## Realisierung & Test

[I.1.1 Änderungsgeschichte 3](#_Toc327447872)

[I.1.2 Usability Tests 5](#_Toc327447873)

[I.1.2.1 Test 1: Wizard of Oz 5](#_Toc327447874)

[I.1.2.1.1 Testszenario 5](#_Toc327447875)

[I.1.2.1.1.1 Aufgabe 5](#_Toc327447876)

[I.1.2.1.2 Resultat & Fazit 5](#_Toc327447877)

[I.1.2.2 Test 2: Reaktion der Nutzer 6](#_Toc327447878)

[I.1.2.3 Test 3: Reaktion auf Demomodus 7](#_Toc327447879)

[I.1.2.4 Test 4: Grafisches Design 8](#_Toc327447880)

[I.1.3 Unit Tests 9](#_Toc327447881)

[I.1.3.1 Videowall-Applikation 9](#_Toc327447882)

[I.1.3.2 Poster-Applikation 9](#_Toc327447883)

[I.1.3.3 Mittagsmenu-Applikation 9](#_Toc327447884)

[I.1.4 Systemtests 11](#_Toc327447885)

[I.1.4.1 Sprint 7 11](#_Toc327447886)

[I.1.4.2 Sprint 8 11](#_Toc327447887)

[I.1.4.3 Sprint 9 11](#_Toc327447888)

[I.1.4.4 Sprint 10 12](#_Toc327447889)

[I.1.4.5 Sprint 11 13](#_Toc327447890)

[I.1.4.6 Sprint 12 13](#_Toc327447891)

[I.1.4.7 Sprint 13 15](#_Toc327447892)

[I.1.4.8 Sprint 14 16](#_Toc327447893)

[I.1.4.9 Sprint 15/16 17](#_Toc327447894)

[I.1.5 Stabilitätstest 19](#_Toc327447895)

[I.1.6 Dokumentation der Realisierung 21](#_Toc327447896)

[I.1.6.1 Übereinstimmung mit Architektur 21](#_Toc327447897)

[I.1.6.1.1 Videowall-Applikation 21](#_Toc327447898)

[I.1.6.1.2 Poster-Applikation 21](#_Toc327447899)

[I.1.6.1.3 Mittagsmenu-Applikation 22](#_Toc327447900)

[I.1.6.2 Code Statistik 23](#_Toc327447901)

[I.1.6.2.1 Testabdeckung 23](#_Toc327447902)

[I.1.6.2.1.1 Videowall-Applikation 23](#_Toc327447903)

[I.1.6.2.1.2 Poster-Applikation 26](#_Toc327447904)

[I.1.6.2.1.3 Mittagsmenu-Applikation 26](#_Toc327447905)

[I.1.6.2.2 Lines of Code (LOC) 27](#_Toc327447906)

[I.1.6.3 Code Qualität 28](#_Toc327447907)

[I.1.6.3.1 Visual Studio 28](#_Toc327447908)

[I.1.6.3.1.1 Videowall Applikation 28](#_Toc327447909)

[I.1.6.3.1.2 Poster-Applikation 28](#_Toc327447910)

[I.1.6.3.1.3 Mittagsmenu-Applikation 29](#_Toc327447911)

[I.1.6.3.2 NDepend 29](#_Toc327447912)

[I.1.6.3.2.1 Videowall-Applikation 29](#_Toc327447913)

[I.1.6.3.2.2 Poster-Applikation 29](#_Toc327447914)

[I.1.6.3.2.3 Mittagsmenu-Applikation 30](#_Toc327447915)

[I.1.6.3.3 Code Warnungen 30](#_Toc327447916)

[I.1.6.4 Coding Standards 31](#_Toc327447917)

[I.1.6.4.1 C# Namenskonventionen 32](#_Toc327447918)

[I.1.6.4.2 Formatierungsstil 32](#_Toc327447919)

[I.1.6.4.3 Auswertung durch ReSharper 32](#_Toc327447920)

[I.1.6.4.4 Cleanup 36](#_Toc327447921)

[I.1.6.5 Dokumentation Quellcode 36](#_Toc327447922)

[I.1.6.5.1 Generierung der Dokumentation 36](#_Toc327447923)

[I.1.7 Beschreibung der Applikationen 37](#_Toc327447924)

[I.1.7.1 Hauptapplikation 37](#_Toc327447925)

[I.1.7.2 Plug-ins 37](#_Toc327447926)

[I.1.7.2.1 Poster Plug-in 37](#_Toc327447927)

