## Realisierung & Test

### Änderungsgeschichte

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Änderung | Autor |
| 16.04.2012 | 1.0 | Erste Version des Dokuments | DT |
| 22.04.2012 | 1.1 | Usability Tests | CH |
| 23.04.2012 | 1.2 | Systemtests | DT |
| 24.04.2012 | 1.3 | Review | DT |
| 27.04.2012 | 1.4 | Systemtests | DT |
| 07.05.2012 | 1.5 | Systemtests | DT |
| 11.05.2012 | 1.6 | Systemtests | DT |
| 12.05.2012 | 1.7 | Einfügen Testdokumentation aus Domain Analyse | DT |
| 14.05.2012 | 1.8 | Systemtests | DT |
| 18.05.2012 | 1.9 | Usability Test: Reaktion auf Demomodus | CH |
| 18.05.2012 | 1.10 | Review | DT |
| 18.05.2012 | 1.11 | Extension Framework | LE |
| 19.05.2012 | 1.12 | Ergänzung Wizard of Oz - Test mit Bild | DT |
| 22.05.2012 | 1.13 | Review Extension Framework | DT |
| 22.05.2012 | 1.14 | Systemtests | DT |
| 22.05.2012 | 1.15 | Usability Test: Grafisches Design | DT |
| 23.05.2012 | 1.16 | Coding Standards & Dokumentation Quellcode | CH |
| 23.05.2012 | 1.17 | Review Usability Test: Grafisches Design | CH |
| 29.05.2012 | 1.18 | Systemtests | CH |
| 31.05.2012 | 1.19 | 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 |
| 08.06.2012 | 1.23 | Code Metriken | 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 |

### Usability Tests

#### Test 1: Wizard of Oz

Am 27. März 2012 wurde der Test (für die Erarbeitung siehe TODO link Domain Analyse) 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 - Testdurchführung Wizard of Oz mit einem Probanden

Der Test wurde mit sieben Personen durchgeführt, welche das nachfolgende Testszenario (siehe I.1.2.1.1 Testszenario) durchspielten. Die Testpersonen wurden gebeten, laut mitzudenken.

Eine genaue Beschreibung der Testapplikation findet sich im Unterkapitel (TODO ref Domain Analyse Wizard of Oz Umsetzung).

##### 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.

###### 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.

##### 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 (TODO).

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

Tabelle - 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.

#### 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 des Gebäudes 4 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 installiert, 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 verunmöglicht 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, durch welche die meisten Passanten auf dem Weg in die Mensa gehen, ein optimaler Erkennungsabstand von 3-4 Metern.



Abbildung - Test 2: Reaktion der Nutzer

Schon das Aufstellen der Geräte im Gebäude 4 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 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 Handpointer bei diesem Usability Test merklich. Dies führte teilweise 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 bei den bevorstehenden Tätigkeiten des Teams aufgeführt und als User Story erfasst.

#### Test 3: Reaktion auf Demomodus



Abbildung - Test 3: Reaktion auf Demomodus

Nachdem der Demomodus implementiert war, sollte auch dieser wieder von den Nutzern geprüft werden. Daher testete das Team den Demomodus am 18.05.2012. Der Aufbau verlief wieder gleich wie schon bei I.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 dem Testaufbau 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 nach dem 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 sollte am besten schon beim Herunterzählen des Countdowns (also noch im Demomodus) angezeigt werden, damit dem Nutzer klar ist, dass er erkannt wurde.
* Bei der Poster-App muss der Teaser-Text 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 wieder einem Usability Test unterzogen.

#### Test 4: Grafisches Design

Mit diesem Usability Test sollte am 22.05.2012 getestet werden, ob das grafische Design verständlich ist. Weiter wurde sichergestellt, dass sich die Verbesserungen am Demomodus, welche nach dem vorhergehenden Usability Test (siehe Unterkapitel I.1.3.3 Test 3: Reaktion auf Demomodus) vorgenommen wurden, bewähren. Der Aufbau verlief wieder gleich wie schon bei I.2.2 Test 2: Reaktion der Nutzer.



