Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht

Daniel Löffelholz

Masterprojekt - SoSe 2009

Fachbereich Elektrotechnik und Informatik
Department of Electrical Engineering and Computer Science

Inhaltsverzeichnis

1. Einleitung

Diese Ausarbeitung stellt einen Bericht über ein Studentenprojekt, welches im Rahmen des Masterstudiengangs für angewandte Informatik 08/09 an der HAW Hamburg statt fand, dar. In diesem Projekt wurde untersucht und erprobt wie Software für interaktive Tische gestaltet und entwickelt werden kann um produktiv in frühen Projektphasen eingesetzt werden zu können. Das Projektteam bestand aus Hauke Wittern, Thorben Pergande, und dem Autor dieses Berichts, Daniel Löffelholz.

1.1. Motivation

Das Projekt, in Abgrenzung zu den begleitenden rein theoretischen Ausarbeitungen, soll dazu dienen abseits der theoretischen Erkenntnisse auch praktische Erfahrungen im Bereich der interaktiven Tische sammeln zu können, sowie erworbenes Theoriewissen praktisch zu erproben. Das Ergebnis des Projekts soll dabei nicht nur in Form von schriftlichen Ausarbeitungen festgehalten werden, sondern auch eine Anwendung zum Ergebnis haben in denen die gemachten Erfahrungen umgesetzt worden sind. Aufgrund der kleinen Teamgröße wurde über die komplette Projektdauer eng zusammengearbeitet und fast alle anfallenden Herausforderungen gemeinsame diskutiert und erarbeitet. Eine direkte grundsätzliche thematische Abgrenzung ist somit nicht möglich, jedoch lässt sich die Arbeit am Projekt durchaus ein einzelne Aufgaben zerlegen die weitestgehend separat bearbeitet wurden.

1.2. Überblick

Kapitel 2 berichtet über die Projekteigenschaften und Projektziele, sprich dessen Anforderungen, das Szenario und die Vision. Das vorletzte Kapitel 3 geht auf den insbesondere vom Autor bearbeiteten Teil des Software und die fachlich dazugehörige Thematik ein. Gemachte Erfahrungen, eine resümierende Zusammenfassung sowie ein Ausblick auf die kommende Fortsetzung des Projekts sind Inhalt von Kapitel 4.

2. Projektziele

Dieses Kapitel schildert die Vision unter welcher das Projektteam das Szenario entwickelt und regelmäßig betrachtet hat. Dieses Szenario wird mit einigen fachlichen und technischen Anforderungen am Ende dieses Kapitels ergänzt.

2.1. Vision und Szenario

Interaktive Tische ermöglichen die Kombination von traditionellen tischbasierten Arbeitsweisen und digitaler Technologie, und verhindern Medienbrüche beim Wechsel zwischen traditionellen (z.B. Papier) und modernen Medien (digitale Dokumente). Aufgrund der bis dato geringen Verbreitung und Eignung für den Massenmarkt existieren jedoch bisher wenige Untersuchungen und Projekte über den Einsatz dieser Möglichkeiten in produktiven Umgebungen. Urspung der Vision war somit die Fragestellung wie diese Technologie produktiv in frühen Projektphasen eingesetzt werden kann und welche technologischen und fachlichen Herausforderungen bei der Entwicklung einer solchen Software entstehen. Dabei wurde ein pragmatischer Ansatz gewählt bei dem nach regelmässigem Ideensammeln neue Konzepte agil umgesetzt, anschließend bewertet und gegebenenfalls überarbeitet oder verworfen wurden.

Die zu entwickelnde Software musste vielfältigen Randbedingungen genügen, welche in Kapitel 2.2 genauer erörtert werden. Jedoch sei vorweg genommen, dass sich das Team als Vision für einen kollaboratives UML-Änderungs- und Präsentationswerkzeug (genannt COSMOS) entschieden hat, da die fachliche Domäne jedem Projektmitglied gut bekannt war und das Reviewing von UML-Modellen ein stark kollaborativer Arbeitsvorgang und fester Bestandteil vieler Projekte ist. Bei dem Szenario geht es um die kollaborative Bearbeitung einer Aufgabe an einem einzelnen interaktiven Tisch. Ziel ist es dabei, dass durch die vermiedenen Medienbrüche und durch die flexible Darstellung die eine digitale Darstellung ermöglicht ein Produtivitätsgewinn erreicht wird. Der Sammelbegriff Produktivitätsgewinn bedeutet in diesem Kontext mehrere Vorteile die im besten Fall erreicht werden können. Durch die vermiedenen Medienbrüche soll Redundanz und veraltete Dokumente vermieden werden. Es muss zudem keine Zeit aufgewendet werden um digitale Daten für die traditionellen Medien vorzubereiten oder zu übertragen, wie formatieren oder ausdrucken, oder die Änderungen auf dem Papier in die digitalen Datenbestände zu übertragen. Die Anzeige von Daten auf digitalen Oberflächen ermöglicht zudem eine schnelle und flexible Anzeige der Daten. Damit kann schnell und effiziente durch große Datenbestände navigiert werden oder unwichtige Details ausgeblendet werden. Auch kann schnell zwischen verschiedenen Arten der Datenaufbereitung und Darstellung gewechselt werden, was für das Verständnis und den Umgang mit den Daten von Vorteil sein kann.

