Universität Konstanz
FB Informatik und Informationswissenschaft
Bachelor-Studiengang Information Engineering

Bachelorarbeit

Gestaltung und Implementierung eines Multi-Fokus und Multi-Display Management-Systems

zur Erlangung des akademischen Grades eines
Bachelor of Science (B.Sc.)

Studienfach:	Information Engineering
Schwerpunkt:	ComputerScience
Themengebiet:	Angewandte Informatik

von
Jonas Schweizer

Matr.-Nr.:	01/641081
Erstgutachter:	Prof. Dr. Harald Reiterer
Zweitgutachter:	Prof. Dr. Daniel A. Keim
Betreuer:	Hans-Christian Jetter
Einreichung:	27.07.2009

Zusammenfassung

Die enorme Zunahme an gespeicherter Information führt dazu, dass heutige Informationskonsumenten mit der Bewältigung dieser Informationsmengen überfordert sind. Diese Bachelorarbeit setzt sich mit dem Problem auseinander, wie Benutzer bei der Interaktion in großen Informationsräumen unterstützt werden können. Es geht dabei um die Frage, wie die kognitive Überlastung des Benutzers mit Hilfe neuer Sichten auf die Informationen gesenkt werden kann. Im Rahmen dieser Arbeit wird eine Taxonomie vorgestellt, die Benutzerinteraktionen und verschiedene Umgebunssituationen in Relation stellt. Die Taxonomie unterscheidet zwischen der *Manipulation*, der *Anzeige* und dem *Verschieben* von Objekten in Single- und Multi-Display Kontexten. Die Taxonomie wird anschließend auf das bestehende ZOIL-Konzept angewendet, um daraus Lösungen für multifokales Arbeiten zu entwickeln. Mit Hilfe von *Split Screen*, *mehrfachem Click Zoom*, *Drag&Drop*, einer *interaktiven Übersicht*, *synchronisierten Ansichten*, dem *Remote Click Zoom* und einem *Shared Clipboard*-Konzept wird ZOIL mit Funktionen erweitert, die das Arbeiten in *multifokalen Arbeitsumgebungen* erleichtert. Um eine möglichst einfache Verwendung und Weiterentwicklung der Konzepte zu ermöglichen, wurden die vorgestellten Konzepte in das ZOIL Framework integriert. Die Arbeit formuliert Verbesserungsvorschläge für die Frameworkarchitektur und schließt mit einem Ausblick auf mögliche Anknüpfungspunkte für weitere Forschungsprojekte.

Inhaltsverzeichnis

1 Einführung

> „Yet information overload is a major concern: we cannot handle 'everything, everywhere, all the time'." [Grudin, 2001]

Der heutige Stand der Informationstechnologie schafft die Möglichkeit, immer mehr Information zu sammeln und zu speichern. Dies führt zu einer ständigen Zunahme der Masse an Information, die es zu verarbeiten gilt. Jeder Mensch hat in der heutigen Zeit, sowohl an seinem Arbeitsplatz, als auch im Privaten mit einer Fülle an Informationsobjekten zu tun. Die steigende Zahl an Webseiten im Internet, reichhaltige Sammlungen von Musik, Fotos, Dokumenten oder wissenschaftlichen Zahlen sind nur wenige Beispiele.

Hinzu kommt, dass der heutige Informationskonsument diese Massen an Information nur noch durch den Computer erforscht. Der klassische Desktop-PC ist dieser Fülle an Information im Bezug auf die Visualisierung von Daten und Benutzerschnittstellen nicht mehr gewachsen. Der Benutzer betrachtet einen riesigen Raum durch ein viel zu kleines Fenster.

Forschungen im Bereich der Mensch-Computer-Interaktion und aus weiteren Disziplinen haben in den vergangenen Jahren verschiedene Richtungen eingeschlagen, um dieses Problem zu lösen. Es geht um die Frage, wie der Mensch dieses reichhaltige Informationsaufkommen bewältigen kann. Im Bereich der Verwaltung großer Datenmengen geht es dabei auch um Interaktions- und Navigationskonzepte, die den Benutzer in der Verwaltung der Daten unterstützen.

Ein Teilaspekt dieser Forschungen beschäftigt sich mit der Frage, bisherige Ansichten der Information zu erweitern. Bleibt man bei dem oben genannten Beispiel, in dem der Benutzer durch ein kleines Fenster blickt, um einen großen dahinter liegenden Raum zu betrachten, so muss man sich die folgende Frage stellen: Wie kann man die Funktionen des Fensters erweitern, um mehr von dem dahinter liegenden Raum zu sehen oder ist es vielleicht besser, neue Fenster einzubauen? An diesem Beispiel wird deutlich, welche unterschiedlichen Vorgehensweisen sich in diesem Bereich herausbilden.

Ein Ansatz versucht dabei den Benutzer an einem einzelnen Display mit neuen Werkzeugen, Möglichkeiten und Techniken auszustatten, so dass er leichter auf die dahinter liegenden Informationen zugreifen kann. Man spricht von einem Multi-Fokus System.

Ein anderer Ansatz beschäftigt sich mit der Erweiterung der Einstiegsmöglichkeiten in den Informationsraum durch neue Geräte, Displays und deren Verbindung. In diesem Fall spricht man von einer Multi-Display Environment (MDE).

Trotz unterschiedlicher forschungsstrategischer Perspektiven verfolgen beide Ansätze das gleiche Ziel, den Benutzer bei der Verarbeitung vieler Informationen kognitiv zu

unterstützen. Schon Donelson kombiniert in seinem „Spatial Data Management System“ Techniken aus Zooming als Navigationskonzept in einem Display und einer MDE als Erweiterung der Einstiegsmöglichkeiten [Donelson, 1978]. Seit dieser Zeit haben sich sowohl Soft- als auch Hardware rasant weiterentwickelt. Zum einen haben Rechen- und im speziellen Grafikleistung ein Niveau erreicht, welches ganz neue Möglichkeiten für grafische Benutzerschnittstellen schafft. Auf der anderen Seite sind Technik und Kosten von Bildschirmen kein limitierender Faktor mehr.

Dies führte dazu, dass in den letzten Jahren verstärkt Forschung in diesem Bereich betrieben wurde. Bereits in den 80er und 90er Jahren beschäftigten sich Forscher mit MDEs [Stefik et al., 1987], neuen Navigationskonzepten [Perlin and Fox, 1993] und multifokalen Sichten [Bier et al., 1993].

Im Rahmen dieser Arbeit werden verschiedene Studien rekonstruiert, um Erkenntnisse aus den einzelnen Teilbereichen zu bündeln. Vordergründiges Ziel ist es, den Benutzer bei der Verarbeitung großer Informationsmengen zu unterstützen. Eine typische Aufgabe in großen Informationsräumen ist das Vergleichen **mehrerer Objekte**. Vor diesem Hintergrund, beschäftigt sich diese Arbeit mit Theorien und praktischen Konzepten, die den Benutzer dabei unterstützen mit mehreren fokussierten Bereichen zu arbeiten, d.h. wenn er multifokal arbeitet.

Um einen Einblick in die Vielfalt des Themas zu bekommen werden in Kapitel 2 einige Arbeiten vorgestellt, die sich mit diversen Thematiken des multifokalen Arbeitens auseinander setzen.

In Kapitel 3 wird ein gemeinsames Verständnis einer *multifokalen Arbeitsumgebung* geschaffen. Anschließend wird ein zunehmend wichtig werdender Aspekt für solche Umgebungen untersucht: die Interaktion. Es wird die Frage gestellt, auf welche Weise Benutzer agieren und wie sich diese Aktionen evtl. in verschiedenen Kontexten unterscheiden. Im Verlauf des Kapitels 3 wird eine Taxonomie vorgestellt, welche den Zusammenhang verschiedener Kontextsituationen und generischer[1] Aktionen innerhalb einer *multifokalen Arbeitsumgebung* aufzeigt.

Im Anschluss wird die Taxonomie auf das bereits bestehende Interface-Paradigma ZOIL[2] angewendet und es werden praktische Lösungen für die einzelnen Kategorien der Taxonomie vorgestellt. Kapitel 5 erläutert, wie die in Kapitel 4 erarbeiteten Komponenten in das bestehende ZOIL Framework[3] integriert wurden, um eine weitere Verwendung zu vereinfachen. Die Arbeit endet mit einem Ausblick, in dem mögliche Erweiterungen der Taxonomie dargelegt und verschiedene Ideen für zukünftige Arbeiten vorgestellt werden.

[1] Der Ausdruck „generisch“ wird im Rahmen dieser Arbeit wie der Ausdruck „im Allgemeinen gültig“ verwendet, vgl. auch [Duden, 2006].

[2] Zoomable Object-oriented Information Landscape (ZOIL)

[3] Das ZOIL Framework bietet die technische Grundlage für das Erstellen von High-Fidelity Prototypen nach dem ZOIL-Paradigma.

2 Verwandte Arbeiten

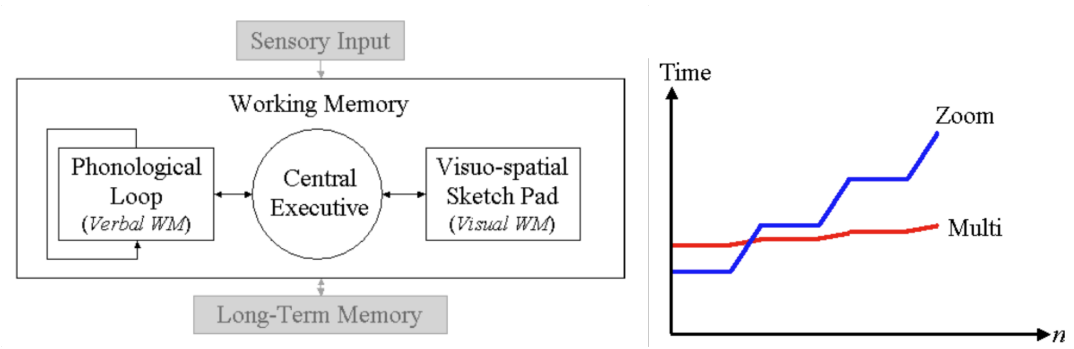
Verschiedene Forschungsfelder im Bereich der Mensch-Computer-Interaktion haben Einfluss auf den Bereich von multifokalem Arbeiten. Das folgende Kapitel stellt einige Forschungsergebnisse dar. Dabei werden die Arbeiten verschiedenen Disziplinen zugeordnet. Es wird deutlich, dass sich Fragestellungen und Ergebnisse dieser Arbeiten gegenseitig ergänzen und bereichern.

2.1 Multifokale Sichten

Zunächst werden Arbeiten vorgestellt, die sich damit beschäftigen, verschiedene Sichten auf den Informationsraum zu generieren. Dabei geht es sowohl um Informationsvisualisierung, wie auch um konkrete Benutzerschnittstellen.

Plumlee und Ware beschäftigen sich mit kognitiven Aspekten bei multifokalen Sichten [Plumlee, 2002]. Sie entwickeln ein kognitives Modell des Arbeitsgedächtnisses (vgl. Abbildung 1a). Das Modell beinhaltet die Speicherung von Informationen im „visuellen Arbeitsgedächtnis“. Diesem Teil des Modells wird die Aufgabe zugeschrieben, kurzzeitig Information visuell zu speichern. Die Autoren stellen fest, dass sich ein Mensch ca. drei Objekte merken kann. Plumlee und Ware leiten darüber hinaus ein mathematisches Modell für die menschliche Performanz bei der Bearbeitung vergleichender Aufgaben ab und evaluieren es in einer Studie. Im Vergleich stehen *Zoom* und *Multiple Window* Techniken. Es zeigt sich, dass *Zoom*-Techniken zunächst besser abschneiden, aber sobald eine bestimmte Anzahl an Objekten überschritten wird, sind *Multiple Windows* schneller (vgl. Abbildung 1b). Diese Beobachtungen führen die Autoren auf das vorher beschriebene kognitive Modell zurück. *Zooming* ist schneller, wenn die Kapazität des visuellen Arbeitsgedächtnisses ausreicht. Sobald nicht mehr genügend Objekte gespeichert werden können, ist das *Multiple Window*-Interface überlegen, da es mehr als ein Objekt visuell anzeigen kann.