Abbildung - 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 daher 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 nach dem, 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 Punkt wird noch im Rahmen dieser Arbeit umgesetzt. Die anderen zwei Beobachtungen wurden als User Stories in den Backlog aufgenommen (TODO Referenz Backlog).

### Unit Tests

#### Videowall-Applikation

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

Abbildung - Unit Tests VideoWall

#### Poster-Applikation

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



Abbildung - Unit Tests PosterApp

#### Mittagsmenu-Applikation

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



Abbildung - Unit Tests LunchMenuApp

### Systemtests

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

#### 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 |

#### 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 |

#### 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 |

#### 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 |
| 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 |

#### 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 |

#### 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 |
| 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 |

#### 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 |
| 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 |

#### 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 |
| 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 |

#### Sprint 15/16

Im Sprint 15/16 wurde Refactoring betrieben.

Testperson: Christina Heidt A = im Architekturprototypen enthalten

| 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 |
| 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 |

### Stabilitätstest

Um die Stabilität der entwickelten Applikation mit Plug-Ins zu testen, wurde diese über 111 Stunden lang auf dem Testsetup mit 9 Bildschirmen (TODO: ref) und einer hohen Auflösung laufen gelassen. Die Applikation befand sich zum grossen Teil im Demomodus, wobei die Applikation aber zwischendurch immer wieder bedient wurde.

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

* Anzahl der offenen Handles: Diese Zahl beschreibt die Anzahl der geöffneten Dateien, Pipes usw. Befindet sich ein Memory Leak in der Applikation, 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 ansteigt.
* Speicherverbrauch in MB: Diese Zahl beschreibt, wie viel Speicher die Applikation beim Betriebssystem reserviert hat. Obwohl beim Entwickeln mit C# dies meistens von der .NET Runtime gemacht wird, so passiert es bei einem Memory Leak, dass dies nicht richtig gemacht werden kann. Dies kann z.B. dann passieren, wenn ein Objekt nicht mehr gebraucht wird, aber trotzdem noch in irgendeiner aktiven Datenstruktur referenziert wird.

In der folgenden Grafik ist ein Überblick über die Entwicklung der Systemressourcen zu sehen. Dabei kann sehr gut erkannt werden, dass sich keine oder höchstens marginale Memory Leaks in der Applikation befinden, da sich einerseits die Anzahl der Handles zwischen 700 und 800 stabilisiert haben und dass der temporäre Speicher 275 MB nie überschritt und sich bei etwa 211 MB eingependelt hat.

Abbildung - Entwicklung der Systemressourcen der laufenden Applikation über 111 Stunden

Zu gewissen Zeiten konnte beobachtet werden, dass der Speicherverbrauch für eine kurze Zeit, ein paar wenige Sekunden, bis zu 275 MB angewachsen war. 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 folgenden Tabelle sind einige weitere Details zum Stabilitätstest zu sehen, die in der oberen Grafik nicht erkennbar sind.

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

Abbildung - 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 beendet. Die genauen Daten befinden sich im Anhang (TODO: ref).

Aus diesem Stabilitätstest über 111h kann geschlossen werden, dass der Prototyp der Videowall gut über eine Dauer von 20h oder 24h betrieben werden kann, ohne dass mit Instabilitäten gerechnet werden muss.

### Code Dokumentation

##### Übereinstimmung mit Architektur

Im Kapitel (TODO Entwurf Architektur) wird die Schichtenarchitektur aufgezeigt. Wie aus den Abbildungen Abbildung 10 - Dependency Graph, Videowall-Applikation, Abbildung 11 - Dependency Graph, Poster-Applikation und Abbildung 12 - Dependency Graph, Mittagsmenu-Applikation ersichtlich ist, stimmt die umgesetzte mit der geplanten Architektur überein. Die Assemblies wurden analog zu den Schichten aufgebaut.