2.2. Anforderungen

Die Anforderungen an die Software und das umzusetzende Szenario 2.1 lassen sich auch hierbei in fachliche und technisch Anforderungen unterteilen. Die technischen Anforderungen resultieren aus der dem Projekt zur Verfügung gestellten interaktiven Tisch, dem Microsoft Surface dessen genaue Eigenschaften in Kapitel 3 ausführlicher aufgeführt sind. Im Laufe des Projekts wurde festgestellt, dass die Hardwareeigenschaften die Gestaltung und Möglichkeiten

der Oberfläche stärker als bei traditioneller Software beeinflussen und somit neue Anforderungen und Herausforderungen generieren. Dieser Tisch unterstützt eine Multitouch-Eingabe von bis zu 52 Eingabepunkten. Aufgrund der Größe des Geräts lasst sich daran mit 2-4 Personen komfortabel arbeiten. Softwareseitig arbeitet der Tisch mit einer Microsoft Windows Oberfläche und unterstützt eine .NET-API und somit die Programmierung in Visual VB und C-Sharp (siehe Corporation (2009)).

Da das Projektteam mit dem Forschungsgebiet der Oberflächengestaltung für interaktive Tische bisher wenig vertraut war, beschränkten sich die fachlichen Anforderungen zunächst auf die Unterstüzung von kollaborativer Arbeit und einer produktiven Nutzbarkeit in einer realen Anwendungsdomäne. Zudem soll die entstehende Software „COSMOS“ möglichst erweiterbar, sowie ein Framework extrahierbar sein, welches für andere Projekte ähnlicher Art wiederverwendbar ist. Falls man die in 2.1 genannten und erstrebten Vorteile einer Kombination von traditioneller tischbasierter Arbeit und digitaler Technologie nutzen will muss man jedoch noch weitere Anforderungen beachten: der nahtlose Austausch von Daten zwischen Einzelarbeitsplatz und interaktiven Tisch muss gewährleistet sein. Die Daten müssen am Tisch modifiziert und annotiert werden können. Die Darstellung und Aufbereitung muss je nach Anwendungsdomäne vielfältig möglich sein. Unterschiedliche Detailstufen und effiziente Navigation durch große Datensätze oder Diagramme ist auch wünschenswert. Die Software muss die Vorteile und Eigenschaften der traditionellen Tischarbeit sowie die Arbeitsgewohnheiten dessen Nutzer beachten und nutzen. Zudem muss der vorhandene Platz optimal genutzt werden. Aufgrund der touchbasierten Bedienung müssen die Bedienelemente eine gewissen Mindestgröße aufweisen. Zudem müssen die Bedienelemente von allen Seiten bedient werden können oder jeder Teilnehmer braucht eigene für ihn ausgerichtete Elemente.

In dem Projekt wurde versucht möglichst viele dieser Anforderungen umzusetzen um der Vision einer in diesem Umfeld produktiv einsetzbarer Software gerecht zu werden. Im folgenden Kapitel werden einige Thematiken mit denen sich der Autor befasst hat vorgestellt und über deren Umsetzung berichtet.

3. Grundlagen und Ausführung

Wie in der Einleitung (1) erwähnt lassen sich die bearbeiteten Aufgaben thematisch nicht komplett abgrenzen da insbesondere fachliche Aspekte in der Regel gemeinsam entwickelt wurden. Folgend sein jedoch die Thematiken aufgeführt mit denen sich der Autor in erster Linie beschäftigt hat. Zu jeder Thematik wird vorhergehend kurz die benötigten Grundlagen aufgeführt die vor der Umsetzung des jeweiligen Themas erarbeitet wurden. Dabei versucht jede Lösung einer bestimmten Anforderung (siehe 2.2) umzusetzen um letztlich dem skizzierten Szenario gerecht zu werden.