Cockburn et al. untersuchen in ihrer State-of-the-art Analyse verschiedene Systeme, die multifokale Sichten ermöglichen [Cockburn et al., 2008]. In dieser Übersicht werden relevante Techniken vorgestellt, die Lösungen beschreiben, um die Massen an Information auf limitiertem Platz besser verarbeiten zu können. Die Techniken werden vier Kategorien zugeordnet: Übersicht+Detail, Zooming, Fokus+Kontext und hinweisbasierte Systeme. Die Autoren hinterfragen die einzelnen Systeme und stellen empirische Untersuchungen vor, die den Erfolg und Misserfolg der Techniken beschreiben. Es zeigt sich, dass keine Technik herausragt und dass die Effektivität einer Technik stark vom Anwendungsfall und den Aufgaben des Benutzers abhängt. Neben den Konzepten, die Cockburn beschreibt, gibt es weitere Arbeiten, die sich mit multifokalen Sichten auseinander gesetzt haben. Zu nennen ist hier die Technik, die Elmqvist et



(a) *Visuelles und verbales Arbeitsgedächtnis in einem kognitiven Modell*

(b) *Zoom-Techniken sind langsamer mit zunehmender Anzahl an Objekten*

Abbildung 1: *Modell und Evaluation zur kognitiven Performanz in vergleichenden Aufgaben [Plumlee, 2002].*

(a) *Eindimensionale Clipview mit zwei Fokussen*

(b) *Zweidimensionale Visualisierung mit mehr als 2 Fokussen*

Abbildung 2: *Multifokale Visualisierung mit Mélange [Elmqvist et al., 2009]*

al. vorstellen. Mélange ist eine neue Technik zur Visualisierung von mehreren Fokussen [Elmqvist et al., 2008]. Mélange faltet dabei Visualisierungen, die in 1D oder 2D dargestellt werden, in einer 3D-Visualisierung in den Raum (vgl. Abbildung 2). Auf diese Weise bleibt mehr Information über Kontext und Distanz zwischen den einzelnen Fokuspunkten erhalten. Mit Hilfe einer Benutzerstudie stellen die Autoren fest, dass Mélange bei normalen Suchaufgaben nicht langsamer als ein Split Screen oder ein normales Interface ist, sich aber deutlich besser verhält, wenn es um Aufgaben mit Bezug zu Kontext oder Distanz geht. In einer weiteren Arbeit erweitern Elmqvist et al. ihre Erläuterungen und stellen sowohl die technische Umsetzung, wie auch weitere Anwendungsszenarien vor [Elmqvist et al., 2009]. Die Antwort über ein schlüssiges Interaktionskonzept bleiben die Autoren schuldig. In den Benutzerstudien werden nur eindimensionale Aufgaben bearbeitet, wodurch dieser Umstand nur wenig zum Tragen kommt. Für den Einsatz von Mélange-Technik, über die Visualisierung hinaus, ist ein schlüssiges Interaktionskonzept notwendig.

2.2 Interaktionstechniken

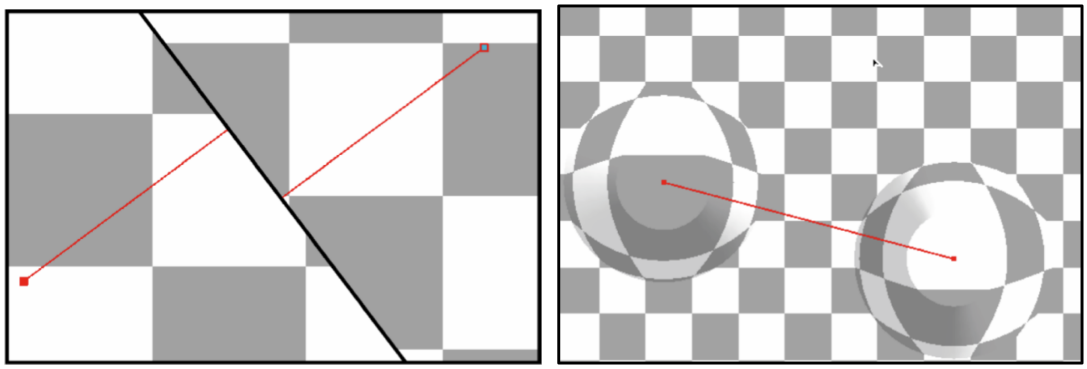
Wobbrock et al. liefern eine sehr interessante Arbeit zu Interaktionstechniken [Wobbrock et al., 2009]. Auch wenn diese nicht explizit auf multifokale Interaktion abzielt, so kann diese Arbeit durchaus als Referenz dienen, wenn es um die Festlegung einer Interaktionsgeste geht. Ziel war es, benutzerdefinierte Gesten zu identifizieren. 20 Probanden mussten auf einen vorgelegten Effekt (Beispiel: Zoom In) eine Interaktion ausführen, mit der sie ihrer Meinung nach den Effekt erreichen könnten. Mit 27 Effekten, 20 Teilnehmern und jeweils einer Unterscheidung zwischen ein- und zweihändig wurden 1080 Gesten aufgezeichnet.

Die Ergebnisse liefern einen Satz an Gesten und Aussagen darüber, welche Zustimmung die einzelnen Gesten erhalten haben. Die Autoren formulieren einige allgemeine Feststellungen: Benutzer verwenden für dichotome Handlungen (Zoom in / Zoom out) umkehrbare Gesten. Weiter vereinfachen Benutzer in ihrem mentalen Modell manche Gesten und halten bspw. „Zoom In“ und „Vergrößern“ für die gleiche Aktion. Die Anzahl der Finger ist häufig nicht ausschlaggebend, wobei die Autoren einen Unterschied sehen zwischen 1-3 und 5 Fingern. Das mentale Modell der Benutzer ist zudem sehr stark an den Windows-Rechner gebunden, bspw. wurden für den Effekt „Minimieren“ häufig die Objekte an den unteren Rand gelegt, in Richtung „Taskbar“.

Überraschenderweise haben die Benutzer nicht nur auf dem Screen agiert sondern auch daneben und darüber. Diese Erkenntnis zeigt neue Möglichkeiten der Interaktion und erfordert die Erweiterung des Trackings auf die Hände über dem Tisch.

Ein Vergleich zwischen den benutzerdefinierten Gesten und von Experten festgelegten Gesten belegt zudem, dass Designer nicht das gleiche Maß an Gesten antizipieren, wie der tatsächliche Benutzer. Nur 60,90 % der Benutzergesten wurden von drei Experten angewendet. Zusätzlich stellen die Autoren eine Taxonomie vor, in der sie Gesten in die Kategorien *Form*, *Natur*, *Bindung* und *Fluss* einordnen. Diese Arbeit zeigt eine benutzerorientierte Methode für das Identifizieren neuer Gesten.

Shoemaker und Gutwin beschreiben drei Techniken für multifokale Interaktion [Shoemaker and Gutwin, 2007]. Neben einer neuen Split Screen Technik, die automatisch splittet, werden zwei Fisheye-Techniken vorgestellt (vgl. Abbildung 3). Die *Split Scrolling* Technik erzeugt einen Split, sobald der Benutzer mit einem Fokuspunkt den Bildschirm verlässt. Beide Interaktionspunkte bleiben sichtbar und sind durch eine Linie getrennt. Die beiden Fisheye-Techniken unterscheiden sich nur in der Interaktion. *Fisheye Auto-Zoom* setzt voraus, dass der derzeitige Skalierungsgrad dem Benutzer ausreicht als Detailansicht und zoomt ähnlich dem *Split Scroll* bei Verlassen des Bildschirm-Bereichs automatisch, so dass immer alle Interatkionspunkte zu sehen sind. Der Zoomfaktor der Fokuspunkte bleibt auf dem Anfangsniveau. Die *Fisheye Overview* Technik setzt voraus, dass der Benutzer die nötigen Interaktionspunkte schon im Bildschirm sieht. Die



(a) *Sobald der Benutzer über den Bildschirm hinaus scrollt, wird ein Split erzeugt*

(b) *Schematische Ansicht bei mehreren Fisheyes*

Abbildung 3: *Split Scroll und Fisheye Technik für multifokale Interaktion [Shoemaker and Gutwin, 2007].*

Fisheye-Fokuspunkte werden auf diese Interaktionspunkte plaziert. D.h. hier behält der Benutzer den gleichen Kontext und lediglich die Fokuspunkte werden durch die Fisheyes vergrößert.

Die *Split Scroll* Technik wird nicht implementiert und evaluiert. Die beiden *Fisheye*-Techniken werden mit klassischen *Pan* bzw. *Zoom* Techniken verglichen. In der Studie mussten Benutzer bestimmte Kontrollpunkte setzen, die nur mit Hilfe der Vergrößerungs-Technik erreichbar waren.

Abbildung 4 zeigt das Ergebnis. Die beiden Fisheye-Techniken waren schneller als die beiden klassischen Varianten. Wie schon erwähnt wurde die Technik des Split Scroll nicht evaluiert.

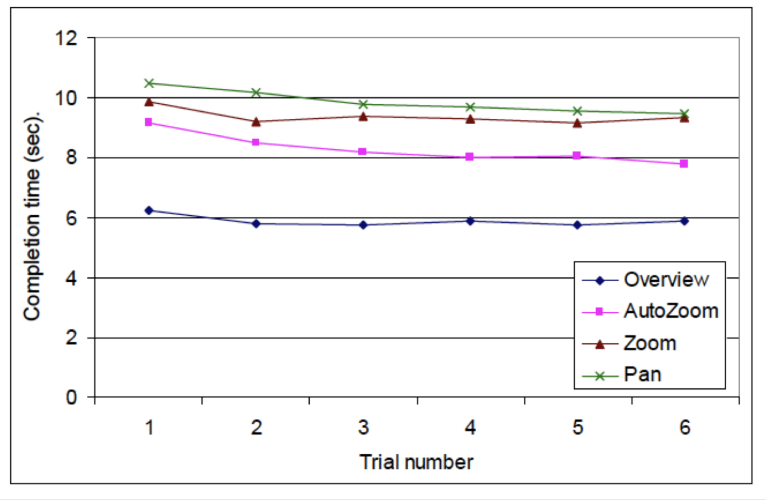


Abbildung 4: *Ergebnisse einer Studie, die vier verschiedene Techniken für die Positionierung von mehreren Fokuspunkten vergleicht [Shoemaker and Gutwin, 2007].*

Naacenta et al. stellen in ihrem Artikel eine Taxonomie für das Verschieben von Objekten zwischen verschiedenen Displays vor [Naacenta et al., 2009]. Zunächst erarbeiten sie ein Model für das Verschieben zwischen Displays und erläutern ihre Bewertungskriterien.