###### Videowall-Applikation

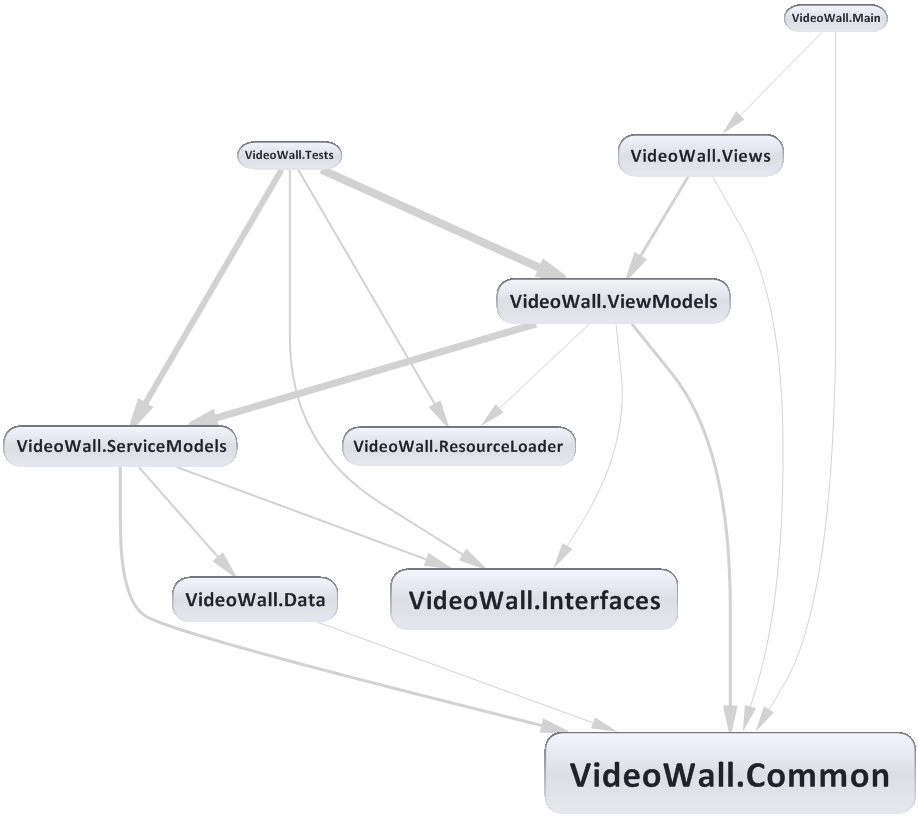


Abbildung 10 - Dependency Graph, Videowall-Applikation

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

###### Poster-Applikation

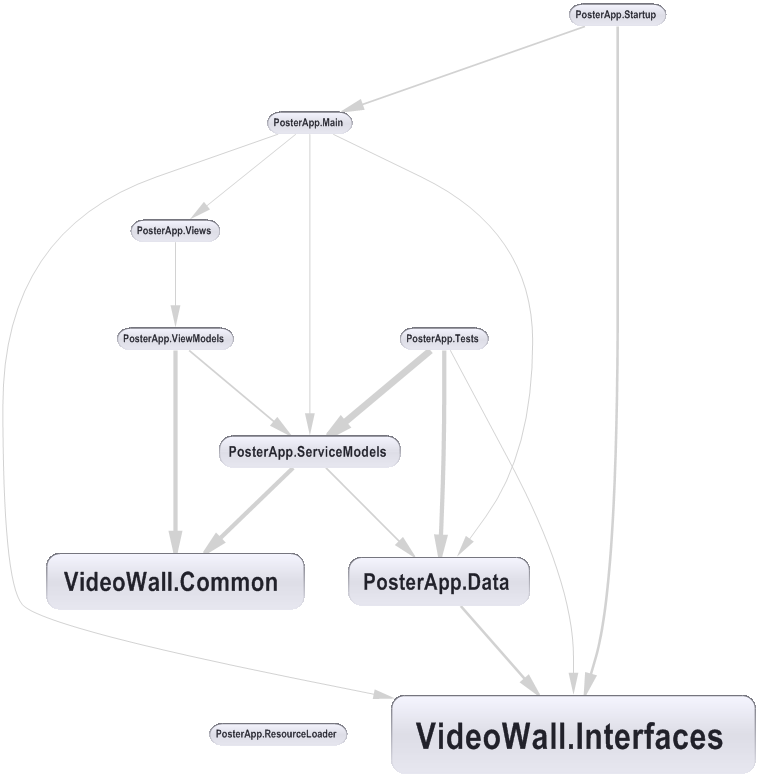


Abbildung - Dependency Graph, Poster-Applikation

Das Assembly PosterApp.Startup dient dazu die Applikation zu starten. So kann die Applikation unabhängig von der Videowall-Applikation gestartet und geprüft werden. Im Assembly PostApp.Main wird das eigentliche Plug-in für den Export definiert (siehe hierzu TODO Entwurf Plug-in Framework).

