Public\Videos\Sample Videos\Wildlife.wmv) verwendet. Dabei wurde folgendes festgestellt:

- VLC<sup>45</sup> (Version 2.0.1): Die Videos laufen flüssig mit den Einstellungen DirectX oder Direct2D, eventuell werden einzelne Frames übersprungen. Mit allen anderen Einstellungen (z.B. OpenGL, für komplett

<sup>42</sup> <http://hacks.mozilla.org/2010/09/hardware-acceleration/>

<sup>43</sup> <http://unigine.com/products/sanctuary/>

<sup>44</sup> <http://epicgames.com/technology/>

<sup>45</sup> <http://www.videolan.org/vlc/>

Liste siehe Abbildung 86 - Videoeinstellungen VLC Media Player) funktioniert die Ausgabe nicht.

Speziell hervorzuheben ist, dass Direct3D nicht unterstützt wird.

- Windows Media Player: Die Videos laufen verlangsamt (ca. 0.6 Mal so schnell wie normal) bis etwa 20 Sekunden des Videos abgespielt sind, danach wird an das Ende des Videos gesprungen.
- WPF-Applikation: Die Videos ruckeln spürbar, können aber trotzdem angesehen werden und es treten keine Fehler auf.

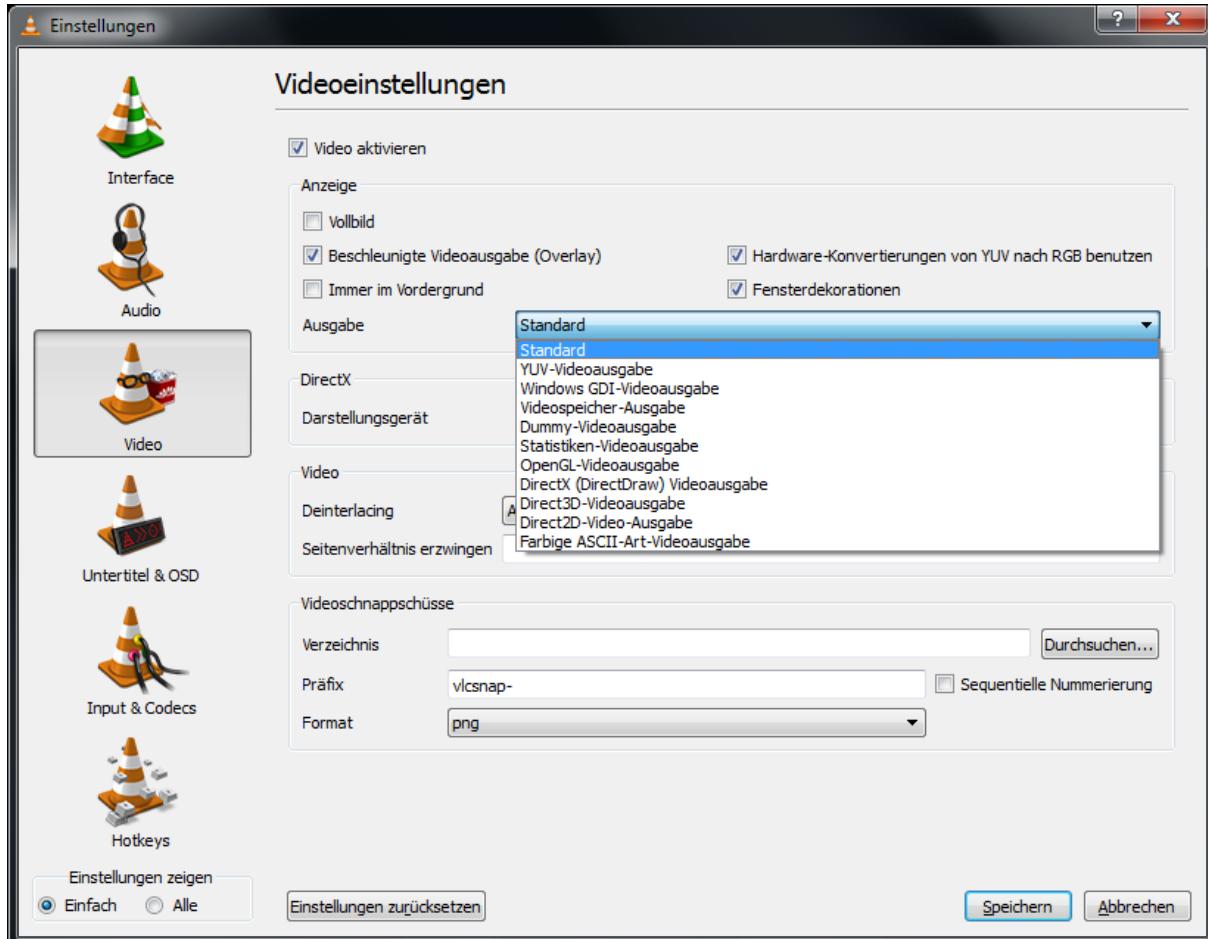


Abbildung 86 - Videoeinstellungen VLCMedia Player

Es wurde die Tendenz, dass die Videos sowie die Spiele bei tieferer Auflösung flüssiger laufen als bei höherer Auflösung, festgestellt.

#### V.7.3.3.2.1 Fazit

Für diese Abklärungen über DirectX konnte nur ein Mitglied des Bachelorteams eingesetzt werden, Experten für ein Setup mit neun Monitoren und zwei Matrox Grafikkarten (beispielsweise Personen von der Firma Matrox) fehlten. Soll tatsächlich mit DirectX gearbeitet werden, so muss das konkrete Setup sowie die Performance mit DirectX nochmals genauer abgeklärt werden.

Die Videowall-Applikation wird daher, wie ursprünglich geplant, weiter mit WPF entwickelt.

### V.7.3.3.3 Tests auf abgeänderter Testhardware mit einer Grafikkarte und acht Monitoren

Am 18.05.2012 testete das Team, ob die Performance von WPF- oder DirectX-Applikationen gesteigert werden kann, wenn die Videowall aus nur acht Monitoren bestehen würde. Dazu wurde die Grafikkarte mit den zwei Anschlüssen (Matrox M9128, siehe Kapitel V.7.3.2 Grafikkarten) aus dem Schulcomputer ausgebaut und nur die andere Grafikkarte mit den acht Anschlüssen verwendet. Die Eignung der 2 x 4 Monitorkonstellation dieses Setups ist im Unterkapitel V.7.3.1.4 Variante D: 2 x 4 55" Monitore beschrieben.

Zur Durchführung der Tests wurde zuerst der WDDM Treiber installiert, danach der XDDM Treiber. Beide erlauben das Zusammenführen der zwei auf der Karte befindlichen GPUs zu einer logischen Grafikkarte. Die Erwartung, dass Applikationen flüssiger laufen, da in diesem Setup keine Kommunikation mit einer weiteren Grafikkarte besteht, wurde nicht erfüllt. Das Team stellte zwar fest, dass die DirectX-Applikation ein bisschen flüssiger (ca. 12 FPS) liefen als bei den Tests im Unterkapitel V.7.3.3.2 Test mit Direct-Applikationen. Der Unterschied ist aber minim.

Die Nutzung von nur einer Grafikkarte bringt dem Test zufolge keine Vorteile.

### V.7.3.3.4 Tests mit verkleinertem Video

Nachdem alle bisher durchgeführten Tests zu keiner zufriedenstellenden Lösung geführt hatten, wurde am 24.05.2012 getestet, ob bei voller Auflösung (3 x Full HD) ein Video in einer WPF-Applikation dargestellt werden kann.

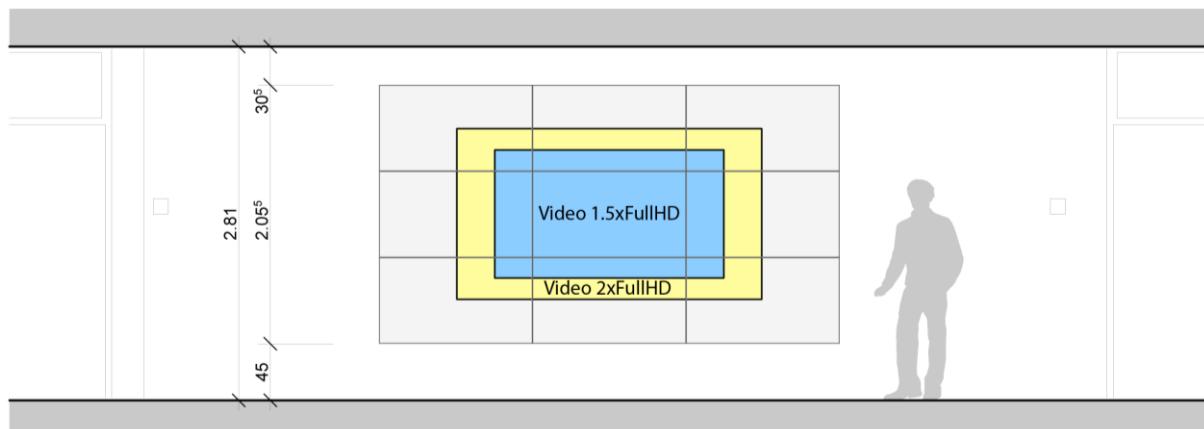


Abbildung 87 - Variante C 3 x 3 55" Monitore mit einem 1.5-fach (blau) und 2-fach (gelb) vergrößerten Video

Ein einzelner Monitor besitzt eine Auflösung von Full HD (1920 x 1080) (siehe Abbildung 87 - Variante C 3 x 3 55" Monitore mit einem 1.5-fach (blau) und 2-fach (gelb) vergrößerten Video). Die blaue resp. gelbe Fläche zeigt die Grösse, die ein WPF-Video mit 1.5-facher resp. 2-facher Vergrösserung hätte.

Um die optimale Konfiguration für ein Video in Full HD oder mit 1.5-facher resp. 2-facher Vergrösserung zu finden, mussten verschiedene Treibermodelle und Monitormodi der Grafikkarten getestet werden. Optimal heisst, dass beim Abspielen des Videos das Bild nicht ruckelt. Nachfolgend eine Zusammenfassung der Resultate:

#	Anzahl Monitore	Treibermodell	Monitormodus	Videogrösse (x*Full HD)	Gut	Knapp	Schlecht
1	8	WDDM	Independent	1.5		x	
2			Partial stretched	1.5		x	
3			Joined & stretched	1.5	x		
4			Joined & stretched	2		x	
5		XDDM	Independent	1		x	
6			Stretched	1	x		
7			Stretched	1.5		x	
8	9	WDDM	Independent	1			x

<b>9</b>	Partial stretched	1	x
<b>10</b>	Joined & partial stretched	1	x
<b>11</b>	XDDM	Independent	1
<b>12</b>		Stretched	1.5 x
<b>13</b>		Stretched	2 x

**Tabelle 19 - Video Performance Test Resultate**

Empfehlungen aus den Testresultaten:

- Der Test #12 (siehe Tabelle 19 - Video Performance Test Resultate) liefert die beste Performance: 1.5-fache Full HD-Videogrösse, 9 Bildschirme, Treibermodell: XDDM, Modus: stretched.
- Sollte die Videowall mit dem WDDM Treiber betrieben werden, so wird empfohlen nur 8 Monitore anzuschliessen. Für die Testergebnisse für diese Konfiguration siehe Test #3 in obenstehender Tabelle.
- Soll das Video mit 2-facher Vergrösserung abgespielt werden, ist die Konfiguration von Test #13 anzuwenden.

Nachfolgend werden die Kategorien der obenstehenden Tabelle (siehe Tabelle 19 - Video Performance Test Resultate) erläutert.

#### Allgemein

Alle getesteten Kriterien beeinflussen die Performance. Leider ist die Optimierung der Performance über diese Kriterien kein lineares Problem, weshalb alle Kombinationen ausprobiert werden mussten.

#### Anzahl Monitore

Da eine 2 x 4 Konfiguration der Monitore auch möglich ist und diese 8 Bildschirme mit einer einzigen Grafikkarte (siehe Kapitel V.7.3.2 Grafikkarten) betrieben werden können, wurde diese Konfiguration zusätzlich zur 3 x 3 Monitorkonstellation getestet.

#### Treibermodell

Weitere Details zu den Treibermodellen sind in den Unterkapiteln V.7.3.3.1.2 WDDM und V.7.3.3.1.3 XDDM zu finden.

#### Monitormodus

Es gibt verschiedene Modi um die Monitore zu betreiben. Nicht alle Modi existieren bei beiden Treibern, folgende Varianten sind verfügbar:

##### Treiber WDDM

- Independent
- Partial stretched
- Joined & stretched (für 8 Monitore)
- Joined & partial stretched (für 9 Monitore)

##### Treiber XDDM

- Independent
- Stretched

Zur besseren Veranschaulichung sind die verwendeten Konfigurationen nachfolgend aufgeführt:

### *Independent*

Alle Monitore sind unabhängig voneinander und werden von Windows als einzelne Monitore erkannt.

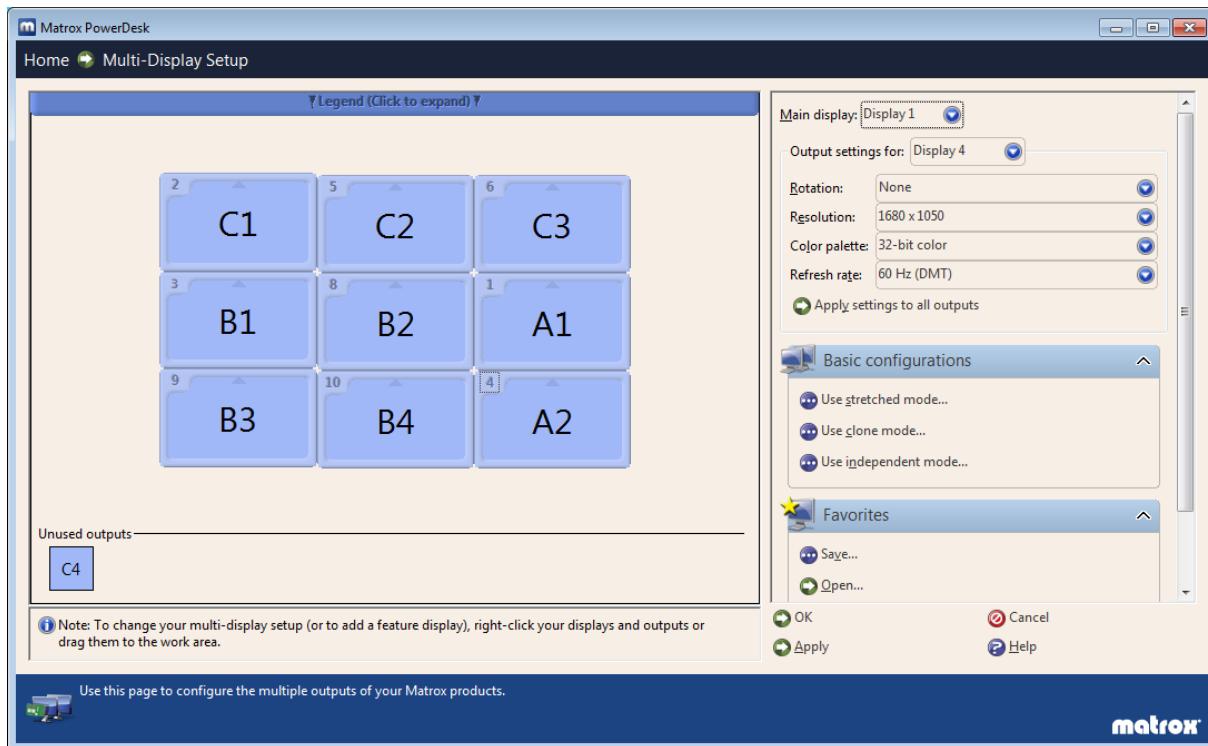


Abbildung 88 - Konfiguration "Independent" (XDDM, WDDM)

### *Stretched*

Verschiedene einzelne Bildschirme werden zu einem grossen virtuellen Bildschirm zusammengeschlossen.

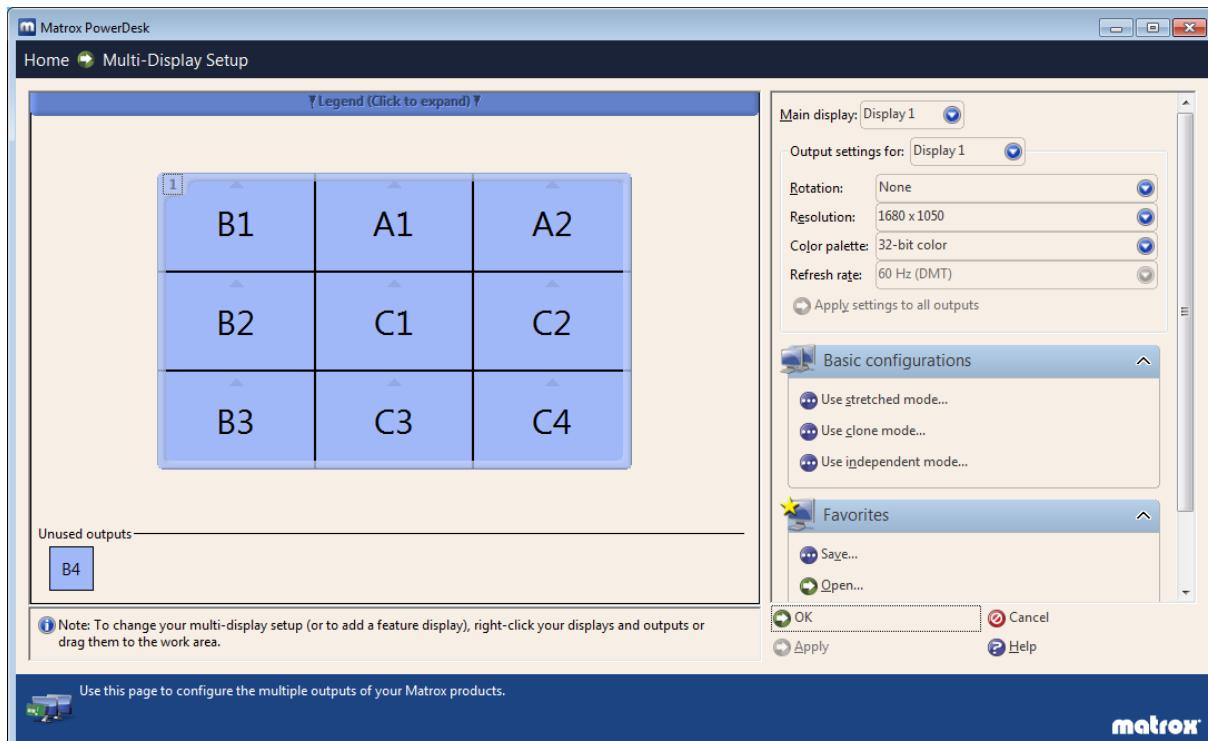


Abbildung 89 - Konfiguration "Stretched" (XDDM)

### *Partial stretched*

Da die beiden Grafikkarten M9128 und M9188 (siehe Unterkapitel V.7.3.2 Grafikkarten) unter Verwendung des WDDM-Treibermodells nicht zusammengeschlossen („Joined“) werden können, beschreibt dieser Modus, dass nur die Bildschirme, welche an der gleichen Grafikkarte angeschlossen sind, zu einem grossen virtuellen Bildschirm zusammengeschlossen werden.

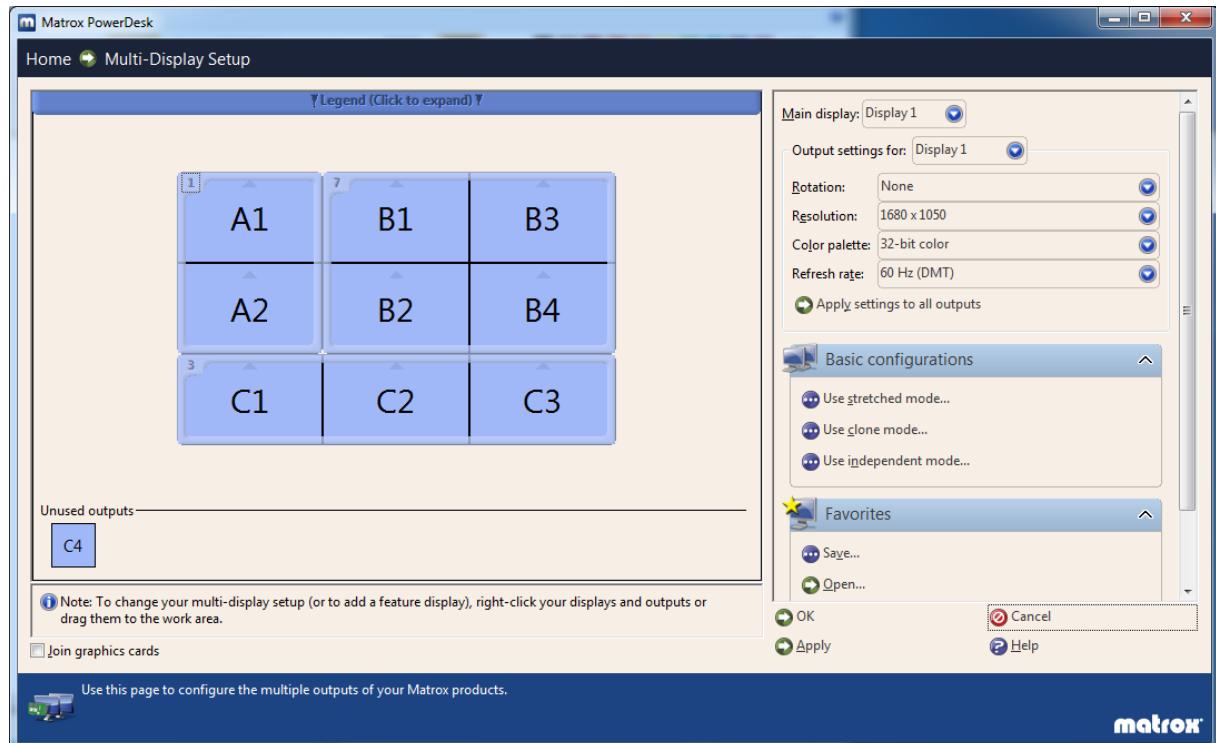


Abbildung 90 - Konfiguration "Partial stretched" (WDDM)

### *Joined & stretched*

Um einen grossen virtuellen Bildschirm (stretched) mit dem Treibermodell WDDM zu erzeugen, ist es nötig, die GPUs der Grafikkarten zusammenzuschliessen. Dies kann über die Option „Joined“ angegeben werden.

Werden nur 8 Monitore verwendet, so wird nur die Grafikkarte mit den 8 Anschlüssen (siehe Matrox M9188 mit 8 Anschlüssen im (Unterkapitel V.7.3.2 Grafikkarten) benötigt. Daher ist nun ein Zusammenschliessen („Joined“) der zwei auf der gleichen Grafikkarte vorhandenen GPUs möglich.

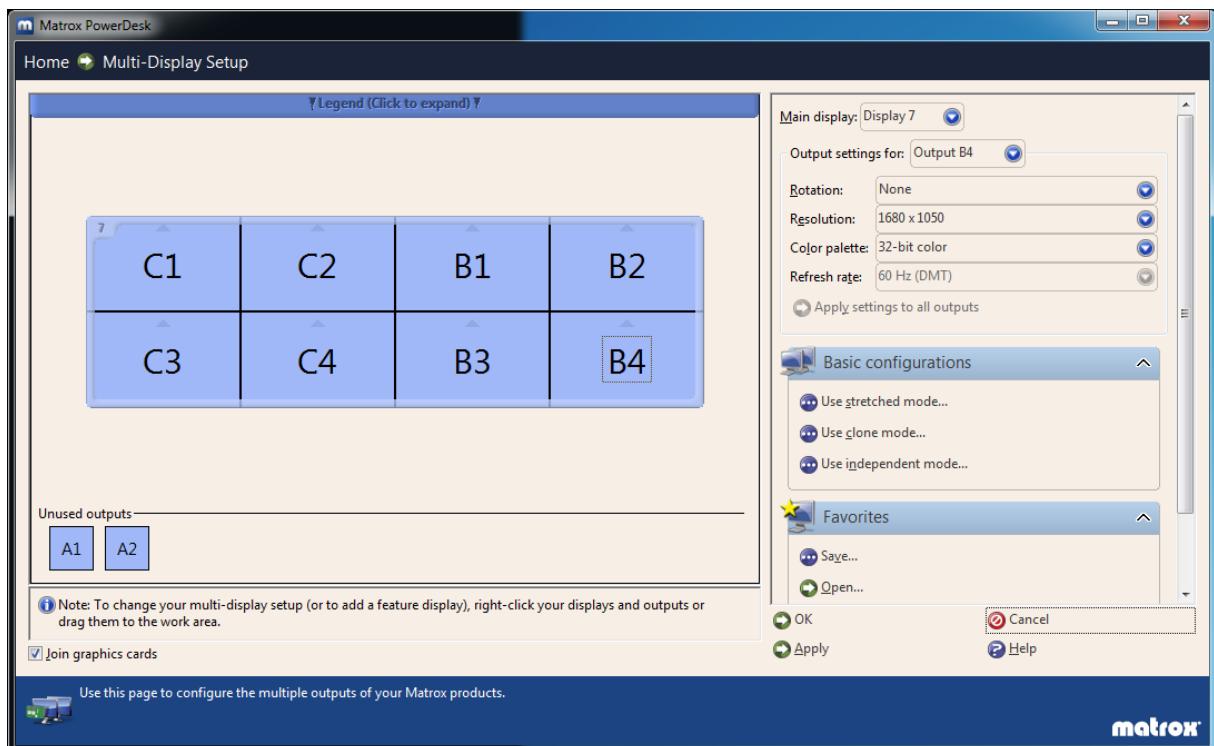


Abbildung 91 - Konfiguration "Joined & stretched" (WDDM)

### *Partial Joined & partial stretched*

Da die GPU der zweiten Grafikkarte mit 2 Anschlüssen nicht mit den zwei GPUs der ersten Grafikkarte mit 8 Anschlüssen zusammengeschlossen („Joined“) werden können, wurde versucht, nur die zwei GPUs der grossen Grafikkarte zusammenzuschliessen (B1, B2, B3, B4, C2, C3) und die zweite Grafikkarte einzeln laufen zu lassen (Monitore A1 und A2). Leider kann der „Joined“-Modus nur dann verwendet werden, wenn der zusammengeschlossene, rosse Screen rechteckig ist. Deshalb konnte der Monitor C1 nicht mit den anderen GPUs der ersten Grafikkarte mit 8 Anschlüssen zusammengeschlossen werden.

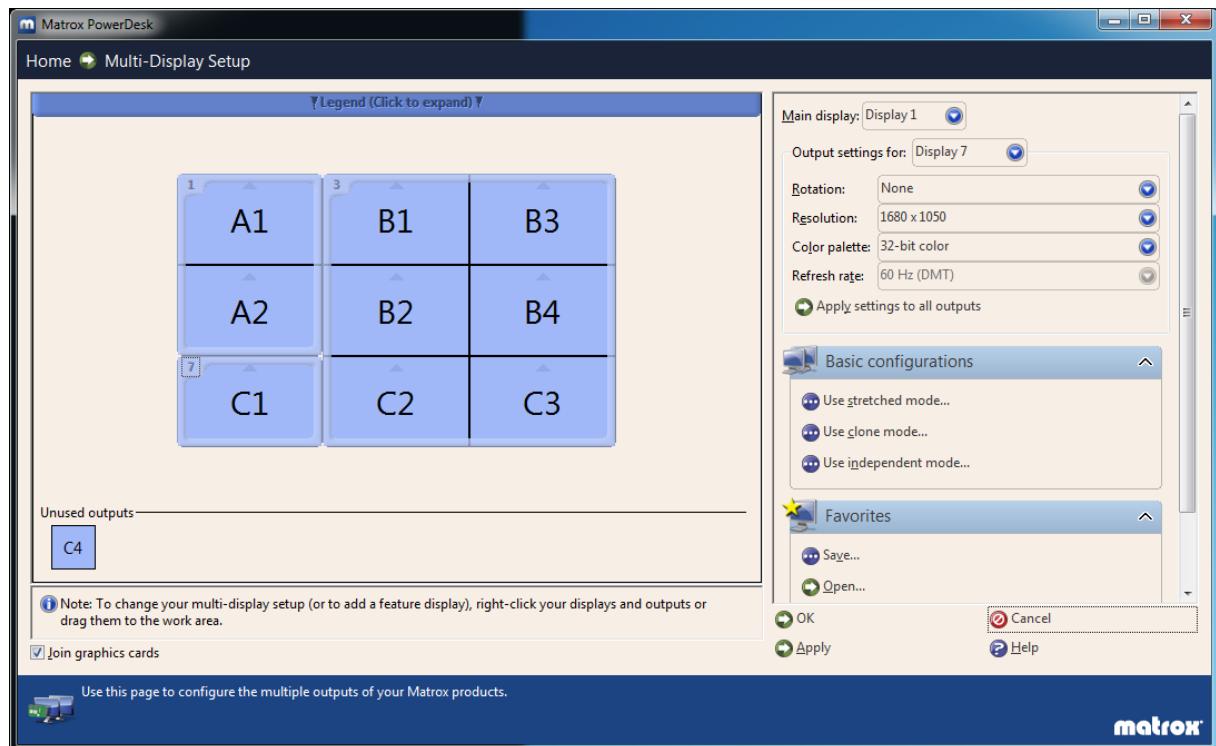


Abbildung 92 - Konfiguration "Joined & partial stretched"

### **Videogrösse (x \* Full HD)**

Die Videogrösse beschreibt die Grösse (nicht Auflösung) des Videos, das in einer maximierten WPF-Applikation (über alle Bildschirme gestreckt) abgespielt wird. Es wurden verschiedene Videogrössen getestet um festzustellen, ab welcher Auflösung das Video nicht mehr ohne Ruckeln abgespielt werden kann.

### **Bewertungen (Gut, Knapp, Schlecht)**

Die Bewertungen beziehen sich auf die Flüssigkeit / Performance des Videos, das in einer WPF-Applikation abgespielt wird.

- Gut bedeutet, dass das Video angenehm anzuschauen ist, d.h. von Auge kein Ruckeln festgestellt werden kann.
- Knapp heisst, dass ein Ruckeln zwar klar erkenntlich ist, das Video aber noch immer angesehen werden kann. Im Notfall könnten Videos mit dieser Bewertung für die Videowall benutzt werden.
- Schlecht bedeutet, dass das Video gar nicht oder mit so viel Verzögerung abgespielt wird, dass es nicht mehr als Video angesehen werden kann (weniger als 2 Bilder pro Sekunde).

---

#### *V.7.3.3.5 Fazit der durchgeführten Tests mit unterschiedlicher Hardwarekonstellation*

Die verschiedenen Performance-Tests, welche bis zum 24.05.2012 durchgeführt wurden, führten zu keiner zufriedenstellenden Lösung. Auf der Suche nach weiteren Lösungen wurde beschlossen, dass ein Video auch verkleinert, also nicht über den ganzen Bildschirm gestreckt, abgespielt werden könnte. Dieser Test (siehe Unterkapitel V.7.3.3.4 Tests mit verkleinertem Video) ergab, dass ein Video mit 1.5-facher Full HD-Grösse und mit Full HD-Auflösung gut abgespielt werden kann, das Video ist angenehm anzusehen.

Trotzdem wird eine maximale Auflösung von 1 x Full HD empfohlen, da hiermit sichergestellt wird, dass die Applikation auch mit vielen Animationen flüssig läuft. Die Poster der verschiedenen Abteilungen der HSR können bis zu einer minimalen Auflösung von 1280 x 800 knapp gelesen werden. Dies gilt jedoch nicht für die Poster der Landschaftsarchitektur. Wie jedoch im Kapitel V.7.6 Lesbarkeit der Poster ersichtlich ist, können diese auch mit einer Auflösung von 3 x Full HD nicht gelesen werden.

Es ist jedoch auch möglich eine Auflösung von 3 x Full HD zu verwenden, hierbei darf die Applikation aber nur über wenige Animationen verfügen. Um mit dieser Auflösung Videos abspielen zu können, wird die Konfiguration von Test #12 (siehe Tabelle 19 - Video Performance Test Resultate) empfohlen. Dieses Setup hat den Vorteil, dass das Bild auf diese Weise sehr scharf ist und die Poster sehr angenehm gelesen werden können. Die Lesbarkeit wird wichtiger bewertet als gute Performance bei Programmen mit vielen Animationen oder Spielen.

---

### **V.7.4 Evaluation Mitsubishi Display Wall**

Zu Beginn des Projektes holte die HSR eine Offerte (VIII Anhang) für die geplante Videowall ein. Die darin aufgelistete Hardware verfügt über ein Daisy Chain Board, welches die Verteilung eines Signals auf neun Monitore über einen Anschluss ermöglicht. Diese Lösung bietet eine maximale Auflösung von 1920 x 1200 (1 x Full HD).

Markus Stolze und das Bachelorteam konnten am 2. Mai 2012 eine Mitsubishi Display Wall bei der Firma CPP AG in Geroldswil besichtigen.

Diese Mitsubishi Display Wall verfügte über 2 x 2 49" LCD Displays. Die Monitore hatten mit insgesamt 5.7 mm eine extrem schmale Rahmenbreite. Das Team testete, ob, und wenn ja wie stark, die Rahmen das Erscheinungsbild eines Posters beeinträchtigen. Der schmale Rahmen wurde aber schon nach kurzer Zeit kaum mehr wahrgenommen. Die Wall wirkte zudem vor allem bei der Visualisierung von Bildern extrem eindrücklich. Bei einer Konstellation aus mehreren Monitoren ergibt sich das Problem, dass die Bildfläche in den Ecken der einzelnen Bildschirme dunkler erscheint. Diese Problematik wurde bei der Mitsubishi Display Wall mit speziellen Monitoren mit LED Backlights und einer digitalen Gradationskontrolle gelöst. Diese Lösung bietet eine gleichmässige Beleuchtung. Auch verfügt die Wall über eine Farbraum- und eine dynamische Helligkeits-Anpassung. Mehr Informationen können dem Datenblatt zur Mitsubishi Display Wall im Anhang entnommen werden (VIII Anhang).

Das Team wollte mit dem Anschauen und Testen einer Mitsubishi Display Wall sicherstellen, dass Poster auf der Wall gelesen werden können. Hierfür verwendeten sie verschiedene Poster der Abteilungen Informatik, Elektrotechnik und Landschaftsarchitektur. Die Poster der Informatik und Elektrotechnik konnten ohne Probleme gelesen werden. Bei jenen der Landschaftsarchitektur konnten hingegen nur die grösseren Übertitel gelesen und die Bilder betrachtet werden. Ist bei der Videowall für die HSR keine höhere Auflösung als 1920 x 1200 (1 x Full HD) möglich, muss eine andere Möglichkeit gesucht werden, um die sehr detaillierten Poster trotzdem lesbar machen zu können. Es besteht die Option, die Poster bis zu einem bestimmten Grad zu vergrössern und von einem bestimmten Bereich aus eine moderierte Navigation innerhalb des Posters anzubieten.

---

## V.7.5 Beschaffungsanalyse

Um eine Empfehlung für das endgültige System machen zu können, wurden mehrere Offerten für Monitore eingeholt. Diese können im Anhang eingesehen werden (VIII Anhang).

Für die Videowall sollen die im Kapitel V.7.3.2 Grafikkarten beschriebenen Grafikkarten favorisiert werden, da diese im Zuge der Hardware-Evaluation bereits gekauft wurden.

---

### V.7.5.1 Videowall mit 3 x 3 55“ Monitoren

Eine Konstellation mit 3 x 3 55“ Monitoren stellt die Wunschkonstellation für die HSR Videowall dar. Diese kann von verschiedenen Anbietern geliefert werden. Konkret wurden zwei Offerten eingeholt, eine zur Mitsubishi Display Wall ohne Daisy Chain Board (siehe V.7.4 Evaluation Mitsubishi Display Wall) und die andere zu den Hyundai Indoor Public Displays. Im Vergleich zur Mitsubishi Display Wall konnten die Hyundai Indoor Public Displays nicht vor Ort besichtigt werden. Bevor der Entscheid auf diese Offerte fällt, müssten daher genauere Untersuchungen gemacht werden. Die Offerten sind im Anhang zu finden (VIII Anhang).

- Vorteile
  - Hohe Auflösung von 1920 x 1080 (1 x Full HD) pro Monitor
  - Sehr schmaler Gehäuserahmen
- Nachteile
  - Sehr kostspielig

---

#### V.7.5.1.1 Verwendung von Daisy Chain Board

Die Mitsubishi Display Wall (siehe V.7.4 Evaluation Mitsubishi Display Wall) kann auch mit einem Daisy Chain Board geliefert werden. Dieses verteilt das Bildsignal über einen Anschluss auf alle neun Monitore.

- Vorteile
  - Die Grafikkarte muss nur über einen Anschluss verfügen
  - Sehr schmaler Gehäuserahmen
- Nachteile
  - Die Auflösung ist auf 1920 x 1200 (1 x Full HD) über alle Monitore begrenzt
  - Sehr kostspielig

---

### V.7.5.2 Videowall mit 3 x 3 46“ Monitoren

Die Konstellation mit 3 x 3 46“ Monitoren entspricht nicht der Wunschkonstellation. Jedoch wurde bei Recherchen festgestellt, dass Monitorwände mit diesen etwas kleineren Monitoren zu wesentlich günstigeren Preisen angeboten werden. Daher wurde auch für diese Konstellation eine Offerte eingeholt (VIII Anhang).

- Vorteile
  - Der Kostenfaktor der Monitore wird um ca. 2/3 reduziert
  - Sehr schmaler Gehäuserahmen
- Nachteile
  - Die Auflösung ist auf 1366 x 768 pro Monitor begrenzt, ergibt über alle Monitore eine Auflösung von ungefähr 2 x Full HD
  - Entspricht nicht der Wunschgrösse

---

## V.7.6 Lesbarkeit der Poster

Dem Team standen 21 Testposter zur Verfügung. Davon waren zehn von der Informatik-, vier von der Elektrotechnik- und sieben von der Landschaftsarchitektur-Abteilung. Bei einer Auflösung von 1 x Full HD sind die Texte der Poster der Abteilungen Informatik und Elektrotechnik problemlos lesbar, die der Landschaftsarchitektur jedoch nicht. Daher wurde geprüft, ob eine Auflösung von 3 x Full HD die Lesbarkeit der Landschaftsarchitektur-Poster verbessern könnte.

Für den Test wurden zwei Bilder vorbereitet. Das eine soll eine Auflösung von 3 x Full HD simulieren. Dazu wurde ein Neuntel eines Landschaftsarchitektur-Posters auf einem weissen Hintergrund mit dem Format 16:9 platziert. Der Ausschnitt des Posters wurde so angepasst und positioniert, dass er 11/12 der Vertikale einnahm. Für das zweite Bild, um eine Auflösung von 1 x Full HD zu simulieren, wurde wiederum ein weisser Hintergrund mit dem Format 16:9 verwendet. Dieses Mal wurde das ganze Landschaftsarchitektur-Poster so angepasst und positioniert, dass es 3/4 der Höhe des Hintergrundes einnahm. Bei beiden Bildern dürfen die Poster nicht den ganzen Platz einnehmen, da der freie Bereich in der Videowall-Applikation für die Darstellung des Menus und das Anzeigen des Skeletts verwendet wird.

Am 25.05.2012 prüfte das Team auf dem HP LD4200tm des Instituts für Software (IFS), ob beide erstellten Abbildungen lesbar sind. Die Abbildungen wurden auf dem 42“ Monitor aus einer Entfernung von drei bis vier Metern betrachtet. Es wurde festgestellt, dass der Text auf dem 1/9-Ausschnitt des Posters (Simulierung einer 3 x Full HD-Auflösung) zwar lesbar ist, das Lesen aber anstrengend für die Augen ist. Es ist davon auszugehen, dass Benutzer der Videowall solche Poster nur bei grossem Interesse lesen werden. Die Poster der Landschaftsarchitektur werden bei einer 3 x Full HD-Auflösung daher als bedingt lesbar eingestuft. In der 1 x Full HD-Variante, in welcher das gesamte Poster auf dem Bild sichtbar ist, können lediglich der Titel und die Hauptüberschriften des Posters gelesen werden. Die Poster der Landschaftsarchitektur werden bei einer Auflösung von 1 x Full HD als nicht lesbar eingestuft.

Um die Problematik mit den schlecht lesbaren Landschaftsarchitektur-Postern zu lösen, wurde im Backlog eine User Story (siehe V.4.3 Funktionale Anforderungen) erstellt. Eine denkbare Lösung wäre eine Zoom-Möglichkeit in der Applikation oder ein vordefinierter Pfad, über den die verschiedenen Ausschnitte des Posters dem Benutzer präsentiert werden.

---

### V.7.6.1 Prozentuale Lesbarkeit

Da die Bachelorposter der Landschaftsarchitektur auch bei einer hohen Auflösung von 3 x Full HD nur mit Mühe lesbar sind, wurde eruiert, wie gross der Anteil von Postern, die nicht oder nur erschwert lesbar sind, auf der Videowall sein wird.

Für die Erstellung der Auswertung wurden die folgenden zwei Annahmen getroffen:

- Die Bachelorarbeiten der Studiengänge Informatik und Elektrotechnik werden in Zweiergruppen durchgeführt.
- Die Arbeiten der übrigen Studiengänge entstehen in Einzelarbeit.

Zum Zeitpunkt der Auswertung am 25.05.2012 stehen noch keine Bachelorarbeiten der Abteilung Erneuerbare Energien und Umwelttechnik zur Verfügung. Für die Auswertung wurden auf der Unterrichtswebsite der HSR die Studenten gezählt, welche sich zwischen dem Frühlingssemester 2008 und dem Frühlingssemester 2012 für die Bachelorarbeit in ihrem Studiengang angemeldet hatten. Die detaillierte Auswertung kann dem Anhang entnommen werden (VIII Anhang). Über die letzten viereinhalb Jahre ergibt sich nachfolgende Verteilung.

## Anzahl Arbeiten pro Abteilung in Prozent

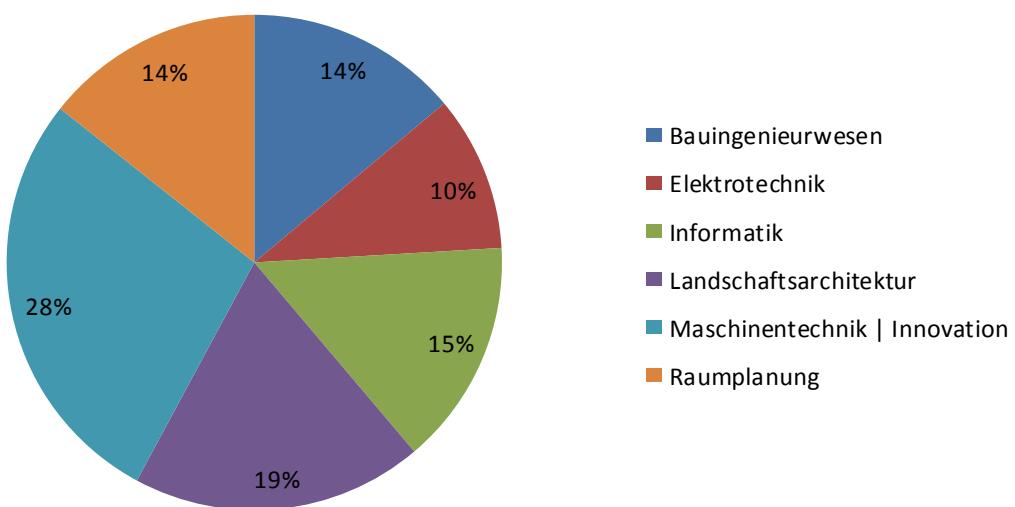


Abbildung 93 - Anzahl der Arbeiten pro Abteilung, Angaben in Prozent

Wie in Abbildung 93 - Anzahl der Arbeiten pro Abteilung, Angaben in Prozent ersichtlich ist, machen die Arbeiten der Landschaftsarchitektur 19% aller Bachelorarbeiten aus. Es wird daher angenommen, dass die restlichen Poster, welche 81% ausmachen, mit beiden Auflösungen von 1 x Full HD und 3 x Full HD lesbar sind.

Aus den Tests mit dem verkleinert dargestellten Video (siehe Unterkapitel V.7.3.3.4 Tests mit verkleinertem Video) ergibt sich, dass Videos mit 1.5- oder 2-facher Full HD-Auflösung performant laufen. Als Folge sind ca. 81% der Bachelorposter gut auf der Videowall lesbar.

## V.8 Realisierung & Test

<b>V.8.1 Änderungsgeschichte .....</b>	<b>138</b>
<b>V.8.2 Usability Tests .....</b>	<b>139</b>
V.8.2.1 Test 1: Wizard of Oz.....	139
V.8.2.1.1 Testszenario.....	139
V.8.2.1.1.1 Aufgabe .....	139
V.8.2.1.1.2 Resultat & Fazit.....	140
V.8.2.2 Test 2: Reaktion der Nutzer .....	140
V.8.2.3 Test 3: Reaktion auf Demomodus .....	141
V.8.2.4 Test 4: Grafisches Design.....	142
<b>V.8.3 Unit Tests .....</b>	<b>143</b>
V.8.3.1 Videowall-Applikation .....	143
V.8.3.2 Poster-Applikation.....	143
V.8.3.3 Mittagsmenu-Applikation.....	144
<b>V.8.4 Systemtests .....</b>	<b>145</b>
V.8.4.1 Sprint 7.....	145
V.8.4.2 Sprint 8.....	145
V.8.4.3 Sprint 9.....	146
V.8.4.4 Sprint 10.....	146
V.8.4.5 Sprint 11.....	147
V.8.4.6 Sprint 12.....	148
V.8.4.7 Sprint 13.....	149
V.8.4.8 Sprint 14.....	150
V.8.4.9 Sprint 15/16 .....	152
<b>V.8.5 Stabilitätstest .....</b>	<b>154</b>
<b>V.8.6 Dokumentation der Realisierung .....</b>	<b>156</b>
V.8.6.1 Übereinstimmung mit Architektur .....	156
V.8.6.1.1 Videowall-Applikation .....	156
V.8.6.1.2 Poster-Applikation.....	157
V.8.6.1.3 Mittagsmenu-Applikation.....	158
V.8.6.2 Code Statistik.....	159
V.8.6.2.1 Testabdeckung .....	159
V.8.6.2.1.1 Videowall-Applikation.....	159
V.8.6.2.1.2 Poster-Applikation.....	160
V.8.6.2.1.3 Mittagsmenu-Applikation.....	160
V.8.6.2.2 Lines of Code (LOC) .....	161
V.8.6.3 Code Qualität.....	162
V.8.6.3.1 Visual Studio.....	162
V.8.6.3.1.1 Videowall Applikation.....	162

V.8.6.3.1.2	Poster-Applikation.....	163
V.8.6.3.1.3	Mittagsmenu-Applikation.....	163
V.8.6.3.2	NDepend .....	163
V.8.6.3.2.1	Videowall-Applikation.....	163
V.8.6.3.2.2	Poster-Applikation.....	164
V.8.6.3.2.3	Mittagsmenu-Applikation.....	165
V.8.6.3.3	Code Warnungen .....	165
V.8.6.4	Coding Standards .....	167
V.8.6.4.1	C# Namenskonventionen.....	167
V.8.6.4.2	Formatierungsstil .....	167
V.8.6.4.3	Auswertung durch ReSharper .....	168
V.8.6.4.4	Cleanup .....	171
V.8.6.5	Dokumentation Quellcode .....	171
V.8.6.5.1	Generierung der Dokumentation .....	171
<b>V.8.7</b>	<b>Beschreibung der Applikationen.....</b>	<b>172</b>
V.8.7.1	Hauptapplikation.....	172
V.8.7.2	Plug-ins.....	172
V.8.7.2.1	Poster Plug-in.....	172
V.8.7.2.2	Mittagsmenu Plug-in .....	172
V.8.7.2.3	Experiment Browser Plug-in.....	172
V.8.7.2.4	Experiment Diagnostic Plug-in.....	172
V.8.7.2.5	Experiment Prezi Plug-in.....	172
V.8.7.2.6	Experiment ShapeGame Plug-in.....	172
V.8.7.3	Mini-Applikationen .....	172
V.8.7.3.1	DemoMode .....	173
V.8.7.3.2	DesignMenu .....	173
V.8.7.3.3	DesignPosterNavigationButtons.....	173
V.8.7.3.4	HandCursorDemoApp .....	173
V.8.7.3.5	KinectHandTracker.....	173
V.8.7.3.6	KinectRecorder .....	173
V.8.7.3.7	KinectWpfViewers .....	173
V.8.7.3.8	ObjectTrackingVisualizer.....	173
V.8.7.3.9	PdfConverter .....	173
V.8.7.3.10	PluginDemo .....	173
V.8.7.3.11	TestXna.....	173
V.8.7.3.12	VideoWithWPF .....	174
V.8.7.3.13	WizardOfOzTest.....	174
V.8.7.3.14	KinectCursorSmoothing .....	174

<b>V.8.8</b>	<b>Code Reviews .....</b>	<b>175</b>
V.8.8.1	Übersicht.....	175
V.8.8.2	Kriterien .....	175
V.8.8.2.1	Code Style Analyse.....	175
V.8.8.2.2	Exception Handling.....	175
V.8.8.2.3	Flow Control.....	175
V.8.8.2.4	Naming.....	176
V.8.8.2.5	Tools .....	176
V.8.8.3	Code Review vom 19.04.2012 .....	176
V.8.8.3.1	Bewertung der Kriterien .....	176
V.8.8.3.1.1	Code Style Analyse.....	176
V.8.8.3.1.2	Exception Handling.....	177
V.8.8.3.1.3	Flow Control .....	177
V.8.8.3.1.4	Naming .....	177
V.8.8.3.1.5	Tools.....	177
V.8.8.4	Code Review vom 03.05.2012 .....	177
V.8.8.4.1	Bewertung der Kriterien .....	178
V.8.8.4.1.1	Code Style Analyse.....	178
V.8.8.4.1.2	Exception Handling.....	179
V.8.8.4.1.3	Flow Control .....	179
V.8.8.4.1.4	Naming .....	179
V.8.8.4.1.5	Tools.....	179
V.8.8.5	Code Review vom 05.06.2012 .....	179
V.8.8.5.1	Bewertung der Kriterien .....	180
V.8.8.5.1.1	Code Style Analyse.....	180
V.8.8.5.1.2	Exception Handling.....	181
V.8.8.5.1.3	Flow Control .....	181
V.8.8.5.1.4	Naming .....	181
V.8.8.5.1.5	Tools.....	181

---

### V.8.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
16.04.2012	1.0	Erste Version des Dokuments	DT
19.04.2012	1.1	Dokumentation Code Review 19.04.12	LE
22.04.2012	1.2	Usability Tests	CH
23.04.2012	1.3	Systemtests	DT
24.04.2012	1.4	Review	DT
27.04.2012	1.5	Systemtests	DT
07.05.2012	1.6	Systemtests	DT
07.05.2012	1.7	Dokumentation Code Review 03.05.2012	DT
11.05.2012	1.8	Systemtests	DT
12.05.2012	1.9	Einfügen Testdokumentation aus Domain Analyse	DT
14.05.2012	1.10	Systemtests	DT
18.05.2012	1.11	Usability Test: Reaktion auf Demomodus	CH
18.05.2012	1.12	Review	DT
18.05.2012	1.13	Extension Framework	LE
19.05.2012	1.14	Ergänzung Wizard of Oz - Test mit Bild	DT
22.05.2012	1.15	Review Extension Framework	DT
22.05.2012	1.16	Systemtests	DT
22.05.2012	1.17	Usability Test: Grafisches Design	DT
23.05.2012	1.18	Coding Standards & Dokumentation Quellcode	CH
23.05.2012	1.19	Review Usability Test: Grafisches Design	CH
29.05.2012	1.20	Systemtests	CH
31.05.2012	1.21	Sandcastle Help File Builder	DT
02.06.2012	1.20	Systemtests	DT
07.06.2012	1.21	Installationsdokumentation	CH
07.06.2012	1.22	Unit Tests	DT
07.06.2012	1.22	Dokumentation Code Review 05.06.2012	LE
08.06.2012	1.23	Code Metriken	CH
09.06.2012	1.23	Review Code Reviews	CH
10.06.2012	1.24	Review und Korrekturen	LE
11.06.2012	1.25	Review Installationsanleitung	DT
11.06.2012	1.26	Code Statistik	CH
12.06.2012	1.27	Systemtests	CH
12.06.2012	1.28	Stabilitätstests	LE
12.06.2012	1.29	Review	DT
14.06.2012	1.30	Korrekturen	DT
14.06.2012	1.31	Korrekturen	CH

## V.8.2 Usability Tests

### V.8.2.1 Test 1: Wizard of Oz

Am 27. März 2012 wurde der Test (für die Erarbeitung siehe V.5.4.1 Empirischer formativer Test zur Eruierung der Navigationsart) durchgeführt. Bei diesem galt es das Konzept „Meine Hand ist die Maus“ zu bestätigen. Weiter sollte der Test sicherstellen, dass die Nutzer durch die Poster browsen und im Menu navigieren können. Um dies zu prüfen, wurde mithilfe einer WPF-Applikation ein Wizard of Oz - Experiment durchgeführt.

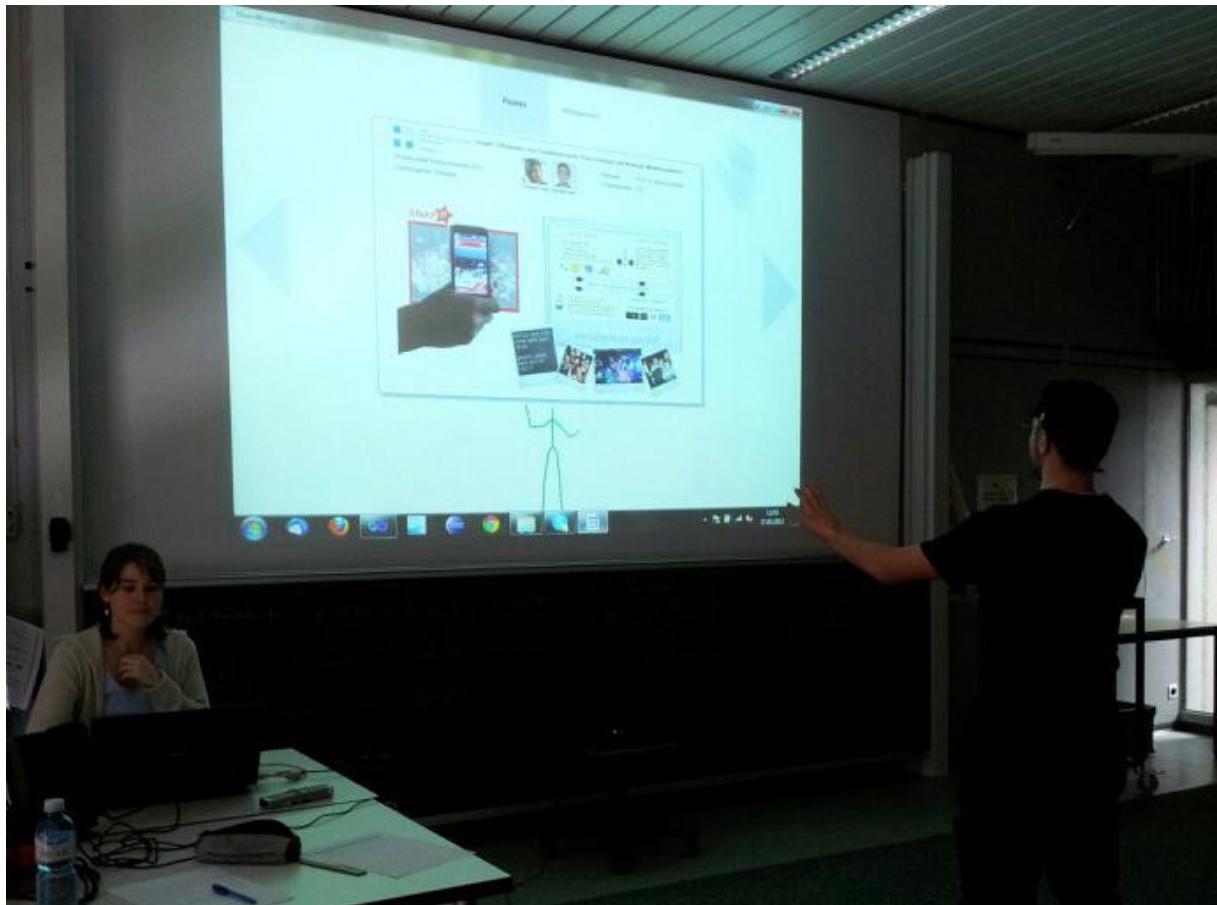


Abbildung 94 - Testdurchführung Wizard of Oz mit einem Probanden

Der Test wurde mit sieben Personen durchgeführt, welche das nachfolgende Testszenario (siehe V.8.2.1.1 Testszenario) durchspielten. Die Testpersonen wurden gebeten, laut mitzudenken.

Eine genaue Beschreibung der Testapplikation findet sich im Unterkapitel V.5.4.1.3 Umsetzung.

#### V.8.2.1.1 Testszenario

Du bist Student/in an der HSR und warst heute Morgen von 8 bis 10 Uhr in einer Vorlesung. Es ist nun Pause und du gehst gerade in die Mensa, um ein Brötchen zu kaufen. Dabei fällt dir die grosse Monitorwand im Eingangsbereich des Gebäudes 4 auf. Du gehst auf die Wall zu.

#### V.8.2.1.1.1 Aufgabe

Du stehst nun also vor der grossen Monitorwand (hier im Test ist das die Projektion des Beamers). Du bist neugierig und möchtest herausfinden, was die Videowall alles für Funktionen bietet.

### V.8.2.1.2 Resultat & Fazit

Alle Testpersonen konnten die im Testszenario gestellte Aufgabe ohne grosse Probleme lösen. Die Beobachtungen und Notizen, welche während der Durchführung des Tests gemacht wurden, sind in der nachfolgenden Tabelle zusammengefasst, die während des Tests gemachten Notizen befinden sich im Anhang (VIII Anhang).

<b>Testperson kam insgesamt ... zurecht.</b>	3 x sehr gut	4 x gut
<b>Testperson hatte Schwierigkeiten bei der Bearbeitung der Aufgabe.</b>	7 x gar nicht	
<b>Testperson zögerte bei der Bearbeitung der Aufgabe.</b>	5 x gar nicht	1 x mittelmässig    1 x ziemlich
<b>Testperson war langsam bei der Bearbeitung der Aufgabe.</b>	6 x gar nicht	1 x kaum
<b>Testperson positionierte sich von Anfang an korrekt.</b>	Sechs von sieben Testpersonen positionierten sich von Anfang an mit dem richtigen Abstand zur Wand und dem Kinect.	
<b>Testperson merkte ..., dass das Skelett ihre Bewegungen imitiert.</b>	5 x ausserordentlich schnell	1 x ziemlich schnell    1 x fast bis zum Schluss nicht

Tabelle 20 - Zusammenfassung Resultat empirischer formativer Test

Weitere Beobachtungen:

- Vier Testpersonen wollten die Schaltfläche (Pfeil oder Menu-Button) mit einer Vorwärtsbewegung der Hand oder durch das Machen einer Faust betätigen.
- Vier Testpersonen hätten gerne das Poster mit einer Zoom-Geste vergrössert.
- Vier Testpersonen wollten die Bilder auf den Postern oder das Poster insgesamt anklicken.
- Zwei Testpersonen wollten auch mit der linken Hand steuern.
- Zwei Testpersonen wollten mit einer Wisch-Geste zum nächsten Poster übergehen.

Weiter merkten die Testpersonen an, dass:

- Sie sich auch vorstellen kann, dass das Poster grösser wird, wenn er näher zur Wall geht.
- Sie sich vorstellen kann auch mit einem Doppelklick oder über eine Zoomleiste (Slider) zu zoomen.

Bei diesem Test wurde die Applikation nicht mit mehr als sieben Personen geprüft, weil die Resultate so deutlich waren, dass eine statistische Analyse nicht für nötig befunden wurde.

Das Fazit des Tests ist, dass das Konzept „Meine Hand ist die Maus“ bestätigt werden konnte. Es zeigte sich, dass die Steuerung intuitiv und eine Hilfestellung daher überflüssig ist. Der Test zeigte auch, dass sich ein Grossteil der Testpersonen auch ohne Markierungen am Boden mit dem optimalen Abstand zum Sensor vor der Projektion positionierte. Zusätzlich konnte auch das GUI verifiziert werden. Für die Testpersonen war sehr schnell klar, für was die Pfeile und das Menu verwendet werden können.

Aufgrund dieses Resultats wird die Applikation so weiterentwickelt, dass die Videowall nicht mit Gesten sondern nur mit der Hand gesteuert wird. Weiter wurde bestimmt, dass nur eine Person gleichzeitig die Videowall steuern kann. Die Person, welche näher am Sensor steht, übernimmt die Steuerung. Das Skelett, welches angezeigt wird, ist immer das des aktiven Benutzers.

### V.8.2.2 Test 2: Reaktion der Nutzer

Nachdem die gewünschte Steuerung der Wall über die Hand implementiert wurde, entschied sich das Team dazu, deren Eignung am 20.04.2012 nochmals zu testen. Zudem sollte beobachtet werden, wie Passanten im Verwaltungsgebäude auf die Videowall reagieren.

Der Test wurde im Eingangsbereich des Gebäudes 4 aufgestellt. Da sich an der Wand, an welcher die Videowall installiert werden soll, zurzeit noch ein Infostand befindet, wurde die gegenüberliegende Wand genutzt. Um die Videowall mit einfachen Mitteln nachstellen zu können, wurde ein Kurzdistanzbeamer verwendet, welcher die Applikation, welche von einem Laptop aus gestartet wurde, an die Wand projizierte. Kinect konnte nicht direkt unterhalb der Projektion platziert werden, da sonst der Kurzdistanzbeamer genau im Interaktionsbereich der Applikation gelegen und eine Bedienung durch den Nutzer verunmöglich hätte. Es wurde daher entschieden, den Sensor in den Bereich zwischen der Wand und dem Beamer, leicht hinter den Beamer

versetzt, zu stellen. Somit ergab sich zwischen dem Sensor und der Zone, welche die meisten Passanten auf dem Weg in die Mensa durchlaufen, ein optimaler Erkennungsabstand von drei bis vier Metern.

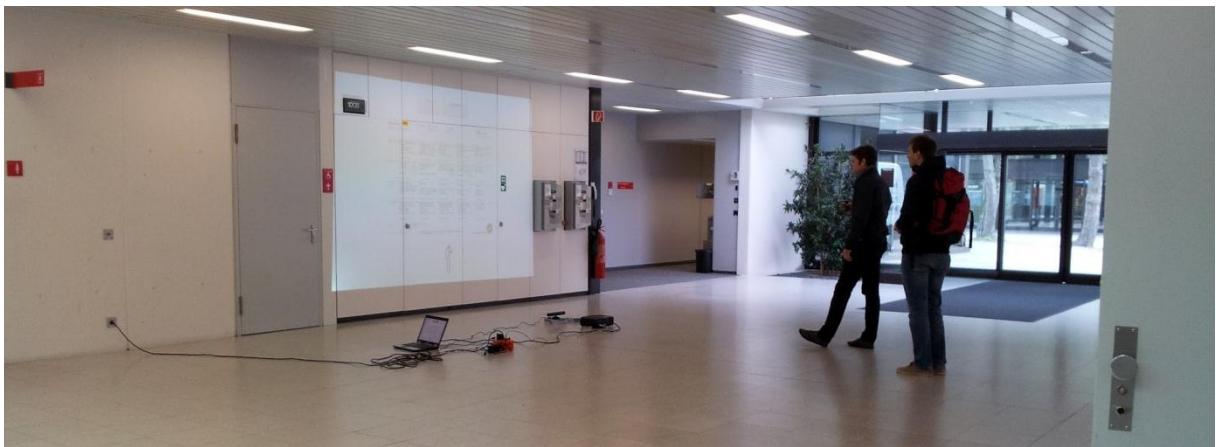


Abbildung 95 - Test 2: Reaktion der Nutzer

Schon das Aufstellen der Geräte im Verwaltungsgebäude zog grosse Aufmerksamkeit auf sich. Als die Applikation schliesslich gestartet war, wurde sie von praktisch allen Passanten registriert. Es liessen sich jedoch nicht alle dazu animieren, anzuhalten und die Applikation genauer zu betrachten. Dies könnte durch einen Teaser wesentlich verbessert werden. Die meisten Passanten wurden erst neugierig auf die Applikation, als ihr Skelett im Vorbeigehen im unteren Bildschirmbereich auftauchte.

Die Steuerung mit der Hand wurde von den meisten schnell verstanden, einige wenige begnügten sich einfach damit, einige Verrenkungen zu machen und zuzuschauen, wie das Skelett diese nachmacht.

Obwohl die Handsteuerung bei kleinen Tests in der Testumgebung des Bachelorzimmers ohne grosse Probleme funktionierte, zuckte der Handcursor bei diesem Usability Test merklich. Dies führte dazu, dass einige Benutzer schnell das Interesse an der Applikation verloren, da diese so schwierig zu bedienen war. Die Verbesserung der Steuerung wurde daher als wichtiger nächster Punkt in den bevorstehenden Tätigkeiten des Teams aufgeführt und als User Story erfasst.

#### V.8.2.3 Test 3: Reaktion auf Demomodus



Abbildung 96 - Test 3: Reaktion auf Demomodus

Nachdem der Demomodus implementiert war, sollte auch dieser wieder von potenziellen Nutzern geprüft werden. Daher testete das Team den Demomodus am 18.05.2012. Der Aufbau verlief wieder gleich wie schon beim Usability Test V.8.2.2 Test 2: Reaktion der Nutzer.

Auch dieses Mal liessen sich nicht alle Personen dazu animieren, vor der Wall stehen zu bleiben. Vor allem Einzelpersonen widmeten der Installation höchstens einen kurzen Blick, gingen aber zielgerichtet daran vorbei. Gruppen blieben hingegen eher stehen. Ein Gruppenmitglied interagierte mit der Wall und die anderen schauten zu. Auch nachdem der Demomodus die Applikation komplettiert, ist das Skelett die Attraktion. Die Benutzer verrenkten sich vor der Wall, um zu sehen, wie das Skelett diese Bewegungen nachahmt.

Folgende Nachteile der aktuellen Implementation des Demomodus wurden durch diesen Test ersichtlich:

- Sobald eine Person im Demomodus erkannt wird, verschwindet der Teaser-Text und ein Timer wird gestartet, der Countdown wird angezeigt. Es dauert zu lange, bis der Timer aktiviert wird, da der Abfragerhythmus zur Erkennung von Passanten noch nicht optimal ist. Daher muss das Intervall zwischen zwei Skelettabfragen verkürzt werden. Ansonsten haben die Personen die Wall schon passiert, bevor eine Reaktion der Wall auf die Erkennung ihres Skelettes ersichtlich ist.
- Das Skelett sollteam besten schon beim Herunterzählen des Countdowns (also noch im Demomodus) angezeigt werden, damit dem Nutzer klar ist, dass er erkannt wurde.
- Der Teaser-Text der Poster-Applikation muss überdacht werden. Dieser lautete „Willst du etwas lernen?“ – was von einigen Passanten im Vorbeilaufen lautstark mit „Nein!“ beantwortet wurde.

