## 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 |
| 07.06.2012 | 1.20 | Installationsdokumentation | CH |
| 07.06.2012 | 1.21 | Unit Tests | 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, die Hypothese „Meine Hand ist die Maus“ zu bestätigen. Um dies zu prüfen, wurde mithilfe einer WPF Applikation ein Wizard of Oz Experiment durchgeführt. Die Testpersonen wurden gebeten, laut mitzudenken.



Abbildung - Testdurchführung Wizard of Oz mit einem Probanden

Der Test wurde mit sieben Personen durchgeführt, welche das Testszenario (siehe I.2.1.2.1 Testszenario) durchspielten. Alle Testpersonen konnten die Aufgabe ohne grosse Probleme lösen. Die während des Tests gemachten Notizen befinden sich im Anhang (TODO).

Nachfolgend ist beschrieben, was die Applikation, welche beim Test eingesetzt wurde, kann und wie sie gesteuert wird.



Abbildung - Testapplikation

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

Die blauen Punkte in der Abbildung 6 - Testapplikation dienen der Beschriftung der einzelnen Komponenten in der Poster-Ansicht:

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

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

##### 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 Wand 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

Die Beobachtungen und Notizen, welche während der Durchführung des Tests gemacht wurden, sind in der nachfolgenden Tabelle zusammengefasst:

|  |  |  |  |
| --- | --- | --- | --- |
| 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 Bewegung der Hand nach vorne oder durch das machen einer Faust betätigen.
* Vier Testpersonen hätten gerne das Poster mittels Zoomgeste 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 Wischgeste zum nächsten Poster übergehen.

Weiter merkten die Testpersonen an, dass:

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

Das Fazit des Tests ist, dass die Hypothese „Meine Hand ist die Maus“ bestätigt werden konnte. Bei diesem Test wurde die Applikation nicht mit mehr als sieben Personen geprüft. Dies, da die Resultate so deutlich waren, dass eine statistische Analyse nicht für nötig befunden wurde. Aufgrund dieses Resultats wird die Applikation so weiterentwickelt, dass die Videowall nicht mit Gesten sondern nur mit der Hand gesteuert wird. Zusätzlich konnte auch das GUI verifiziert werden. Für die Testpersonen war sehr schnell klar, für was die Pfeile und das Menu verwendet werden können.

#### 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, wurden 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 Mittagsmenü 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 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

#### LunchMenuApp

Für die LunchMenuApp konnte mit 16 Unit Tests eine Testabdeckung von 100% erreicht werden.

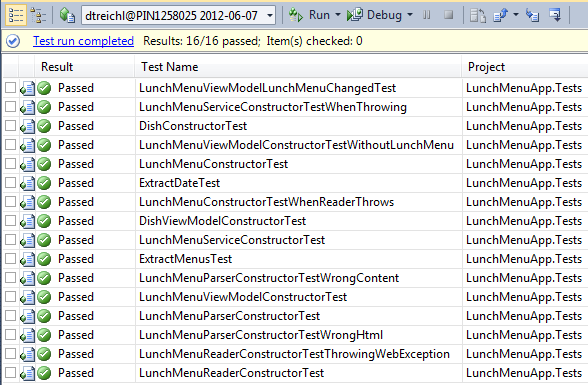


Abbildung - Unit Tests LunchMenuApp

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.

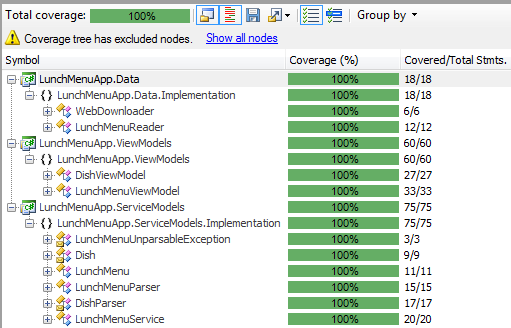


Abbildung - Test Coverage LunchMenuApp

#### PosterApp

Für die PosterApp konnte mit 6 Unit Tests eine Testabdeckung von 100% erreicht werden.