Das Modell sieht vier Verarbeitungsschritte vor (vgl. auch Abbildung 5):

1. Zunächst müssen die Anforderungen der Aufgabe in eine „Intention zum Bewegen“ umgesetzt werden, wozu auch gehört, das richtige Display auszuwählen
2. Anschließend wird ein Aktionsplan ermittelt
3. Im nächsten Schritt wird die Aktion ausgeführt
4. In manchen Fällen muss der Benutzer eine Feedback-Schleife durchlaufen und die Bewegung anpassen

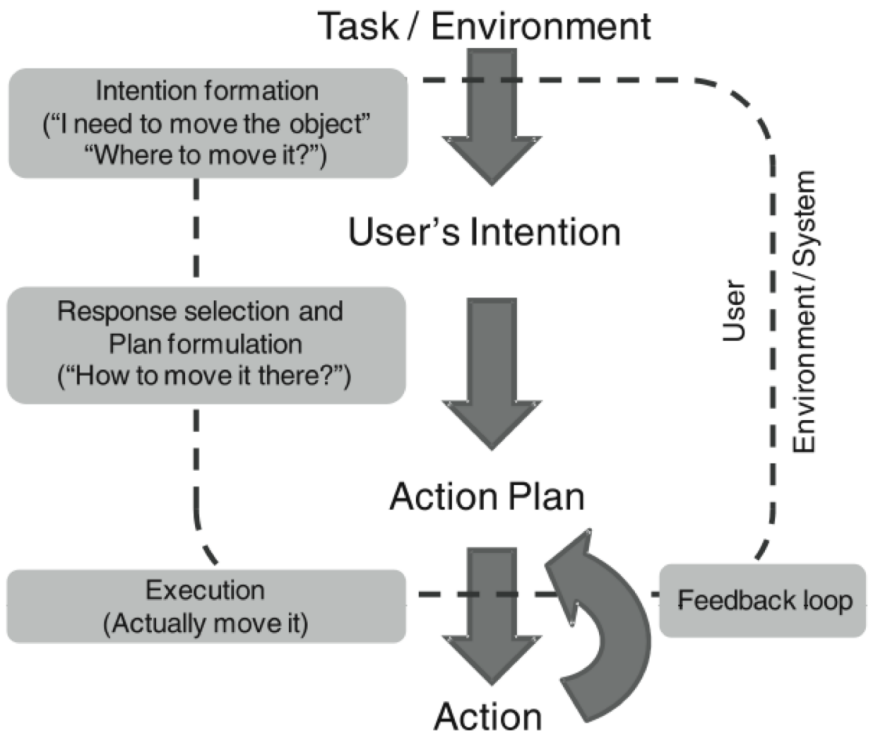


Abbildung 5: *Modell kognitiver Prozesse zur Ausführung einer Objektbewegung zwischen Displays [Naacenta et al., 2009].*

Die Autoren bewerten die einzelnen Konzepte nach drei Kriterien: Performanz, Mächtigkeit und Feedback. Die Taxonomie für das Bewegen von Objekten zwischen verschiedenen Displays gliedert sich in drei Stufen: *referentielle Domäne*, *Displaykonfiguration* und *Kontrollparadigma* (vgl. Abbildung 6). Die Taxonomie orientiert sich an dem zuvor

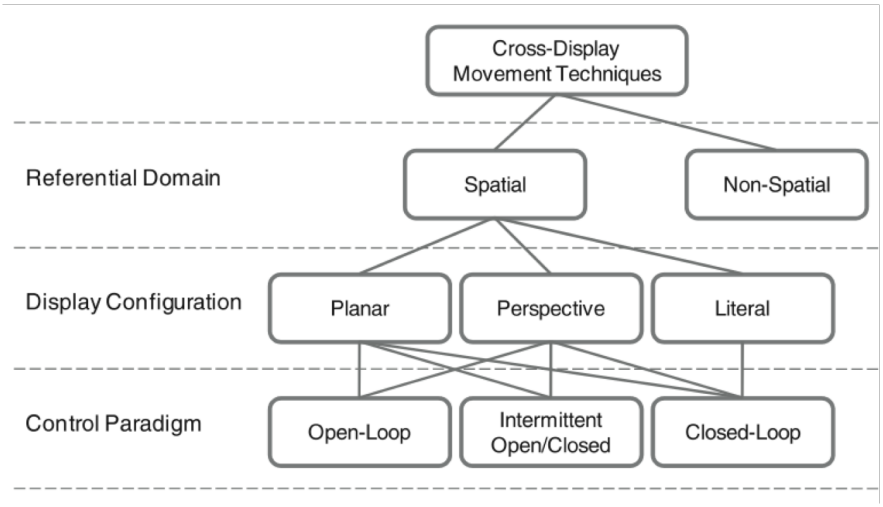


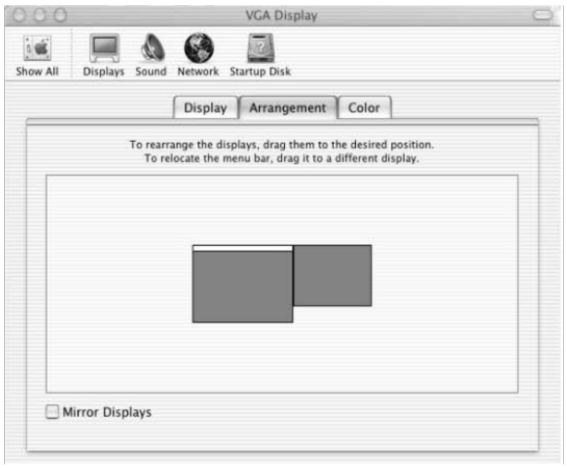
Abbildung 6: *Taxonomie für das Bewegen von Objekten zwischen verschiedenen Displays [Naacenta et al., 2009].*

beschriebenen Modell und umfasst hauptsächlich mentale Prozesse, welche bei einer Objektbewegung eine Rolle spielen.

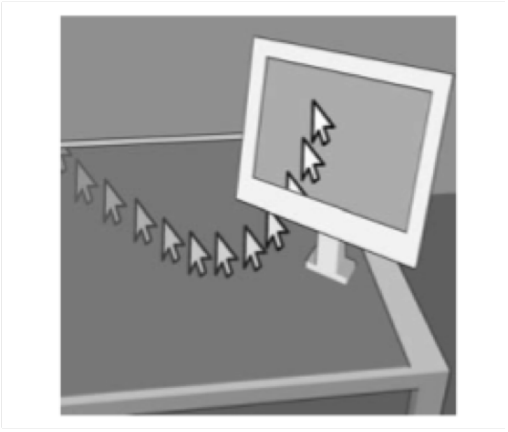
Referentielle Domäne Bevor der Benutzer eine Aktion ausführen kann, muss er ein Display auswählen. Hierbei wird zwischen „räumlicher“ und „nicht-räumlicher“ Zuordnung unterschieden.

Displaykonfiguration Diese Kategorie stellt die Frage: Wie wird das physikalische Layout der Konfiguration intern verarbeitet. Die erste Möglichkeit ist, dass der Benutzer die Monitore in einer Art „Ebene“ anordnet und diese virtuelle Anordnung auf das tatsächliche Layout mappt (vgl. Abbildung 7a). Die zweite Option ist die Zuordnung mit Hilfe der Perspektive des Benutzers auf das Monitor-Setting (vgl. Abbildung 7b). Die dritte Option ist eine buchstäbliche Zuordnung, direkt referenziert durch die physikalische Anordnung. Beispiel wäre die Pick-and-Drop Technik [Rekimoto, 1997], in der die Zuordnung direkt über den physischen Gegenstand zum Display erfolgt.

Kontrollparadigma Unter Kontrollparadigma versteht man die Möglichkeit der Kontrolle während einer Aktion. Es werden drei Arten unterschieden. Ist eine Aktion „offen“ so bekommt der Benutzer entweder kein Feedback oder er hat nach Ausführen der Aktion keine Möglichkeit mehr in den Prozess einzugreifen. Bei einer „geschlossenen“ Technik, kann der Benutzer zu jedem Zeitpunkt, während die Aktion läuft, noch eingreifen. Eine Mischung aus beiden ergibt sich, wenn das Feedback bspw. nicht kontinuierlich gegeben ist. Beispiel wäre die MouseEther Technik [Baudisch et al., 2004], bei der man mit der Maus auch den Raum **zwischen** zwei Displays überbrücken muss. Während die Maus zwischen den Displays po-



(a) *Zuordnung über planare Anordnung der Konfigurationsoberfläche*



(b) *Zuordnung aus der Perspektive des Benutzers*

Abbildung 7: *Planare und perspektivische Zuordnung zwischen physikalischem Layout und Konfigurationsoberfläche [Naacenta et al., 2009].*

sitioniert ist, hat der Benutzer kein Feedback über die Position. Man spricht dann auch von einem „abwechselnden offenen / geschlossenen“ Paradigma.

Nacenta et al. stellen mit ihrer Arbeit eine systematische Beschreibung vor, die viele verschiedene Facetten der Bewegung von Objekten zwischen Displays abdeckt.

2.3 Multi-Display Umgebungen

Multi-Display Umgebungen, also Hardware Settings, in denen zwei oder mehr getrennte Displays vorhanden sind, spielen in den vergangenen Jahren eine wachsende Rolle. Fallende Preise und verbesserte Technologie ermöglichen einen raschen Einzug in Büros, an öffentlichen Plätzen oder in privaten Haushalten [Balakrishnan and Baudisch, 2009, Dearman and Pierce, 2008].

Eine umfassende Einführung in das Thema bietet das Journal Human-Computer Interaction in seiner Ausgabe zum Thema „Ubiquitous Multi-Display Environments“[4]. Die Autoren orientieren sich dabei an zwei hauptsächlichen Untersuchungsfragen [Balakrishnan and Baudisch, 2009]:

1. Auf welche Weise beeinflussen verschiedene Arten von Displays Benutzer und menschliche Zusammenarbeit?

2. Wie kann man Multi-Display Umgebungen gestalten, damit sie nahtloser miteinander arbeiten?

[4] Für eine Übersicht über das Journal, siehe `http://www.informaworld.com/smpp/title~db=all~content=g910607683`

Das Journal bietet einen guten Überblick über verschiedene Teilaspekte im Bereich der Multi-Display Umgebungen. Darüber hinaus sind zahlreiche Arbeiten entstanden, die sich mit speziellen Thematiken auseinander setzen.

Einige Arbeiten untersuchen, welche Rollen verschiedene Displays in einer Multi-Display Umgebung einnehmen können und welche Aufgabe sie dabei unterstützen.

In einer Studie von Grudin wurden 18 Teilnehmer in strukturierten Interviews nach ihrem Verhalten in Multi-Display Umgebungen befragt [Grudin, 2001]. Es werden fünf Beobachtungen angeführt: Ein einzelnes Fenster wird in der Regel nicht über mehrere Monitore hinweg angezeigt; Benutzer ordnen einem Display eine Hauptaufgabe zu und bearbeiten auf weiteren Displays sekundäre Aufgaben, wie E-Mail oder Chat; Benutzer sind erleichtert über die Möglichkeit mehrere Fenster gleichzeitig sichtbar zu haben und dadurch weniger zwischen den Fenstern wechseln zu müssen; die Befragten haben gerne mehrere Displays benutzt und Anwendungen unterstützen Multi-Display Umgebungen zu wenig. Grudin macht darüber hinaus Aussagen zum Vorteil von Multi-Display Umgebungen, im Gegensatz zu großen Displays. Der Vorteil von mehreren getrennten Bildschirmen liegt demnach in der Möglichkeit Objekte auf ein peripheres Display zu legen. Diese Möglichkeit ist auf einem großen Display nicht gleichermaßen gegeben.

Perry und O'Hara führten eine ähnliche Studie durch [Perry and O'Hara, 2003]. 14 Personen wurden befragt welche Arten von Displays sie benutzen, aus welchen Gründen sie diese einsetzen und auf welche Weise sie bestimmte Arten von Displays erzeugen. Dabei sprechen sie auch von nicht digitalen Displays, wie bspw. Pinnwänden.