###### Mittagsmenu-Applikation

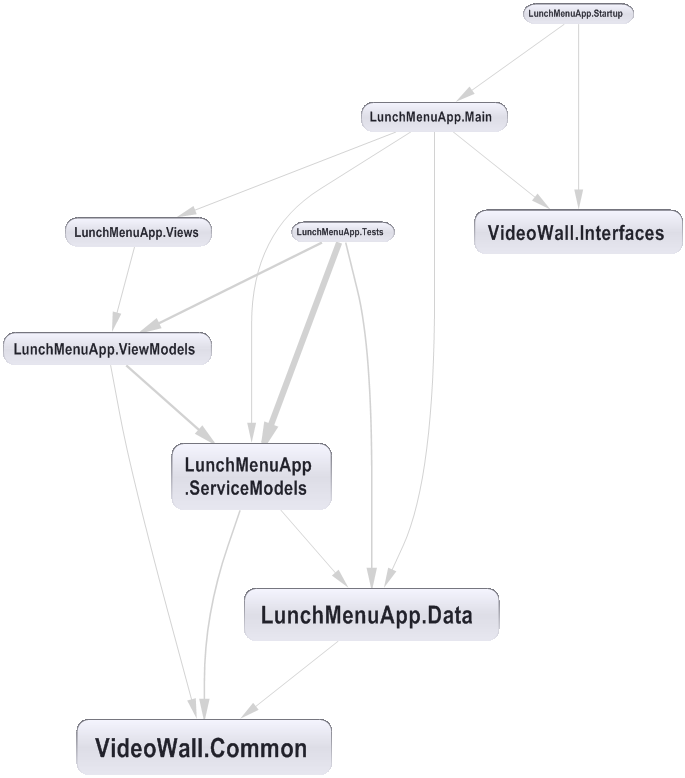


Abbildung - Dependency Graph, Mittagsmenu-Applikation

Das Assembly LunchMenuApp.Startup dient dazu die Applikation zu starten. So kann die Applikation unabhängig von der Videowall-Applikation gestartet und geprüft werden. Im Assembly LunchMenuApp.Main wird das eigentliche Plug-in für den Export definiert (siehe hierzu TODO Entwurf Plug-in Framework).

#### Code Statistik

##### Testabdeckung

###### 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 11 - Test Coverage VideoWall ersichtlich ist, bestehen grosse Unterschiede in der Testabdeckung in den verschiedenen Projekten.



Abbildung - 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 Unterkapitel I.1.4 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 Unterkapitel I.1.4 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.

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



Abbildung - Test Coverage PosterApp

###### 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 - Test Coverage LunchMenuApp

##### Lines of Code (LOC)

Für die Auswertung der Lines of Code (LOC) wurden die Angaben von Visual Studio (siehe I.1.5.2.1 Visual Studio) übernommen.

|  |  |
| --- | --- |
| Hierarchie | Lines of Code (LOC) |
| ***VideoWall*** |  |
| VideoWall.Common | 85 |
| VideoWall.Data | 92 |
| VideoWall.Interfaces | 4 |
| VideoWall.Main | 0 |
| VideoWall.ResourceLoader | 9 |
| VideoWall.ServiceModels | 381 |
| VideoWall.Tests | 179 |
| VideoWall.ViewModels | 232 |
| VieoWall.Views | 5 |
| **Total VideoWall** | **987** |
|  |  |
| ***PosterApp*** |  |
| PosterApp.Data | 6 |
| PosterApp.Main | 18 |
| PosterApp.ResourceLoader | 0 |
| PosterApp.ServiceModels | 12 |
| PosterApp.Startup | 9 |
| PosterApp.Tests | 63 |
| PosterApp.ViewModels | 19 |
| PosterApp.Views | 0 |
| **Total PosterApp** | **127** |
|  |  |
| ***LunchMenuApp*** |  |
| LunchMenuApp.Data | 13 |
| LunchMenuApp.Main | 19 |
| LunchMenuApp.ServiceModels | 66 |
| LunchMenuApp.Startup | 3 |
| LunchMenuApp.Tests | 139 |
| LunchMenuApp.ViewModels | 29 |
| LunchMenuApp.Views | 0 |
| **Total LunchMenuApp** | **269** |
|  |  |
| **Total Gesamt** | **1383** |