3.1. Graphen zeichnen

UML-Diagramme und deren Kanten und Knoten sind eine Form von Graphen. Mit dem Zeichnen von Graphen, sprich dem Anordnen der Knoten sowie deren Verbindungen beschäftigt sich das Themengebiet „Graphlayout“ oder „Graph drawing“ der Informatik. Dabei werden Algorithmen gesucht, mit deren HIlfe man unter dem einhalten bestimmter Randbedigungen, Vorgaben, oder Priorisierungen die Kanten und Knoten automatisch anordnen kann. Dabei unterscheidet man zwischen statischem und dynamischen Graphzeichnen. Bei Letzterem werden Sequenzen von Graphzeichnen visualisiert, während beim statischen Graphzeichnen der Ganze Graph angezeigt werden soll, was beim UML-Modellierungswerkzeug der Fall sein soll. Dabei gibt es keinen universellen Algorithmus wie der Graph optimal gezeichnet wird. Je nach Anwendungsfall unterscheiden sich die Anforderungen stark. Zum einen ob die Elemente eine flexible Form unterstützen, oder ob die Linien nur rein orthogonal verlaufen dürfen, oder ob sie auch geschwungen möglich sind. Für Organigramme ist eine hierarichische Anordnung sinnvoll, wobei es für die Elektrotechnik über kurze Verbindungsrecken und wenig Überschneidungen geht. Weitere Anforderungen könnten Anzah der Knicke, Abstand benachbarter Knoten oder die Symmetrie sein. Im Fall der UML-Diagramme geht es in der Regel um Übersicht. Mit Hilfe einer passenden Graphdarstellung versuchen wir die Anforderungen an die Übersicht umzusetzen (vgl. 2.2). Viele Knicke oder häufige Überschneidungen mindern die Übersichtlichkeit. Somit entschieden wir uns für einen Algorithmus der im optimalen Fall mit wenigen Knicken auskommt.

Im vorliegenden UML-Tool wird zur Verbindung zweier Elemente der Editiermodus eines Elements aktiviert, und mit dem Finger eine imaginäre Linie von einem Andockpunkt zum zweiten Element gezogen. Da der zweite Andockpunkt nicht vorgegeben ist, wird aufgrund des Verhältnisses wie die beiden Elemente zueinander platziert sind und an welcher Seite des ersten Elements die ausgehende Kante sein soll, ein Andockpunkt ausgewählt. Falls sich das zweite Element beispielsweise mit vollem Umfang oberhalb und rechts vom ersten Elements befindet, sind immernoch zwei der vier Andockpunkte sinnvoll wählbar: Falls der Andockpunkt auf der Unterseite des Elements eins gewählt wurde, wird dieser auch an der Unterseite des Elements zwei gewählt. Falls das Element eins jedoch oben die ausgehende Kante hat, ist eine Andockung auf linker Seite am sinnvollsten (siehe 1). Insgesamt existieren 8 mögliche Platzierungsverhältnisse, die aufgrund des ersten Andockpunkts nochmals 2-3 mögliche zweite Andockpunkt haben. Insgesamt also 24 Möglichkeiten für den zweiten Punkt.

Der zweite Teil des Layout-Alorithmusses berechnet den tatsächlichen Verlauf der Kanten. Oberste Priorität war hier wie erwähnt eine geringe Knickanzahl. Aus diesem Grund verläuft jede Kante zunächst abhängig von den Andockpunkten bis auf Höhe des Zielobjekts zunächst horizonal oder vertikal. Falls auf dem Web zum nächsten geplanten Punkt ein weiteres Element liegt wird eine Ausweichmethode aufgerufen um die Linie um das Objekt zu ziehen. Von den Elementen sind dem Algorithmus die X/Y-Koordinaten, die Breite, Höhe, und der Dreh-