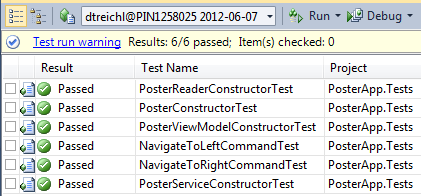


Abbildung - Unit Tests PosterApp

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.

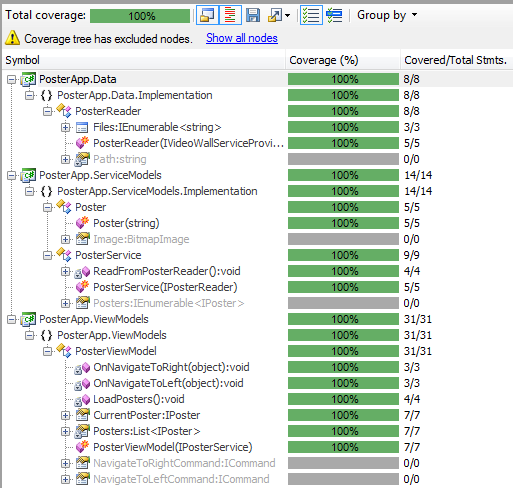


Abbildung - Test Coverage PosterApp

#### VideoWall

Für die Videowall konnte mit 21 Unit Tests eine Testabdeckung von 42% erreicht werden.

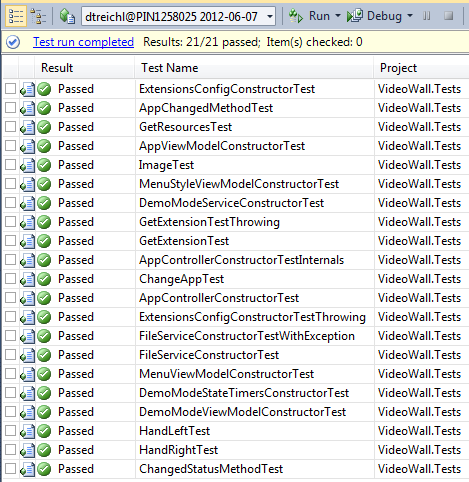


Abbildung - Unit Tests VideoWall

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. 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 steht. 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, welche die unterschiedlichen Extensions managen (Apps.Implementation) 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 erschwerend und mühsam 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.

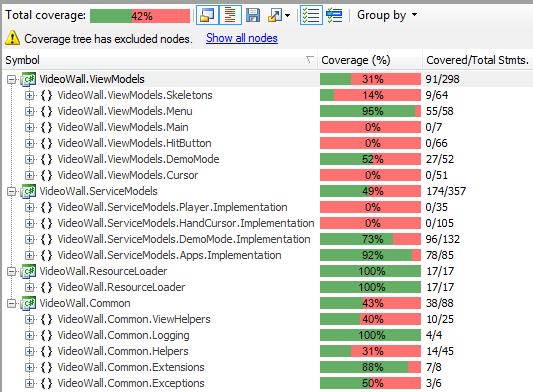


Abbildung - Test Coverage VideoWall

### 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 | Mittagsmenü App in Plug-in umgewandelt | Die Mittagsmenü-Applikation besteht als Plug-in und kann durch Nr. 14 in der Hauptapplikation angezeigt werden. | Ok | 22.05.2012 |
| 856 | Das Mittagsmenü wird angezeigt. | Das Mittagsmenü 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. 14 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 | Mittagsmenü App automatisch aktualisiert | Das Mittagsmenü 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. |  | 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 | Mittagsmenü App in Plug-in umgewandelt | Die Mittagsmenü-Applikation besteht als Plug-in und kann durch Nr. 14 in der Hauptapplikation angezeigt werden. | Ok | 29.05.2012 |
| 856 | Das Mittagsmenü wird angezeigt. | Das Mittagsmenü 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. 14 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 | Mittagsmenü App automatisch aktualisiert | Das Mittagsmenü 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 |

### Code Dokumentation

#### Coding Standards

Die Codings 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:

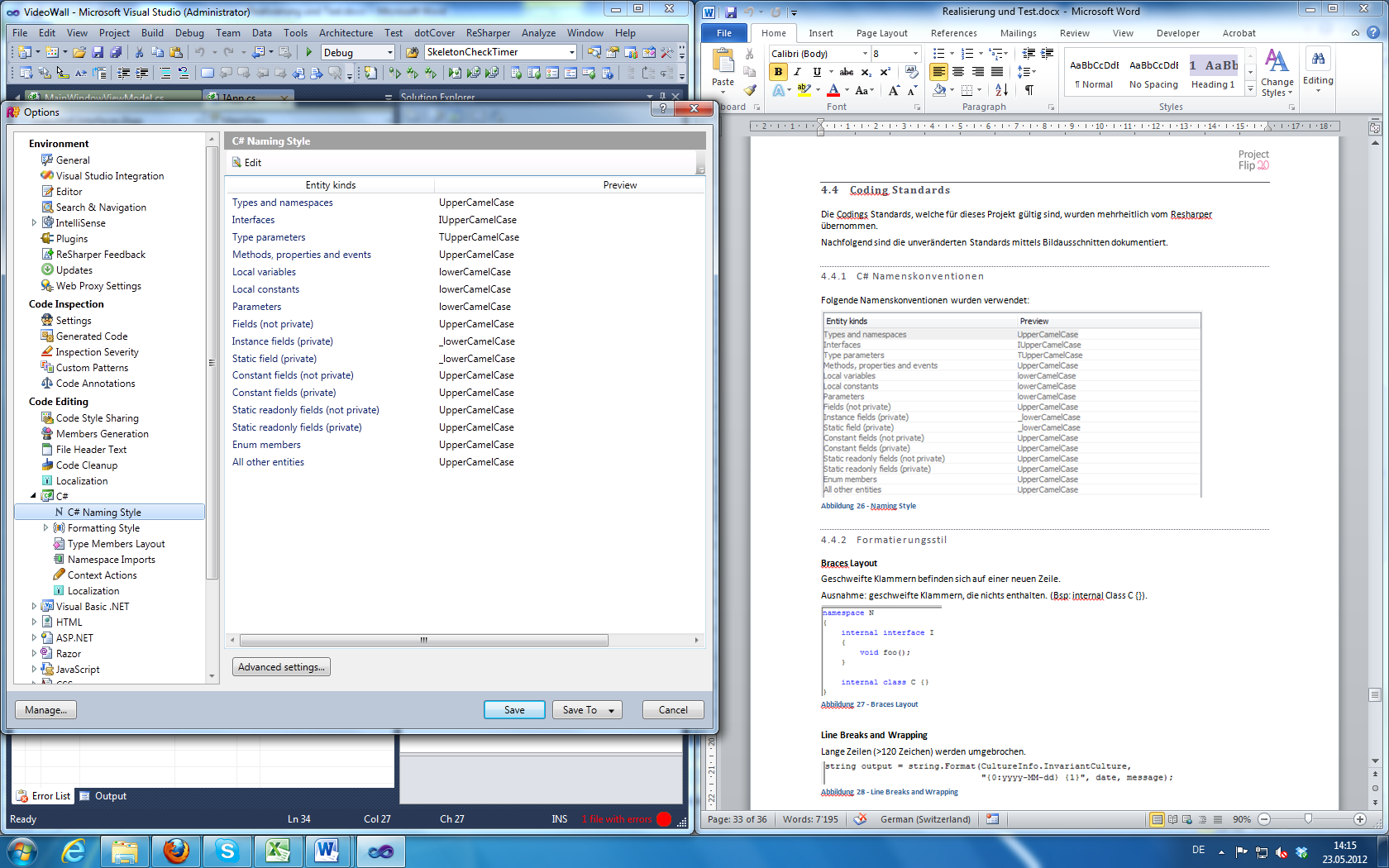


Abbildung - Naming Style

##### Formatierungsstil

**Braces Layout**

Geschweifte Klammern befinden sich auf einer neuen Zeile.