Drei Kernpunkte werden in einer Taxonomie zusammengefasst. Der schnelle Zugang zu Information, die soziale Orientierung sowie Koordination und Planung. Auch in dieser Arbeit werden aus den Beobachtungen Design-Ideen abgeleitet. Sie stellen dabei heraus, dass es nicht notwendig ist, elektronische Duplikate von papierbasierten Systemen zu erstellen, sondern dass es notwendig ist, die Motivation und die Gründe der Benutzer zu verstehen, warum und wie sie ein Display benutzen. Sie weisen zudem darauf hin, dass man keine Displays entwirft, die mit ihrer Rolle mit anderen Displays in Konflikt stehen und dadurch andere Aktivitäten behindern. Darüber hinaus erläutern sie einige Hinweise: Information sollte zu jeder Zeit lesbar sein, Information muss leicht positioniert werden können, die Position eines Displays ist wichtig und sollte leicht zu verändern sein. Displays sollten Erinnerungsfunktionen übernehmen, Informationen müssen klar voneinander zu unterscheiden sein und die räumliche Anordnung des Displays und der Information innerhalb des Displays helfen dem Benutzer.

Während sich diese Arbeiten eher auf die Funktionen und Rollen verschiedener Displays fokussieren, spielt in weiteren Untersuchungen eher das Zusammenspiel verschiedener Displays und Geräte eine Rolle.

Dearman und Pierce gehen mit einer Studie, an der 27 Personen aus Wissenschaft und

Industrie teilnahmen, den Fragen nach, wie und warum Benutzer mehrere Geräte einsetzen [Dearman and Pierce, 2008]. Aus der Befragung gehen vier Ergebnisse hervor:

1. Eine Zuordnung zwischen Aktivität und Gerät ist nicht mehr so einfach möglich, da Aktivitäten mehrere Geräte überspannen.
2. Die Nutzung von Geräten hängt von den Benutzern und den Umständen ab.
3. Benutzer aus der Industrie wollen in der Regel Geräte zwischen Arbeit und Privatem trennen, aber das gelingt häufig nicht.
4. Nutzer beschäftigen sich viel mit Techniken, die den Zugriff auf Informationen von verschiedenen Geräten ermöglichen.

Die Autoren formulieren folgende Verbesserungsvorschläge. Der Fokus der Betrachtung sollte auf dem Benutzer und nicht auf den Geräten liegen, Rollen und Kontexte müssen klar herausgestellt werden, Informationsübertragung muss leichter möglich sein und Synchronisation als Mittel muss besser unterstützt werden.

Auch Tang und Fels untersuchen in ihrer Studie traditionelle Multi-Display Umgebungen [Tang and Fels, 2008]. Sie differenzieren zwischen zwei Schwerpunkten: einerseits die Arbeit mit traditionellen Whiteboards, andererseits eine Laborumgebung mit verschiedenen Displays. Auch hier können Displays als Anzeigeflächen verstanden werden und sowohl digital, wie auch nicht digital sein.

Das Ergebnis dieser Studie wird in einer Taxonomie verortet, die vier Folgerungen herausstellt:

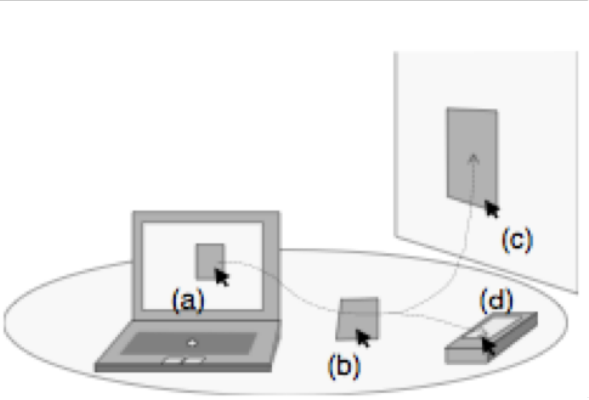
Visuelle Persistenz Visuelle Persistenz bedeutet, dass die Information erst gelöscht wird, wenn eine explizite Aktion zum Löschen ausgeführt wird. Der Vorteil von visueller Persistenz liegt darin, dass die Information sehr schnell abgerufen werden kann, zum Teil nur mit einem kurzen Blick.

Fixe Funktion Bestimmte Displays haben eine feste Funktion. Beispiele hierfür sind In / Out Boards, die anzeigen, wer sich im Büro aufhält oder Listen von bekannten Software-Bugs, die noch zu lösen sind.

Semantische Konstruktion Semantische Konstruktion bedeutet, dass Displays in der Regel nicht nur eine bestimmte Aufgabe erfüllen. Vielmehr bestimmen die Benutzer durch verschiedene Arten von Inhalt, welche Funktion das Displays verwendet. Diese semantische Interpretation kann sich im Laufe der Zeit ändern. Die Autoren stellen fest, dass heutige digitale Displays oft eine zu starre Funktion unterstützen.

Immobilität von Information Ein weiteres Problem ist die schwierige Informationsübertragung. Benutzer machen sich daher sehr genau Gedanken, welche Information sie auf welches Display positionieren.

(a) *Ein ununterbrochener Arbeitsraum, der kollaborative Arbeit unterstützt durch Einbindung mobiler Geräte*



(b) *Hyperdragging: Technik zum Verschieben von Objekten zwischen den Geräte (Schema)*

Abbildung 8: *Augmented Surfaces: Ein Mutli-Display Arbeitsbereich [Rekimoto and Saitoh, 1999].*

Neben den Analysen, wie Benutzer in Multi-Display Umgebungen interagieren und welche Rolle bestimmte Displays einnehmen, beschäftigen sich weitere Arbeiten mit konkreten Settings.

Ein Beispiel sind die „Augmented Surfaces“, eine räumlich nicht unterbrochene Arbeitsumgebung, in welche verschiedene Geräte integriert sind [Rekimoto and Saitoh, 1999]. Die Landschaft besteht aus vorinstallierten Geräten wie dem „Infotable“ (Tisch mit Anzeigefläche) und der „Infowall“ (Anzeigefläche an der Wand) (siehe Abbildung 8a). Zusätzlich werden über ein Kamerasystem neue Geräte per Marker erkannt und dem virtuellen Arbeitsbereich hinzugefügt. Eine neue Technik, das sogenannte „Hyperdragging“ ermöglicht es, Objekte von verschiedenen Geräten auf einer gemeinsamen Landschaft zu bewegen und so zwischen den Geräten zu verschieben. Das System erkennt die Position aller Geräte. Aktionen der Maus werden auf den zusammenhängenden Arbeitsbereich gemappt (vgl. Abbildung 8b). Durch die Verbindung dieser verschiedenen Techniken entsteht ein logisch zusammenhängender Arbeitsplatz, auf dem Informationen zwischen verschiedenen Geräten und Displays verschoben werden können.

Weitere Beispiele für interaktive Umgebungen sind der WeSpace [Wigdor and Keywords, 2009], der iRoom [Johanson et al., 2002] oder I-Land [Streitz et al., 1999]. Letzteres hat vor allem auch Einfluss auf kooperative Arbeit und wird im nächsten Kapitel vorgestellt.

2.4 CSCW

Besonders der Bereich von Computer Supported Cooperative Work (CSCW), also der vom Computer unterstützen kooperativen Arbeit, beschäftigt sich mit dem Thema verschiedener Sichten auf einen Datenbestand, verteilt auf mehrere Geräte und Displays. Multi-Display Umgebungen bieten erstaunliche Möglichkeiten für Kollaboration in Gruppen [Biehl et al., 2008]. Im Rahmen dieser Arbeit geht man davon aus, dass hauptsächlich ein Benutzer mit einem System interagiert, während die weiteren Benutzer passiv oder abwechselnd interagieren. Dennoch werden im Folgenden einige Konzepte vorgestellt, die mit Hilfe von multifokalen Techniken das kollaborative Arbeiten unterstützen. Diese kurze Analyse soll zeigen, wo die Schnittmengen zwischen CSCW und dieser Arbeit liegen.

Mit IMPROMPTU stellen Biehl et al. ein Interaktionsframework zur Unterstützung kollaborativer Arbeit vor [Biehl et al., 2008]. IMPROMPTU wurde für ein Setting entwickelt, in dem Benutzer sich gemeinsam in einem Raum befinden und dabei sowohl individuell arbeiten, wie auch gemeinsam Aufgaben lösen. In der Studie, welche die Autoren vorstellen, ging es konkret um ein Team von Software-Entwicklern, die gemeinsam in einem Büro arbeiteten. IMPROMPTU bietet dabei folgende Features: Standardapplikation können ohne Veränderungen geteilt[5] werden, d.h. die visuelle Repräsentation wird zwischen verschiedenen Geräten übertragen. Dieses Verteilen kann auf mehrere Displays erfolgen. Zudem ist es möglich, dass die lokale Eingabe auf andere Displays gemappt wird. Dies fördert die kollaborative Interaktion. Benutzer können außerdem gleichzeitig ihre Applikationen auf einem großen Display anzeigen.

Das Interface von IMPROMPTU besteht aus einem „Collaboration Control“, zur Freigabe von Applikationen, einer „Collaborator Bar“, in der die Kollegen ihre Applikationen teilen und „Shared Screen Docs“, die als Repräsentanten großer Displays dienen (siehe Abbildung 9).

Nach einer kurzen Beschreibung der Implementierung folgt die Analyse der Feldstudie. Dabei wurden zwei Teams (Team Alpha und Team Beta) drei Wochen lang beobachtet. Mitglieder von Team Alpha waren stets in einem Raum, während die Mitglieder von Team Beta individuelle Büros hatten.

Die Studie macht deutlich, dass die Funktion „Anwendungen teilen“ sowohl innerhalb der Mitglieder, als auch für das große, gemeinsame Display genutzt. Für die Autoren war der Einsatz großer Displays, um passiv die Aufmerksamkeit auf eine bestimmte Information zu richten, sehr interessant. Es wurde bspw. ein wichtiger Nachrichtenartikel auf dem Display sichtbar gemacht, ohne näher darauf aufmerksam zu machen.

Diese und weitere Beobachtungen führen zu dem Schluss, dass vorhandene Funktionen genutzt und von den Benutzern für einen bestimmten Zweck eingesetzt wurden.

[5] Der Begriff „teilen“ bedeutet, dass mehrere Benutzer gemeinsam Applikationen nutzen können.

Die Tatsache, dass Benutzer an ihren eigenen Arbeitsplätzen bleiben, und dennoch mit ihren Kollegen in Kollaboration treten können, wird besonders positiv hervorgehoben.

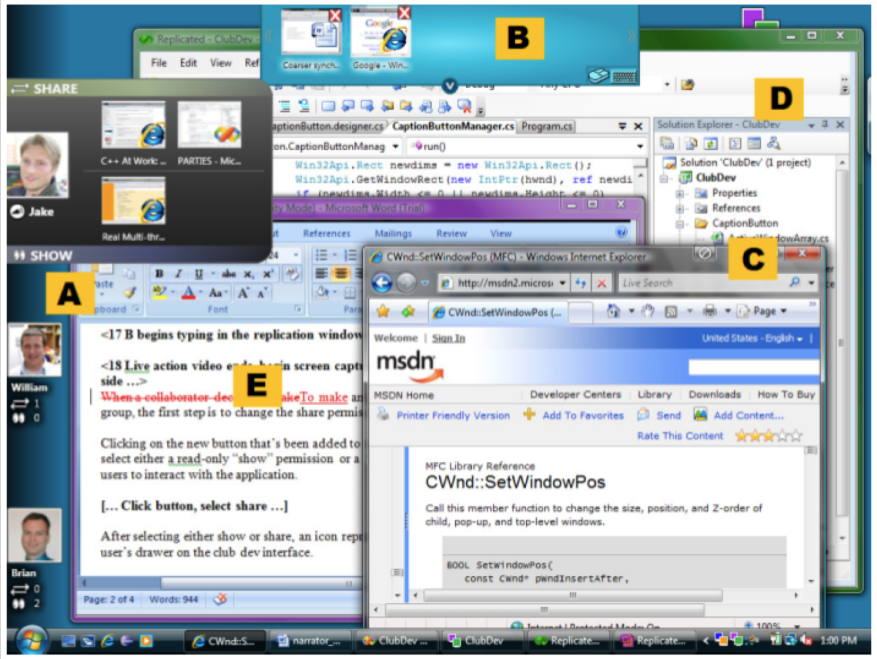


Abbildung 9: *Screenshot des Impromptu-Interface: Links die „Collaborator Bar“ (A), oben ein „Shared Screen Doc“ (B), die „Collaboration Control“ (C). Zusätzlich sieht man verschiedene Arten von Fenstern. D ist eine Fenster, dass bearbeitet werden kann, während E nur zum Betrachten, nicht zum Bearbeiten freigegeben ist [Biehl et al., 2008].*

Streitz et al. sprechen von einer Veränderung der Arbeitspraktiken [Streitz et al., 1999]. Verteilte Standorte, mobile Arbeiter oder geteilte Schreibtische sind nur der Anfang. Zusätzlich schreiben die Autoren unserer Umgebung, also Möbeln, Räumen und Wänden einen hohen Aufforderungscharakter zu. Es gilt entsprechend den Aufforderungscharakter von Architektur mit Information anzureichern.