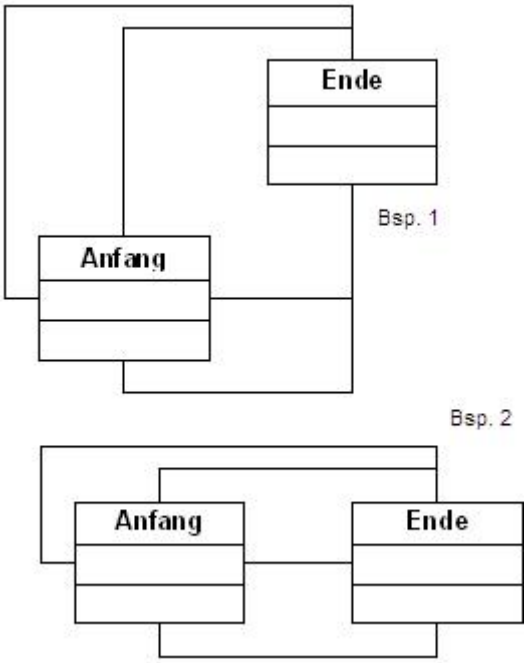

Abbildung 1: Beispiel Verbindungen

winkel bekannt. Mit Hilfe von Winkelfunktionen wird ein um ein gedreht liegendes Objekt eine Hilfsrechteckt berechnet dessen Seiten parallel zu den Tischkanten verläuft (siehe 2). Dieses Rechteck wird nun als Basis für das Ausweichen genommen. Falls beim Umzeichnen eine weitere Kollision mit anderen Elementen festgestellt wird, wird rekursiv die Ausweichmethode wieder aufgerufen. Es sind Ausweichmethoden für jeweils für horizontale oder vertikale Verbindungen vorhanden. Durch die Rekursion können auf jeder Verbindung theoretisch unbeschränkt viele Elemente mit der Linie kollidieren. Ein momentanes Manko ist jedoch, dass falls durch Verschieben der Elemente keine der oben genannten 24 optimalen Verbindungsmöglichkeiten entsteht. Sobald die Andockung jedoch nach Verschieben der Elemente aktualisiert und gegebenenfalls verändert wird, findet der Algorithmus immer eine passende Verbindung.

3.2. Menüelemente auf interaktiven multitouch Tischen

Wie in den Anforderungen bereits angedeutet, erfordert der Entwurf von Bedienelementen bei interaktiven Tischen mit gestenbasierter Bedienung besonderes Augenmerk (vgl. auch Microsoft Corporation (2009)). Mit einer speziell für interaktive Tische gestalteten Oberfläche widmeten wir uns somit den Punkten Intuitivität und effiziente Bedienbarkeit der Anforderungen

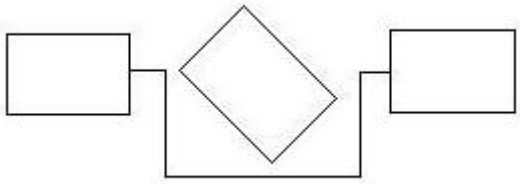

Abbildung 2: Beispiel: Ausweichen

(vgl. 2.2). Buttons sollten eine Mindestgröße von etwa 2x2 cm aufweisen damit sie ohne Probleme bedient werden können. Selbst bei einem noch geringen Funktionsumfang würden die Bedienelemente bei permanenter Anzeige einen Großteil des Platzes verbrauchen ohne möglicherweise regelmäßig benötigt zu werden. Morris u. a. (2006) zeigten, dass replizierte sich am Rand befindende Bedienelemente besser geeignet sind als geteilte zentrale Bedienelemente. Diese haben insbesondere die Nachteile, dass die Gestaltung und Lesbarkeit der Beschriftung nicht für alle Teilnehmer optimal ist, zudem in der Regel der Raum in der Mitte als Platz für Daten gedacht ist, und die Personen zufälligen Körperkontakt oft als störend empfinden. (vgl. Morris u. a. (2006))

Fachlicher Entwurf

Für den UML-Editor führten wir zwei unterschiedliche „Toolboxen“ ein. Eine für Input/Outputoperationen wie Speichern und Laden, sowie ein Kontextsensitives für Funktionen die sich speziell auf das Diagramm und die Aktionen beziehen, wie „neue Klasse“ oder „Undo“. Dabei führten wir mit Hilfe des eingebauten Tag-Systems greifbare Bedienelemente ein, welche die jeweilige Menüart auf dem Display erscheinen lassen. Durch die Bewegbarkeit der Elemente erhielten wir somit eine völlig frei wählbare Platzierung sowie eine schnelle Aktivierung oder Deaktivierung der Menüs. Durch eine artequarte Gestaltung der realen Objekte kann der Benutzer schnell erkennen um welchen Menütyp es sich handelt. Durch die Verwendung weiterer markierter Objekte sind beliebig viele Menüs auf dem Display platzierbar. Der jeweilige Winkel des des Tags wird von der API bereitgestellt. Somit konnten die Menüs zudem drehbar gemacht werden was die Platzierung weiter vereinfacht. Die technische Realisierung des Tag-Anlegens sowie Funktionsverknüpfung übernahm hier Thorben Pergande. Mit dem Konzept sowie der optischen Gestaltung beschäftigte sich der Autor.