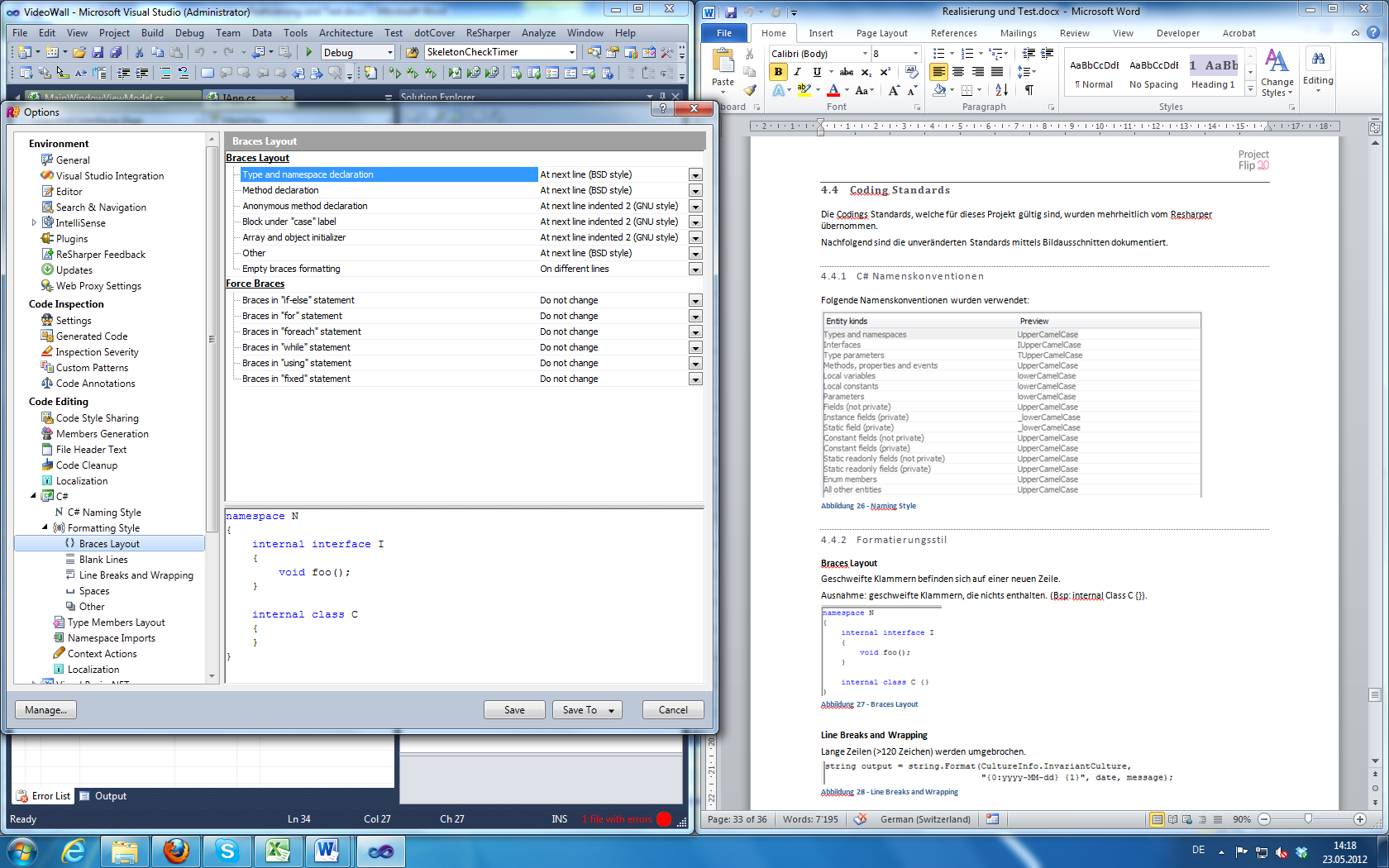


Abbildung 13 – Formatierungsstil, Braces Layout

**Line Breaks and Wrapping**

Lange Zeilen (>120 Zeichen) werden umgebrochen.