[I.1.7.2.2 Mittagsmenu Plug-in 37](#_Toc327447928)

[I.1.7.2.3 Experiment Browser Plug-in 37](#_Toc327447929)

[I.1.7.2.4 Experiment Diagnostic Plug-in 37](#_Toc327447930)

[I.1.7.2.5 Experiment ShapeGame Plug-in 37](#_Toc327447931)

[I.1.7.3 Mini-Applikationen 37](#_Toc327447932)

[I.1.7.3.1 DemoMode 37](#_Toc327447933)

[I.1.7.3.2 DesignMenu 38](#_Toc327447934)

[I.1.7.3.3 DesignPosterNavigationButtons 38](#_Toc327447935)

[I.1.7.3.4 HandCursorDemoApp 38](#_Toc327447936)

[I.1.7.3.5 KinectHandTracker 38](#_Toc327447937)

[I.1.7.3.6 KinectRecorder 38](#_Toc327447938)

[I.1.7.3.7 KinectWpfViewers 38](#_Toc327447939)

[I.1.7.3.8 ObjectTrackingVisualizer 38](#_Toc327447940)

[I.1.7.3.9 PdfConverter 38](#_Toc327447941)

[I.1.7.3.10 PluginDemo 38](#_Toc327447942)

[I.1.7.3.11 TestXna 38](#_Toc327447943)

[I.1.7.3.12 VideoWithWPF 38](#_Toc327447944)

[I.1.7.3.13 WizardOfOzTest 39](#_Toc327447945)

[I.1.7.3.14 KinectCursorSmoothing 39](#_Toc327447946)

[I.1.8 Code Reviews 40](#_Toc327447947)

[I.1.8.1 Übersicht 40](#_Toc327447948)

[I.1.8.2 Kriterien 40](#_Toc327447949)

[I.1.8.2.1 Code Style Analyse 40](#_Toc327447950)

[I.1.8.2.2 Exception Handling 40](#_Toc327447951)

[I.1.8.2.3 Flow Control 40](#_Toc327447952)

[I.1.8.2.4 Naming 41](#_Toc327447953)

[I.1.8.2.5 Tools 41](#_Toc327447954)

[I.1.8.3 Code Review vom 19.04.2012 41](#_Toc327447955)

[I.1.8.3.1 Bewertung der Kriterien 41](#_Toc327447956)

[I.1.8.3.1.1 Code Style Analyse 41](#_Toc327447957)

[I.1.8.3.1.2 Exception Handling 42](#_Toc327447958)

[I.1.8.3.1.3 Flow Control 42](#_Toc327447959)

[I.1.8.3.1.4 Naming 42](#_Toc327447960)

[I.1.8.3.1.5 Tools 42](#_Toc327447961)

[I.1.8.4 Code Review vom 03.05.2012 42](#_Toc327447962)

[I.1.8.4.1 Bewertung der Kriterien 43](#_Toc327447963)

[I.1.8.4.1.1 Code Style Analyse 43](#_Toc327447964)

[I.1.8.4.1.2 Exception Handling 44](#_Toc327447965)

[I.1.8.4.1.3 Flow Control 44](#_Toc327447966)

[I.1.8.4.1.4 Naming 44](#_Toc327447967)

[I.1.8.4.1.5 Tools 44](#_Toc327447968)

[I.1.8.5 Code Review vom 05.06.2012 44](#_Toc327447969)

[I.1.8.5.1 Bewertung der Kriterien 45](#_Toc327447970)

[I.1.8.5.1.1 Code Style Analyse 45](#_Toc327447971)

[I.1.8.5.1.2 Exception Handling 45](#_Toc327447972)

[I.1.8.5.1.3 Flow Control 46](#_Toc327447973)

[I.1.8.5.1.4 Naming 46](#_Toc327447974)

[I.1.8.5.1.5 Tools 46](#_Toc327447975)

### 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 |

### 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 1 - 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 1 - 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 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 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, welche die meisten Passanten auf dem Weg in die Mensa durchlaufen, ein optimaler Erkennungsabstand von drei bis vier Metern.



Abbildung 2 - 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.

#### Test 3: Reaktion auf Demomodus



Abbildung 3 - 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 I.1.2.2 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 sollte am 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.

#### 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 Demomodus - Usability Test (siehe Unterkapitel I.1.2.3 Test 3: Reaktion auf Demomodus) vorgenommen wurden, bewähren. Der Aufbau verlief wieder gleich wie schon beim Test I.1.2.2 Test 2: Reaktion der Nutzer.