Im Laufe des Projekts wurden mehrere Bedienelemente identifiziert die nicht den Anforderungen von interaktiven Tischen genügten oder bisher nicht vorhanden gewesen und neu entworfen werden mussten. Dazu zählten eine "Toolbox"für kontextbezogene Interaktion mit dem

UML-Diagram mit den Funktionen: Zoom, „Undo“, „neue Klasse hinzufügen“, und „Ausrichtung“ (siehe Abbildung 3). Zusätzlich ist hier ein Untermenü für weitere Operationen gestaltet, jedoch bisher ohne Funktion. Zudem eine zweite Toolbox die für allgemeinen Dateioperationen: „Speichern“, „Laden“, „Drucken“. Die runden Buttons sind halbkreisförmig um das Objekt platziert. Diese Platzierung benötigt die geringste Fläche auf dem Display und die runde Form verdeutlicht den Knopfcharacter. Durch einen 3D-Effekt durch Schattenwurf sind sie deutlich von der 2-dimensionalen Diagrammdarstellung zu unterscheiden. Ein weitere Herausforderung bei Touchbedienung ist die fehlende Haptik. Da in vielen Projektumgebungen eine audio-basierte Rückmeldung eines Knopfes nicht wünscheswert wäre, muss diese per grafischer Veränderung des Eingabeelements passieren. Die Buttons der Werkzeugleisten des UML-Editors verdunkeln sich bei Aktivierung, um ein hineindrücken zu simulieren. Zudem erhellt sich der Rand um die Aktivierung zu unterstreichen. Diese Animationen wurden mit Hilfe von der Storyboardtechnologie des WPF-Frameworks entworfen. Weiterhin ist es möglich Untermenüs anzulegen, Diese erscheinen bei Aktivierung des vorangehenden Buttons halbkreis-förmig unterhalb. Auf den Buttons befinden sich für die jeweilige Funktion passende Icons, wie die „Diskette“ zum Speichern, die Lupe für den Zoom, oder einen nach links zeigenden gekrümmten Pfeil für „Undo“.

Abbildung 3: Beispiel: Menü

Zudem wurde für den Papierkorb ein passendes grafisches Element entworfen (siehe Abbildung 4). Wichtig ist hier wieder die optische Rückmeldung, sowie ein intuitives Navigieren durch die bereits abgelegten Elemente. Der Papierkorb hat dabei die Form einer Papierablage mit leicht geschwungenen Formen. Für die Rückmeldung ist auf die Unterseite des Elements ein leuchtenartiger Punkt eingelassen, welcher nach dem Ausführen einer Aktion durch aufleuchten eine optische Rückmeldung abgibt. Die Navigation durch die abgelegten Elemente passiert mittels einer wischenden Fingergeste, wie sie von anderen Touchgeräten bekannt ist.

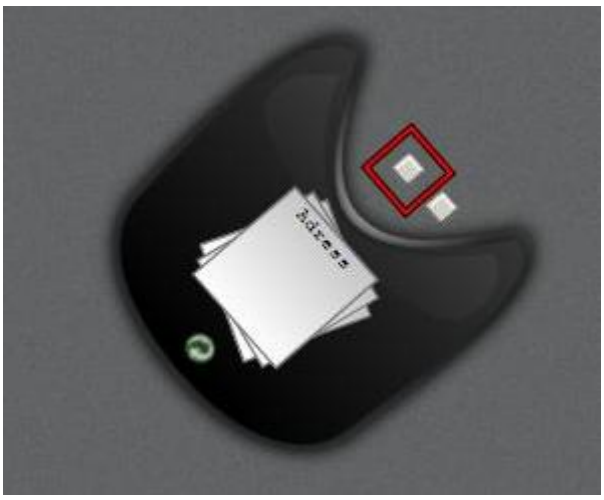

Abbildung 4: Papierkorb

Daraufhin wird eine Animation ausgelöst, die das Herunternehmen und unten Anfügen an den Stapel optisch simuliert. Dazu wird das auf oberster Ebene liegende Element trapezförmig verformt und die Breite asymtotisch auf 0 gesenkt. Anschliessend vergrößert eine ähnliche Animation das Element wieder und der neue Inhalt wird eingeblendet.

Im folgenden Kapitel wird erläutert wie diese Konzepte umgesetzt wurden um neue C-Sharp-konforme Bedienelemente und die zugehörigen Animationen zu gestalten.

Microsoft Blend als Entwurfswerkzeug von Animationen und komplexer grafischer Oberflächen

Microsoft Expression Blend 2 ist die Verknüpfung von reinen Grafikanwendungen und der Oberflächenprogrammierung. Es soll die professionelle Enwicklung von Oberflächen für Webanwendungen vereinfachen. Es unterstützt dabei die plattformübergreifenden Technologien WPF und Silverlight. Es kann somit auch zur Entwicklung von WPF-basierten Oberflächen für Desktopanwendungen, oder auch Anwendungen für den Microsoft Surface verwendet werden.