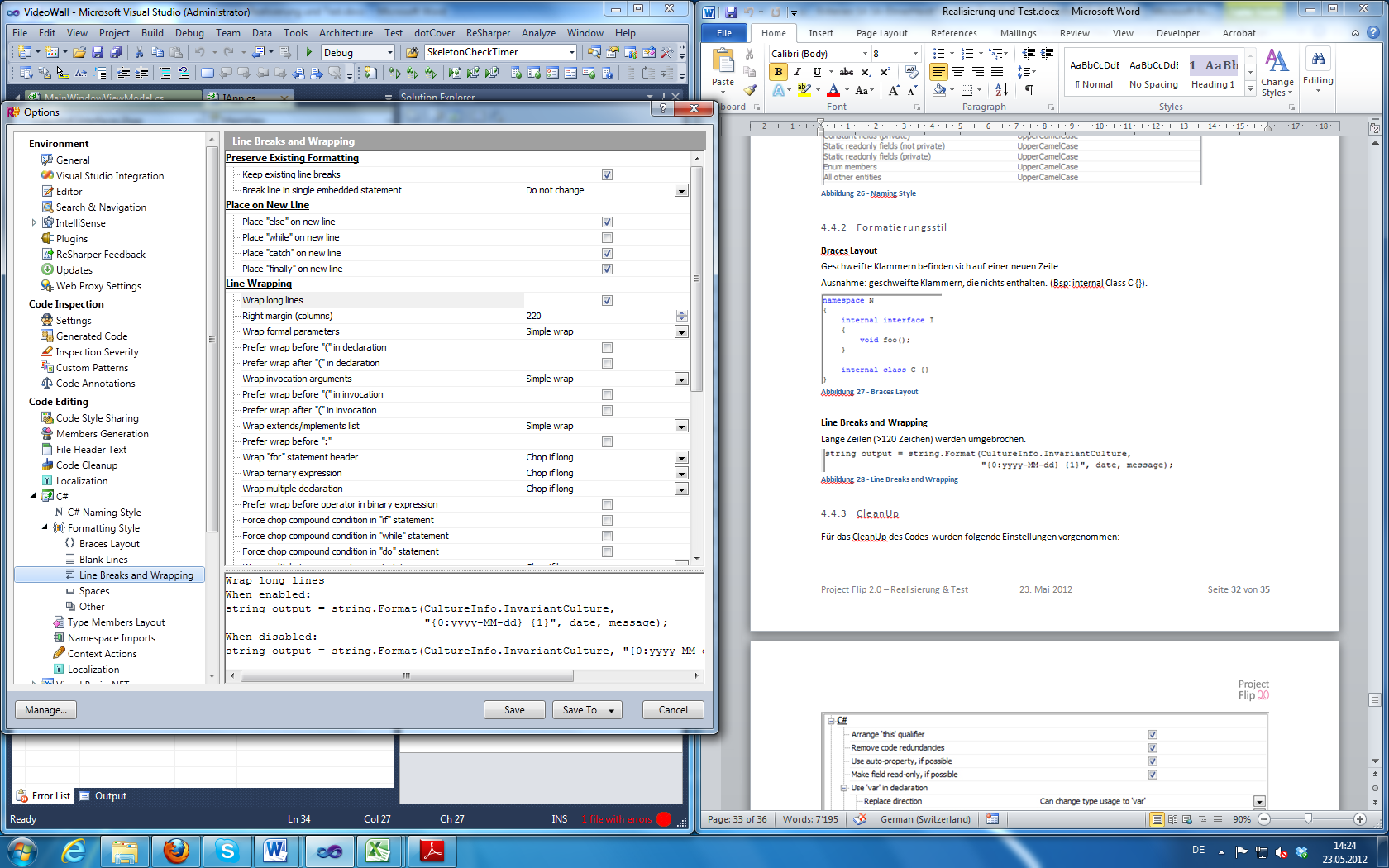


Abbildung - Formatierungsstil, Line Breaks and Wrapping

##### Auswertung durch Resharper

Durch den Resharper können Coding Issues angezeigt werden. Diese sehen wie folgt aus.

TODO: Rechtsklick auf Solution -> code Issues -> Screenshot Inspection Results

Wie ersichtlich ist, werden alle Coding Standards eingehalten.