Abbildung 4 - 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 (TODO Referenz Backlog).

### Unit Tests

#### Videowall-Applikation

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

Abbildung 5 - Unit Tests VideoWall

#### Poster-Applikation

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



Abbildung 6 - Unit Tests PosterApp

#### Mittagsmenu-Applikation

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



Abbildung 7 - 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 |

Tabelle 2 - Systemtests Sprint 7

#### 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 3 - Systemtests Sprint 8

#### 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 4 - Systemtests Sprint 9

#### 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 |

Tabelle 5 - Systemtests Sprint 10

#### 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 6 - Systemtests Sprint 11

#### 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 |

Tabelle 7 - Systemtests Sprint 12

#### 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 |

Tabelle 8 - Systemtests Sprint 13

#### 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 |

Tabelle 9 - Systemtests Sprint 14

#### 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 |

Tabelle 10 - Systemtests Sprint 15/16

### Stabilitätstest

Um die Stabilität der entwickelten Applikation mit Plug-ins zu testen wurde diese über eine Zeitspanne von 111 Stunden auf dem Videowall-Testsetup (TODO: ref) 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 8 - Entwicklung der Systemressourcen der laufenden Applikation über 111 Stunden, MB sind 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 9 - 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 (TODO: ref).

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.

### Dokumentation der Realisierung

#### Übereinstimmung mit Architektur

Im Kapitel (TODO Entwurf Architektur) ist die Schichtenarchitektur aufgezeigt. Wie in den nachfolgenden 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