Nach der Behebung dieser Nachteile wird die Applikation erneut einem Usability Test unterzogen.

#### V.8.2.4 Test 4: Grafisches Design

Mit diesem Usability Test sollteam 22.05.2012 getestet werden, ob das grafische Design verständlich ist. Weiter wurde sichergestellt, dass sich die Verbesserungen am Demomodus, welche nach dem vorhergehenden Demomodus - Usability Test (siehe Unterkapitel V.8.2.3 Test 3: Reaktion auf Demomodus) vorgenommen wurden, bewähren. Der Aufbau verlief wieder gleich wie schon beim Test V.8.2.2 Test 2: Reaktion der Nutzer.



**Abbildung 97 - Test 4: Grafisches Design**

Folgende Beobachtungen konnten während der Durchführung des Tests gemacht werden:

- Die Tabs im Menu sind noch nicht deutlich als Tabs ersichtlich, weshalb einige Nutzer nicht wussten, wo sie klicken können.
- Der Handcursor soll sich nicht drehen, wenn er sich im Menu auf dem Tab befindet, welches bereits aktiv ist. Einige Benutzer versuchten im Menu zu den Postern zu wechseln, obwohl diese Applikation bereits aktiv war.
- Einige Nutzer versuchten, das Mittagsmenu oder Elemente auf den Postern anzuklicken. Der Handcursor soll, je nachdem, ob er sich über einem interaktiven Objekt (z.B. ein Button) befindet oder nicht, anders gekennzeichnet sein. Beispielsweise soll die Hand durchgestrichen sein oder das Bild soll mehr Transparenz haben.

Der erste Beobachtungspunkt wird noch im Rahmen dieser Arbeit umgesetzt. Die anderen zwei Beobachtungen wurden als User Stories in den Backlog aufgenommen (siehe V.4.3 Funktionale Anforderungen).

### V.8.3 Unit Tests

#### V.8.3.1 Videowall-Applikation

Für die Solution VideoWall wurden 21 Unit Tests geschrieben und erfolgreich ausgeführt.

The screenshot shows a Windows application window titled "Test run completed". The status bar indicates "dtreichl@PIN1258025 2012-06-07". The main area displays a table of test results:

Result	Test Name	Project
Passed	ExtensionsConfigConstructorTest	VideoWall.Tests
Passed	AppChangedMethodTest	VideoWall.Tests
Passed	GetResourcesTest	VideoWall.Tests
Passed	AppViewModelConstructorTest	VideoWall.Tests
Passed	ImageTest	VideoWall.Tests
Passed	MenuStyleViewModelConstructorTest	VideoWall.Tests
Passed	DemoModeServiceConstructorTest	VideoWall.Tests
Passed	GetExtensionTestThrowing	VideoWall.Tests
Passed	GetExtensionTest	VideoWall.Tests
Passed	AppControllerConstructorTestInternals	VideoWall.Tests
Passed	ChangeAppTest	VideoWall.Tests
Passed	AppControllerConstructorTest	VideoWall.Tests
Passed	ExtensionsConfigConstructorTestThrowing	VideoWall.Tests
Passed	FileServiceConstructorTestWithException	VideoWall.Tests
Passed	FileServiceConstructorTest	VideoWall.Tests
Passed	MenuViewModelConstructorTest	VideoWall.Tests
Passed	DemoModeStateTimersConstructorTest	VideoWall.Tests
Passed	DemoModeViewModelConstructorTest	VideoWall.Tests
Passed	HandLeftTest	VideoWall.Tests
Passed	HandRightTest	VideoWall.Tests
Passed	ChangedStatusMethodTest	VideoWall.Tests

Abbildung 98 - Unit Tests VideoWall

#### V.8.3.2 Poster-Applikation

Für die Solution PosterApp wurden 6 Unit Tests geschrieben und erfolgreich ausgeführt.

The screenshot shows a Windows application window titled "Test run warning". The status bar indicates "dtreichl@PIN1258025 2012-06-07". The main area displays a table of test results:

Result	Test Name	Project
Passed	PosterReaderConstructorTest	PosterApp.Tests
Passed	PosterConstructorTest	PosterApp.Tests
Passed	PosterViewModelConstructorTest	PosterApp.Tests
Passed	NavigateToLeftCommandTest	PosterApp.Tests
Passed	NavigateToRightCommandTest	PosterApp.Tests
Passed	PosterServiceConstructorTest	PosterApp.Tests

Abbildung 99 - Unit Tests PosterApp

### V.8.3.3 Mittagsmenü-Applikation

Für die Solution LunchMenuApp wurden 16 Tests geschrieben und erfolgreich ausgeführt.

The screenshot shows a Windows application window titled "Test run completed". The status bar indicates "dtreichl@PIN1258025 2012-06-07". The main area displays a table of test results:

Result	Test Name	Project
Passed	LunchMenuViewModelLunchMenuChangedTest	LunchMenuApp.Tests
Passed	LunchMenuServiceConstructorTestWhenThrowing	LunchMenuApp.Tests
Passed	DishConstructorTest	LunchMenuApp.Tests
Passed	LunchMenuViewModelConstructorTestWithoutLunchMenu	LunchMenuApp.Tests
Passed	LunchMenuConstructorTest	LunchMenuApp.Tests
Passed	ExtractDateTest	LunchMenuApp.Tests
Passed	LunchMenuConstructorTestWhenReaderThrows	LunchMenuApp.Tests
Passed	DishViewModelConstructorTest	LunchMenuApp.Tests
Passed	LunchMenuServiceConstructorTest	LunchMenuApp.Tests
Passed	ExtractMenusTest	LunchMenuApp.Tests
Passed	LunchMenuParserConstructorTestWrongContent	LunchMenuApp.Tests
Passed	LunchMenuViewModelConstructorTest	LunchMenuApp.Tests
Passed	LunchMenuParserConstructorTest	LunchMenuApp.Tests
Passed	LunchMenuParserConstructorTestWrongHtml	LunchMenuApp.Tests
Passed	LunchMenuReaderConstructorTestThrowingWebException	LunchMenuApp.Tests
Passed	LunchMenuReaderConstructorTest	LunchMenuApp.Tests

Abbildung 100 - Unit Tests LunchMenuApp

## V.8.4 Systemtests

Die Systemtests orientieren sich an den definierten User Stories, die dann im entsprechenden Sprint umgesetzt wurden.

### V.8.4.1 Sprint 7

Testperson: Delia Treichler A = im Architekturprototypen enthalten

Ticket#	Titel	Beschreibung	Resultat	Datum
A	Poster werden angezeigt	Es wird das aktuelle Poster angezeigt.	Ok	16.04.2012
A	Poster browsen	Es kann zum nächsten und zum vorhergehenden Poster gewechselt werden.	Ok	16.04.2012
A	Handcursor wird dargestellt	Der Handcursor wird als blauer Punkt dargestellt.	Ok	16.04.2012
A	Eigenes Skelett wird dargestellt	Die verschiedenen Skelettjoints des Benutzers werden angezeigt.	Ok	16.04.2012
A	Sofortiges Erfolgserlebnis für Einstieg sichergestellt	Das sofortige Erfolgserlebnis ist durch das Skelett sichergestellt.	Ok	16.04.2012

Tabelle 21 - Systemtests Sprint 7

### V.8.4.2 Sprint 8

Testperson: Delia Treichler A = im Architekturprototypen enthalten

Ticket#	Titel	Beschreibung	Resultat	Datum
A	Poster werden angezeigt	Es wird das aktuelle Poster angezeigt.	Ok	23.04.2012
A	Poster browsen	Es kann zum nächsten und zum vorhergehenden Poster gewechselt werden.	Ok	23.04.2012
A	Handcursor wird dargestellt	Der Handcursor wird als blauer Punkt dargestellt.	Ok	16.04.2012
A	Eigenes Skelett wird dargestellt	Die verschiedenen Skelettjoints des Benutzers werden angezeigt.	Ok	16.04.2012
A	Sofortiges Erfolgserlebnis für Einstieg sichergestellt	Das sofortige Erfolgserlebnis ist durch das Skelett sichergestellt.	Ok	23.04.2012
769	Pointer für die Hand schön dargestellt	Der Handcursor wird als rechte Hand dargestellt.	Ok	23.04.2012
779	Skelett schön dargestellt	Das Skelett wird als Strichmännlein dargestellt.	Ok	23.04.2012
870	Handcursor ruckelt weniger 1	Der Handcursor zittert nicht mehr so fest.	Ok	23.04.2012

Tabelle 22 - Systemtests Sprint 8

### V.8.4.3 Sprint 9

Testperson: Delia Treichler

A = im Architekturprototypen enthalten

Ticket#	Titel	Beschreibung	Resultat	Datum
A	Poster werden angezeigt	Es wird das aktuelle Poster angezeigt.	Ok	27.04.2012
A	Poster browsen	Es kann zum nächsten und zum vorhergehenden Poster gewechselt werden.	Ok	27.04.2012
A	Handcursor wird dargestellt	Der Handcursor wird als blauer Punkt dargestellt.	Ok	16.04.2012
A	Eigenes Skelett wird dargestellt	Die verschiedenen Skelettjoints des Benutzers werden angezeigt.	Ok	16.04.2012
A	Sofortiges Erfolgserlebnis für Einstieg sichergestellt	Das sofortige Erfolgserlebnis ist durch das Skelett sichergestellt.	Ok	27.04.2012
769	Pointer für die Hand schön dargestellt	Der Handcursor wird als rechte Hand dargestellt.	Ok	27.04.2012
779	Skelett schön dargestellt	Das Skelett wird als Strichmännlein dargestellt.	Ok	27.04.2012
870	Handcursor ruckelt weniger 1	Der Handcursor zittert nicht mehr so fest.	Ok	27.04.2012
786	Video wird dargestellt	Das Video wird in WPF dargestellt.	Ok	27.04.2012
785	Applikation ist mit linker Hand bedienbar	Die Applikation kann sowohl mit der rechten als auch der linken Hand bedient werden.	Ok	27.04.2012

Tabelle 23 - Systemtests Sprint 9

### V.8.4.4 Sprint 10

Testperson: Delia Treichler

A = im Architekturprototypen enthalten

Ticket#	Titel	Beschreibung	Resultat	Datum
A	Poster werden angezeigt	Es wird das aktuelle Poster angezeigt.	Ok	07.05.2012
A	Poster browsen	Es kann zum nächsten und zum vorhergehenden Poster gewechselt werden.	Ok	07.05.2012
A	Handcursor wird dargestellt	Der Handcursor wird als blauer Punkt dargestellt.	Ok	16.04.2012
A	Eigenes Skelett wird dargestellt	Die verschiedenen Skelettjoints des Benutzers werden angezeigt.	Ok	16.04.2012
A	Sofortiges Erfolgserlebnis für Einstieg sichergestellt	Das sofortige Erfolgserlebnis ist durch das Skelett sichergestellt.	Ok	07.05.2012
769	Pointer für die Hand schön dargestellt	Der Handcursor wird als rechte Hand dargestellt.	Ok	07.05.2012
779	Skelett schön dargestellt	Das Skelett wird als Strichmännlein dargestellt.	Ok	07.05.2012
870	Handcursor ruckelt weniger 1	Der Handcursor zittert nicht mehr so fest.	Ok	07.05.2012
786	Video wird dargestellt	Das Video wird in WPF dargestellt.	Ok	07.05.2012

Ticket#	Titel	Beschreibung	Resultat	Datum
785	Applikation ist mit linker Hand bedienbar	Die Applikation kann sowohl mit der rechten als auch der linken Hand bedient werden.	Ok	07.05.2012
798	Plug-in Möglichkeit	Ein Plug-in kann automatisch in die Main-Applikation geladen werden.	Ok	07.05.2012

Tabelle 24 - Systemtests Sprint 10

#### V.8.4.5 Sprint 11

Testperson: Delia Treichler A = im Architekturprototypen enthalten

Ticket#	Titel	Beschreibung	Resultat	Datum
A	Poster werden angezeigt	Es wird das aktuelle Poster angezeigt.	Ok	14.05.2012
A	Poster browsen	Es kann zum nächsten und zum vorhergehenden Poster gewechselt werden.	Ok	14.05.2012
A	Handcursor wird dargestellt	Der Handcursor wird als blauer Punkt dargestellt.	Ok	16.04.2012
A	Eigenes Skelett wird dargestellt	Die verschiedenen Skelettjoints des Benutzers werden angezeigt.	Ok	16.04.2012
A	Sofortiges Erfolgserlebnis für Einstieg sichergestellt	Das sofortige Erfolgserlebnis ist durch das Skelett sichergestellt.	Ok	14.05.2012
769	Pointer für die Hand schön dargestellt	Der Handcursor wird als rechte Hand dargestellt.	Ok	14.05.2012
779	Skelett schön dargestellt	Das Skelett wird als Strichmännlein dargestellt.	Ok	14.05.2012
870	Handcursor ruckelt weniger 1	Der Handcursor zittert nicht mehr so fest.	Ok	14.05.2012
786	Video wird dargestellt	Das Video wird in WPF dargestellt.	Ok	14.05.2012
785	Applikation ist mit linker Hand bedienbar	Die Applikation kann sowohl mit der rechten als auch der linken Hand bedient werden.	Ok	14.05.2012
798	Plug-in Möglichkeit	Ein Plug-in kann automatisch in die Main-Applikation geladen werden.	Ok	14.05.2012
833	Demomodus: Vom Demomodus wird zum Interaktionsmodus gewechselt	Wenn die Applikation im Demomodus ist und ich sie bedienen möchte (Skeletterkennung), wechselt sie automatisch in den Interaktionsmodus.	Ok	14.05.2012
834	Demomodus: Vom Interaktionsmodus wird zum Demomodus gewechselt	Wenn die Applikation im Interaktionsmodus ist und niemand die Applikation bedient (Skeletterkennung), so wechselt sie automatisch in den Demomodus.	Ok	14.05.2012
799	Bild der Hand ist auf die rechte bzw. linke Hand abgestimmt	Bediene ich die Applikation mit der rechten Hand, so wird der Cursor als rechte Hand dargestellt. Bediene ich die Applikation mit der linken Hand, so ist das Bild des Cursors eine linke Hand.	Ok	14.05.2012

Tabelle 25 - Systemtests Sprint 11

## V.8.4.6 Sprint 12

Testperson: Delia Treichler

A = im Architekturprototypen enthalten

Ticket#	Titel	Beschreibung	Resultat	Datum
A	Poster werden angezeigt	Es wird das aktuelle Poster angezeigt.	Ok	22.05.2012
A	Poster browsen	Es kann zum nächsten und zum vorhergehenden Poster gewechselt werden.	Ok	22.05.2012
A	Handcursor wird dargestellt	Der Handcursor wird als blauer Punkt dargestellt.	Ok	16.04.2012
A	Eigenes Skelett wird dargestellt	Die verschiedenen Skelettjoints des Benutzers werden angezeigt.	Ok	16.04.2012
A	Sofortiges Erfolgserlebnis für Einstieg sichergestellt	Das sofortige Erfolgserlebnis ist durch das Skelett sichergestellt.	Ok	22.05.2012
769	Pointer für die Hand schön dargestellt	Der Handcursor wird als rechte Hand dargestellt.	Ok	22.05.2012
779	Skelett schön dargestellt	Das Skelett wird als Strichmännlein dargestellt.	Ok	22.05.2012
870	Handcursor ruckelt weniger 1	Der Handcursor zittert nicht mehr so fest.	Ok	22.05.2012
786	Video wird dargestellt	Das Video wird in WPF dargestellt.	Ok	22.05.2012
785	Applikation ist mit linker Hand bedienbar	Die Applikation kann sowohl mit der rechten als auch der linken Hand bedient werden.	Ok	22.05.2012
798	Plug-in Möglichkeit	Ein Plug-in kann automatisch in die Main-Applikation geladen werden.	Ok	22.05.2012
800	Mittagsmenu App in Plug-in umgewandelt	Die Mittagsmenu-Applikation besteht als Plug-in und kann durch Nr. 798 in der Hauptapplikation angezeigt werden.	Ok	22.05.2012
856	Das Mittagsmenu wird angezeigt.	Das Mittagsmenu wird in der Wochenübersicht angezeigt.	Ok	22.05.2012
802	Poster App in Plug-in App umgewandelt	Die Poster-Applikation besteht als Plug-in und kann durch Nr. 798 in der Hauptapplikation angezeigt werden.	Ok	22.05.2012
833	Demomodus: Vom Demomodus wird zum Interaktionsmodus gewechselt	Wenn die Applikation im Demomodus ist und ich sie bedienen möchte (Skeletterkennung), wechselt sie automatisch in den Interaktionsmodus.	Ok	22.05.2012
834	Demomodus: Vom Interaktionsmodus wird zum Demomodus gewechselt	Wenn die Applikation im Interaktionsmodus ist und niemand die Applikation bedient (Skeletterkennung), so wechselt sie automatisch in den Demomodus.	Ok	22.05.2012
836	Demomodus: Demotext zu aktiver App wird angezeigt	Wenn die Applikation im Demomodus ist, wird ein attraktiver Teaser-Text angezeigt.	Ok	22.05.2012
835	Demomodus: Apps werden automatisch gewechselt	Wenn die Applikation im Demomodus ist und sich niemand für die Applikation interessiert, so wechselt der Text nach einer definierten Zeit.	Ok	22.05.2012

Ticket#	Titel	Beschreibung	Resultat	Datum
799	Bild der Hand ist auf die rechte bzw. linke Hand abgestimmt	Bediene ich die Applikation mit der rechten Hand, so wird der Cursor als rechte Hand dargestellt. Bediene ich die Applikation mit der linken Hand, so ist das Bild des Cursors eine linke Hand.	Ok	22.05.2012
856	Mittagsmenu App automatisch aktualisiert	Das Mittagsmenu für den aktuellen Tag wird angezeigt.	Ok	22.05.2012
872	Navigation mit schönen "Tabs" ermöglichen	Die Navigation findet über die Tabs im Menu statt.	Ok	22.05.2012
855	Deployment Entwickler PC möglich	Die Solution kann nach dem SVN-Checkout geöffnet und es kann daran gearbeitet werden.	Ok	22.05.2012

Tabelle 26 - Systemtests Sprint 12

#### V.8.4.7 Sprint 13

Testperson: Christina Heidt A = im Architekturprototypen enthalten

Ticket#	Titel	Beschreibung	Resultat	Datum
A	Poster werden angezeigt	Es wird das aktuelle Poster angezeigt.	Ok	29.05.2012
A	Poster browsen	Es kann zum nächsten und zum vorhergehenden Poster gewechselt werden.	Ok	29.05.2012
A	Handcursor wird dargestellt	Der Handcursor wird als blauer Punkt dargestellt.	Ok	16.04.2012
A	Eigenes Skelett wird dargestellt	Die verschiedenen Skelettjoints des Benutzers werden angezeigt.	Ok	16.04.2012
A	Sofortiges Erfolgserlebnis für Einstieg sichergestellt	Das sofortige Erfolgserlebnis ist durch das Skelett sichergestellt.	Ok	29.05.2012
769	Pointer für die Hand schön dargestellt	Der Handcursor wird als rechte Hand dargestellt.	Ok	29.05.2012
779	Skelett schön dargestellt	Das Skelett wird als Strichmännlein dargestellt.	Ok	29.05.2012
870	Handcursor ruckelt weniger 1	Der Handcursor zittert nicht mehr so fest.	Ok	29.05.2012
786	Video wird dargestellt	Das Video wird in WPF dargestellt.	Ok	29.05.2012
785	Applikation ist mit linker Hand bedienbar	Die Applikation kann sowohl mit der rechten als auch der linken Hand bedient werden.	Ok	29.05.2012
798	Plug-in Möglichkeit	Ein Plug-in kann automatisch in die Main-Applikation geladen werden.	Ok	29.05.2012
800	Mittagsmenu App in Plug-in umgewandelt	Die Mittagsmenu-Applikation besteht als Plug-in und kann durch Nr. 798 in der Hauptapplikation angezeigt werden.	Ok	29.05.2012
856	Das Mittagsmenu wird angezeigt.	Das Mittagsmenu wird in der Wochenübersicht angezeigt.	Ok	29.05.2012
802	Poster App in Plug-in App umgewandelt	Die Poster-Applikation besteht als Plug-in und kann durch Nr. 798 in der Hauptapplikation angezeigt werden.	Ok	29.05.2012

Ticket#	Titel	Beschreibung	Resultat	Datum
833	Demomodus: Vom Demomodus wird zum Interaktionsmodus gewechselt	Wenn die Applikation im Demomodus ist und ich sie bedienen möchte (Skeletterkennung), wechselt sie automatisch in den Interaktionsmodus.	Ok	29.05.2012
834	Demomodus: Vom Interaktionsmodus wird zum Demomodus gewechselt	Wenn die Applikation im Interaktionsmodus ist und niemand die Applikation bedient (Skeletterkennung), so wechselt sie automatisch in den Demomodus.	Ok	29.05.2012
836	Demomodus: Demotext zu aktiver App wird angezeigt	Wenn die Applikation im Demomodus ist, wird ein attraktiver Teaser-Text angezeigt.	Ok	29.05.2012
835	Demomodus: Apps werden automatisch gewechselt	Wenn die Applikation im Demomodus ist und sich niemand für die Applikation interessiert, so wechselt der Text nach einer definierten Zeit.	Ok	29.05.2012
799	Bild der Hand ist auf die rechte bzw. linke Hand abgestimmt	Bediene ich die Applikation mit der rechten Hand, so wird der Cursor als rechte Hand dargestellt. Bediene ich die Applikation mit der linken Hand, so ist das Bild des Cursors eine linke Hand.	Ok	29.05.2012
856	Mittagsmenu App automatisch aktualisiert	Das Mittagsmenu für den aktuellen Tag wird angezeigt.	Ok	29.05.2012
872	Navigation mit schönen "Tabs" ermöglichen	Die Navigation findet über die Tabs im Menu statt.	Ok	29.05.2012
855	Deployment Entwickler PC möglich	Die Solution kann nach dem SVN-Checkout geöffnet und es kann daran gearbeitet werden.	Ok	29.05.2012

Tabelle 27 - Systemtests Sprint 13

#### V.8.4.8 Sprint 14

Im Sprint 14 wurde die letzte User Story implementiert.

Testperson: Delia Treichler A = im Architekturprototypen enthalten

Ticket#	Titel	Beschreibung	Resultat	Datum
A	Poster werden angezeigt	Es wird das aktuelle Poster angezeigt.	Ok	02.06.2012
A	Poster browsen	Es kann zum nächsten und zum vorhergehenden Poster gewechselt werden.	Ok	02.06.2012
A	Handcursor wird dargestellt	Der Handcursor wird als blauer Punkt dargestellt.	Ok	16.04.2012
A	Eigenes Skelett wird dargestellt	Die verschiedenen Skelettjoints des Benutzers werden angezeigt.	Ok	16.04.2012
A	Sofortiges Erfolgserlebnis für Einstieg sichergestellt	Das sofortige Erfolgserlebnis ist durch das Skelett sichergestellt.	Ok	02.06.2012
769	Pointer für die Hand schön dargestellt	Der Handcursor wird als rechte Hand dargestellt.	Ok	02.06.2012
779	Skelett schön dargestellt	Das Skelett wird als Strichmännlein dargestellt.	Ok	02.06.2012
870	Handcursor ruckelt weniger 1	Der Handcursor zittert nicht mehr so fest.	Ok	02.06.2012

Ticket#	Titel	Beschreibung	Resultat	Datum
786	Video wird dargestellt	Das Video wird in WPF dargestellt.	Ok	02.06.2012
785	Applikation ist mit linker Hand bedienbar	Die Applikation kann sowohl mit der rechten als auch der linken Hand bedient werden.	Ok	02.06.2012
798	Plug-in Möglichkeit	Ein Plug-in kann automatisch in die Main-Applikation geladen werden.	Ok	02.06.2012
800	Mittagsmenu App in Plug-in umgewandelt	Die Mittagsmenu-Applikation besteht als Plug-in und kann durch Nr. 798 in der Hauptapplikation angezeigt werden.	Ok	02.06.2012
856	Das Mittagsmenu wird angezeigt.	Das Mittagsmenu wird in der Wochenübersicht angezeigt.	Ok	02.06.2012
802	Poster App in Plug-in App umgewandelt	Die Poster-Applikation besteht als Plug-in und kann durch Nr. 798 in der Hauptapplikation angezeigt werden.	Ok	02.06.2012
833	Demomodus: Vom Demomodus wird zum Interaktionsmodus gewechselt	Wenn die Applikation im Demomodus ist und ich sie bedienen möchte (Skeletterkennung), wechselt sie automatisch in den Interaktionsmodus.	Ok	02.06.2012
834	Demomodus: Vom Interaktionsmodus wird zum Demomodus gewechselt	Wenn die Applikation im Interaktionsmodus ist und niemand die Applikation bedient (Skeletterkennung), so wechselt sie automatisch in den Demomodus.	Ok	02.06.2012
836	Demomodus: Demotext zu aktiver App wird angezeigt	Wenn die Applikation im Demomodus ist, wird ein attraktiver Teaser-Text angezeigt.	Ok	02.06.2012
835	Demomodus: Apps werden automatisch gewechselt	Wenn die Applikation im Demomodus ist und sich niemand für die Applikation interessiert, so wechselt der Text nach einer definierten Zeit.	Ok	02.06.2012
799	Bild der Hand ist auf die rechte bzw. linke Hand abgestimmt	Bediene ich die Applikation mit der rechten Hand, so wird der Cursor als rechte Hand dargestellt. Bediene ich die Applikation mit der linken Hand, so ist das Bild des Cursors eine linke Hand.	Ok	02.06.2012
856	Mittagsmenu App automatisch aktualisiert	Das Mittagsmenu für den aktuellen Tag wird angezeigt.	Ok	02.06.2012
872	Navigation mit schönen "Tabs" ermöglichen	Die Navigation findet über die Tabs im Menu statt.	Ok	02.06.2012
855	Deployment Entwickler PC möglich	Die Solution kann nach dem SVN-Checkout geöffnet und es kann daran gearbeitet werden.	Ok	02.06.2012
871	Externes Design festgelegt und validiert	Als Entwickler möchte ich für die Design User Stories eine "Definition of Done" festlegen können, damit der Abschluss der User Stories validiert werden kann.	Ok	02.06.2012

Tabelle 28 - Systemtests Sprint 14

## V.8.4.9 Sprint 15/16

Im Sprint 15/16 wurde Refactoring betrieben.

Testperson: Christina Heidt      A = im Architekturprototypen enthalten,  
NF = nichtfunktionale Anforderungen

Ticket#	Titel	Beschreibung	Resultat	Datum
A	Poster werden angezeigt	Es wird das aktuelle Poster angezeigt.	Ok	12.06.2012
A	Poster browsen	Es kann zum nächsten und zum vorhergehenden Poster gewechselt werden.	Ok	12.06.2012
A	Handcursor wird dargestellt	Der Handcursor wird als blauer Punkt dargestellt.	Ok	16.04.2012
A	Eigenes Skelett wird dargestellt	Die verschiedenen Skelettjoints des Benutzers werden angezeigt.	Ok	16.04.2012
A	Sofortiges Erfolgserlebnis für Einstieg sichergestellt	Das sofortige Erfolgserlebnis ist durch das Skelett sichergestellt.	Ok	12.06.2012
769	Pointer für die Hand schön dargestellt	Der Handcursor wird als rechte Hand dargestellt.	Ok	12.06.2012
779	Skelett schön dargestellt	Das Skelett wird als Strichmännlein dargestellt.	Ok	12.06.2012
870	Handcursor ruckelt weniger 1	Der Handcursor zittert nicht mehr so fest.	Ok	12.06.2012
786	Video wird dargestellt	Das Video wird in WPF dargestellt.	Ok	12.06.2012
785	Applikation ist mit linker Hand bedienbar	Die Applikation kann sowohl mit der rechten als auch der linken Hand bedient werden.	Ok	12.06.2012
798	Plug-in Möglichkeit	Ein Plug-in kann automatisch in die Main-Applikation geladen werden.	Ok	
800	Mittagsmenu App in Plug-in umgewandelt	Die Mittagsmenu-Applikation besteht als Plug-in und kann durch Nr. 798 in der Hauptapplikation angezeigt werden.	Ok	12.06.2012
856	Das Mittagsmenu wird angezeigt.	Das Mittagsmenu wird in der Wochenübersicht angezeigt.	Ok	12.06.2012
802	Poster App in Plug-in App umgewandelt	Die Poster-Applikation besteht als Plug-in und kann durch Nr. 798 in der Hauptapplikation angezeigt werden.	Ok	12.06.2012
833	Demomodus: Vom Demomodus wird zum Interaktionsmodus gewechselt	Wenn die Applikation im Demomodus ist und ich sie bedienen möchte (Skeletterkennung), wechselt sie automatisch in den Interaktionsmodus.	Ok	12.06.2012
834	Demomodus: Vom Interaktionsmodus wird zum Demomodus gewechselt	Wenn die Applikation im Interaktionsmodus ist und niemand die Applikation bedient (Skeletterkennung), so wechselt sie automatisch in den Demomodus.	Ok	12.06.2012
836	Demomodus: Demotext zu aktiver App wird angezeigt	Wenn die Applikation im Demomodus ist, wird ein attraktiver Teaser-Text angezeigt.	Ok	12.06.2012
835	Demomodus: Apps werden automatisch gewechselt	Wenn die Applikation im Demomodus ist und sich niemand für die Applikation interessiert, so wechselt der Text nach einer definierten Zeit.	Ok	12.06.2012

Ticket#	Titel	Beschreibung	Resultat	Datum
799	Bild der Hand ist auf die rechte bzw. linke Hand abgestimmt	Bediene ich die Applikation mit der rechten Hand, so wird der Cursor als rechte Hand dargestellt. Bediene ich die Applikation mit der linken Hand, so ist das Bild des Cursors eine linke Hand.	Ok	12.06.2012
856	Mittagsmenu App automatisch aktualisiert	Das Mittagsmenu für den aktuellen Tag wird angezeigt.	Ok	12.06.2012
872	Navigation mit schönen "Tabs" ermöglichen	Die Navigation findet über die Tabs im Menu statt.	Ok	12.06.2012
855	Deployment Entwickler PC möglich	Die Solution kann nach dem SVN-Checkout geöffnet und es kann daran gearbeitet werden.	Ok	12.06.2012
871	Externes Design festgelegt und validiert	Als Entwickler möchte ich für die Design User Stories eine "Definition of Done" festlegen können, damit der Abschluss der User Stories validiert werden kann.	Ok	12.06.2012
NF	Nichtfunktionale Anforderungen	Alle nichtfunktionalen Anforderungen wurden geprüft und sind erfüllt.	Ok	12.06.2012

Tabelle 29 - Systemtests Sprint 15/16

## V.8.5 Stabilitätstest

Um die Stabilität der entwickelten Applikation mit Plug-ins zu testen wurde diese über eine Zeitspanne von 111 Stunden auf dem Videowall-Testsetup (siehe V.7.3.3 Testhardware) mit einer hohen Auflösung laufen gelassen. Die Applikation befand sich zu einem grossen Teil im Demomodus, jedoch ab und zu wurde sie bedient.

Das Augenmerk wurde bei diesem Stabilitätstest auf die folgenden zwei Merkmale gelegt:

- Anzahl der offenen Handles: Diese Zahl beschreibt die Anzahl der geöffneten Dateien, Pipes usw. Befindet sich in der Applikation ein Memory Leak, z.B. weil die Dispose-Methode auf einem IDisposable-Objekt nicht aufgerufen wurde, so kann in C# häufig beobachtet werden, wie die Anzahl der offenen Handles einer laufenden Applikation ansteigt.
- Speicherverbrauch in MiB: Diese Zahl beschreibt, wie viel Speicher für die Applikation beim Betriebssystem reserviert ist. Obwohl diese Aufgabe beim Entwickeln mit C# meistens von der .NET Runtime übernommen wird, kann es sein, dass die Speicherzuteilung bei einem Memory Leak nicht mehr korrekt ist. Dies passiert beispielsweise, wenn ein Objekt, welches nicht mehr gebraucht wird, trotzdem noch in einer aktiven Datenstruktur referenziert wird.

In der folgenden Grafik ist ein Überblick über die Entwicklung der Systemressourcen während dem 111-stündigen Stabilitätstest zu sehen. Dabei kann sehr gut erkannt werden, dass sich keine bis höchstens marginale Memory Leaks in der Applikation befinden. Einerseits hat sich bis zum Testende die Anzahl der Handles zwischen 700 und 800 stabilisiert. Andererseits hat der temporäre Speicher 275 MiB nie überschritten und sich bis zum Testende bei etwa 211 MiB eingependelt.

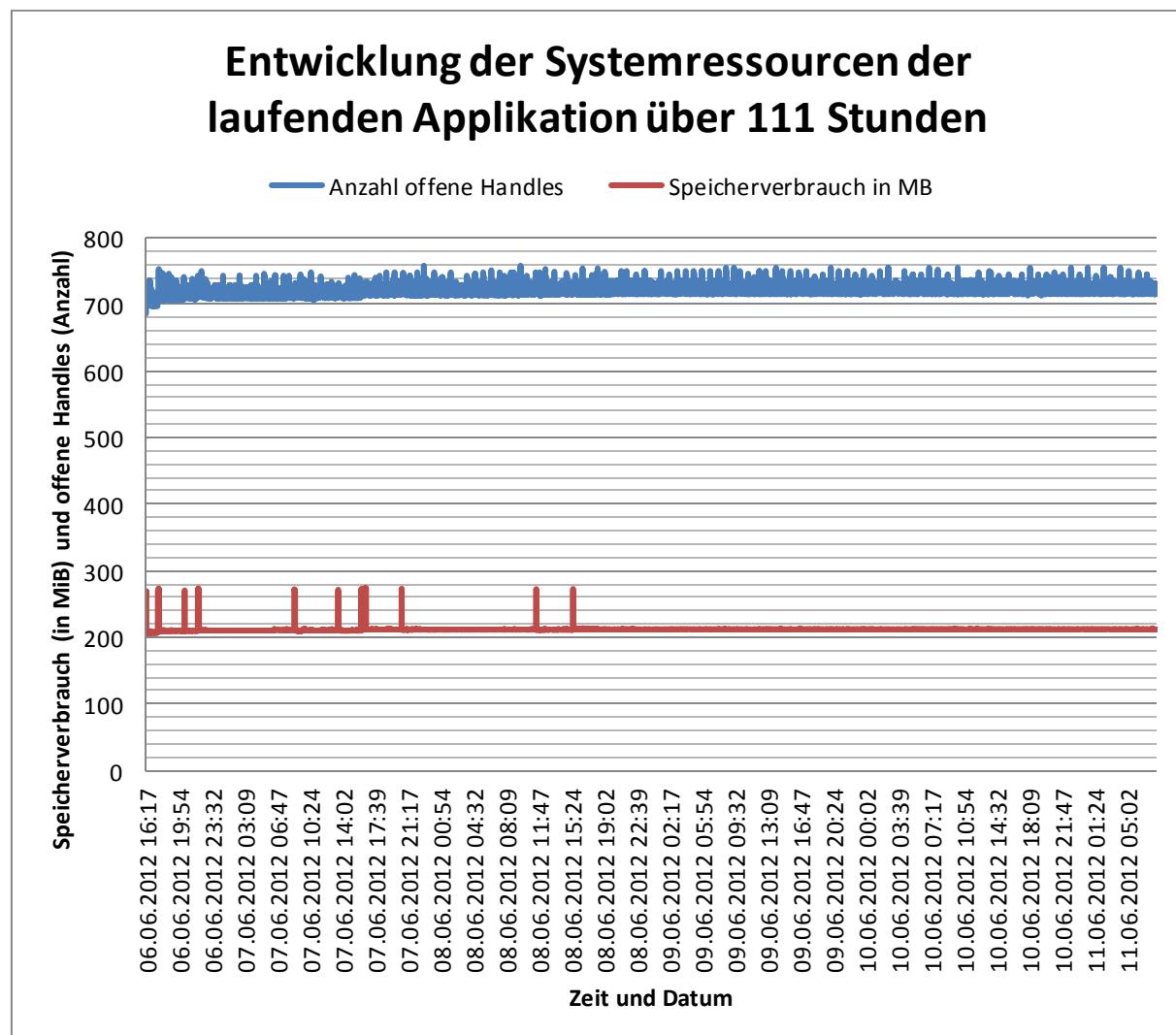


Abbildung 101 - Entwicklung der Systemressourcen der laufenden Applikation über 111 Stunden, MB = MiB

Zu gewissen Zeitpunkten ist der Speicherverbrauch für eine kurze Dauer von ein paar wenigen Sekunden bis zu 275 MiB angewachsen. Dieser Zuwachs stellt aber nicht in Korrelation zur Bedienung der Applikation weshalb

davon ausgegangen wird, dass diese lokalen Maxima durch das .NET Framework von Microsoft verursacht wurden.

In der nachfolgenden Tabelle sind Details zum Stabilitätstest, die in der oberen Grafik nicht erkennbar sind, festgehalten.

Bezeichnung	Speicherverbrauch in MiB	Anzahl offene Handles
Maximum	274.11	758
Minimum	205.57	686
Mittelwert	211.92	722
Median	211.91	722
Stichprobe 07.06.2012 00:00	210.13	713
Stichprobe 08.06.2012 00:00	211.46	721
Stichprobe 09.06.2012 00:00	211.89	715
Stichprobe 10.06.2012 00:00	212.10	750
Stichprobe 11.06.2012 00:00	211.96	728

**Abbildung 102 - Minimum, Maximum, Mittelwert, Median, Stichproben des Stabilitätstests**

Der Stabilitätstest wurde am 06.06.2012 um 16:17 Uhr gestartet und am 11.06.2012 um 08:27 Uhr beendet. Die genauen Daten befinden sich im Anhang (VIII Anhang).

Aus den Daten des Stabilitätstests über 111 Stunden kann geschlossen werden, dass der Prototyp der Videowall problemlos über eine Dauer von 20 oder 24 Stunden betrieben werden kann, ohne dass mit Instabilitäten gerechnet werden muss.

## V.8.6 Dokumentation der Realisierung

### V.8.6.1 Übereinstimmung mit Architektur

Im Kapitel V.6.5 Architektur ist die Schichtenarchitektur aufgezeigt. Wie in den nachfolgenden Abbildungen Abbildung 103 - Dependency Graph, Videowall-Applikation, Abbildung 104 - Dependency Graph, Poster-Applikation und Abbildung 105 - Dependency Graph, Mittagsmenu-Applikation ersichtlich ist, stimmt die umgesetzte mit der geplanten Architektur überein. Die Assemblies wurden analog zu den Schichten aufgebaut.

#### V.8.6.1.1 Videowall-Applikation

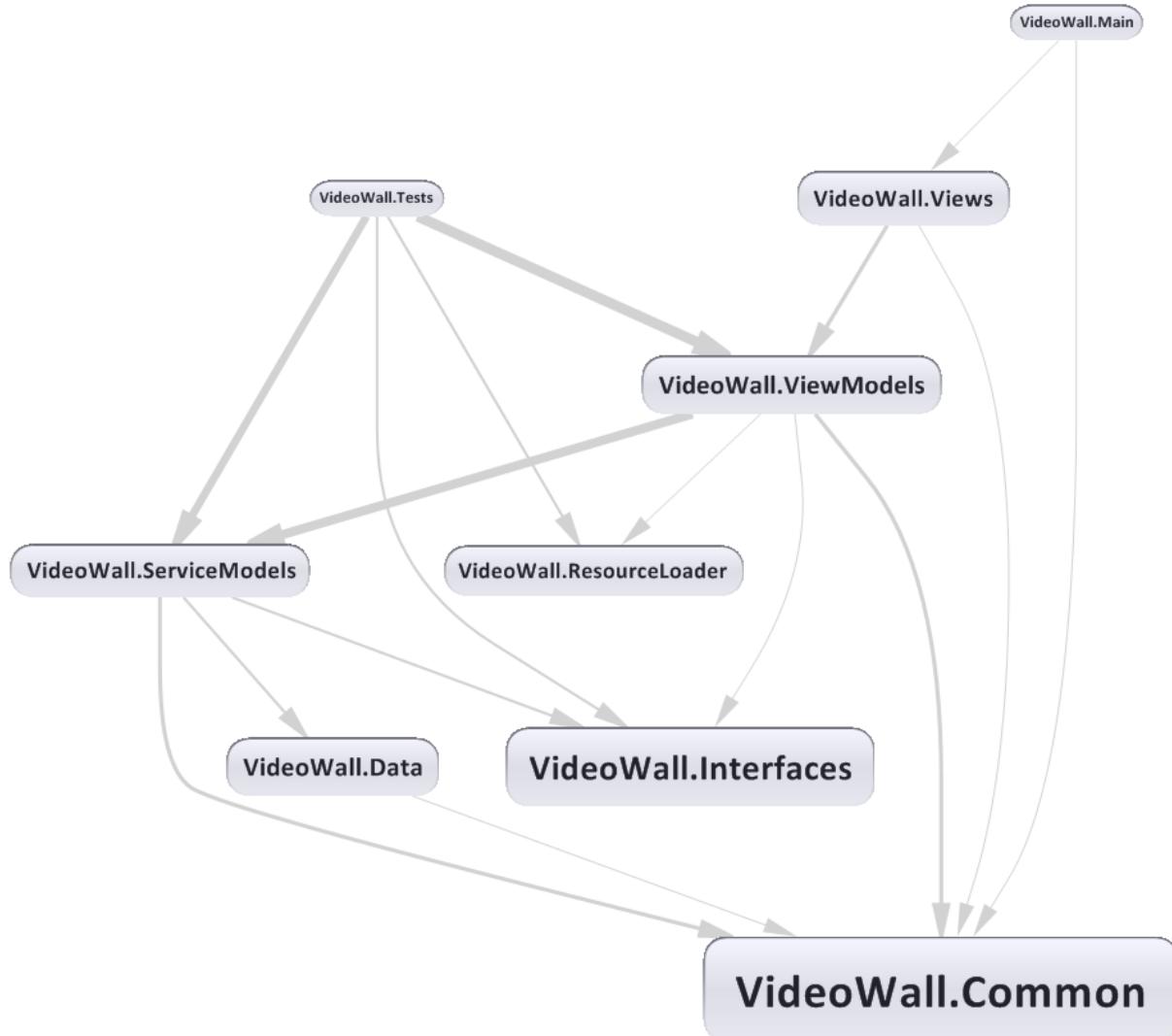


Abbildung 103 - Dependency Graph, Videowall-Applikation

Das Assembly `VideoWall.Main` wird zusätzlich benötigt, um die Applikation zu starten.

#### V.8.6.1.2 Poster-Applikation

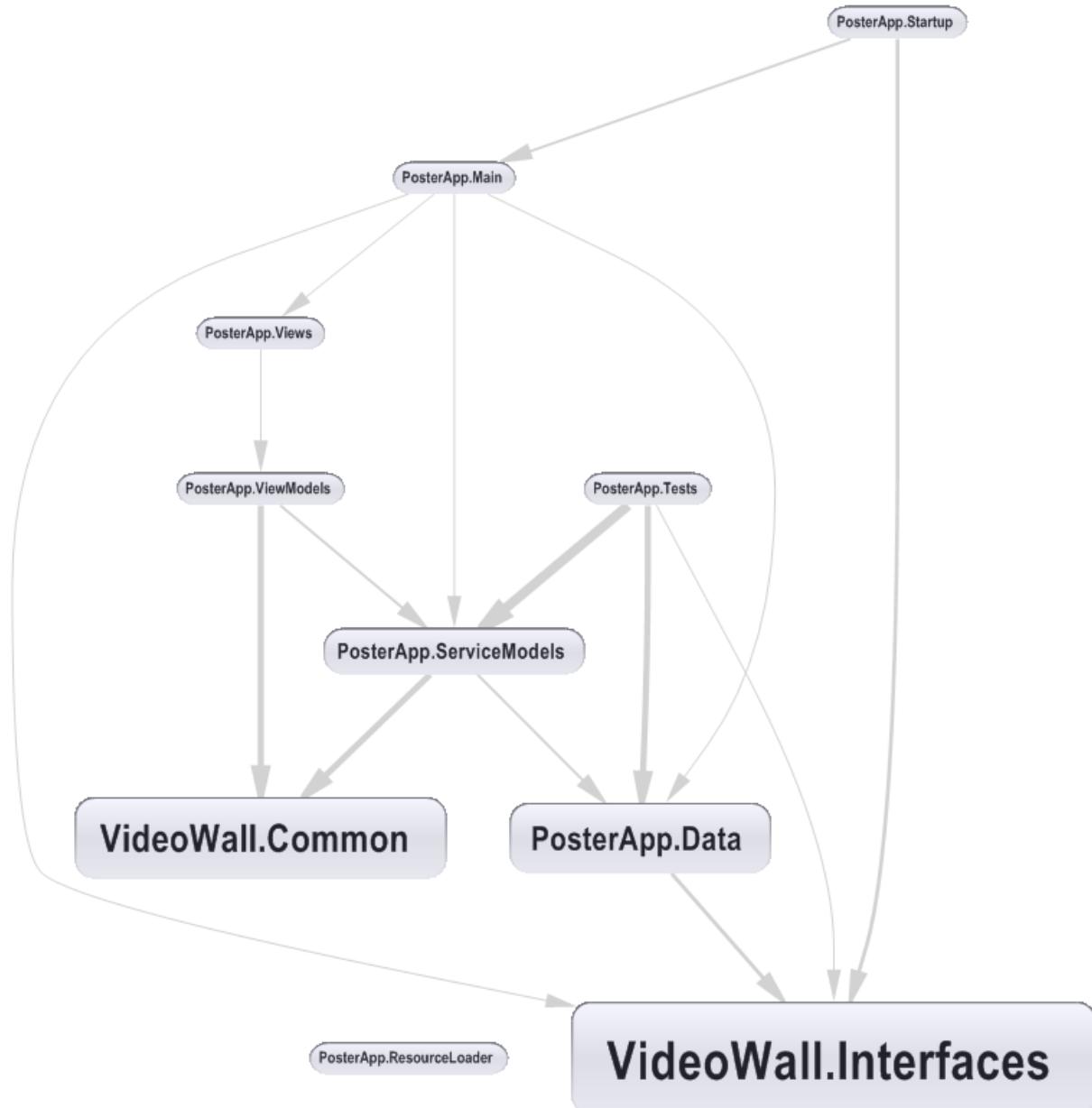


Abbildung 104 - Dependency Graph, Poster-Applikation

Das Assembly `PosterApp.Startup` wird verwendet um die Applikation zu starten. Dank diesem Assembly kann die Poster-Applikation unabhängig von der Videowall-Applikation gestartet und getestet werden. Im Assembly `PosterApp.Main` ist das eigentliche Plug-in für den Export definiert (siehe hierzu V.6.6 Plug-in Framework).

### V.8.6.1.3 Mittagsmenu-Applikation

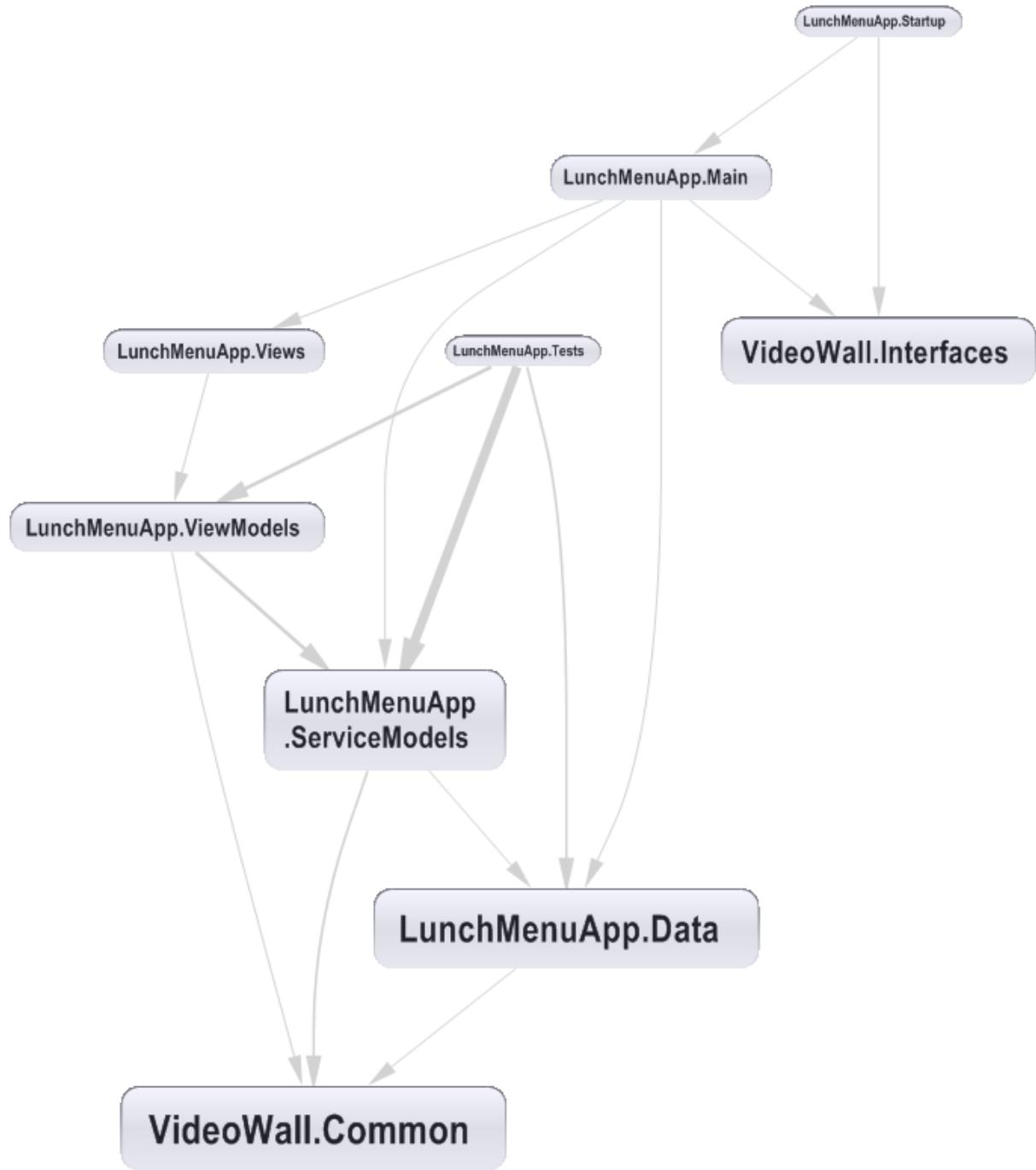


Abbildung 105 - Dependency Graph, Mittagsmenu-Applikation

Das Assembly `LunchMenuApp.Startup` wird verwendet um die Applikation zu starten. Dank diesem Assembly kann die Mittagsmenu-Applikation unabhängig von der Videowall-Applikation gestartet und getestet werden. Im Assembly `LunchMenuApp.Main` ist das eigentliche Plug-in für den Export definiert (siehe hierzu V.6.6 Plug-in Framework).

## V.8.6.2 Code Statistik

### V.8.6.2.1 Testabdeckung

#### V.8.6.2.1.1 Videowall-Applikation

In der Solution VideoWall konnte eine Testabdeckung von 42% erreicht werden.

Von der Abdeckung ausgenommen sind die Nodes VideoWall.Interfaces und VideoWall.Tests.

Wie in Abbildung 106 - Test Coverage VideoWall ersichtlich ist, bestehen grosse Unterschiede in der Testabdeckung in den verschiedenen Projekten.

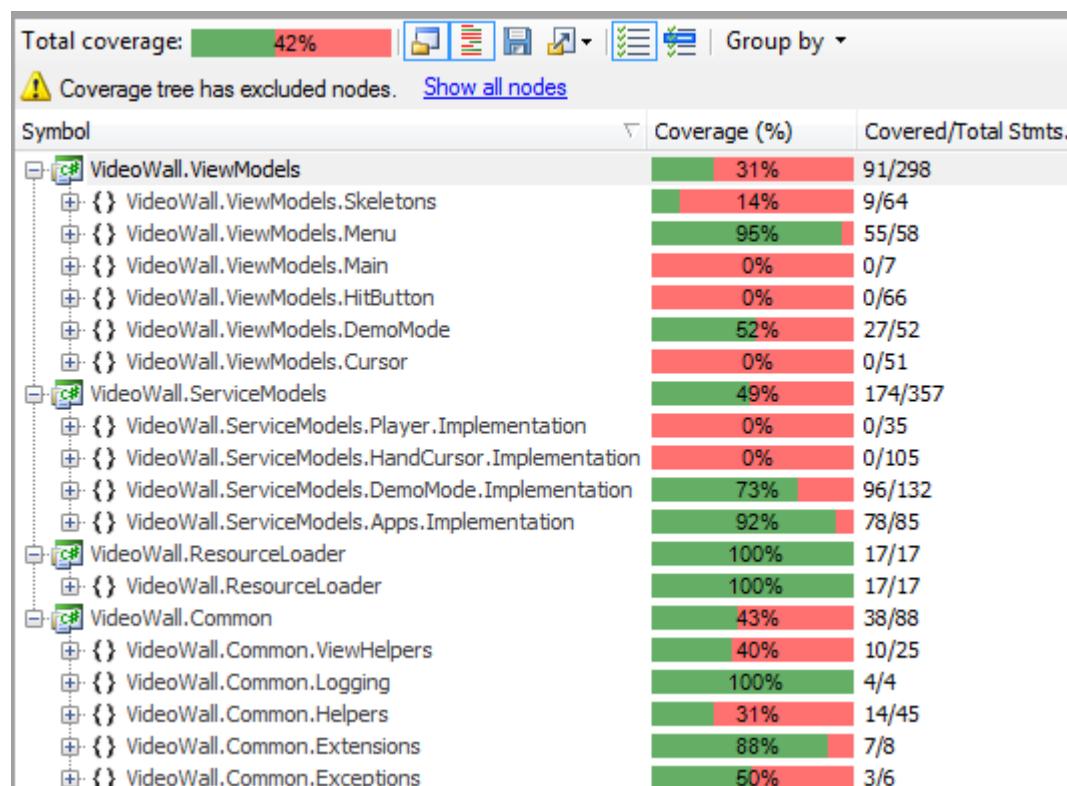


Abbildung 106 - Test Coverage VideoWall

Daher nachfolgend eine Erläuterung zur Testabdeckung:

#### VideoWall.ViewModels

Die Menu-ViewModels, welche die Tabs und deren Inhalte darstellen, sind fast vollständig und der Demomodus ist zu gut 50% mit Tests abgedeckt.

Das MainWindowViewModel lässt sich schwer testen. In der Applikation wird es vom UnityContainer gemanaged.

Einige Klassen wie die aus den Namespaces Skeletons, HitButton und Cursor sind eng mit dem Kinect Skeleton vernetzt und sind schwierig und aufwändig zu testen. Ein weiterer Punkt ist die begrenzte Zeit, die für das Erstellen der Test zur Verfügung stand. Die manuellen Systemtests (siehe V.8.4 Systemtests) dienen der Sicherstellung der Qualität dieser Klassen.

#### VideoWall.ServiceModels

Es wurde Wert darauf gelegt, die Qualität der Klassen (Apps.Implementation), welche die unterschiedlichen Plug-ins managen, und des Demomodus mit einer hohen Abdeckung sicherzustellen.

Die Klassen in den Namespaces Player.Implementation und HandCursor.Implementation arbeiten mit dem Skelett des Kinect Sensors zusammen, was das Testen schwierig und aufwändig macht. Ein weiterer Punkt ist die begrenzte Zeit, die für das Erstellen der Test zur Verfügung steht. Die manuellen Systemtests (siehe V.8.4 Systemtests) dienen der Sicherstellung der Qualität dieser Klassen.

### **VideoWall.ResourceLoader**

Hier beträgt die Abdeckung 100%.

### **VideoWall.Common**

Soweit es in der verbleibenden Zeit für sinnvoll erachtet wurde, wurden für die Klassen im Projekt Common Tests geschrieben.

#### **V.8.6.2.1.2 Poster-Applikation**

In der Solution PosterApp konnte eine Testabdeckung von 100% erreicht werden.

Von der Abdeckung ausgenommen sind die Nodes VideoWall.Interfaces, VideoWall.Common und PosterApp.Tests.

Die Tests für die Klassen aus VideoWall.Common befinden sich, wie die Klassen selbst, im Framework Videowall.

Die in Abbildung 107 - Test Coverage PosterApp in hellgrauer Schrift dargestellten Properties (Path, Image, Posters, NavigateToRightCommand und NavigateToLeftCommand) sind Auto-Properties und werden daher nicht getestet.

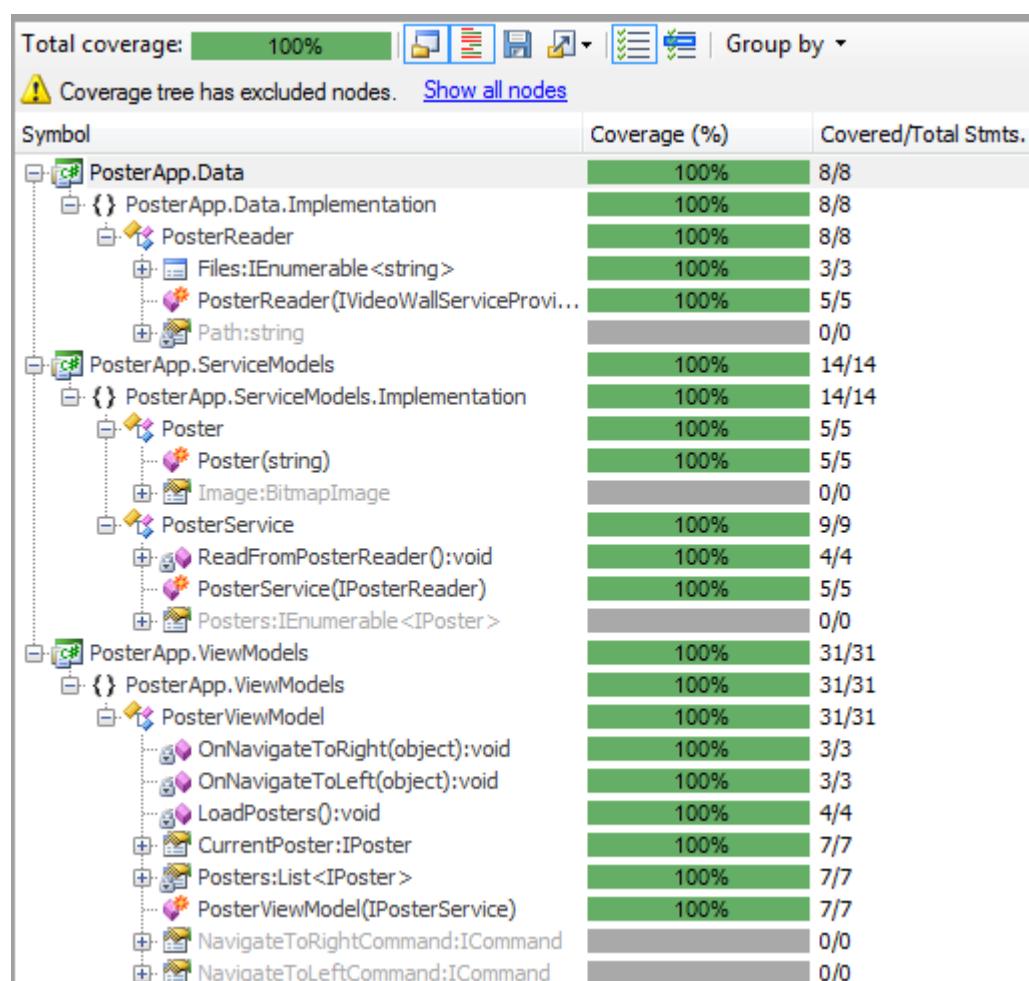


Abbildung 107 - Test Coverage PosterApp

#### **V.8.6.2.1.3 Mittagsmenu-Applikation**

In der Solution LunchMenuApp konnte eine Testabdeckung von 100% erreicht werden.

Von der Abdeckung ausgenommen sind die Nodes VideoWall.Common und LunchMenuApp.Tests.

Die Tests für die Klassen aus VideoWall.Common befinden sich, wie die Klassen selbst, im Framework Videowall.

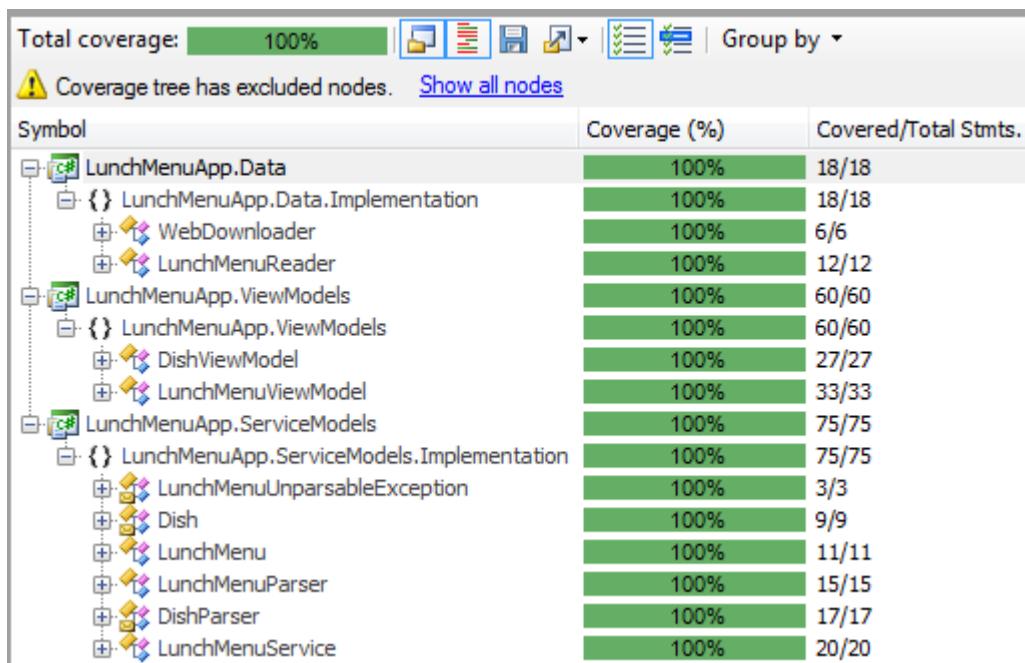


Abbildung 108 - Test Coverage LunchMenuApp

#### V.8.6.2.2 Lines of Code (LOC)

Für die Auswertung der Lines of Code (LOC) wurden die Angaben von Visual Studio (siehe V.8.6.3.1 Visual Studio) übernommen.

Hierarchie	Lines of Code (LOC)
<b>VideoWall</b>	
VideoWall.Common	85
VideoWall.Data	92
VideoWall.Interfaces	4
VideoWall.Main	0
VideoWall.ResourceLoader	9
VideoWall.ServiceModels	381
VideoWall.Tests	179
VideoWall.ViewModels	232
VideoWall.Views	5
<b>Total VideoWall</b>	<b>987</b>
 <b>PosterApp</b>	
PosterApp.Data	6
PosterApp.Main	18
PosterApp.ResourceLoader	0
PosterApp.ServiceModels	12
PosterApp.Startup	9
PosterApp.Tests	63
PosterApp.ViewModels	19
PosterApp.Views	0
<b>Total PosterApp</b>	<b>127</b>

Hierarchie	Lines of Code (LOC)
<b>LunchMenuApp</b>	
LunchMenuApp.Data	13
LunchMenuApp.Main	19
LunchMenuApp.ServiceModels	66
LunchMenuApp.Startup	3
LunchMenuApp.Tests	139
LunchMenuApp.ViewModels	29
LunchMenuApp.Views	0
<b>Total LunchMenuApp</b>	<b>269</b>
<b>Total Gesamt</b>	<b>1383</b>

Tabelle 30 - Lines of Code (LOC)

### V.8.6.3 Code Qualität

Für die Metrikanalyse des Codes wurden das Visual Studio selbst und NDepend (siehe V.4.2 Tools) verwendet. Somit ist eine objektivere Bewertung des Codes möglich.

#### V.8.6.3.1 Visual Studio

Zielsetzung war es, einen „Maintainability Index“ von mindestens 50% zu erreichen, auf Ebene Projekt. Der „Maintainability Index“ setzt sich aus verschiedenen Kriterien zusammen und liegt zwischen 0 und 100. Ein Index zwischen 0 und 9 weist auf schlechte, ein Wert zwischen 10 und 19 auf eine moderate und zwischen 20 und 100 eine gute Wartbarkeit hin. Wie aus den nachfolgenden Abbildungen (Abbildung 109 - Metriken Visual Studio, Videowall-Applikation, Abbildung 110 - Metriken Visual Studio, Poster-Applikation und Abbildung 111 - Metriken Visual Studio, Mittagsmenu-Applikation) entnommen werden kann, ist der Maintainability Index bei jeder Applikation in allen Projekten der Solution über 50, was den nichtfunktionalen Anforderungen entspricht.