##### CleanUp

Für das CleanUp des Codes wurden folgende Einstellungen vorgenommen:

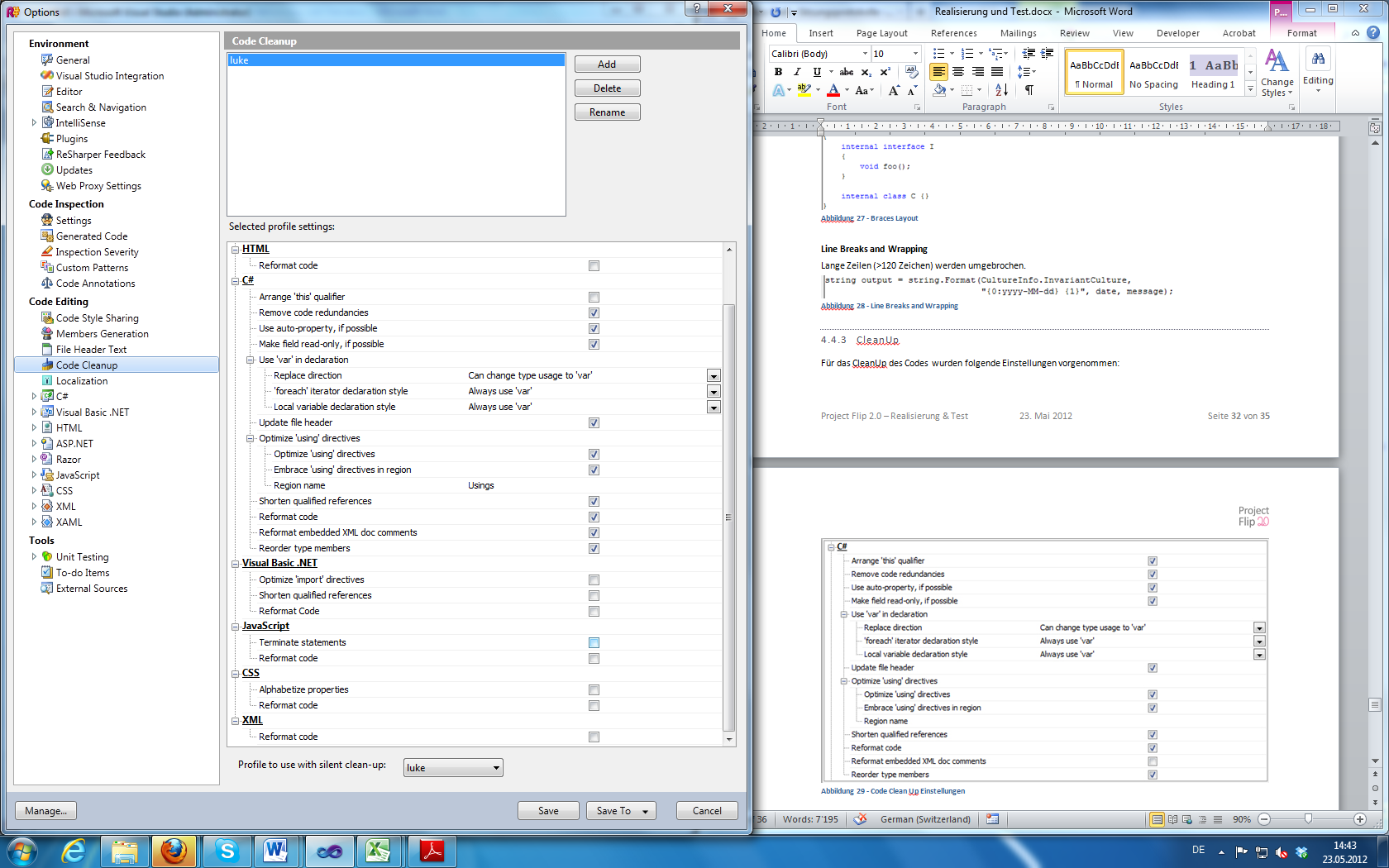


Abbildung - CleanUp Einstellungen

#### 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.

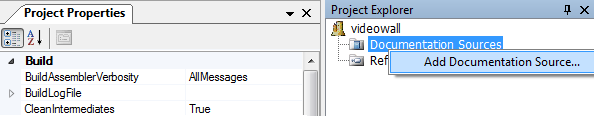


Abbildung - Sandcastle Help File Builder, Hinzufügen der Visual Studio Solution

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

Um die Applikation zu verwenden wird folgendes benötigt:

* Kompilierte Version des Projektes
* .NET, Kinect Runtime
* Internetverbindung

Die kompilierten Dateien sollen als erstes in den gewünschten Zielordner kopiert werden. Der Applikation können verschiedene Plug-ins hinzugefügt werden. Wie dessen Ordnerstruktur aufgebaut sein soll, wird im Kapitel I.1.5.3 Plugins beschrieben.

#### Dokumentation für Entwickler

Um die Applikation weiterentwickeln zu können, muss ein SVN Checkout auf dem gesamten SVP Repository durchgeführt werden. Dazu wird ein SVN Tool wie TortoiseSVN benötigt (siehe Tools TODO).

Zudem werden folgende Werkzeuge benötigt:

* Visual Studio 10
* Expression Blend
* Resharper
* Kinect for Windows SDK
* NuGet

Die Applikation kann konfiguriert werden, dies geschieht über die app.config Datei. Wie die Konfiguration angepasst werden kann wird im Kapitel I.1.5.4 Konfigurationsdatei erläutert. Zudem können der Applikation Plug-ins hinzugefügt werden, dies wird im nachfolgenden Kapitel aufgezeigt.

#### Plugins

Um den Pfad zum Plug-ins Ordner konfigurieren zu können, besteht das app.config File. Wie der Pfad angepasst werden kann, wird im Kapitel I.1.5.4.6 Plug-ins Pfad erläutert.

Für jede Plug-in Applikation muss ein Unterordner im Plug-ins Ordner (siehe I.1.5.4.6 Plug-ins Pfad) erstellt werden. Befindet sich in diesem Ordner kein Plug-in, so wird beim Start der Videowall Applikation eine entsprechende Fehlermeldung angezeigt. Bestimmte Applikationen benötigen zusätzliche Dateien. Bei diesen kann es sich beispielsweise um Bilder handeln. Diese müssen in einer festen Ordnerstruktur abgelegt werden und immer den gleichen Namen tragen. Konkret handelt es sich hierbei um einen Unterordner mit dem namen „Files“ im Ordner des Plug-ins. Fehlt dieser Ordner, so wird eine entsprechende Fehlermeldung beim Start der Hauptapplikation angezeigt.

#### Konfigurationsdatei

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

##### Konfigurationssektionen

Durch den Abschnitt configSections können verschiedene Konfigurationssektionen definiert werden.

  <configSections>

    <section name="unity" type="Microsoft.Practices.Unity.Configuration.UnityConfigurationSection, Microsoft.Practices.Unity.Configuration"/>

  </configSections>

##### Unity

In der Konfigurationssektion von Unity wird zum einen angegeben welche Namespaces verwendet werden und in wechlem 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"/>

Zum anderen können 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"/>

##### 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* angeboten. Hierbei werden mit Kinect aufgenommene Skelettdaten abgespielt. Im Falle von *AutoPlayFileSleketonReader* wird das File immer wieder wiederholt. Die abzuspielenden Skelettdaten können über ein Replayfile definiert werden, dieses wird unter I.1.5.3.9 KinectReplayFile erklärt.

        <!-- FileSkeletonReader -->

        <!-- AutoPlayFileSkeletonReader -->

        <!-- KinectSkeletonReader -->

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

          <lifetime type="singleton"/>

        </type>

##### Cursor

Ist Kinect angeschlossen so ist für das Mapping des ICursorViewModel KinectCursorViewModel zu verwenden. Dadurch wird der Cursor als kleine Hand dargestellt und kann mit den Händen gesteuert werden.

Wird das Schlüsselwort MouseCursorViewModel verwendet so lässt sich die Applikation mit der Maus bedienen und es wird keine Hand angezeigt.