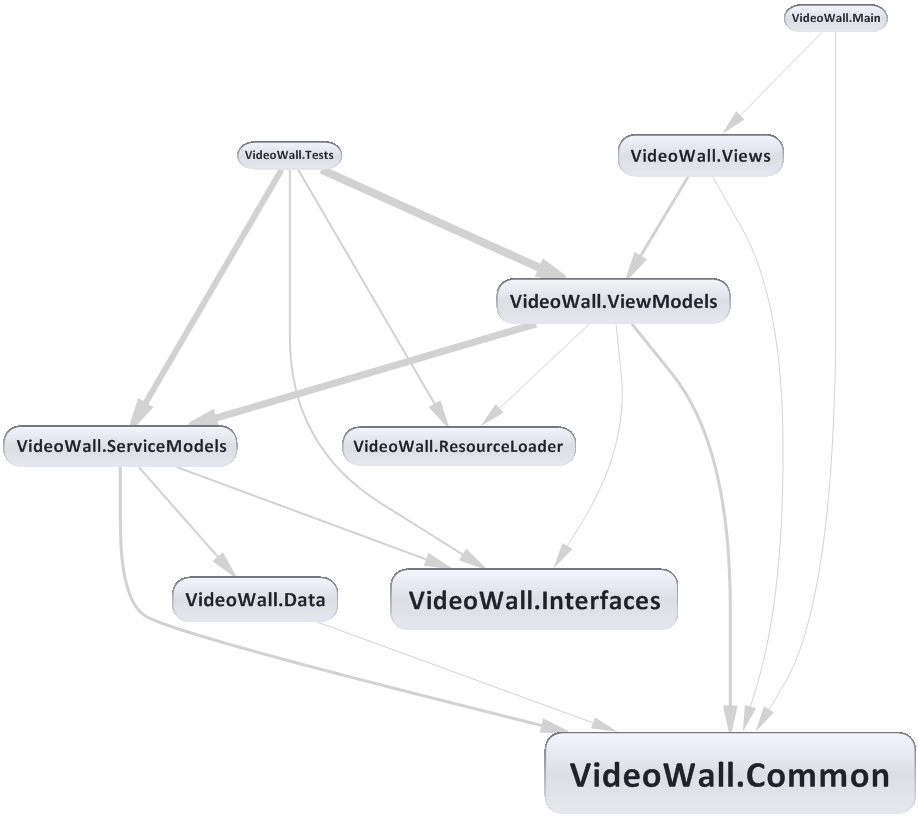


Abbildung 10 - Dependency Graph, Videowall-Applikation

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

##### Poster-Applikation

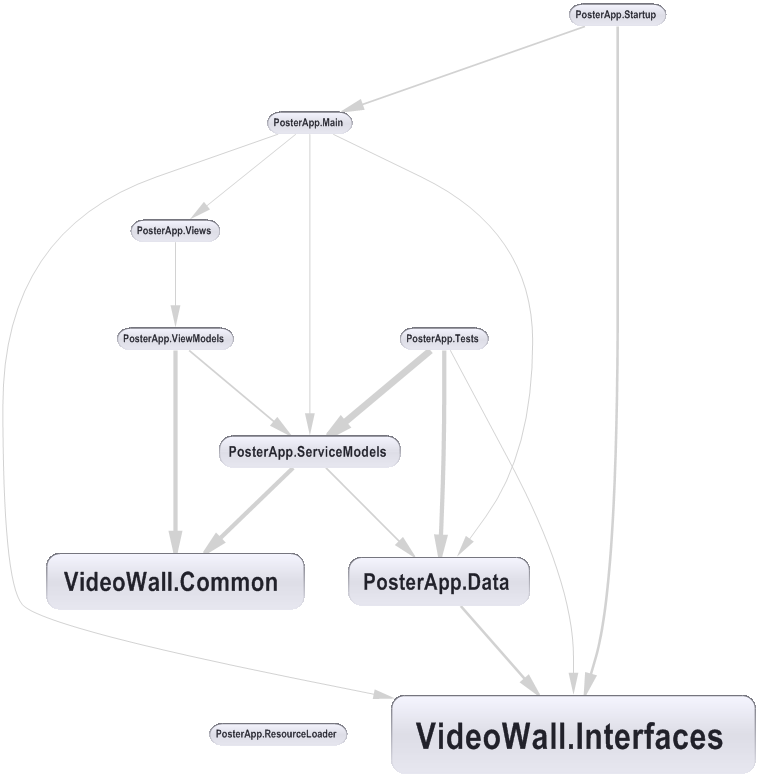


Abbildung 11 - Dependency Graph, Poster-Applikation

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 PostApp.Main ist das eigentliche Plug-in für den Export definiert (siehe hierzu TODO Entwurf Plug-in Framework).

##### Mittagsmenu-Applikation

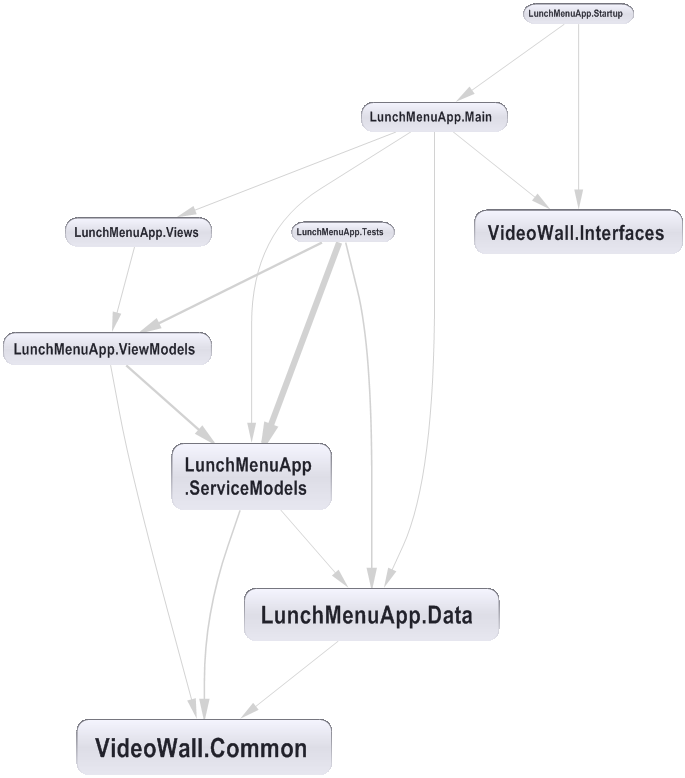


Abbildung 12 - Dependency Graph, Mittagsmenu-Applikation

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



Abbildung 13 - 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 0

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 0

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



Abbildung 14 - 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 15 - Test Coverage LunchMenuApp

##### Lines of Code (LOC)

Für die Auswertung der Lines of Code (LOC) wurden die Angaben von Visual Studio (siehe I.1.6.3.1 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 11 - 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 16 - Metriken Visual Studio, Videowall-Applikation, Abbildung 17 - Metriken Visual Studio, Poster-Applikation und Abbildung 18 - 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 16 - Metriken Visual Studio, Videowall-Applikation