##### V.8.6.3.1.1 Videowall Applikation

Hierarchy	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
▷ VideoWall.Common (Debug)	80	45	2	30	85
▷ VideoWall.Data (Debug)	86	52	2	28	92
▷ VideoWall.Interfaces (Debug)	99	11	2	7	4
▷ VideoWall.Main (Debug)	100	0	0	0	0
▷ VideoWall.ResourceLoader (Debug)	89	8	1	5	9
▷ VideoWall.ServiceModels (Debug)	86	229	2	73	381
▷ VideoWall.Tests (Debug)	75	59	1	60	179
▷ VideoWall.ViewModels (Debug)	86	134	2	61	232
▷ VideoWall.Views (Debug)	80	3	1	6	5

Abbildung 109 - Metriken Visual Studio, Videowall-Applikation

### V.8.6.3.1.2 Poster-Applikation

Hierarchy	Maintainability Ind...	Cyclomatic Comple...	Depth of Inheritance	Class Coupling	Lines of Code
► PosterApp.Data (Debug)	95	5	1	5	6
► PosterApp.Main (Debug)	82	9	1	15	18
► PosterApp.ResourceLoader (Debug)	100	0	0	0	0
► PosterApp.ServiceModels (Debug)	95	11	1	10	12
► PosterApp.Startup (Debug)	88	5	1	7	9
► PosterApp.Tests (Debug)	80	24	1	31	63
► PosterApp.ViewModels (Debug)	89	12	2	8	19
► PosterApp.Views (Debug)	100	0	0	0	0
► VideoWall.Common (Debug)	80	45	2	30	85
► VideoWall.Interfaces (Debug)	99	11	2	7	4

Abbildung 110 - Metriken Visual Studio, Poster-Applikation

### V.8.6.3.1.3 Mittagsmenu-Applikation

Hierarchy	Maintainability Index	Cyclomatic Comple...	Depth of Inheritance	Class Coupling	Lines of Code
► LunchMenuApp.Data (Debug)	91	9	1	10	13
► LunchMenuApp.Main (Debug)	82	9	1	19	19
► LunchMenuApp.ServiceModels (Debug)	90	44	2	24	66
► LunchMenuApp.Startup (Debug)	100	2	1	2	3
► LunchMenuApp.Tests (Debug)	82	59	1	35	139
► LunchMenuApp.ViewModels (Debug)	86	16	2	13	29
► LunchMenuApp.Views (Debug)	100	0	0	0	0
► VideoWall.Common (Debug)	80	45	2	30	85
► VideoWall.Interfaces (Debug)	99	11	2	7	4

Abbildung 111 - Metriken Visual Studio, Mittagsmenu-Applikation

### V.8.6.3.2 NDepend

Neben der Analyse mit Visual Studio (siehe V.8.6.3.1 Visual Studio) wurde der Code zusätzlich mit NDepend ausgewertet. Folgende Statistiken wurden direkt von NDepend für die verschiedenen Applikationen generiert:

### V.8.6.3.2.1 Videowall-Applikation

Application Metrics		Note: Further Application Statistics are available.	
# Lines of code : 793	# IL instruction : 7,360	# Exception types : 1	Third Party Usage Percentage ...
# Assemblies : 9	# Lines of comment : 3,302	# Attribute types : 0	... code coverage : N/A
# Namespaces : 33	# Classes : 109	# Delegate types : 0	... of comment : 80%
# Types : 123	# Abstract classes : 1	# Enumeration types : 2	... of public types : 45.53%
# Methods : 529	# Interfaces : 12	# Generic methods : 6	... of public methods : 63.33%
# Fields : 213	# Value types : 0	# Generic types : 2	... of classes with public field(s) : %
# C# source files : 122			

Abbildung 112 - Metriken NDepend, Übersicht, Videowall-Applikation

Assemblies Metrics															
<p>If you wish to define thresholds on assemblies' Code Metrics, consider writing some Rules.          Clicking column header arrows sorts values.          Clicking column header title text redirect to the online Code Metric definition.</p>															
<b>Show 50 ▾ entries</b>															
Assemblies	# lines of code	# IL instruction	# Types	# Abstract Types	# lines of comment	% Comment	% Coverage	Afferent Coupling	Efferent Coupling	Relational Cohesion	Instability	Abstractness	Distance		
VideoWall.Common v1.0.0.0	47	549	18	1	339	87	-	22	39	0.5	0.64	0.06	0.22		
VideoWall.Interfaces v1.0.0.0	0	23	7	5	-	-	-	20	7	0.71	0.26	0.71	0.02		
VideoWall.Data v1.0.0.0	55	555	8	1	310	84	-	1	31	1.25	0.97	0.12	0.07		
VideoWall.ServiceModels v1.0.0.0	209	2396	33	5	1062	83	-	13	62	1.15	0.83	0.15	0.02		
VideoWall.ResourceLoader v1.0.0.0	7	60	1	0	78	91	-	3	10	1	0.77	0	0.16		
VideoWall.ViewModels v1.0.0.0	164	1484	18	1	759	82	-	7	62	0.83	0.9	0.06	0.03		
VideoWall.Tests v1.0.0.0	146	959	20	0	358	71	-	0	62	0.75	1	0	0		
VideoWall.Views v1.0.0.0	125	1077	14	0	268	68	-	1	82	0.57	0.99	0	0.01		
VideoWall.Main v1.0.0.0	40	257	4	0	128	76	-	0	49	0.25	1	0	0		
Assemblies	# lines of code	# IL instruction	# Types	# Abstract Types	# lines of comment	% Comment	% Coverage	Afferent Coupling	Efferent Coupling	Relational Cohesion	Instability	Abstractness	Distance		

Abbildung 113 - Metriken NDepend, Assemblies, Videowall-Applikation

### V.8.6.3.2.2 Poster-Applikation

Application Metrics								Note: Further <a href="#">Application Statistics</a> are available.							
# Lines of code : 182	# IL instruction : 1,613	# Exception types : 1	Third Party Usage	Percentage ...											
# Assemblies : 10	# Lines of comment : 1,321	# Attribute types : 0	# Assemblies used : 10	... code coverage : N/A											
# Namespaces : 21	# Classes : 55	# Delegate types : 0	# Namespaces used : 27	... of comment : 87%											
# Types : 63	# Abstract classes : 1	# Enumeration types : 0	# Types used : 97	... of public types : 53.97%											
# Methods : 157	# Interfaces : 8	# Generic methods : 5	# Methods used : 79	... of public methods : 73.89%											
# Fields : 47	# Value types : 0	# Generic types : 1	# Fields used : 4	... of classes with public field(s) : %											
# C# source files : 62															

Abbildung 114 - Metriken NDepend, Übersicht, Poster-Applikation

Assemblies Metrics															
<p>If you wish to define thresholds on assemblies' Code Metrics, consider writing some Rules.          Clicking column header arrows sorts values.          Clicking column header title text redirect to the online Code Metric definition.</p>															
<b>Show 50 ▾ entries</b>															
Assemblies	# lines of code	# IL instruction	# Types	# Abstract Types	# lines of comment	% Comment	% Coverage	Afferent Coupling	Efferent Coupling	Relational Cohesion	Instability	Abstractness	Distance		
VideoWall.Interfaces v1.0.0.0	0	23	7	5	-	-	-	9	7	0.71	0.44	0.71	0.11		
PosterApp.Data v1.0.0.0	3	39	4	1	105	97	-	5	9	0.5	0.64	0.25	0.08		
VideoWall.Common v1.0.0.0	43	549	18	1	339	88	-	2	37	0.5	0.95	0.06	0		
PosterApp.ServiceModels v1.0.0.0	5	76	6	2	145	96	-	6	12	1	0.67	0.33	0		
PosterApp.ViewModels v1.0.0.0	15	138	2	0	106	87	-	1	16	0.5	0.94	0	0.04		
PosterApp.Views v1.0.0.0	17	94	3	0	90	84	-	1	24	0.33	0.96	0	0.03		
PosterApp.Main v1.0.0.0	11	85	2	0	80	87	-	1	21	0.5	0.95	0	0.03		
PosterApp.Startup v1.0.0.0	22	181	6	0	156	87	-	0	38	0.5	1	0	0		
PosterApp.ResourceLoader v1.0.0.0	9	54	2	0	71	88	-	0	16	0.5	1	0	0		
PosterApp.Tests v1.0.0.0	57	374	13	0	229	80	-	0	33	0.69	1	0	0		
Assemblies	# lines of code	# IL instruction	# Types	# Abstract Types	# lines of comment	% Comment	% Coverage	Afferent Coupling	Efferent Coupling	Relational Cohesion	Instability	Abstractness	Distance		

Abbildung 115 - Metriken NDepend, Assemblies, Poster-Applikation

### V.8.6.3.2.3 Mittagsmenu-Applikation

Application Metrics				Note: Further <a href="#">Application Statistics</a> are available.			
# Lines of code : 271	# IL instruction : 2,383	# Exception types : 2	Third Party Usage	Percentage ...			
# Assemblies : 9	# Lines of comment : 1,697	# Attribute types : 0	# Assemblies used : 12	... code coverage : N/A			
# Namespaces : 19	# Classes : 60	# Delegate types : 0	# Namespaces used : 35	... of comment : 86%			
# Types : 71	# Abstract classes : 1	# Enumeration types : 0	# Types used : 108	... of public types : 57.75%			
# Methods : 204	# Interfaces : 11	# Generic methods : 4	# Methods used : 95	... of public methods : 80.39%			
# Fields : 56	# Value types : 0	# Generic types : 1	# Fields used : 5	... of classes with public field(s) : %			
# C# source files : 77							

Abbildung 116 - Metriken NDepend, Übersicht, Mittagsmenu-Applikation

Assemblies Metrics													
<p>If you wish to define thresholds on assemblies' Code Metrics, consider writing some Rules. Clicking column header arrows sorts values. Clicking column header title text redirect to the online Code Metric definition.</p>													
Show 50 entries	Search:												
Assemblies	# lines of code	# IL instruction	# Types	# Abstract Types	# lines of comment	% Comment	% Coverage	Afferent Coupling	Efferent Coupling	Relational Cohesion	Instability	Abstractness	Distance
VideoWall.Common v1.0.0.0	48	549	18	1	339	87	-	7	39	0.5	0.85	0.06	0.07
LunchMenuApp.Data v1.0.0.0	9	79	6	2	138	93	-	6	13	0.67	0.68	0.33	0.01
LunchMenuApp.ServiceModels v1.0.0.0	39	356	12	4	293	88	-	12	22	1.58	0.65	0.33	0.01
LunchMenuApp.ViewModels v1.0.0.0	31	185	3	0	128	80	-	3	19	0.67	0.86	0	0.1
VideoWall.Interfaces v1.0.0.0	2	23	7	5	191	98	-	3	7	0.71	0.7	0.71	0.29
LunchMenuApp.Views v1.0.0.0	15	79	3	0	86	85	-	1	16	0.33	0.94	0	0.04
LunchMenuApp.Main v1.0.0.0	9	91	2	0	83	90	-	1	25	0.5	0.96	0	0.03
LunchMenuApp.Startup v1.0.0.0	12	166	5	0	117	90	-	0	31	0.4	1	0	0
LunchMenuApp.Tests v1.0.0.0	106	855	15	0	322	75	-	0	37	0.6	1	0	0

Abbildung 117 - Metriken NDepend, Assemblies, Mittagsmenu-Applikation

### V.8.6.3.3 Code Warnungen

Wie aus den nachfolgenden Abbildungen ersichtlich ist, treten beim Kompilieren des Codes der verschiedenen Applikationen keine Fehler oder Warnungen auf.

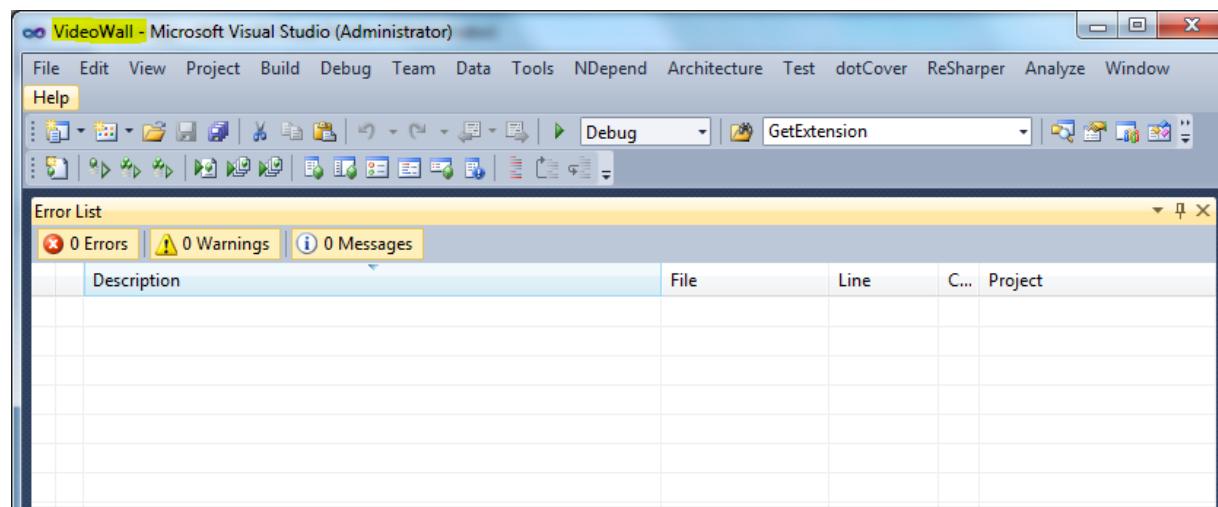


Abbildung 118 - Warnings, Videowall-Applikation

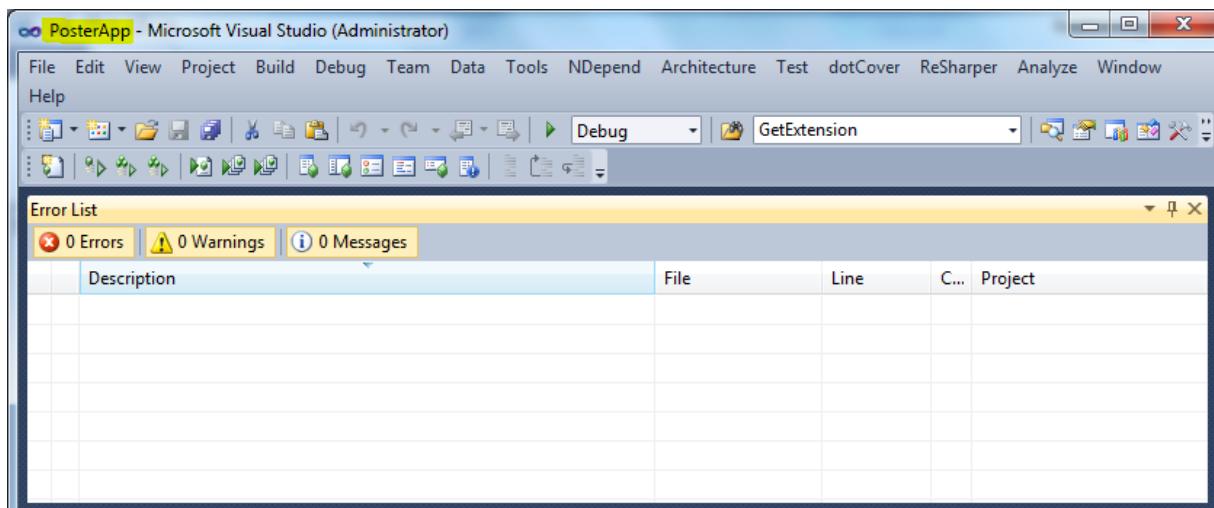


Abbildung 119 - Warnings, Poster-Applikation

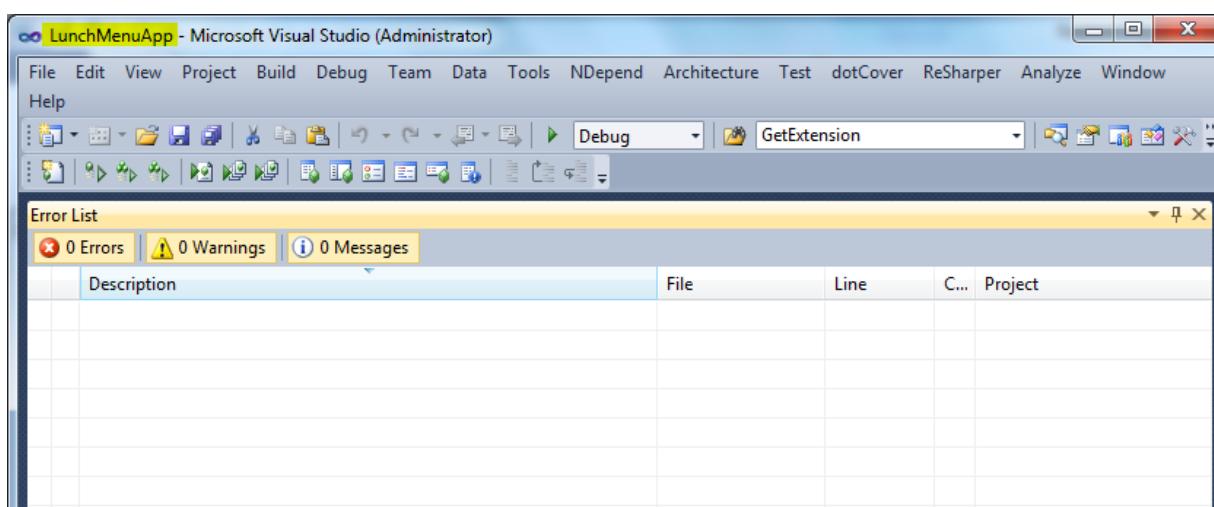


Abbildung 120 - Warnings, Mittagsmenu-Applikation

## V.8.6.4 Coding Standards

Die Coding Standards, welche für dieses Projekt gültig sind, wurden mehrheitlich von ReSharper übernommen.

Das exportierte ReSharper-Profil kann über den Pfad „code/reSharper\_settings/reSharper.DotSettings“ gefunden werden. Nachfolgend wurden die wichtigsten Einstellungen dokumentiert.

### V.8.6.4.1 C# Namenskonventionen

Folgende Namenskonventionen wurden verwendet:

Entity kinds	Preview
Types and namespaces	UpperCamelCase
Interfaces	IUpperCamelCase
Type parameters	TUpperCamelCase
Methods, properties and events	UpperCamelCase
Local variables	lowerCamelCase
Local constants	lowerCamelCase
Parameters	lowerCamelCase
Fields (not private)	UpperCamelCase
Instance fields (private)	_lowerCamelCase
Static field (private)	_lowerCamelCase
Constant fields (not private)	UpperCamelCase
Constant fields (private)	UpperCamelCase
Static readonly fields (not private)	UpperCamelCase
Static readonly fields (private)	UpperCamelCase
Enum members	UpperCamelCase
All other entities	UpperCamelCase

Abbildung 121 - Naming Style

### V.8.6.4.2 Formatierungsstil

#### Braces Layout

Geschweifte Klammern befinden sich auf einer neuen Zeile.

```
namespace N
{
    internal interface I
    {
        void foo();
    }

    internal class C
    {
    }
}
```

Abbildung 122 - Formatierungsstil, Braces Layout

#### Line Breaks and Wrapping

Lange Zeilen (>120 Zeichen) werden umgebrochen.

```
Wrap long lines
When enabled:
string output = string.Format(CultureInfo.InvariantCulture,
    "{0:yyyy-MM-dd} {1}", date, message);
```

Abbildung 123 - Formatierungsstil, Line Breaks and Wrapping

#### V.8.6.4.3 Auswertung durch ReSharper

Durch den ReSharper können Coding Issues angezeigt werden. Diese werden für die drei Projekte nachfolgend aufgezeigt und erklärt.

##### Videowall-Applikation

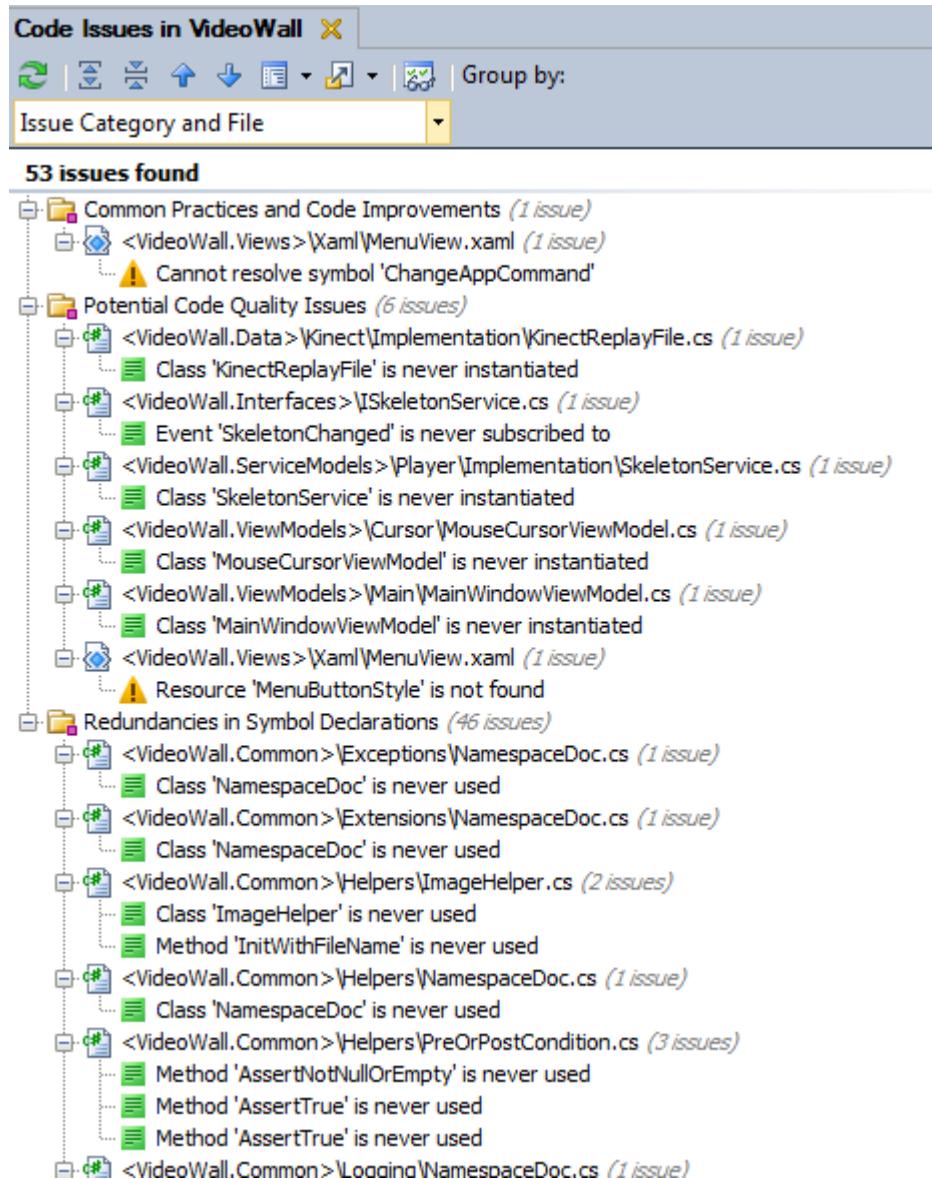


Abbildung 124 - Coding Issues, Videowall-Applikation

Die Issues der Videowall-Applikation lassen sich wie folgt erläutern:

- Die Issues, welche im .xaml-Code entstanden sind, können nicht behoben werden.
- Die Assemblies Videowall.Common und Videowall.Interfaces werden zusätzlich in die Mittagsmenu-Applikation und die Poster-Applikation eingebunden. Dort wird beispielsweise der ImageHelper verwendet. Aus Sicht des Videowall-Projekts werden diese aber nie genutzt und daher als Issue angezeigt.
- Die Klassen werden zur Laufzeit durch Unity instanziert. Dadurch ist für ReSharper nicht ersichtlich, dass diese Klassen verwendet werden.
- Die Klassen mit dem Namen NamespaceDoc werden nicht verwendet, sie sind aber für die Dokumentation der Namespaces nötig.
- Die Klasse PreOrPostCondition wurde durch das Institut für Software (IFS) vorgegeben und wurde daher übernommen, auch wenn zurzeit noch nicht alle Methoden davon verwendet werden.

## Poster-Applikation

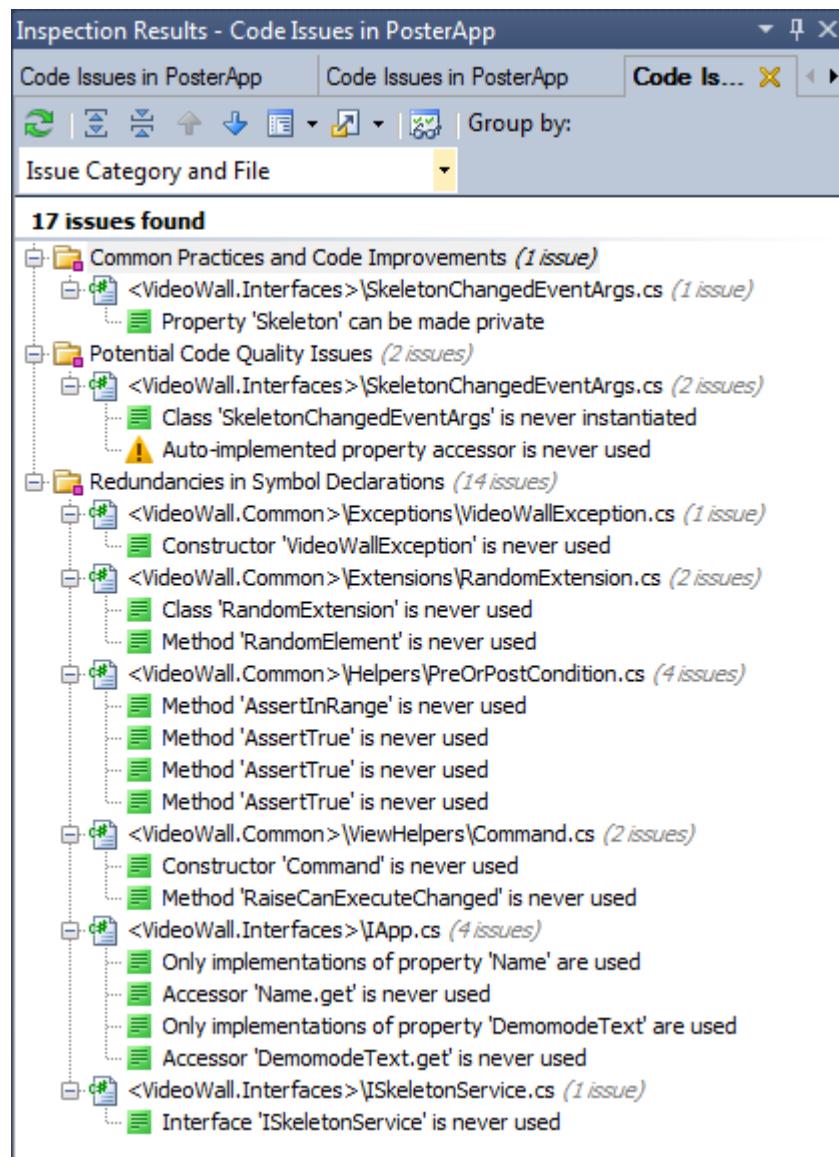


Abbildung 125 - Coding Issues, Poster-Applikation

Die Issues der Poster-Applikation lassen sich wie folgt erläutern:

- Die Assemblies Videowall.Common und Videowall.Interfaces werden von der Poster-Applikation, wie auch von anderen Projekten, verwendet. In der Poster-Applikation selbst werden nicht alle Klassen oder Methoden aus den Assemblies genutzt. Diese kommen aber in den anderen Projekten zum Einsatz. Dies wird aber von ReSharper nicht erkannt.

## Mittagsmenu-Applikation

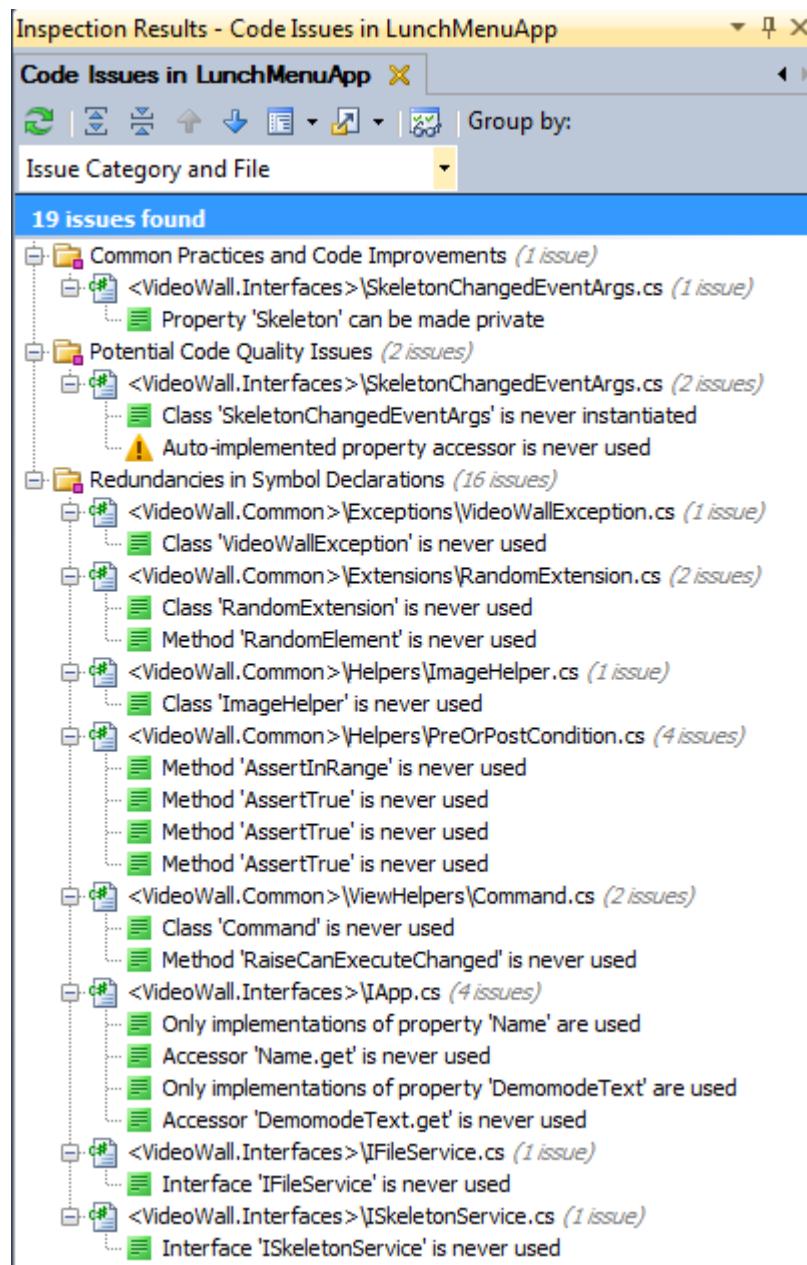


Abbildung 126 - Coding Issues, Mittagsmenu-Applikation

Die Issues der Mittagsmenu-Applikation lassen sich wie folgt erläutern:

- Die Assemblies Videowall.Common und Videowall.Interfaces werden von der Mittagsmenu-Applikation, wie auch von anderen Projekten, verwendet. In der Mittagsmenu-Applikation selbst werden nicht alle Klassen oder Methoden aus den Assemblies genutzt. Diese kommen aber in den anderen Projekten zum Einsatz. Dies wird aber von ReSharper nicht erkannt.

#### V.8.6.4.4 Cleanup

Für das Cleanup des Codes wurden folgende Einstellungen vorgenommen:

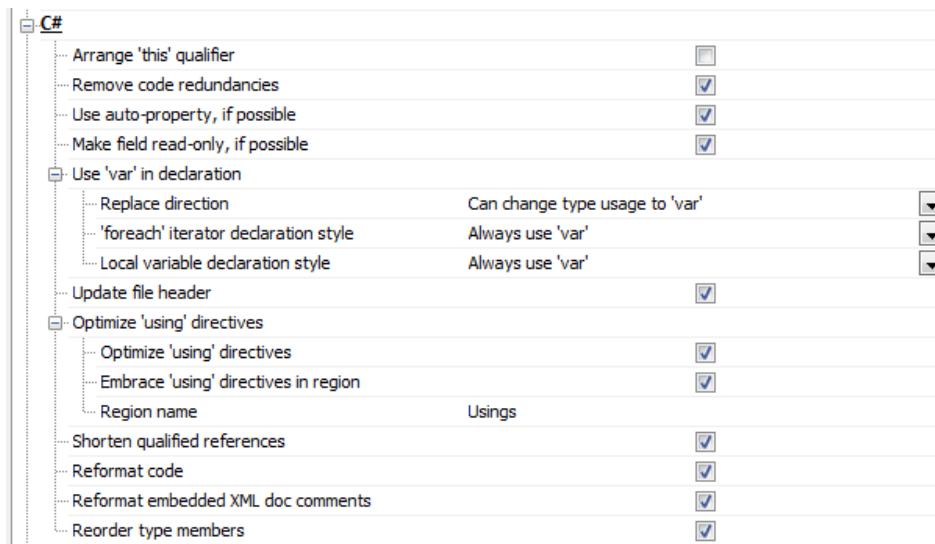


Abbildung 127 - Cleanup Einstellungen

#### V.8.6.5 Dokumentation Quellcode

Zur Dokumentation des Quellcodes wurde GhostDoc verwendet. Damit war die Dokumentation einiges einfacher, da sinnvolle Standardkommentare generiert werden, die bei Bedarf erweitert werden können. Zudem ist es möglich, eine Dokumentation der XML-Kommentare zu generieren (für mehr Informationen siehe Unterkapitel V.8 Generierung der Dokumentation).

Es wurden alle Interfaces sowie Methoden oder Properties, welche mit Hilfe eines Kommentars besser verstanden werden können, kommentiert. Ausgenommen davon sind die XAML-Dateien. Durch die Kommentare sind der Programmcode und besonders komplexere Methoden für Entwickler leichter verständlich.

#### V.8.6.5.1 Generierung der Dokumentation

Zur Generierung einer Dokumentation des Quellcodes wird die Software Sandcastle Help File Builder (SHFB)<sup>46</sup> benötigt.

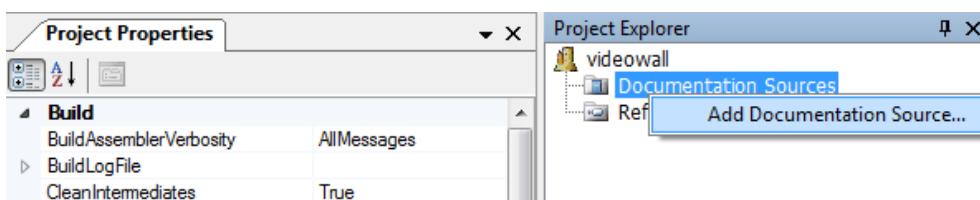


Abbildung 128 - Sandcastle Help File Builder, Hinzufügen der Visual Studio Solution

Zur Generierung der Dokumentation wird folgendermassen vorgegangen:

- Zuerst muss ein neues SHFB-Project erstellt werden (File-> New Project).
- Im Project Explorer wird über den Befehl Add Documentation Source die Visual Studio Solution des Projektes, dessen Code dokumentiert werden soll, ausgewählt.
- In den Project Properties können der Titel („HelpTitle“), der Dokumentename („HtmlHelpName“) und der Speicherort („OutputPath“) des Help Files spezifiziert werden.
- Danach kann die Dokumentation mittels des Build-Befehls erstellt werden (Documentation -> Build Project).

<sup>46</sup> <http://shfb.codeplex.com/>

---

## V.8.7 Beschreibung der Applikationen

---

### V.8.7.1 Hauptapplikation

Die Videowall Applikation stellt die Hauptapplikation dar. Sie sorgt dafür, dass die verschiedenen Plug-ins geladen (diese Funktion wird im Kapitel V.6.6 Plug-in Framework) und visualisiert werden. Auch das Handtracking (siehe V.6.8 Interaktion durch Handtracking) ist darin umgesetzt. Zudem stellt sie die verschiedenen Interfaces zu Verfügungen, welche von den Plug-ins implementiert werden.

---

### V.8.7.2 Plug-ins

Durch das Team wurden zwei Inhalte für die Videowall definiert (Poster und Mittagsmenu).

Zusätzlich wurden drei weitere Plug-Ins entwickelt, die aber nur als Experimente dienen. Sie wurden keinen Code Reviews unterzogen und sie erfüllen weder die für die Videowall definierte Code Qualität oder Code Richtlinien. Die Experimente dienen lediglich dazu, Möglichkeiten von Plug-Ins aufzuzeigen.

Die Plug-ins werden nachfolgend aufgezeigt.

---

#### V.8.7.2.1 Poster Plug-in

Das Poster Plug-in zeigt die Bachelorposter der verschiedenen Studiengänge. Es wird jeweils ein Poster in Grossansicht angezeigt und über die Pfeile kann zum nächsten oder vorhergehenden Poster navigiert werden.

---

#### V.8.7.2.2 Mittagsmenu Plug-in

Das Mittagsmenu Plug-in zeigt das aktuelle Mensa-Mittagsmenu der SV Group der HSR an.

---

#### V.8.7.2.3 Experiment Browser Plug-in

Das Browser Plug-in zeigt, wie Inhalte direkt aus dem Internet auf der Videowall angezeigt werden können.

---

#### V.8.7.2.4 Experiment Diagnostic Plug-in

Das Diagnostic Plug-in ist ein Beispiel, wie ein minimales Plug-in implementiert werden kann und die darin verwendeten Standardbuttons mit der Videowall-Applikation anklickbar sind.

---

#### V.8.7.2.5 Experiment Prezi Plug-in

Das Prezi Plug-in zeigt, dass eine Prezi Präsentation auf der Videowall abgespielt werden kann.

---

#### V.8.7.2.6 Experiment ShapeGame Plug-in

Das ShapeGame Plug-in zeigt, dass es ohne viel Aufwand möglich ist, bestehend Kinect Applikationen in ein Plug-in umzuschreiben, damit diese dann in der Hauptapplikation laufen gelassen werden kann.

---

### V.8.7.3 Mini-Applikationen

Im Verlauf des Projektes wurden diverse kleine Anwendungen erstellt, welche Prototypen von essentiellen Teilen der ganzen Applikation sind. Diese befinden sich im Versionsverwaltungssystem unter code\miniapps\trunk und werden nachfolgend kurz erläutert.

Wie bei den Plug-In Experimenten befinden sich diese Applikationen in einem experimentellen Stadium. Und so sind auch diese Applikationen nicht ins Code Review miteinbezogen worden und erfüllen die Code Qualität und Code Richtlinien nicht und dienen bloss zur Nachvollziehbarkeit und die Weiterentwicklung der Hauptapplikation.

---

### *V.8.7.3.1 DemoMode*

Der Demomodus wurde zu Beginn ohne die Daten des Kinect Skeletts erstellt, die Skeletterkennung wird in dieser Mini-Applikation durch das Drücken eines Buttons simuliert. Sobald die Logik des Demomodus korrekt umgesetzt war, wurde dieser ins Hauptprojekt integriert.

---

### *V.8.7.3.2 DesignMenu*

Die DesignMenu-Applikation wurde benutzt, um direkt in Expression Blend verschiedene Designs für die Tabs des Menüs zu erstellen und untereinander zu vergleichen.

---

### *V.8.7.3.3 DesignPosterNavigationButtons*

Um verschiedene Varianten der Navigationsschaltflächen für die Poster-Applikation zu prüfen, wurde ein separates Projekt erstellt. Die verschiedene Designs wurden in Expression Blend erstellt und verglichen.

---

### *V.8.7.3.4 HandCursorDemoApp*

Für den Wizard of Oz Test musste in der Applikation anstelle des Maus-Cursors eine Hand dargestellt werden, mit welcher die Bewegungen des Skeletts der Testperson nachgeahmt wurden.

---

### *V.8.7.3.5 KinectHandTracker*

Beim KinectHandTracker handelt es sich um ein Projekt, in welchem das Handtracking erstmals implementiert und erprobt wurde, bevor es in die Hauptapplikation eingebunden wurde.

---

### *V.8.7.3.6 KinectRecorder*

In der KinectRecorder-Applikation lassen sich Skelettbewegungen aufzeichnen und wieder abspielen. Das wiederabspielbare File kann in der Hauptapplikation dazu verwendet werden, ein Skelett und dessen Bewegungen zu simulieren auch wenn kein Kinect Sensor am Computer angeschlossen ist.

---

### *V.8.7.3.7 KinectWpfViewers*

Dies sind vordefinierte Klassen des Kinect SDKs. Sie wurden als Beispiel und als Hilfe, beispielsweise beim Einbinden des Skeletts in die View, verwendet.

---

### *V.8.7.3.8 ObjectTrackingVisualizer*

Das Projekt ist eine Testapplikation, die zeigt, wann Personen erkannt werden und wann ihrem Skelett eine neue Identität zugewiesen wird. Der Wechsel der ID geschieht beispielsweise dann, wenn die Person den Erkennungsbereich verlässt und ihn danach erneut betritt.

---

### *V.8.7.3.9 PdfConverter*

Der PDF Converter ist dazu da, um PDF Dateien in Bilder zu konvertieren. Dies ist eine einfache Applikation, die bis jetzt primär für die Entwickler entwickelt wurde. In Zukunft kann die Funktionalität dieses Programmes in das Content Management System eingebaut werden, damit die PDFs automatisch umgewandelt werden können.

---

### *V.8.7.3.10 PluginDemo*

PluginDemo ist ein Beispielprojekt für ein mit MEF erstelltes Plugin und zeigt, wie dieses in die Hauptapplikation eingebunden werden kann.

---

### *V.8.7.3.11 TestXna*

Ein minimalistisches „Game“, das mit XNA implementiert ist und über DirectX läuft. Diese wurde dazu benötigt, um DirectX auf dem Testsetup zu testen.

---

#### *V.8.7.3.12 VideoWithWPF*

Dies ist eine Applikation, welche ein Video abspielt. Sie wurde benötigt, um feststellen zu können, wie flüssig Videos in einer WPF-Applikation auf der Videowall laufen.

---

#### *V.8.7.3.13 WizardOfOzTest*

Die WizardOfOzTest-Applikation wurde für den Wizard of Oz Test zu Beginn der Implementation (V.8.2.1 Test 1: Wizard of Oz) verwendet.

---

#### *V.8.7.3.14 KinectCursorSmoothing*

Ein Ansatz für das Cursor Smoothing mit Kinect.

## V.8.8 Code Reviews

Um die Qualität des Codes und der Architektur sicherzustellen ist es nötig, Code Reviews durchzuführen. Diese werden einerseits im Team periodisch und kollegial durchgeführt. Andererseits werden informelle und formelle Code Reviews mit Michael Gfeller und Silvan Gehrig gemacht.

Bei den Code Reviews sind jeweils drei Komponenten zu beurteilen:

- Die Videowall Core Applikation (Framework, Handtracking, Demomodus)
- Das Mittagsmenu Plug-in
- Das Poster Plug-in

Alle anderen Programmierungen werden bei den Code Reviews nicht berücksichtigt, da es sich bei den anderen Teilen mehr um Experimente und um den Beweis von Konzepten handelt. Diese Applikationen müssten sowieso noch ausgebessert und besser strukturiert werden, bevor sie tatsächlich eingesetzt werden könnten. Es lohnt sich daher nicht, Code Reviews für diesen Code durchzuführen bzw. diesen Code überhaupt zu beurteilen.

### V.8.8.1 Übersicht

Ziel dieses Dokuments ist die Vorbereitung und Dokumentation von Code Reviews. Zu Beginn werden die für das Code Review festgelegten Kriterien aufgezeigt, danach folgen die einzeln dokumentierten Code Reviews mit den jeweiligen Kommentaren.

### V.8.8.2 Kriterien

Die Kriterienliste wurde am 19.04.12 durch das Team erstellt.

#### V.8.8.2.1 Code Style Analyse

	Erfüllt	N. Erfüllt
Die Code Richtlinien von HSR Videowall wurden eingehalten		
Der Code wirkt durch seine Anordnung & Verschachtelung übersichtlich		
Die HSR Videowall Headers sind in allen nicht generierten Sourcen vorhanden		
Die XML-Kommentar Kompilation wurde in den Projekten aktiviert		
Alle Public / Protected Members sind ausreichend Dokumentiert		
Die auskommentierten Programm-Stücke sind ausreichend erklärt		
Die Projekte enthalten keine toten Programm-Klassen		
Die fehlenden Programmstücke sind mittels TODO-Kommentar beschrieben		
Der Code übersetzt ohne Compiler Warnings, die nicht dokumentiert sind und sich nicht in den Test Projekten befinden		
Es gibt keine Bad Smells im Code		

#### V.8.8.2.2 Exception Handling

	Erfüllt	N. Erfüllt
Der Code enthält keine abgefangenen und ignorierten Ausnahmen		
Fehler in asynchronen Prozessen werden mittels Event weitergeleitet		
Das Logging erfasst alle Fehler aus allen Funktionalitätsschichten		
Das Before/After Pattern wird, wo möglich, mittels using(){} angewendet		
IDisposable.Dispose() Methoden werden in jedem Fall aufgerufen		

#### V.8.8.2.3 Flow Control

	Erfüllt	N. Erfüllt
Es existieren keine Schleifen ohne Abbruchkriterien		
Es existieren keine nicht dokumentierte, tote Programmstücke (z.B. if(false) / while		

(false) / ...)

Rekursive Calls haben immer eine Verankerung und Abbruchbedingung

#### V.8.8.2.4 Naming

	Erfüllt	N. Erfüllt
Die Namen der Klassen / Variablen sind selbstbeschreibend		
Verwirrende oder falsche Namen sind nicht vorhanden		
Interface-Klassen beginnen immer mit I (z.B. IDisposable)		
Klassen / Properties / Methoden werden mit PascalCasing geschrieben		
Lokale Variablen / Argumente werden mit camelCasing geschrieben		
Konstanten werden in PascalCasing geschrieben		
Felder in Klassen werden mit _camelCasing geschrieben		
Der Code enthält keine Magic Numbers		

#### V.8.8.2.5 Tools

	Erfüllt	N. Erfüllt
Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben		
Die Errors von FxCop werden wo sinnvoll behoben		

### V.8.8.3 Code Review vom 19.04.2012

Das erste Code Review wurde am 19.04.2012 durchgeführt. Anwesend waren Lukas Elmer, Christina Heidt und Delia Treichler. Es wurde zuerst im Speziellen eine Klasse untersucht, die für die Umrechnung der Maus Position zwischen der Kinect Hand Position und dem Fenster zuständig ist. Danach wurden die Kriterien bewertet.

Das Review zieht einige Verbesserungen nach sich, die sich aus den negativ bewerteten Kriterien ergeben. Im Speziellen sind das:

Beschreibung	Bereinigung geprüft am	Kürzel
XML Dokumentation im Code, Headers	05.06.2012	LE
Genaue Untersuchung, ob die IDisposable.Dispose() Methoden immer aufgerufen werden	05.06.2012	LE
Es wurden „Magic Numbers“ gefunden, die dokumentiert und werden müssen ausgelagert (in statisches Attribut oder in Konfiguration).	05.06.2012	LE

Tabelle 31 - Annotationen und Kommentare Code Review 19.04.2012

#### V.8.8.3.1 Bewertung der Kriterien

##### V.8.8.3.1.1 Code Style Analyse

	Erfüllt	N. Erfüllt
Die Code Richtlinien von HSR Videowall wurden eingehalten	X	
Der Code wirkt durch seine Anordnung & Verschachtelung übersichtlich	X	
Die HSR Videowall Headers sind in allen nicht generierten Sourcen vorhanden		X
Die XML-Kommentar Kompilation wurde in den Projekten aktiviert		X
Alle Public / Protected Members sind ausreichend Dokumentiert		X
Die auskommentierten Programm-Stücke sind ausreichend erklärt	X	
Die Projekte enthalten keine toten Programm-Klassen	X	
Die fehlenden Programmstücke sind mittels TODO-Kommentar beschrieben	X	
Der Code übersetzt ohne Compiler Warnings, die nicht dokumentiert sind und sich nicht in den Test Projekten befinden	X	
Es gibt keine Bad Smells im Code	X	

### V.8.8.3.1.2 Exception Handling

	Erfüllt	N. Erfüllt
Der Code enthält keine abgefangenen und ignorierten Ausnahmen	?	
Fehler in asynchronen Prozessen werden mittels Event weitergeleitet	?	
Das Logging erfasst alle Fehler aus allen Funktionalitätsschichten	?	
Das Before/After Pattern wird, wo möglich, mittels using(){} angewendet	X	
IDisposable.Dispose() Methoden werden in jedem Fall aufgerufen		?

### V.8.8.3.1.3 Flow Control

	Erfüllt	N. Erfüllt
Es existieren keine Schleifen ohne Abbruchkriterien	X	
Es existieren keine nicht dokumentierte, tote Programmstücke (z.B. if(false) / while (false) / ...)	X	
Rekursive Calls haben immer eine Verankerung und Abbruchbedingung	X	

### V.8.8.3.1.4 Naming

	Erfüllt	N. Erfüllt
Die Namen der Klassen / Variablen sind selbstbeschreibend	X	
Verwirrende oder falsche Namen sind nicht vorhanden	X	
Interface-Klassen beginnen immer mit I (z.B. IDisposable)	X	
Klassen / Properties / Methoden werden mit PascalCasing geschrieben	X	
Lokale Variablen / Argumente werden mit camelCasing geschrieben	X	
Konstanten werden in PascalCasing geschrieben	X	
Felder in Klassen werden mit _camelCasing geschrieben	X	
Der Code enthält keine Magic Numbers		X

### V.8.8.3.1.5 Tools

	Erfüllt	N. Erfüllt
Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben	X	
Die Errors von FxCop werden wo sinnvoll behoben	X	

## V.8.8.4 Code Review vom 03.05.2012

Das zweite Code Review wurde am 03.05.2012 durchgeführt. Silvan Gehrig und Michael Gfeller vom IFS gingen zusammen den Code durch und machten Notizen. Bei deren Besprechung waren auch Lukas Elmer, Christina Heidt und Delia Treichler anwesend.

Die Annotationen und Kommentare zum Code sind in der Tabelle 32 - Annotationen und Kommentare Code Review 03.05.2012 festgehalten.

Beschreibung	Bereinigung geprüft am	Kürzel
FileNotFoundException bei Startup, separates Resourcen-Projekt eliminiert Pfade (AutoPlayFileSkeletonReader.cs und LunchMenuReader.cs)	05.06.2012	LE
Organisation der Namespaces, Converters gehören in View (BoolToVisibilityConverter.cs)	05.06.2012	LE
Bild-Anzeige, mit RenderOptions.BitmapScalingMode="HighQuality" wird Image wesentlich besser dargestellt.	05.06.2012	LE
Code ist an einigen Stellen noch nicht kommentiert (z.B. ExtendedVisualTreeHelper.cs)	05.06.2012	LE
Internal Klasse mit public Methoden in ExtendedVisualTreeHelper.cs	-	-
Utils in eigenes Package evtl. Common GUI-Library (ExtendedVisualTreeHelper.cs)	-	-
Verletzung des Information Experts in ImageExtension.cs.	05.06.2012	LE
Initialisierungsmethode LunchMenu.CreateFrom (string fileName) nutzen		
UI Elemente im ViewModel sind unschön und zerstört die Testbarkeit ->	-	-

Beschreibung	Bereinigung geprüft am	Kürzel
Verschiebung in View (z.B. HitStateArgs.cs)		
Console.WriteLine() in Code ist sehr unschön, Logger verwenden (z.B. HitTestHelper.cs)	05.06.2012	LE
OutOfMemoryProblem Exception, geladene Posters benötigen extrem viel Memory	Nicht reproduzierbar	
Filestream wird nicht geschlossen (ImageExtension.cs)	03.05.2012	LE
Kein ServiceModel vorhanden (für Domain Objekte)	05.06.2012	LE
Dispose() Pattern nicht überall vollständig implementiert (z.B. KinectSkeletonReader.cs)	05.06.2012, LE	
Anstelle Loop HitTest-Methode verwenden (Methode OnModelChanged in HitTestHelper)	-	-
ViewModel-Verschachtelung: MenuViewModel weiss mehr/mächtiger als MainWindowViewModel	05.06.2012, LE	
PropertyChanged wirklich nur für Änderung am Property verwenden. (z.B. in LunchMenuService.cs: auf LunchMenuReader nicht PropertyChanged aufrufen)	05.06.2012	LE
DispatcherTimer bietet Funktionen, welche Thread based Timer macht (HitTestHelper.cs)	05.06.2012	LE
Assert in Methode RaiseEventOfUIElement in MainWindow.xaml.cs prüft Funktionalität des Frameworks	05.06.2012	LE
Regionen erstellen	05.06.2012	LE
Reihenfolge und Strukturierung von Properties / Methoden / Konstruktoren ist inkonsistent	05.06.2012	LE
Code-Guidelines (vorhanden?) einhalten	05.06.2012	LE
PreConditions einsetzen	05.06.2012	LE

Tabelle 32 - Annotationen und Kommentare Code Review 03.05.2012

Die original notierten Kommentare und Annotationen von Silvan Gehrig und Michael Gfeller sind im Anhang (VIII Anhang) zu finden.

#### V.8.8.4.1 Bewertung der Kriterien

Die Kriterienbewertung wurde von Silvan Gehrig und Michael Gfeller am 04.05.2012 vorgenommen.

##### V.8.8.4.1.1 Code Style Analyse

	Erfüllt	N. Erfüllt
Die Code Richtlinien von HSR Videowall wurden eingehalten	X	
Der Code wirkt durch seine Anordnung & Verschachtelung übersichtlich	X	
Die HSR Videowall Headers sind in allen nicht generierten Sourcen vorhanden	X	
Die XML-Kommentar Kompilation wurde in den Projekten aktiviert	X	
Alle Public / Protected Members sind ausreichend Dokumentiert	X	
Die auskommentierten Programm-Stücke sind ausreichend erklärt	X	
Die Projekte enthalten keine toten Programm-Klassen		
Die fehlenden Programmstücke sind mittels TODO-Kommentar beschrieben	X	

Der Code übersetzt ohne Compiler Warnings, die nicht dokumentiert sind und sich nicht in den Test Projekten befinden	X
Es gibt keine Bad Smells im Code	X

#### V.8.8.4.1.2 Exception Handling

	Erfüllt	N. Erfüllt
Der Code enthält keine abgefangenen und ignorierten Ausnahmen	X	
Fehler in asynchronen Prozessen werden mittels Event weitergeleitet	X	
Das Logging erfasst alle Fehler aus allen Funktionalitätsschichten	X	
Das Before/After Pattern wird, wo möglich, mittels using() {} angewendet	X	
IDisposable.Dispose() Methoden werden in jedem Fall aufgerufen	X	

#### V.8.8.4.1.3 Flow Control

	Erfüllt	N. Erfüllt
Es existieren keine Schleifen ohne Abbruchkriterien	X	
Es existieren keine nicht dokumentierte, tote Programmstücke (z.B. if(false) / while (false) / ...)	X	
Rekursive Calls haben immer eine Verankerung und Abbruchbedingung	X	

#### V.8.8.4.1.4 Naming

	Erfüllt	N. Erfüllt
Die Namen der Klassen / Variablen sind selbstbeschreibend	X	
Verwirrende oder falsche Namen sind nicht vorhanden		X
Interface-Klassen beginnen immer I (z.B. IDisposable)	X	
Klassen / Properties / Methoden werden mit PascalCasing geschrieben	X	
Lokale Variablen / Argumente werden mit camelCasing geschrieben	X	
Konstanten werden in PascalCasing geschrieben	X	
Felder in Klassen werden mit _camelCasing geschrieben	X	
Der Code enthält keine Magic Numbers		X

#### V.8.8.4.1.5 Tools

	Erfüllt	N. Erfüllt
Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben	?	
Die Errors von FxCop werden wo sinnvoll behoben	?	

### V.8.8.5 Code Review vom 05.06.2012

Das letzte Code Review wurde am 05.06.2012 durchgeführt. Anwesend waren Michael Gfeller, Silvan Gehrig und Lukas Elmer.

Zuerst wurde der Code allgemein von Michael Gfeller untersucht, die gefundenen Probleme wurden besprochen. Danach wurden die Bewertungen des Code Reviews von Michael Gfeller am 12.06.2012 eingetragen.

Das Review zog einige Verbesserungen nach sich, die sich aus den negativ bewerteten Kriterien ergaben. Im Speziellen waren das:

Beschreibung	Bereinigung geprüft am	Kürzel
Exceptions, die applikationsbedingt sind, durch eine spezifische VideoWallException ersetzen.	07.06.2012	LE
Exception, die keine VideoWallException ist, nicht anzeigen. Nur anzeigen, dass ein Fehler aufgetreten ist ohne spezifisch auf den Fehler einzugehen (zu viele Technische Informationen für den Benutzer).	07.06.2012	LE
Das Canvas des Mousecursors könnte in UserControl ausgelagert werden (MainWindow der VideoWall).	-	-

Beschreibung	Bereinigung geprüft am	Kürzel
Wenn Applikation beendet wird sicherstellen, dass sie wirklich beendet wird mittels Process.CurrentProcess.Kill().	07.06.2012	LE
Mensa Menu vertikal zentrieren.	07.06.2012	LE
Vollbild über mehrere Bildschirme sicherstellen: Manuell programmieren mithilfe der Screen Klasse.	07.06.2012	LE
State Machine Demo Modus: Timer könnte ausgelagert werden und die Methode Tick() könnte public gemacht werden.	-	-
State Machine Demo Modus: Switch Statement könnte mithilfe eines State Patterns umgesetzt werden. Es wurde aber besprochen, dass dies nicht mehr gemacht werden soll, da das Switch Statement eine gute Übersicht bietet und die aktuelle Lösung pragmatisch ist.	07.06.2012, LE Wie besprochen, ist nicht nötig	
Für die PreOrPostCondition.Assert [...] gibt es evt. ein Debug Attribut, damit der Debugger nicht in diese Klasse springt sondern in der Klasse bleibt, wo die Condition tatsächlich fehlgeschlagen hat.	07.06.2012, LE Debug Attribut nicht gefunden	
Tests: Test Directory kann über Konfiguration hinzugefügt werden. Damit können Dateien und Ordner in den Tests verwendet werden.	07.06.2012	LE
Um zu zeigen, dass die Applikation im Betrieb keine (gravierenden) Memory Leaks beinhaltet, soll die Applikation über 24h laufen lassen werden. Danach soll analysiert werden, ob der Memory Verbrauch etwa gleichmäßig ist.	07.06.2012	LE
Das IDisposable Interface wird nicht von allen Klassen implementiert. Dies könnte beim Herunterfahren der Applikation im schlimmsten Fall zu einem Absturz führen. Da dies für die Videowall bis jetzt nicht kritisch ist, muss dies aber nicht noch implementiert werden.	13.06.2012, LE Da dieses Kriterium bisher nicht auf der Kriterienliste des Code Reviews war, wurde dies noch hinzugefügt: „Alle Elemente, die (beim Herunterfahren der Applikation) aufgeräumt werden müssen, implementieren das Interface IDisposable“	

Tabelle 33 - Annotationen und Kommentare Code Review 05.06.2012

#### V.8.8.5.1 Bewertung der Kriterien

Die roten Kommentare wurden von Michael Gfeller eingefügt oder gemäss den Mails zwischen Michael Gfeller und Lukas Elmer vom 12. Juli (VIII Anhang) ergänzt.

##### V.8.8.5.1.1 Code Style Analyse

	Erfüllt	N. Erfüllt
Die Code Richtlinien von HSR Video Wall wurden eingehalten	X	
Der Code wirkt durch seine Anordnung & Verschachtelung übersichtlich	X	
Die HSR Video Wall Headers sind in allen nicht generierten Sourcen vorhanden	X	
Die XML-Kommentar Kompilation wurde in den Projekten aktiviert	X	
<b>Problem nur im DiagnosticApp, das nicht zum Code Review gehört</b>		
Alle Public / Protected Members sind ausreichend Dokumentiert	X	
<b>Problem nur im DiagnosticApp, das nicht zum Code Review gehört</b>		
Die auskommentierten Programm-Stücke sind ausreichend erklärt	X	
Die Projekte enthalten keine toten Programm-Klassen	X	
Die fehlenden Programmstücke sind mittels TODO-Kommentar beschrieben	X	
Der Code übersetzt ohne Compiler Warnings, die nicht dokumentiert sind und sich nicht in den Test Projekten befinden	X	
Es gibt keine Bad Smells im Code		
<b>Diverse Kleine sind noch vorhanden.</b>	X	

### V.8.8.5.1.2 Exception Handling

	Erfüllt	N. Erfüllt
Der Code enthält keine abgefangenen und ignorierten Ausnahmen	X	
Fehler in asynchronen Prozessen werden mittels Event weitergeleitet	X	
Das Logging erfasst alle Fehler aus allen Funktionalitätsschichten	X	
Das Before/After Pattern wird, wo möglich, mittels using() {} angewendet	X	
IDisposable.Dispose() Methoden werden in jedem Fall aufgerufen	X	
Alle Elemente, die (beim Herunterfahren der Applikation) aufgeräumt werden müssen, implementieren das Interface IDisposable		X
Wie besprochen; Noch nicht komplett implementiert		

### V.8.8.5.1.3 Flow Control

	Erfüllt	N. Erfüllt
Es existieren keine Schleifen ohne Abbruchkriterien	X	
Es existieren keine nicht dokumentierte, tote Programmstücke (z.B. if(false) / while (false) / ...)	X	
Rekursive Calls haben immer eine Verankerung und Abbruchbedingung	X	

### V.8.8.5.1.4 Naming

	Erfüllt	N. Erfüllt
Die Namen der Klassen / Variablen sind selbstbeschreibend	X	
Verwirrende oder falsche Namen sind nicht vorhanden	X	
Interface-Klassen beginnen immer mit I (z.B. IDisposable)	X	
Klassen / Properties / Methoden werden mit <i>PascalCasing</i> geschrieben	X	
Lokale Variablen / Argumente werden mit <i>camelCasing</i> geschrieben	X	
Konstanten werden in <i>PascalCasing</i> geschrieben	X	
Felder in Klassen werden mit <i>_camelCasing</i> geschrieben	X	
Der Code enthält keine Magic Numbers	X	

### V.8.8.5.1.5 Tools

	Erfüllt	N. Erfüllt
Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben Die „disabled Warnings“ vom Resharper stören den Lesefluss im Programmcode erheblich. Zusätzlich besteht die Gefahr, dass bei neuen Anpassungen gerade weitere Unschönheiten durch die bestehenden „disabling“-Anweisung ausgeblendet werden. Daher: Lieber nicht disablen und im Source (z.B. als Remark / ...) beschreiben, warum dies so sein muss als zukünftige Implementationen erschweren. Ansonsten ist dies ok.	X	

Die Errors von FxCop werden wo sinnvoll behoben

X
---

## V.9 Betriebskonzept

<b>V.9.1 Änderungsgeschichte .....</b>	<b>183</b>
<b>V.9.2 Betrieb und Administration der Videowall .....</b>	<b>184</b>
V.9.2.1 Betrieb der Videowall.....	184
V.9.2.2 Ablaufsempfehlungen zur Datenverwaltung .....	185
V.9.2.2.1 In der Grundapplikation enthalte Plug-ins .....	185
V.9.2.2.1.1 Poster (Poster-Applikation) .....	185
V.9.2.2.1.2 Mittagsmenu der SV Group (Mittagsmenu -Applikation) .....	185
V.9.2.2.2 Plug-ins im Allgemeinen.....	185
V.9.2.3 Administrationsoberfläche.....	186
V.9.2.3.1 Evaluation der verschiedenen Möglichkeiten.....	186
V.9.2.3.1.1 Administration über Typo3 CMS.....	186
V.9.2.3.1.2 Administration über Web Server .....	187
V.9.2.3.1.3 Administration über WPF-Applikation.....	187
V.9.2.3.2 Umsetzungsempfehlung .....	187
<b>V.9.3 Installationsdokumentation .....</b>	<b>188</b>
V.9.3.1 Dokumentation für Administrator .....	188
V.9.3.2 Dokumentation für Entwickler.....	188
V.9.3.3 Plug-ins.....	188
V.9.3.4 Konfigurationsdatei .....	188
V.9.3.4.1 Konfigurationssektionen.....	188
V.9.3.4.2 Unity .....	189
V.9.3.4.2.1 Interfaces auf konkrete Klassen mappen .....	189
V.9.3.4.2.2 Skelettdaten .....	189
V.9.3.4.2.3 Cursor .....	190
V.9.3.4.3 Padding .....	190
V.9.3.4.3.1 Pfad zu den Plug-ins.....	191
V.9.3.4.3.2 Demomodus .....	191
V.9.3.4.3.3 Singleton .....	191
V.9.3.4.3.4 KinectReplayFile.....	192
V.9.3.4.4 Runtime Version.....	192
<b>V.10.1 Änderungsgeschichte .....</b>	<b>193</b>
<b>V.10.2 Ausblick .....</b>	<b>194</b>
V.10.2.1 Hardware.....	194
<b>V.10.3 Weiterentwicklung .....</b>	<b>194</b>
V.10.3.1 Content Management System .....	194
V.10.3.2 Plug-in Applikationen .....	194
V.10.3.3 Datenverwaltung der Plug-ins .....	195
V.10.3.4 User Stories .....	195

V.10.3.5	Attraktivität verbessern .....	195
V.10.3.5.1	Punktgenaue Beschallung.....	195
V.10.3.6	Unschönheiten und kleinere Fehler.....	195
V.10.3.7	Design.....	196
V.10.3.7.1	CI/CD der HSR .....	196
<b>V.10.4</b>	<b>Zeitplan .....</b>	<b>196</b>
V.10.4.1	Vorschlag zur Weiterentwicklung der HSR Videowall .....	196

---

## V.9.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
25.05.2012	1.1	Erste Version des Dokuments, Administration der Videowall	LE
27.05.2012	1.2	Review Administration der Videowall	CH
27.05.2012	1.3	Review Administration der Videowall	DT
07.06.2012	1.4	Installationsdokumentation	CH
12.06.2012	1.5	Review	DT

## V.9.2 Betrieb und Administration der Videowall

Im folgenden Kapitel wird beschrieben, wie die Videowall gewartet werden soll. Dazu gehören einerseits der allgemeine Betrieb sowie andererseits die Administration der Inhalte der Videowall.

Dieses Kapitel beschreibt den aktuellen Stand und muss bei einer Weiterführung des Projektes nochmals überarbeitet und verfeinert werden.

### V.9.2.1 Betrieb der Videowall

Da die Videowall in der Nacht nicht genutzt wird, ist es möglich, die Wall in dieser Zeit auszuschalten. Dadurch kann der Stromverbrauch gesenkt und die Abnutzung der Hardware, speziell der Monitore, verringert werden.

Da um etwa 7.30 Uhr die ersten Personen an der HSR eintreffen, wird der Videowall PC um etwa 7 Uhr über eine Zeitschaltuhr hochgefahren. Somit verbleiben noch 30 Minuten für allfällige Updates und den Systemstart.

Falls eine neue Version der Software oder der Plug-ins existiert, wird ein automatisches Deployment durchgeführt. Die Informationen und Dateien für dieses Deployment werden automatisch von einem Deployment Server (beispielsweise Team Foundation Server<sup>47</sup>) heruntergeladen. Sollte beim Deployment etwas schief gehen, wird der Videowall PC wieder heruntergefahren.

Nach dem automatischen Deployment wird die Videowall-Applikation gestartet und die Plug-ins werden geladen. Die Bildschirme der Videowall um etwa 7.30 Uhr eingeschaltet. Die Videowall läuft dann den ganzen Tag lang ohne Unterbruch. Sollte ein Fehler auftreten, wird zuerst versucht, die Applikation neu zu starten. Schlägt dies fehl, so wird die Videowall heruntergefahren.

Sobald die Videowall nicht mehr gebraucht wird (ca. um 20 Uhr) wird der Videowall PC heruntergefahren und die Bildschirme ausgeschaltet.