Ausgehend von einem grundlegenden technischen Framework, das den Prinzipien der „Augmented Reality“ [6] und des „Ubiquitous Computing“ [7] folgt, und mit dem Ziel kreatives Arbeit zu unterstützen, führten sie eine Benutzerstudie bei fünf Unternehmen durch, in denen sie „kreative Teams“ vorfanden. Die vorhandene schlechte Infrastruktur entsprach nicht den Erwartungen der Autoren, wohingegen die große Bereitschaft für neue Möglichkeiten ein positives Ergebnis war. Ohne die Studien näher zu beschreiben, erläutern die Autoren, dass die Teilnehmer eine hohe Flexibilität der physischen Umgebung und der Informationsumgebung erwarten.

[6] Augmented Reality bezeichnet einen Begriff für eine neue Art von Benutzerschnittstellen, die sehr stark in Architektur und die Umgebung integriert ist [Buxton, 1997].

[7] Ubiquitous Computing beschreibt eine Art von Benutzerschnittstellen, die sich nahtlos in das Leben integriert und somit aus dem Bewusstsein „verschwindet“ [Weiser, 1999].

Ausgehend von den Studien und den allgemeinen Annahmen der Augmented Reality und des Ubiquitous Computing erläutern sie „I-Land“. I-Land ist ein Raum mit an Technologie angereicherten Gegenständen, der sogenannten „Roomware“. „Roomware“ sind normale Alltagsgegenstände, die mit Informationstechnologie versehen sind. Derzeitige Komponenten sind der „InteracTable“, ein interaktiver Tisch, die „DynaWall“ eine interaktive Fläche und die sogenannten „CommChairs“, Stühle mit integrierter Computerunterstützung. Während vorherige Studien explizit die Rollen bestimmter Displays untersuchen, schreiben die Autoren hier den einzelnen Geräten bestimmte Aufgaben und damit Rollen zu: Dabei soll die „DynaWall“ dem Erstellen und Organisieren von Information dienen, die „CommChairs“ dienen eher Annotation von entfernten Objekten und der „InteracTable“ macht das Erstellen, Anzeigen, Diskutieren und Annotationen von Information möglich.

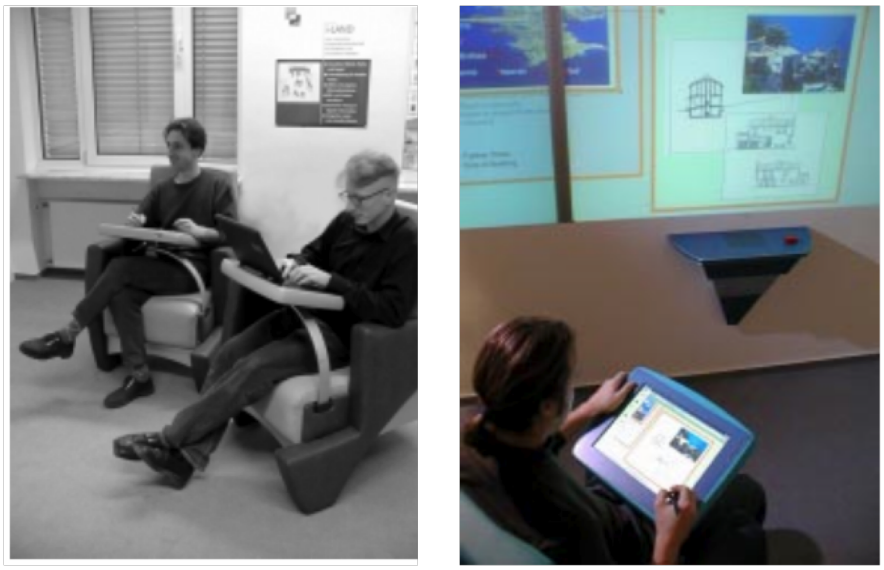


Abbildung 10: *Roomware Komponenten aus der I-Land Umgebung [Streitz et al., 1999]*

Die Autoren schließen mit dem Ausblick auf eine notwendige Evaluation und das Tracking von Personen und Geräten.

Die vorgestellten Arbeiten gruppieren sich um das Gebiet multifokaler Sichten in verschiedenen Situationen und mit unterschiedlichen Schwerpunkten. Das folgende Kapitel zeigt, wie sich der Schwerpunkt dieser Arbeit in diese verwandten Forschungsarbeiten einordnet.

2.5 Einordnung der Arbeit

Aus den vorgestellten Arbeiten geht hervor, welche Fülle von Aspekten eine Rolle spielen, wenn man sich mit der Thematik multifokaler Sichten auseinandersetzt. Zusätzlich

zu den hier vorgestellten Bereichen sind weitere Aspekte interessant. „Konfiguration der Display Umgebung“, „Anwendung im Multi-User Bereich“ oder „typische Aufgaben und Aktivitäten“ sind Anknüpfungspunkte, die sich mit ähnlichen Problemen beschäftigen, aber nicht vorgestellt wurden.

Ziel dieser Arbeit ist es, einen konzeptionellen Rahmen für die Interaktion in multifokalen Umgebungen zu schaffen. Die Arbeit soll eine Grundlage bilden für das Designen von neuen Konzepten, die multifokale Sichten unterstützen. Abbildung 11 ordnet den Gegenstand dieser Arbeit grafisch in die bisher genannten Felder ein.

Es geht darum Aktionen herauszustellen, die sowohl im Single-Display, als auch im Multi-Display Kontext ausgeführt werden. Diese Aktionen bilden dabei einen Teil der gesamten Interaktion mit Single- und Multi-Display Umgebungen.

Mit diesen Interaktionsmöglichkeiten soll der Benutzer die Möglichkeit erhalten, bestimmte Aufgaben in einer multifokalen Umgebung zu erledigen. Aufgaben bzw. Aktivitäten beschreiben mehr als nur atomare Aktionen, vielmehr setzen sie sich aus mehreren Aktionen zusammen. Möchte ein Benutzer bspw. die Aktion „Beantworten einer E-Mail“ durchführen, so muss er zunächst die empfangene E-Mail anzeigen, eine neue Nachricht öffnen und diese dann bearbeiten. Die Aufgabe besteht also aus verschiedenen Aktionen. Weitere Aufgaben wären z.B. die Suche nach Information oder das Vergleichen von Informationsobjekten.

Aufgaben und weitere Bereiche, wie *Konfiguration* verschiedener Umgebungen, die Einbeziehung *mehrerer Benutzer* oder die Unterstützung *kollaborativer Arbeit* sind nicht explizit Teil dieser Arbeit.

Im Folgenden wird zunächst der Begriff einer *multifokalen Arbeitsumgebung* näher definiert. Anschließend werden im Rahmen einer Taxonomie Aktionen beschrieben, die während der Interaktion mit einer *multifokalen Arbeitsumgebung* stattfinden. Die gesamte Taxonomie wird im Anschluss praktisch auf ein bestehendes Konzept angewendet.

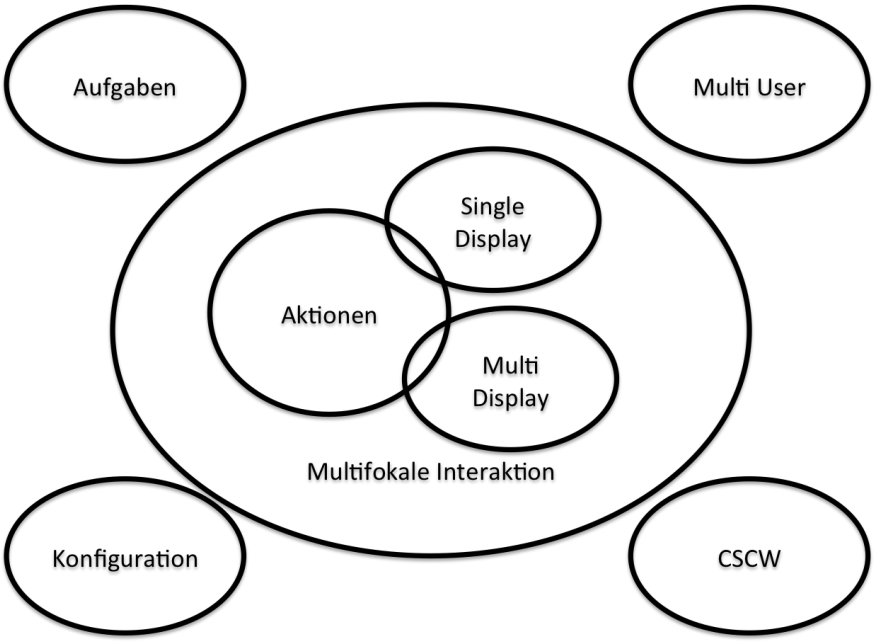


Abbildung 11: *Einordnung der Arbeit in weitere Forschungsfelder im näheren Kontext der Arbeit.*

3 Aktionen im Kontext - eine Taxonomie

Die beschriebene vielfältige Entwicklung im Bereich der Displaytechnologie geht einher mit einer zu geringen Analyse der Benutzerinteressen und -anforderungen. Es fehlt ein theoretischer Bezugsrahmen, der eine Einordnung und Systematisierung der Aktionen eines Benutzers im Kontext einer *multifokalen Arbeitsumgebung* erlaubt. Im Rahmen dieser Arbeit wurde deshalb eine Taxonomie entwickelt, die eine systematische Einordnung grundlegender Aktionen der Nutzer in einer *multifokalen Arbeitsumgebung* ermöglicht. Mit Hilfe dieser Taxonomie lassen sich generische Aktionen auf verschiedene Kontextsituationen abbilden.

Die Taxonomie beschränkt sich dabei nicht auf Multi-Display Umgebungen. Das grundlegende Verständnis des menschlichen Arbeitsgedächtnisses und die wichtige Rolle des visuellen Arbeitsgedächtnisses erfordern eine allgemeine Unterstützung für multifokales Arbeiten [Plumlee, 2002]. Der Benutzer muss in verschiedenen Kontexten kognitiv unterstützt werden. Dabei darf die Entlastung durch mehrere mögliche Fokusbereiche nicht durch den Mehraufwand an Management und Koordination der Sichten aufgehoben werden.

Im Anschluss wird zunächst der Begriff einer *multifokalen Arbeitsumgebung* definiert, bevor die Dimensionen der Taxonomie erklärt werden. Anschließend folgt die praktische Anwendung auf ein bestehendes Konzept.