###### Poster-Applikation



Abbildung 17 - Metriken Visual Studio, Poster-Applikation

###### Mittagsmenu-Applikation



Abbildung 18 - Metriken Visual Studio, Mittagsmenu-Applikation

##### NDepend

Neben der Analyse mit Visual Studio (siehe I.1.6.3.1 Visual Studio) wurde der Code zusätzlich mit NDepend ausgewertet. Folgende Statistiken wurden direkt von NDepend für die verschiedenen Applikationen generiert:

###### Videowall-Applikation



Abbildung 19 - Metriken NDepend, Übersicht, Videowall-Applikation



Abbildung 20 - Metriken NDepend, Assemblies, Videowall-Applikation

###### Poster-Applikation



Abbildung 21 - Metriken NDepend, Übersicht, Poster-Applikation



Abbildung 22 - Metriken NDepend, Assemblies, Poster-Applikation

###### Mittagsmenu-Applikation



Abbildung 23 - Metriken NDepend, Übersicht, Mittagsmenu-Applikation



Abbildung 24 - 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 25 - Warnings, Videowall-Applikation



Abbildung 26 - Warnings, Poster-Applikation



Abbildung 27 - 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 28 - 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 30 - 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 31 - 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 32 - 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 33 - 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 34 - Cleanup Einstellungen

#### 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 I.1 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.

##### Generierung der Dokumentation

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



Abbildung 35 - 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 Dokumentname („HtmlHelpName“) und der Speicherort („OutputPath“) des Help Files spezifiziert werden.
* Danach kann die Dokumentation mittels des Build-Befehls erstellt werden (Documentation -> Build Project).

### Beschreibung der Applikationen

#### Hauptapplikation

Die Videowall Applikation stellt die Hauptapplikation dar. Sie sorgt dafür, dass die verschiedenen Plug-ins geladen (diese Funktion wird im Kapitel TODO Ref Entwurf Plug-in Framework erläutert) und visualisiert werden. Auch das Handtracking (siehe TODO Entwurf Handtracking) ist darin umgesetzt. Zudem stellt sie die verschiedenen Interfaces zu Verfügungen, welche von den Plug-ins implementiert werden.

#### Plug-ins

Durch das Team wurden zwei Inhalte für die Videowall definiert (Poster und Mittagsmenu). Diese werden nachfolgend aufgezeigt.

Zusätzlich wurden drei weitere Plug-Ins entwickelt, die aber nur als Experimente anzusehen sind. Sie wurden keinen Code Reviews unterzogen und sie erfüllen weder die für die Videowall definierte Code Qualität oder Code Richtlinien und dienen nur dazu, Möglichkeiten von Plug-Ins aufzuzeigen. Diese werden nachfolgend auch kurz beschrieben.

##### 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.

##### Mittagsmenu Plug-in

Das Mittagsmenu Plug-in zeigt das aktuelle Mensa-Mittagsmenu der SV Group der HSR an.

##### Experiment Browser Plug-in

Das Browser Plug-in zeigt, wie Inhalte direkt aus dem Internet auf der Videowall angezeigt werden können.

##### Experiment Diagnostic Plug-in

Das Diagnostic Plug-in zeigt eine Applikation, das ein minimales Plug-in implementiert werden kann und wie einfach es ist, Buttons in eine Applikation einzubinden.

##### 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.

#### 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.

##### 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.

##### DesignMenu

Die DesignMenu-Applikation wurde benutzt, um direkt in Expression Blend verschiedene Designs für die Tabs des Menus zu erstellen und untereinander zu vergleichen.

##### 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.

##### 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.

##### 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.

##### 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.

##### KinectWpfViewers

Dies sind vordefinierte Klassen des Kinect SDKs. Sie wurden als Beispiel und als Hilfe, beispielsweise beim Einbinden des Skeletts in die View, verwendet.

##### 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.

##### 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.

##### PluginDemo

PluginDemo ist ein Beispielprojekt für ein mit MEF erstelltes Plugin und zeigt, wie dieses in die Hauptapplikation eingebunden werden kann.

##### TestXna

Ein minimalistisches „Game“, das mit XNA implementiert ist und über DirectX läuft. Diese wurde dazu benötigt, um DirectX auf dem Testsetup zu testen.

##### 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.

##### WizardOfOzTest

Die WizardOfOzTest-Applikation wurde für den Wizard of Oz Test zu Beginn der Implementation (I.1.2.1 Test 1: Wizard of Oz) verwendet.

##### KinectCursorSmoothing

Ein Ansatz für das Cursor Smoothing mit Kinect.

### 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. Anderseits 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.

#### 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.

#### Kriterien

Die Kriterienliste wurde am 19.04.12 durch das Team erstellt.