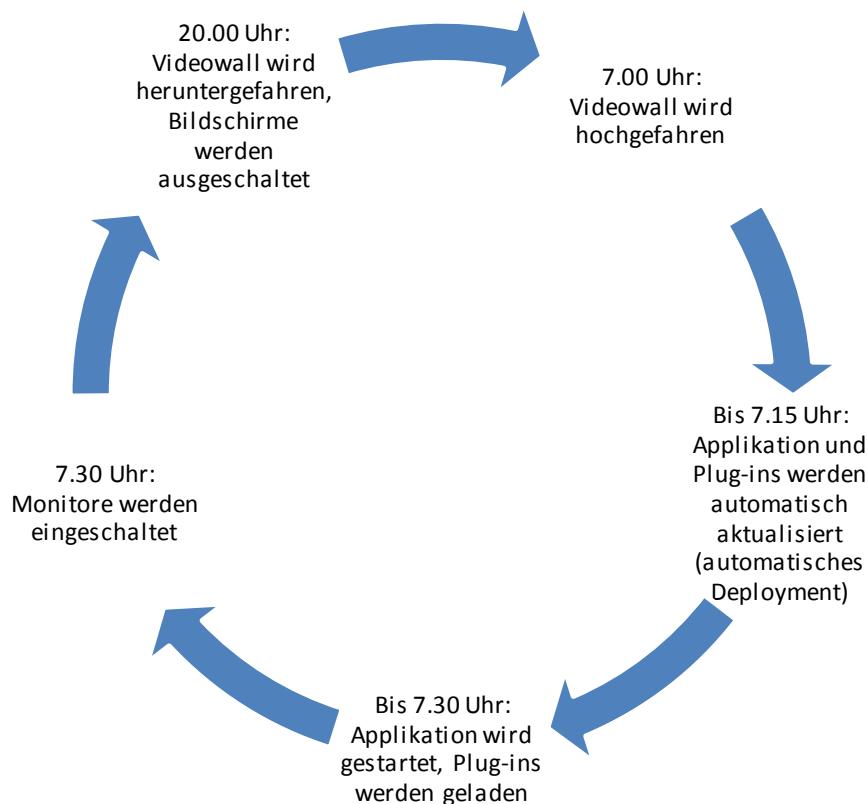


Abbildung 129 - Betrieb der Videowall

Sollte zu irgendeinem Zeitpunkt ein Fehler auftreten, so wird automatisch ein Mail mit dem Log und einem Stack Trace generiert und an die Verantwortlichen geschickt, um herauszufinden, um für einen Fehler es sich

<sup>47</sup> <http://msdn.microsoft.com/en-us/vstudio/ff637362.aspx>

handelt. Zusätzlich wird die Videowall von einem externen Tool (Bsp. Nagios<sup>48</sup>) überwacht, welches bei einem Problem ebenfalls die Verantwortlichen benachrichtigt.

---

### V.9.2.2 Ablaufsempfehlungen zur Datenverwaltung

Plug-ins sind Applikationen, die auf der Videowall dargestellt werden können und müssen verwaltet werden.

---

#### V.9.2.2.1 In der Grundapplikation enthalte Plug-ins

Bis zum Ende des Projektes sind folgende zu verwaltende Inhalte vorgesehen.

##### V.9.2.2.1.1 Poster (Poster-Applikation)

Jedes Semester entstehen neue Bachelorposter, welche für die Videowall aufbereitet werden müssen. Es ist noch nicht bestimmt, wer diese Aufgabe übernehmen wird.

Folgender Ablauf ist denkbar:

1. Die Bachelorposter werden von den Studierenden erstellt.
2. Die Studiengangleiter sind verantwortlich, diese Poster in elektronischer Form entgegenzunehmen und inhaltlich zu kontrollieren.
3. Die Poster werden von den Studiengangleitern dem Sekretariat der HSR übergeben.
4. Das Sekretariat pflegt die Inhalte über ein CMS Interface in die Videowall-Applikation ein.
5. Berichtigungen können dem Sekretariat gemeldet werden, welches die Korrekturen ins CMS Interface einpflegt.

##### V.9.2.2.1.2 Mittagsmenu der SV Group (Mittagsmenu-Applikation)

Das Mittagsmenu ist an den Wochentagen auf der Internetseite der SV Group verfügbar (Typo3 CMS der SV Group) (siehe Unterkapitel V.5.3.3.2 Mittagsmenu). Leider bietet die SV Group keine Schnittstelle außer der HTTP/HTML-Version für das Menu an. Um das Mittagsmenu trotzdem aktuell zu halten, wird der Menuplan beim Start der Applikation in HTML heruntergeladen, die nötigen Informationen herausgelesen und in einer Form zwischengespeichert, die sich für WPF eignet. Die HTTP/HTML-Schnittstelle ist technisch gesehen labil. Eine Design-Änderung an der Website der SV Group könnte dazu führen, dass das Menu nicht mehr richtig eingelesen werden kann. Daher wurden spezifisch für das Mensamenu Unit Tests geschrieben. Sie bieten dem Entwickler die Möglichkeit, die Schnittstelle einfach und schnell zu testen. Dazu wird eine aktuelle Version des HTML heruntergeladen und im Unit Test eingebunden.

Um das Menu zu aktualisieren, gibt es zwei Möglichkeiten:

- Die Applikation wird neu gestartet.
- Es wird ein Cronjob oder ein (Dispatcher-)Timer mit hohem Aktualisierungsintervall programmiert, der beispielsweise alle zwei Stunden das Mittagsmenu aktualisiert.

---

#### V.9.2.2.2 Plug-ins im Allgemeinen

Durch das Plug-in System können Studenten Innovation in die Applikation einfließen zu lassen. Da die Videowall aber an einem prominenten Ort steht und die HSR direkt repräsentiert, ist es wichtig, dass nicht beliebige Inhalte auf der Videowall publiziert werden (z.B. gewaltverherrlichende oder erotische Inhalte). Ein weitere Problematik besteht darin, dass durch die Plug-ins die Stabilität der Videowall negativ beeinträchtigt werden könnte.

Um also qualitativ hochwertige und politisch korrekte Plug-ins sicherzustellen, ist es notwendig, die interessierten Studenten auf gewisse Restriktionen und Regeln aufmerksam zu machen. Es wird vorgeschlagen, dass die Studierenden nach der Entwicklung eines Plug-ins zu einem Code Review eingeladen werden, bei dem der Quellcode des Plug-ins analysiert wird. Zusätzlich müssen die Studenten mit ihrem Namen dafür bürgen, dass durch ihre Erweiterung keine politisch unkorrekten Inhalte auf der Videowall erscheinen. Sollte dies doch passieren, sind im Vorhinein Massnahmen zu definieren, welche bei einer Verletzung der Vorschriften eingeleitet werden.

---

<sup>48</sup> <http://www.nagios.org/>

Folgender Ablauf ist für das Erstellen und Deployen eines Plug-ins denkbar:

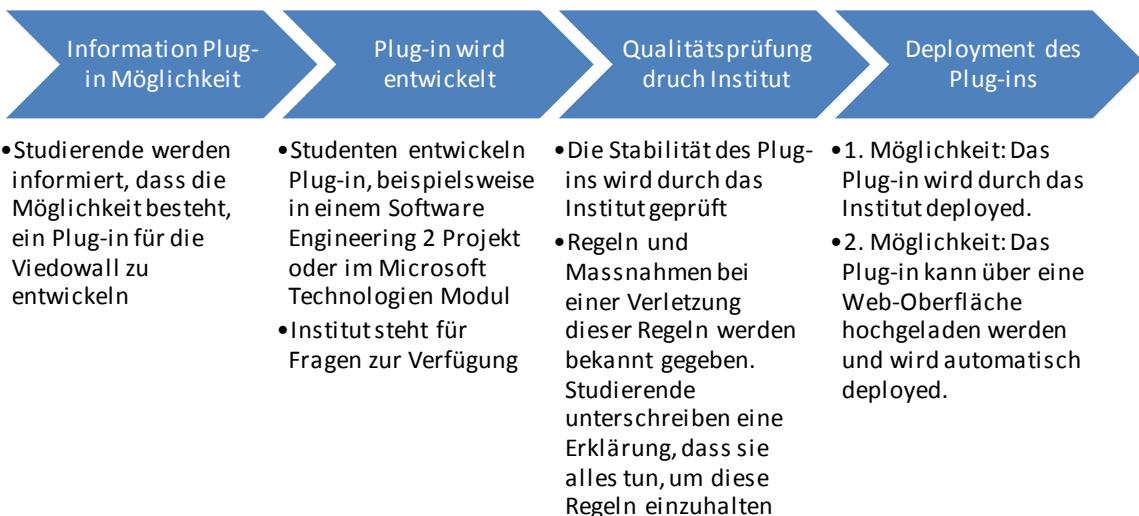


Abbildung 130 - Initialer Deployment Prozess

### V.9.2.3 Administrationsoberfläche

Für die Realisierung einer Administrationsoberfläche der Inhalte gibt es verschiedene Möglichkeiten.

#### V.9.2.3.1 Evaluation der verschiedenen Möglichkeiten

Die drei wichtigsten sind in den nachfolgenden Unterkapiteln kurz beschrieben und bewertet.

##### V.9.2.3.1.1 Administration über Typo3 CMS

Das Sekretariat der HSR<sup>49</sup> arbeitet bereits mit einem Typo3 CMS<sup>50</sup>. Eine Integration der Administration der Videowall in dieses System wäre eine Option. Dabei könnte auf zwei Arten vorgegangen werden:

###### V.9.2.3.1.1.1 TYPO3 EXTENSION MIT TYPO3 DATENBANK

Es wird eine Typo3 Extension<sup>51</sup> für den Administrationsbereich entwickelt. Diese Applikation wird sehr einfach gehalten und zeichnet sich vor allem durch XML Konfigurationen aus, die das Datenbankschema beschreiben. Durch die Installation der Extension im Typo3 wird die Datenbank automatisch erzeugt und die Inhalte können sofort über den Administrationsbereich von Typo3 bearbeitet werden. Die grafische Oberfläche des Administrationsbereichs wird automatisch vom Framework generiert und sieht so wie eine gewöhnliche Typo3 Extension aus. Gewisse Standardfunktionalitäten wie die Archivierungsoption oder das Ausblenden von einzelnen Datensätzen werden ebenfalls durch das Framework angeboten.

Der Nachteil an dieser Lösungsvariante ist, dass die Daten von der Typo3 Datenbank zur Videowall migriert werden müssen. Eine Synchronisation des unidirektionalen Informationsflusses von der Typo3 Datenbank zur Videowall kann aber auf einfache Art per Cronjob (z.B. alle 15 Minuten) eingerichtet werden.

Diese Variante eignet sich gut, wenn alle Personen, die an der Videowall etwas ändern müssen, Zugriff auf Typo3 haben. Speziell für das Sekretariat ist diese Art von Interface einfach zu bedienen, es wird bereits täglich benutzt.

###### V.9.2.3.1.1.2 WEB INTERFACE UND TYPO3 EXTENSION MIT IFRA

Wie in der ersten Lösungsvariante (siehe Unterkapitel V.9.2.3.1.1.1 Typo3 Extension mit Typo3 Datenbank) ist auch hier eine Typo3 Extension vorgesehen. Dieses Mal wird die Extension aber so erstellt, dass nur ein Iframe

<sup>49</sup> <http://www.hsr.ch/typo3/index.php>

<sup>50</sup> <http://typo3.org>

<sup>51</sup> <http://typo3.org/extensions/repository/>

programmiert wird, das auf einen anderen Web Server verweist. Somit kann die Administrationsoberfläche Typo3-technologieunabhängig entwickelt werden, zum Beispiel mit ASP.NET MVC3.

Die Hauptvorteile dieses Ansatzes sind, dass die Administrationsoberfläche nicht mit Typo3 programmiert werden muss und trotzdem ins Typo3 integriert ist. Die Administrationsoberfläche kann auch ohne Typo3 bearbeitet werden, gegebenenfalls mit einem SSO. Die Implementierung dieser Lösungsvariante ist allerdings etwas zeitaufwändiger.

#### V.9.2.3.1.2 Administration über Web Server

Ähnlich wie in der zweiten Typo3-Lösungsvariante V.9.2.3.1.1.2 Web Interface und Typo3 Extension mit Iframe beschrieben ist, wird bei dieser Lösung auf einem Web Server (z.B. mit ASP.NET MVC3) eine Administrationsoberfläche entwickelt, die gegebenenfalls mit dem SSO der HSR gekoppelt wird. Als Transportprotokoll dient HTTPS/HTML.

Die Vorteile liegen darin, dass das System klar von anderen Applikationen abgegrenzt ist. Auch ist es einfach möglich, eine mobile Applikation mit HTML5 zu entwickeln. Als Nachteil ist jedoch aufzuführen, dass ohne Typo3 Extension die Benutzer auf eine separate URL zugreifen müssen und ihnen das System nicht sofort bekannt vorkommt. Für diese Variante könnte statt ASP.NET auch Silverlight eingesetzt werden.

#### V.9.2.3.1.3 Administration über WPF-Applikation

In dieser Variante geht es darum, einen WPF Client zu schreiben, mit dem die Inhalte bearbeitet werden können. Als Transportprotokoll würde WCF eingesetzt werden.

Bei dieser Variante können grosse Teile aus den Daten- und Serviceschichten (siehe V.6.5.2 Logische Sicht) der bestehenden Software wiederverwendet werden, was ein Vorteil ist. Nachteilig ist, dass durch WPF die Plattform eingeschränkt wird und die Entwicklung einer mobilen Applikation so nicht möglich ist.

Diese Variante wird nicht empfohlen.

---

#### V.9.2.3.2 Umsetzungsempfehlung

Das Bachelorteam hätte bei genügend Zeit die Lösungsvariante, welche im Unterkapitel V.9.2.3.1.1.2 Web Interface und Typo3 Extension mit Iframe beschrieben ist, für die Videowall umgesetzt.

Vor der Umsetzung ist die ausgewählte Lösungsmöglichkeit mit der Kommunikationsstelle der HSR zu validieren.

## V.9.3 Installationsdokumentation

### V.9.3.1 Dokumentation für Administrator

Damit die Applikation verwendet werden kann wird folgendes benötigt:

- Kompilierte Version des Projektes
- .NET, Kinect Runtime
- Internetverbindung

Die kompilierten Dateien müssen als erstes in den gewünschten Zielordner kopiert werden.

Weiter können verschiedene Plug-ins zur Applikation hinzugefügt werden. Damit dies möglich ist, muss die Ordnerstruktur nach der Anleitung im Kapitel V.9.3.3 Plug-ins aufgebaut werden.

### V.9.3.2 Dokumentation für Entwickler

Damit die Applikation weiterentwickelt werden kann, muss ein SVN Checkout auf dem gesamten SVP Repository durchgeführt werden. Dazu wird ein SVN Tool wie z.B. TortoiseSVN benötigt (siehe V.4.2 Tools).

Zudem werden folgende Werkzeuge benötigt:

- Visual Studio 10
- Expression Blend
- ReSharper
- Kinect for Windows SDK
- NuGet

Die Konfiguration der Applikation findet in der app.config-Datei statt. Wie die Konfiguration angepasst werden kann, wird im Kapitel V.9.3.4 Konfigurationsdatei erläutert.

Weiter können der Applikation Plug-ins hinzugefügt werden, dies wird im nachfolgenden Kapitel (V.9.3.3 Plug-ins) aufgezeigt.

### V.9.3.3 Plug-ins

Im app.config-File kann der Pfad zum Ordner, welcher die Plug-ins enthalten soll, konfiguriert werden. Wie dieser Pfad gesetzt und angepasst werden kann, wird im Unterkapitel V.9.3.4.3.1 Pfad zu den Plug-ins erläutert.

Für jede Plug-in-Applikation, die man auf der Videowall anzeigen möchte, muss ein entsprechender Unterordner im Plug-ins Ordner (siehe V.9.3.4.3.1 Pfad zu den Plug-ins) erstellt werden, in welchem dann die Files des Plug-ins gespeichert werden. Ein solcher Ordner darf nie leer sein, da sonst beim Starten der Videowall-Applikation eine entsprechende Fehlermeldung angezeigt wird.

Bestimmte Plug-in-Applikationen benötigen zusätzliche Dateien wie beispielsweise Bilder. Diese müssen in einem Unterordner mit dem Namen „Files“ im Ordner des Plug-ins abgelegt werden. Fehlt dieser Ordner, so wird eine entsprechende Fehlermeldung beim Starten der Hauptapplikation angezeigt.

### V.9.3.4 Konfigurationsdatei

Die Einstellungen, welche in der Konfigurationsdatei getätigten werden können, sind nachfolgend aufgezeigt.

#### V.9.3.4.1 Konfigurationssektionen

Im Abschnitt configSections können verschiedene Konfigurationssektionen definiert werden.

```
<configSections>
    <section name="unity" type="Microsoft.Practices.Unity.Configuration.UnityConfigurationSection, Microsoft.Practices.Unity.Configuration"/>
</configSections>
```

Abbildung 131 - app.config, Konfigurationssektionen

#### V.9.3.4.2 Unity

In der Konfigurationssektion Unity werden die Namespaces angegeben, in denen die Klassen gesucht werden sollen, welche über Unity instanziiert werden sollen. Zu den Namespaces muss weiter angegeben werden, in welchem Assembly diese sich befinden.

```
<namespace name="VideoWall.Data.Kinect.Implementation"/>
<namespace name="VideoWall.Data.Kinect.Interfaces"/>
<assembly name="VideoWall.Data"/>

<namespace name="VideoWall.ServiceModels.Apps.Implementation"/>
<namespace name="VideoWall.ServiceModels.Apps.Interfaces"/>
<namespace name="VideoWall.ServiceModels.DemoMode.Implementation"/>
<namespace name="VideoWall.ServiceModels.DemoMode.Interfaces"/>
<namespace name="VideoWall.ServiceModels.HandCursor.Implementation"/>
<namespace name="VideoWall.ServiceModels.HandCursor.Interfaces"/>
<namespace name="VideoWall.ServiceModels.Player.Implementation"/>
<namespace name="VideoWall.ServiceModels.Player.Interfaces"/>
<assembly name="VideoWall.ServiceModels"/>

<namespace name="VideoWall.ViewModels"/>
<namespace name="VideoWall.ServiceModels.Apps"/>
<namespace name="VideoWall.ViewModels.Cursor"/>
<namespace name="VideoWall.ViewModels.HitButton"/>
<namespace name="VideoWall.ViewModels.Menu"/>
<namespace name="VideoWall.ViewModels.Skeletons"/>
<assembly name="VideoWall.ViewModels"/>
```

Abbildung 132 - app.config, Sektion Unity, Namespaces&Assemblies

##### V.9.3.4.2.1 Interfaces auf konkrete Klassen mappen

Weiter können in der Sektion Unity mithilfe des Tags <container> Interfaces auf konkrete Klassen gemappt werden. Dies wird wie folgt angegeben:

```
<type type="IDemoModeService" mapTo="DemoModeService"/>
<type type="IAppController" mapTo="AppController"/>
<type type="IHandCursorPositionCalculator" mapTo="HandCursorPositionCalculator"/>
<type type="IPlayer" mapTo="Player"/>
```

Abbildung 133 - app.config, Sektion Unity, Mapping von Interfaces auf Klassen

##### V.9.3.4.2.2 Skelettdaten

Der *ISkeletonReader* dient dazu, die Skelettdaten zu Lesen.

Der *KinectSkeletonReader* wird verwendet, wenn Kinect angeschlossen ist und die Videowall normal betrieben wird.

Für Testzwecke werden der *FileSkeletonReader* und der *AutoPlayFileSkeletonReader*, welche mit Kinect aufgenommene Skelettdaten abspielen, angeboten. Der *FileSkeletonReader* spielt das File einmal, der *AutoPlayFileSkeletonReader* spielt das File immer wieder von Neuem ab. Das abzuspielende File mit den Skelettdaten kann über den Typ *KinectReplayFile* definiert werden, was im Unterkapitel V.9.3.4.3.4 *KinectReplayFile* erläutert wird.

```
<!-- FileSkeletonReader -->
<!-- AutoPlayFileSkeletonReader -->
<!-- KinectSkeletonReader -->
<type type="ISkeletonReader" mapTo="KinectSkeletonReader">
    <lifetime type="singleton"/>
</type>
```

Abbildung 134 - app.config, Sektion Unity, Mapping für ISkeletonReader

#### V.9.3.4.2.3 Cursor

Ist der Kinect Sensor am Computer angeschlossen, so ist für das Mapping vorzugsweise das *KinectCursorViewModel* zu verwenden. Der Cursor, welcher in der Applikation als kleine Hand dargestellt wird, kann so durch Handbewegungen vor dem Kinect Sensor gesteuert werden. Ist das Mapping für den *ISkeletonReader* (siehe Unterkapitel V.9.3.4.2.2 Skelettdaten) auf *FileSkeletonReader* oder *AutoPlayFileSkeletonReader* gesetzt, so wird der Handcursor durch das Skelett im abzuspielenden File gesteuert.

Wird das Schlüsselwort *MouseCursorViewModel* verwendet, so lässt sich die Applikation mit der Maus bedienen. Trotzdem wird auch hier eine Hand angezeigt.

```
<!-- KinectCursorViewModel -->
<!-- MouseCursorViewModel -->
<type type="ICursorViewModel" mapTo="MouseCursorViewModel">
    <lifetime type="singleton"/>
</type>
```

Abbildung 135 - app.config, Sektion Unity, Mapping für ICursorViewModel

#### V.9.3.4.3 Padding

Damit die Applikation angenehm mit Kinect zu bedienen ist, wurde einer Interaktionszone für die Hand definiert. Das Prinzip dieser Zone wird im Kapitel V.6.8 Interaktion durch Handtracking genauer erläutert. Die rote Fläche in der Abbildung 136 - Interaktionszone ist auf den Arm des Nutzers ausgerichtet und stellt die eigentliche Interaktionszone dar. Die schwarz umrahmte Fläche stellt den Erkennungsbereich von Kinect dar. Das *RelativePadding* wird verwendet, um diese Zone zu definieren. Möchte man die Position und Grösse der Zone verändern, so kann dies durch Anpassungen an den Werten der aufgelisteten Schlüsselwörter *left*, *top*, *right* und *bottom* getan werden. Welchen Abstand diese Schlüsselwörter verändern ist in der Abbildung 136 - Interaktionszone ersichtlich.

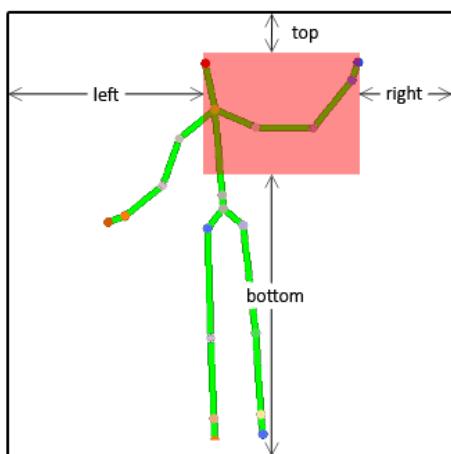


Abbildung 136 - Interaktionszone

```
<type type="RelativePadding" mapTo="RelativePadding">
    <constructor>
        <param name="left" value="0.45"/>
        <param name="top" value="0.1"/>
        <param name="right" value="0.3"/>
        <param name="bottom" value="0.49"/>
    </constructor>
</type>
```

Abbildung 137 - app.config, Sektion Unity, Padding

### V.9.3.4.3.1 Pfad zu den Plug-ins

Die Applikation kann durch Plug-ins dynamisch erweitert werden. Diese müssen in einem bestimmten Ordner abgelegt werden. Der Parameter `extensionsDirectoryPath` muss auf den gewünschten Pfad gesetzt werden.

```
<type type="ExtensionsConfig" mapTo="ExtensionsConfig">
<constructor>
    <param name="extensionsDirectoryPath" value="../../../../Extensions"/>
</constructor>
</type>
```

Abbildung 138 - app.config, Sektion Unity, Pfad zu den Plug-ins

### V.9.3.4.3.2 Demomodus

Der Demomodus wird aktiv, wenn Kinect keinen Nutzer erkennt. Einstellungen für den Demomodus können im Bereich `IDemoModeConfig` vorgenommen werden.

```
<type type="IDemoModeConfig" mapTo="DemoModeConfig">
<constructor>
    <param name="backgroundColors">
        <array>
            <value value="#ff0065a3" />
            <value value="#ff6e1c50" />
            <value value="#fff548c86" />
            <value value="#fff7b6951" />
            <value value="#ff00738d" />
            <value value="#ffbabd5d" />
        </array>
    </param>
    <param name="skeletonCheckTimeSpan" value="00:00:00.01" />
    <param name="changeAppTimeSpan" value="00:00:05" />
    <param name="fromActiveToDemoModeTimeSpan" value="00:00:15" />
    <param name="countdownTimeSpan" value="00:00:04.999" />
    <param name="skeletonTrackingTimeoutTimeSpan" value="00:00:00.5" />
</constructor>
</type>
```

Abbildung 139 - app.config, Sektion Unity, Demomodus

Als Werte für den Parameter `backgroundColors` können Farben für den Hintergrund des Demomodus angegeben werden.

Des Weiteren können die Werte der verschiedenen Timer angepasst werden.

Der `skeletonCheckTimeSpan` gibt an, nach wie vielen Millisekunden erneut geprüft wird, ob gerade ein Nutzer erkannt wurde.

Über den `changeAppTimeSpan` kann die Zeitspanne bis zum Wechsel des Teaser-Textes und der Hintergrundfarbe im aktiven Demomodus definiert werden.

Wenn die Applikation aktiv ist aber kein Skelett mehr erkennt, wechselt sie nach Ablauf des `fromActiveToDemoModeTimeSpan` in den Demomodus.

Wird im Demomodus ein Nutzer erkannt, so wird ein Countdownzähler angezeigt, welcher den Übergang in den Interaktionsmodus andeutet. Die Länge dieses Zählers kann über den `countdownTimeSpan` verändert werden.

Nachdem ein Nutzer von Kinect erkannt wurde, wird immer wieder geprüft, ob der Nutzer immer noch erkannt wird. Die Zeitspanne bis zur nächsten Überprüfung wird über den `skeletonTrackingTimeoutTimeSpan` angegeben.

### V.9.3.4.3.3 Singelton

Von bestimmten Klassen darf es immer nur eine Instanz geben, welche dann an mehreren Orten verwendet werden kann.

Die Klasse `Player` muss ein Singleton sein: Das Skelett des Kinect Players wird zum einen beim Übergang vom Demomodus in den Interaktionsmodus und schliesslich im Interaktionsmodus selbst angezeigt. Die Problematik mit Kinect ist, dass das Gerät nur von einer Instanz angesteuert werden kann. Bestünden nun zwei

verschiedene Skelette, die das Kinect ansteuern und Daten vom Gerät verlangen, würde das in einem Fehler enden.

Im Falle des MenuViewModels darf nur eine Instanz existieren, welche alle Plug-ins im Menu aufzeigt und den Inhalten des aktuell angewählten auf der Videowall anzeigen.

```
<type type="Player" mapTo="Player">
  <lifetime type="singleton"/>
</type>

<type type="MenuViewModel" mapTo="MenuViewModel">
  <lifetime type="singleton"/>
</type>
```

Abbildung 140 - app.config, Sektion Unity, Singleton

#### V.9.3.4.3.4 KinectReplayFile

Wird im app.config beim Mapping auf den ISkeletonReader (siehe V.9.3.4.2.2 Skelettdaten) der *FileSkeletonReader* oder der *AutoPlayFileSkeletonReader* verwendet, so muss ein *KinectReplayFile* angegeben werden. Dieses beinhaltet Daten zu Skelettbewegungen. Der Pfad zu dieser Datei kann an folgender Stelle beim Schlüsselwort *path* angegeben werden:

Abbildung 141 - app.config, Sektion Unity, KinectReplayFile

Der Pfad zum Replay-File ist relativ anzugeben.

#### *V.9.3.4.4 Runtime Version*

Über das Schlüsselwort *runtime* können die Pfade zu den neuen Assembly-Versionen und deren Speicherort angegeben werden. Diese Angaben werden benötigt um Assembly-Versionskonflikte zu vermeiden.

```
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="Microsoft.Kinect" publicKeyToken="31BF3856AD364E35"
culture="neutral"/>
      <bindingRedirect oldVersion="1.0.0.0-1.5.0.0" newVersion="1.5.0.0"/>
    </dependentAssembly>
  </assemblyBinding>
</runtime>
```

Abbildung 142 - app.config, Runtime

## V.10 Ausblick

<b>V.10.1 Änderungsgeschichte .....</b>	<b>193</b>
<b>V.10.2 Ausblick .....</b>	<b>194</b>
V.10.2.1     Hardware.....	194
<b>V.10.3 Weiterentwicklung .....</b>	<b>194</b>
V.10.3.1     Content Management System .....	194
V.10.3.2     Plug-in Applikationen .....	194
V.10.3.3     Datenverwaltung der Plug-ins .....	195
V.10.3.4     User Stories .....	195
V.10.3.5     Attraktivität verbessern .....	195
V.10.3.5.1     Punktgenaue Beschallung.....	195
V.10.3.6     Unschönheiten und kleinere Fehler.....	195
V.10.3.7     Design.....	196
V.10.3.7.1     CI/CD der HSR .....	196

---

### V.10.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
05.06.2012	1.0	Erste Version des Dokuments	CH
06.06.2012	1.1	Review	DT
10.06.2012	1.2	Review und Korrekturen	LE
14.06.2012	1.3	Ergänzungen	LE
14.06.2012	1.4	Korrekturen	DT

---

## V.10.2 Ausblick

Die Bachelorarbeit ist eine Machbarkeitsstudie. Mit ihr wurde eruiert, ob eine Anschaffung einer Videowall für die HSR sinnvoll ist, was im Laufe der Arbeit erwiesen werden konnte. Die Machbarkeitsstudie ist die Grundlage für eine mögliche Weiterentwicklung durch das Institut für Software (IFS).

Da die Videowall zusammen mit Kinect so viele neue Möglichkeiten bietet, muss unbedingt darauf geachtet werden, dass die Entwicklung pragmatisch und iterativ erfolgt. Denn nur so ist es möglich, die nötigen Entscheidungen richtig zu treffen, die umgesetzte Anwendung durch Usability Tests zu prüfen und auf die dadurch nötigen Anpassungen zu reagieren.

Bei den Hardwarekomponenten ist eine definitive Entscheidung für eine bestimmte Konfiguration zu treffen. Der Entscheid kann auf eine der bereits eingeholten Offerten (VIII Anhang) fallen oder es müssen noch weiterführende Nachforschungen und Tests gemacht werden, um die optimale Videowall für die HSR beschaffen zu können.

---

### V.10.2.1 Hardware

Es ist damit zu rechnen, dass sich die 4k Video Technologie (2 x Full HD) verbreiten wird und es deshalb möglich wird oder sogar schon ist, mit einer hohen Auflösung von 3840x2160 Inhalte flüssig abzuspielen. Allerdings ergibt die Verteilung einer solch hohen Auflösung auf 9 Monitore Probleme. Hierzu gibt es jedoch bereits ganz neue Produkte, z.B. x4 Wall Controllers von der Firma Datapath<sup>52</sup>, die diese Verteilung unterstützen. Es ist allerdings zu beachten, dass diese Lösungsvariante nur zu 2 x Full HD führt. Es ist somit leider nicht möglich, die volle Auflösung der neun Monitore mit Full HD-Auflösung auszunutzen.

Es wird empfohlen, die Entwicklung der Grafikkarten weiter abzuwarten und zu beobachten. Die ideale Lösung ist eine Grafikkarte, die eine Auflösung von 5760x3240 (3 x Full HD) bietet und es möglich ist, diese hohe Auflösung performant auf neun Monitore zu verteilen.

---

## V.10.3 Weiterentwicklung

---

### V.10.3.1 Content Management System

Bei einer Weiterführung der Videowall muss primär ein Content Management zur Administration der Inhalte der Videowall entwickelt werden.

---

### V.10.3.2 Plug-in Applikationen

Die Videowall verfügt derzeit über zwei Inhalte: die Poster-Applikation und das Mittagsmenu der Mensa. Abzuklären wäre hierbei, ob weitere Applikationen zum Grundumfang der Videowall-Anwendung gehören sollen.

Soll die Poster-Applikation weiter betrieben werden, so sind zwei Themen zu besprechen und zu lösen: Mit der in der Machbarkeitsstudie erarbeiteten Hardware-Lösung sind nicht alle Poster lesbar. Es muss daher eine Möglichkeit erarbeitet werden, diese Poster lesbar zu machen. Dies könnte einerseits über eine Zoom-Möglichkeit gelöst werden oder über einen vordefinierten Pfad, über den der Benutzer durch das vergrößert angezeigte Poster geführt wird. Diesbezüglich wurde im Verlauf des Projekts die Verwendung von Prezi<sup>53</sup> besprochen. Dabei handelte es sich um ein Präsentationshilfsmittel, mit welchem mittels Zoom bestimmte Bereiche einfach vergrößert werden können. Prezi läuft im Browser und ein Browser wiederum kann einfach in WPF eingebunden werden.

Die Bachelorposter sind möglicherweise in ihrer statischen Form nicht attraktiv genug. Interaktive Elemente auf einem Plakat könnten diese Attraktivität wesentlich steigern, wodurch der Nutzer auf spielerische Art Informationen sammeln könnte.

---

<sup>52</sup> <http://www.datapath.co.uk/products/multi-display-products/datapath-x4>

<sup>53</sup> Weitere Informationen: <http://prezi.com/>

Für die Mittagsmenu-Applikation fehlt ein Cronjob, welcher zu Beginn eines neuen Tages das Mittagsmenu der Mensa aktualisiert. Dieser wurde vorerst weggelassen, da davon ausgegangen wird, dass die Videowall über Nacht ausgeschaltet werden wird.

Wollen Studenten eine Applikation für die Wall erstellen, müssen klare Regeln für den Ablauf der Erstellung und Abnahme und den Inhalt der Anwendung aufgestellt werden.

---

#### V.10.3.3 Datenverwaltung der Plug-ins

Jedes Plug-in kann über eigene Daten verfügen. Die Poster-Applikation benötigt beispielsweise die Bilder der anzuzeigenden Poster. Das Framework könnte ein Interface zur Verfügung stellen, über welches die Daten der Plug-ins verwaltet werden können. Die Entwickler eines Plug-ins definieren die Objekte, welche verwaltet werden sollen. Das Framework generiert dann automatisch eine Benutzeroberfläche für deren Bearbeitung.

Alternativ könnte jedes Plug-ins eine eigene Administrationsoberfläche anbieten. Da bei dieser Variante Funktionen (beispielsweise das Speichern der Daten in einer Datenbank) redundant programmiert werden müssten und die Bedienung nicht einheitlich wäre, ist sie jedoch weniger geeignet.

---

#### V.10.3.4 User Stories

Zu Beginn und während des Projekts wurde ein Backlog für die Entwicklung der Videowall erstellt. Dieser enthält User Stories, die noch nicht abgearbeitet werden konnten (siehe V.4.3 Funktionale Anforderungen). Diese könnten bei einer Weiterentwicklung umgesetzt werden.

---

#### V.10.3.5 Attraktivität verbessern

Um die Attraktivität zu verbessern, könnten Animationen in die Applikation eingebaut werden. Diese sollen dem Nutzer helfen, gewisse Aktionen und Prozesse besser nachvollziehen zu können, wie beispielsweise der Wechsel der Menu-Tabs.

Weiter könnte der Demomodus verbessert werden. Es ist vorstellbar, dass gewisse Inhalte wie das Mittagsmenu oder Bilder dazu bereit in diesem Modus im Hintergrund angezeigt werden.

---

#### V.10.3.5.1 Punktgenaue Beschallung

Vorerst verfügt die Videowall über keinen Ton, da dieser die Mitarbeitenden des Verwaltungsgebäudes stören könnte. Jedoch gibt es Systeme, welche eine punktgenaue Beschallung<sup>54</sup> ermöglichen. Dies bedeutet, dass Töne nur in einem bestimmten Bereich hörbar sind. Solche Systeme werden beispielsweise für Messen verwendet und könnten für die Videowall im Kinect-Erkennungsbereich eingesetzt werden.

---

#### V.10.3.6 Unschönheiten und kleinere Fehler

Es mehrere kleinere Unschönheiten bekannt. Eine, welche es zu bereinigen gilt, ist, dass wenn sich die Applikation im Demomodus befindet sind die Elemente im GUI weiterhin anklickbar (z.B. die Navigation). Weitere Punkte sind als User Stories in den Backlog (siehe V.4.3 Funktionale Anforderungen) aufgenommen worden.

Bei den Gesprächen mit Michael Gfeller wurde die Problematik mit relativen Pfaden angesprochen, die über die Konfigurationsdatei angepasst werden müssen. Gemäss Michael Gfeller wäre es besser, wenn solche relativen Pfade nicht existieren würden.

---

<sup>54</sup> Wird beispielsweise von den Firmen i-AUDIOPOINT (<http://www.i-audiopoint.com>) und audionovum (<http://www.audionovum.ch>) angeboten.

---

### V.10.3.7 Design

Vereinzelte Design Elemente der Applikation, wie beispielsweise das Menu, müssen nach dem Erwerb der Videowall auf die Grösse der Monitorfläche bzw. der Auflösung angepasst werden.

---

#### V.10.3.7.1 CI/CD der HSR

Gegen Ende des Projekts wurde mit der Stelle für Kommunikation an der HSR abgeklärt, ob das grafische Design der Applikation den Anforderungen der HSR genügen würde. Die ist grundsätzlich der Fall ist, es wurden trotzdem noch Verbesserungsvorschläge gemacht (siehe V.5.4.4.4 Corporate Design HSR).

Für den produktiven Einsatz der Videowall wird es als wichtig erachtet, das Design von der Kommunikationsstelle der HSR absegnen zu lassen. Es wird vorgeschlagen, dies erst dann zu tun, wenn die Applikation auf der Videowall-Hardware läuft, da der Anblick der Applikation auf der Wall einen ganz anderen Eindruck machen wird als nur ein Printscreen des Designs.

---

## V.10.4 Zeitplan

---

### V.10.4.1 Vorschlag zur Weiterentwicklung der HSR Videowall

Zum Zeitpunkt der Abgabe dieser Bachelorarbeit besteht ein funktionsfähiger Prototyp. Damit daraus ein fertiges Produkt entstehen kann, ist noch Weiterentwicklungsarbeit nötig. Diese Weiterentwicklung wird durch das Institut für Software an der HSR (IFS) erfolgen.

Wie im Kapitel V.9.2.1 Betrieb der Videowall beschrieben, ist ein automatisches Deployment der Applikation vorgesehen. Dabei handelt es sich um eine zentrale Funktion, die entwickelt werden soll. Zusätzlich ist die Möglichkeit der Administration der Inhalte über ein CMS wichtig, diese muss ebenfalls umgesetzt werden.

Für den Lebenszyklus werden die folgenden groben Meilensteine vorgeschlagen:

- Bis Herbst 2012: Beschaffung der Videowall Hardware
- Bis Oktober 2012: Weiterentwicklung der Videowall bis im Oktober 2012 (siehe V.10 Ausblick)
- Ende Oktober: erstes Deployment der Videowall und des CMS
- November 2012: Erfahrungen sammeln und wichtige Verbesserungen und Anpassungen implementieren

Danach soll eine Weiterentwicklung in einem halbjährlichen Zyklus erfolgen, jeweils immer nachdem die neuen Bachelorposter auf der HSR Videowall verfügbar sind.

Während dem Betrieb der Videowall ist es zudem notwendig, dass eine verantwortliche Instanz definiert wird, die bei Problemen und Fehlern der Videowall diese bearbeiten kann.

# VI. Projekt Retrospektive

<b>VI.1 Änderungsgeschichte .....</b>	<b>198</b>
<b>VI.2 Methoden und Technologien .....</b>	<b>199</b>
<b>VI.3 Persönliche Berichte .....</b>	<b>200</b>
VI.3.1 Lukas Elmer .....	200
VI.3.2 Christina Heidt.....	202
VI.3.3 Delia Treichler.....	203
<b>VI.4 Aufwandsanalyse .....</b>	<b>204</b>
VI.4.1 Sprints .....	204
VI.4.1.1 Sprint 1.....	205
VI.4.1.2 Sprint 2.....	206
VI.4.1.3 Sprint 3.....	206
VI.4.1.4 Sprint 4.....	207
VI.4.1.5 Sprint 5.....	207
VI.4.1.6 Sprint 6.....	208
VI.4.1.7 Sprint 7.....	208
VI.4.1.8 Sprint 8.....	209
VI.4.1.9 Sprint 9.....	210
VI.4.1.10 Sprint 10.....	210
VI.4.1.11 Sprint 11.....	211
VI.4.1.12 Sprint 12 .....	212
VI.4.1.13 Sprint 13 .....	213
VI.4.1.14 Sprint 14 .....	214
VI.4.1.15 Sprint 15 .....	214
VI.4.1.16 Sprint 16 .....	215
VI.4.2 Personenaufwand .....	217
VI.4.3 Tätigkeiten .....	218
VI.4.4 Arbeitslisten .....	220
VI.4.4.1 Lukas Elmer .....	220
VI.4.4.2 Christina Heidt.....	223
VI.4.4.3 Delia Treichler.....	228

## VI.1 Änderungsgeschichte

Datum	Version	Änderung	Autor
12.06.2012	1.0	Erste Version des Dokuments	CH
13.06.2012	1.1	Persönlicher Bericht Christina Heidt	CH
13.06.2012	1.2	Methoden und Technologien, Aufwandsanalyse	CH
14.06.2012	1.3	Persönlicher Bericht Lukas Elmer	LE
14.06.2012	1.4	Persönlicher Bericht Delia Treichler	DT
14.06.2012	1.5	Review	DT

## VI.2 Methoden und Technologien

Zu Beginn des Projekts wurde eine grobe Projektplanung in Redmine (Projektplanungstool und Ticketingsystem) gemacht, in der feste Termine für Sitzungen, Abgaben und Abschlussarbeiten, wie beispielsweise die Erstellung der persönlichen Berichte oder der Aufwandsanalyse, festgehalten wurden.

In der ersten Hälfte des Projekts wurde stark risikoorientiert gearbeitet. So wurde zum Beispiel am Anfang eine Passantenanalyse durchgeführt, um festzustellen, ob die Passanten überhaupt im Erkennungsbereich von Kinect durchlaufen. Die Durchführung der verschiedenen Hardware-Tests gehörte ebenfalls zu den Risiko-Themen.

In der zweiten Hälfte des Projekts wurde nach Scrum gearbeitet. Die im Backlog enthaltenen User Stories konnten einfach priorisiert und wieder neu priorisiert werden, damit der Prototyp schliesslich die gewünschten Kernfunktionen anbieten kann. Das Projekt konnte aufzeigen, dass sich eine agile und eine risikoorientierte Entwicklung gut ergänzen.

Neben den rein technischen Risiken orientierten sich einige davon auch an den Nutzerbedürfnissen. So wurden die Steuerung mittels Kinect und später der Prototyp immer wieder durch Usability Tests geprüft und verifiziert. Die nutzerorientierte Komponente führte auch zur Ausarbeitung von Personas und Szenarien.

Um eine spätere, einfache Weiterentwicklung durch die Assistenten des IFS zu gewährleisten, wurden mehrere Code Reviews mit den Assistenten des Instituts durchgeführt. Am Ende des Projekts wurde zudem von einem einzelnen Teammitglied nochmals Refactoring betrieben, um den Code möglichst einheitlich und dadurch einfach weiterentwickelbar zu machen.

Kurz nach Projektbeginn wurde das Team auf den Microsoft Imagine Cup aufmerksam. Um mitmachen zu können galtes, einen Projektplan zu erstellen. Da der erste Abgabetermin für diesen Wettbewerb schon bald nach Projektbeginn anstand, blieb den Team nur kurze Zeit für die Erstellung dieses Plans. Mit der grossen Hilfe von Kevin Gaunt konnte ein für die Gruppe sehr zufriedenstellender Projektplan erstellt werden.

Unglücklicherweise entsprach dieser nicht den Vorstellung der Veranstalter des Imagine Cups, weshalb wir als Team nicht in die zweite Runde weiterkamen.

In der Semesterarbeit hatte das Team die Möglichkeit, erste Erfahrungen mit .NET und WPF zu sammeln. Dieses Wissen konnte für die Bachelorarbeit bestens eingesetzt werden.

Für die Bachelorarbeit wurden verschiedenste Analysen durchgeführt. Aufgrund des beschränkten Zeitrahmens war es erforderlich, diese zu priorisieren, was oftmals schwierig war. Trotz dieser Herausforderung ist es gelungen, viele neue Erkenntnisse zu schaffen und einen funktionstüchtigen Prototyp zu erstellen.

## VI.3 Persönliche Berichte

### VI.3.1 Lukas Elmer

Schon während der Studienarbeit im Herbstsemester 2011/2012 wurden wir von Markus Stolze auf eine spannende Bachelorarbeit mit sehr spannenden Technologien aufmerksam gemacht. Vor den Prüfungen im Winter erhielten wir bereits eine Xbox und einen Kinect Sensor, mit dem erste Versuche durchgeführt werden konnten.

Das Projekt war von Beginn an sehr offen ausgelegt. Einerseits bedeutete dies, dass sehr viel Spielraum für die Gestaltung des Projekts und das Setzen des Fokus möglich war. Andererseits bedeutete dies auch, dass kein vorgegebenes Ziel definiert war, nach dem gestrebt werden konnte. Speziell die Erkenntnis, dass in dieser beschränkten Zeit nicht alle Ziele erreicht würden, führte leider zeitweise zu einer stressigen Atmosphäre im Team. Zusätzlich baute sich eine Spannung auf, dass die einzelnen Teammitglieder eigentlich in verschiedene Richtungen streben wollten: manche eher in die technische Umsetzung eines mathematisch komplexen Problems, andere mehr in die Richtung, wie denn die Applikation am einfachsten bedient werden könnte, hätte man mehr Zeit für die Entwicklung. Andererseits war es schwierig, einen Plan zu erstellen und diesen dann umzusetzen, da viele Dinge sehr viel Spielraum ließen (z.B. Kinect Interaktion mit Gesten oder ohne Gesten oder viele verschiedene Bildschirmkonstellationen und Auflösungen). Und bei gewissen Punkten (z.B. die Fragestellung, ob die Größe des Bildschirms „gefühlt“ zu gross für einen tiefen Raum sei) konnte nicht einfach gemessen und beurteilt werden, sondern waren für jede Person subjektiv.

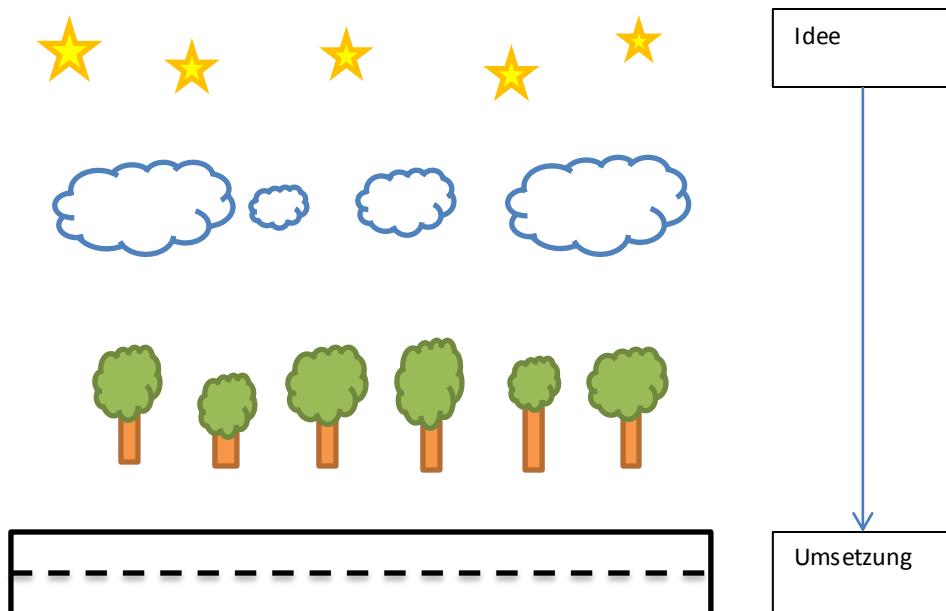
Neben den Problemen mit dem Projekt spielten auch andere persönliche Belastungen eine nicht zu vernachlässigende Rolle. Und auch im Studium mit den Vorlesungen im 6. Semester konnten sich die Teammitglieder nicht richtig anfreunden, da die ausgewählten Vertiefungsmodule die Erwartungen nicht richtig erfüllen konnten oder schon so tief ins Detail gingen, manchmal schon fast zu einfach oder dann nicht praxisnah genug waren. Und für einen Teil des Teams wirkte der Unterrichtsstil in der einen oder anderen Vorlesung gewisse Frustrationen aus. Eigentlich schade nach 5 Semestern mit z.T. sehr tollen Vorlesungen und Übungen. Vielleicht lag es auch daran, dass wir die „tollen“ Module bereits ein Jahr zuvor vorgeholt hatten. Oder vielleicht, dass drei Jahre Studium reichen und sich die Studenten schon prähungsgerecht auf einen Job freuen, um an vielen tollen Projekten zu arbeiten, statt die Schulbank zu drücken... ☺

Weiter war die Arbeit technisch sehr fordernd. Speziell dann, wenn der Prototyp bezüglich Code und Architektur den hohen Anforderungen der Assistenten Michael Gfeller und Silvan Gehrig gerecht werden musste und in diesem Bereich schon viel Erfahrung mit sich bringen. Und andererseits gab es da noch das Microsoft Kinect SDK, das erst im Februar 2012 erschienen war und noch nicht ausgereift war. Und dann gab es auch noch Grafikkarten, die doch noch nicht so gebaut sind, dass über 9 Monitore mit je Full HD-Auflösung flüssige Animationen oder Videos abgespielt werden konnten, ohne dass ein ruckeln festgestellt werden kann. In der Mitte des Projekts wurde sogar daran gezweifelt, ob WPF die richtige Technologie für die Videowall sei oder ob es doch besser gewesen wäre, mit DirectX oder sogar ein verteiltes System mit einzelnen PCs und Grafikkarten zu bauen. Es wurde sogar einmal diskutiert, ob die Abteilung Elektrotechnik an der HSR eine für das Problem besser geeignete Hardware herstellen könnte.

Ein weiterer Punkt war, dass wir die formalen und von der HSR geforderten Kriterien möglichst perfekt erfüllen wollten. Das hieß unter anderem für uns, dass alle Tests und alle Experimente, die durchgeführt wurden, auch dokumentiert werden mussten. Dies ist grundsätzlich sinnvoll, aber meiner Meinung nach verursachte dies in einem so experimentellen Projekt ein wenig zu viel Overhead, speziell da unser Team in diesem Punkt entschlossen war, sehr präzise zu arbeiten. Andererseits bieten die erarbeiteten Dokumente eine solide Basis, um das Projekt weiterzuentwickeln.

Auf jeden Fall gab es sehr viele Probleme zu überwinden, die sich speziell zu gewissen Zeitpunkten des Projekts zu einem riesig scheinenden Berg erhoben. Zum Teil war die Spitze dieses Berges nicht einmal mehr sichtbar. Deshalb denke ich, dass wir das als Team auch gut gelöst haben, obwohl zum Teil kritische Phasen im Projekt auftauchten. Denn schlussendlich brauchte es für diese Arbeit alle beteiligten Personen und hätte ohne die wertvollen Beiträge jeder einzelnen Person nicht funktioniert. Ganz speziell stand bei schwierigen (Team)Situationen Markus Stolze immer hinter uns und hat mit uns zusammen konstruktive, lösungsorientierte und teamorientierte Lösungen erarbeitet, stand uns als neutrale, erfahrene Ansprechperson immer zur Seite und nahm sich für das Projekt viel Zeit. Dies wissen wir ganz besonders zu schätzen.

Durch den Experten dieser Arbeit, Markus Flückiger, wurden wir bei den Meetings zu neuen Ideen inspiriert. Er hat uns auch gezeigt, dass ein Projekt, bei dem sehr viel in den Sternen steht (siehe Abbildung 143 - Von der Idee zur Umsetzung), es sehr schwierig sein kann, die „richtige“ Idee zu wählen, zu einem Konzept weiterzuverarbeiten und dann noch umzusetzen. Denn es ist sehr schwierig, Ideen zu beurteilen, die noch in den Sternen stehen. Und es kann passieren, dass es auch einmal nicht funktioniert, eine solche Idee von den Sternen herunter bis zur Straße zu bringen. Ich denke, dass sich das Projekt irgendwo auf der Höhe der Bäume befindet und in gewissen Bereichen vielleicht sogar noch in die Wolken herausragt.



**Abbildung 143 - Von der Idee zur Umsetzung**

Schlussendlich habe ich das Projekt einerseits als sehr interessant, andererseits auch als sehr herausfordernd empfunden, wie gesagt nicht nur wegen den technischen Aspekten. Es gab viele Herausforderungen, die wir als Team erfolgreich bestritten haben und ich bin mir sicher, dass wir alle viel fürs Leben gelernt haben. Obwohl es noch einige Dinge gibt, die ich wirklich gerne noch umgesetzt hätte, sehe ich ein, dass die Zeit dafür einfach nicht reichte. Wahrscheinlich werde ich mich weiter in meiner Freizeit und/oder im Master Studium mit dieser neuen Art der Interaktion beschäftigen und Projekte durchführen, die hoffentlich wieder so herausfordernd, spannend, knifflig, vielseitig und erfolgreich sein werden, wie es das Projekt HSR Videowall war. Ich würde es auch schön finden, wenn ich bei einem künftigen Besuch der HSR eine durch Kinect gesteuerte Videowall vorfinden würde, mit der sich die HSR als moderne Hochschule für Technik repräsentiert.

---

### **VI.3.2 Christina Heidt**

Wie bereits bei der Semesterarbeit, empfand ich auch in diesem Projekt die Betreuung durch Prof. Dr. Markus Stolze als sehr partnerschaftlich und motivierend. Im Projektverlauf kam es zu zwei Zeitpunkten zu Spannungen im Team. Ist erst einmal eine negative Stimmung aufgetreten, ist es schwer, diese wieder in eine positive zu wandeln. Daher wendeten wir uns in beiden Fällen an Markus Stolze. Durch seine Hilfe und klärende Gespräche konnten die internen Spannungen wieder gelöst werden. In beiden Fällen stellte Markus Stolze eine neutrale Instanz dar, welche sich nie gegen eine Seite stellte, sondern versuchte, beiden Seiten die Gründe des Verhaltens der anderen aufzuzeigen. Solche Probleme sind in keinem Projekt wünschenswert, anderseits konnte ich dadurch erfahren, wie solche Spannungen in zukünftigen Projekten vermieden oder gelöst werden könnten. Trotz diesen Problemen empfinde ich den Verlauf des Projekts und die Zusammenarbeit des Teams rückblickend als gut aber auch als kräftezehrend.

Wir entschieden uns während des Projekts Kontakt mit dem Experten Markus Flückiger aufzunehmen, um ihm das Projekt vorzustellen und allfällige Verbesserungsvorschläge einzuarbeiten. Durch seine langjährige Erfahrung als Usability Engineer konnte er uns viele Ideen und auch andere Sichtweisen aufzeigen, die sehr lehrreich und spannend waren.

Im Verlauf des Projekts waren viele Fragenstellungen abzuklären. Durch den begrenzten Zeitrahmen der Arbeit war es manchmal schwer eine gute Balance zwischen all diesen Fragen zu finden. Durch die Vielzahl an Fragestellungen verkleinerte sich der eigentliche Programmierteil auch wesentlich. Einerseits finde ich dies schade, anderseits hatten wir durch das Projekt die Möglichkeit uns auch mit völlig anderen Themen auseinanderzusetzen. Trotz des kleinen Zeitfensters für die Implementation konnte ein sinnvoller Prototyp der Applikation erarbeitet werden. Dies geschah unter anderem dadurch, dass wir schon im letzten Projekt mit WPF und C# gearbeitet hatten und eine Einarbeitungszeit wegfiel.

Zu Beginn standen vor allem die Nutzerstudien im Zentrum. Die Erarbeitung und Umsetzung einer Umfrage in einem solchen Massenstellte für mich Neuland dar, dessen Betreten ich höchst interessant fand. Auch die Verwendung von Kinect, welche völlig neue Interaktionskonzepte bietet, war spannend und brachte eine spielerische Komponente in das Projekt. Ich gehe auch davon aus, dass ich nicht mehr so schnell an einem Projekt arbeiten werde, welches Kinect verwendet. Die Hardware Evaluierung sowie die Erarbeitung des Plug-in Frameworks stellten spannende Herausforderungen dar.

Abschliessend ist zu sagen, dass ich das Projekt als überaus spannend aber auch als sehr fordernd empfand. Ich hoffe sehr, dass die Videowall durch die HSR angeschafft und eingesetzt wird. Ich bin fest davon überzeugt, dass eine solche Wall sehr eindrucksvoll wirken wird und neue, spannende Präsentationsmöglichkeiten bieten wird.

---

### **VI.3.3 Delia Treichler**

Ich habe das Projekt von Beginn bis Ende als spannend und herausfordernd empfunden. Am Anfang gab es so viele offene Fragen, die es zu klären galt: Was ist die optimale Monitorkonstellation für eine Videowall? Wie können Passanten angelockt werden, welcher Inhalt ist für sie interessant? Welche Videokarten bieten eine hohe Auflösung und eine gute Performance zugleich?

- Zu den Vorarbeiten gehörte das Visualisieren der verschiedenen Monitoranordnungen mittels Hellraumprojektionen.
- Im Gebäude 4 klebten wir Abstandsmarkierungen auf den Boden. Wir hielten uns in der Mittagszeit dort auf um die vorbeilaufenden Passanten zu zählen.
- Über 200 Studenten sprachen wir an, überreichten ihnen einen Fragebogen und ließen sie diesen ausfüllen.
- Im Arbeitszimmer bauten wir eine Test-Videowall auf und machten Grafikkartenexperimente mit aufwändigen Animationen.

Das alles war ganz neu für mich und daher spannend.

Diese Abklärungen waren ausgesprochen aufwändig, diese Zeit fehlte deshalb teilweise fürs Programmieren.

Die Arbeit zu dritt war in verschiedenen Hinsichten lehrreich. Unser Team besteht aus drei sehr unterschiedlichen Leuten mit verschiedenster Vorbildung. Jedes einzelne Teammitglied hatte eine andere Arbeitsweise und gewisse Vorlieben. Die Jobaufteilung und das Arbeitsverständnis differierten zwischen den Mitgliedern. Es gab einerseits begehrte, interessante Aufgaben und andererseits Pflichtaufgaben, die Bestandteil einer guten Bachelorarbeits sind. Die Verteilung dieser Arbeiten war eine Herausforderung und führte auch zu Spannungen im Team.

Aus klärenden Gesprächen ergab sich eine bessere Aufteilung der Arbeiten, so dass alle Teammitglieder möglichst viel profitieren konnten, aber auch das Projekt gut vorwärts kam.

Für das Gelingen des Projektes waren sowohl kreative Arbeit als auch viel Fleiss gefragt. Ich habe mich für die Koordination des Projektes eingesetzt. So behielten wir den Überblick. Die gerechte Aufteilung der anstehenden Aufgaben war schwierig. Sprachlich hat das Dokumentieren hohe Anforderungen an das Team gestellt. Das war viel Aufwand.

Das Projekt konnte ungemein vom grafischen Talent der Teamkollegin Christina Heidt und von den fundierten Programmierkenntnissen des Teamkollegen Lukas Elmer profitieren.

Gerne möchte ich Markus Flückiger für die vielen Denkanstöße in Richtung Usability und Markus Stolze für die partnerschaftliche Zusammenarbeit und seine Unterstützung als Betreuer in allen Belangen, sowohl fachlich wie auch menschlich, danken.

Es freut mich, im Team eine Applikation geschaffen zu haben, die andere Leute erstaunt, die noch unbekannt und neu ist und sich zeigen lässt.

Die Technik ist noch nicht ausgereift, um eine Applikation mit hoher Auflösung und flüssiger Animation zugleich über 9 Bildschirme laufen zu lassen. Das Feld ist weit und noch offen für spannende Forschungsarbeiten, die noch gemacht werden können. Es gibt noch viel zu tun und zu verbessern.

Ich finde, dass die Videowall für die HSR ein echter Blickfang wäre und die Innovativität der Hochschule demonstriert.

## VI.4 Aufwandsanalyse

Das Projekt lief über das gesamte Semester und zwei weiterführende Wochen und dauerte gesamthaft 17 Wochen. Für das Modul Bachelorarbeit Informatik werden 12 ETCS-Punkte pro Student vergeben. Pro ECTS-Punkt wird mit einem Aufwand von 30 Stunden gerechnet. Daher standen für die Durchführung des Projektes  $3 * 12 * 30 = 1080$  Stunden zur Verfügung. Dies ergab pro Sprint (SP 1 - SP 16) je 67.5 Stunden.

### VI.4.1 Sprints

Die Tabelle 34 - Aufwand Übersicht zeigt die geplanten und die tatsächlich benötigten Stunden für das Projekt. Es ist zudem ersichtlich, wie gross die Abweichung zwischen Aufwand und Schätzung ist.

Sprint	Geschätzter Aufwand in Stunden	Aufgewendete Zeit in Stunden	Abweichung in Stunden	Absolute Abweichung in Stunden
1	61	68.5	7.5	7.5
2	78	81.75	3.75	3.75
3	95	95.75	0.75	0.75
4	71.5	64.25	-7.25	7.25
5	61	70	9	9
6	51.5	56.25	4.75	4.75
7	103.5	118.25	14.75	14.75
8	66.5	67.25	0.75	0.75
9	53.75	53.75	0	0
10	76	72.5	-3.5	3.5
11	75	75.5	0.5	0.5
12	87.5	93.25	5.75	5.75
13	91	90.25	-0.75	0.75
14	102	100.5	-1.5	1.5
15	171.5	174.75	3.25	3.25
16	113.5	104.75	-8.75	8.75
<b>Total</b>	<b>1358.25</b>	<b>1387.25</b>	<b>29</b>	<b>72.5</b>

Tabelle 34 - Aufwand Übersicht

Abbildung 144 - Verlauf geschätzte und aufgewendete Zeit zeigt die gleichen Daten in einem Diagramm.

Die blaue Linie stellt die geplanten Stunden dar. In diesen geplanten Stunden ist Timeboxing eingerechnet. Timeboxen bezeichnet das Verschieben von Features, welche für den aktuellen Sprint geplant waren, aber nicht fertig entwickelt werden konnten, in den nächsten Sprint. Die rote Linie zeigt den tatsächlichen Aufwand. Die grüne Linie stellt den durchschnittlichen Wert pro Sprint dar, um gesamthaft auf die verlangten 1080 Stunden zu kommen. Die violette Linie stellt die Differenz zwischen Geplant und Aufwand dar, die türkise Linie zeigt die absolute Abweichung.

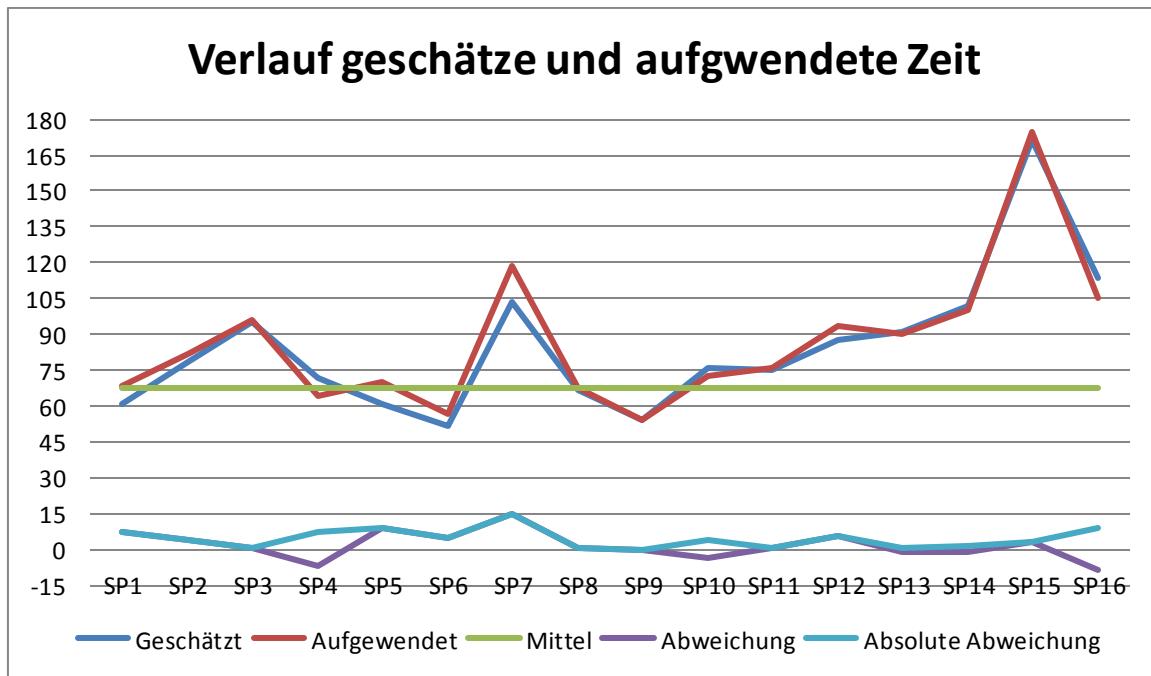


Abbildung 144 - Verlauf geschätzte und aufgewendete Zeit

Der Aufwand in Sprint 3 ist höher, da sich das Team kurzfristig dazu entschloss, beim Microsoft Imagine Cup mitzumachen und dazu einen Projektplan erstellen musste. Der Sprint 7 dauerte nicht eine sondern zwei Wochen, der Aufwand ist daher höher. Im Sprint 15 wurden die letzten Korrekturen und Verbesserungen umgesetzt wodurch der Aufwand dort ebenfalls höher ist.

Nachfolgend werden die einzelnen Sprints detailliert betrachtet und analysiert. Eine Übersicht über die in den einzelnen Sprints durchgeführten Arbeiten ist im Projektplan zu finden (V.2.2.2 Milestones / Sprints).