3.1 Definition einer multifokalen Arbeitsumgebung

> „In any of these physical configurations, there can be a single user, collaborative user groups, or multiple users in non-collaborative situations.“ [Hutchings et al., 2005]

Naacenta et al. definieren eine Multi-Display Environment als ein „interaktives Computersystem mit zwei oder mehr Displays, die sich im selben Raum befinden“ [Naacenta et al., 2009]. Darin enthalten sind sowohl Umgebungen, die nur ein System mit mehreren Displays beinhalten, wie auch Umgebungen, die mehrere Systeme enthalten, verbunden durch ein Netzwerk.

Für Biehl et al. „besteht eine Multi-Display Umgebung aus benachbarten, persönlichen Geräten (wie z.B. Laptops) und geteilten Geräten (wie z.B. großen Displays), die vernetzt sind, um einen integrierten, virtuellen Arbeitsplatz zu bilden“ [Biehl et al., 2008].

Während sich diese Definitionen eher am Hardware-Setting orientieren, zielen weitere Definitionen auf die grundlegende Thematik von mehreren Fokussen ab.

Elmqvist et al. entwickeln bspw. eine neue Visualisierung für „mehrere fokussierte Regionen“ [Elmqvist et al., 2008].

Shoemaker und Gutwin sprechen von „multi-point interaction tasks“ [Shoemaker and Gutwin, 2007] und demnach von Interaktionsaufgaben, die sich auf zwei oder mehr Punkte beziehen.

Im Rahmen dieser Arbeit wird von einer *multifokalen Arbeitsumgebung* gesprochen. Diese Bezeichnung impliziert einige der eben aufgeführten Merkmale. In einer *multifokalen Arbeitsumgebung* hat der Benutzer die Möglichkeit mehrere fokussierte Bereiche in einem Interface festzulegen. Es spielt dabei grundlegend keine Rolle, ob mehrere Fokuspunkte in einem Display festgelegt sind oder ob diese auf verschiedene Displays verteilt werden. In dieser Definition sind sowohl Arbeitsplätze enthalten, die nur mit einem Display ausgestattet sind, ebenso Arbeitsplätze, die über mehr als ein Display verfügen. Beispiele für *multifokale Arbeitsumgebungen* zeigt Abbildung 12. Benutzer in einer *multifokalen Arbeitsumgebung* arbeiten sowohl individuell, als auch kollaborativ, entweder an privaten Geräten oder in einer MDE.

Darüber hinaus ist eine Integration eines eigenständigen Systems in ein Setting, das mehrere Systeme miteinander verbindet, möglich und eröffnet dem Benutzer die Gelegenheit, sich schnell in eine multifokale Arbeitsumgebung zu integrieren.

Diese Arbeitsumgebung setzt die Möglichkeit voraus, von unterschiedlichen Systemen, auf einen gemeinsamen Bestand an Daten zugreifen zu können. Unter dieser Annahme ist ein schneller Übergang zwischen selbstständiger Arbeit und Kollaboration möglich.

(a) *Interaktion am Tabletop, zwei periphere Displays [Forlines et al., 2006]*

(b) *Ein Team von Softwareentwicklern arbeitet kollaborativ in einem Raum [Biehl et al., 2008]*

Abbildung 12: *Beispiele multifokaler Arbeitsumgebungen*

3.2 Dimensionen der Taxonomie

> „Although MDEs are clearly one of the directions in which computing environments are moving, we still know little about how to design and choose interaction techniques that enable fundamental tasks in these environments.“ [Naacenta et al., 2009]

Die Autoren dieser Aussage bemängeln, dass trotz der Tatsache, dass Multi-Display Umgebungen sicher ein Weg sind, die zukünftige Computer-Umgebungen gehen werden, immer noch zu wenig darüber bekannt ist, wie man Interaktionstechniken gestalten und auswählen kann, die grundlegende Aufgaben in diesen Umgebungen ermöglichen.

Mit Hilfe der in dieser Arbeit entstandenen Taxonomie werden Aktionen beschrieben, welche in *multifokalen Arbeitsumgebungen* häufig ausgeführt werden. Die Taxonomie ordnet generische *Aktionen* jeweils einem *Kontext* zu. Ausgehend von der Definition einer *multifokalen Arbeitsumgebung*, wird im Rahmen dieser Arbeit zwischen „Single-Display Kontext“ und „Multi-Display Kontext“ unterschieden.

Die Taxonomie gibt dem Designer *multifokaler Arbeitsumgebungen* die Möglichkeit, den Aufgaben, die er unterstützen möchte, generische Aktionen zuzuordnen. Darüber hinaus wird durch die Taxonomie deutlich, wo die Unterschiede zwischen Single-Display und Multi-Display Umgebungen im Bezug auf die grundlegende Interaktion liegen.

Es gilt dabei zu beachten, dass die Interaktion mit Anwendungsfenstern aus der WIMP[8]-Welt auf der einen Seite und mit Objekten in einem Objekt-orientierten User Interface[9] (OOUI) auf der anderen Seite Unterschiede aufweisen. Die Dimensionen der Taxo-

[8] Windows, Icons, Menus, Pointing Device (WIMP)

[9] Ein Objekt-orientiertes User Interface nimmt Abstand von herrkömmlichen Applikationsfenster und setzt das Objekt an sich in den Blickpunkt der Betrachtung.

nomie können auf beide Interface-Arten angewendet werden. Die Ausprägungen der einzelnen Felder ist später jedoch zu unterscheiden.

Abbildung 13 zeigt das Schema der Taxonomie. Die Taxonomie bildet eine 3x2 Matrix, mit den beiden Dimensionen *Kontext* und *Aktionen*.

Aktionen / Kontext	Manipulation	Anzeige	Verschieben
Single-Display Kontext			
Multi-Display Kontext			

Abbildung 13: *Taxonomie der Aktionen in einer multifokalen Arbeitsumgebung.*

Im Folgenden werden zunächst die Dimensionen der Taxonomie allgemein erläutert. Anschließend werden die Felder der Taxonomie in Anwendung auf ZOIL näher spezifiziert.

3.2.1 Kontext

Im Rahmen dieser Arbeit wird zwischen zwei Kontexten, in denen multifokale Lösungen entwickelt werden können, unterschieden. Eine *multifokale Arbeitsumgebung* beinhaltet sowohl Single-Display, als auch Multi-Display Umgebungen. Für Naacenta et al. sind Single-Display und Multi-Display Umgebungen „fundamental“ unterschiedlich und es bedarf einer besonderen Betrachtung hinsichtlich des Interaktionsdesigns [Naacenta et al., 2009].

Dies bedeutet, dass beide Kontexte in der Taxonomie auf ihre Interaktionsmerkmale untersucht werden müssen. Aus diesem Grund unterscheidet die Taxonomie zwischen *Single-Display Kontext* und *Multi-Display Kontext*.

Die Art und Beschaffenheit der einzelnen Displays spielen in dieser Betrachtung keine Rolle. Ein Display kann sowohl ein normales Monitor-Display, wie auch ein Laptop-Display oder gar ein Tabletop sein.

In den folgenden Beschreibungen werden kurz die Eigenschaften der verschiedenen Kontexte herausgearbeitet.

Single-Display Kontext Im Single-Display Kontext werden Systeme betrachtet, die über ein einziges Display verfügen und mit denen in der Regel ein Benutzer interagiert. Gerade bei Verwendung von nur einem Display, also nur einer Hardware-Komponente, ist es notwendig, dem Benutzer die Möglichkeit zu geben, dennoch mehrere fokussierte Bereiche festzulegen. Speziell in einem Zoomable User Interface, wie ZOIL, bedarf es der kognitiven Unterstützung des Benutzers [Plumlee, 2002]. Laut Plumlee kann ein Benutzer nicht mehr als drei Objekte in seinem *Visual Work Memory* speichern und es bedarf daher, bspw. für vergleichende Aufgaben einer Option, mehrere Bereiche in einem Interface zu fokussieren, und damit unabhängig vom *Visual Work Memory* miteinander vergleichen zu können. Ziel muss es demnach sein, gerade für den Single-Display Kontext multifokale Lösungen zu entwickeln.

Multi-Display Kontext Im Multi-Display Kontext werden Systeme betrachtet die mehrere Displays beinhalten. Diese Displays befinden sich oft in räumlicher Nähe zueinander. Diese Displays können an verschiedenen Systemen angeschlossen sein, welche über ein Netzwerk verbunden sind. Es entsteht ein gemeinsamer, logischer Arbeitsbereich [Naacenta et al., 2009, Biehl et al., 2008]. Laut Naacenta et al. haben solche Umgebungen, trotz der damit verbundenen Komplexität, das Potenzial, die Art und Weise unserer Arbeitens zu verändern [Naacenta et al., 2009].

Diese Weiterentwicklung in der Benutzung mehrerer Systeme und somit auch mehrerer Displays, erfordert die Betrachtung des Benutzers in einem Multi-Display Kontext.

Diese Arbeit geht von dem Verständnis aus, dass ein Benutzer in einem Multi-Display Kontext nur mit einem Display **direkt** interagiert.

Gruding unterscheidet zwischen Monitoren mit „primären Aufgaben“ und „sekundären Aufgaben“ [Grudin, 2001].Es liegt nahe, dass ein Großteil der Interaktion mit den „primären Aufgaben“ verbunden ist.

Naacenta unterscheidet hingegen bei einer Multi-Display Umgebung explizit zwischen einem „Current Display“ und einem „Remote Display“. Die Interaktion findet am „Current Display“ statt, kann dabei aber Auswirkungen auf das „Remote Display“ haben.

Die Betrachtung der Aktionen in einer Multi-Display Umgebung im folgenden Kapitel bezieht sich auf dieses Verständnis und schließt sich der Unterscheidung zwischen *Current Display* und *Remote Display* an.

3.2.2 Aktionen

In der zweiten Dimension der Taxonomie werden generische Aktionen identifiziert, die innerhalb einer *multifokalen Arbeitsumgebung* stattfinden.

Im Rahmen dieser Arbeit werden drei Aktionen in den Vordergrund gestellt: die *Manipulation*, das *Anzeigen* und das *Verschieben* von Objekten. Diese drei Aktionen bilden eine Schnittmenge aus einer Reihe von Aktionen, die in anderen Arbeiten veröffentlicht wurden.

Im Folgenden werden diese kurz genannt und den Aktionen zugeordnet.

Biehl et al. sprechen von der Möglichkeit, Informationsartefakte auf verschiedenen geteilten Displays zu platzieren (*Anzeige, Verschieben*) und gemeinsam Information erstellen und bearbeiten (*Manipulation*) zu können [Biehl et al., 2008].

Naacenta et al. beschreiben verschiedene spezifische Aktionen für Multi-Display Umgebungen [Naacenta et al., 2009]. Das Hauptaugenmerk liegt dabei auf dem Bewegen von Objekten zwischen verschiedenen Displays (*Verschieben*). Darüber hinaus sehen sie weitere Aktionen, wie ad-hoc Konfiguration verteilter Räume, oder das Kopieren von Objekten auf andere Displays (*Anzeige*).

Dearman et al. beschreiben das Problem, dass Informationsobjekte oft an einzelne Geräte gebunden sind. Das macht das Managen und Verteilen von Information (*Anzeige, Verschieben*) notwendig [Dearman and Pierce, 2008].

Streitz et al. sprechen davon, dass man seit Beginn der Informationstechnologie dazu übergegangen ist, Information zu erstellen, zu speichern und zu kommunizieren (*Anzeige, Verschieben*) [Streitz et al., 1999]. Darüber hinaus erwähnen sie die Problematik, dass die limitierte Displayoberfläche oft in einem komplexen Handling von Fenstern endet (*Anzeige*).

Perry und O'Hara benennen in ihren Design-Faktoren notwendige Unterstützung sowohl für „within-display movement", als auch für „inter-display movement" (*Verschieben*) [Perry and O'Hara, 2003].