##### 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 |  |  |

##### Exception Handling

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Der Code enthält keine abgefangenen und ignorierten Ausnahmen |  |  |
| Fehler in asynchronen Prozessen werden mittels Event weitergeleiltet |  |  |
| 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 |  |  |

##### 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 |  |  |

##### Naming

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Namen der Klassen / Variabeln sind selbstbeschreibend |  |  |
| Verwirrende oder falsche Namen sind nicht vorhanden |  |  |
| Interface-Klassen beginnen immer 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 |  |  |

##### Tools

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben |  |  |
| Die Errors von FxCop werden wo sinnvoll behoben |  |  |

#### 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 12 - Annotationen und Kommentare Code Review 19.04.2012

##### Bewertung der Kriterien

###### 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 |  |

###### Exception Handling

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Der Code enthält keine abgefangenen und ignorierten Ausnahmen | ? |  |
| Fehler in asynchronen Prozessen werden mittels Event weitergeleiltet | ? |  |
| 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 |  | ? |

###### 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 |  |

###### Naming

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Namen der Klassen / Variabeln 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 |

###### Tools

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben | X |  |
| Die Errors von FxCop werden wo sinnvoll behoben | X |  |

#### 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 13 - 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 evt. Common GUI-Library (ExtendedVisualTreeHelper.cs) | - | - |
| Verletzung des Information Experts in ImageExtension.cs. Initialisierungsmethode LunchMenu.CreateFrom (string fileName) nutzen | 05.06.2012 | LE |
| UI Elemente im ViewModel sind unschön und zerstört die Testbarkeit -> 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  Dispose Pattern weggelassen, da während der Applikation keine Resourcen durch dieses Pattern freigegeben werden müssen. | |
| Anstelle Loop HitTest-Methode verwenden (Methode OnModelChanged  in HitTestHelper) | - | - |
| ViewModel-Verschachtelung: MenuViewModel weiss mehr/mächtiger als MainWindowViewModel | 05.06.2012, LE  MainWindowViewModel kennt das MenuViewModel, nicht aber umgekehrt. So wie es jetzt gelöst ist sind die Informationen nicht redundant. Evt. Könnte MenuViewModel in AppsViewModel umbenannt werden. | |
| 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.xmal.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 13 - Annotationen und Kommentare Code Review 03.05.2012

Die original notierten Kommentare und Annotationen von Silvan Gehrig und Michael Gfeller sind im Anhang (TODO: Code Review File in Anhang und verlinken) zu finden.

##### Bewertung der Kriterien

Die Kriterienbewertung wurde von Silvan Gehrig und Michael Gfeller am 04.05.2012 vorgenommen.

###### 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 |

###### Exception Handling

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Der Code enthält keine abgefangenen und ignorierten Ausnahmen | X |  |
| Fehler in asynchronen Prozessen werden mittels Event weitergeleiltet | 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 |

###### 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 |  |

###### Naming

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Namen der Klassen / Variabeln 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 |

###### Tools

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben | ? |  |
| Die Errors von FxCop werden wo sinnvoll behoben | ? |  |

#### 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). | - | - |
| 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 mihilfe 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 gelaufen 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 14 - Annotationen und Kommentare Code Review 05.06.2012

##### Bewertung der Kriterien

Die roten Kommentare wurden von Michael Gfeller eingefügt oder gemäss den Mails zwischen Michael Gfeller und Lukas Elmer vom 12. Juli (TODO: Anhang, ist schon im Anhang Ordner) ergänzt.

###### 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 Problem nur im DiagnosticApp, das nicht zum Code Review gehört | X |  |
| Alle Public / Protected Members sind ausreichend Dokumentiert  Problem nur im DiagnosticApp, das nicht zum Code Review gehört | 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 Diverse Kleine sind noch vorhanden. | X |  |

###### Exception Handling

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Der Code enthält keine abgefangenen und ignorierten Ausnahmen | X |  |
| Fehler in asynchronen Prozessen werden mittels Event weitergeleiltet | 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  Wie besprochen; Noch nicht komplett implementiert |  | X |

###### 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 |  |

###### Naming

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Namen der Klassen / Variabeln 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 |  |

###### Tools

|  |  |  |
| --- | --- | --- |
|  | Erfüllt | N. Erfüllt |
| Die Warnings von Resharper 4.XXX werden, wo sinnvoll, behoben  Die „disabled Warnings“ vom Reshaper 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 |  |

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