#### VI.4.1.1 Sprint 1

Die folgenden Tickets wurden im Sprint 1 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
642	Entwicklungsumgebung eingerichtet/installiert	6	6.5	20.02.2012	21.02.2012
643	Redmine, SVN und Sitzungsprotokolle sind eingerichtet	2	2.5	20.02.2012	20.02.2012
644	Dokumentvorlage erstellt	6	6	20.02.2012	27.02.2012
645	Meeting, 20.02.12	3	4.5	20.02.2012	20.02.2012
646	Logo erstellt	4	4	20.02.2012	27.02.2012
648	Brainstorming Grob-Interaktionskonzept durchgeführt	2	3	21.02.2012	24.02.2012
652	Testsetup evaluiert	5	5.5	21.02.2012	27.02.2012
657	Sprint 1 und Sprint 15/16 geplant	3	4.5	21.02.2012	24.02.2012
660	Risiken identifiziert	1	1	21.02.2012	24.02.2012
661	Projektplan erstellt	2	2	21.02.2012	27.02.2012
666	Studium Technologien durchgeführt	24	23.5	21.02.2012	24.02.2012
667	Meeting, 24.02.12	3	5.5	24.02.2012	24.02.2012
<b>Total</b>		<b>65</b>	<b>68.5</b>		

Tabelle 35 - Tickets Sprint 1

## VI.4.1.2 Sprint 2

Die folgenden Tickets wurden im Sprint 2 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
647	Windows Kinect SDK Framework ausprobiert	2	2	27.02.2012	05.03.2012
649	Personen in der Mensa beobachtet, ausgewertet und dokumentiert	8	10.75	27.02.2012	05.03.2012
653	Mini Prototyp erstellt	12	12.5	27.02.2012	05.03.2012
658	Benutzer Befragung durchgeführt und dokumentiert	20	20	27.02.2012	05.03.2012
668	Meeting, 02.03.12	3	5.25	02.03.2012	02.03.2012
669	Fragebogen erstellt	6	5.75	27.02.2012	05.03.2012
672	Risikomanagement nachgeführt	1	1.25	27.02.2012	05.03.2012
703	Grundriss ausgemessen und aufgezeichnet	10	10.5	27.02.2012	05.03.2012
716	Testsetup evaluiert	10	5.25	27.02.2012	02.03.2012
717	Kinect record Möglichkeit gefunden und entwickelt	6	8.5	05.03.2012	05.03.2012
<b>Total</b>		<b>78</b>	<b>81.75</b>		

Tabelle 36 - Tickets Sprint 2

## VI.4.1.3 Sprint 3

Die folgenden Tickets wurden im Sprint 3 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
659	Vision erstellt	10	12	05.03.2012	09.03.2012
656	Bachelor Poster PDFs organisiert	0.5	0.25	05.03.2012	09.03.2012
670	Meeting, 09.03.12	3	1.5	09.03.2012	09.03.2012
675	Risikomanagement nachgeführt	0.5	1	05.03.2012	09.03.2012
719	Projekt Plan Imagine Cup erstellt	40	39	05.03.2012	06.03.2012
720	Meeting, 05.03.12, Imagine Cup	4.5	5	05.03.2012	05.03.2012
721	Testsetup dokumentiert	8	7.75	05.03.2012	09.03.2012
722	Benutzer Befragung durchgeführt und dokumentiert	4	5	05.03.2012	12.03.2012
723	Personen in der Mensa beobachtet, ausgewertet und dokumentiert	3	1.75	05.03.2012	12.03.2012
724	Kinect Framework mithilfe Nutzwertanalyse ausgewählt	8	8.5	05.03.2012	12.03.2012
725	Meeting, 06.03.12, Imagine Cup	9	9.5	06.03.2012	06.03.2012
731	Sprint 04 geplant	4.5	4.5	09.03.2012	09.03.2012
<b>Total</b>		<b>95</b>	<b>95.75</b>		

Tabelle 37 - Tickets Sprints 3

#### VI.4.1.4 Sprint 4

Die folgenden Tickets wurden im Sprint 4 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
650	Personas erstellt	8	7.75	12.03.2012	19.03.2012
651	Szenarien erstellt	9	9.25	12.03.2012	19.03.2012
655	Formativer empirischer Test erstellt, Anforderungen an Gesten definiert	8	7.5	12.03.2012	19.03.2012
671	Meeting, 16.03.12	3	2	16.03.2012	16.03.2012
678	Risikomanagement nachgeführt	0.5	0.5	12.03.2012	19.03.2012
718	Kinect replay Möglichkeit gefunden und entwickelt	9	9	14.03.2012	21.03.2012
726	TFS Server Installation abgeschlossen	7	6.5	12.03.2012	19.03.2012
728	Backlog erstellt	7	6.75	12.03.2012	19.03.2012
730	Skeleton-Aufnahmen im Gebäude 4 gemacht	8	4	12.03.2012	19.03.2012
732	Test Videowall aufgebaut	12	11	13.03.2012	19.03.2012
<b>Total</b>		<b>71.5</b>	<b>64.25</b>		

Tabelle 38 - Tickets Sprint 4

#### VI.4.1.5 Sprint 5

Die folgenden Tickets wurden im Sprint 5 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
673	Meeting, 23.03.12	3	3	23.03.2012	23.03.2012
681	Risikomanagement nachgeführt	0.5	1	19.03.2012	26.03.2012
733	Formativer empirischer Test erstellt, Anforderungen an Gesten definiert	13	22	19.03.2012	26.03.2012
735	Codereview durchgeführt und im Quellcode dokumentiert	4	3	19.03.2012	26.03.2012
736	Vorstudie Dokument überarbeitet	6	6.5	19.03.2012	26.03.2012
737	Skeleton-Aufnahmen im Gebäude 4 gemacht	4	5	19.03.2012	26.03.2012
738	Sitzung mit Markus Flückiger abgemacht	1	1	20.03.2012	26.03.2012
739	WPF Applikation auf Test Hardware getestet und dokumentiert	4	4	20.03.2012	26.03.2012
740	Sprint 5 geplant	6	6	20.03.2012	26.03.2012
741	Teamfördernde Massnahmen	7.5	7.5	21.03.2012	21.03.2012
742	Grobarchitektur erarbeitet und implementiert	12	11	22.03.2012	26.03.2012
<b>Total</b>		<b>61</b>	<b>70</b>		

Tabelle 39 - Tickets Sprint 5

#### VI.4.1.6 Sprint 6

Die folgenden Tickets wurden im Sprint 6 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
674	Meeting, 02.04.12	3	4.5	30.03.2012	30.03.2012
684	Risikomanagement nachgeführt	0.5	0.5	26.03.2012	02.04.2012
727	Formativer empirischer Test durchgeführt	12	13.5	26.03.2012	02.04.2012
729	Formativer empirischer Test dokumentiert	8	6.5	27.03.2012	02.04.2012
744	Tieferen Auflösungen getestet und dokumentiert	8	10	26.03.2012	02.04.2012
745	Architektur mit Silvan besprochen	6	6.75	29.03.2012	29.03.2012
746	Architekturprototyp erstellt: Handtracking	7	7.5	26.03.2012	02.04.2012
747	Architekturprototyp erstellt: Menu mit Mittagsmenu und Poster	4	4	26.03.2012	
750	Sprint 6 geplant	3	3	26.03.2012	02.04.2012
<b>Total</b>		<b>51.5</b>	<b>56.25</b>		

Tabelle 40 - Tickets Sprint 6

#### VI.4.1.7 Sprint 7

Die folgenden Tickets wurden im Sprint 7 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
676	Meeting, 17.04.12	3	7.25	13.04.2012	13.04.2012
687	Risikomanagement nachgeführt	0.5	0.5	06.04.2012	16.04.2012
734	Backlog erstellt, priorisiert	4.5	7.5	09.04.2012	16.04.2012
748	Architekturprototyp erstellt: Navigation zwischen Poster	1	1	02.04.2012	16.04.2012
749	Architekturprototyp erstellt: Kinect Skelett	6	7.5	02.04.2012	16.04.2012
751	Architekturprototyp erstellt: Menu mit Mittagsmenu und Poster (Bild)	8	8	03.04.2012	16.04.2012
752	Architekturprototyp erstellt: PDF zu Bildern konvertieren	6	6	03.04.2012	16.04.2012
754	Architekturentscheide sind dokumentiert	4	6.5	03.04.2012	16.04.2012
755	Architekturprototyp erstellt: Handtracking	7	9.5	03.04.2012	16.04.2012
756	Vorstudie gemäss Sitzung angepasst	1	1.5	03.04.2012	16.04.2012
757	Poster L sind organisiert	1	1	03.04.2012	16.04.2012
758	Architektur mit Silvan besprochen, Zugangsdaten Server eingerichtet und an Silvan gesendet	3	1.5	03.04.2012	16.04.2012
759	Sprint 7 geplant	3	3	03.04.2012	16.04.2012

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
761	Architekturprototyp erstellt: Projekte vereinigt	8	10	03.04.2012	16.04.2012
762	Architekturprototyp erstellt: Mit Kinect "klicken"	16	16	06.04.2012	10.04.2012
763	Meeting. 12.04.12	15	16	12.04.2012	12.04.2012
764	Web Architektur von MS studiert haben	8	8	11.04.2012	16.04.2012
765	Tiers-Diagramm erstellt und beschrieben	8	7	16.04.2012	16.04.2012
851	Systemtests durchgeführt und dokumentiert	0.5	0.5	16.04.2012	16.04.2012
<b>Total</b>		<b>103.5</b>	<b>118.25</b>		

Tabelle 41 - Tickets Sprint 7

#### VI.4.1.8 Sprint 8

Die folgenden Tickets wurden im Sprint 8 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
677	Meeting, 23.04.12	6	6.5	23.04.2012	23.04.2012
689	Risikomanagement nachgeführt	0.5	0.5	16.04.2012	23.04.2012
753	Bilderprototyp auf Videowall getestet und dokumentiert	4	5	16.04.2012	23.04.2012
760	Refactoring durchgeführt	8	6.25	16.04.2012	23.04.2012
767	Sprint 08 geplant	4.5	6.5	16.04.2012	23.04.2012
769	Hand Cursor schön dargestellt	3	3	17.04.2012	23.04.2012
770	Begründung für Poster festgehalten	6	6	17.04.2012	23.04.2012
771	DirectX auf Videowall ist getestet	5	5	17.04.2012	23.04.2012
772	Usability Test mit Architekturprotoyp organisiert	2	2	17.04.2012	23.04.2012
773	Usability Test mit Architekturprotoyp durchgeführt und dokumentiert	8	6.75	18.04.2012	18.04.2012
775	Sofortiges Erfolgserlebnis dokumentiert	2	1.75	17.04.2012	23.04.2012
776	Poster E sind organisiert	1	1	16.04.2012	23.04.2012
777	Code Review durchgeführt	6	6	16.04.2012	23.04.2012
779	Skelett schön dargestellt	6	6	19.04.2012	23.04.2012
780	Backlog: Definition of Done für aktuelle User Stories erfasst	2	2.5	19.04.2012	23.04.2012
781	Meeting 26.04.12 vorbereitet	1	1	23.04.2012	23.04.2012
852	Systemtests durchgeführt und dokumentiert	0.5	0.5	23.04.2012	23.04.2012
870	Hand Cursor ruckelt weniger 1	1	1	21.05.2012	23.04.2012
<b>Total</b>		<b>66.5</b>	<b>67.25</b>		

Tabelle 42 - Tickets Sprint 8

### VI.4.1.9 Sprint 9

Die folgenden Tickets wurden im Sprint 9 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
679	Meeting, 27.04.12	3	4.25	27.04.2012	27.04.2012
707	Risikomanagement nachgeführt	0.5	0.5	23.04.2012	30.04.2012
766	Web Architektur von MS studiert haben	3	3	23.04.2012	30.04.2012
774	Ideen gesammelt wie Personen von Videowall angezogen werden	6	6	23.04.2012	30.04.2012
778	Dokument Vorstudie: Varianz bei Umfrage eingetragen	9	9	23.04.2012	30.04.2012
782	Sprint 9 geplant	4.5	4.5	23.04.2012	30.04.2012
785	Applikation ist mit linker Hand bedienbar	4	4	23.04.2012	30.04.2012
786	WPF Applikation mit Video erstellt/recherchiert	1.5	0.5	23.04.2012	30.04.2012
787	Backlog ist aktuell	1	1	23.04.2012	30.04.2012
788	Meeting 26.04.12	4.5	4.75	23.04.2012	30.04.2012
789	SVN Base Architecture verschieben, Tag erstellt	1	1	23.04.2012	30.04.2012
790	SVN Tag erstellt	1	0.75	23.04.2012	30.04.2012
791	BA/Master-Vorstudie, Wissen ausgetauscht	2.25	2.25	24.04.2012	30.04.2012
792	Einführung ins Projekt für Silvan	1	1	24.04.2012	30.04.2012
794	MEF (Managed Extensibility Framework) studiert haben	10	9.75	25.04.2012	30.04.2012
795	Projektmanagement / Administratives	1	1	26.04.2012	30.04.2012
853	Systemtests durchgeführt und dokumentiert	0.5	0.5	27.04.2012	27.04.2012
<b>Total</b>		<b>53.75</b>	<b>53.75</b>		

Tabelle 43 - Tickets Sprint 9

### VI.4.1.10 Sprint 10

Die folgenden Tickets wurden im Sprint 10 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
798	Plug-in Möglichkeit entwickelt	8	8	30.04.2012	07.05.2012
680	Code Review vom 03.05.12 durchgeführt	6	5.5	03.05.2012	03.05.2012
708	Risikomanagement nachgeführt	0.5	0.5	30.04.2012	07.05.2012
783	Refactoring durchgeführt	1	1	30.04.2012	07.05.2012
793	DirectX auf Videowall ist dokumentiert	4	4.25	30.04.2012	07.05.2012
796	Ideen gesammelt und dokumentiert wie Personen von Videowall angezogen werden	14	14	01.05.2012	04.05.2012
797	Mitsubishi Wall angesehen und dokumentiert	20	17.75	02.05.2012	07.05.2012

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
803	Sprint10 geplant	4	3.75	30.04.2012	30.04.2012
804	Web Architektur von MS studiert haben	1	1	30.04.2012	07.05.2012
805	Anpassungen gemäss Besprechung mit Herrn Heinzmann durchgeführt	5	4.75	04.05.2012	07.05.2012
806	Code Review vom 03.05.12 dokumentiert	3	2.5	04.05.2012	07.05.2012
807	Anpassung bezüglich Code Review vom 03.05.12 gemacht	1.5	1.5	04.05.2012	07.05.2012
808	Gesamtplanung anhand Kriterienliste überprüft und geplant	6	6	07.05.2012	07.05.2012
831	SVN Tag erstellt	1.5	1.5	07.05.2012	07.05.2012
854	Systemtests durchgeführt und dokumentiert	0.5	0.5	07.05.2012	07.05.2012
<b>Total</b>		<b>76</b>	<b>72.5</b>		

Tabelle 44 - Tickets Sprint 10

#### VI.4.1.11 Sprint 11

Die folgenden Tickets wurden im Sprint 11 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
682	Meeting, 11.05.12	5	5	11.05.2012	11.05.2012
709	Risikomanagement nachgeführt	0.5	0.5	07.05.2012	14.05.2012
784	Demomodus (Verfolgung von Passanten) Kraftfeld besprochen und dokumentiert	7	7.25	07.05.2012	14.05.2012
799	Bild der Hand ist auf die rechte bzw. linke Hand abgestimmt	2	2.5	07.05.2012	14.05.2012
800	Mittagsmenu App in Plugin umgewandelt	4	5.5	07.05.2012	14.05.2012
802	Poster App in Plugin umgewandelt	6	5	07.05.2012	14.05.2012
832	Sprint11 geplant	6	7.5	07.05.2012	08.05.2012
833	Demomodus: Vom Demomodus wird in den Interaktionsmodus gewechselt	10	8.5	08.05.2012	14.05.2012
834	Demomodus: Vom Interaktionsmodus wird in den Demomodus gewechselt	5	5.5	08.05.2012	14.05.2012
837	Demomodus: externes Design erstellt	4	4.25	08.05.2012	14.05.2012
838	Systemtests durchgeführt und dokumentiert	4	2.5	08.05.2012	14.05.2012
839	Refactoring durchgeführt	7	8.5	07.05.2012	14.05.2012
840	Anpassung bezüglich Code Review vom 03.05.12 gemacht	2	2	07.05.2012	14.05.2012
841	Aufwändige Anpassung bezüglich Code Review vom 03.05.12 gemacht	8	6.5	10.05.2012	14.05.2012

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
842	Meeting, 14.05.12	4.5	4.5	14.05.2012	14.05.2012
<b>Total</b>		<b>75</b>	<b>75.5</b>		

Tabelle 45 - Tickets Sprint 11

#### VI.4.1.12 Sprint 12

Die folgenden Tickets wurden im Sprint 12 abgearbeitet:

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
683	Meeting, 22.05.12	6	5.75	22.05.2012	22.05.2012
698	Usability Tests durchgeführt und protokolliert	4	4.75	14.05.2012	21.05.2012
710	Risikomanagement nachgeführt	0.5	0.5	14.05.2012	21.05.2012
801	Plugin Schnittstelle definiert und dokumentiert	8	10.5	14.05.2012	21.05.2012
810	TODOs in Dokumenten abgearbeitet	5	4.75	14.05.2012	21.05.2012
816	Diskussion Accessibility dokumentiert	3	2.75	14.05.2012	21.05.2012
835	Demomodus: Apps werden automatisch gewechselt	2	1.5	14.05.2012	21.05.2012
836	Demomodus: Demotext zu aktiver App wird angezeigt	1	1.25	14.05.2012	21.05.2012
843	Demomodus: Zustandsdiagramm erstellt und dokumentiert	5	4.75	14.05.2012	21.05.2012
844	Oliver Rehmann wegen Posterstand am 15.6.12 kontaktiert	1	1.25	14.05.2012	21.05.2012
845	2x4 Monitore-Setup mit Hellaumprojektor getestet und dokumentiert	1.5	2.25	15.05.2012	21.05.2012
846	2x4 Monitore-Setup mit Test-Wall getestet und dokumentiert	3	3.25	15.05.2012	21.05.2012
847	Notifier Problem gelöst/umgangen und dokumentiert	1.5	1.5	22.05.2012	22.05.2012
848	Gewichtung bei Nutzwertanalyse begründet	2	3.25	15.05.2012	21.05.2012
849	Refactoring durchgeführt	10	11.5	15.05.2012	21.05.2012
850	Sprint 12 geplant	3	3	15.05.2012	21.05.2012
855	Deployment Entwickler PC möglich	2.5	1	15.05.2012	21.05.2012
856	Das Mittagsmenü wird angezeigt	4	4	18.05.2012	18.05.2012
857	Leichte Usability Test Korrekturen umgesetzt und dokumentiert	5	4.75	18.05.2012	21.05.2012
858	Navigation mit schönen "Tabs" dargestellt	6	6	23.05.2012	23.05.2012
867	Systemtests durchgeführt und dokumentiert	0.5	1	22.05.2012	21.05.2012
871	Externes Design festgelegt und validiert	3	4	22.05.2012	21.05.2012

<b>872</b>	Mittagsmenu App automatisch aktualisiert	4	4	18.05.2012	18.05.2012
<b>873</b>	Abschluss der Arbeit geplant	6	6	22.05.2012	21.05.2012
<b>Total</b>		<b>87.5</b>	<b>93.25</b>		

Tabelle 46 - Tickets Sprint 12

#### VI.4.1.13 Sprint 13

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
<b>685</b>	Meeting, 25.05.12	3	3.25	25.05.2012	25.05.2012
<b>702</b>	Abstract geschrieben	6	5.75	21.05.2012	28.05.2012
<b>711</b>	Risikomanagement nachgeführt	0.5	0.5	21.05.2012	28.05.2012
<b>715</b>	Poster erarbeitet	6.5	6.5	21.05.2012	28.05.2012
<b>811</b>	Ein Buch, ein ACM und IEEE Paper zitiert	6	6	21.05.2012	28.05.2012
<b>814</b>	Konkurrenz analysiert und dokumentiert	6	8.5	21.05.2012	28.05.2012
<b>817</b>	Domain Model erstellt und dokumentiert	5	4.25	21.05.2012	28.05.2012
<b>823</b>	Unit-Tests erstellt	6	6	21.05.2012	28.05.2012
<b>825</b>	Diskussion Notwendigkeit statistische Analyse dokumentiert	1	1	21.05.2012	28.05.2012
<b>829</b>	Coding Standards dokumentiert und eingehalten	3	1.5	21.05.2012	28.05.2012
<b>859</b>	Sprint 13 geplant	6	6	22.05.2012	28.05.2012
<b>860</b>	Refactoring durchgeführt	2.5	2.5	22.05.2012	28.05.2012
<b>861</b>	Stakeholderanalyse erstellt	4	4.75	22.05.2012	28.05.2012
<b>862</b>	Domain Analyse: Daten beschrieben	2	0.75	22.05.2012	28.05.2012
<b>863</b>	Backlog ist aktuell	1.5	0.25	22.05.2012	28.05.2012
<b>864</b>	Korrekturen Markus Stolze umgesetzt	3	4	22.05.2012	28.05.2012
<b>865</b>	Usability Tests durchgeführt und dokumentiert	2.5	2.75	22.05.2012	28.05.2012
<b>868</b>	Systemtests durchgeführt und dokumentiert	0.5	0.5	22.05.2012	28.05.2012
<b>869</b>	Notifier Problem gelöst	3	2.5	21.05.2012	28.05.2012
<b>874</b>	Administration der Videowall Inhalte definiert	10	9.5	24.05.2012	28.05.2012
<b>875</b>	Lesbarkeit der L-Poster überprüft und dokumentiert	3	3.25	24.05.2012	28.05.2012
<b>877</b>	Verkleinertes Video auf Videowall abspielbar und dokumentiert	6	8	24.05.2012	28.05.2012
<b>881</b>	Prozentuale Poster Lesbarkeit analysiert	4	2.25	25.05.2012	28.05.2012
<b>Total</b>		<b>91</b>	<b>90.25</b>		

Tabelle 47 - Tickets Sprint 13

---

#### VI.4.1.14 Sprint 14

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
<b>686</b>	Meeting, 01.06.12	14	13.5	01.06.2012	01.06.2012
<b>693</b>	Extended Management Summary geschrieben	7	7.25	28.05.2012	04.06.2012
<b>712</b>	Risikomanagement nachgeführt	0.5	0.5	28.05.2012	04.06.2012
<b>812</b>	Nicht-funktionale Anforderungen dokumentiert	2	1.5	28.05.2012	04.06.2012
<b>813</b>	Funktionale Anforderungen dokumentiert	2	2	28.05.2012	04.06.2012
<b>818</b>	User Environment Diagram oder Screen Map für Anwendungen mit mehreren Screens erstellt und dokumentiert	6	2.5	28.05.2012	04.06.2012
<b>821</b>	Architektur ist beschrieben	11	11	28.05.2012	04.06.2012
<b>827</b>	CHM Files generiert	5	5.25	28.05.2012	04.06.2012
<b>879</b>	Externes Design festgelegt und validiert	9	9	28.05.2012	04.06.2012
<b>882</b>	Aufgabenstellung gelesen und sichergestellt, dass alles dokumentiert	3	4	28.05.2012	04.06.2012
<b>883</b>	Poster erarbeitet	3	3	28.05.2012	04.06.2012
<b>884</b>	Unit-Tests erstellt	2	1	28.05.2012	04.06.2012
<b>885</b>	Miniapps dokumentiert	1.5	1.75	29.05.2012	04.06.2012
<b>886</b>	Sprint 14 geplant	3	2.25	29.05.2012	04.06.2012
<b>888</b>	Abstract geschrieben und abgegeben	3	3	28.05.2012	04.06.2012
<b>889</b>	Refactoring durchgeführt	30	33	28.05.2012	04.06.2012
<b>Total</b>		<b>102</b>	<b>100.5</b>		

Tabelle 48 - Tickets Sprint 14

---

#### VI.4.1.15 Sprint 15

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
<b>688</b>	Meeting, 06.06.12	7	6.75	08.06.2012	08.06.2012
<b>690</b>	Video und Wiki Seite erstellt	12	12	04.06.2012	11.06.2012
<b>692</b>	Persönlicher Bericht geschrieben	2	2.25	04.06.2012	11.06.2012
<b>694</b>	Installationsanleitung geschrieben	4	4	04.06.2012	11.06.2012
<b>696</b>	Code dokumentiert	3	5.5	04.06.2012	11.06.2012
<b>697</b>	Weiterentwicklung dokumentiert	3	2.25	04.06.2012	11.06.2012
<b>699</b>	Tools sind beschrieben	1	1	04.06.2012	11.06.2012
<b>700</b>	Einleitung Technischer Bericht geschrieben	3	1.5	04.06.2012	11.06.2012
<b>701</b>	Allgemeine Korrekturen, kleine Anpassungen	12	12	04.06.2012	11.06.2012
<b>815</b>	Design Constraints dokumentiert	1.5	1	04.06.2012	11.06.2012
<b>819</b>	GUI Design Entscheide	1	2.75	04.06.2012	11.06.2012

	dokumentiert					
820	GUI Guidelines dokumentiert	2	3	04.06.2012	11.06.2012	
824	Tests dokumentiert	4.5	4.5	04.06.2012	11.06.2012	
826	Anleitung für Entwickler dokumentiert	2	2	04.06.2012	11.06.2012	
828	Code Qualität dokumentiert	5	3.75	04.06.2012	11.06.2012	
830	Warnings und Coding Issues dokumentiert und evt. bereinigt	3	4	04.06.2012	11.06.2012	
887	Konkurrenzanalyse überarbeitet	12	12.5	04.06.2012	11.06.2012	
890	Fragen zur Videowall beantwortet	3	2.5	04.06.2012	11.06.2012	
892	Poster erarbeitet	1.5	1.5	04.06.2012	11.06.2012	
893	Abstract geschrieben und abgegeben	2	2	04.06.2012	11.06.2012	
894	Sprint 15 geplant	3	1.5	04.06.2012	11.06.2012	
895	CHM Files generiert	1	1	04.06.2012	11.06.2012	
896	Architektur ist beschrieben	20	20.5	04.06.2012	11.06.2012	
897	Externes Design dokumentiert	1	1.5	04.06.2012	11.06.2012	
898	Extended Management Summary geschrieben und mit Bildern versehen	3	2.5	04.06.2012	11.06.2012	
899	Unit-Tests erstellt	12	12.5	04.06.2012	11.06.2012	
900	Domain Analyse: Content Prozess zu Domain Model erstellt und dokumentiert	3	1.5	04.06.2012	11.06.2012	
901	CI/CD mit HSR abgeklärt	2	6	04.06.2012	11.06.2012	
902	Offerten angefordert	2	1.5	04.06.2012	11.06.2012	
905	Dokument-Korrekturen von Markus übernommen	8	8	04.06.2012	11.06.2012	
906	Stabilitätstest durchgeführt und dokumentiert	5	6	07.06.2012	11.06.2012	
907	Codereview 3 durchgeführt. dokumentiert und Anpassungen implementiert	16	15.5	07.06.2012	11.06.2012	
908	Refactoring durchgeführt	5	5	07.06.2012	11.06.2012	
909	Plug-in Framework Bild erstellt	6	5	07.06.2012	11.06.2012	
<b>Total</b>		<b>171.5</b>	<b>174.75</b>			

Tabelle 49 - Tickets Sprint 15

#### VI.4.1.16 Sprint 16

#	Thema	Geschätzter Aufwand	Aufgewendete Zeit	Beginn	Abgabedatum
695	Dokumente zusammenfügen. PDF generieren	20	14.5	11.06.2012	15.06.2012
809	Aufwand dokumentiert (Piechart)	3	7	11.06.2012	15.06.2012
880	Meeting, 13.06.12	6	6	13.06.2012	15.06.2012
910	Poster. Prezi. Flip Video in Applikation eingebunden	5	5	11.06.2012	15.06.2012
911	Tool geschrieben. um die PDF Poster zu Bildern zu konvertieren	1	1	11.06.2012	15.06.2012

<b>916</b>	Persönlicher Bericht geschrieben	10	7.5	11.06.2012	15.06.2012
<b>920</b>	Korrekturen und kleine Änderungen an Dokumenten durchgeführt	16	17.25	12.06.2012	15.06.2012
<b>704</b>	Lizenzvereinbarung unterschrieben	2	1	11.06.2012	15.06.2012
<b>691</b>	CD gebrannt und abgegeben	3	3	11.06.2012	15.06.2012
<b>706</b>	Erklärung eigenständige Arbeit unterschrieben	1	1	11.06.2012	15.06.2012
<b>904</b>	Test Videowall abgebaut	5	0.5	12.06.2012	12.06.2012
<b>912</b>	Tests dokumentiert	1	1	11.06.2012	15.06.2012
<b>913</b>	Video und Wiki Seite erstellt	4	4	11.06.2012	15.06.2012
<b>914</b>	Poster erarbeitet	3.5	2	11.06.2012	15.06.2012
<b>915</b>	Abstract geschrieben und abgegeben	1	1	11.06.2012	15.06.2012
<b>917</b>	Codereview 3 Feedback Michael dokumentiert	2	3	11.06.2012	15.06.2012
<b>918</b>	Bericht und Poster sind ausgedruckt	6	6	12.06.2012	15.06.2012
<b>919</b>	HSR-Forum-Stand aufgebaut. betreut und wieder abgebaut	24	24	12.06.2012	15.06.2012
<b>Total</b>		<b>113.5</b>	<b>104.75</b>		

**Tabelle 50 - Tickets Sprint 16**

## VI.4.2 Personenaufwand

Der Arbeitsaufwand pro Person ist, wie in Tabelle 51 - Personenaufwand Übersicht ersichtlich, sehr ausgeglichen:

Mitglied	Aufgewendete Zeit
Lukas Elmer	468.75
Christina Heidt	450.75
Delia Treichler	467.75
<b>Total</b>	<b>1387.25</b>

Tabelle 51 - Personenaufwand Übersicht

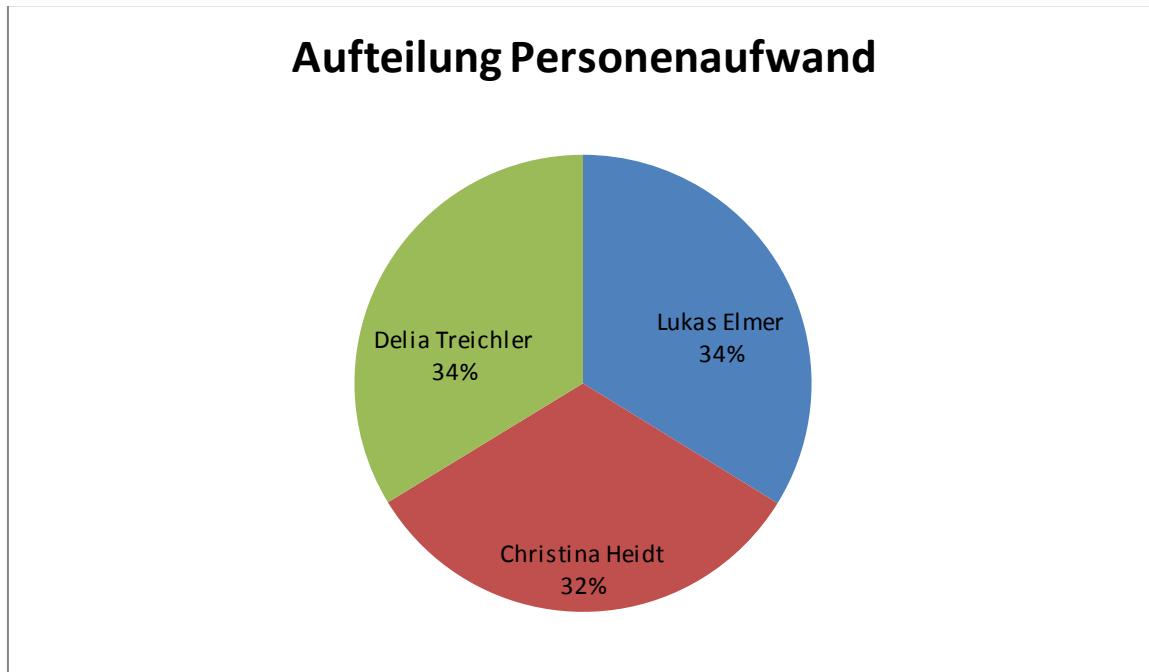


Abbildung 145 - Aufteilung Personenaufwand

Im Abbildung 146 - Personenaufwand pro Sprint ist der Verlauf des Aufwands pro Person über alle Sprints ersichtlich. Hierbei fällt deutlich auf, dass die aufgewendete Zeit in den jeweiligen Sprints sehr unterschiedlich ist. Zu Beginn des Projektes einigte sich das Team auf eine Sprintlänge von etwa 3 Tagen pro Woche ( $3 \times 22.5$  Stunden). Später wurde festgestellt, dass die Sprint-Länge mit 3 Tagen zu kurz ist. Daher musste oft Timeboxing durchgeführt werden oder der Task wurde komplett in den neuen Sprint verschoben oder im aktuellen Sprint wurde mehr gearbeitet, um den Task beenden zu können.

Der hohe Aufwand in den Sprints 7 und 15 wird im Kapitel VI.4.1 Sprints erläutert.

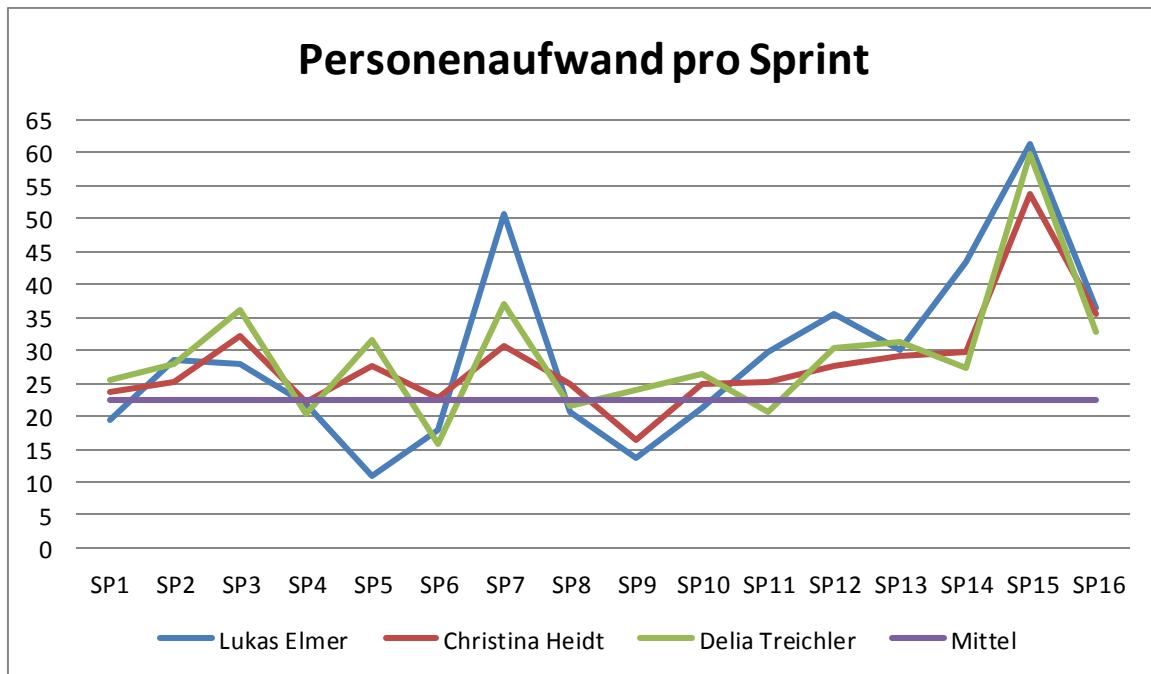


Abbildung 146 - Personenaufwand pro Sprint

Wird der Personenaufwand pro Woche ausgewertet, so ergeben sich weniger starke Schwankungen. Dies bestätigt, dass die kurze Sprintdauer zu Timeboxing oder Verschieben des Tasks in den nächsten Sprint führte und daher sehr unausgeglichene Sprints entstanden.

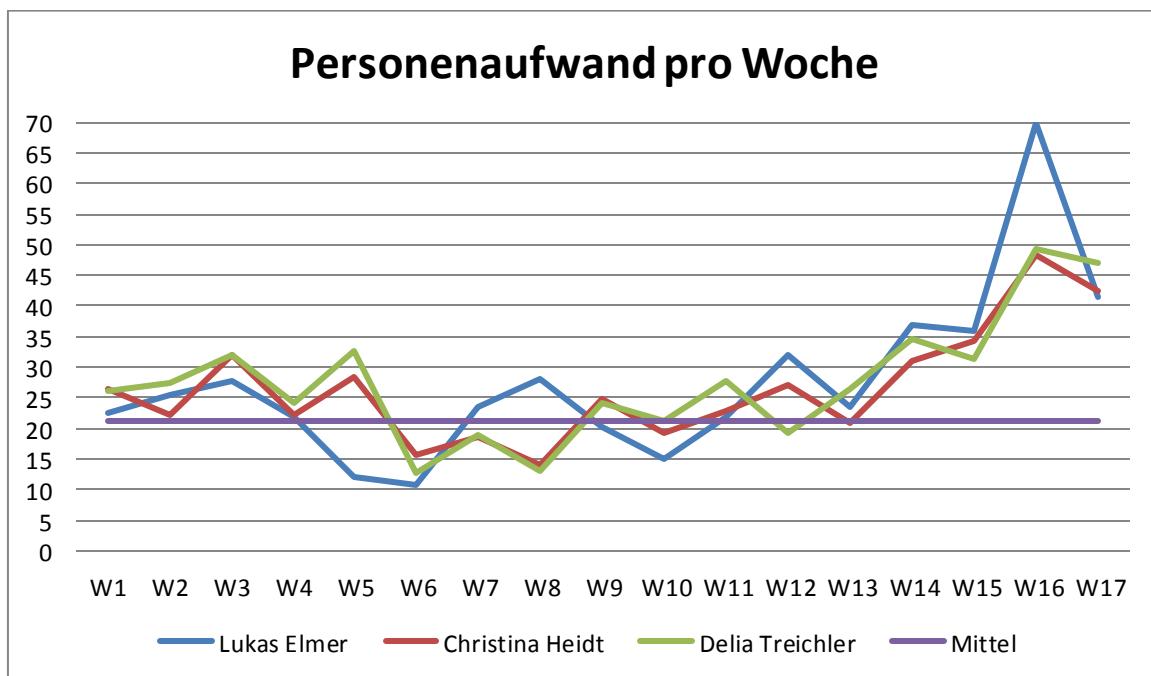


Abbildung 147 - Personenaufwand pro Woche

#### VI.4.3 Tätigkeiten

Wie im nachfolgenden Diagramm ersichtlich ist, wurde die meiste Zeit verwendet, um Ergebnisse zu dokumentieren. Dies kommt daher, da in diesem Projekt vielfältige Abklärungen zu tätigen waren und diese alle festgehalten werden mussten. Folglich macht auch der Implementationsteil nur 14% der gesamten Arbeit aus. Um die Qualität des Codes und der Dokumentation hoch zu halten, wurden 9% der gesamten Zeit in die Qualitätssicherung investiert.

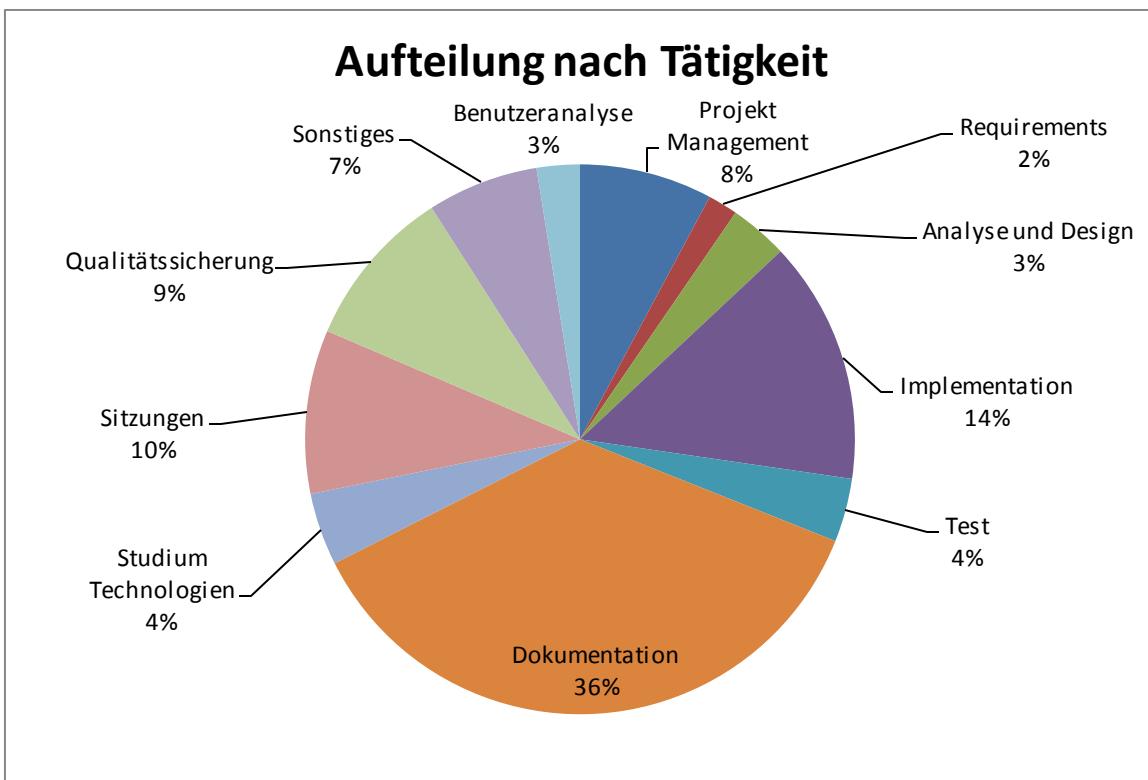


Abbildung 148 - Aufteilung nach Tätigkeit

Der Abbildung 149 - Aktivitäten nach Personen gruppiert kann entnommen werden, dass Lukas Elmer wesentlich mehr implementierte als die anderen Teammitglieder. Das ist darum so, weil das Team beschloss, dass das abschliessende Refactoring aus einer Hand gemacht werden soll, damit die Applikation in einem Fluss und korrekt strukturiert ist und somit einfach übernommen werden kann. Dafür verkleinerte sich sein Dokumentationsteil dementsprechend.

Unter Sonstiges fallen Tätigkeiten wie zum Beispiel der Aufbau der Videowall - Testhardware oder die Organisation der Bachelorposter.

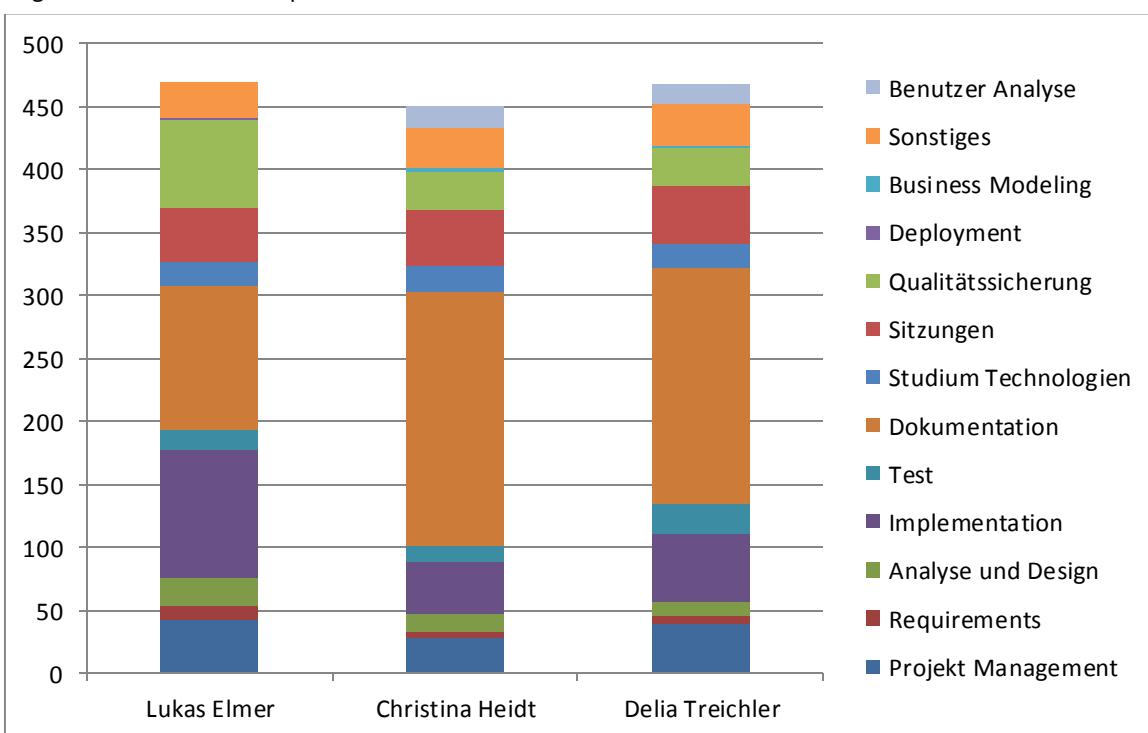


Abbildung 149 - Aktivitäten nach Personen gruppiert

## VI.4.4 Arbeitslisten

In diesem Abschnitt werden die Arbeitslisten pro Person mit zugehörigem Aufwand aufgelistet.

### VI.4.4.1 Lukas Elmer

Version	Ticket	Aufgewendete Zeit
<b>SP1</b>		<b>19.5</b>
	Feature #642: Entwicklungsumgebung eingerichtet/installiert	2.5
	Feature #643: Redmine, SVN und Sitzungsprotokolle sind eingerichtet	2.5
	Feature #645: Meeting, 20.02.12	1.5
	Feature #648: Brainstorming Grob-Interaktionskonzept durchgeführt	3
	Feature #652: Testsetup evaluiert	2
	Feature #657: Sprint 1 und Sprint 15/16 geplant	1
	Feature #666: Studium Technologien durchgeführt	5.5
	Feature #667: Meeting, 24.02.12	1.5
<b>SP2</b>		<b>28.5</b>
	Feature #647: Windows Kinect SDK Framework ausprobiert	2
	Feature #653: Mini Prototyp erstellt	12.5
	Feature #668: Meeting, 02.03.12	1.75
	Feature #672: Risikomanagement nachgeführt	1
	Feature #716: Testsetup evaluiert	2.75
	Feature #717: Kinect record Möglichkeit gefunden und entwickelt	8.5
<b>SP3</b>		<b>27.75</b>
	Feature #656: Bachelor Poster PDFs organisiert	0.25
	Feature #670: Meeting, 09.03.12	0.5
	Feature #675: Risikomanagement nachgeführt	0.5
	Feature #719: Projekt Plan Imagine Cup erstellt	14.5
	Feature #720: Meeting, 05.03.12. Imagine Cup	1.5
	Feature #724: Kinect Framework mithilfe Nutzwertanalyse ausgewählt	6.5
	Feature #725: Meeting, 06.03.12. Imagine Cup	3
	Feature #731: Sprint 04 geplant	1
<b>SP4</b>		<b>21.75</b>
	Feature #655: Formativer empirischer Test erstellt. Anforderungen an Gesten definiert	2
	Feature #718: Kinect replay Möglichkeit gefunden und entwickelt	9
	Feature #726: TFS Server Installation abgeschlossen	6.5
	Feature #728: Backlog erstellt	2.75
	Feature #732: Test Videowall aufgebaut	1.5
<b>SP5</b>		<b>11</b>
	Feature #673: Meeting, 23.03.12	1
	Feature #681: Risikomanagement nachgeführt	0.5
	Feature #738: Sitzung mit Markus Flückiger abgemacht	1
	Feature #740: Sprint 5 geplant	2
	Feature #741: Teamfördernde Massnahmen	2.5
	Feature #742: Grobarchitektur erarbeitet und implementiert	4

Version	Ticket	Aufgewendete Zeit
<b>SP6</b>		<b>17.75</b>
	Feature #674: Meeting, 02.04.12	1.5
	Feature #727: Formativer empirischer Test durchgeführt	2.5
	Feature #744: Tiefere Auflösungen getestet und dokumentiert	3
	Feature #745: Architektur mit Silvan besprochen	2.25
	Feature #746: Architekturprototyp erstellt: Handtracking	7.5
	Feature #750: Sprint 6 geplant	1
<b>SP7</b>		<b>50.5</b>
	Feature #676: Meeting, 17.04.12	2
	Feature #687: Risikomanagement nachgeführt	0.5
	Feature #734: Backlog erstellt, priorisiert	2.5
	Feature #755: Architekturprototyp erstellt: Handtracking	8.5
	Feature #757: Poster L sind organisiert	1
	Feature #758: Architektur mit Silvan besprochen. Zugangsdaten Server eingerichtet und an Silvan gesendet	1.5
	Feature #759: Sprint 7 geplant	1
	Feature #761: Architekturprototyp erstellt: Projekte vereinigt	1.5
	Feature #762: Architekturprototyp erstellt: Mit Kinect "klicken"	16
	Feature #763: Meeting, 12.04.12	5.5
	Feature #764: Web Architektur von MS studiert haben	8
	Feature #765: Tiers-Diagramm erstellt und beschrieben	2.5
<b>SP8</b>		<b>20.75</b>
	Feature #677: Meeting, 23.04.12	2.5
	Feature #753: Bilderprototyp auf Videowall getestet und dokumentiert	2.5
	Feature #767: Sprint 08 geplant	2.25
	Feature #769: Hand Cursor schön dargestellt	1.5
	Feature #771: DirectX auf Videowall ist getestet	3
	Feature #773: Usability Test mit Architekturprototyp durchgeführt und dokumentiert	1
	Feature #776: Poster E sind organisiert	1
	Feature #777: Code Review durchgeführt	2.5
	Feature #779: Skelett schön dargestellt	0.5
	Feature #780: Backlog: Definition of Done für aktuelle User Stories erfasst	2
	Feature #781: Meeting 26.04.12 vorbereitet	1
	Feature #870: Hand Cursor ruckelt weniger	1
<b>SP9</b>		<b>13.5</b>
	Feature #679: Meeting, 27.04.12	1.5
	Feature #774: Ideen gesammelt wie Personen von Videowall angezogen werden	2
	Feature #782: Sprint 9 geplant	1.5
	Feature #786: WPF Applikation mit Video erstellt/recherchiert	0.5
	Feature #788: Meeting 26.04.12	1.5
	Feature #789: SVN Base Architecture verschieben. Tag erstellt	1
	Feature #790: SVN Tag erstellt	0.75
	Feature #791: BA/Master-Vorstudie. Wissen ausgetauscht	0.75

<b>Version</b>	<b>Ticket</b>	<b>Aufgewendete Zeit</b>
	Feature #792: Einführung ins Projekt für Silvan	1
	Feature #794: MEF (Managed Extensibility Framework) studiert haben	3
<b>SP10</b>		<b>21.25</b>
	Feature #680: Code Review vom 03.05.12 durchgeführt	1.75
	Feature #783: Refactoring durchgeführt	1
	Feature #793: DirectX auf Videowall ist dokumentiert	2.5
	Feature #796: Ideen gesammelt und dokumentiert wie Personen von Videowall angezogen werden	2.5
	Feature #797: Mitsubishi Wall angesehen und dokumentiert	3
	Feature #798: Plugin Möglichkeit entwickelt	4
	Feature #803: Sprint 10 geplant	1.75
	Feature #805: Anpassungen gemäss Besprechung mit Herrn Heinzmann durchgeführt	0.75
	Feature #807: Anpassung bezüglich Code Review vom 03.05.12 gemacht	1.5
	Feature #808: Gesamtplanung anhand Kriterienliste überprüft und geplant	2
	Feature #831: SVN Tag erstellt	0.5
<b>SP11</b>		<b>29.75</b>
	Feature #682: Meeting, 11.05.12	1.75
	Feature #784: Demomodus (Verfolgung von Passanten) Kraftfeld besprochen und dokumentiert	2
	Feature #800: Mittagsmenu App in Plugin umgewandelt	0.5
	Feature #802: Poster App in Plugin umgewandelt	5
	Feature #832: Sprint 11 geplant	2.5
	Feature #839: Refactoring durchgeführt	8.5
	Feature #840: Anpassung bezüglich Code Review vom 03.05.12 gemacht	2
	Feature #841: Aufwändige Anpassung bezüglich Code Review vom 03.05.12 gemacht	6.5
	Feature #842: Meeting, 14.05.12	1
<b>SP12</b>		<b>35.5</b>
	Feature #683: Meeting, 22.05.12	2
	Feature #698: Usability Tests durchgeführt und protokolliert	1.5
	Feature #801: Plugin Schnittstelle definiert und dokumentiert	8
	Feature #847: Notifier Problem gelöst/umgangen und dokumentiert	1.5
	Feature #849: Refactoring durchgeführt	0.5
	Feature #850: Sprint 12 geplant	1
	Feature #855: Deployment Entwickler PC möglich	1
	Feature #856: Das Mittagsmenu wird angezeigt	4
	Feature #858: Navigation mit schönen "Tabs" dargestellt	6
	Feature #871: Externes Design festgelegt und validiert	4
	Feature #872: Mittagsmenu App automatisch aktualisiert	4
	Feature #873: Abschluss der Arbeit geplant	2
<b>SP13</b>		<b>30</b>
	Feature #685: Meeting, 25.05.12	0.5
	Feature #811: Ein Buch, ein ACM und IEEE Paper zitiert	6
	Feature #814: Konkurrenz analysiert und dokumentiert	4.5
	Feature #859: Sprint 13 geplant	2

Version	Ticket	Aufgewendete Zeit
	Feature #860: Refactoring durchgeführt	2
	Feature #869: Notifier Problem gelöst	2.5
	Feature #874: Administration der Videowall Inhalte definiert	6
	Feature #877: Verkleinertes Video auf Videowall abspielbar und dokumentiert	6.5
<b>SP14</b>		<b>43.5</b>
	Feature #686: Meeting. 01.06.12	3.25
	Feature #879: Externes Design festgelegt und validiert	6.5
	Feature #882: Aufgabenstellung gelesen und sichergestellt, dass alles dokumentiert	1.5
	Feature #886: Sprint 14 geplant	0.75
	Feature #888: Abstract geschrieben und abgegeben	0.5
	Feature #889: Refactoring durchgeführt	31
<b>SP15</b>		<b>61.25</b>
	Feature #688: Meeting. 06.06.12	1.75
	Feature #896: Architektur ist beschrieben	16.5
	Feature #899: Unit-Tests erstellt	12
	Feature #906: Stabilitätstest durchgeführt und dokumentiert	6
	Feature #907: Codereview 3 durchgeführt. dokumentiert und Anpassungen implementiert	15
	Feature #908: Refactoring durchgeführt	5
	Feature #909: Plug-in Framework Bild erstellt	5
<b>SP16</b>		<b>36.5</b>
	Feature #691: CD gebrannt und abgegeben	1
	Feature #695: Dokumente zusammenfügen. PDF generieren	3
	Feature #704: Lizenzvereinbarung unterschrieben	0.25
	Feature #706: Erklärung eigenständige Arbeit unterschrieben	0.25
	Feature #809: Aufwand dokumentiert (Piechart)	1
	Feature #880: Meeting, 13.06.12	2
	Feature #916: Persönlicher Bericht geschrieben	3
	Feature #917: Codereview 3 Feedback Michael dokumentiert	3
	Feature #918: Bericht und Poster sind ausgedruckt	2
	Feature #919: HSR-Forum-Stand aufgebaut. betreut und wieder abgebaut	8
	Feature #920: Korrekturen und kleine Änderungen an Dokumenten durchgeführt	13
<b>Total</b>		<b>468.75</b>

Tabelle 52 - Arbeitsliste Lukas Elmer

#### VI.4.4.2 Christina Heidt

Version	Ticket	Aufgewendete Zeit
<b>SP1</b>		<b>23.5</b>
	Feature #642: Entwicklungsumgebung eingerichtet/installiert	2
	Feature #644: Dokumentvorlage erstellt	4
	Feature #645: Meeting, 20.02.12	1.5
	Feature #646: Logo erstellt	4

<b>Version</b>	<b>Ticket</b>	<b>Aufgewendete Zeit</b>
	Feature #652: Testsetup evaluiert	3
	Feature #661: Projektplan erstellt	0.5
	Feature #666: Studium Technologien durchgeführt	7
	Feature #667: Meeting, 24.02.12	1.5
<b>SP2</b>		<b>25.25</b>
	Feature #649: Personen in der Mensa beobachtet, ausgewertet und dokumentiert	5.25
	Feature #658: Benutzer Befragung durchgeführt und dokumentiert	4
	Feature #668: Meeting, 02.03.12	1.75
	Feature #669: Fragebogen erstellt	4
	Feature #703: Grundriss ausgemessen und aufgezeichnet	9
	Feature #716: Testsetup evaluiert	1.25
<b>SP3</b>		<b>32</b>
	Feature #659: Vision erstellt	8
	Feature #670: Meeting, 09.03.12	0.5
	Feature #719: Projekt Plan Imagine Cup erstellt	12
	Feature #720: Meeting, 05.03.12, Imagine Cup	1.5
	Feature #721: Testsetup dokumentiert	5
	Feature #723: Personen in der Mensa beobachtet, ausgewertet und dokumentiert	0.5
	Feature #725: Meeting, 06.03.12, Imagine Cup	3
	Feature #731: Sprint 04 geplant	1.5
<b>SP4</b>		<b>22.25</b>
	Feature #650: Personas erstellt	5.75
	Feature #651: Szenarien erstellt	4.5
	Feature #655: Formativer empirischer Test erstellt. Anforderungen an Gesten definiert	3.5
	Feature #671: Meeting, 16.03.12	1
	Feature #728: Backlog erstellt	1.25
	Feature #730: Skeleton-Aufnahmen im Gebäude 4 gemacht	2
	Feature #732: Test Videowall aufgebaut	4.25
<b>SP5</b>		<b>27.5</b>
	Feature #673: Meeting, 23.03.12	1
	Feature #681: Risikomanagement nachgeführt	0.5
	Feature #733: Formativer empirischer Test erstellt, Anforderungen an Gesten definiert	4.5
	Feature #735: Codereview durchgeführt und im Quellcode dokumentiert	3
	Feature #736: Vorstudie Dokument überarbeitet	3
	Feature #737: Skeleton-Aufnahmen im Gebäude 4 gemacht	4
	Feature #739: WPF Applikation auf Test Hardware getestet und dokumentiert	3.5
	Feature #740: Sprint 5 geplant	2
	Feature #741: Teamfördernde Massnahmen	2.5
	Feature #742: Grobarchitektur erarbeitet und implementiert	3.5
<b>SP6</b>		<b>22.75</b>
	Feature #674: Meeting, 02.04.12	1.5

Version	Ticket	Aufgewendete Zeit
	Feature #727: Formativer empirischer Test durchgeführt	6
	Feature #729: Formativer empirischer Test dokumentiert	1
	Feature #744: Tieferen Auflösungen getestet und dokumentiert	7
	Feature #745: Architektur mit Silvan besprochen	2.25
	Feature #747: Architekturprototyp erstellt: Menu mit Mittagsmenu und Poster	4
	Feature #750: Sprint 6 geplant	1
<b>SP7</b>		<b>30.75</b>
	Feature #676: Meeting, 17.04.12	2.75
	Feature #734: Backlog erstellt, priorisiert	2.5
	Feature #748: Architekturprototyp erstellt: Navigation zwischen Poster	1
	Feature #751: Architekturprototyp erstellt: Menu mit Mittagsmenu und Poster (Bild)	8
	Feature #754: Architekturentscheide sind dokumentiert	5.5
	Feature #756: Vorstudie gemäss Sitzung angepasst	0.5
	Feature #759: Sprint 7 geplant	1
	Feature #763: Meeting, 12.04.12	5.5
	Feature #765: Tiers-Diagramm erstellt und beschrieben	4
<b>SP8</b>		<b>25</b>
	Feature #677: Meeting, 23.04.12	2
	Feature #689: Risikomanagement nachgeführt	0.5
	Feature #753: Bilderprototyp auf Videowall getestet und dokumentiert	2.5
	Feature #760: Refactoring durchgeführt	3.75
	Feature #767: Sprint 08 geplant	2
	Feature #769: Hand Cursor schön dargestellt	1
	Feature #770: Begründung für Poster festgehalten	3.75
	Feature #771: DirectX auf Videowall ist getestet	2
	Feature #773: Usability Test mit Architekturprotoyp durchgeführt und dokumentiert	3
	Feature #775: Sofortiges Erfolgserlebnis dokumentiert	0.5
	Feature #777: Code Review durchgeführt	2.5
	Feature #779: Skelett schön dargestellt	1
	Feature #780: Backlog: Definition of Done für aktuelle User Stories erfasst	0.5
<b>SP9</b>		<b>16.25</b>
	Feature #679: Meeting, 27.04.12	1.5
	Feature #774: Ideen gesammelt wie Personen von Videowall angezogen werden	2
	Feature #778: Dokument Vorstudie: Varianz bei Umfrage eingetragen	0.25
	Feature #782: Sprint 9 geplant	1.5
	Feature #785: Applikation ist mit linker Hand bedienbar	4
	Feature #787: Backlog ist aktuell	0.5
	Feature #788: Meeting, 26.04.12	1
	Feature #791: BA/Master-Vorstudie, Wissen ausgetauscht	0.75
	Feature #794: MEF (Managed Extensibility Framework) studiert haben	4.75
<b>SP10</b>		<b>24.75</b>
	Feature #680: Code Review vom 03.05.12 durchgeführt	1.75

<b>Version</b>	<b>Ticket</b>	<b>Aufgewendete Zeit</b>
	Feature #708: Risikomanagement nachgeführt	0.5
	Feature #793: DirectX auf Videowall ist dokumentiert	0.5
	Feature #796: Ideen gesammelt und dokumentiert wie Personen von Videowall angezogen werden	3
	Feature #797: Mitsubishi Wall angesehen und dokumentiert	10
	Feature #798: Plugin Möglichkeit entwickelt	3
	Feature #803: Sprint 10 geplant	0.5
	Feature #805: Anpassungen gemäss Besprechung mit Herrn Heinzmann durchgeführt	3
	Feature #808: Gesamtplanung anhand Kriterienliste überprüft und geplant	2
	Feature #831: SVN Tag erstellt	0.5
<b>SP11</b>		<b>25.25</b>
	Feature #682: Meeting, 11.05.12	1.75
	Feature #784: Demomodus (Verfolgung von Passanten) Kraftfeld besprochen und dokumentiert	2
	Feature #799: Bild der Hand ist auf die rechte bzw. linke Hand abgestimmt	2.5
	Feature #832: Sprint 11 geplant	2.5
	Feature #833: Demomodus: Vom Demomodus wird in den Interaktionsmodus gewechselt	7
	Feature #834: Demomodus: Vom Interaktionsmodus wird in den Demomodus gewechselt	5
	Feature #837: Demomodus: externes Design erstellt	4
	Feature #842: Meeting, 14.05.12	0.5
<b>SP12</b>		<b>27.5</b>
	Feature #683: Meeting, 22.05.12	1.75
	Feature #698: Usability Tests durchgeführt und protokolliert	2.25
	Feature #710: Risikomanagement nachgeführt	0.5
	Feature #810: TODOs in Dokumenten abgearbeitet	4
	Feature #816: Diskussion Accessability dokumentiert	2.5
	Feature #835: Demomodus: Apps werden automatisch gewechselt	0.25
	Feature #843: Demomodus: Zustandsdiagramm erstellt und dokumentiert	2.25
	Feature #844: Oliver Rehmann wegen Posterstand am 15.6.12 kontaktiert	0.5
	Feature #845: 2x4 Monitore-Setup mit Hellaumprojektor getestet und dokumentiert	1.5
	Feature #846: 2x4 Monitore-Setup mit Test-Wall getestet und dokumentiert	0.75
	Feature #848: Gewichtung bei Nutzwertanalyse begründet	1.75
	Feature #849: Refactoring durchgeführt	2.25
	Feature #850: Sprint 12 geplant	1
	Feature #857: Leichte Usability Test Korrekturen umgesetzt und dokumentiert	4.25
	Feature #873: Abschluss der Arbeit geplant	2
<b>SP13</b>		<b>29</b>
	Feature #685: Meeting, 25.05.12	1.25
	Feature #702: Abstract geschrieben	4
	Feature #711: Risikomanagement nachgeführt	0.5
	Feature #715: Poster erarbeitet	6.5

Version	Ticket	Aufgewendete Zeit
	Feature #817: Domain Model erstellt und dokumentiert	3.5
	Feature #825: Diskussion Notwendigkeit statistische Analyse dokumentiert	1
	Feature #829: Coding Standards dokumentiert und eingehalten	1.5
	Feature #859: Sprint 13 geplant	2
	Feature #861: Stakeholderanalyse erstellt	0.5
	Feature #862: Domain Analyse: Daten beschrieben	0.5
	Feature #863: Backlog ist aktuell	0.25
	Feature #864: Korrekturen Markus Stolze umgesetzt	2
	Feature #865: Usability Tests durchgeführt und dokumentiert	0.25
	Feature #868: Systemtests durchgeführt und dokumentiert	0.5
	Feature #874: Administration der Videowall Inhalte definiert	1.5
	Feature #875: Lesbarkeit der L-Poster überprüft und dokumentiert	1.5
	Feature #881: Prozentuale Poster Lesbarkeit analysiert	1.75
<b>SP14</b>		<b>29.75</b>
	Feature #686: Meeting, 01.06.12	5
	Feature #693: Extended Management Summary geschrieben	4.5
	Feature #812: Nicht-funktionale Anforderungen dokumentiert	0.5
	Feature #813: Funktionale Anforderungen dokumentiert	0.5
	Feature #818: User Environment Diagram oder Screen Map für Anwendungen mit mehreren Screens erstellt und dokumentiert	2
	Feature #821: Architektur ist beschrieben	11
	Feature #883: Poster erarbeitet	3
	Feature #885: Miniapps dokumentiert	1
	Feature #886: Sprint 14 geplant	0.75
	Feature #888: Abstract geschrieben und abgegeben	1.5
<b>SP15</b>		<b>53.75</b>
	Feature #688: Meeting, 06.06.12	2.25
	Feature #690: Video und Wiki Seite erstellt	12
	Feature #692: Persönlicher Bericht geschrieben	1.5
	Feature #694: Installationsanleitung geschrieben	3.5
	Feature #697: Weiterentwicklung dokumentiert	1.5
	Feature #699: Tools sind beschrieben	0.5
	Feature #700: Einleitung Technischer Bericht geschrieben	1.5
	Feature #701: Allgemeine Korrekturen, kleine Anpassungen	1.5
	Feature #815: Design Constraints dokumentiert	0.75
	Feature #819: GUI Design Entscheide dokumentiert	1.5
	Feature #820: GUI Guidelines dokumentiert	2
	Feature #826: Anleitung für Entwickler dokumentiert	0.5
	Feature #828: Code Qualität dokumentiert	3.5
	Feature #830: Warnings und Coding Issues dokumentiert und evt. bereinigt	4
	Feature #887: Konkurrenzanalyse überarbeitet	4
	Feature #890: Fragen zur Videowall beantwortet	0.5
	Feature #892: Poster erarbeitet	1.25
	Feature #893: Abstract geschrieben und abgegeben	1.5

Version	Ticket	Aufgewendete Zeit
	Feature #894: Sprint 15 geplant	0.5
	Feature #896: Architektur ist beschrieben	2.5
	Feature #897: Externes Design dokumentiert	1
	Feature #898: Extended Management Summary geschrieben und mit Bildern versehen	1
	Feature #900: Domain Analyse: Content Prozess zu Domain Model erstellt und dokumentiert	1
	Feature #902: Offerten angefordert	1.5
	Feature #905: Dokument-Korrekturen von Markus übernommen	2
	Feature #907: Codereview 3 durchgeführt, dokumentiert und Anpassungen implementiert	0.5
<b>SP16</b>		<b>35.5</b>
	Feature #691: CD gebrannt und abgegeben	1
	Feature #695: Dokumente zusammenfügen. PDF generieren	6
	Feature #704: Lizenzvereinbarung unterschrieben	0.5
	Feature #706: Erklärung eigenständige Arbeit unterschrieben	0.5
	Feature #809: Aufwand dokumentiert (Piechart)	6
	Feature #880: Meeting, 13.06.12	2
	Feature #912: Tests dokumentiert	1
	Feature #913: Video und Wiki Seite erstellt	4
	Feature #914: Poster erarbeitet	2
	Feature #915: Abstract geschrieben und abgegeben	1
	Feature #916: Persönlicher Bericht geschrieben	1.5
	Feature #918: Bericht und Poster sind ausgedruckt	2
	Feature #919: HSR-Forum-Stand aufgebaut, betreut und wieder abgebaut	8
<b>Total</b>		<b>450.75</b>

Tabelle 53 - Arbeitsliste Christina Heidt

#### VI.4.4.3 Delia Treichler

Version	Ticket	Aufgewendete Zeit
<b>SP1</b>		<b>25.5</b>
	Feature #642: Entwicklungsumgebung eingerichtet/installiert	2
	Feature #644: Dokumentvorlage erstellt	2
	Feature #645: Meeting, 20.02.12	1.5
	Feature #652: Testsetup evaluiert	0.5
	Feature #657: Sprint 1 und Sprint 15/16 geplant	3.5
	Feature #660: Risiken identifiziert	1
	Feature #661: Projektplan erstellt	1.5
	Feature #666: Studium Technologien durchgeführt	11
	Feature #667: Meeting, 24.02.12	2.5
<b>SP2</b>		<b>28</b>
	Feature #649: Personen in der Mensa beobachtet. ausgewertet und dokumentiert	5.5
	Feature #658: Benutzer Befragung durchgeführt und dokumentiert	16
	Feature #668: Meeting, 02.03.12	1.75