Hutchings et al. leiten aus ihren Beobachtungen über Distributed Display Environment (DDE)[10] zwei wichtige Aspekte ab [Hutchings et al., 2005]. Zum einen gewinnt die Anzeige von Information auf nicht aktiven Displayregionen mehr an Bedeutung (*Anzeige*). Als zweiten Punkt ihrer Beobachtungen beschreiben sie die Notwendigkeit, die Interaktion mit DDEs zu verbessern und diese nicht nur als Informationsdisplay zu verwenden (*Manipulation, Verschieben*).

Mit den drei Aktionstypen *Manipulation*, *Anzeige* und *Verschieben* besteht eine Grundmenge generischer Aktionen, die sich aus verschiedenen Beobachtungen, in verschiedenen Umgebungen ableitet.

Die vorgestellte Taxonomie bietet einen konzeptionellen Rahmen für das Entwickeln neuer Benutzerschnittstellen in *multifokalen Arbeitsumgebungen*. Mit Hilfe der Zuordnung generischer Aufgaben zu einem Kontext erhält der Designer eine Systematik, die

[10] Distributed Display Environments werden im Rahmen dieser Arbeit mit Multi-Display Environments gleich gesetzt.

sowohl die Analyse bestehender, wie auch die Entwicklung neuer Schnittstellen erleichtert und strukturiert.

Im Folgenden wird die Taxonomie auf das bestehende ZOIL-Konzept angewendet.

3.3 Die Taxonomie im Kontext von ZOIL

Wie bereits beschrieben, ist die Taxonomie eine allgemeine Grundlage zur Kategorisierung von Aktionen in *multifokalen Arbeitsumgebungen*. Im Folgenden wird die Taxonomie am Beispiel von ZOIL näher erläutert (vgl. Abbildung 14). Dabei werden die einzelnen Felder erklärt und es werden einige Beispiele genannt, die in die jeweilige Kategorie eingeordnet werden können.

Aktionen / Kontext	Manipulation	Anzeige	Verschieben
Single-Display Kontext	Mehrere Objekte können gleichzeitig betrachtet bzw. bearbeitet werden		Objekte werden zwischen Fokusbereichen verschoben
Multi-Display Kontext	Remote-Objekt wird lokal bearbeitet	Anzeige von Remote-Objekten kann lokal gesteuert werden	Objekte werden zwischen Displays verschoben

Abbildung 14: *Taxonomie der Aktionen in einer multifokalen Arbeitsumgebung.*

Manipulation und Anzeige im Single-Display Kontext In einem OOUI sind mögliche Werkzeuge zum Bearbeiten von Objekten direkt an das Objekt gebunden. Applikationsfenster, wie sie heutzutage die Regel sind, haben dort keine Anwendung mehr. Aus diesem Grund, kann man bei einem OOUI die beiden Felder *Manipulation* und *Anzeige* zusammenfassen.

Unterstützt ein Interface diese beiden Aktionen, so kann ein Benutzer mehrere Objekte gleichzeitig betrachten und bearbeiten. Wie schon in [Schweizer, 2009b] erarbeitet, gibt es einige Anforderungen an die Visualisierung. Hervorzuheben ist im Rahmen der Anzeige von Objekten die Möglichkeit unterschiedlicher Skalierung. Die Navigation mit einzelnen Fokusbereichen muss unabhängig voneinander stattfinden können.

Ähnliches gilt für die Manipulation von Objekten. Objekte müssen unabhängig voneinander direkt manipulierbar sein. Hat der Benutzer mehr als einen Fokusbereich festgelegt, so kann er auch gleichzeitig mehrere Objekte bearbeiten.

Beispiele für Techniken, die mehrere fokussierte Bereiche auf einem Display ermöglichen sind DragMag [Ware and Lewis, 1995], Mélange [Elmqvist et al., 2008] oder Multiple Views [Roberts, 2007].

Verschieben im Single-Display Kontext Die direkte Interaktion zwischen Fokuspunkten wird durch das dritte Feld im Single-Display Kontext abgedeckt. Der Benutzer muss die Möglichkeit haben, zwischen den verschiedenen Fokusbereichen, Objekte verschieben zu können. Zu beachten ist dabei ein möglicher Unterschied in den Skalierungsgraden.

Viele Arbeiten greifen diese Idee auf, aber es gibt zu wenig konkrete Techniken, die diese Problematik abdecken. Dies hängt häufig damit zusammen, dass Techniken, die multifokale Sichten behandeln, zur Informationsvisualisierung gedacht sind und nicht für Objekt-orientiertes Arbeiten.

Während im Single-Display Kontext *Manipulation* und *Anzeige* zusammengefasst wurden, so werden sie in einem Multi-Display Kontext unterschieden. Der Grund dafür liegt in der Tatsache, dass man Objekte an einem *Remote Display* betrachten kann, ohne die Möglichkeit zu besitzen, sie zu manipulieren. Sitzt der Benutzer bspw. am Schreibtisch und betrachtet ein peripheres Display, so hat er nicht immer die Option, Objekte zu manipulieren, auch wenn er sie betrachten kann.

Manipulation im Multi-Display Kontext Die Manipulation eines Objekts in einem Multi-Display Kontext umfasst die Aktion, dass ein Objekt, das sich auf einem *Remote Display* befindet, lokal, am *Current Display* bearbeitet werden kann. Um diese Aktion auszuführen, hat man grundsätzlich zwei Möglichkeiten. Entweder man erfasst die lokale Interaktion und leitet diese an das *Remote Display* weiter oder man ermöglicht eine Kopie des entfernten Objekts auf dem *Current Display* und bearbeitet es temporär lokal.

Ein Beispiel für die erste Technik findet sich in IMPROMPTU [Biehl et al., 2008]. Dort können Benutzer ihre lokalen Eingaben an ein „shared display“ weiterleiten und so

entfernte Objekte manipulieren. Die I-Land Umgebung dagegen bietet mit den „CommChairs“ eine Möglichkeit der „remote annotation“ [Streitz et al., 1999]. Objekte an der „DynaWall“ können von den Stühlen aus annotiert werden. Dies ist ein Beispiel für eine lokale Kopie des entfernten Objekts.

Weitere Beispiele wären die RemoteDesktop - Verbindung heutiger PC-Systeme, die es ermöglichen die gesamte Anzeige eines entfernten Displays auf das lokale Display zu replizieren und zu manipulieren oder die Technik der RadarViews, die auch eine Manipulation entfernter Objekte ermöglicht [Reetz et al., 2006].

Anzeige im Multi-Display Kontext Die Kategorie „Anzeige im Multi-Display Kontext“ umfasst die Möglichkeit, dass der Benutzer ein Objekt, auf dass er lokal Zugriff hat, auf einem *Remote Display* anzeigen kann. Besonders wenn sich mehrere Benutzer im Raum befinden, bietet dies eine Möglichkeit, ein Objekt von einem lokalen System schnell auf einem anderen Display anzuzeigen, bspw. auf einem großen, hochauflösendem Display, das für alle Personen einsehbar ist.

Das IMPROMPTU-Interface bietet hierfür die Möglichkeit Applikationsfenster in ein „Shared Sreen Doc“ zu legen, welches die Anzeige auf dem zugeordneten Display ermöglicht [Biehl et al., 2008]. Forlines et al. integrieren in ihrem System eine Möglichkeit die Anzeige eines Wall-Displays von einem Tabletop aus zu steuern [Forlines et al., 2006].

Verschieben im Multi-Display Kontext Naacenta et al. bezeichnen das Verschieben von Objekten innerhalb einer Multi-Display Umgebung als „Kernfunktionalität“, welche eine fließende Interaktion mit verschiedenen Displays erlaubt [Naacenta et al., 2009].

Im Kontext von ZOIL geht es darum, Informationsobjekte von einem Display auf ein weiteres zu übertragen. Dabei sind unterschiedlichste Techniken denkbar, bspw. Gesteninteraktion oder physikalische Objekte.

Verschiedene Untersuchungen haben gezeigt, dass gerade das nahtlose Verschieben von Objekten in einer Multi-Display Umgebung zu den größten Problemen gehört [Dearman and Pierce, 2008, Naacenta et al., 2009, Biehl et al., 2008]. Zu den Techniken, die diese Form der Aktion unterstützen, gehören das Hyperdragging (siehe 2.4), das Versenden von E-Mails, Drag-and-Pop [Baudisch et al., 2003] oder Pick-and-Drop [Rekimoto, 1997].

In diesem Kapitel wurde eine Taxonomie vorgestellt, die generische Aktionen und verschiedene Kontextsituationen einander zuordnet. Mit Hilfe der Taxonomie wurde ein bestehendes Konzept untersucht und die Kategorien der Taxonomie näher spezifiziert. Diese Analyse zeigt nun, an welchen Stellen das Konzept durch neue Ideen erweitert werden muss, um den Einsatz in *multifokalen Arbeitsumgebungen* zu ermöglichen. Das

folgende Kapitel erläutert verschiedene Konzepte, die im Rahmen dieser Arbeit entstanden sind.

4 Lösungskonzepte

> „the same technique could prove a perfect choice or a usability disaster, depending on the application domain or the task“ [Naacenta et al., 2009]

Die vorausgegangene Analyse einer *multifokalen Arbeitsumgebung* hat gezeigt, dass der Benutzer sowohl in einem Single-Display Kontext, wie auch in einem Multi-Display Kontext bei der Ausführung generischer Aktionen unterstützt werden muss.

Im Hinblick auf das ZOIL-Paradigma, das auf einem Zoomable User Interface (ZUI) basiert, war es wichtig, dem Benutzer diese Aktionen zu erleichtern. Nach Bederson et al. gehört die Unterstützung mehrerer Sichten auf die Informationslandschaft zu den Grundprinzipien in einem ZUI [Bederson et al., 2000].

Im Rahmen dieser Arbeit wurden anhand der Taxonomie Konzepte erarbeitet, die den Benutzer bei der Arbeit in einer *multifokalen Arbeitsumgebung* kognitiv unterstützen. Dabei lag der Schwerpunkt auf einem Umgebungssetting. Die *mulitfokale Arbeitsumgebung* bestand dabei aus einem Tabletop und zwei peripheren Wall-Displays in einiger Entfernung (vgl. Abbildung 15). Die beschriebenen Konzepte wurden hauptsächlich vor diesem Hintergrund konzipiert, auch wenn generell eine Anwendung in weiteren Settings möglich ist.

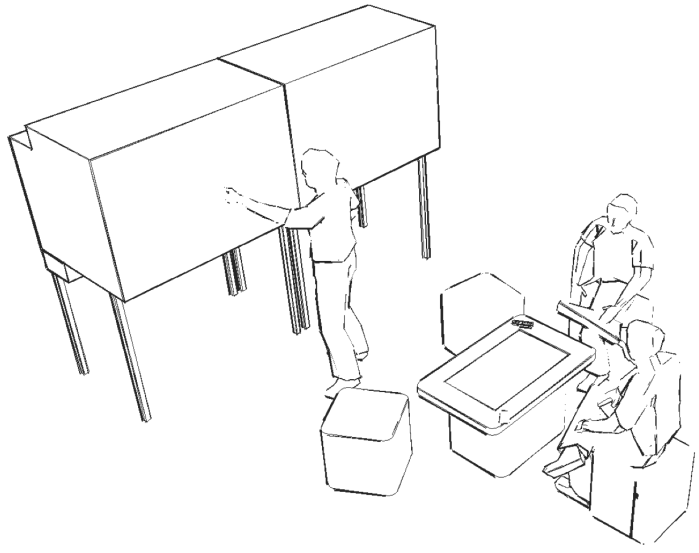


Abbildung 15: *Taxonomie der Aktionen in einer multifokalen Arbeitsumgebung.*

Die Konzepte werden im Folgenden konzeptionell beschrieben. Es wird die grundlegende Idee erläutert und auf einige Usability-Aspekte eingegangen. Eine detaillierte Beschreibung der technischen Realisation der einzelnen Konzepte ist der Projekt-Dokumentation dieser Arbeit zu entnehmen [Schweizer, 2009a].