Tabelle 2 - Lines of Code (LOC)

#### Code Qualität

Für die Metrikanalyse des Codes wurden das Visual Studio selbst und NDepend (TODO ref Kapitel Tools) verwendet. Somit ist eine objektivere Bewertung des Codes möglich.

##### 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 12 - Metriken Visual Studio, Videowall-Applikation, Abbildung 13 - Metriken Visual Studio, Poster-Applikation und Abbildung 14 - 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.

###### Videowall Applikation



Abbildung - Metriken Visual Studio, Videowall-Applikation

###### Poster-Applikation



Abbildung - Metriken Visual Studio, Poster-Applikation

###### Mittagsmenu-Applikation



Abbildung - Metriken Visual Studio, Mittagsmenu-Applikation

##### NDepend

Neben der Analyse mit Visual Studio (siehe I.1.5.1.2.1 Visual Studio 10) wurde der Code zusätzlich mit NDepend ausgewertet. Folgende Statistiken wurden direkt von NDepend für die verschiedenen Applikationen generiert:

TODO: braucht es hier nicht noch eine Erklärung, ein Fazit?

###### Videowall-Applikation



Abbildung - Metriken NDepend, Übersicht, Videowall-Applikation



Abbildung - Metriken NDepend, Assemblies, Videowall-Applikation

###### Poster-Applikation



Abbildung - Metriken NDepend, Übersicht, Poster-Applikation



Abbildung - Metriken NDepend, Assemblies, Poster-Applikation

###### Mittagsmenu-Applikation



Abbildung - Metriken NDepend, Übersicht, Mittagsmenu-Applikation



Abbildung - Metriken NDepend, Assemblies, Mittagsmenu-Applikation

##### Code Warnungen

Wie aus den nachfolgenden Abbildungen ersichtlich ist, treten beim Kompilieren des Codes der verschiedenen Applikationen keine Fehler oder Warnungen auf.



Abbildung - Warnings, Videowall-Applikation



Abbildung - Warnings, Poster-Applikation



Abbildung - Warnings, Mittagsmenu-Applikation

#### 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.

##### C# Namenskonventionen

Folgende Namenskonventionen wurden verwendet:



Abbildung - Naming Style

##### Formatierungsstil

**Braces Layout**

Geschweifte Klammern befinden sich auf einer neuen Zeile.



Abbildung 29 – Formatierungsstil, Braces Layout

**Line Breaks and Wrapping**

Lange Zeilen (>120 Zeichen) werden umgebrochen.



Abbildung - Formatierungsstil, Line Breaks and Wrapping

##### Auswertung durch Resharper

Durch den Resharper können Coding Issues angezeigt werden. Diese werden für die drei Projekte nachfolgend aufgezeigt und erklärt.

**Videowall-Applikation**

Abbildung - 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 instanziiert. 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 - 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 - 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.

##### Cleanup

Für das Cleanup des Codes wurden folgende Einstellungen vorgenommen:



Abbildung - Cleanup Einstellungen

#### Dokumentation Quellcode

Zur Dokumentation des Quellcodes wurde GhostDoc verwendet. Dadurch war die Dokumentation einiges einfacher, da sinnvolle Standardkommentare generiert werden, die bei Bedarf erweitert werden können. Durch die geschriebenen XML-Kommentare ist es auch möglich, eine Dokumentation zu generieren (für mehr Informationen siehe Unterkapitel I.1 Generierung der Dokumentation).

Es wurden alle Interfaces, sowie Methoden oder Properties, welche mit Hilfe eines Kommentars besser verstanden werden, kommentiert. Ausgenommen davon sind die Tests und die XAML-Dateien. Durch die Kommentare sind der Programmcode und besonders komplexere Methoden für Entwickler leichter verständlich.