Version	Ticket	Aufgewendete Zeit
	Feature #669: Fragebogen erstellt	1.75
	Feature #672: Risikomanagement nachgeführt	0.25
	Feature #703: Grundriss ausgemessen und aufgezeichnet	1.5
	Feature #716: Testsetup evaluiert	1.25
<b>SP3</b>		<b>36</b>
	Feature #659: Vision erstellt	4
	Feature #670: Meeting, 09.03.12	0.5
	Feature #675: Risikomanagement nachgeführt	0.5
	Feature #719: Projekt Plan Imagine Cup erstellt	12.5
	Feature #720: Meeting, 05.03.12. Imagine Cup	2
	Feature #721: Testsetup dokumentiert	2.75
	Feature #722: Benutzer Befragung durchgeführt und dokumentiert	5
	Feature #723: Personen in der Mensa beobachtet, ausgewertet und dokumentiert	1.25
	Feature #724: Kinect Framework mithilfe Nutzwertanalyse ausgewählt	2
	Feature #725: Meeting, 06.03.12, Imagine Cup	3.5
	Feature #731: Sprint 04 geplant	2
<b>SP4</b>		<b>20.25</b>
	Feature #650: Personas erstellt	2
	Feature #651: Szenarien erstellt	4.75
	Feature #655: Formativer empirischer Test erstellt. Anforderungen an Gesten definiert	2
	Feature #671: Meeting, 16.03.12	1
	Feature #678: Risikomanagement nachgeführt	0.5
	Feature #728: Backlog erstellt	2.75
	Feature #730: Skeleton-Aufnahmen im Gebäude 4 gemacht	2
	Feature #732: Test Videowall aufgebaut	5.25
<b>SP5</b>		<b>31.5</b>
	Feature #673: Meeting, 23.03.12	1
	Feature #733: Formativer empirischer Test erstellt. Anforderungen an Gesten definiert	17.5
	Feature #736: Vorstudie Dokument überarbeitet	3.5
	Feature #737: Skeleton-Aufnahmen im Gebäude 4 gemacht	1
	Feature #739: WPF Applikation auf Test Hardware getestet und dokumentiert	0.5
	Feature #740: Sprint 5 geplant	2
	Feature #741: Teamfördernde Massnahmen	2.5
	Feature #742: Grobarchitektur erarbeitet und implementiert	3.5
<b>SP6</b>		<b>15.75</b>
	Feature #674: Meeting, 02.04.12	1.5
	Feature #684: Risikomanagement nachgeführt	0.5
	Feature #727: Formativer empirischer Test durchgeführt	5
	Feature #729: Formativer empirischer Test dokumentiert	5.5
	Feature #745: Architektur mit Silvan besprochen	2.25
	Feature #750: Sprint 6 geplant	1
<b>SP7</b>		<b>37</b>

<b>Version</b>	<b>Ticket</b>	<b>Aufgewendete Zeit</b>
	Feature #676: Meeting, 17.04.12	2.5
	Feature #734: Backlog erstellt. priorisiert	2.5
	Feature #749: Architekturprototyp erstellt: Kinect Skelett	7.5
	Feature #752: Architekturprototyp erstellt: PDF zu Bildern konvertieren	6
	Feature #754: Architekturentscheide sind dokumentiert	1
	Feature #755: Architekturprototyp erstellt: Handtracking	1
	Feature #756: Vorstudie gemäss Sitzung angepasst	1
	Feature #759: Sprint 7 geplant	1
	Feature #761: Architekturprototyp erstellt: Projekte vereinigt	8.5
	Feature #763: Meeting, 12.04.12	5
	Feature #765: Tiers-Diagramm erstellt und beschrieben	0.5
	Feature #851: Systemtests durchgeführt und dokumentiert	0.5
<b>SP8</b>		<b>21.5</b>
	Feature #677: Meeting, 23.04.12	2
	Feature #760: Refactoring durchgeführt	2.5
	Feature #767: Sprint 08 geplant	2.25
	Feature #769: Hand Cursor schön dargestellt	0.5
	Feature #770: Begründung für Poster festgehalten	2.25
	Feature #772: Usability Test mit Architekturprotoyp organisiert	2
	Feature #773: Usability Test mit Architekturprotoyp durchgeführt und dokumentiert	2.75
	Feature #775: Sofortiges Erfolgserlebnis dokumentiert	1.25
	Feature #777: Code Review durchgeführt	1
	Feature #779: Skelett schön dargestellt	4.5
	Feature #852: Systemtests durchgeführt und dokumentiert	0.5
<b>SP9</b>		<b>24</b>
	Feature #679: Meeting, 27.04.12	1.25
	Feature #707: Risikomanagement nachgeführt	0.5
	Feature #766: Web Architektur von MS studiert haben	3
	Feature #774: Ideen gesammelt wie Personen von Videowall angezogen werden	2
	Feature #778: Dokument Vorstudie: Varianz bei Umfrage eingetragen	8.75
	Feature #782: Sprint 9 geplant	1.5
	Feature #787: Backlog ist aktuell	0.5
	Feature #788: Meeting, 26.04.12	2.25
	Feature #791: BA/Master-Vorstudie. Wissen ausgetauscht	0.75
	Feature #794: MEF (Managed Extensibility Framework) studiert haben	2
	Feature #795: Projektmanagement / Administratives	1
	Feature #853: Systemtests durchgeführt und dokumentiert	0.5
<b>SP10</b>		<b>26.5</b>
	Feature #680: Code Review vom 03.05.12 durchgeführt	2
	Feature #793: DirectX auf Videowall ist dokumentiert	1.25
	Feature #796: Ideen gesammelt und dokumentiert wie Personen von Videowall angezogen werden	8.5
	Feature #797: Mitsubishi Wall angesehen und dokumentiert	4.75

Version	Ticket	Aufgewendete Zeit
	Feature #798: Plugin Möglichkeit entwickelt	1
	Feature #803: Sprint 10 geplant	1.5
	Feature #804: Web Architektur von MS studiert haben	1
	Feature #805: Anpassungen gemäss Besprechung mit Herrn Heinzmann durchgeführt	1
	Feature #806: Code Review vom 03.05.12 dokumentiert	2.5
	Feature #808: Gesamtplanung anhand Kriterienliste überprüft und geplant	2
	Feature #831: SVN Tag erstellt	0.5
	Feature #854: Systemtests durchgeführt und dokumentiert	0.5
<b>SP11</b>		<b>20.5</b>
	Feature #682: Meeting, 11.05.12	1.5
	Feature #709: Risikomanagement nachgeführt	0.5
	Feature #784: Demomodus (Verfolgung von Passanten) Kraftfeld besprochen und dokumentiert	3.25
	Feature #800: Mittagsmenu App in Plugin umgewandelt	5
	Feature #832: Sprint 11 geplant	2.5
	Feature #833: Demomodus: Vom Demomodus wird in den Interaktionsmodus gewechselt	1.5
	Feature #834: Demomodus: Vom Interaktionsmodus wird in den Demomodus gewechselt	0.5
	Feature #837: Demomodus: externes Design erstellt	0.25
	Feature #838: Systemtests durchgeführt und dokumentiert	2.5
	Feature #842: Meeting, 14.05.12	3
<b>SP12</b>		<b>30.25</b>
	Feature #683: Meeting, 22.05.12	2
	Feature #698: Usability Tests durchgeführt und protokolliert	1
	Feature #801: Plugin Schnittstelle definiert und dokumentiert	2.5
	Feature #810: TODOs in Dokumenten abgearbeitet	0.75
	Feature #816: Diskussion Accessibility dokumentiert	0.25
	Feature #835: Demomodus: Apps werden automatisch gewechselt	1.25
	Feature #836: Demomodus: Demotext zu aktiver App wird angezeigt	1.25
	Feature #843: Demomodus: Zustandsdiagramm erstellt und dokumentiert	2.5
	Feature #844: Oliver Rehmann wegen Posterstand am 15.6.12 kontaktiert	0.75
	Feature #845: 2x4 Monitore-Setup mit Hellraumprojektor getestet und dokumentiert	0.75
	Feature #846: 2x4 Monitore-Setup mit Test-Wall getestet und dokumentiert	2.5
	Feature #848: Gewichtung bei Nutzwertanalyse begründet	1.5
	Feature #849: Refactoring durchgeführt	8.75
	Feature #850: Sprint 12 geplant	1
	Feature #857: Leichte Usability Test Korrekturen umgesetzt und dokumentiert	0.5
	Feature #867: Systemtests durchgeführt und dokumentiert	1
	Feature #873: Abschluss der Arbeit geplant	2
<b>SP13</b>		<b>31.25</b>
	Feature #685: Meeting, 25.05.12	1.5
	Feature #702: Abstract geschrieben	1.75

<b>Version</b>	<b>Ticket</b>	<b>Aufgewendete Zeit</b>
	Feature #814: Konkurrenz analysiert und dokumentiert	4
	Feature #817: Domain Model erstellt und dokumentiert	0.75
	Feature #823: Unit-Tests erstellt	6
	Feature #859: Sprint 13 geplant	2
	Feature #860: Refactoring durchgeführt	0.5
	Feature #861: Stakeholderanalyse erstellt	4.25
	Feature #862: Domain Analyse: Daten beschrieben	0.25
	Feature #864: Korrekturen Markus Stolze umgesetzt	2
	Feature #865: Usability Tests durchgeführt und dokumentiert	2.5
	Feature #874: Administration der Videowall Inhalte definiert	2
	Feature #875: Lesbarkeit der L-Poster überprüft und dokumentiert	1.75
	Feature #877: Verkleinertes Video auf Videowall abspielbar und dokumentiert	1.5
	Feature #881: Prozentuale Poster Lesbarkeit analysiert	0.5
<b>SP14</b>		<b>27.25</b>
	Feature #686: Meeting, 01.06.12	5.25
	Feature #693: Extended Management Summary geschrieben	2.75
	Feature #712: Risikomanagement nachgeführt	0.5
	Feature #812: Nicht-funktionale Anforderungen dokumentiert	1
	Feature #813: Funktionale Anforderungen dokumentiert	1.5
	Feature #818: User Environment Diagram oder Screen Map für Anwendungen mit mehreren Screens erstellt und dokumentiert	0.5
	Feature #827: CHM Files generiert	5.25
	Feature #879: Externes Design festgelegt und validiert	2.5
	Feature #882: Aufgabenstellung gelesen und sichergestellt, dass alles dokumentiert	2.5
	Feature #884: Unit-Tests erstellt	1
	Feature #885: Miniapps dokumentiert	0.75
	Feature #886: Sprint 14 geplant	0.75
	Feature #888: Abstract geschrieben und abgegeben	1
	Feature #889: Refactoring durchgeführt	2
<b>SP15</b>		<b>59.75</b>
	Feature #688: Meeting, 06.06.12	2.75
	Feature #692: Persönlicher Bericht geschrieben	0.75
	Feature #694: Installationsanleitung geschrieben	0.5
	Feature #696: Code dokumentiert	5.5
	Feature #697: Weiterentwicklung dokumentiert	0.75
	Feature #699: Tools sind beschrieben	0.5
	Feature #701: Allgemeine Korrekturen. kleine Anpassungen	10.5
	Feature #815: Design Constraints dokumentiert	0.25
	Feature #819: GUI Design Entscheide dokumentiert	1.25
	Feature #820: GUI Guidelines dokumentiert	1
	Feature #824: Tests dokumentiert	4.5
	Feature #826: Anleitung für Entwickler dokumentiert	1.5
	Feature #828: Code Qualität dokumentiert	0.25

Version	Ticket	Aufgewendete Zeit
	Feature #887: Konkurrenzanalyse überarbeitet	8.5
	Feature #890: Fragen zur Videowall beantwortet	2
	Feature #892: Poster erarbeitet	0.25
	Feature #893: Abstract geschrieben und abgegeben	0.5
	Feature #894: Sprint 15 geplant	1
	Feature #895: CHM Files generiert	1
	Feature #896: Architektur ist beschrieben	1.5
	Feature #897: Externes Design dokumentiert	0.5
	Feature #898: Extended Management Summary geschrieben und mit Bildern versehen	1.5
	Feature #899: Unit-Tests erstellt	0.5
	Feature #900: Domain Analyse: Content Prozess zu Domain Model erstellt und dokumentiert	0.5
	Feature #901: CI/CD mit HSR abgeklärt	6
	Feature #905: Dokument-Korrekturen von Markus übernommen	6
<b>SP16</b>		<b>32.75</b>
	Feature #691: CD gebrannt und abgegeben	1
	Feature #695: Dokumente zusammenfügen, PDF generieren	5.5
	Feature #704: Lizenzvereinbarung unterschrieben	0.25
	Feature #706: Erklärung eigenständige Arbeit unterschrieben	0.25
	Feature #880: Meeting, 13.06.12	2
	Feature #904: Test Videowall abgebaut	0.5
	Feature #910: Poster, Prezi, Flip Video in Applikation eingebunden	5
	Feature #911: Tool geschrieben, um die PDF Poster zu Bildern zu konvertieren	1
	Feature #916: Persönlicher Bericht geschrieben	3
	Feature #918: Bericht und Poster sind ausgedruckt	2
	Feature #919: HSR-Forum-Stand aufgebaut, betreut und wieder abgebaut	8
	Feature #920: Korrekturen und kleine Änderungen an Dokumenten durchgeführt	4.25
<b>Total</b>		<b>467.75</b>

Tabelle 54 - Arbeitsliste Delia Treichler

# VII. Verzeichnisse

VII.1	Abbildungsverzeichnis .....	235
VII.2	Tabellenverzeichnis .....	238

## VII.1 Abbildungsverzeichnis

Abbildung 1 - Videowall im Eingangsbereich des Verwaltungsgebäudes .....	4
Abbildung 2 - Kinect, Bildquelle: <a href="http://www.wikipedia.org">www.wikipedia.org</a> .....	5
Abbildung 3 - Projektion der 3 x 3 55" Monitorkonstellation im Eingangsbereich des Verwaltungsgebäudes .....	6
Abbildung 4 - Testhardware.....	6
Abbildung 5 - Auslastung der Abstandszonen (aus Passantenanalyse) und Kinect Skelett-Erkennungsbereich .....	8
Abbildung 6 - Usability Test .....	9
Abbildung 1 - Gebäude der HSR, Bildquelle: <a href="http://www.hsr.ch">www.hsr.ch</a> .....	25
Abbildung 8 - Konfiguration der HoloWall, Bildquelle [matsushita03].....	30
Abbildung 9 - Touch Wall Setup: It's Mine, Don't Touch!, Bildquelle: [peltonen08] .....	31
Abbildung 10 - Extending Touch, Bildquelle [schick09].....	31
Abbildung 11 - Skeletal Tracking, Bildquelle [zhang12].....	32
Abbildung 12 - Leap Motion Sensor, Bildquelle: <a href="http://www.technobuffalo.com">www.technobuffalo.com</a> .....	32
Abbildung 13 - Panasonic D-IMager, Bildquelle: <a href="http://www.panasonic.biz">www.panasonic.biz</a> .....	33
Abbildung 14 - Dance Dance Revolution Game, Bildquelle : <a href="http://www.wikipedia.org">www.wikipedia.org</a> .....	33
Abbildung 15 - Anzahl Personen über die Zeit.....	35
Abbildung 16 - Auslastung der Abstandszonen .....	36
Abbildung 17 - Auslastung der Abstandszonen, Grundriss Verwaltungsgebäude.....	37
Abbildung 18 - Vorkommen der Gruppengrößen.....	38
Abbildung 19 - Aufteilung Einzelpersonen zu Gruppen .....	38
Abbildung 20 - Kinect Skelett-Erkennungsbereich, Grundriss Verwaltungsgebäude .....	39
Abbildung 21 - Total aller Studiengänge .....	42
Abbildung 22 - Vergleich der Studiengänge .....	43
Abbildung 23 - Auswertung nach Quartilen .....	44
Abbildung 24 - Meinungsverteilung.....	45
Abbildung 25 - Peter Posterleser, Bildquelle: <a href="http://www.office.com">www.office.com</a> .....	46
Abbildung 26 - Noemi Nichtinteressiert, Bildquelle: <a href="http://www.office.com">www.office.com</a> .....	47
Abbildung 27 - Erich Eventbesucher, Bildquelle: <a href="http://www.office.com">www.office.com</a> .....	49
Abbildung 28 - Handerkennung bei Arm hinter dem Rücken .....	62
Abbildung 29 - Systemübersicht, gewünschtes Endsystem .....	65
Abbildung 30 - Poster Ist-Prozess.....	67
Abbildung 31 - Poster Soll-Prozess .....	67
Abbildung 32 - Domain Model VideoWall.....	68
Abbildung 33 - Domain Model PosterApplication.....	69
Abbildung 34 - Domain Model LunchMenuApplication .....	69
Abbildung 35 - Anforderungen an den Test.....	71
Abbildung 36 - Posteransicht .....	72
Abbildung 37 - Unterteilung in Tabs .....	73
Abbildung 38 - Zonenmarkierung.....	73
Abbildung 39 - Skizze Testapplikation .....	74
Abbildung 40 - Testapplikation .....	75
Abbildung 41 - Demomodus, Ideen 1-3 .....	76
Abbildung 42 - Demomodus, Ideen 4-8 .....	77
Abbildung 43 - Demomodus, Idee 12, Erweiterung zu Idee 8 .....	78
Abbildung 44 - Demomodus, Ideen 9 und 10 .....	79
Abbildung 45 - Demomodus, Idee 11 .....	80
Abbildung 46 - Screen Map.....	81
Abbildung 47 - Handcursor Animation .....	81
Abbildung 48 - Externes Design, Videowall-Applikation .....	83
Abbildung 49 - Externes Design, Poster-Applikation (Plug-in) .....	83
Abbildung 50 - Externes Design, Demomodus Teaser-Text .....	84
Abbildung 51 - Externes Design, Demomodus Countdown .....	85
Abbildung 52 - Übersicht des Lebenszyklus.....	94
Abbildung 53 - Der Startup Prozess.....	95
Abbildung 54 - Demomodus Ablauf .....	96
Abbildung 55 - Navigation zwischen einzelnen Plug-in Applikationen .....	96

Abbildung 56 - Systemübersicht.....	97
Abbildung 57 - Architektur Diagramm.....	97
Abbildung 58 - Dispatcher Queue.....	99
Abbildung 59 - Poster-Applikation (Extension) wird über [Export(typeof(IApp))] als IApp exportiert .....	101
Abbildung 60 - AppController koordiniert den Import der Apps .....	102
Abbildung 61 - Der ExtensionFolder (Videowall-Applikation) importiert über das Attribut [Import] die Klassen, die das Interface IApp implementieren und sich in einem bestimmten Ordner (Directory) befinden. ....	103
Abbildung 62 - Der ExtensionManager führt den Import schliesslich mithilfe von MEF aus .....	104
Abbildung 63 - Anfängliche Implementation des Interfaces IApp.....	105
Abbildung 64 - Das IApp Interface.....	106
Abbildung 65 - Durch den IVideoWallServiceProvider können weitere Extensions geladen werden .....	106
Abbildung 66 - Sequenzdiagramm, Ablauf des Ladens und Aktivierens von Applikationen durch das Framework .....	107
Abbildung 67 - Teilaufgaben des Demomodus "Kraftfeld" .....	108
Abbildung 68 - Ideen zur Bewegungsart der Teilchen .....	109
Abbildung 69 - Zustandsdiagramm Interaktions- und Demomodus .....	110
Abbildung 70 - Beispiel eines Skeletts .....	111
Abbildung 71 - Skelett mit Zone (rot) für das Handtracking.....	111
Abbildung 72 - Beispiel Monitor mit Handtracking.....	112
Abbildung 73 - Handcursor auf nicht anklickbarem Element .....	112
Abbildung 74 - Ablauf eines Klicks auf einen Button .....	112
Abbildung 75 - Variante A: 3 x 3 55" Monitore, Ansicht.....	116
Abbildung 76 - Variante A: 3 x 3 55" Monitore, Hellraumprojektor Test.....	116
Abbildung 77 - Variante B: 2 x 2 55" Monitore, Ansicht.....	117
Abbildung 78 - Variante B: 2 x 2 55" Monitore, Hellraumprojektor Test .....	117
Abbildung 79 - Variante C: 1 x 6 55" Monitore, Ansicht.....	118
Abbildung 80 - Variante C: 1 x 6 55" Monitore, Hellraumprojektor Test .....	118
Abbildung 81 - Variante D: 2 x 4 55" Monitore, Ansicht .....	119
Abbildung 82 - Variante D: 2 x 4 55" Monitore, Hellraumprojektor Test .....	119
Abbildung 83 - Matrox M9188.....	121
Abbildung 84 - Matrox M9128.....	121
Abbildung 85 - Testhardware.....	122
Abbildung 86 - Videoeinstellungen VLC Media Player.....	124
Abbildung 87 - Variante C 3 x 3 55" Monitore mit einem 1.5-fach (blau) und 2-fach (gelb) vergrösserten Video .....	125
Abbildung 88 - Konfiguration "Independent" (XDDM, WDDM) .....	127
Abbildung 89 - Konfiguration "Stretched" (XDDM) .....	127
Abbildung 90 - Konfiguration "Partial stretched" (WDDM) .....	128
Abbildung 91 - Konfiguration "Joined & stretched" (WDDM) .....	129
Abbildung 92 - Konfiguration "Joined & partial stretched" .....	130
Abbildung 93 - Anzahl der Arbeiten pro Abteilung, Angaben in Prozent .....	134
Abbildung 94 - Testdurchführung Wizard of Oz mit einem Probanden .....	139
Abbildung 95 - Test 2: Reaktion der Nutzer .....	141
Abbildung 96 - Test 3: Reaktion auf Demomodus .....	141
Abbildung 97 - Test 4: Grafisches Design .....	142
Abbildung 98 - Unit Tests VideoWall .....	143
Abbildung 99 - Unit Tests PosterApp .....	143
Abbildung 100 - Unit Tests LunchMenuApp .....	144
Abbildung 101 - Entwicklung der Systemressourcen der laufenden Applikation über 111 Stunden, MB = MiB .....	154
Abbildung 102 - Minimum, Maximum, Mittelwert, Median, Stichproben des Stabilitätstests .....	155
Abbildung 103 - Dependency Graph, Videowall-Applikation.....	156
Abbildung 104 - Dependency Graph, Poster-Applikation .....	157
Abbildung 105 - Dependency Graph, Mittagsmenu-Applikation.....	158
Abbildung 106 - Test Coverage VideoWall.....	159
Abbildung 107 - Test Coverage PosterApp .....	160
Abbildung 108 - Test Coverage LunchMenuApp .....	161
Abbildung 109 - Metriken Visual Studio, Videowall-Applikation.....	162
Abbildung 110 - Metriken Visual Studio, Poster-Applikation.....	163

Abbildung 111 - Metriken Visual Studio, Mittagsmenu-Applikation.....	163
Abbildung 112 - Metriken NDepend, Übersicht, Videowall-Applikation.....	163
Abbildung 113 - Metriken NDepend, Assemblies, Videowall-Applikation.....	164
Abbildung 114 - Metriken NDepend, Übersicht, Poster-Applikation.....	164
Abbildung 115 - Metriken NDepend, Assemblies, Poster-Applikation.....	164
Abbildung 116 - Metriken NDepend, Übersicht, Mittagsmenu-Applikation.....	165
Abbildung 117 - Metriken NDepend, Assemblies, Mittagsmenu-Applikation.....	165
Abbildung 118 - Warnings, Videowall-Applikation.....	165
Abbildung 119 - Warnings, Poster-Applikation.....	166
Abbildung 120 - Warnings, Mittagsmenu-Applikation.....	166
Abbildung 121 - Naming Style.....	167
Abbildung 122 - Formatierungsstil, Braces Layout.....	167
Abbildung 123 - Formatierungsstil, Line Breaks and Wrapping.....	167
Abbildung 124 - Coding Issues, Videowall-Applikation.....	168
Abbildung 125 - Coding Issues, Poster-Applikation.....	169
Abbildung 126 - Coding Issues, Mittagsmenu-Applikation.....	170
Abbildung 127 - Cleanup Einstellungen.....	171
Abbildung 128 - Sandcastle Help File Builder, Hinzufügen der Visual Studio Solution.....	171
Abbildung 129 - Betrieb der Videowall.....	184
Abbildung 130 - Initialer Deployment Prozess.....	186
Abbildung 131 - app.config, Konfigurationssektionen.....	188
Abbildung 132 - app.config, Sektion Unity, Namespaces&Assemblies .....	189
Abbildung 133 - app.config, Sektion Unity, Mapping von Interfaces auf Klassen .....	189
Abbildung 134 - app.config, Sektion Unity, Mapping für ISkeletonReader .....	189
Abbildung 135 - app.config, Sektion Unity, Mapping für ICursorViewModel .....	190
Abbildung 136 - Interaktionszone .....	190
Abbildung 137 - app.config, Sektion Unity, Padding.....	190
Abbildung 138 - app.config, Sektion Unity, Pfad zu den Plug-ins.....	191
Abbildung 139 - app.config, Sektion Unity, Demomodus .....	191
Abbildung 140 - app.config, Sektion Unity, Singleton.....	192
Abbildung 141 - app.config, Sektion Unity, KinectReplayFile.....	192
Abbildung 142 - app.config, Runtime .....	192
Abbildung 143 - Von der Idee zur Umsetzung.....	201
Abbildung 144 - Verlauf geschätzte und aufgewendete Zeit .....	205
Abbildung 145 - Aufteilung Personenaufwand .....	217
Abbildung 146 - Personenaufwand pro Sprint.....	218
Abbildung 147 - Personenaufwand pro Woche.....	218
Abbildung 148 - Aufteilung nach Tätigkeit.....	219
Abbildung 149 - Aktivitäten nach Personen gruppiert .....	219

## VII.2 Tabellenverzeichnis

Tabelle 1 - Releases .....	16
Tabelle 2 - Die geplanten Sprints .....	17
Tabelle 3 - Scrum Elemente .....	19
Tabelle 4 - Initiale Stakeholderanalyse, Projekt-Stakeholder .....	26
Tabelle 5 - Initiale Stakeholderanalyse, Produkt-Stakeholder .....	26
Tabelle 6 - Vor- und Nachteile der Posterpräsentationsarten .....	28
Tabelle 7 - Bestehende Videowalls mit Kurzbeschreibung .....	29
Tabelle 8 - Beobachtungszeitabschnitte .....	35
Tabelle 9 - Anzahl Fragebögen pro Abteilung .....	40
Tabelle 10 - Tools .....	52
Tabelle 11 - User Stories .....	59
Tabelle 12 - Attribute VideoWallApplication .....	68
Tabelle 13 - Attribute PosterApplication .....	69
Tabelle 14 - Attribute LunchMenu .....	69
Tabelle 15 - Attribute Dish .....	69
Tabelle 16 - Nutzeranalyse: Auswahl Kinect Framework .....	91
Tabelle 17 - Nutzwertanalyse: PDF-Darstellung .....	93
Tabelle 18 - Nutzwertanalyse: Auswahl Monitorkonstellation für Videowall .....	120
Tabelle 19 - Video Performance Test Resultate .....	126
Tabelle 20 - Zusammenfassung Resultat empirischer formativer Test .....	140
Tabelle 21 - Systemtests Sprint 7 .....	145
Tabelle 22 - Systemtests Sprint 8 .....	145
Tabelle 23 - Systemtests Sprint 9 .....	146
Tabelle 24 - Systemtests Sprint 10 .....	147
Tabelle 25 - Systemtests Sprint 11 .....	147
Tabelle 26 - Systemtests Sprint 12 .....	149
Tabelle 27 - Systemtests Sprint 13 .....	150
Tabelle 28 - Systemtests Sprint 14 .....	151
Tabelle 29 - Systemtests Sprint 15/16 .....	153
Tabelle 30 - Lines of Code (LOC) .....	162
Tabelle 31 - Annotationen und Kommentare Code Review 19.04.2012 .....	176
Tabelle 32 - Annotationen und Kommentare Code Review 03.05.2012 .....	178
Tabelle 33 - Annotationen und Kommentare Code Review 05.06.2012 .....	180
Tabelle 34 - Aufwand Übersicht .....	204
Tabelle 35 - Tickets Sprint 1 .....	205
Tabelle 36 - Tickets Sprint 2 .....	206
Tabelle 37 - Tickets Sprints 3 .....	206
Tabelle 38 - Tickets Sprint 4 .....	207
Tabelle 39 - Tickets Sprint 5 .....	207
Tabelle 40 - Tickets Sprint 6 .....	208
Tabelle 41 - Tickets Sprint 7 .....	209
Tabelle 42 - Tickets Sprint 8 .....	209
Tabelle 43 - Tickets Sprint 9 .....	210
Tabelle 44 - Tickets Sprint 10 .....	211
Tabelle 45 - Tickets Sprint 11 .....	212
Tabelle 46 - Tickets Sprint 12 .....	213
Tabelle 47 - Tickets Sprint 13 .....	213
Tabelle 48 - Tickets Sprint 14 .....	214
Tabelle 49 - Tickets Sprint 15 .....	215
Tabelle 50 - Tickets Sprint 16 .....	216
Tabelle 51 - Personenaufwand Übersicht .....	217
Tabelle 52 - Arbeitsliste Lukas Elmer .....	223
Tabelle 53 - Arbeitsliste Christina Heidt .....	228
Tabelle 54 - Arbeitsliste Delia Treichler .....	233

# VIII.Anhang

## VIII.1 Anhang A

- Glossar
- Literaturverzeichnis und Referenzen

## VIII.2 Glossar

### VIII.2.1 Begriffserklärung

Begriff	Beschreibung
.NET	Software-Plattform der Microsoft Corporation
.NET Runtime	Laufzeitumgebung von .NET, interpretiert den Zwischencode
ASP.NET MVC3	Framework um Webapplikationen zu erstellen
Avatar	Grafische Darstellung einer echten Person in der virtuellen Welt
Content Management System	System, über welches Inhalte(z.B. der Videowall) verwaltet werden können
Daisy Chain Board	Ermöglicht es, mehrere Monitore in Serie zu schalten, das DVI-Signal wird vom einen an den nächsten Monitor weitergegeben
Frames per second	Ein Mass für die Bildfrequenz.
Graphical User Interface	Die grafische Benutzeroberfläche ermöglicht dem Benutzer die Interaktion mit dem Computer über grafische Symbole.
Graphics Processing Unit	Grafikprozessor
Iframe	Das Inline Frame (Iframe) positioniert ein anderes HTML Dokument in einem Frame.
Inversion of Control / Dependency Injection	Inversion of Control bezeichnet das Prinzip, bei dem die Steuerung der Ausführung bestimmter Programme an das Framework übergeben wird. Dependency Injection übergibt die Verantwortung zur Erzeugung und Initialisierung von Objekten an das Framework ab.
Kinect	Ein Sensor von Microsoft, der Körperbewegungen erkennt, damit so ein Spiel gesteuert werden kann.
Managed Extensibility Framework	Microsoft Framework für einfach erweiterbare Anwendungen
Natural User Interface	Ermöglicht einen natürlichen Umgang mit Interaktionen durch die Nutzung von Gesten
NiTE	Die Natural Interaction Middleware der Firma PrimeSense.
PrimeSense	Die Firma, welche die 3D sensing technology für Kinect bereitstellt
Rastergrafik	Eine Rastergrafik besteht aus einer festen Anordnung von Bildpunkten, welchen je eine Farbe zugeordnet ist. Daher hat eine Rastergrafik auch eine feste Bildgrösse.
Rational Unified Process	Vorgehensmodell zur iterativen Softwareentwicklung
Scrum	Vorgehensmodell zur empirischen und iterativen Softwareentwicklung
Silverlight	Entwicklungsgerüst für Web- oder Mobile-Applikationen. Teil des .NET Frameworks
Single Sign-on	Durch SSO (Single Sign-on) muss sich der Nutzer nur einmal authentifizieren um auf Dienste oder Rechner zugreifen zu können.
Teaser	Element (Bild, Video, Text, etc.) welches potentielle Nutzer neugierig macht
Timeboxing	Dieser Vorgang findet statt, wenn eine geplante Aufgabe in Form eines Tickets nicht erledigt werden kann. Um trotzdem den Sprint abschliessen zu können, wird in einem späteren Sprint ein neues Ticket mit dem gleichen Namen eröffnet und das alte geschlossen.
Usability Test	Dient der Überprüfung der Gebrauchstauglichkeit einer Software, wird mit potenziellen Nutzern durchgeführt
User Centered Design	Vorgehen zur benutzerorientierten, gebrauchstauglichen Gestaltung von Software
Vektorgrafik	Eine Vektorgrafik wird durch Linien/Kurven, Liniengrößen und Farben beschrieben. Vektorgrafiken sind beliebig skalierbar.
Videowall	Durch mehrere Monitore wird eine grosse Bildschirmfläche geformt.
Windows Communication Foundation	Serviceorientiertes Kommunikationsmittel für verteilte Anwendungen der Microsoft Corporation. Teil des .NET Frameworks

Begriff	Beschreibung
Windows Presentation Foundation	Grafikframework der Microsoft Corporation. Teil des .NET Frameworks
Wizard of Oz - Experiment	Ein Experiment, bei dem ein Mensch mit einem Computer kommuniziert. Im Hintergrund erzeugt aber ein anderer Mensch die Reaktionen des Systems, nicht der Computer selbst.
XML Paper Specification	Vektorisiertes Dateiformat der Microsoft Corporation

---

### VIII.2.2 Abkürzungserläuterung

Abkürzung	Erläuterung
CMS	Content Management System
FPS	Frames per second
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HSR	Hochschule für Technik Rapperswil
IFS	Institut für Software der HSR
MEF	Managed Extensibility Framework
NUI	Natural User Interface
RUP	Rational Unified Process
SSO	Single Sign-on
UCD	User Centered Design
WCF	Windows Communication Foundation
WPF	Windows Presentation Foundation
XPS	XML Paper Specification

### VIII.3 Literaturverzeichnis

- [chaudhri09] Chaudhri, I., Ording, B., Anzures, F. A., Van Os, M., Lemay, S. O., Forstall, S., Christie, G. (2009) Unlocking a Device by Performing Gestures on an Unlock Image. U.S. Patent US8046721.  
<http://www.google.com/patents/US20090241072>
- [eilbrecht07] Karl Eilbrecht, Gernot Starkte, Patterns kompakt, Entwurfsmuster für effective Software-Entwicklung, 2. Auflage, Spektrum Verlag, ISBN-13: 978-3-8274-1591-2, 2007
- [egli11] Felix Egli, Schnyder Michael, Kinect Bodyscanner, 02.09.2011,  
<http://eprints3.hsr.ch/180/>
- [elmer11] Lukas Elmer, Christina Heidt, Delia Treichler, Project Flip 2.0, 02.04.2012,  
<http://eprints3.hsr.ch/220/>
- [han05] Han, J. Low-cost multi-touch sensing through frustrated total internal reflection. In Proc. of UIST '05. ACM, NY, USA, 2005, pp. 115–118.  
<http://dl.acm.org/citation.cfm?id=1095054>
- [hsr11] Hochschule für Technik Rapperswil (HSR), Corporate Design, Version 1, 10.01.2011
- [leapmotion12] Leap Motion, Inc., Pre-order,  
<https://live.leapmotion.com/order.html>  
letzter Zugriff 05.06.2012
- [matsushita03] Matsushita, N. and Rekimoto, J. HoloWall: designing a finger, hand, body, and object sensitive wall. In Proc. UIST 2003, ACM Press (2003), New York, 159–168.  
<http://dl.acm.org/citation.cfm?id=263549>
- [microsoft06] Microsoft Corporation, Windows Vista Display Driver Model,  
<http://msdn.microsoft.com/en-US/library/aa480220.aspx>  
letzter Zugriff: 16.05.2012
- [microsoft09] Microsoft Corporation, WPF Apps With The Model-View-ViewModel Design Pattern,  
<http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>  
letzter Zugriff: 10.06.2012
- [microsoft10] Microsoft Corporation, Kinect Skeletal Tracking,  
<http://www.microsoft.com/about/technicalrecognition/Kinect-Skeletal-Tracking.aspx>,  
letzter Zugriff: 05.06.2012
- [microsoft12] Microsoft Corporation, Windows 2000 Display Driver Model (XDDM) Design Guide,  
<http://msdn.microsoft.com/en-us/library/windows/hardware/ff570584%28v=vs.85%29.aspx>  
letzter Zugriff: 16.05.2012
- [microsoft12.1] Microsoft Corporation, Documentation for MEF,  
<http://mef.codeplex.com/documentation>  
letzter Zugriff: 22.05.2012
- [microsoft12.2] Microsoft Corporation, Kinect for Windows Human Interface Guidelines, Version 1.5.0, 21.05.2012,  
<http://go.microsoft.com/fwlink/?LinkId=247735>
- [microsoft12.3] Microsoft Corporation, Threading Model,  
<http://msdn.microsoft.com/en-us/library/ms741870>  
letzter Zugriff 12.06.2012
- [peltonen08] Peltonen, P., Kurvinen, E., Salovaara, A., Jacucci, G., Ilmonen, T., Evans, J., Oulasvirta, A., and Saarikko, P. It's Mine, Don't Touch!: interactions at a large multitouch display in a city centre. In Proc. of CHI '08. ACM, NY, USA, 2008, pp. 1285–1294.  
<http://dl.acm.org/citation.cfm?id=1357255>
- [schick09] Alexander Schick, Florian van de Camp, Joris Ijsselmuider and Rainer Stiefelhagen, Extending Touch: Towards Interaction with Large-Scale Surfaces. ACM International Conference on InteractiveTabletops and Surfaces 2009, Banff, Alberta, Canada.  
<http://dl.acm.org/citation.cfm?id=1731903.1731927>
- [schmidt00] Schmidt, Douglas C.; Michael Stal, Hans Rohnert, Frank Buschmann (2000). Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects. John Wiley & Sons. ISBN: 0-471-60695-2
- [zhang12] Zhengyou Zhang; , "Microsoft Kinect Sensor and Its Effect," Multimedia, IEEE , vol.19, no.2, pp.4-10, Feb. 2012  
doi: 10.1109/MMUL.2012.24  
<http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6190806&isnumber=6190801>

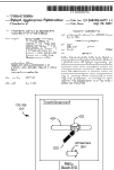


Sign in

## Patents

[Read this application](#)[Download PDF](#)**Unlocking a Device by Performing Gestures on an Unlock Image**

Imran Chaudhri et al


[Overview](#)  
[Abstract](#)  
[Drawings](#)  
[Description](#)  
[Claims](#)

3

**Application number:** 12/477,075**Publication number:**

US 2009/0241072 A1

**Filing date:** Jun 2, 2009**Issued patent:** [US8046721](#) (Issue date Oct 25, 2011)[Go](#)

A device with a touch-sensitive display may be unlocked via gestures performed on the touch-sensitive display. The device is unlocked if contact with the display corresponds to a predefined gesture for unlocking the device. The device displays one or more unlock images with respect to which the predefined gesture is to be performed in order to unlock the device. The performance of the predefined gesture with respect to the unlock image may include moving the unlock image to a predefined location and/or moving the unlock image along a predefined path. The device may also display visual cues of the predefined gesture on the touch screen to remind a user of the gesture.

**Inventors:** Imran Chaudhri, Bas Ording, Freddy Allen Anzures, Marcel Van Os, Stephen O. Lemay, Scott Forstall, Greg Christie

**Current U.S. Classification:** 715/863

[View patent at USPTO](#)[Search USPTO Assignment Database](#)[Download USPTO Public PAIR data](#)**Referenced by**

Citing Patent	Filing date	Issue date	Original Assignee	Title
<a href="#">US7793225</a>	Dec 29, 2008	Sep 7, 2010	Apple Inc.	Indication of progress towards satisfaction of a user input condition
<a href="#">US7882234</a>	Apr 20, 2004	Feb 1, 2011	Canon Kabushiki Kaisha	Wireless communication system, wireless communication device, and control method for establishing a one-to-one relationship
<a href="#">US8131859</a>	Apr 20, 2004	Mar 6, 2012	Canon Kabushiki Kaisha	Wireless communication system, and wireless communication device and control method

**Claims**

1. A method of unlocking a hand-held electronic device, the device including a touch-sensitive display, the method comprising:

detecting a contact with the touch-sensitive display at a first predefined location corresponding to an unlock image;

moving the unlock image on the touch-sensitive display in accordance with movement of the contact while continuous contact with the touch screen is maintained; and

unlocking the hand-held electronic device if the moving the unlock image on the touch-sensitive display results in movement of the unlock image from the first predefined location to a predefined unlock region on the touch-sensitive display.

2. The method of claim 1, wherein the moving comprises movement along any desired path.

3. The method of claim 1, wherein the moving comprises movement along a predefined channel from the first predefined location to the predefined unlock region.

4. The method of claim 1, further comprising displaying visual cues to communicate a direction of movement of the unlock image required to unlock the device.

5. The method of claim 4, wherein the visual cues comprise text.

6. The method of claim 4, wherein said visual cues comprise an arrow indicating a general direction of movement.

7. A portable electronic device, comprising:

a touch-sensitive display;

memory;

one or more processors; and

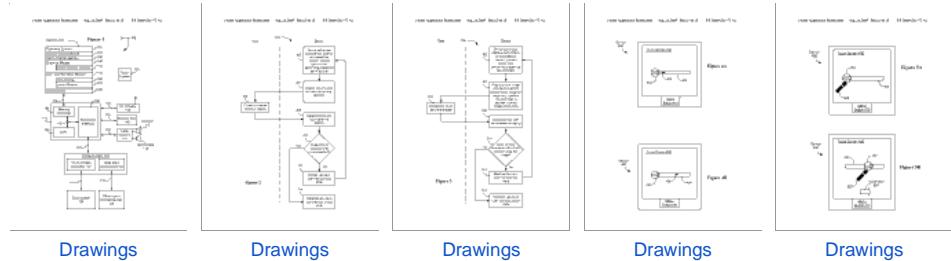
one or more modules stored in the memory and configured for execution by the one or more processors, the one or more modules including instructions:

to detect a contact with the touch-sensitive display at a first predefined location corresponding to an unlock image;

to move the unlock image on the touch-sensitive display in accordance with movement of the detected contact while continuous contact with the touch-sensitive display is maintained; and

8. The device of claim 7, further comprising instructions to display visual cues to communicate a direction of movement of the unlock image required to unlock the device.
9. The device of claim 8, wherein the visual cues comprise text.
10. The device of claim 8, wherein said visual cues comprise an arrow indicating a general direction of movement.
11. A portable electronic device, comprising:
- a touch-sensitive display;
  - means for displaying an unlock image at a first predefined location on the touch-sensitive display while the device is in a user-interface lock state;
  - means for detecting contact with the touch-sensitive display; and
  - means for moving the unlock image on the touch-sensitive display in response to detecting the contact in accordance with movement of the contact while continuous contact with the touch screen is maintained; and
  - means for transitioning the device to a user-interface unlock state if the moving the unlock image on the touch-sensitive display results in movement of the unlock image from the first predefined location to a predefined unlock region on the touch-sensitive display.
12. A computer program product for use in conjunction with a portable electronic device comprising a touch-sensitive display, the computer program product comprising a computer readable storage medium and a computer program mechanism embedded therein, the computer program mechanism comprising instructions for:
- detecting a contact with the touch-sensitive display at a first predefined location corresponding to an unlock image;
  - moving the unlock image on the touch-sensitive display in accordance with movement of the contact while continuous contact with the touch screen is maintained; and
  - unlocking the hand-held electronic device if the moving the unlock image on the touch-sensitive display results in movement of the unlock image from the first predefined location to a predefined unlock region on the touch-sensitive display.

## Drawings



## Kinect Bodyscanner

Egli, Felix and Schnyder, Michael (2011) *Kinect Bodyscanner*. Bachelor thesis, HSR Hochschule für Technik Rapperswil.



PDF - Supplemental Material  
[Download \(33Mb\)](#) | [Preview](#)

### Abstract

Im Rahmen dieser Bachelorarbeit sollte ein Ausstellungsstück für die Sonderausstellung „Der vermessen(d)e Mensch“ im Technorama definiert und mit Hilfe geeigneter Technologien entwickelt werden. Dies mit dem Ziel, es anschliessend in den Ausstellungsbetrieb zu integrieren. Aus diesem Grund musste in diesem Projekt besonderes Augenmerk auf die Erfüllung von nicht-funktionalen Qualitätskriterien wie Verständlichkeit und Stabilität gelegt werden. Die Microsoft Kinect-Technologie, welche ursprünglich für die Gebärdensprachsteuerung der XBox entwickelt wurde, ermöglicht zum ersten Mal die kostengünstige Verarbeitung von Tiefenbildern. Diese Tiefeninformationen können über das inoffizielle OpenNI-Framework aus dem Kinect-Sensor ausgelesen werden. Auf Basis dieser Möglichkeiten wurden unterschiedliche technische Prototypen erstellt, welche jeweils auf Umsetzbarkeit und Resultatgenauigkeit geprüft wurden. Zusammen mit dem Technorama, der Umfang eines ersten Prototyps definiert werden. Ein wichtiger Bestandteil der Arbeit war der Test des erstellten Prototyps im Technorama. Hierbei wurde speziell auf die Verständlichkeit und Stabilität der Applikation geachtet. Zusätzlich zum Kinect-Sensor wird auch eine Waage für die Personenmessung beigezogen. Diese Hardware wird über das GoIO-Framework angesprochen. Als Entwicklungssprache kam .NET mit C# zum Einsatz. Die grafische Umsetzung wurde mit WPF realisiert. Für die Überwachung der Lösung wurden PerformanceMeter eingesetzt. Diese stellen die Performance-Daten über die in Windows integrierte Schnittstelle auch anderen Applikationen zur Verfügung. Es wurde ein funktionsfähiger und benutzerfreundlicher Prototyp erstellt, welcher die gesetzten Anforderungen an den Prototypen erfüllt. Durch den Benutzertest im Technorama und Langzeit- und Performancetests konnte dies verifiziert werden. Es sind nur noch wenige, bereits dokumentierte Anpassungen nötig, bis das Ausstellungsobjekt in die Ausstellung integriert werden kann.

**Item Type:** Thesis (Bachelor)

[Area of Application > Industry](#)

**Subjects:** [Area of Application > Multimedia > Microsoft Kinect](#)

[Technologies > Programming Languages > C#](#)

[Brands > Microsoft](#)

**Divisions:** [Bachelor of Science FHO in Informatik > Bachelor Thesis](#)

Creators	Contributor Type	Email
Egli, Felix		
Schnyder, Michael		

Contributors	Contribution	Name	Email
Thesis advisor		Flückiger, Markus	

**Funders:** Swiss Science Center Technorama

**Depositing User:** [HSR Deposit User](#)

**Date Deposited:** 02 Sep 2011 09:46

**Last Modified:** 02 Sep 2011 09:46

**URI:** <http://eprints3.hsr.ch/id/eprint/180>

Actions (login required)



[View Item](#)

## Project Flip 2.0

Elmer, Lukas and Heidt, Christina and Treichler, Delia (2011) *Project Flip 2.0*. Student Research Project thesis, HSR Hochschule für Technik Rapperswil.



PDF - Supplemental Material  
[Download \(11Mb\)](#) | [Preview](#)

### Abstract

Durch den Surface 2 von Microsoft ergeben sich völlig neue Möglichkeiten, Informationen interaktiv zu präsentieren. Dies möchte die Zühlke Engineering AG für die Visualisierung ihrer Projekte nutzen. Momentan stehen die Projektinformationen (nachfolgend als Project Note bezeichnet) in Papierform im Wartebereich zur Verfügung. Dabei ergibt sich einerseits das Problem, dass nie alle Projekte zur gleichen Zeit ausgestellt werden können und andererseits erschwert sich die Suche nach spezifischen Inhalten. Diese Nachteile möchte die Zühlke Engineering AG durch die Verwendung einer Surface Applikation beheben. Zudem möchte sie damit auch ihre Expertise im Bereich Clienttechnologien untermauern und Kunden die Möglichkeiten von Microsoft Surface als Ausstellungsgerät aufzeigen. Als Ausgangslage diente Project Flip 1.0, welches für ein Dell Multi-Touch Tablet implementiert wurde. Dieses Projekt war jedoch primär darauf ausgelegt, bei Kundengesprächen einfacher auf Projekte zugreifen zu können. Die in diesem Projekt erworbenen Erkenntnisse konnten aber für Project Flip 2.0 wichtige Informationen liefern. Um die daraus erkannten Probleme umgehen zu können, wurde für Project Flip 2.0 gleich zu Beginn ein Prototyp ausgearbeitet. Für die Entwicklung dieses Projektes wurde ein benutzerorientiertes Vorgehen gewählt. Die aus den Interviews abgeleiteten Anforderungen wurden dann in der Entwicklung umgesetzt. Der Fokus dieser Arbeit war die Erarbeitung einer einfachen, intuitiven Anwendung, welche gleich zu Beginn ohne Hilfe bedient werden kann. Obwohl für alle Teammitglieder WPF und .NET Neuland darstellten, ist es gelungen, einen funktionstüchtigen und ansprechenden Prototyp zu entwickeln. Da der Surface 2 zum Zeitpunkt des Projektes noch nicht erhältlich war, ist die Installation und Inbetriebnahme der Applikation auf dem Gerät durch die Zühlke Engineering AG noch ausstehend. Neben dem Client-Teil wurde auch eine Anbindung an den Server umgesetzt, über welche die Project Notes heruntergeladen werden können. Auch wurde das User Interface durch Papierprototypen und abschliessend durch einen Usability Test durch die Benutzer validiert. Zudem wurde ein externes Design erarbeitet, um ein Konzept für die Gestaltung des User Interfaces zu definieren. Zu jeder Project Note existieren Metadaten, welche die Project Note mit verschiedenen Begriffen umschreibt. In dieser Arbeit wurde eine Technik zur Vereinfachten Darstellung von Metadaten entwickelt. Dies war notwendig, um die grosse Menge von Metadaten auf ein für Besucher handhabbares Mass zu reduzieren. Des Weiteren setzte sich das Team mit fortgeschrittenen Visualisierungskonzepten auseinander, indem sie einen WPF Prototyp für die Perspective Wall Visualisierung entwickelte. Da die Fertigstellung einer solchen Lösung jedoch den Projektaufwand überschritten hätte, konnte sie nicht in das Projekt integriert werden.

**Item Type:** Thesis (Student Research Project)

[Area of Application > Consumer oriented](#)

[Area of Application > Multimedia > Microsoft Surface](#)

**Subjects:** [Technologies > Frameworks and Libraries > .NET](#)

[Technologies > Frameworks and Libraries > WPF](#)

[Metatags > IFS \(Institute for Software\)](#)

**Divisions:** [Bachelor of Science FHO in Informatik > Student Research Project](#)

Creators	Creators	Email
Elmer, Lukas		
Heidt, Christina		
Treichler, Delia		

Contributors	Contribution	Name	Email
Thesis advisor		Stolze, Markus	

**Funders:** Zühlke Engineering AG

**Depositing User:** [HSR Deposit User](#)

**Date Deposited:** 02 Apr 2012 16:04

**Last Modified:** 02 Apr 2012 16:04

**URI:** <http://eprints3.hsr.ch/id/eprint/220>

Actions (login required)



[View Item](#)



## Low-cost multi-touch sensing through frustrated total internal reflection

Full Text:  [Pdf](#)

2005 Article

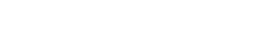
Author: [Jefferson Y. Han](#) [New York University, New York, NY](#)

Bibliometrics

- Downloads (6 Weeks): 132
- Downloads (12 Months): 1,274
- Citation Count: 247

Published in:

- Proceeding  
[UIST '05 Proceedings of the 18th annual ACM symposium on User interface software and technology](#)  
Pages 115 - 118  
ACM New York, NY, USA ©2005  
[table of contents](#) ISBN: 1-59593-271-2 doi:>[10.1145/1095034.1095054](https://doi.org/10.1145/1095034.1095054)



## Tools and Resources

 [Request Permissions](#) [TOC Service:](#) [Email](#) [RSS](#) [RSS](#) [Save to Binder](#) [Export Formats:](#)[BibTeX](#) [EndNote](#) [ACM Ref](#) [Upcoming Conference:](#)[UIST '12](#)

Share:

**Tags:** [frustrated total internal reflection](#) [human factors](#) [input devices and strategies](#) [more tags](#) [Feedback](#) | Switch to [single page view](#) (no tabs)[Abstract](#) [Authors](#) [References](#) [Cited By](#) [Index Terms](#) [Publication](#) [Reviews](#) [Comments](#) [Table of Contents](#)

This paper describes a simple, inexpensive, and scalable technique for enabling high-resolution multi-touch sensing on rear-projected interactive surfaces based on *frustrated total internal reflection*. We review previous applications of this phenomenon to sensing, provide implementation details, discuss results from our initial prototype, and outline future directions.

Powered by **THE ACM GUIDE TO COMPUTING LITERATURE**

The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2012 ACM, Inc.  
[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)



PRE-ORDER

ABOUT

DEVELOPERS

BLOG

JOBS

INVESTORS

CONTACT



A limited number of Leap devices are available for immediate pre-order. Your credit card will not be charged until your Leap is shipped. You can cancel your order at any time.

[Privacy Policy](#)[Terms and Conditions](#)

We are estimating that these units will ship in December (2012) or January (2013).

Name \*    
First Name      Last Name

Email \*

Quantity      1 (\$69.99)

Shipping Method      FedEx Ground (\$5.99)

Total Cost      \$ 75.98

## Billing

Credit Card \*

Expiration Date \*

Street Address \*

Address 2

City \*

Country \* United States

State Alabama

Zip/Post Code \*

Billing Phone \*

## Shipping

Street Address (if different)

Address 2

Country United States

**State**

**Zip/Post Code**

**City**

**Submit Form**

## HoloWall: designing a finger, hand, body, and object sensitive wall

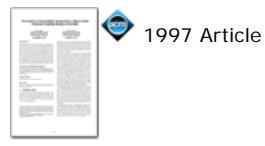
Full Text: [!\[\]\(f864718ce74e5168928593c055983f93\_img.jpg\) Pdf](#)

Authors: [Nobuyuki Matsushita](#) Department of Computer Science, Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa 223 Japan  
[Jun Rekimoto](#) Sony Computer Science Laboratory Inc., 3-14-13 Higashitanda, Shinagawa-ku, Tokyo 141, Japan

Published in:



- Proceeding  
[UIST '97 Proceedings of the 10th annual ACM symposium on User interface software and technology](#)  
 Pages 209 - 210  
[ACM New York, NY, USA ©1997](#)  
[table of contents](#) ISBN: 0-89791-881-9 doi:>[10.1145/263407.263549](https://doi.org/10.1145/263407.263549)



1997 Article

### Bibliometrics

- Downloads (6 Weeks): 12
- Downloads (12 Months): 90
- Citation Count: 68

### Tools and Resources

 [Request Permissions](#)

 [TOC Service:](#)

 [Email](#)  [RSS](#)

 [Save to Binder](#)

 [Export Formats:](#)

[BibTeX](#) [EndNote](#) [ACM Ref](#)

 [Upcoming Conference:](#)

[UIST '12](#)

Share:



**Tags:** [augmented reality](#)  
[design](#) [evaluation/methodology](#)  
[human factors](#) [infrared input](#)  
[devices and strategies](#) [interaction](#)  
[styles](#) [ubiquitous computing](#) [wall](#)  
[interfaces](#)

 [Feedback](#) | Switch to [single page view](#) (no tabs)

[Abstract](#) [Authors](#) [References](#) [Cited By](#) [Index Terms](#) [Publication](#) [Reviews](#) [Comments](#) [Table of Contents](#)



[Nobuyuki Matsushita](#)

No contact information provided yet.

### **Bibliometrics:** publication history

Publication years	1997-2003
Publication count	7
Citation Count	128
Available for download	6
Downloads (6 Weeks)	36
Downloads (12 Months)	271

[View colleagues](#) of Nobuyuki Matsushita

### **Bibliometrics:** publication history

Publication years	1992-2012
Publication count	77
Citation Count	1,636
Available for download	59
Downloads (6 Weeks)	718
Downloads (12 Months)	6,802

[View colleagues](#) of Jun Rekimoto

Powered by **THE ACM GUIDE TO COMPUTING LITERATURE**

The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2012 ACM, Inc.  
[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)

# Windows Vista Display Driver Model



48 out of 139 rated this helpful - [Rate this topic](#)

Microsoft Corporation

July 2006

## Contents

[Introduction](#)

[Overall WDDM Benefits](#)

[Desktop Window Manager](#)

[Video Improvements](#)

[Deployment](#)

[Mobility and Power Features](#)

[New Graphics APIs](#)

**Summary:** The Windows Display Driver Model (WDDM) is a new display driver architecture supported in Windows Vista. This display architecture is an overhaul of the Windows XP-based display architecture and gives users a better performing, more reliable desktop experience, while supporting new scenarios, graphics, and applications.

WDDM also provides video content playback that rivals typical consumer electronics devices. It does this by making it easy to connect to external monitors, providing for protected HD video playback and increasing overall video playback quality. For the first time in Windows, graphics processing unit (GPU) multitasking is possible, enabling users to run more than one GPU-intensive application simultaneously.

Finally, WDDM improves the PC gaming experience by simplifying the generalized GPU programming model for developers, maintaining a consistency in hardware capabilities, which translates into a PC gaming experience that will leapfrog that of even the latest consoles.

This new display driver model is a crucial piece of the new desktop experience and thus having a GPU that supports WDDM is a requirement for the Windows Vista "Premium Ready" marketing designation and will be applicable to future Windows Vista logo programs. (7 printed pages)

## Introduction

The rates of performance improvements for GPUs have exceeded Moore's law by a wide margin. Experts say GPUs have been improving at a "Moore's Law Cubed" rate, which roughly translates to a doubling of graphics processing power every year. To date, this massive graphics processing power has predominantly been used for video games, multimedia-rich suites, like video editing, special effects applications, and for high-end technical applications like computer-aided design. As a result, even though virtually every modern PC includes one, the power of the GPU is rarely used in day-to-day activities, resulting in GPUs not being a top priority in the PC design process.

Windows Vista harnesses the power of the GPU for more than just gaming. For starters, Windows Vista relies on the GPU to give all customers a better overall day-to-day desktop experience. The Windows video playback infrastructure relies on WDDM to deliver high-definition (HD) video playback and to take video playback to a level on par with the latest consumer electronics devices. In addition, there are a number of other key applications that now utilize the GPU, such as the new Windows Photo Gallery. This wider utilization of the GPU, concurrently by the operating system and multiple applications, is enabled by WDDM.

This paper examines the general and specific benefits of WDDM including:

- Overall WDDM benefits
- Desktop Window Manager (DWM)
- Video improvements, including Protected Video Playback (PVP)
- Deployment
- Mobility and power
- New graphics APIs

## Overall WDDM Benefits

### • Stability

In Windows XP, display drivers, which are large and complex, can be a major source of system instability. These drivers execute entirely in kernel mode (i.e., deep in the system code) and hence a single problem in the driver will often force the entire system to reboot. According to the crash analysis data collected during the Windows XP timeframe, display drivers are responsible for up to 20 percent of all *blue screens*. Thus it became one of the primary goals of Windows Vista to reduce such crashes and provide customers with a more reliable desktop experience while also lowering partner support costs. Reliability is even more important now, as there are mainstream features that require the graphics pipeline to perform efficiently around the clock. The new [Desktop Window Manager](#), which draws onscreen windows, is an example of such a feature. The benefits of experiencing a stable desktop experience, free of redrawing errors, certainly apply to all customers, and not just gamers.

At a technical level, WDDM display drivers have two components, a kernel mode driver (KMD) that is very streamlined, and a user-mode driver that does most of the intense computations. With this model, most of the code is moved out of kernel mode. That is, the kernel mode piece is now solely responsible for lower-level functionality and the user mode piece takes on heavier functionality such as facilitating the translation from higher-level API constructs to direct GPU commands while maintaining application compatibility. This greatly reduces the chance of a fatal blue screen and most graphics driver-related problems result in at worst one application being affected.

WDDM also provides fault-tolerance against display driver hangs. This enables Windows Vista to detect system hangs and restart the display driver again without the need of a system reboot.

Additionally, display drivers in Windows Vista have been significantly simplified by eliminating the need to include code for the support of various device driver interfaces introduced over many years. Thus, Windows Vista implements only a single interface while ensuring that all the older drivers are recognized and function optimally.

### • Performance

When an application requiring the GPU is launched, control is wrested away from any other applications currently using the GPU. This happens because there is no ability to share the GPU across multiple applications simultaneously. In Windows Vista, the GPU is utilized for many common activities, like basic window management and video rendering. Additionally, with the Windows Presentation Foundation (part of the .NET Framework 3.0 development platform), there are new APIs that enable developers to easily use the GPU to provide non-gaming applications with rich user-friendly interfaces. Since all these applications and features are dependent on the GPU, GPU multitasking is critical.

WDDM enables multiple applications to utilize the GPU simultaneously by implementing the following:

- GPU memory manager—arbitrates video memory allocation
- GPU scheduler—schedules various GPU applications according to their priority

With these technologies, applications no longer have to cede the GPU when another application requiring its services starts-up. Instead, the GPU is scheduled in a more efficient

fashion.

- **Security**

Security is a major concern on any platform that supports multitasking. A secure OS needs to ensure that resources used by one application are isolated from another. This isolation of applications is a requirement for protected HD content playback.

WDDM, via the security model built into the GPU memory manager and scheduler, provides this needed isolation. Current Windows XP display drivers do not have such facilities and are limited in the types of HD content they can playback.

## Desktop Window Manager

The Desktop Window Manager (DWM) is the technology in Windows Vista that controls the display and update of windows on the desktop. In order to eliminate drawing artifacts that are apparent on earlier versions of Windows, where applications draw directly to the screen asynchronously, the DWM composites onscreen content, such as application windows, in a back- or off-screen buffer, before drawing them onto the user's display. The DWM is, in essence, a Direct3D application that is active from the moment the computer is turned on. Thus, core features of the WDDM, such as the ability to share the GPU resources and processing, become essential in this scenario. For Windows Vista, there were two areas of investment in relation to the DWM: overall improvement in quality and a captivating user experience. The WDDM plays a key role in enabling all these advances, as we will see below. Built on top of the DWM is the Windows Vista signature end-user experience, Windows Aero.

- **Quality**

In Windows XP, applications update their windows directly when the OS requests them to. These requests could be executed asynchronously with respect to the refresh rate of the monitor or to any updates that may be currently running. The effect of these requests is that the user sees windows tearing and re-drawing incorrectly or slowly. The DWM style of window presentation eliminates the tearing artifacts, providing a high quality desktop experience. The benefit to the end user is that the system appears to be more responsive and the experience is cleaner.

- **Windows Aero**

The DWM also makes the new Windows Aero user experience possible. Aero is Windows Vista's best designed, highest performing desktop experience. It requires a PC equipped with an appropriate graphics configuration, which includes support for WDDM; the PC must also be equipped for Windows Vista Ultimate, Windows Vista Enterprise, Windows Vista Business, or Windows Vista Home Premium product editions. The DWM delivers a number of features that enable end users to find and access windows on their desktop in more convenient and direct ways. These features rely on being able to share graphics memory between applications and the DWM, which is enabled by WDDM. Other features, like Windows Flip and Windows Flip 3D, let you confidently manage the windows on your desktop, helping you to see them in a visually striking yet convenient way. See [Windows Vista Upgrade Advisor beta](#) for exact hardware requirements, and see [Windows Vista Capable and Premium Ready PCs](#) to check for compatibility.

The DWM enables the Windows Aero experience to provide the following end user features:

- Live taskbar thumbnails for open and minimized applications.
- Windows Flip (Alt + Tab), and Windows Flip 3D (Windows key + Tab)—two new visually stunning ways to manage open application windows.
- DPI scaling—support for higher DPI screens so that text displays reliably no matter the resolution of your monitor.

- **Visual refresh**

A noticeably new element of the Windows Aero experience is the professional looking,

transparent glass design, with subtle effects like dynamic reflections and smooth animations. Visual elements such as the glass effect rely on the Direct3D pixel-shader pipeline, and the virtualization of the GPU by the WDDM. But, beyond the new graphics and visual polish, the Windows Aero desktop experience performs as smoothly and professionally as it looks, giving the user a simple and high quality experience.

## Video Improvements

One of the goals of Windows Vista is to match the quality of video playback that people expect from mainstream consumer-electronics devices. Since video playback is dependent on the driver architecture, Windows Vista, with its upgraded driver model, is able to provide a number of video playback improvements.

- **Easy TV-out support**

Today, connecting a PC to a TV is difficult for the average consumer. Unlike a consumer electronics device, simply plugging in a TV does not work without extensive configuration. The setup can often require an additional monitor and third-party products. WDDM eliminates connectivity issues between a TV or monitor and the PC, giving users plug and play simplicity.

- **Improved video playback quality**

Video playback on current operating systems suffers from quality issues like excessive video glitching and poor color fidelity when compared to consumer electronic devices. WDDM provides a number of facilities to mitigate these problems.

First, WDDM enables Windows Vista and running applications to queue frames to be presented on the GPU. Second, working closely with the queuing feature is a feedback mechanism that determines when frames are presented. Together, these two features can immensely improve the quality of video playback by constantly maintaining the synchronicity between audio and video presentations, thus improving video playback and reducing video glitching substantially.

Finally, WDDM drivers also provide support for better color (gamma) correction via the Direct3D9x and Direct3D10 APIs, which in turn requires the GPU to support these APIs.

- **HD video playback**

As mentioned above in the security section, WDDM provides support for secure playback of HD video content, a requirement of many content providers.

## Deployment

A major frustration for Windows end-users is software updates that force the system through a reboot. Graphics drivers formerly required a reboot after an update. With Windows Vista and a WDDM-supporting GPU, users are no longer required to reboot when the graphics driver is updated.

## Mobility and Power Features

There are also WDDM benefits for mobile new Mobility features in Windows Vista that depend on WDDM.

- **Hot plug detection of displays and projectors**

With Windows XP connecting a laptop to a projector can prove to be a multistep process, where each step might be totally different from one laptop vendor to another. In Windows Vista, this process of connecting to an external display or projector is greatly simplified. When an additional external display or a projector is connected to the display adapter, WDDM detects this new state instantly and automatically notifies the Transient Mobility Manager (TMM) module, which is the underlying technology for the new Windows Vista External Display wizard. The wizard makes it simple to extend or clone the desktop to the second display

device.

- **"Hybrid" Sleep**

The Windows Vista advanced Sleep state combines the fast on/off of Standby function with the reliability of the Hibernate function. In Windows XP, Hibernate causes the contents of system memory to be saved to the hard disk and the system to be powered off; Standby causes the contents of memory to be preserved with a small amount of power, while the remainder of the system is powered off.

However, in Windows Vista, "Hybrid" Sleep causes the contents of system memory to be saved to the hard disk and the system placed into Standby at the same time. Therefore, the system is able to resume from system memory extremely fast—in less than 2 seconds for Windows Premium-capable PCs. However, if the battery drains fully or a power loss causes the contents of system memory to be lost, the system can resume from the hard disk.