Die ersten Teile der Oberfläche des im Projekt entwickelten UML-Tools und die ersten Animationen wurden manuell mit WPF und XAML entwickelt. Die Entwicklung von eigenen Bedienelementen und Animationen gestaltete sich jedoch sehr aufwändig und war nicht immer zufriedenstellend. Aufgrund dessen wurde nach Alternativen zur Oberflächengestaltung gesucht und letztlich Microsoft Expression Blend ausgewählt.

Das Entwerfen der verktorbasierten Grafiken erfolgt hier per WYSIWYG-Editor oder integriertem Codeeditor. Die Verknüpfung mit den in Visual Studio gespeicherten Projekten funktioniert gut. Animationen werden hier per Storyboards bzw Zeitleisten angelegt und ermöglichen mittels Aufzeichnung der Veränderung eine Animation anzulegen.

In Abschnitt C des Anhangs ist eine Abbildung der Oberfläche des Programms mit geöffnetem Eingabebuttonprojekt angefügt. Aus ihr wird ersichtlich, dass das Anlegen der Grafiken wie aus anderen Grafikprogrammen gewohnt mit Hilfe unterschiedlicher Layer funktioniert. Das Storyboard ist den aus Animationsprogrammen wie Adobe Flash CS4 ähnlich.

3.3. UML-Persistierung und Austausch

Der entwickelte UML-Editor hat nicht den Anspruch ein vollwertiges UML-Entwicklungswerkzeug zu sein. Dazu ist die Plattform interaktiver Tisch aus sicht der Projektteilnehmer nicht geeignet. Die Software soll viel mehr Diskussions- und Anpassmöglichkeiten bieten. Das Szenario sieht vor, dass der Benutzer sein am Desktopcomputer entwickeltes UML-Diagramm auf den Tisch übertragt. Wie in den Anforderungen (2.2) und von Scott u. a. (2003) aufgeführt, benötigt man aber einen möglichst nahtlosen Austausch zwischen Desktopcomputer und interativem Tisch (vgl. auch Rekimoto (2002)). Das beinhaltet dass zu einen die entworfenen Daten mit vielen UML Entwicklungswerkzeugen ausgetauscht werden können. Zum anderen dass auch das Layout entweder automatisch, oder in einem speziellen Format persistiert und ausgetauscht werden können. Als Standard für die Datenspeicherung bot sich hier der UML Metadata Interchange (XMI) an. Dieser ist ein Standard der Object Management Group (OMG) und wird zunehmend als Austauschformat zwischen Software-Entwicklungswerkzeugen verwendet. Diese werden im XML-Format auf Basis der Meta-Metamodellen nach der Meta-Object Facility (MOF) gespeichert. Es können beliebige UML-Diagramm oder weitere Metadaten ausgetauscht werden, sofern sie mit der MOF ausdrückbar sind. Ursprünglich war geplant, dass ab der Version 2.0 auch das Layout der UML-Diagramme speicherbar wird. Dies ist nun aber lediglich mit dem unbekannteren Diagramm Layout-Standard möglich, der sich bisher jedoch kaum durchgesetzt hat.

Zur Erstellung des XML-Dokuments wurde der XML-Persistierer von C-Sharp verwendet. Hierbei muss für jedes Element des Dokuments eine Klasse erstellt werden, in denen die Attribute mit den auszugebenen Tags annotiert sind. Im Anhang A ist beispielhaft die Umsetzung einer XML-Klasse aufgeführt. Aktuell unterstützt COSMOS die Persistierung sowie das Auslesen von XMI1.0 sowie den aktuellen Standard XMI 2.1. Da der Diagramm Layout-Standard kaum verbreitet ist wird ein proprietäres Format verwendet um die aktuelle Position und weitere Formateigenschaften wie Ausrichtung zu speichern. Hierzu wird ebenfalls der XML-Persistierer eingesetzt und ein zweites XML-Dokument erstellt.

4. Fazit und Ausblick

Das folgende Kapitel fasst die resultierenden Ergebnisse kurz zusammen, bewertet diese, und führt die gemachten Erfahrungen auf. Anschliessend wird ein Ausblick auf den aufbauenden kommenden zweiten Teil des Masterprojekts gegeben.

4.1. Fazit

Die Ergebnisse dieses Projekts lassen sich in zwei Kategorien unterteilen: Ergebnisse, die die Durchführung, die Planung, das Entwicklungsvorgehen und die Zusammenarbeit betreffen, sowie solche die rein das Fachliche betreffen. Für ein möglichst optimales Ergebnis in der Fortsetzung des Projekts müssen beide Arten von Ergebnissen mit einbezogen werden.