        <!-- KinectCursorViewModel -->

        <!-- MouseCursorViewModel -->

        <type type="ICursorViewModel" mapTo="MouseCursorViewModel">

          <lifetime type="singleton"/>

        </type>

##### Padding

Damit die Applikation angenehm mit Kinect zu bedienen ist, wurde einer Interaktionszone definiert. Das Prinzip dieser Zone wird im Kapitel (TODO. Entwurf, Handtracking) genauer erläutert. Die rote Fläche in der Abbildung Abbildung 11 - Interaktionszone ist auf den Arm ausgerichtet und stellt die eigentliche Interaktionszone dar. Die schwarz umrahmte Fläche stellt den Erkennungsbereich von Kinect dar. Um die rote Zone zu definieren, wird das *RelativePadding* verwendet. Möchte man die Zone verändern so kann dies durch die aufgelisteten Schlüsselwörter *left*, *top*, *right* und *bottom* getan werden. Welchen Bereich diese Schlüsselwörter verändern ist in der Abbildung 11 - Interaktionszone ersichtlich.

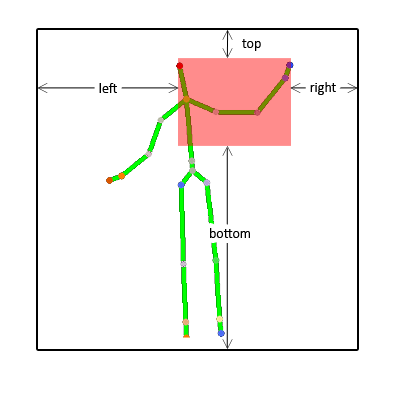


Abbildung 17 - Interaktionszone

        <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>

##### Plug-ins Pfad

Die Applikation kann durch Plug-ins dynamisch erweitert werden. Diese müssen in einem bestimmten Ordner abgelegt werden. Dieser kann über das *ExtensionsConfig* gesetzt werden.

        <type type="ExtensionsConfig" mapTo="ExtensionsConfig">

          <constructor>

            <param name="extensionsDirectoryPath" value="../../../Extensions"/>

          </constructor>

        </type>

Der Parameter *extensionsDirectoryPath* muss auf den gewünschten Pfad gesetzt werden.

##### Demomodus

Der Demomodus wird aktiv, wenn Kinect keinen Nutzer erkennt. Dieser kann im Bereich *IDemoModeConfig* angepasst 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>

Durch den Parameter *backgroundColors* können Farben für den Hintergrund des Demomodus angegeben werden. Des Weiteren können verschiedene Timer angepasst werden. *skeletonCheckTimeSpan* beschreibt den Intervall, in welchem geprüft wird, ob gerade ein Nutzer erkannt wurde. Ist der Demomodus aktiv so wechselt der Teaster-Text und der Hintergrund in einem gewissen Intervall. Dieser kann durch *changeAppTimeSpan* angepasst werden. Ist die Applikation aktiv, erkennt aber kein Skelett, wechselt sie nach Ablauf von *fromActiveToDemoModeTimeSpan* zum Demomodus. Wird im Demomodus ein Nutzer erkannt so wird diesem dies durch ein Countdownzähler visualisiert, dieser kann durch *countdownTimeSpan* verändert werden. Nachdem ein Nutzer erkannt wird, wird der letzte Timer *skeletonTrackingTimeoutTimeSpan* dazu gebraucht, um intervallsweise zu prüfen ob der Nutzer immer noch erkannt wird.

##### Singelton

Von bestimmten Klassen soll es immer nur eine Instanz geben, welche an mehrerer Orten verwendet werden kann. Dies kann durch Angabe der Klasse und dem Schlüsselwort *lifetime* bezweckt werden.

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

          <lifetime type="singleton"/>

        </type>

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

          <lifetime type="singleton"/>

        </type>

##### KinectReplayFile

Wird bei I.1.5.3.3 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 angegeben werden:

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

          <lifetime type="singleton"/>

          <constructor>

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

          </constructor>

        </type>

Durch den Parameter path kann der Pfad zu dem Replayfile angepasst werden und ist relativ anzugeben.

##### Runtime Version

Durch das Schlüsselwort startup kann die unterstütze Runtime des .NET Frameworks angegeben werden

  <startup>

    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>

  </startup>

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