**Note** Hybrid Sleep requires WDDM.

## New Graphics APIs

WDDM implements a much more functional graphical resource management. This functionality is critical to the proper functioning of the Direct3D9x and Direct3D10 APIs that are available to developers.

- **Windows Presentation Foundation**

Windows Presentation Foundation is the next generation of the Windows client development platform for user interface, documents, graphics, desktop-based, and connected applications, as well as content. It enables a wider spectrum of developers to harness the power of the GPU, without necessarily having to be aware of low-level graphics programming constructs. It is now easier for designers and developers to build user interfaces and content that give customers a richer and more engaging experience. As a result we will soon see a new generation of applications, both local and Web-based, that provide more engaging experiences, better visualization of data, and greater improvements to the reading experience.

While Windows Presentation Foundation is available for Windows XP SP2 and Server 2003 versions, on Windows Vista Windows Presentation Foundation benefits from WDDM in terms of scalability, as one can more reliably run multiple WPF applications concurrently.

- **Console-like simplicity for gaming**

Direct3D10 is the new Direct3D API that enables next-generation 3-D graphics and sophisticated parallel computing. This API depends on the resource management functionality of WDDM to provide some of its expressive power. Key benefits of Direct3D10 APIs are:

- **Generalized GPU programming model** Elimination of limits to the GPU programming, made possible in part by WDDM resource management, enables a more powerful GPU programming model that can be used for both next-generation 3-D graphics as well as for more general purpose uses that can benefit from highly parallel processing such as image processing.
- **Consistency in hardware capabilities** Like in a console, all Direct3D10-capable hardware looks identical to the developer. In other words, capability bits, which distinguish between different hardware configurations running D3D9.x, do not exist. This enables a much simpler implementation of software.
- **Visual effects** Features such as the Geometry Shader and Stream Out aid the developer in creating more life-like and realistic graphics that go beyond what the next-generation consoles can deliver today.
- **Better system performance** Better performance is achieved because processing can be offloaded consistently when required by the CPU.

Did you find this helpful?  Yes  No

© 2012 Microsoft. All rights reserved.

## Patterns

# WPF Apps With The Model-View-ViewModel Design Pattern

Josh Smith

This article discusses:

- Patterns and WPF
- MVP pattern
- Why MVVM is better for WPF
- Building an application with MVVM

This article uses the following technologies:

WPF, data binding

Code download available from the [MSDN Code Gallery](#)

Browse the Code Online

**Contents**

- [Order vs. Chaos](#)  
[The Evolution of Model-View-ViewModel](#)  
[Why WPF Developers Love MVVM](#)  
[The Demo Application](#)  
[Relaying Command Logic](#)  
[ViewModel Class Hierarchy](#)  
[ViewModelBase Class](#)  
[CommandViewModel Class](#)  
[MainWindowViewModel Class](#)  
[Applying a View to a ViewModel](#)  
[The Data Model and Repository](#)  
[New Customer Data Entry Form](#)  
[All Customers View](#)  
[Wrapping Up](#)

**Developing the user interface** of a professional software application is not easy. It can be a murky blend of data, interaction design, visual design, connectivity, multithreading, security, internationalization, validation, unit testing, and a touch of voodoo. Considering that a user interface exposes the underlying system and must satisfy the unpredictable stylistic requirements of its users, it can be the most volatile area of many applications.

There are popular design patterns that can help to tame this unwieldy beast, but properly separating and addressing the multitude of concerns can be difficult. The more complicated the patterns are, the more likely that shortcuts will be used later on which undermine all previous efforts to do things the right way.

It is not always the design patterns at fault. Sometimes we use complicated design patterns, which require writing a lot of code because the UI platform in use does not lend itself well to a simpler pattern. What's needed is a platform that makes it easy to build UIs using simple, time-tested, developer-approved design patterns. Fortunately, Windows Presentation Foundation (WPF) provides exactly that.

As the software world continues to adopt WPF at an increasing rate, the WPF community has been developing its own ecosystem of patterns and practices. In this article, I'll review some of those best practices for designing and implementing client applications with WPF. By leveraging some core features of WPF in conjunction with the Model-View-ViewModel (MVVM) design pattern, I will walk through an example program that demonstrates just how simple it can be to build a WPF application the "right way."

By the end of this article, it will be clear how data templates, commands, data binding, the resource system, and the MVVM pattern all fit together to create a simple, testable, robust framework on which any WPF application can thrive. The demonstration program that accompanies this article can serve as a template for a real WPF application that uses MVVM as its core architecture. The unit tests in the demo solution show how easy it is to test the functionality of an application's user interface when that functionality exists in a set of ViewModel classes. Before diving into the details, let's review why you should use a pattern like MVVM in the first place.

**Order vs. Chaos**

It is unnecessary and counterproductive to use design patterns in a simple "Hello, World!" program. Any competent developer can understand a few lines of code at a glance. However, as the number of features in a program increases, the number of lines of code and moving parts increase accordingly. Eventually, the complexity of a system, and the recurring problems it contains, encourages developers to organize their code in such a way that it is easier to comprehend, discuss, extend, and troubleshoot. We diminish the cognitive chaos of a complex system by applying well-known names to certain entities in the source code. We determine the name to apply to a piece of code by considering its functional role in the system.

Developers often intentionally structure their code according to a design pattern, as opposed to letting the patterns emerge organically. There is nothing wrong with either approach, but in this article, I examine the benefits of explicitly using MVVM as the architecture of a WPF application. The names of certain classes include well-known terms from the MVVM pattern, such as ending with "ViewModel" if the class is an abstraction of a view. This approach helps avoid the cognitive chaos mentioned earlier. Instead, you can happily exist in a state of controlled chaos, which is the natural state of affairs in most professional software development projects!

**MSDN Magazine Blog****Standing Desk Experiment**

Like a lot of developers, my job has me spending a LOT of time sitting in front of a computer monitor. So much so that I'm constantly struggling with... [More...](#)

Tuesday, Jun 5

**Developer Guide: Working with Windows Phone and the Cloud**

The good folks over at Microsoft Patterns & Practices are up to their old tricks again, this time releasing a useful publication aimed at Windows Phon... [More...](#)

Friday, Jun 1

[More MSDN Magazine Blog entries >](#)**Current Issue**[Browse All MSDN Magazines](#)**Subscribe to MSDN Flash newsletter**

Receive the MSDN Flash e-mail newsletter every other week, with news and information personalized to your interests and areas of focus.

## The Evolution of Model-View-ViewModel

Ever since people started to create software user interfaces, there have been popular design patterns to help make it easier. For example, the Model-View-Presenter (MVP) pattern has enjoyed popularity on various UI programming platforms. MVP is a variation of the Model-View-Controller pattern, which has been around for decades. In case you have never used the MVP pattern before, here is a simplified explanation. What you see on the screen is the View, the data it displays is the model, and the Presenter hooks the two together. The view relies on a Presenter to populate it with model data, react to user input, provide input validation (perhaps by delegating to the model), and other such tasks. If you would like to learn more about the Model View Presenter, I suggest you read Jean-Paul Boodhoo's [August 2006 Design Patterns column](#).

Back in 2004, Martin Fowler published an article about a pattern named [Presentation Model](#) (PM). The PM pattern is similar to MVP in that it separates a view from its behavior and state. The interesting part of the PM pattern is that an abstraction of a view is created, called the Presentation Model. A view, then, becomes merely a rendering of a Presentation Model. In Fowler's explanation, he shows that the Presentation Model frequently updates its View, so that the two stay in sync with each other. That synchronization logic exists as code in the Presentation Model classes.

In 2005, John Gossman, currently one of the WPF and Silverlight Architects at Microsoft, unveiled the [Model-View-ViewModel \(MVVM\)](#) pattern on his blog. MVVM is identical to Fowler's Presentation Model, in that both patterns feature an abstraction of a View, which contains a View's state and behavior. Fowler introduced Presentation Model as a means of creating a UI platform-independent abstraction of a View, whereas Gossman introduced MVVM as a standardized way to leverage core features of WPF to simplify the creation of user interfaces. In that sense, I consider MVVM to be a specialization of the more general PM pattern, tailor-made for the WPF and Silverlight platforms.

In Glenn Block's excellent article "[Prism: Patterns for Building Composite Applications with WPF](#)" in the September 2008 issue, he explains the Microsoft Composite Application Guidance for WPF. The term ViewModel is never used. Instead, the term Presentation Model is used to describe the abstraction of a view. Throughout this article, however, I'll refer to the pattern as MVVM and the abstraction of a view as a ViewModel. I find this terminology is much more prevalent in the WPF and Silverlight communities.

Unlike the Presenter in MVP, a ViewModel does not need a reference to a view. The view binds to properties on a ViewModel, which, in turn, exposes data contained in model objects and other state specific to the view. The bindings between view and ViewModel are simple to construct because a ViewModel object is set as the DataContext of a view. If property values in the ViewModel change, those new values automatically propagate to the view via data binding. When the user clicks a button in the View, a command on the ViewModel executes to perform the requested action. The ViewModel, never the View, performs all modifications made to the model data.

The view classes have no idea that the model classes exist, while the ViewModel and model are unaware of the view. In fact, the model is completely oblivious to the fact that the ViewModel and view exist. This is a very loosely coupled design, which pays dividends in many ways, as you will soon see.

## Why WPF Developers Love MVVM

Once a developer becomes comfortable with WPF and MVVM, it can be difficult to differentiate the two. MVVM is the lingua franca of WPF developers because it is well suited to the WPF platform, and WPF was designed to make it easy to build applications using the MVVM pattern (amongst others). In fact, Microsoft was using MVVM internally to develop WPF applications, such as Microsoft Expression Blend, while the core WPF platform was under construction. Many aspects of WPF, such as the look-less control model and data templates, utilize the strong separation of display from state and behavior promoted by MVVM.

The single most important aspect of WPF that makes MVVM a great pattern to use is the data binding infrastructure. By binding properties of a view to a ViewModel, you get loose coupling between the two and entirely remove the need for writing code in a ViewModel that directly updates a view. The data binding system also supports input validation, which provides a standardized way of transmitting validation errors to a view.

Two other features of WPF that make this pattern so usable are data templates and the resource system. Data templates apply Views to ViewModel objects shown in the user interface. You can declare templates in XAML and let the resource system automatically locate and apply those templates for you at run time. You can learn more about binding and data templates in my July 2008 article, "[Data and WPF: Customize Data Display with Data Binding and WPF](#)."

If it were not for the support for commands in WPF, the MVVM pattern would be much less powerful. In this article, I will show you how a ViewModel can expose commands to a View, thus allowing the view to consume its functionality. If you aren't familiar with commanding, I recommend that you read Brian Noyes's comprehensive article, "[Advanced WPF: Understanding Routed Events and Commands in WPF](#)," from the September 2008 issue.

In addition to the WPF (and Silverlight 2) features that make MVVM a natural way to structure an application, the pattern is also popular because ViewModel classes are easy to unit test. When an application's interaction logic lives in a set of ViewModel classes, you can easily write code that tests it. In a sense, Views and unit tests are just two different types of ViewModel consumers. Having a suite of tests for an application's ViewModels provides free and fast regression testing, which helps reduce the cost of maintaining an application over time.

In addition to promoting the creation of automated regression tests, the testability of ViewModel classes can assist in properly designing user interfaces that are easy to skin. When you are designing an application, you can often decide whether something should be in the view or the ViewModel by imagining that you want to write a unit test to consume the ViewModel. If you can write unit tests for the ViewModel without creating any UI objects, you can also completely skin the ViewModel because it has no dependencies on specific visual elements.

Lastly, for developers who work with visual designers, using MVVM makes it much easier to create a smooth designer/developer workflow. Since a view is just an arbitrary consumer of a ViewModel, it is easy to just rip one view out and drop in a new view to render a ViewModel. This simple step allows for rapid prototyping and evaluation of user interfaces made by the designers.

The development team can focus on creating robust ViewModel classes, and the design team can focus on making user-friendly Views. Connecting the output of both teams can involve little more than ensuring that the correct bindings exist in a view's XAML file.

## The Demo Application

At this point, I have reviewed MVVM's history and theory of operation. I also examined why it is so popular amongst WPF developers. Now it is time to roll up your sleeves and see the pattern in action. The demo application that accompanies this article uses MVVM in a variety of ways. It provides a fertile source of examples to help put the concepts into a meaningful context. I created the demo application in Visual Studio 2008 SP1, against the Microsoft .NET Framework 3.5 SP1. The unit tests run in the Visual Studio unit testing system.

The application can contain any number of "workspaces," each of which the user can open by clicking on a command link in the navigation area on the left. All workspaces live in a TabControl on the main content area. The user can close a workspace by clicking the Close button on that workspace's tab item. The application has two available workspaces: "All Customers" and "New Customer." After running the application and opening some workspaces, the UI looks something like **Figure 1**.

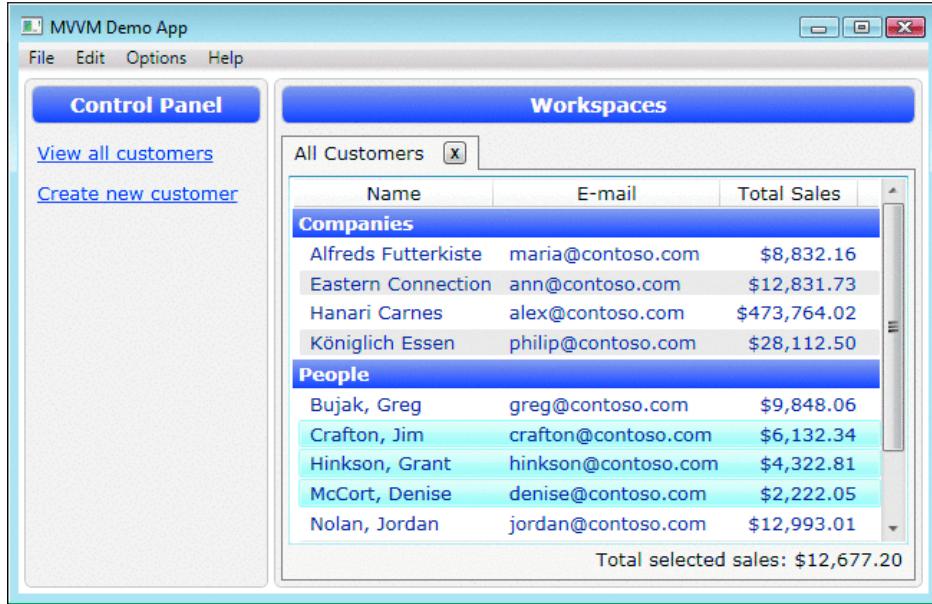


Figure 1 Workspaces

Only one instance of the "All Customers" workspace can be open at a time, but any number of "New Customer" workspaces can be open at once. When the user decides to create a new customer, she must fill in the data entry form in **Figure 2**.

Figure 2 New Customer Data Entry Form

After filling in the data entry form with valid values and clicking the Save button, the new customer's name appears in the tab item and that customer is added to the list of all customers. The application does not have support for deleting or editing an existing customer, but that functionality, and many other features similar to it, are easy to implement by building on top of the existing application architecture. Now that you have a high-level understanding of what the demo application does, let's investigate how it was designed and implemented.

### Relying Command Logic

Every view in the app has an empty codebehind file, except for the standard boilerplate code that calls InitializeComponent in the class's constructor. In fact, you could remove the views' codebehind files from the project and the application would still compile and run correctly. Despite the lack of event handling methods in the views, when the user clicks on buttons, the application reacts and satisfies the user's requests. This works because of bindings that were established on the Command property of Hyperlink, Button, and MenuItem controls displayed in the UI. Those bindings ensure that when the user clicks on the controls, ICommand objects exposed by the ViewModel execute. You can think of the command object as an adapter that makes it easy to consume a ViewModel's functionality from a view declared in XAML.

When a ViewModel exposes an instance property of type ICommand, the command object typically uses that ViewModel object to get its job done. One possible implementation pattern is to create a private nested class within the ViewModel class, so that the command has access to private members of its containing ViewModel and does not pollute the namespace. That nested class implements the ICommand interface, and a reference

to the containing ViewModel object is injected into its constructor. However, creating a nested class that implements ICommand for each command exposed by a ViewModel can bloat the size of the ViewModel class. More code means a greater potential for bugs.

In the demo application, the RelayCommand class solves this problem. RelayCommand allows you to inject the command's logic via delegates passed into its constructor. This approach allows for terse, concise command implementation in ViewModel classes. RelayCommand is a simplified variation of the DelegateCommand found in the [Microsoft Composite Application Library](#). The RelayCommand class is shown in **Figure 3**.

**Figure 3 The RelayCommand Class**

```
public class RelayCommand : ICommand
{
    #region Fields

    readonly Action<object> _execute;
    readonly Predicate<object> _canExecute;

    #endregion // Fields

    #region Constructors

    public RelayCommand(Action<object> execute)
        : this(execute, null)
    {
    }

    public RelayCommand(Action<object> execute, Predicate<object> canExecute)
    {
        if (execute == null)
            throw new ArgumentNullException("execute");

        _execute = execute;
        _canExecute = canExecute;
    }
    #endregion // Constructors

    #region ICommand Members

    [DebuggerStepThrough]
    public bool CanExecute(object parameter)
    {
        return _canExecute == null ? true : _canExecute(parameter);
    }

    public event EventHandler CanExecuteChanged
    {
        add { CommandManager.RequerySuggested += value; }
        remove { CommandManager.RequerySuggested -= value; }
    }

    public void Execute(object parameter)
    {
        _execute(parameter);
    }

    #endregion // ICommand Members
}
```

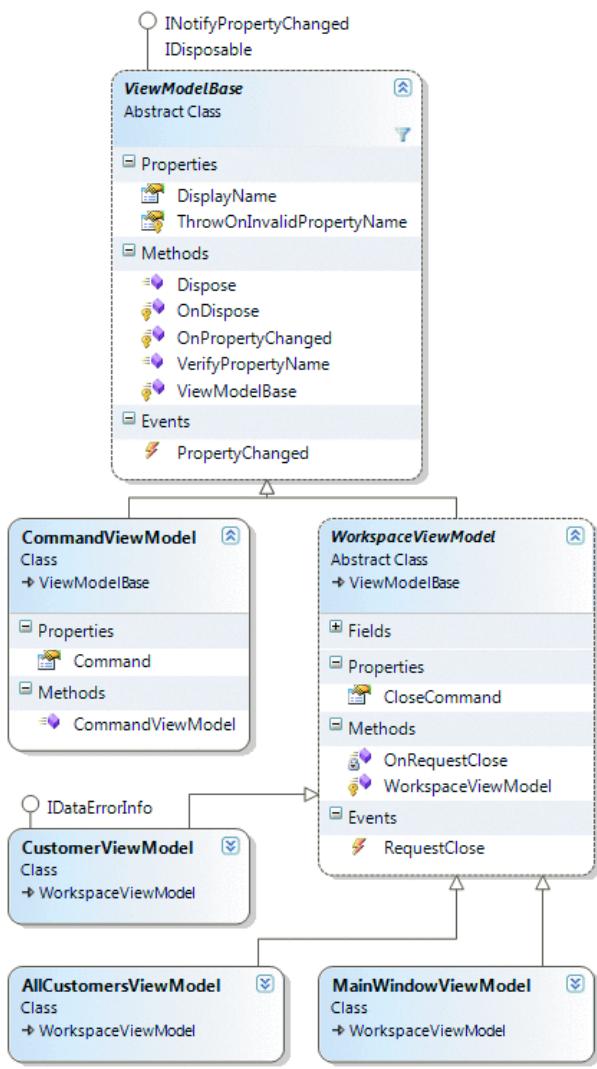
The CanExecuteChanged event, which is part of the ICommand interface implementation, has some interesting features. It delegates the event subscription to the CommandManager.RequerySuggested event. This ensures that the WPF commanding infrastructure asks all RelayCommand objects if they can execute whenever it asks the built-in commands. The following code from the CustomerViewModel class, which I will examine in-depth later, shows how to configure a RelayCommand with lambda expressions:

```
RelayCommand _saveCommand;
public ICommand SaveCommand
{
    get
    {
        if (_saveCommand == null)
        {
            _saveCommand = new RelayCommand(param => this.Save(),
                param => this.CanSave );
        }
        return _saveCommand;
    }
}
```

## ViewModel Class Hierarchy

Most ViewModel classes need the same features. They often need to implement the INotifyPropertyChanged interface, they usually need to have a user-friendly display name, and, in the case of workspaces, they need

the ability to close (that is, be removed from the UI). This problem naturally lends itself to the creation of a ViewModel base class or two, so that new ViewModel classes can inherit all of the common functionality from a base class. The ViewModel classes form the inheritance hierarchy seen in **Figure 4**.



**Figure 4 Inheritance Hierarchy**

Having a base class for all of your ViewModels is by no means a requirement. If you prefer to gain features in your classes by composing many smaller classes together, instead of using inheritance, that is not a problem. Just like any other design pattern, MVVM is a set of guidelines, not rules.

### ViewModelBase Class

**ViewModelBase** is the root class in the hierarchy, which is why it implements the commonly used **INotifyPropertyChanged** interface and has a **DisplayName** property. The **INotifyPropertyChanged** interface contains an event called **PropertyChanged**. Whenever a property on a **ViewModel** object has a new value, it can raise the **PropertyChanged** event to notify the WPF binding system of the new value. Upon receiving that notification, the binding system queries the property, and the bound property on some UI element receives the new value.

In order for WPF to know which property on the **ViewModel** object has changed, the **PropertyChangedEventArgs** class exposes a **PropertyName** property of type **String**. You must be careful to pass the correct property name into that event argument; otherwise, WPF will end up querying the wrong property for a new value.

One interesting aspect of **ViewModelBase** is that it provides the ability to verify that a property with a given name actually exists on the **ViewModel** object. This is very useful when refactoring, because changing a property's name via the Visual Studio 2008 refactoring feature will not update strings in your source code that happen to contain that property's name (nor should it). Raising the **PropertyChanged** event with an incorrect property name in the event argument can lead to subtle bugs that are difficult to track down, so this little feature can be a huge timesaver. The code from **ViewModelBase** that adds this useful support is shown in **Figure 5**.

**Figure 5 Verifying a Property**

```

// In ViewModelBase.cs
public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName)
{
    this.VerifyPropertyName(propertyName);

    PropertyChangedEventHandler handler = this.PropertyChanged;
  
```

```

        if (handler != null)
    {
        var e = new PropertyChangedEventArgs(propertyName);
        handler(this, e);
    }
}

[Conditional("DEBUG")]
[DebuggerStepThrough]
public void VerifyPropertyName(string propertyName)
{
    // Verify that the property name matches a real,
    // public, instance property on this object.
    if (TypeDescriptor.GetProperties(this)[propertyName] == null)
    {
        string msg = "Invalid property name: " + propertyName;

        if (this.ThrowOnInvalidPropertyName)
            throw new Exception(msg);
        else
            Debug.Fail(msg);
    }
}

```

### CommandViewModel Class

The simplest concrete ViewModelBase subclass is CommandViewModel. It exposes a property called Command of type ICommand. MainWindowViewModel exposes a collection of these objects through its Commands property. The navigation area on the left-hand side of the main window displays a link for each CommandViewModel exposed by MainWindowViewModel, such as "View all customers" and "Create new customer." When the user clicks on a link, thus executing one of those commands, a workspace opens in the TabControl on the main window. The CommandViewModel class definition is shown here:

---

```

public class CommandViewModel : ViewModelBase
{
    public CommandViewModel(string displayName, ICommand command)
    {
        if (command == null)
            throw new ArgumentNullException("command");

        base.DisplayName = displayName;
        this.Command = command;
    }

    public ICommand Command { get; private set; }
}

```

---

In the MainWindowResources.xaml file there exists a DataTemplate whose key is "CommandsTemplate". MainWindow uses that template to render the collection of CommandViewModels mentioned earlier. The template simply renders each CommandViewModel object as a link in an ItemsControl. Each Hyperlink's Command property is bound to the Command property of a CommandViewModel. That XAML is shown in **Figure 6**.

**Figure 6** Render the List of Commands

---

```

<!-- In MainWindowResources.xaml -->
<!--
This template explains how to render the list of commands on
the left side in the main window (the 'Control Panel' area).
-->
<DataTemplate x:Key="CommandsTemplate">
    <ItemsControl ItemsSource="{Binding Path=Commands}">
        <ItemsControl.ItemTemplate>
            <DataTemplate>
                <TextBlock Margin="2,6">
                    <Hyperlink Command="{Binding Path=Command}">
                        <TextBlock Text="{Binding Path=DisplayName}" />
                    </Hyperlink>
                </TextBlock>
            </DataTemplate>
        </ItemsControl.ItemTemplate>
    </ItemsControl>
</DataTemplate>

```

---

### MainWindowViewModel Class

As previously seen in the class diagram, the WorkspaceViewModel class derives from ViewModelBase and adds the ability to close. By "close," I mean that something removes the workspace from the user interface at run time. Three classes derive from WorkspaceViewModel: MainWindowViewModel, AllCustomersViewModel, and CustomerViewModel. MainWindowViewModel's request to close is handled by the App class, which creates the MainWindow and its ViewModel, as seen in **Figure 7**.

**Figure 7 Create the ViewModel**

```
// In App.xaml.cs
protected override void OnStartup(StartupEventArgs e)
{
    base.OnStartup(e);

    MainWindow window = new MainWindow();

    // Create the ViewModel to which
    // the main window binds.
    string path = "Data/customers.xml";
    var viewModel = new MainWindowViewModel(path);

    // When the ViewModel asks to be closed,
    // close the window.
    viewModel.RequestClose += delegate
    {
        window.Close();
    };

    // Allow all controls in the window to
    // bind to the ViewModel by setting the
    // DataContext, which propagates down
    // the element tree.
    window.DataContext = viewModel;

    window.Show();
}
```

MainWindow contains a menu item whose Command property is bound to the MainWindowViewModel's CloseCommand property. When the user clicks on that menu item, the App class responds by calling the window's Close method, like so:

```
<!-- In MainWindow.xaml -->
<Menu>
    <MenuItem Header="_File">
        <MenuItem Header="_Exit" Command="{Binding Path=CloseCommand}" />
    </MenuItem>
    <MenuItem Header="_Edit" />
    <MenuItem Header="_Options" />
    <MenuItem Header="_Help" />
</Menu>
```

MainWindowViewModel contains an observable collection of WorkspaceViewModel objects, called Workspaces. The main window contains a TabControl whose ItemsSource property is bound to that collection. Each tab item has a Close button whose Command property is bound to the CloseCommand of its corresponding WorkspaceViewModel instance. An abridged version of the template that configures each tab item is shown in the code that follows. The code is found in MainWindowResources.xaml, and the template explains how to render a tab item with a Close button:

```
<DataTemplate x:Key="ClosableTabItemTemplate">
    <DockPanel Width="120">
        <Button
            Command="{Binding Path=CloseCommand}"
            Content="X"
            DockPanel.Dock="Right"
            Width="16" Height="16"
            />
        <ContentPresenter Content="{Binding Path=DisplayName}" />
    </DockPanel>
</DataTemplate>
```

When the user clicks the Close button in a tab item, that WorkspaceViewModel's CloseCommand executes, causing its RequestClose event to fire. MainWindowViewModel monitors the RequestClose event of its workspaces and removes the workspace from the Workspaces collection upon request. Since the Main-Window's TabControl has its ItemsSource property bound to the observable collection of WorkspaceViewModels, removing an item from the collection causes the corresponding workspace to be removed from the TabControl. That logic from MainWindowViewModel is shown in **Figure 8**.

**Figure 8 Removing Workspace from the UI**

```
// In MainWindowViewModel.cs

ObservableCollection<WorkspaceViewModel> _workspaces;

public ObservableCollection<WorkspaceViewModel> Workspaces
{
    get
    {
        if (_workspaces == null)
```

```

    {
        _workspaces = new ObservableCollection<WorkspaceViewModel>();
        _workspaces.CollectionChanged += this.OnWorkspacesChanged;
    }
    return _workspaces;
}

void OnWorkspacesChanged(object sender, NotifyCollectionChangedEventArgs e)
{
    if (e.NewItems != null && e.NewItems.Count != 0)
        foreach (WorkspaceViewModel workspace in e.NewItems)
            workspace.RequestClose += this.OnWorkspaceRequestClose;

    if (e.OldItems != null && e.OldItems.Count != 0)
        foreach (WorkspaceViewModel workspace in e.OldItems)
            workspace.RequestClose -= this.OnWorkspaceRequestClose;
}

void OnWorkspaceRequestClose(object sender, EventArgs e)
{
    this.Workspaces.Remove(sender as WorkspaceViewModel);
}

```

In the UnitTests project, the MainWindowViewModelTests.cs file contains a test method that verifies that this functionality is working properly. The ease with which you can create unit tests for ViewModel classes is a huge selling point of the MVVM pattern, because it allows for simple testing of application functionality without writing code that touches the UI. That test method is shown in **Figure 9**.

**Figure 9** The Test Method

```

// In MainWindowViewModelTests.cs
[TestMethod]
public void TestCloseAllCustomersWorkspace()
{
    // Create the MainWindowViewModel, but not the MainWindow.
    MainWindowViewModel target =
        new MainWindowViewModel(Constants.CUSTOMER_DATA_FILE);

    Assert.AreEqual(0, target.Workspaces.Count, "Workspaces isn't empty.");

    // Find the command that opens the "All Customers" workspace.
    CommandViewModel commandVM =
        target.Commands.First(cvm => cvm.DisplayName == "View all customers");

    // Open the "All Customers" workspace.
    commandVM.Command.Execute(null);
    Assert.AreEqual(1, target.Workspaces.Count, "Did not create viewmodel.");

    // Ensure the correct type of workspace was created.
    var allCustomersVM = target.Workspaces[0] as AllCustomersViewModel;
    Assert.IsNotNull(allCustomersVM, "Wrong viewmodel type created.");

    // Tell the "All Customers" workspace to close.
    allCustomersVM.CloseCommand.Execute(null);
    Assert.AreEqual(0, target.Workspaces.Count, "Did not close viewmodel.");
}

```

## Applying a View to a ViewModel

MainWindowViewModel indirectly adds and removes WorkspaceViewModel objects to and from the main window's TabControl. By relying on data binding, the Content property of a TabItem receives a ViewModelBase-derived object to display. ViewModelBase is not a UI element, so it has no inherent support for rendering itself. By default, in WPF a non-visual object is rendered by displaying the results of a call to its ToString method in a TextBlock. That clearly is not what you need, unless your users have a burning desire to see the type name of our ViewModel classes!

You can easily tell WPF how to render a ViewModel object by using typed DataTemplates. A typed DataTemplate does not have an x:Key value assigned to it, but it does have its DataType property set to an instance of the Type class. If WPF tries to render one of your ViewModel objects, it will check to see if the resource system has a typed DataTemplate in scope whose DataType is the same as (or a base class of) the type of your ViewModel object. If it finds one, it uses that template to render the ViewModel object referenced by the tab item's Content property.

The MainWindowResources.xaml file has a ResourceDictionary. That dictionary is added to the main window's resource hierarchy, which means that the resources it contains are in the window's resource scope. When a tab item's content is set to a ViewModel object, a typed DataTemplate from this dictionary supplies a view (that is, a user control) to render it, as shown in **Figure 10**.

**Figure 10** Supplying a View

```

<!--
This resource dictionary is used by the MainWindow.
-->
<ResourceDictionary

```

```

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:vm="clr-namespace:DemoApp.ViewModel"
xmlns:vw="clr-namespace:DemoApp.View"
>

<!--
This template applies an AllCustomersView to an instance
of the AllCustomersViewModel class shown in the main window.
-->
<DataTemplate DataType="{x:Type vm:AllCustomersViewModel}">
    <vw:AllCustomersView />
</DataTemplate>

<!--
This template applies a CustomerView to an instance
of the CustomerViewModel class shown in the main window.
-->
<DataTemplate DataType="{x:Type vm:CustomerViewModel}">
    <vw:CustomerView />
</DataTemplate>

<!-- Other resources omitted for clarity... -->

</ResourceDictionary>

```

You do not need to write any code that determines which view to show for a ViewModel object. The WPF resource system does all of the heavy lifting for you, freeing you up to focus on more important things. In more complex scenarios, it is possible to programmatically select the view, but in most situations that is unnecessary.

## The Data Model and Repository

You have seen how ViewModel objects are loaded, displayed, and closed by the application shell. Now that the general plumbing is in place, you can review implementation details more specific to the domain of the application. Before getting deep into the application's two workspaces, "All Customers" and "New Customer," let's first examine the data model and data access classes. The design of those classes has almost nothing to do with the MVVM pattern, because you can create a ViewModel class to adapt just about any data object into something friendly to WPF.

The sole model class in the demo program is *Customer*. That class has a handful of properties that represent information about a customer of a company, such as their first name, last name, and e-mail address. It provides validation messages by implementing the standard *IDataErrorInfo* interface, which existed for years before WPF hit the street. The *Customer* class has nothing in it that suggests it is being used in an MVVM architecture or even in a WPF application. The class could easily have come from a legacy business library.

Data must come from and reside somewhere. In this application, an instance of the *CustomerRepository* class loads and stores all *Customer* objects. It happens to load the customer data from an XML file, but the type of external data source is irrelevant. The data could come from a database, a Web service, a named pipe, a file on disk, or even carrier pigeons: it simply does not matter. As long as you have a .NET object with some data in it, regardless of where it came from, the MVVM pattern can get that data on the screen.

The *CustomerRepository* class exposes a few methods that allow you to get all the available *Customer* objects, add new a *Customer* to the repository, and check if a *Customer* is already in the repository. Since the application does not allow the user to delete a customer, the repository does not allow you to remove a customer. The *CustomerAdded* event fires when a new *Customer* enters the *CustomerRepository*, via the *AddCustomer* method.

Clearly, this application's data model is very small, compared to what real business applications require, but that is not important. What is important to understand is how the ViewModel classes make use of *Customer* and *CustomerRepository*. Note that *CustomerViewModel* is a wrapper around a *Customer* object. It exposes the state of a *Customer*, and other state used by the *CustomerView* control, through a set of properties. *CustomerViewModel* does not duplicate the state of a *Customer*; it simply exposes it via delegation, like this:

---

```

public string FirstName
{
    get { return _customer.FirstName; }
    set
    {
        if (value == _customer.FirstName)
            return;
        _customer.FirstName = value;
        base.OnPropertyChanged("FirstName");
    }
}

```

When the user creates a new customer and clicks the Save button in the *CustomerView* control, the *CustomerViewModel* associated with that view will add the new *Customer* object to the *CustomerRepository*. That causes the repository's *CustomerAdded* event to fire, which lets the *AllCustomersViewModel* know that it should add a new *CustomerViewModel* to its *AllCustomers* collection. In a sense, *CustomerRepository* acts as a synchronization mechanism between various ViewModels that deal with *Customer* objects. Perhaps one might think of this as using the Mediator design pattern. I will review more of how this works in the upcoming sections, but for now refer to the diagram in **Figure 11** for a high-level understanding of how all the pieces fit together.

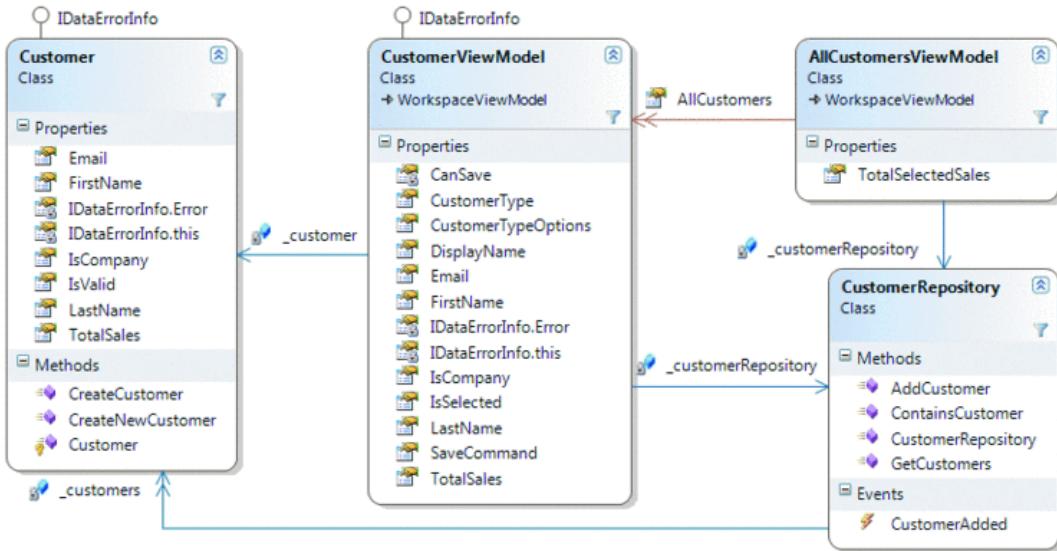


Figure 11 Customer Relationships

### New Customer Data Entry Form

When the user clicks the "Create new customer" link, MainWindowViewModel adds a new CustomerViewModel to its list of workspaces, and a CustomerView control displays it. After the user types valid values into the input fields, the Save button enters the enabled state so that the user can persist the new customer information. There is nothing out of the ordinary here, just a regular data entry form with input validation and a Save button.

The Customer class has built-in validation support, available through its IDataErrorInfo interface implementation. That validation ensures the customer has a first name, a well-formed e-mail address, and, if the customer is a person, a last name. If the Customer's IsCompany property returns true, the LastName property cannot have a value (the idea being that a company does not have a last name). This validation logic might make sense from the Customer object's perspective, but it does not meet the needs of the user interface. The UI requires a user to select whether a new customer is a person or a company. The Customer Type selector initially has the value "(Not Specified)". How can the UI tell the user that the customer type is unspecified if the IsCompany property of a Customer only allows for a true or false value?

Assuming you have complete control over the entire software system, you could change the IsCompany property to be of type Nullable<bool>, which would allow for the "unselected" value. However, the real world is not always so simple. Suppose you cannot change the Customer class because it comes from a legacy library owned by a different team in your company. What if there is no easy way to persist that "unselected" value because of the existing database schema? What if other applications already use the Customer class and rely on the property being a normal Boolean value? Once again, having a ViewModel comes to the rescue.

The test method in **Figure 12** shows how this functionality works in CustomerViewModel. CustomerViewModel exposes a CustomerTypeOptions property so that the Customer Type selector has three strings to display. It also exposes a CustomerType property, which stores the selected String in the selector. When CustomerType is set, it maps the String value to a Boolean value for the underlying Customer object's IsCompany property. **Figure 13** shows the two properties.

Figure 12 The Test Method

```

// In CustomerViewModelTests.cs
[TestMethod]
public void TestCustomerType()
{
    Customer cust = Customer.CreateNewCustomer();
    CustomerRepository repos = new CustomerRepository(
        Constants.CUSTOMER_DATA_FILE);
    CustomerViewModel target = new CustomerViewModel(cust, repos);

    target.CustomerType = "Company";
    Assert.IsTrue(cust.IsCompany, "Should be a company");

    target.CustomerType = "Person";
    Assert.IsFalse(cust.IsCompany, "Should be a person");

    target.CustomerType = "(Not Specified)";
    string error = (target as IDataErrorInfo)["CustomerType"];
    Assert.IsFalse(String.IsNullOrEmpty(error), "Error message should
        be returned");
}
  
```

Figure 13 CustomerType Properties

```

// In CustomerViewModel.cs

public string[] CustomerTypeOptions
{
  
```

```

    get
    {
        if (_customerTypeOptions == null)
        {
            _customerTypeOptions = new string[]
            {
                "(Not Specified)",
                "Person",
                "Company"
            };
        }
        return _customerTypeOptions;
    }
}

public string CustomerType
{
    get { return _customerType; }
    set
    {
        if (value == _customerType || String.IsNullOrEmpty(value))
            return;

        _customerType = value;

        if (_customerType == "Company")
        {
            _customer.IsCompany = true;
        }
        else if (_customerType == "Person")
        {
            _customer.IsCompany = false;
        }

        base.OnPropertyChanged("CustomerType");
        base.OnPropertyChanged("LastName");
    }
}
}

```

The CustomerView control contains a ComboBox that is bound to those properties, as seen here:

---

```
<ComboBox
    ItemsSource="{Binding CustomerTypeOptions}"
    SelectedItem="{Binding CustomerType, ValidatesOnDataErrors=True}"
/>
```

When the selected item in that ComboBox changes, the data source's `IDataErrorInfo` interface is queried to see if the new value is valid. That occurs because the `SelectedItem` property binding has `ValidatesOnDataErrors` set to true. Since the data source is a `CustomerViewModel` object, the binding system asks that `CustomerViewModel` for a validation error on the `CustomerType` property. Most of the time, `CustomerViewModel` delegates all requests for validation errors to the `Customer` object it contains. However, since `Customer` has no notion of having an unselected state for the `IsCompany` property, the `CustomerViewModel` class must handle validating the new selected item in the ComboBox control. That code is seen in **Figure 14**.

**Figure 14 Validating a CustomerViewModel Object**

---

```
// In CustomerViewModel.cs
string IDataErrorInfo.this[string propertyName]
{
    get
    {
        string error = null;

        if (propertyName == "CustomerType")
        {
            // The IsCompany property of the Customer class
            // is Boolean, so it has no concept of being in
            // an "unselected" state. The CustomerViewModel
            // class handles this mapping and validation.
            error = this.ValidateCustomerType();
        }
        else
        {
            error = (_customer as IDataErrorInfo)[propertyName];
        }

        // Dirty the commands registered with CommandManager,
        // such as our Save command, so that they are queried
        // to see if they can execute now.
        CommandManager.InvalidateRequerySuggested();

        return error;
    }
}
```

```

string ValidateCustomerType()
{
    if (this.CustomerType == "Company" ||
        this.CustomerType == "Person")
        return null;

    return "Customer type must be selected";
}

```

The key aspect of this code is that CustomerViewModel's implementation of IDataErrorInfo can handle requests for ViewModel-specific property validation and delegate the other requests to the Customer object. This allows you to make use of validation logic in Model classes and have additional validation for properties that only make sense to ViewModel classes.

The ability to save a CustomerViewModel is available to a view through the SaveCommand property. That command uses the RelayCommand class examined earlier to allow CustomerViewModel to decide if it can save itself and what to do when told to save its state. In this application, saving a new customer simply means adding it to a CustomerRepository. Deciding if the new customer is ready to be saved requires consent from two parties. The Customer object must be asked if it is valid or not, and the CustomerViewModel must decide if it is valid. This two-part decision is necessary because of the ViewModel-specific properties and validation examined previously. The save logic for CustomerViewModel is shown in **Figure 15**.

**Figure 15** The Save Logic for CustomerViewModel

```

// In CustomerViewModel.cs
public ICommand SaveCommand
{
    get
    {
        if (_saveCommand == null)
        {
            _saveCommand = new RelayCommand(
                param => this.Save(),
                param => this.CanSave
            );
        }
        return _saveCommand;
    }
}

public void Save()
{
    if (!_customer.IsValid)
        throw new InvalidOperationException("...");

    if (this.IsNewCustomer)
        _customerRepository.AddCustomer(_customer);

    base.OnPropertyChanged("DisplayName");
}

bool IsNewCustomer
{
    get
    {
        return !_customerRepository.ContainsCustomer(_customer);
    }
}

bool CanSave
{
    get
    {
        return
            String.IsNullOrEmpty(this.ValidateCustomerType()) &&
            _customer.IsValid;
    }
}

```

The use of a ViewModel here makes it much easier to create a view that can display a Customer object and allow for things like an "unselected" state of a Boolean property. It also provides the ability to easily tell the customer to save its state. If the view were bound directly to a Customer object, the view would require a lot of code to make this work properly. In a well-designed MVVM architecture, the codebehind for most Views should be empty, or, at most, only contain code that manipulates the controls and resources contained within that view. Sometimes it is also necessary to write code in a View's codebehind that interacts with a ViewModel object, such as hooking an event or calling a method that would otherwise be very difficult to invoke from the ViewModel itself.

## All Customers View

The demo application also contains a workspace that displays all of the customers in a ListView. The customers in the list are grouped according to whether they are a company or a person. The user can select one or more customers at a time and view the sum of their total sales in the bottom right corner.

The UI is the AllCustomersView control, which renders an AllCustomersViewModel object. Each ListViewItem

represents a CustomerViewModel object in the AllCustomers collection exposed by the AllCustomerViewModel object. In the previous section, you saw how a CustomerViewModel can render as a data entry form, and now the exact same CustomerViewModel object is rendered as an item in a ListView. The CustomerViewModel class has no idea what visual elements display it, which is why this reuse is possible. AllCustomersView creates the groups seen in the ListView. It accomplishes this by binding the ListView's ItemsSource to a CollectionViewSource configured like **Figure 16**.

**Figure 16 CollectionViewSource**

---

```
<!-- In AllCustomersView.xaml -->
<CollectionViewSource
  x:Key="CustomerGroups"
  Source="{Binding Path=AllCustomers}"
>
<CollectionViewSource.GroupDescriptions>
  <PropertyGroupDescription PropertyName="IsCompany" />
</CollectionViewSource.GroupDescriptions>
<CollectionViewSource.SortDescriptions>
  <!--
    Sort descending by IsCompany so that the 'True' values appear first,
    which means that companies will always be listed before people.
  -->
  <scm:SortDescription PropertyName="IsCompany" Direction="Descending" />
  <scm:SortDescription PropertyName="DisplayName" Direction="Ascending" />
</CollectionViewSource.SortDescriptions>
</CollectionViewSource>
```

---

The association between a ListViewItem and a CustomerViewModel object is established by the ListView's ItemContainerStyle property. The Style assigned to that property is applied to each ListViewItem, which enables properties on a ListViewItem to be bound to properties on the CustomerViewModel. One important binding in that Style creates a link between the IsSelected property of a ListViewItem and the IsSelected property of a CustomerViewModel, as seen here:

---

```
<Style x:Key="CustomerItemStyle" TargetType="{x:Type ListViewItem}">
  <!-- Stretch the content of each cell so that we can
      right-align text in the Total Sales column. -->
  <Setter Property="HorizontalContentAlignment" Value="Stretch" />
  <!--
    Bind the IsSelected property of a ListViewItem to the
    IsSelected property of a CustomerViewModel object.
  -->
  <Setter Property="IsSelected" Value="{Binding Path=IsSelected,
    Mode=TwoWay}" />
</Style>
```

---

When a CustomerViewModel is selected or unselected, that causes the sum of all selected customers' total sales to change. The AllCustomersViewModel class is responsible for maintaining that value, so that the ContentPresenter beneath the ListView can display the correct number. **Figure 17** shows how AllCustomersViewModel monitors each customer for being selected or unselected and notifies the view that it needs to update the display value.

**Figure 17 Monitoring for Selected or Unselected**

---

```
// In AllCustomersViewModel.cs
public double TotalSelectedSales
{
    get
    {
        return this.AllCustomers.Sum(
            custVM => custVM.IsSelected ? custVM.TotalSales : 0.0);
    }
}

void OnCustomerViewModelPropertyChanged(object sender,
    PropertyChangedEventArgs e)
{
    string IsSelected = "IsSelected";

    // Make sure that the property name we're
    // referencing is valid. This is a debugging
    // technique, and does not execute in a Release build.
    (sender as CustomerViewModel).VerifyPropertyName(IsSelected);

    // When a customer is selected or unselected, we must let the
    // world know that the TotalSelectedSales property has changed,
    // so that it will be queried again for a new value.
    if (e.PropertyName == IsSelected)
        this.OnPropertyChanged("TotalSelectedSales");
}
```

---

The UI binds to the TotalSelectedSales property and applies currency (monetary) formatting to the value. The ViewModel object could apply the currency formatting, instead of the view, by returning a String instead of a

Double value from the TotalSelectedSales property. The ContentStringFormat property of ContentPresenter was added in the .NET Framework 3.5 SP1, so if you must target an older version of WPF, you will need to apply the currency formatting in code:

```
<!-- In AllCustomersView.xaml -->
<StackPanel Orientation="Horizontal">
    <TextBlock Text="Total selected sales: " />
    <ContentPresenter
        Content="{Binding Path=TotalSelectedSales}"
        ContentStringFormat="c"
    />
</StackPanel>
```

## Wrapping Up

WPF has a lot to offer application developers, and learning to leverage that power requires a mindset shift. The Model-View-ViewModel pattern is a simple and effective set of guidelines for designing and implementing a WPF application. It allows you to create a strong separation between data, behavior, and presentation, making it easier to control the chaos that is software development.

*I would like to thank John Gossman for his help with this article.*

**Josh Smith** is passionate about using WPF to create great user experiences. He was awarded the Microsoft MVP title for his work in the WPF community. Josh works for Infragistics in the Experience Design Group. When he is not at a computer, he enjoys playing the piano, reading about history, and exploring New York City with his girlfriend. You can visit Josh's blog at [joshsmithonwpf.wordpress.com](http://joshsmithonwpf.wordpress.com).

## Technical Recognition Awards

TECHNICAL COMMUNITY NETWORK

[Home](#) | [Hall of Legends](#) | [Charity Outreach](#) | [About](#) | [FAQ](#)

[◀ Back to Outstanding Technical Achievement](#)



### Outstanding Technical Achievement

This team award recognizes an outstanding and innovative technical achievement that has profoundly transformed the world of software and addressed some of the most urgent technological challenges facing the world today.



#### 2012 AWARD RECIPIENTS

[Kinect Skeletal Tracking Team,](#)  
[Outstanding Technical Achievement](#)

[Alex Kipman,](#)  
[Outstanding Technical Leadership](#)

## Kinect Skeletal Tracking

Momin Al-Ghosien, Matt Brodner, Robert Craig, Mark Finocchio, Alex Kipman, Samuel Mann, Parham Mohadjer, Craig Peeler, and Jamie Shotton

### 2012 Outstanding Technical Achievement

Kinect isn't just the fastest selling consumer electronic device, it is the best example yet of a magical experience that comes when the computer knows you instead of you having to learn about your computer.

The magic of Kinect was predicated on the amazing work of people to bring together the capacity to economically manufacture Kinect at scale, to make the Kinect understand when you talk, know who you are when you walk up to it, and be able to interpret your movements and translate into a format where developers can build experiences that were never before imagined. This magic starts with Skeletal Tracking, and Microsoft honors the Kinect Skeletal Tracking team members as the recipient of the 2012 Outstanding Technical Achievement award.

The operational envelope demands for commercially viable Skeletal Tracking is enormous. Simply put, Skeletal Tracking must ideally work for every person on the planet, in every household, and without any calibration. A dauntingly high number of dimensions describe this envelope, such as distance from the Kinect sensor and the sensor tilt angle. Entire sets of dimensions are necessary to describe the unique individuals including size, shape, hair, clothing, motions, and poses. Household environment dimensions are also necessary for lighting, furniture and other household furnishings, and pets.

The fields of machine vision and learning began decades ago, advancing greatly in the last 20 years, yet commercially viable Skeletal Tracking was an academic dream. To bring the vision and possibilities demonstrated by Alex Kipman's incubation group to shipping reality, Robert Craig formed and led the Skeleton Crew, a virtual team across the Interactive Entertainment Business and Microsoft Research divisions. Product engineers, graphics and vision experts, machine learning theorists, and mathematicians were brought together to rapidly explore solutions to Kinect's many open challenges.

Delivering Skeletal Tracking required its core team members to be scientist engineers, conducting research with rigorous application of the scientific method, while delivering production-quality solutions. The result is a tracking pipeline capable of filling Kinect's ambitious operational envelope. Leveraged by the outstanding creativity of application developers, magical never-before-seen experiences emerge.

Skeletal Tracking shipped to Xbox developers in Fall 2009, and Kinect shipped for the 2010 Holiday Season and became the fastest selling electronic device of all time. Skeletal Tracking's innovation continues since Kinect's launch. The Kinect for Windows SDK includes APIs, sample code and drivers. Skeletal Tracking's operational envelope has also expanded to enable developers to create experiences in a seated position on Xbox 360 for Holiday 2011. Improving Skeletal Tracking quality opens possibilities to more developers, applications, and industries, going well beyond Kinect's inception in living room entertainment.

[Contact Us](#) | [Terms of Use](#) | [Trademarks](#) | [Privacy Statement](#) | © 2010 Microsoft



# Windows 2000 Display Driver Model (XDDM) Design Guide

Introduction to Display (Windows 2000)  
Display Drivers (Windows 2000 Model)  
DirectDraw  
Direct3D  
DirectX Video Acceleration  
Video Miniport Drivers in the Windows 2000 Display Driver Model  
Implementation Tips and Requirements for the Microsoft Windows Vista Display Driver Model  
GDI

This topic has not yet been rated - [Rate this topic](#)

Display adapter drivers that run on Windows Vista can adhere to one of two models: the [Windows Vista display driver model \(WDDM\)](#) or the Microsoft Windows 2000 display driver model (XDDM). Drivers that adhere to the Windows Vista display driver model run only on the Windows Vista operating system. Drivers that adhere to the Windows 2000 display driver model run on Windows 2000 and later operating systems (including Windows Vista).

The following sections describe the Windows 2000 display driver model:

- [Introduction to Display \(Windows 2000 Model\)](#)
- [Display Drivers \(Windows 2000 Model\)](#)
- [DirectDraw](#)
- [Direct3D](#)
- [DirectX Video Acceleration](#)
- [Video Miniport Drivers in the Windows 2000 Display Driver Model](#)
- [Implementation Tips and Requirements for the Microsoft Windows Vista Display Driver Model](#)
- [GDI](#)

**Note** The documentation for the Windows 2000 display driver model no longer includes information about how to create a display driver that runs on the Microsoft Windows 98/Me platforms. If you want to create a display driver for Windows 98/Me, you can use the WDK documentation that released with Windows Vista. You can obtain the WDK for Windows Vista RTM from the [Microsoft Connect website](#).

[Send comments about this topic to Microsoft](#)

Build date: 5/20/2012

Did you find this helpful?  Yes  No



© 2012 Microsoft. All rights reserved.

[Terms of use](#) | [Trademarks](#) | [Privacy statement](#) | [Site feedback](#) | [United States \(English\)](#)

[Subscribe to Project](#)[HOME](#)[DOWNLOADS](#)[DOCUMENTATION](#)[DISCUSSIONS](#)[ISSUE TRACKER](#)[SOURCE CODE](#)[PEOPLE](#)[LICENSE](#)[Page Info](#) | [Change History \(all pages\)](#)1163 people following this project ([follow](#))**MEF in web and .NET for Metro style apps:**

- » Please read the [migration guide](#)
- » [Avoiding reference \(memory\) leaks in MEF for Metro style apps](#)
- » Creating a [Standalone Web API dependency resolver using Microsoft.Composition](#)

[Search Wiki & Documentation](#)**SYSTEM REQUIREMENTS**

There are currently no defined requirements.

**Important:** the documentation below is for [preview versions](#) of MEF. Most users should use the [introductory](#) and [reference documentation on MSDN](#) (more).

- » Learn more about MEF
- » [Read the MEF Programming Guide](#)
- » [Read our sample documentation](#)
- » [Architecture Overview](#)

*Please bear with us while we move the remaining 'version 1' documentation from this site to the .NET Framework Developer Center on MSDN.*

Last edited Fri at 9:05 PM by [nblumhardt](#), version 8

**COMMENTS**

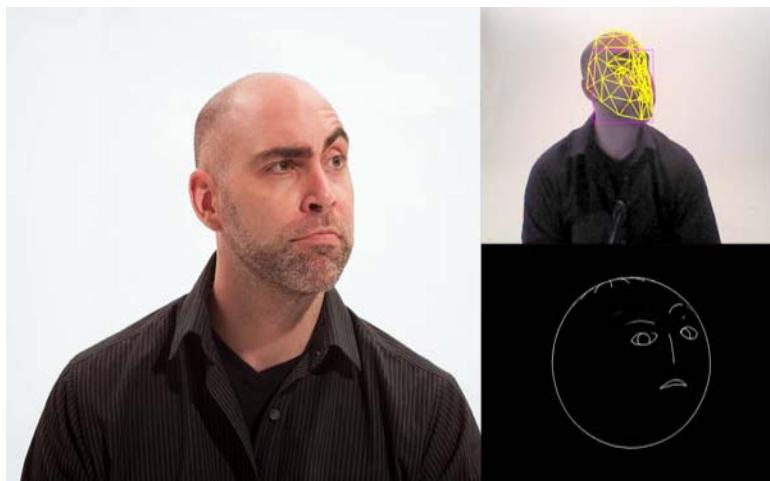
No comments yet.

[Sign in to add a comment](#)

[HOME](#) [DISCOVER](#) [PURCHASE](#) [NEWS](#) [PARTNERS](#)[DEVELOP](#)[what's new](#)[downloads](#)[learn](#)[support](#)FOLLOW US: [blog](#)Search This Site 

## LEARNING RESOURCES AND DOCUMENTATION

These resources can help you start using the Kinect for Windows SDK and sensor to develop applications and further utilize the Kinect natural user interface capabilities.



### Get started

Download the Human Interface Guidelines, access documentation on MSDN, and view videos to help you get started quickly using the Kinect for Windows SDK and sensor.

#### Kinect for Windows SDK documentation

View programming guide, sample browser, reference guides, and more for the Kinect for Windows SDK v1.5

#### Kinect for Windows Human Interface Guidelines

These comprehensive guidelines present how to design interactions and interfaces for Kinect for Windows applications.

#### Developer How-to Videos

Provides step-by-step instructions, using Visual Basic or C#, for installing, using, and setting up development environment, working with depth data, the camera, audio streams, and skeletal tracking

#### Frequently Asked Questions

Get answers to commonly asked questions about the Kinect for Windows hardware, software, and runtime

### New to Windows Development

- Beginner Developer Learning Center for Windows Development

### Visual Studio

- Getting Started with Visual Studio
- Resources for Visual C# Express
- Visual C++ Developer Center
- How Do I Use the Visual Studio 2010 Integrated Development Environment (IDE)

### Game Development

- Microsoft Speech
- DirectX Developer Center
- XNA, Xbox, and Game Development

### Kinect for Windows Sensor

The Kinect for Windows sensor is designed to be used with Kinect for Windows SDK.

[PURCHASE](#)

### Kinect for Windows SDK

Download the software development kit (SDK) and start developing today.

[DOWNLOAD NOW](#)

### Top Developer Activities

[Access Kinect for Windows learning resources](#) ›  
[View documentation, licensing agreements, and more](#)

[Participate in Kinect for Windows communities](#) ›  
Visit technical forums on MSDN, read blog posts, and visit Coding4Fun.

[Get answers to Kinect for Windows questions](#) ›  
View the Kinect for Windows developer FAQ.

[View Kinect for Windows how-to videos](#) ›  
These videos also include links to in-depth articles and resources.

### Support

[Get technical support](#) ›

### Career opportunities

[Join the Kinect for Windows team](#) ›



# Threading Model

## .NET Framework 4

24 out of 30 rated this helpful - [Rate this topic](#)

Windows Presentation Foundation (WPF) is designed to save developers from the difficulties of threading. As a result, the majority of WPF developers won't have to write an interface that uses more than one thread. Because multithreaded programs are complex and difficult to debug, they should be avoided when single-threaded solutions exist.

No matter how well architected, however, no UI framework will ever be able to provide a single-threaded solution for every sort of problem. WPF comes close, but there are still situations where multiple threads improve user interface (UI) responsiveness or application performance. After discussing some background material, this paper explores some of these situations and then concludes with a discussion of some lower-level details.

This topic contains the following sections.

- [Overview and the Dispatcher](#)
- [Threads in Action: The Samples](#)
- [Technical Details and Stumbling Points](#)
- [Related Topics](#)

## Overview and the Dispatcher

Typically, WPF applications start with two threads: one for handling rendering and another for managing the UI. The rendering thread effectively runs hidden in the background while the UI thread receives input, handles events, paints the screen, and runs application code. Most applications use a single UI thread, although in some situations it is best to use several. We'll discuss this with an example later.

The UI thread queues work items inside an object called a [Dispatcher](#). The [Dispatcher](#) selects work items on a priority basis and runs each one to completion. Every UI thread must have at least one [Dispatcher](#), and each [Dispatcher](#) can execute work items in exactly one thread.

The trick to building responsive, user-friendly applications is to maximize the [Dispatcher](#) throughput by keeping the work items small. This way items never get stale sitting in the [Dispatcher](#) queue waiting for processing. Any perceivable delay between input and response can frustrate a user.

How then are WPF applications supposed to handle big operations? What if your code involves a large calculation or needs to query a database on some remote server? Usually, the answer is to handle the big operation in a separate thread, leaving the UI thread free to tend to items in the [Dispatcher](#) queue. When the big operation is complete, it can report its result back to the UI thread for display.

Historically, Windows allows UI elements to be accessed only by the thread that created them. This means that a background thread in charge of some long-running task cannot update a text box when it is finished. Windows does this to ensure the integrity of UI components. A list box could look strange if its contents were updated by a background thread during painting.

WPF has a built-in mutual exclusion mechanism that enforces this coordination. Most classes in WPF derive from [DispatcherObject](#). At construction, a [DispatcherObject](#) stores a reference to the [Dispatcher](#) linked to the currently running thread. In effect, the [DispatcherObject](#) associates with the thread that creates it. During program execution, a [DispatcherObject](#) can call its public [VerifyAccess](#) method. [VerifyAccess](#) examines the [Dispatcher](#) associated with the current thread and compares it to the [Dispatcher](#) reference stored during construction. If they don't match, [VerifyAccess](#) throws an exception. [VerifyAccess](#) is intended to be called at the beginning of every method belonging to a [DispatcherObject](#).

If only one thread can modify the UI, how do background threads interact with the user? A background thread can ask the UI thread to perform an operation on its behalf. It does this by registering a work item with the [Dispatcher](#) of the UI thread. The [Dispatcher](#) class provides two

methods for registering work items: [Invoke](#) and [BeginInvoke](#). Both methods schedule a delegate for execution. [Invoke](#) is a synchronous call – that is, it doesn't return until the UI thread actually finishes executing the delegate. [BeginInvoke](#) is asynchronous and returns immediately.

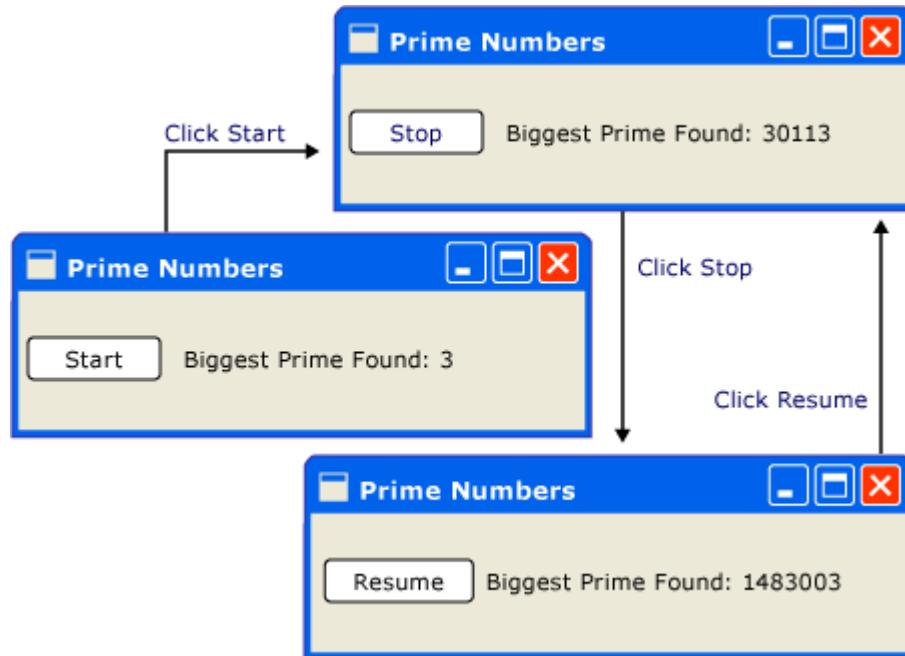
The [Dispatcher](#) orders the elements in its queue by priority. There are ten levels that may be specified when adding an element to the [Dispatcher](#) queue. These priorities are maintained in the [DispatcherPriority](#) enumeration. Detailed information about [DispatcherPriority](#) levels can be found in the Windows SDK documentation.

## Threads in Action: The Samples

### A Single-Threaded Application with a Long-Running Calculation

Most graphical user interfaces (GUIs) spend a large portion of their time idle while waiting for events that are generated in response to user interactions. With careful programming this idle time can be used constructively, without affecting the responsiveness of the UI. The WPF threading model doesn't allow input to interrupt an operation happening in the UI thread. This means you must be sure to return to the [Dispatcher](#) periodically to process pending input events before they get stale.

Consider the following example:



This simple application counts upwards from three, searching for prime numbers. When the user clicks the **Start** button, the search begins. When the program finds a prime, it updates the user interface with its discovery. At any point, the user can stop the search.

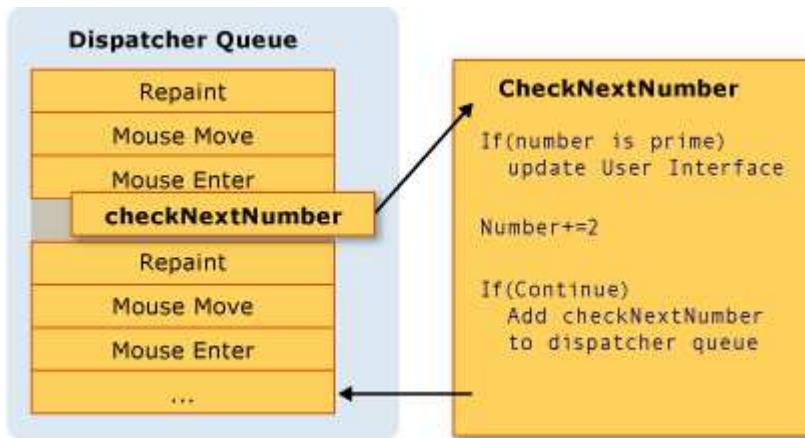
Although simple enough, the prime number search could go on forever, which presents some difficulties. If we handled the entire search inside of the click event handler of the button, we would never give the UI thread a chance to handle other events. The UI would be unable to respond to input or process messages. It would never repaint and never respond to button clicks.

We could conduct the prime number search in a separate thread, but then we would need to deal with synchronization issues. With a single-threaded approach, we can directly update the label that lists the largest prime found.

If we break up the task of calculation into manageable chunks, we can periodically return to the [Dispatcher](#) and process events. We can give WPF an opportunity to repaint and process input.

The best way to split processing time between calculation and event handling is to manage calculation from the [Dispatcher](#). By using the [BeginInvoke](#) method, we can schedule prime number checks in the same queue that UI events are drawn from. In our example, we schedule only a single prime number check at a time. After the prime number check is complete, we schedule the next check immediately. This check proceeds only after pending UI events have

been handled.



Microsoft Word accomplishes spell checking using this mechanism. Spell checking is done in the background using the idle time of the UI thread. Let's take a look at the code.