Durch die kleine Teamgröße war es möglich, alle wichtigen fachlichen Aspekte du diskutieren und regelmässig Ideen auszutauschen. Zudem konnten somit Leerlaufzeiten minimiert und die Arbeitszeit effizient genutzt werden. Von Nachteil ist, dass mit drei Personen natürlich nur beschränkt viel zu leisten war. Für die optimale Umsetzung wären in den Augen des Autors eine Teamgröße von 5 Personen optimal.

Problematisch war dass durch geringe fachliche, und in Bezug auf C-Sharp technologische, Vorkenntnisse bei den Teammitgliedern eine konkrete Projektplanung anfangs nicht wirklich möglich war. Die Einarbeitung in das technische und fachliche Gebiet benötigte einen Großteil der Entwicklungszeit. Aufgrund des Erfahrungszuwachses wurden zudem entwickelte Funktionen, Module, oder Elemente des Programms wieder verworfen oder mussten neu entwickelt werden. Obgleich dies natürlich eine suboptimale Vorgehensweise darstellt, bietet die aktuelle Version des Programms eine optimale Grundlage für weitere Erweiterungen und Weiterentwicklungen im kommenden Semester. Auch die vielen positiven Rückmeldungen lassen auf ein gutes Fundament für aufbauende Arbeit schliessen.

4.2. Ausblick

Der erste Teil des Masterprojekts war auf Ideenfindung und Einarbeitung in die Thematik ausgelegt. Im zweiten Teil des Projektes ist es nun wichtig, auf Basis der entwickelten Technologie die vorhandenen Konzepte weiterzuentwickeln und zu erweitern. Hier bei muss jedoch auch das Entwicklungsvorgehen dementsprechend angepasst werden. Da die Anwendung stetig komplexer wurde, ist es wichtig dass die Projektplanung und qualitätssichernde Konzepte mehr Anwendung finden. Insbesondere größere architektonische oder funktionale Änderungen sollten vorher gründlich diskutiert und geplant werden, um Fehlentwicklungen oder Nebeneffekte an anderen Komponenten zu vermeiden. Auch sollte beim weiteren Vorgehen stärker auf

verwandte Arbeiten geachtet werden um die Konzepte solide fachlich zu untermauern. Die Ergebnisse sollten regelmässig durch Usability-Tests bewertet werden, um sicherzustellen dass die aufgestellten Anforderungen, beispielsweise bezüglich intuitiver Bedienung und Effizienzgewinn auch umgesetzt werden konnten.

Konzeptionell gibt es vielfältige Möglichkeiten inwiefern man die Anwendung erweitern kann: Um den Frameworkcharakter deutlicher herauszuarbeiten wäre es sinnvoll eine zweite, auf COSMOS basierende Anwendung zu entwickeln. Dazu würde sich ein in der MOF abbildbare Diagrammart eignen, wie beispielsweise Petrinetze. Insbesondere sollte der Datenaustausch zwischen den Einzelplatzrechnern und dem interaktiven Tisch verbessert werden, so dass intuitiv und nahtlos Daten übertragen werden können. Dies schliesst unter Umständen Konzepte wie private und öffentliche Bereiche auf dem Tisch mit ein. (vgl. Smith und Piekarski (2008)). Zudem sollte man Navigationsformen durch größere Datenbestände, wie beispielsweise große Umldiagramme, entwerfen, da dies insbesondere von digitaler Technologie ermöglicht wird. Somit hat man mit einer solchen Funktion einen Vorteil gegenüber den meisten traditionellen Medien.

Literatur

[Microsoft Corporation 2009a] MICROSOFT CORPORATION: In: *http://www.microsoft.com/surface* (2009)

[Microsoft Corporation 2009b] MICROSOFT CORPORATION: Microsoft Surface User Experience Guidelines. In: *http://community.surface.com/downloads/p/156.aspx* (2009)

[Morris u. a. 2006] MORRIS, Meredith R. ; PAEPCKE, Andreas ; WINOGRAD, Terry ; STAMBERGER, Jeannie: TeamTag: exploring centralized versus replicated controls for co-located tabletop groupware. In: *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*. New York, NY, USA : ACM, 2006, S. 1273–1282. – ISBN 1-59593-372-7

[Rekimoto 2002] REKIMOTO, Jun: SmartSkin: an infrastructure for freehand manipulation on interactive surfaces. In: *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 2002, S. 113–120. – ISBN 1-58113-453-3