##### Generierung der Dokumentation

Zur Generierung einer Dokumentation des Quellcodes wird die Software Sandcastle Help File Builder (SHFB)[[1]](#footnote-1) benötigt.



Abbildung - Sandcastle Help File Builder, Hinzufügen der Visual Studio Solution

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 Dokumentname („HtmlHelpName“) und der Speicherort („OutputPath“) des Help Files spezifiziert werden.  
Danach kann die Dokumentation mittels des Build-Befehls erstellt werden (Documentation -> Build Project).

### Installationsdokumentation

#### 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 I.1.6.3 Plug-ins aufgebaut werden.

#### Dokumentation für Entwickler

Damit die Applikation weiterentwickeln werden kann, muss ein SVN Checkout auf dem gesamten SVP Repository durchgeführt werden. Dazu wird ein SVN Tool wie z.B. TortoiseSVN benötigt (siehe Tools TODO).

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 I.1.5.4 Konfigurationsdatei erläutert.   
Weiter können der Applikation Plug-ins hinzugefügt werden, dies wird im nachfolgenden Kapitel (I.1.6.3 Plug-ins) aufgezeigt.

#### 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 I.1.6.4.6 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 I.1.6.4.6 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.

#### Konfigurationsdatei

Die Einstellungen, welche in der Konfigurationsdatei getätigt werden können, sind nachfolgend aufgezeigt.

##### 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 - app.config, Konfigurationssektionen

##### 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 37 - app.config, Sektion Unity, Namespaces&Assemblies

###### 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 - app.config, Sektion Unity, Mapping von Interfaces auf Klassen

###### 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 *AutoPlayFileSleketonReader* spielt das File immer wieder von Neuem ab. Das abzuspielende File mit den Skelettdaten kann über den Typ *KinectReplayFile* definiert werden, was im Unterkapitel I.1.5.3.9 KinectReplayFile erläutert wird.

        <!-- FileSkeletonReader -->

        <!-- AutoPlayFileSkeletonReader -->

        <!-- KinectSkeletonReader -->

        <type type="ISkeletonReader" mapTo="KinectSkeletonReader">

          <lifetime type="singleton"/>

        </type>

Abbildung 39 - app.config, Sektion Unity, Mapping für ISkeletonReader

###### 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 I.1.6.4.2.1 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 - app.config, Sektion Unity, Mapping für ICursorViewModel

##### Padding

Damit die Applikation angenehm mit Kinect zu bedienen ist, wurde einer Interaktionszone für die Hand definiert. Das Prinzip dieser Zone wird im Kapitel (TODO. Entwurf, Handtracking) genauer erläutert. Die rote Fläche in der Abbildung 11 - 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 rote 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 11 - Interaktionszone ersichtlich.



Abbildung 41 - 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 42 - app.config, Sektion Unity, Padding

###### 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 43 - app.config, Sektion Unity, Pfad zu den Plug-ins

###### 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="#ff548c86"/>

                <value value="#ff7b6951"/>

                <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 - 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.

###### 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 Inhalte des aktuell angewählten auf der Videowall anzeigt.

        <type type="Player" mapTo="Player">

          <lifetime type="singleton"/>

        </type>

        <type type="MenuViewModel" mapTo="MenuViewModel">

          <lifetime type="singleton"/>

        </type>

Abbildung 45 - app.config, Sektion Unity, Singleton

###### KinectReplayFile

Wird im app.config beim Mapping auf den ISkeletonReader (siehe I.1.6.4.2.2 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:

 <type type="KinectReplayFile" mapTo="KinectReplayFile">

          <lifetime type="singleton"/>

          <constructor>

            <param name="path" value="..\..\..\..\..\..\kinect\_records\20120312\_lukas\\_1.replay"/>

          </constructor>

        </type>

Abbildung 46 - app.config, Sektion Unity, KinectReplayFile

Der Pfad zum Replay-File ist relativ anzugeben.

##### 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 - app.config, Runtime

1. <http://shfb.codeplex.com/> [↑](#footnote-ref-1)