The following example shows the XAML that creates the user interface.

```

<Window x:Class="SDKSamples.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Prime Numbers" Width="260" Height="75"
    >
<StackPanel Orientation="Horizontal" VerticalAlignment="Center" >
    <Button Content="Start"
        Click="StartOrStop"
        Name="startStopButton"
        Margin="5,0,5,0"
        />
    <TextBlock Margin="10,5,0,0">Biggest Prime Found:</TextBlock>
    <TextBlock Name="bigPrime" Margin="4,5,0,0">3</TextBlock>
</StackPanel>
</Window>

```

```

<Window x:Class="SDKSamples.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Prime Numbers" Width="260" Height="75"
    >
<StackPanel Orientation="Horizontal" VerticalAlignment="Center" >
    <Button Content="Start"
        Click="StartOrStop"
        Name="startStopButton"
        Margin="5,0,5,0"
        />
    <TextBlock Margin="10,5,0,0">Biggest Prime Found:</TextBlock>
    <TextBlock Name="bigPrime" Margin="4,5,0,0">3</TextBlock>
</StackPanel>
</Window>

```

The following example shows the code-behind.

### C#

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Threading;
using System.Threading;

namespace SDKSamples
{
    public partial class Window1 : Window
    {
        public delegate void NextPrimeDelegate();

        //Current number to check
        private long num = 3;

        private bool continueCalculating = false;

        public Window1() : base()
        {
            InitializeComponent();
        }

        private void StartOrStop(object sender, EventArgs e)
        {
            if (continueCalculating)
            {
                continueCalculating = false;
                startStopButton.Content = "Resume";
            }
            else
            {
                continueCalculating = true;
                startStopButton.Content = "Stop";
                startStopButton.Dispatcher.BeginInvoke(
                    DispatcherPriority.Normal,
                    new NextPrimeDelegate(CheckNextNumber));
            }
        }

        public void CheckNextNumber()
        {
            // Reset flag.
            NotAPrime = false;

            for (long i = 3; i <= Math.Sqrt(num); i++)
            {
                if (num % i == 0)
                {
                    // Set not a prime flag to true.
                    NotAPrime = true;
                    break;
                }
            }

            // If a prime number.
            if (!NotAPrime)
```

```
        {
            bigPrime.Text = num.ToString();
        }

        num += 2;
        if (continueCalculating)
        {
            startStopButton.Dispatcher.BeginInvoke(
                System.Windows.Threading.DispatcherPriority.SystemIdle,
                new NextPrimeDelegate(this.CheckNextNumber));
        }
    }

    private bool NotAPrime = false;
}
}
```

The following example shows the event handler for the [Button](#).

### C#

```
private void StartOrStop(object sender, EventArgs e)
{
    if (continueCalculating)
    {
        continueCalculating = false;
        startStopButton.Content = "Resume";
    }
    else
    {
        continueCalculating = true;
        startStopButton.Content = "Stop";
        startStopButton.Dispatcher.BeginInvoke(
            DispatcherPriority.Normal,
            new NextPrimeDelegate(CheckNextNumber));
    }
}
```

Besides updating the text on the [Button](#), this handler is responsible for scheduling the first prime number check by adding a delegate to the [Dispatcher](#) queue. Sometime after this event handler has completed its work, the [Dispatcher](#) will select this delegate for execution.

As we mentioned earlier, [BeginInvoke](#) is the [Dispatcher](#) member used to schedule a delegate for execution. In this case, we choose the [SystemIdle](#) priority. The [Dispatcher](#) will execute this delegate only when there are no important events to process. UI responsiveness is more important than number checking. We also pass a new delegate representing the number-checking routine.

### C#

```
public void CheckNextNumber()
{
    // Reset flag.
    NotAPrime = false;

    for (long i = 3; i <= Math.Sqrt(num); i++)
```

```
{  
    if (num % i == 0)  
    {  
        // Set not a prime flag to true.  
        NotAPrime = true;  
        break;  
    }  
}  
  
// If a prime number.  
if (!NotAPrime)  
{  
    bigPrime.Text = num.ToString();  
}  
  
num += 2;  
if (continueCalculating)  
{  
    startStopButton.Dispatcher.BeginInvoke(  
        System.Windows.Threading.DispatcherPriority.SystemIdle,  
        new NextPrimeDelegate(this.CheckNextNumber));  
}  
}  
  
private bool NotAPrime = false;
```

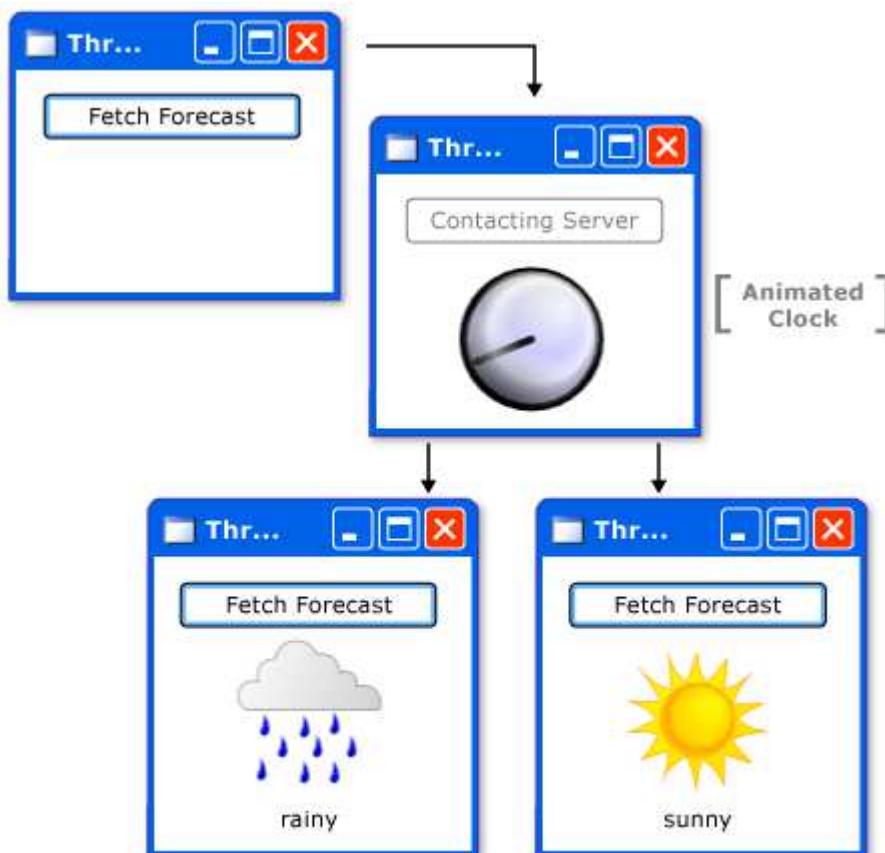
This method checks if the next odd number is prime. If it is prime, the method directly updates the `bigPrime TextBlock` to reflect its discovery. We can do this because the calculation is occurring in the same thread that was used to create the component. Had we chosen to use a separate thread for the calculation, we would have to use a more complicated synchronization mechanism and execute the update in the UI thread. We'll demonstrate this situation next.

For the complete source code for this sample, see the [Single-Threaded Application with Long-Running Calculation Sample](#)

### Handling a Blocking Operation with a Background Thread

Handling blocking operations in a graphical application can be difficult. We don't want to call blocking methods from event handlers because the application will appear to freeze up. We can use a separate thread to handle these operations, but when we're done, we have to synchronize with the UI thread because we can't directly modify the GUI from our worker thread. We can use `Invoke` or `BeginInvoke` to insert delegates into the `Dispatcher` of the UI thread. Eventually, these delegates will be executed with permission to modify UI elements.

In this example, we mimic a remote procedure call that retrieves a weather forecast. We use a separate worker thread to execute this call, and we schedule an update method in the `Dispatcher` of the UI thread when we're finished.

**C#**

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Windows.Threading;
using System.Threading;

namespace SDKSamples
{
    public partial class Window1 : Window
    {
        // Delegates to be used in placing jobs onto the Dispatcher.
        private delegate void NoArgDelegate();
        private delegate void OneArgDelegate(String arg);

        // Storyboards for the animations.
        private Storyboard showClockFaceStoryboard;
        private Storyboard hideClockFaceStoryboard;
        private Storyboard showWeatherImageStoryboard;
        private Storyboard hideWeatherImageStoryboard;

        public Window1(): base()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            // Load the storyboard resources.
        }
    }
}
```

```
showClockFaceStoryboard =
    (Storyboard)this.Resources["ShowClockFaceStoryboard"];
hideClockFaceStoryboard =
    (Storyboard)this.Resources["HideClockFaceStoryboard"];
showWeatherImageStoryboard =
    (Storyboard)this.Resources["ShowWeatherImageStoryboard"];
hideWeatherImageStoryboard =
    (Storyboard)this.Resources["HideWeatherImageStoryboard"];
}

private void ForecastButtonHandler(object sender, RoutedEventArgs e)
{
    // Change the status image and start the rotation animation.
    fetchButton.IsEnabled = false;
    fetchButton.Content = "Contacting Server";
    weatherText.Text = "";
    hideWeatherImageStoryboard.Begin(this);

    // Start fetching the weather forecast asynchronously.
    NoArgDelegate fetcher = new NoArgDelegate(
        this.FetchWeatherFromServer);

    fetcher.BeginInvoke(null, null);
}

private void FetchWeatherFromServer()
{
    // Simulate the delay from network access.
    Thread.Sleep(4000);

    // Tried and true method for weather forecasting - random numbers.
    Random rand = new Random();
    String weather;

    if (rand.Next(2) == 0)
    {
        weather = "rainy";
    }
    else
    {
        weather = "sunny";
    }

    // Schedule the update function in the UI thread.
    tomorrowsWeather.Dispatcher.BeginInvoke(
        System.Windows.Threading.DispatcherPriority.Normal,
        new OneArgDelegate(UpdateUserInterface),
        weather);
}

private void UpdateUserInterface(String weather)
{
    //Set the weather image
    if (weather == "sunny")
    {
        weatherIndicatorImage.Source = (ImageSource)this.Resources[
            "SunnyImageSource"];
    }
    else if (weather == "rainy")
    {
        weatherIndicatorImage.Source = (ImageSource)this.Resources[
```

```
        "RainingImageSource"];
    }

    //Stop clock animation
    showClockFaceStoryboard.Stop(this);
    hideClockFaceStoryboard.Begin(this);

    //Update UI text
    fetchButton.IsEnabled = true;
    fetchButton.Content = "Fetch Forecast";
    weatherText.Text = weather;
}

private void HideClockFaceStoryboard_Completed(object sender,
EventArgs args)
{
    showWeatherImageStoryboard.Begin(this);
}

private void HideWeatherImageStoryboard_Completed(object sender,
EventArgs args)
{
    showClockFaceStoryboard.Begin(this, true);
}
}
```

The following are some of the details to be noted.

- Creating the Button Handler

C#

```
private void ForecastButtonHandler(object sender, RoutedEventArgs e)
{
    // Change the status image and start the rotation animation.
    fetchButton.IsEnabled = false;
    fetchButton.Content = "Contacting Server";
    weatherText.Text = "";
    hideWeatherImageStoryboard.Begin(this);

    // Start fetching the weather forecast asynchronously.
    NoArgDelegate fetcher = new NoArgDelegate(
        this.FetchWeatherFromServer);

    fetcher.BeginInvoke(null, null);
}
```

When the button is clicked, we display the clock drawing and start animating it. We disable the button. We invoke the `FetchWeatherFromServer` method in a new thread, and then we return, allowing the `Dispatcher` to process events while we wait to collect the weather forecast.

- Fetching the Weather

C#

```
private void FetchWeatherFromServer()
{
    // Simulate the delay from network access.
    Thread.Sleep(4000);

    // Tried and true method for weather forecasting - random numbers.
    Random rand = new Random();
    String weather;

    if (rand.Next(2) == 0)
    {
        weather = "rainy";
    }
    else
    {
        weather = "sunny";
    }

    // Schedule the update function in the UI thread.
    tomorrowsWeather.Dispatcher.BeginInvoke(
        System.Windows.Threading.DispatcherPriority.Normal,
        new OneArgDelegate(UpdateUserInterface),
        weather);
}
```

To keep things simple, we don't actually have any networking code in this example. Instead, we simulate the delay of network access by putting our new thread to sleep for four seconds. In this time, the original UI thread is still running and responding to events. To show this, we've left an animation running, and the minimize and maximize buttons also continue to work.

When the delay is finished, and we've randomly selected our weather forecast, it's time to report back to the UI thread. We do this by scheduling a call to `UpdateUserInterface` in the UI thread using that thread's `Dispatcher`. We pass a string describing the weather to this scheduled method call.

- Updating the UI

### C#

```
private void UpdateUserInterface(String weather)
{
    //Set the weather image
    if (weather == "sunny")
    {
        weatherIndicatorImage.Source = (ImageSource)this.Resources[
            "SunnyImageSource"];
    }
    else if (weather == "rainy")
    {
        weatherIndicatorImage.Source = (ImageSource)this.Resources[
            "RainingImageSource"];
    }

    //Stop clock animation
    showClockFaceStoryboard.Stop(this);
    hideClockFaceStoryboard.Begin(this);

    //Update UI text
```

```
fetchButton.IsEnabled = true;
fetchButton.Content = "Fetch Forecast";
weatherText.Text = weather;
}
```

When the [Dispatcher](#) in the UI thread has time, it executes the scheduled call to [UpdateUserInterface](#). This method stops the clock animation and chooses an image to describe the weather. It displays this image and restores the "fetch forecast" button.

## Multiple Windows, Multiple Threads

Some WPF applications require multiple top-level windows. It is perfectly acceptable for one Thread/[Dispatcher](#) combination to manage multiple windows, but sometimes several threads do a better job. This is especially true if there is any chance that one of the windows will monopolize the thread.

Windows Explorer works in this fashion. Each new Explorer window belongs to the original process, but it is created under the control of an independent thread.

By using a WPF [Frame](#) control, we can display Web pages. We can easily create a simple Internet Explorer substitute. We start with an important feature: the ability to open a new explorer window. When the user clicks the "new window" button, we launch a copy of our window in a separate thread. This way, long-running or blocking operations in one of the windows won't lock all the other windows.

In reality, the Web browser model has its own complicated threading model. We've chosen it because it should be familiar to most readers.

The following example shows the code.

```
<Window x:Class="SDKSamples.Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MultiBrowse"
    Height="600"
    Width="800"
    Loaded="OnLoaded"
    >
    <StackPanel Name="Stack" Orientation="Vertical">
        <StackPanel Orientation="Horizontal">
            <Button Content="New Window"
                Click="NewWindowHandler" />
            <TextBox Name="newLocation"
                Width="500" />
            <Button Content="GO!"
                Click="Browse" />
        </StackPanel>
        <Frame Name="placeHolder"
            Width="800"
            Height="550"></Frame>
    </StackPanel>
</Window>
```

C#

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Threading;
using System.Threading;

namespace SDKSamples
{
    public partial class Window1 : Window
    {

        public Window1() : base()
        {
            InitializeComponent();
        }

        private void OnLoaded(object sender, RoutedEventArgs e)
        {
            placeHolder.Source = new Uri("http://www.msn.com");
        }

        private void Browse(object sender, RoutedEventArgs e)
        {
            placeHolder.Source = new Uri(newLocation.Text);
        }

        private void NewWindowHandler(object sender, RoutedEventArgs e)
        {
            Thread newWindowThread = new Thread(new ThreadStart(ThreadStartingPoint));
            newWindowThread.SetApartmentState(ApartmentState.STA);
            newWindowThread.IsBackground = true;
            newWindowThread.Start();
        }

        private void ThreadStartingPoint()
        {
            Window1 tempWindow = new Window1();
            tempWindow.Show();
            System.Windows.Threading.Dispatcher.Run();
        }
    }
}
```

The following threading segments of this code are the most interesting to us in this context:

### C#

```
private void NewWindowHandler(object sender, RoutedEventArgs e)
{
    Thread newWindowThread = new Thread(new ThreadStart(ThreadStartingPoint));
    newWindowThread.SetApartmentState(ApartmentState.STA);
    newWindowThread.IsBackground = true;
    newWindowThread.Start();
}
```

This method is called when the "new window" button is clicked. It creates a new thread and starts it asynchronously.

C#

```
private void ThreadStartingPoint()
{
    Window1 tempWindow = new Window1();
    tempWindow.Show();
    System.Windows.Threading.Dispatcher.Run();
}
```

This method is the starting point for the new thread. We create a new window under the control of this thread. WPF automatically creates a new [Dispatcher](#) to manage the new thread. All we have to do to make the window functional is to start the [Dispatcher](#).

## Technical Details and Stumbling Points

### Writing Components Using Threading

The Microsoft .NET Framework Developer's Guide describes a pattern for how a component can expose asynchronous behavior to its clients (see [Event-based Asynchronous Pattern Overview](#)). For instance, suppose we wanted to package the `FetchWeatherFromServer` method into a reusable, nongraphical component. Following the standard Microsoft .NET Framework pattern, this would look something like the following.

C#

```
public class WeatherComponent : Component
{
    //gets weather: Synchronous
    public string GetWeather()
    {
        string weather = "";

        //predict the weather

        return weather;
    }

    //get weather: Asynchronous
    public void GetWeatherAsync()
    {
        //get the weather
    }

    public event GetWeatherCompletedEventHandler GetWeatherCompleted;
}

public class GetWeatherCompletedEventArgs : AsyncCompletedEventArgs
{
    public GetWeatherCompletedEventArgs(Exception error, bool canceled,
        object userState, string weather)
    :
        base(error, canceled, userState)
```

```
{  
    _weather = weather;  
}  
  
public string Weather  
{  
    get { return _weather; }  
}  
private string _weather;  
}  
  
public delegate void GetWeatherCompletedEventHandler(object sender,  
    GetWeatherEventArgs e);
```

`GetWeatherAsync` would use one of the techniques described earlier, such as creating a background thread, to do the work asynchronously, not blocking the calling thread.

One of the most important parts of this pattern is calling the `MethodNameCompleted` method on the same thread that called the `MethodNameAsync` method to begin with. You could do this using WPF fairly easily, by storing `CurrentDispatcher`—but then the nongraphical component could only be used in WPF applications, not in Windows Forms or ASP.NET programs.

The `DispatcherSynchronizationContext` class addresses this need—think of it as a simplified version of `Dispatcher` that works with other UI frameworks as well.

### C#

```
public class WeatherComponent2 : Component  
{  
    public string GetWeather()  
    {  
        return fetchWeatherFromServer();  
    }  
  
    private DispatcherSynchronizationContext requestingContext = null;  
  
    public void GetWeatherAsync()  
    {  
        if (requestingContext != null)  
            throw new InvalidOperationException("This component can only handle 1 async request at a time");  
  
        requestingContext = (DispatcherSynchronizationContext)DispatcherSynchronizationContext.Current;  
  
        NoArgDelegate fetcher = new NoArgDelegate(this.fetchWeatherFromServer);  
  
        // Launch thread  
        fetcher.BeginInvoke(null, null);  
    }  
  
    private void RaiseEvent(GetWeatherEventArgs e)  
    {  
        if (GetWeatherCompleted != null)  
            GetWeatherCompleted(this, e);  
    }  
  
    private string fetchWeatherFromServer()  
    {  
        // do stuff  
        string weather = "";
```

```
GetWeatherCompletedEventArgs e =
    new GetWeatherCompletedEventArgs(null, false, null, weather);

SendOrPostCallback callback = new SendOrPostCallback(DoEvent);
requestingContext.Post(callback, e);
requestingContext = null;

return e.Weather;
}

private void DoEvent(object e)
{
    //do stuff
}

public event GetWeatherCompletedEventHandler GetWeatherCompleted;
public delegate string NoArgDelegate();
}
```

## Nested Pumping

Sometimes it is not feasible to completely lock up the UI thread. Let's consider the `Show` method of the `MessageBox` class. `Show` doesn't return until the user clicks the OK button. It does, however, create a window that must have a message loop in order to be interactive. While we are waiting for the user to click OK, the original application window does not respond to user input. It does, however, continue to process paint messages. The original window redraws itself when covered and revealed.



Some thread must be in charge of the message box window. WPF could create a new thread just for the message box window, but this thread would be unable to paint the disabled elements in the original window (remember the earlier discussion of mutual exclusion). Instead, WPF uses a nested message processing system. The `Dispatcher` class includes a special method called `PushFrame`, which stores an application's current execution point then begins a new message loop. When the nested message loop finishes, execution resumes after the original `PushFrame` call.

In this case, `PushFrame` maintains the program context at the call to `MessageBox.Show`, and it starts a new message loop to repaint the background window and handle input to the message box window. When the user clicks OK and clears the pop-up window, the nested loop exits and control resumes after the call to `Show`.

## Stale Routed Events

The routed event system in WPF notifies entire trees when events are raised.

```

<Canvas MouseLeftButtonDown="handler1"
    Width="100"
    Height="100"
    >
    <Ellipse Width="50"
        Height="50"
        Fill="Blue"
        Canvas.Left="30"
        Canvas.Top="50"
        MouseLeftButtonDown="handler2"
    />
</Canvas>

```

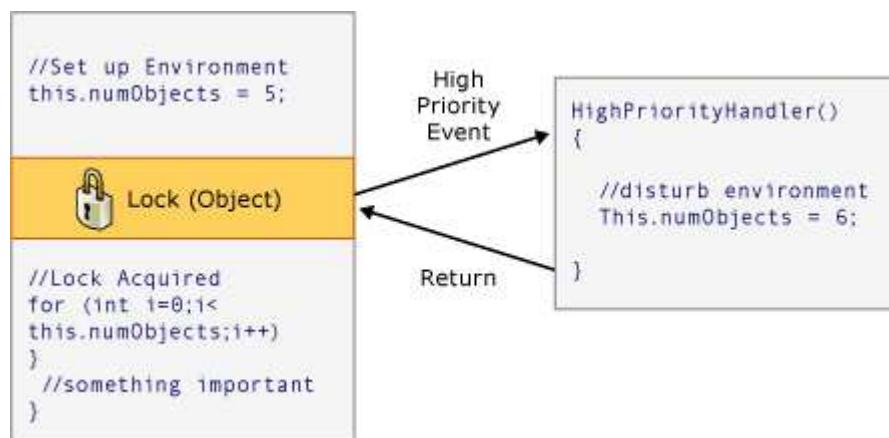
When the left mouse button is pressed over the ellipse, `handler2` is executed. After `handler2` finishes, the event is passed along to the `Canvas` object, which uses `handler1` to process it. This happens only if `handler2` does not explicitly mark the event object as handled.

It's possible that `handler2` will take a great deal of time processing this event. `handler2` might use `PushFrame` to begin a nested message loop that doesn't return for hours. If `handler2` does not mark the event as handled when this message loop is complete, the event is passed up the tree even though it is very old.

### Reentrancy and Locking

The locking mechanism of the common language runtime (CLR) doesn't behave exactly as one might imagine; one might expect a thread to cease operation completely when requesting a lock. In actuality, the thread continues to receive and process high-priority messages. This helps prevent deadlocks and make interfaces minimally responsive, but it introduces the possibility for subtle bugs. The vast majority of the time you don't need to know anything about this, but under rare circumstances (usually involving Win32 window messages or COM STA components) this can be worth knowing.

Most interfaces are not built with thread safety in mind because developers work under the assumption that a UI is never accessed by more than one thread. In this case, that single thread may make environmental changes at unexpected times, causing those ill effects that the `DispatcherObject` mutual exclusion mechanism is supposed to solve. Consider the following pseudocode:



Most of the time that's the right thing, but there are times in WPF where such unexpected reentrancy can really cause problems. So, at certain key times, WPF calls `DisableProcessing`, which changes the lock instruction for that thread to use the WPF reentrancy-free lock, instead of the usual CLR lock.

So why did the CLR team choose this behavior? It had to do with COM STA objects and the

finalization thread. When an object is garbage collected, its **Finalize** method is run on the dedicated finalizer thread, not the UI thread. And therein lies the problem, because a COM STA object that was created on the UI thread can only be disposed on the UI thread. The CLR does the equivalent of a [BeginInvoke](#) (in this case using Win32's **SendMessage**). But if the UI thread is busy, the finalizer thread is stalled and the COM STA object can't be disposed, which creates a serious memory leak. So the CLR team made the tough call to make locks work the way they do.

The task for WPF is to avoid unexpected reentrancy without reintroducing the memory leak, which is why we don't block reentrancy everywhere.

## See Also

### Other Resources

[Single-Threaded Application with Long-Running Calculation Sample](#)

Did you find this helpful?  Yes  No

© 2012 Microsoft. All rights reserved.



## It's Mine, Don't Touch!: interactions at a large multi-touch display in a city centre

Full Text:  [PDF](#)

Authors: [Peter Peltonen](#) [Helsinki Institute for Information Technology and University of Helsinki, Helsinki, Finland](#)  
[Esko Kurvinen](#) [Helsinki Institute for Information Technology and University of Helsinki, Helsinki, Finland](#)  
[Antti Salovaara](#) [Helsinki Institute for Information Technology and University of Helsinki, Helsinki, Finland](#)  
[Giulio Jacucci](#) [Helsinki Institute for Information Technology and University of Helsinki, Helsinki, Finland](#)  
[Tommi Ilmonen](#) [Helsinki Institute for Information Technology and University of Helsinki, Helsinki, Finland](#)  
[John Evans](#) [Helsinki Institute for Information Technology and University of Helsinki, Helsinki, Finland](#)  
[Antti Oulasvirta](#) [Helsinki Institute for Information Technology and University of Helsinki, Helsinki, Finland](#)  
[Petri Saarikko](#) [Helsinki Institute for Information Technology and University of Helsinki, Helsinki, Finland](#)

Published in:



- Proceeding  
**CHI '08** Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems  
 Pages 1285-1294  
[ACM](#) New York, NY, USA ©2008  
[table of contents](#) ISBN: 978-1-60558-011-1 doi:>[10.1145/1357054.1357255](https://doi.org/10.1145/1357054.1357255)



2008 Article



Bibliometrics

- Downloads (6 Weeks): 76
- Downloads (12 Months): 505
- Citation Count: 58



### Tools and Resources

[Request Permissions](#)

[TOC Service:](#)

[Email](#)  [RSS](#)  [RSS](#)

[Save to Binder](#)

[Export Formats:](#)

[BibTeX](#) [EndNote](#) [ACM Ref](#)

[Upcoming Conference:](#)

[CHI '13](#)

Share:



**Tags:** [design](#)

[evaluation/methodology](#) [human](#)

[factors](#) [multi-user interfaces](#)

[situated public displays](#) [urban](#)

[environments](#)

[Feedback](#) | Switch to [single page view](#) (no tabs)

[Abstract](#) [Authors](#) [References](#) [Cited By](#) [Index Terms](#) [Publication](#) [Reviews](#) [Comments](#) [Table of Contents](#)

We present data from detailed observations of CityWall, a large multi-touch display installed in a central location in Helsinki, Finland. During eight days of installation, 1199 persons interacted with the system in various social configurations. Videos of these encounters were examined qualitatively as well as quantitatively based on human coding of events. The data convey phenomena that arise uniquely in public use: crowding, massively parallel interaction, teamwork, games, negotiations of transitions and handovers, conflict management, gestures and overt remarks to co-present people, and "marking" the display for others. We analyze how public availability is achieved through social learning and negotiation, why interaction becomes performative and, finally, how the display restructures the public space. The multi-touch feature, gesture-based interaction, and the physical display size contributed differentially to these uses. Our findings on the social organization of the use of public displays can be useful for designing such systems for urban environments.

Powered by **THE ACM GUIDE TO COMPUTING LITERATURE**

The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2012 ACM, Inc.  
[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)



## Extending touch: towards interaction with large-scale surfaces

Full Text:  [Pdf](#)see [source materials](#) below for [more options](#)Authors: [Alexander Schick](#) Interactive Analysis and Diagnosis[Florian van de Camp](#) Interactive Analysis and Diagnosis[Joris Ijselmuizen](#) Interactive Analysis and Diagnosis[Rainer Stiefelhagen](#) [Interactive Analysis and Diagnosis](#) and [Universität Karlsruhe \(TH\)](#)

2009 Article

 [Bibliometrics](#)

- Downloads (6 Weeks): 9
- Downloads (12 Months): 163
- Citation Count: 7

Published in:



- Proceeding

[ITS '09 Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces](#)  
Pages 117-124  
[ACM](#) New York, NY, USA ©2009  
[table of contents](#) ISBN: 978-1-60558-733-2 doi:>[10.1145/1731903.1731927](https://doi.org/10.1145/1731903.1731927)

## Tools and Resources

 [Request Permissions](#) [TOC Service:](#) [Email](#)  [RSS](#)  [RSS](#) [Save to Binder](#) [Export Formats:](#)[BibTeX](#) [EndNote](#) [ACM Ref](#) [Upcoming Conference:](#)[ITS'12](#)

Share:

**Tags:** [computer vision](#) [input devices and strategies](#) [large surfaces](#) [multitouch](#) [touch and pointing interaction](#) [visual hull](#) [Feedback](#) | Switch to [single page view](#) (no tabs)[Abstract](#) [Source Materials](#) [Authors](#) [References](#) [Cited By](#) [Index Terms](#) [Publication](#) [Reviews](#) [Comments](#) [Table of Contents](#)

Touch is a very intuitive modality for interacting with objects displayed on arbitrary surfaces. However, when using touch for large-scale surfaces, not every point is reachable. Therefore, an extension is required that keeps the intuitivity provided by touch: pointing. We will present our system that allows both input modalities in one single framework. Our method is based on 3D reconstruction, using standard RGB cameras only, and allows seamless switching between touch and pointing, even while interacting. Our approach scales very well with large surfaces without modifying them. We present a technical evaluation of the system's accuracy, as well as a user study. We found that users preferred our system to a touch-only system, because they had more freedom during interaction and could solve the presented task significantly faster.

Powered by **THE ACM GUIDE TO COMPUTING LITERATURE**

The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2012 ACM, Inc.  
[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)

## Microsoft Kinect Sensor and Its Effect

Zhengyou Zhang  
*Microsoft Research*

Recent advances in 3D depth cameras such as Microsoft Kinect sensors ([www.xbox.com/en-US/kinect](http://www.xbox.com/en-US/kinect)) have created many opportunities for multimedia computing. Kinect was built to revolutionize the way people play games and how they experience entertainment. With Kinect, people are able to interact with the games with their body in a natural way. The key enabling technology is human body-language understanding; the computer must first understand what a user is doing before it can respond. This has always been an active research field in computer vision, but it has proven formidably difficult with video cameras. The Kinect sensor lets the computer directly sense the third dimension (depth) of the players and the environment, making the task much easier. It also understands when users talk, knows who they are when they walk up to it, and can interpret their movements and translate them into a format that developers can use to build new experiences.

Kinect's impact has extended far beyond the gaming industry. With its wide availability and low cost, many researchers and practitioners in computer science, electronic engineering, and robotics are leveraging the sensing technology to develop creative new ways to interact with machines and to perform other tasks, from helping children with autism to assisting doctors in operating rooms. Microsoft calls this

the Kinect Effect. On 1 February 2012, Microsoft released the Kinect Software Development Kit (SDK) for Windows ([www.microsoft.com/en-us/kinectforwindows](http://www.microsoft.com/en-us/kinectforwindows)), which will undoubtedly amplify the Kinect Effect. The SDK will potentially transform human-computer interaction in multiple industries—education, healthcare, retail, transportation, and beyond.

The activity on the news site and discussion community KinectHacks.net helps illustrate the excitement behind the Microsoft Kinect technology. Kinect was launched on 4 November 2010. A month later there were already nine pages containing brief descriptions of approximately 90 projects, and the number of projects posted on KinectHacks.net has grown steadily. Based on my notes, there were 24 pages on 10 February 2011, 55 pages on 2 August 2011, 63 pages on 12 January 2012, and 65 pages on 18 February while I was writing this article. This comment from KinectHacks.net nicely summarizes the enthusiasm about Kinect: "Every few hours new applications are emerging for the Kinect and creating new phenomenon that is nothing short of revolutionary."

### Kinect Sensor

The Kinect sensor incorporates several advanced sensing hardware. Most notably, it contains a depth sensor, a color camera, and a four-microphone array that provide full-body 3D motion capture, facial recognition, and voice recognition capabilities (see Figure 1). A detailed report of the components in the Kinect sensor is available at [www.waybeta.com/news/58230/microsoft-kinect-somatosensory-game-device-full-disassembly-report\\_microsoft-xbox](http://www.waybeta.com/news/58230/microsoft-kinect-somatosensory-game-device-full-disassembly-report_microsoft-xbox). This article focuses on the vision aspect of the Kinect sensor. (See related work for details on the audio component.<sup>1</sup>)

### Editor's Note

Sales of Microsoft's controller-free gaming system Kinect topped 10 million during the first three months after its launch, setting a new Guinness World Record for the Fastest-Selling Consumer Electronics Device. What drove this phenomenal success? This article unravels the enabling technologies behind Kinect and discusses the Kinect Effect that potentially will transform human-computer interaction in multiple industries.

Figure 1b shows the arrangement of the infrared (IR) projector, the color camera, and the IR camera. The depth sensor consists of the IR projector combined with the IR camera, which is a monochrome complementary metal-oxide semiconductor (CMOS) sensor. The depth-sensing technology is licensed from the Israeli company PrimeSense ([www.primesense.com](http://www.primesense.com)). Although the exact technology is not disclosed, it is based on the structured light principle. The IR projector is an IR laser that passes through a diffraction grating and turns into a set of IR dots. Figure 2 shows the IR dots seen by the IR camera.

The relative geometry between the IR projector and the IR camera as well as the projected IR dot pattern are known. If we can match a dot observed in an image with a dot in the projector pattern, we can reconstruct it in 3D using triangulation. Because the dot pattern is relatively random, the matching between the IR image and the projector pattern can be done in a straightforward way by comparing small neighborhoods using, for example, normalized cross correlation.

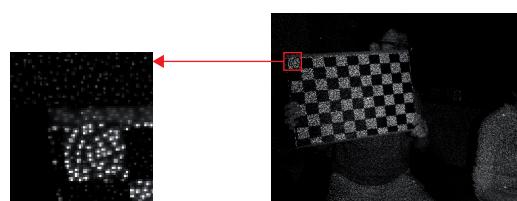
Figure 3 shows the depth map produced by the Kinect sensor for the IR image in Figure 2. The depth value is encoded with gray values; the darker a pixel, the closer the point is to the camera in space. The black pixels indicate that no depth values are available for those pixels. This might happen if the points are too far (and the depth values cannot be computed accurately), are too close (there is a blind region due to limited fields of view for the projector and the camera), are in the cast shadow of the projector (there are no IR dots), or reflect poor IR lights (such as hairs or specular surfaces).

The depth values produced by the Kinect sensor are sometimes inaccurate because the calibration between the IR projector and the IR camera becomes invalid. This could be caused by heat or vibration during transportation or a drift in the IR laser. To address this problem, together with the Kinect team, I developed a recalibration technique using the card in Figure 4 that is shipped with the Kinect sensor. If users find that the Kinect is not responding accurately to their actions, they can recalibrate the Kinect sensor by showing it the card. The idea is an adaptation of my earlier camera calibration technique.<sup>2</sup>

The depth value produced by the Kinect sensor is assumed to be an affine transformation



**Figure 1.** Microsoft Kinect sensor. (a) The Kinect sensor for Xbox 360. (b) The infrared (IR) projector, IR camera, and RGB camera inside a Kinect sensor.



**Figure 2.** The infrared (IR) dots seen by the IR camera. The image on the left shows a close-up of the red boxed area.



**Figure 3.** Kinect sensor depth image. The sensor produced this depth image from the infrared (IR) dot image in Figure 2.

**Figure 4.** Kinect calibration card. To recalibrate the Kinect sensor, the RGB camera's coordinate system determines the 3D coordinates of the feature points on the calibration card, which are considered to be the true values.



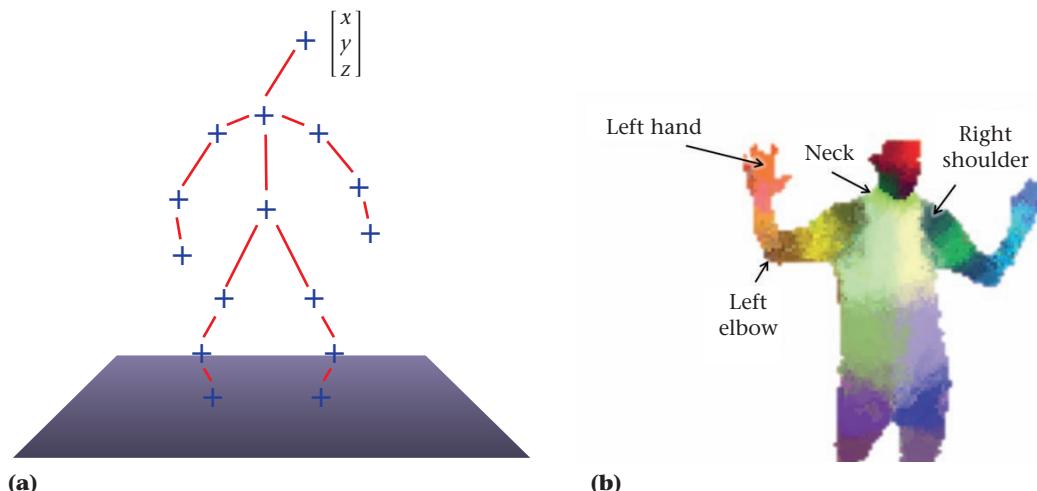
of the true depth value—that is,  $Z_{\text{measured}} = \alpha Z_{\text{true}} + \beta$ —which we found to be a reasonably good model. The goal of recalibration is to determine  $\alpha$  and  $\beta$ . (We could also use a more complex distortion model that applies the same technique.) Using the RGB camera, the recalibration technique determines the 3D coordinates of the feature points on the calibration card in the RGB camera's coordinate system, which are considered to be the true values. At the same time, the Kinect sensor also produces the measured 3D coordinates of those feature points in the IR camera's coordinate system.

Minimizing the distances between the two point sets, the Kinect sensor can estimate the values of  $\alpha$  and  $\beta$  and the rigid transformation between the RGB camera and the IR camera.

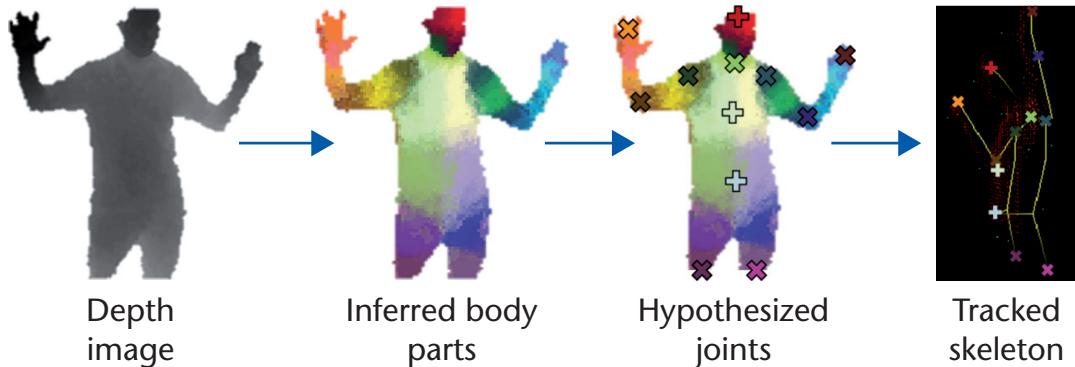
### Kinect Skeletal Tracking

The innovation behind Kinect hinges on advances in skeletal tracking. The operational envelope demands for commercially viable skeletal tracking are enormous. Simply put, skeletal tracking must ideally work for every person on the planet, in every household, without any calibration. A dauntingly high number of dimensions describe this envelope, such as the distance from the Kinect sensor and the sensor tilt angle. Entire sets of dimensions are necessary to describe unique individuals, including size, shape, hair, clothing, motions, and poses. Household environment dimensions are also necessary for lighting, furniture and other household furnishings, and pets.

In skeletal tracking, a human body is represented by a number of joints representing body parts such as head, neck, shoulders, and arms (see Figure 5a). Each joint is represented by its 3D coordinates. The goal is to determine all the 3D parameters of these joints in real time to allow fluent interactivity and with limited computation resources allocated on the Xbox 360 so as not to impact gaming performance. Rather than trying to determine directly the body pose in this high-dimensional space, Jamie Shotton and his team met the



**Figure 5.** Skeletal tracking. (a) Using a skeletal representation of various body parts, (b) Kinect uses per-pixel, body-part recognition as an intermediate step to avoid a combinatorial search over the different body joints.



**Figure 6.** The Kinect skeletal tracking pipeline. After performing per-pixel, body-part classification, the system hypothesizes the body joints by finding a global centroid of probability mass and then maps these joints to a skeleton using temporal continuity and prior knowledge.

challenge by proposing per-pixel, body-part recognition as an intermediate step (see Figure 5b).<sup>3</sup> Due to their innovative work, Microsoft honored the Kinect Skeletal Tracking team members with the 2012 Outstanding Technical Achievement Award ([www.microsoft.com/about/technicalrecognition/Kinect-Skeletal-Tracking.aspx](http://www.microsoft.com/about/technicalrecognition/Kinect-Skeletal-Tracking.aspx)).

Shotton's team treats the segmentation of a depth image as a per-pixel classification task (no pairwise terms or conditional random field are necessary). Evaluating each pixel separately avoids a combinatorial search over the different body joints. For training data, we generate realistic synthetic depth images of humans of many shapes and sizes in highly varied poses sampled from a large motion-capture database. We train a deep randomized decision forest classifier, which avoids overfitting by using hundreds of thousands of training images. Simple, discriminative depth comparison image features yield 3D translation invariance while maintaining high computational efficiency.

For further speedup, the classifier can be run in parallel on each pixel on a graphics processing unit (GPU). Finally, spatial modes of the inferred per-pixel distributions are computed using mean shift resulting in the 3D joint proposals. An optimized implementation of our algorithm runs in under 5 ms per frame (200 frames per second) on the Xbox 360 GPU. It works frame by frame across dramatically differing body shapes and sizes, and the learned discriminative approach naturally handles self-occlusions and poses cropped by the image frame.

Figure 6 illustrates the whole pipeline of Kinect skeletal tracking. The first step is to perform per-pixel, body-part classification. The second step is to hypothesize the body joints by

finding a global centroid of probability mass (local modes of density) through mean shift. The final stage is to map hypothesized joints to the skeletal joints and fit a skeleton by considering both temporal continuity and prior knowledge from skeletal train data.

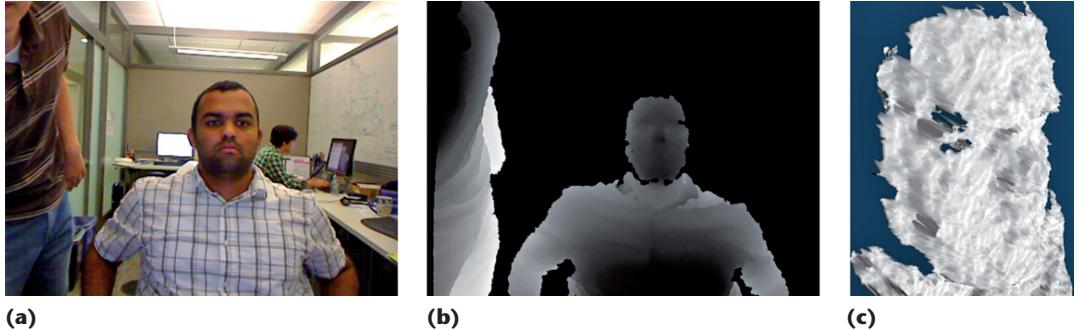
### Head-Pose and Facial-Expression Tracking

Head-pose and facial-expression tracking has been an active research area in computer vision for several decades. It has many applications including human-computer interaction, performance-driven facial animation, and face recognition. Most previous approaches focus on 2D images, so they must exploit some appearance and shape models because there are few distinct facial features. They might still suffer from lighting and texture variations, occlusion of profile poses, and so forth.

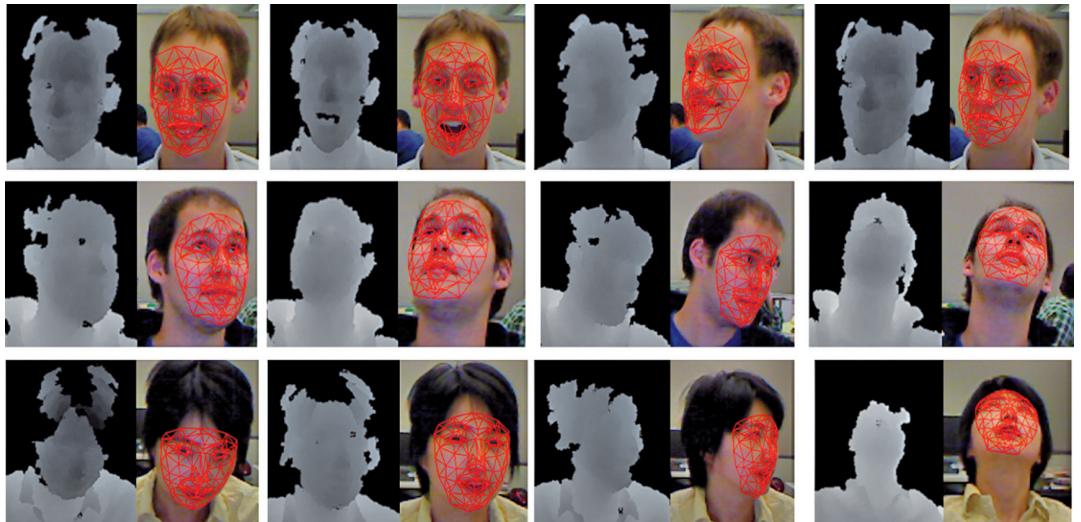
Related research has also focused on fitting morphable models to 3D facial scans. These 3D scans are usually obtained by high-quality laser scanners or structured light systems. Fitting these high-quality range data with a morphable face model usually involves the well-known iterative closest point (ICP) algorithm and its variants. The results are generally good, but these capturing systems are expensive to acquire or operate and the capture process is long.

A Kinect sensor produces both 2D color video and depth images at 30 fps, combining the best of both worlds. However, the Kinect's depth information is not very accurate. Figure 7 shows an example of the data captured by Kinect. Figure 7c, a close-up of the face region rendered from a different viewpoint, shows that the depth information is much noisier than laser-scanned data.

**Figure 7.** An example of a human face captured by the Kinect sensor.  
**(a)** Video frame (texture), **(b)** depth image, and **(c)** close up of the facial surface.



**Figure 8.** Facial expression tracking. These sample images show the results of Kinect tracking 2D feature points in video frames using a projected face mesh overlay.



We developed a regularized maximum-likelihood deformable model fitting (DMF) algorithm for 3D face tracking with Kinect.<sup>4</sup> We use a linear deformable head model with a linear combination of a neutral face, a set of shape basis units with coefficients that represent a particular person and are static over time, and a set of action basis units with coefficients that represent a person's facial expression and are dynamic overtime. Because a face cannot perform all facial expressions simultaneously, we believe in general the set of coefficients for the action basis units should be sparse, and thus we impose a  $L_1$  regularization.

The depth values from Kinect do not have the same accuracy. Depth is determined through triangulation, similar to stereovision. The depth error increases with the distance squared. Thus, in formulating the distance between the face model and the depth map, although we still use the ICP concept, each point from the depth map has its proper covariance matrix to model its uncertainty, and the distance is actually the Mahalanobis distance. Furthermore, the 2D feature points in the video frames are

tracked across frames and integrated into the DMF framework seamlessly. In our formulation, the 2D feature points do not necessarily need to correspond to any vertices or to semantic facial features such as eye corners and lip contours in the deformable face model. The sequence of images in Figure 8 demonstrates the effectiveness of the proposed method.

Microsoft Avatar Kinect has adopted similar technology ([www.xbox.com/en-us/kinect/avatar-kinect](http://www.xbox.com/en-us/kinect/avatar-kinect)). With Avatar Kinect, you can control your avatar's facial expression and head through facial-expression tracking and its arm movements through skeletal tracking (see Figure 9). As you talk, frown, smile, or scowl, your voice and facial expressions are enacted by your avatar, bringing it to life. Avatar Kinect offers 15 unique virtual environments to reflect your mood and to inspire creative conversations and performances. In a virtual environment you choose, you can invite up to seven friends to join you for a discussion or have them join you at the performance stage where you can put on a show. Thus, you can see your friends' actual expressions in real time through their avatars.

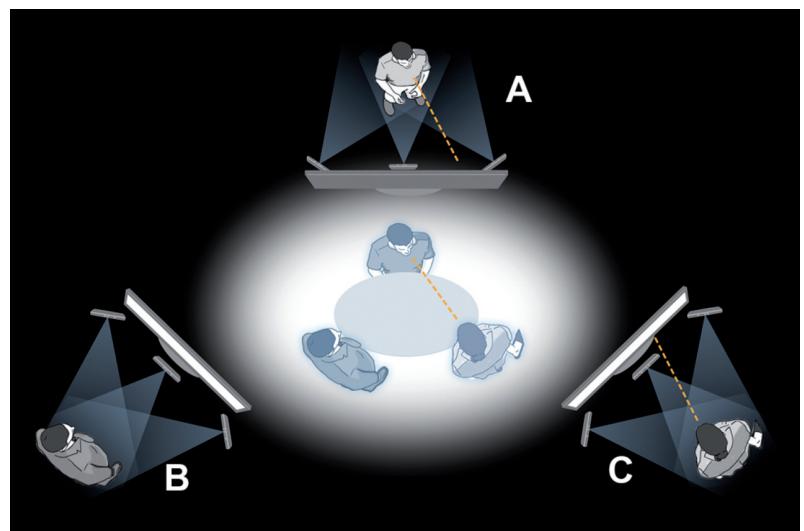


**Figure 9. Avatar Kinect virtual environment.**  
Users can control their avatars' facial expressions through facial-expression tracking and body movements through skeletal tracking.

### Teleimmersive Conferencing

With increasing economic globalization and workforce mobilization, there is a strong need for immersive experiences that enable people across geographically distributed sites to interact collaboratively. Such advanced infrastructures and tools require a deep understanding of multiple disciplines. In particular, computer vision, graphics, and acoustics are indispensable to capturing and rendering 3D environments that create the illusion that the remote participants are in the same room. Existing video-conferencing systems, whether they are available on desktop and mobile devices or in dedicated conference rooms with built-in furniture and life-sized high-definition video, leave a great deal to be desired—mutual gaze, 3D, motion parallax, spatial audio, to name a few. For the first time, the necessary immersive technologies are emerging and coming together to enable real-time capture, transport, and rendering of 3D holograms, and we are much closer to realizing man's dream reflected in Hollywood movies, from *Star Trek* and *Star Wars* to *The Matrix* and *Avatar*.

The Immersive Telepresence project at Microsoft Research addresses the scenario of a fully distributed team. Figure 10 illustrates three people joining a virtual/synthetic meeting from their own offices in three separate locations. A capture device (one or multiple Kinect sensors) at each location captures users in 3D with high fidelity (in both geometry and appearance). They are then put into a virtual



**Figure 10. Immersive telepresence.** One or multiple Kinect sensors at each location captures users in 3D with high fidelity. The system maintains mutual gaze between remote users and produces spatialized audio to help simulate a more realistic virtual meeting.

room as if they were seated at the same table. The user's position is tracked by the camera so the virtual room is rendered appropriately at each location from the user's eye perspective, which produces the right motion parallax effect, exactly like what a user would see in the real world if the three people met face to face. Because a consistent geometry is maintained and the user's position is tracked, the mutual gaze between remote users is maintained.

In Figure 10, users A and C are looking at each other, and B will see that A and C are



**Figure 11.** A screen shot of two remote people viewed from a third location. An enhanced 3D capture device runs in real time with multiple infrared (IR) projectors, IR cameras, and RGB cameras.

looking at each other because B only sees their side views. Furthermore, the audio is also spatialized, and the voice of each remote person comes from his location in the virtual room. The display at each location can be 2D or 3D, flat or curved, single or multiple, transparent or opaque, and so forth—the possibilities are numerous. In general, the larger a display is, the more immersive the user's experience.

Because each person must be seen from different angles by remote people, a single Kinect does not provide enough spatial coverage, and the visual quality is insufficient. Cha Zhang at Microsoft Research, with help from others, has developed an enhanced 3D capture device that runs in real time with multiple IR projectors, IR cameras, and RGB cameras. Figure 11 illustrates the quality of the 3D capture we can currently obtain with that device.

A similar system is being developed at the University of North Carolina at Chapel Hill that uses multiple Kinect sensors at each location.<sup>5</sup>

### Conclusion

The Kinect sensor offers an unlimited number of opportunities for old and new applications. This article only gives a taste of what is possible. Thus far, additional research areas include hand-gesture recognition,<sup>6</sup> human-activity recognition,<sup>7</sup> body biometrics estimation (such as weight, gender, or height),<sup>8</sup> 3D surface reconstruction,<sup>9</sup> and healthcare applications.<sup>10</sup> Here, I have included just one reference per application area, not trying to be exhaustive. Visit [www.xbox.com/en-US/Kinect/Kinect-Effect](http://www.xbox.com/en-US/Kinect/Kinect-Effect) and [www.kinecthacks.net](http://www.kinecthacks.net) for more examples. **MM**

### References

1. I. Tashev, "Recent Advances in Human-Machine Interfaces for Gaming and Entertainment," *Int'l Information Technology and Security*, vol. 3, no. 3, 2011, pp. 69–76.
2. Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, 2000, pp. 1330–1334.
3. J. Shotton et al., "Real-Time Human Pose Recognition in Parts from a Single Depth Image," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, IEEE CS Press, 2011, pp. 1297–1304.
4. Q. Cai et al., "3D Deformable Face Tracking with a Commodity Depth Camera," *Proc. 11th European Conf. Computer Vision (ECCV)*, vol. III, Springer-Verlag, 2010, pp. 229–242.
5. A. Maimone and H. Fuchs, "Encumbrance-Free Telepresence System with Real-Time 3D Capture and Display Using Commodity Depth Cameras," *Proc. IEEE Int'l Symp. Mixed and Augmented Reality (ISMAR)*, IEEE CS Press, 2011, pp. 137–146.
6. Z. Ren, J. Yuan, and Z. Zhang, "Robust Hand Gesture Recognition Based on Finger-Earth Movers Distance with a Commodity Depth Camera," *Proc. 19th ACM Int'l Conf. Multimedia (ACM MM)*, ACM Press, 2011, pp. 1093–1096.
7. W. Li, Z. Zhang, and Z. Liu, "Action Recognition Based on A Bag of 3D Points," *Proc. IEEE Int'l Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB)*, IEEE CS Press, 2010, pp. 9–14.
8. C. Velardo and J.-L. Dugelay, "Real Time Extraction of Body Soft Biometric from 3D Videos," *Proc. ACM Int'l Conf. Multimedia (ACM MM)*, ACM Press, 2011, pp. 781–782.
9. S. Izadi et al., "KinectFusion: Real-Time Dynamic 3D Surface Reconstruction and Interaction," *Proc. ACM SIGGRAPH*, 2011.
10. S. Bauer et al., "Multi-modal Surface Registration for Markerless Initial Patient Setup in Radiation Therapy Using Microsoft's Kinect Sensor," *Proc. IEEE Workshop on Consumer Depth Cameras for Computer Vision (CDC4CV)*, IEEE Press, 2011, pp. 1175–1181.

**Zhengyou Zhang** is a principal researcher and research manager of the Multimedia, Interaction, and Communication (MIC) Group at Microsoft Research. His research interests include computer vision, speech signal processing, multisensory fusion, multimedia computing, real-time collaboration, and human-machine interaction. Zhang has PhD and DSc degrees in computer science from the University of Paris XI. He is a fellow of IEEE and the founding editor in chief of the *IEEE Transactions on Autonomous Mental Development*. Contact him at zhang@microsoft.com.

**CN** Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.

## I.1 Anhang B

- Aufgabenstellung
- Erklärung
- Vereinbarung

# Aufgabenstellung Bachelorarbeit Abteilung I, FS 2012

## Lukas Elmer, Christina Heidt, Delia Treichler

### ***HSR Videowall***

#### **1. Auftraggeber und Betreuer**

*Praxispartner und Auftraggeber dieser Bachelorarbeit ist die HSR vertreten durch die Abteilung Informatik (Prof. Dr. Markus Stolze)*

*Betreuer HSR:*

Prof. Dr. Markus Stolze, Institut für Software [mstolze@hsr.ch](mailto:mstolze@hsr.ch)

Weitere fachliche Unterstützung durch die folgenden Assistenten:

- Kevin Gaunt (UX Design) [kgaunt@hsr.ch](mailto:kgaunt@hsr.ch)
- Michael Gfeller & Silvan Gehrig (Architektur & Implementation) [mgfeller@hsr.ch](mailto:mgfeller@hsr.ch),  
[sgehrig@hsr.ch](mailto:sgehrig@hsr.ch)

#### **2. Ausgangslage**

Die HSR möchte eine interaktive „Videowall“ im Eingangsbereich der Mensa aufzustellen, um die Arbeiten von Studenten und Instituten einem breiteren Publikum bekannt zu machen. Der genaue Standort, die Ausstattung und die Funktion der „Videowall“ sind zum Zeitpunkt der Aufgabenstellung noch nicht bekannt. Aktuell wird davon ausgegangen, dass auf der Wand die Poster der Bachelor-Arbeiten aller HSR Studiengänge ausgestellt werden könnten.

#### **3. Ziele der Arbeit**

In dieser Bachelorarbeit soll die Machbarkeit der Ausstellung der HSR Bachelor Poster auf einer Videowall abgeklärt werden. Hierbei sollen insbesondere die folgenden Fragen geklärt werden:

- Welche unterschiedlichen Gruppen von Passanten im Durchgang Gebäude 4 sind zu berücksichtigen? Wie wird der Durchgang im Gebäude 4 aktuell von den Gruppen genutzt? Wie können Passanten auf die Videowall aufmerksam gemacht werden? Wie können Passanten motiviert werden mit der Wand zu interagieren? Welche Informationen sind für die unterschiedlichen Gruppen interessant? Wie können insbesondere Studenten zur wiederholten Nutzung der Wand motiviert werden? Besteht Interesse auf Seite der Studenten an der Erstellung eines Videos (statt Poster) zur attraktiveren Präsentation ihrer Arbeiten auf der Videowall? Die Benutzeranalyse ist mit Daten aus konkreten Beobachtungen zu untermauern.
- Es ist zu klären, ob die Bachelor-Poster der verschiedenen Studiengänge alle in nutzbarem elektronischen Format vorliegen. Welche Auflösung (Pixel) ist nötig, um eine Lesbarkeit eines Grossteils der Poster ohne Zoom sicher zu stellen? Wie viel Prozent der Poster (ca.) lassen sich gut

---

lesbar darstellen, wenn eine Bildschirm-Auflösung gewählt wird, welche auch eine füssige Darstellung von Videos erlaubt?

- Welche unterschiedlichen Medien (z.B. Dokumente/Poster/Pdf, Video, Mini-Spiele) können mit der vorgeschlagenen Konfiguration auf der Wand dargestellt werden (und welche nicht)? Wie (und vom wem) können diese Inhalte auf der Videowall publiziert werden: Wie soll das „Content Management“ für die Videowall organisiert werden? Hierzu soll abklärt werden inwieweit eine Anbindung an das bestehende Typo-3 CMS möglich ist.
- Es ist zu klären, ob ein Wand aus 3x3 55“ Bildschirmen die beste Konfiguration ist, oder ob eine andere Konfiguration der Displays (und möglicherweise auch andere Displays) besser auf die zu unterstützenden Nutzungsszenarien passt. Hierbei sollte sowohl die Eindrücklichkeit der Präsentation, die Angemessenheit des Formates für den Raum und die Inhalte sowie die technische Machbarkeit (Auflösung und Performanz der Darstellung) mit verschiedenen Software- und Hardware-Konfigurationen (inkl. Video Karten Konfiguration) analysiert werden.
- Es ist zu klären ob Benutzer-Input mit einem einzigen Kinect Sensor entgegen genommen werden kann? Zur Analyse von möglichen Interaktionsformen soll in einem ersten Schritt ein Wizard-of-Oz Experiment durchgeführt werden. Vorschläge für Interaktionstechniken sollen grob ausgearbeitet werden und bewertet werden. In einem zweiten Schritt soll ein funktionstüchtiger Prototyp erstellt werden um mit diesem das Konzept zur Benutzerinteraktion zu verfeinern. Es ist auch zu klären was passiert, wenn mehr als eine Person vor der Wand steht: soll die Wand von mehr als einer Person gleichzeitig genutzt werden können? Wird von Personen erkannt, wer der aktive Benutzer ist (bzw. die aktiven Benutzer sind). Das Benutzungskonzept soll so optimiert sein, dass der überwiegende Teil der Erst-Nutzer die Videowall ohne Probleme bedienen können.
- Der technische Prototyp soll weiterhin zeigen, ob die Software so performant gemacht werden kann, dass die Benutzerinteraktion mit dem Input Device (z.B. Kinect Sensor) schnell genug erkannt wird und Inhalte interaktiv auf der Videowall so flüssig bewegt werden, dass Benutzer das Gefühl der „direkten Manipulation“ erhalten. Gleichzeitig soll der entwickelte technische Prototyp in seiner Grundstruktur (Software Architektur) eine gute Wartbarkeit aufweisen, so dass eine Weiterentwicklung durch andere Personen nach der Abgabe möglich ist.
- Zudem soll geklärt werden, wie die Anwendung im Betrieb mit nachträglich entwickelten Plug-In-Anwendungen (z.B. Mini-Spiele welche von Studenten der HSR entwickelt werden), erweitert werden kann. Eine solche Zusatzanwendung ist zur Demonstration der Machbarkeit zu entwickeln.
- Im Falle einer positiven Gesamtevaluation soll eine Spezifikation für eine zu beschaffende Videowall mit den folgenden Elementen vorgelegt werden: (1) Bildschirme: Anzahl, Grösse, Auflösung, und Positionierung (2) Videokarte(n) (3) Kinect/Input Device(s) (4) Steuerungs PC. Die notwenigen Experimente hierfür (z.B. optimale Auflösung) sind zu spezifizieren und durchzuführen.

Das Resultat dieser Arbeit ist die Analyse von Benutzerbedürfnissen, eine technische Machbarkeitsstudie und eine Empfehlung zu einer sinnvollen Videowall Hardware Konfiguration.

## 4. Zur Durchführung

Mit dem HSR-Betreuer finden in der Regel wöchentliche Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen.

Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse in einem Protokoll zu dokumentieren, welches stets zugreifbar ist.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen (gemäß Projektplan) sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein Feedback. Eine definitive Beurteilung erfolgt aufgrund der am Abgabetermin abgelieferten Dokumentation. Die Evaluation erfolgt aufgrund des separat abgegebenen Kriterienkatalogs in Übereinstimmung mit den Kriterien zur BA Beurteilung. Es sollten hierbei auch die Hinweise aus dem abgegebenen Dokument „Tipps für die Strukturierung und Planung von Studien-, Diplom- und Bachelorarbeiten“ beachtet werden.

## 5. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäß den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollen den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD/DVD in 2 Exemplaren abzugeben.

Zudem ist eine kurze Projektresultatdokumentation im öffentlichen Wiki von Prof. M. Stolze zu erstellen. Diese muss einen Link auf ein öffentlich zugängliches (z.B. YouTube) „Video Poster“ enthalten, welche das Resultat der Arbeit dokumentiert (das Video sollte auch im Original auf der CD/DVD enthalten sein).

Dieses Video sollte nicht mehr als 2 Minuten lang sein. Das Video enthält einen Intro-Screen mit HSR-Logo, Titel der Arbeit, Namen der Studenten und Namen des Betreuers/Dozenten und Industriepartner (Standbild PPT Folie im HSR Corporate Design, 5 sec). Im ersten Inhaltsteil des Videos (30 sec) wird erklärt warum das gelöste Problem relevant ist: Warum braucht es das System, warum ist die Arbeit ohne System mühsam oder unmöglich? Der zweite Inhaltsteil des Videos (max. 50 sec) gibt eine Systemübersicht und zeigt anhand eines Einsatzbeispiels was das Resultat dieser Arbeit wem bringt. Abschluss mit Zusammenfassung der "Nutzer-Benefits" auf PPT Standbild. Im dritten Inhaltsteil (max. 30 sec) werden die „Erfolgenschaften“ der Arbeit prägnant dargestellt. Insbesondere werden gemeisteerte technische und organisatorische Herausforderungen aufgelistet (z.B. Architektur auf PPT Folie im HSR Corporate Design). Im Abspann (5 sec) sollte der Intro-Screen wieder eingeblendet werden. Das Material für die Video-Erstellung wird von der HSR gestellt (Multimediatelle). Das Video ist mit Untertiteln auszustatten. Auf eine Tonspur sowie Musik ist zu verzichten. Beispiele von HSR-Videos zu verschiedenen Arbeiten finden Sie bei YouTube (z.B. Kinect Bodyscanner <http://www.youtube.com/watch?v=Q1ngxAkiaRg>)

## 6. Weitere Regeln und Termine

Im Weiteren gelten die allgemeinen Regeln zu Bachelor und Studienarbeiten „Abläufe und Regelungen Studien- und Bachelorarbeiten im Studiengang Informatik“ (<https://www.hsr.ch/Ablauefe-und-Regelungen-Studie.7479.0.html>)

Der Terminplan ist hier ersichtlich (HSR Intranet)

<https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html>

## 7. Beurteilung

Eine erfolgreiche BA zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Entsprechend sollten ca. 350h Arbeit für die Bachelorarbeit aufgewendet werden. Dies entspricht ungefähr 25h pro Woche (auf 14 Wochen) und damit ca. 3 Tage Arbeit pro Woche pro Student.

Für die Beurteilung sind der HSR-Betreuer verantwortlich unter Einbezug des Beisitzers und allfälligen Feedbacks des Auftraggebers.

Die Bewertung der Arbeit erfolgt entsprechend der verteilten Kriterienliste.

Originale Aufgabenstellung 20. Februar 2012, letzte Überarbeitung 30.5.2012

Rapperswil, 30.5.2012



Prof. Dr. Markus Stolze  
Institut für Software  
Hochschule für Technik Rapperswil

## Erklärung über die eigenständige Arbeit

Wir erklären hiermit,

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

Ort, Datum: Rapperswil, 14. 06. 2012

Lukas Elmer: 

Christina Heidt: 

Delia Treichler: 

## Vereinbarung

### 1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Bachelorarbeit HSR Videowall von Lukas Elmer, Christina Heidt und Delia Treichler unter der Betreuung von Markus Stolze geregelt.

### 2. Urheberrecht

Die Urheberrechte stehen den Studenten zu.

### 3. Verwendung

Die Ergebnisse der Arbeit dürfen sowohl von den Studenten wie von der HSR nach Abschluss der Arbeit verwendet und weiter entwickelt werden.

Ort, Datum: Rapperswil, 13.06.12

Prof. Dr. Markus Stolze, Betreuer HSR: MS

Lukas Elmer, Student HSR: L. Elmer

Christina Heidt, Studentin HSR: C. Heidt

Delia Treichler, Studentin HSR: D. Treichler

## I.1 Anhang C

Folgende Anhänge befinden sich auf der CD:

- Anfragen Grafikkarten
- Code Review Dokument
- Fragebögen
- Fragebögen Statistik
- Hardware Spec Dump
- Lesbarkeit Poster
- Microsoft Imagine Cup Projektplan
- Mail Markus Flückiger
- Mails Code Review
- Notizen Testdurchführung
- Offerten
- Redmine Tickets
- Risikomanagement
- Sitzungsprotokolle
- Stability Test
- User Stories