[Scott u. a. 2003] SCOTT, Stacey D. ; GRANT, Karen D. ; MANDRYK, Regan L.: System Guidelines for Co-located, Collaborative Work on a Tabletop Display. In: *Proc. ECSCW 2003*, 2003, S. 159–178

[Smith und Piekarski 2008] SMITH, Ross T. ; PIEKARSKI, Wayne: Public and private workspaces on tabletop displays. In: *AUIC '08: Proceedings of the ninth conference on Australasian user interface*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2008, S. 51–54. – ISBN 978-1-920682-57-6

A. XMI-Persistierung: UML-Klasse

```
/// <summary>
/// UMLClass
/// </summary>

public class UMLClass : UMLOwnedElement
{
        [XmlArray("Classifier.feature", Namespace = "org.omg.xmi.namespace.UML")]
        [XmlArrayItem(Type = typeof(UMLAttribute),ElementName = "Attribute")]
        [XmlArrayItem(Type = typeof(UMLOperation), ElementName = "Operation")]
        public List<UMLOwnedElement> UMLOwnedElements =
                new List<UMLOwnedElement>();

        [XmlAttribute("visibility")]
        public string Visible = serializationUtils.enumVisibility.Public.GetStringValue();

        public UMLClass():this(string.Empty,string.Empty) { }

        public UMLClass(string name,string id) : base(name,id)
        {
                this.Visible = serializationUtils.enumVisibility.Public.GetStringValue();
        }

        public void addAttribute(string name,string id)
        {
                this.UMLOwnedElements.Add(new UMLAttribute(name,id));
        }

        public void addAttribute(UMLAttribute a) { this.UMLOwnedElements.Add(a); }

        public void addOperation(string name,string id)
        {
                this.UMLOwnedElements.Add(new UMLOperation(name,id));
        }

        public void addOperation(UMLOperation op) { this.UMLOwnedElements.Add(op); }
}
```

B. Beispiellisting: XMI-Dokument

```
<?xml version="1.0" encoding="utf-8"?>
<xmi:xmi xmlns:uml="org.omg.xmi.namespace.UML" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  <xmi:documentation exporter="COSMOS" exporterVersion="0.001" />
  <uml:model xmi:type="uml:Model" xmi:id="anId" name="SurfaceModel">
    <packagedElement xmi:type="uml:package" xmi:id="p1" name="package1" visibility="package"
    <packagedElement xmi:type="uml:package" xmi:id="p2" name="package2" visibility="package"
      <packagedElement xmi:type="uml:Class" xmi:id="c1" name="class1" />
      <packagedElement xmi:type="uml:Class" xmi:id="c2" name="class2">
        <ownedOperation xmi:type="uml:Operation" />
        <generalization xmi:type="uml:Generalization" />
        <ownedAttribute xmi:type="uml:Property" />
      </packagedElement>
      <packagedElement xmi:type="uml:Association" xmi:id="a1" name="asso1">
        <memberEnd />
        <memberEnd />
      </packagedElement>
    </packagedElement>
  </uml:model>
</xmi:xmi>
```

C. Abbildung Expression Blend

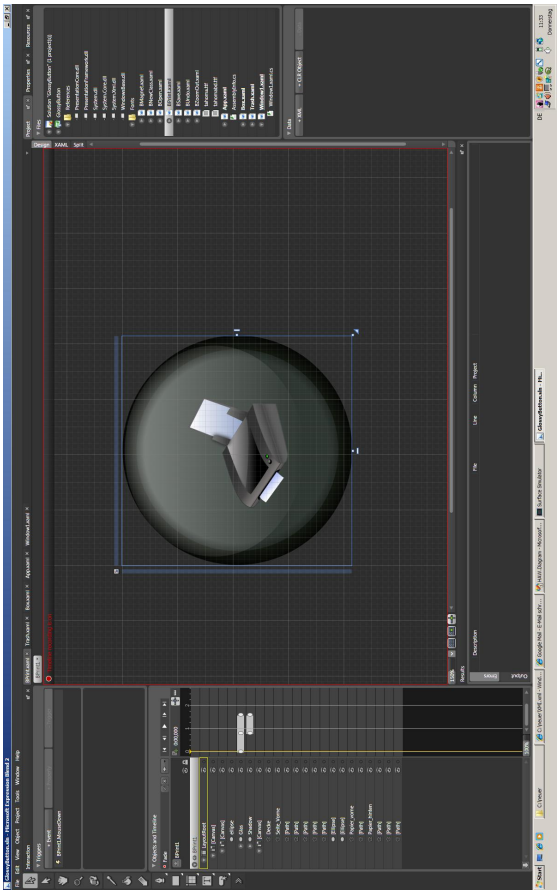

Abbildung 5: Screenshot