Die fehlende Umsetzung multifokaler Sichten im ZOIL Framework [Engl, 2008] wird durch die Integration dieser Konzepte in das Framework aufgehoben. In Kapitel 5 wird detailliert erläutert, welche technischen Grundlagen gelegt wurden und wie die behandelten Konzepte abschließend in das Framework integriert sind.

4.1 Manipulation und Anzeige im Single-Display Kontext

4.1.1 Split Screen

> „Split-screen is perhaps the most common and most straightforward approach to multi-focus tasks.“ [Elmqvist et al., 2008]

Um dem Benutzer in einem Single-Display Kontext die Manipulation und Anzeige mehrere Objekte zu ermöglichen, wurde ein Konzept für einen *Split Screen* erarbeitet.

Die nachfolgende Beschreibung erläutert, wie Navigation, Visualisierung und Interaktion durch das implementierte Konzept gelöst werden.

Abbildung 16 zeigt die Ansicht eines *Split Screen*. Wir sehen zwei *Ausschnitte* einer Landschaft, die horizontal voneinander getrennt sind. Es ist zu sehen, dass die fokussierten Bereiche unabhängig voneinander, unterschiedliche Skalierungsgrade aufweisen und verschiedene Bereiche der Landschaft anzeigen. In der Abbildung sind die beiden *Ausschnitte* farblich voneinander getrennt. Die *Ausschnitte* bieten jeweils ihre eigene Sicht auf die gleiche Datenlandschaft. Sie unterscheiden sich lediglich in Skalierungsgrad und Position. Im Folgenden wird diese Kombination aus **x,y**-Koordinate und **Breite** und **Höhe** des angezeigten Bereichs als *Viewport* bezeichnet. Ein fokussierter Bereich wird als *Ausschnitt* bezeichnet. Somit ergibt sich eine grundlegende Unterscheidung zwischen einer *Informationslandschaft* (die zugrunde liegende Informationslandschaft mit allen Datenobjekten), dem *Viewport* (Koordinaten und Größe des angezeigten Bereichs) und dem *Ausschnitt* als Begriff für eine Sicht auf die *Informationslandschaft*, der einen bestimmten *Viewport* anzeigt.

Navigation Die Interaktion zwischen verschiedenen Eingabegräten[11] und einem *Ausschnitt* sind direkt aneinander gekoppelt. D.h. jeder *Ausschnitt* empfängt seine eigenen Input-Events. Eine solche direkte Verbindung zwischen Objekt und Interaktion ermöglicht eine flexible Handhabung der gewünschten Interaktion.

Die Bindung von Interaktion und *Ausschnitt* lässt beim *Split Screen* eine unabhängige Navigation der verschiedenen Bereiche zu. Jeder *Ausschnitt* empfängt den Input, der

[11] Im Rahmen dieser Arbeit werden verschiedenste Formen von Eingabegeräte unter diesem Begriff zusammengefasst. Dazu gehören: Maus, Tastatur, Laserpointer, Wiimote Control und Multitouchoberflächen

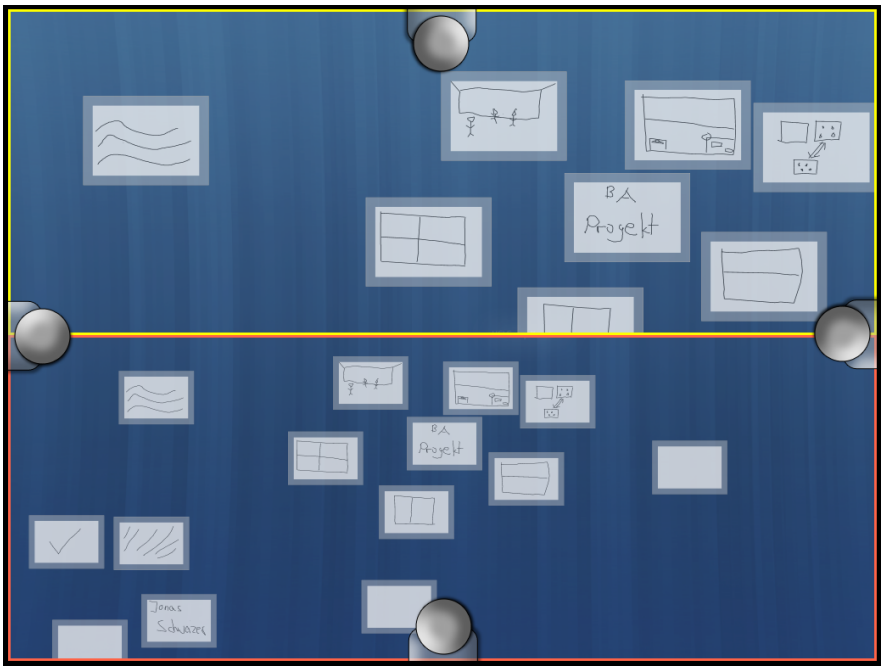


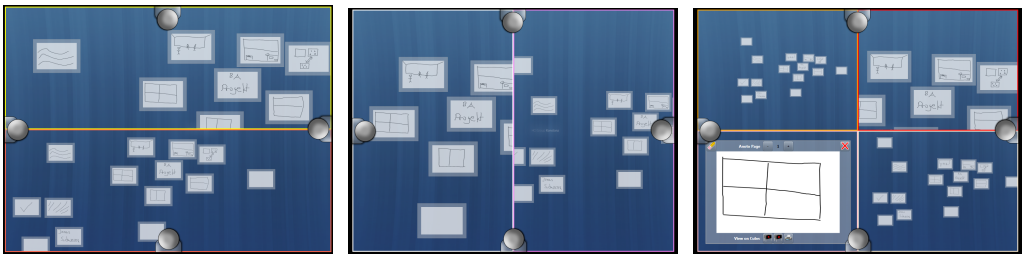
Abbildung 16: *Ansicht auf einen* Split Screen *mit zwei Ausschnitten. Die beiden Ausschnitte sind horizontal voneinander getrennt.*

auf ihm passiert und kann diesen individuell verarbeiten. Dadurch muss der Benutzer keine neue Navigationstechnik erlernen und navigiert in der gewohnten Art und Weise.

Ausgehend von einer einzigen Sicht auf eine *Informationslandschaft* muss der Benutzer die Möglichkeit erhalten verschiedene Bereiche zu fokussieren, also neue *Ausschnitte* festzulegen. Möchte der Benutzer die Zahl der *Ausschnitte* erhöhen, so kann er einen *Split* auslösen. Dabei wird die Anzahl der *Ausschnitte* abhängig von Orientierung und bisherigem Layout erhöht. Die gegensätzliche Aktion ist ein *Merge*. Die Anzahl der *Ausschnitte* verringert sich. Zunächst werden die beiden Operationen beschrieben, bevor die Interaktionsmöglichkeiten für den Benutzer aufgezeigt werden, welche die beiden Operationen auslösen.

Split und Merge Löst der Benutzer einen Split aus, so wird der *Viewport* des aktuellen *Ausschnitts* zunächst nicht verändert. D.h. der Zoomfaktor und die aktuelle Position des *Ausschnitts* bleiben erhalten. Der betrachtete *Ausschnitt* wird nur neu *gelayoutet*. Abbildung 17 zeigt 3 verschiedene *Layouts*, die derzeit möglich sind.

Durch Erhaltung des *Viewports*, muss sich der Benutzer zunächst einmal nicht neu orientieren und kann die gewünschte Aktion direkt auf den vorgesehenen Objekten ausführen. Somit geschieht bei einem *Split* nichts weiter, als dass eine neues *Layout* erstellt wird, der *Viewport* dabei erhalten bleibt und der Benutzer im Anschluss mit neuen fokussierten Bereichen interagieren kann.

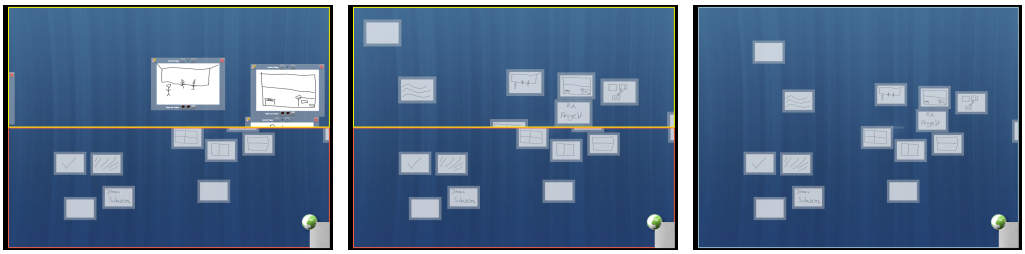


(a) *Horizontales Layout mit zwei Ausschnitten*

(b) *Vertikales Layout mit zwei Ausschnitten*

(c) *Layout mit vier Ausschnitten in einer Matrix*

Abbildung 17: *Verschiedene Layouts im* Split Screen



(a) *Zwei Ausschnitte mit unterschiedlichen Skalierungsgraden*

(b) *Merge-Operation wurde ausgelöst und der obere Ausschnitt wurde an die Zoomstufe des unteren angepasst*

(c) *Farbliche Rahmen entfernt, Anzahl der Ausschnitte auf 1 reduziert*

Abbildung 18: *Storyboard der* Merge-*Operation.*

Führt der Benutzer einen *Merge* aus, so muss ein weiterer Aspekt beachtet werden. Es besteht in der Regel die Situation, dass die beiden *Ausschnitte* nicht den gleichen Skalierungsgrad haben. Die Veränderung des *Layouts* führt bei einem *Merge* zu einer verringerten Zahl an *Ausschnitten*. Der Benutzer darf, ähnlich wie bei einem *Split*, die Orientierung nicht verlieren.

Der Ablauf einer *Merge*-Operation verläuft aus diesem Grund nach einem vorgegebenen Muster (vgl. Abbildung 18). Unter der Annahme, dass mehr Kontext zu mehr Orientierung führt, wird der *Ausschnitt* beibehalten, der einen größeren *Viewport* (und somit auch mehr Kontext) darstellt. Mit anderen Worten: Es „gewinnt“ der *Ausschnitt*, welcher weniger in die Landschaft gezoomt hat. Der „Verlierer-Ausschnitt“ wird animiert auf das Niveau des anderen *Ausschnitts* angeglichen. Anschließend wird die Zahl der *Ausschnitte* reduziert und die farblichen Rahmen werden angepasst.

Der Ablauf einer *Merge*-Operation kann dabei immer auf zwei *Ausschnitte* reduziert werden, da selbst bei einem *Merge* von vier *Ausschnitten* jeweils zwei *Ausschnitte* zueinander angeglichen werden, um den kognitiven Aufwand gering zu halten.

Wie bereits kurz erwähnt, werden *Split* und *Merge* situationsabhängig ausgeführt. Für die Benutzerinteraktion wird dabei nicht zwischen den beiden Operationen unterschie-

den. Der Benutzer führt lediglich eine bestimmte Geste[12] aus und daraufhin passiert ein *Split* oder *Merge*. Zunächst scheint dieses Verhalten zu wenig vorhersagbar zu sein. Die Erwartungskonformität wird aber aus einer anderen Betrachtungsweise klarer. Der Benutzer kann mit der selben Aktion, mit der er ein Verhalten ausgelöst hat (*Split*), dieses auch wieder rückgängig machen (*Merge*). Nutzer müssen keine verschiedenen Gesten lernen.

Abbildung 19 zeigt die verschiedenen *Layouts* und Layout-Übergänge. Je nach *Layout* wird entschieden, welche konkrete Operation durchgeführt wird.

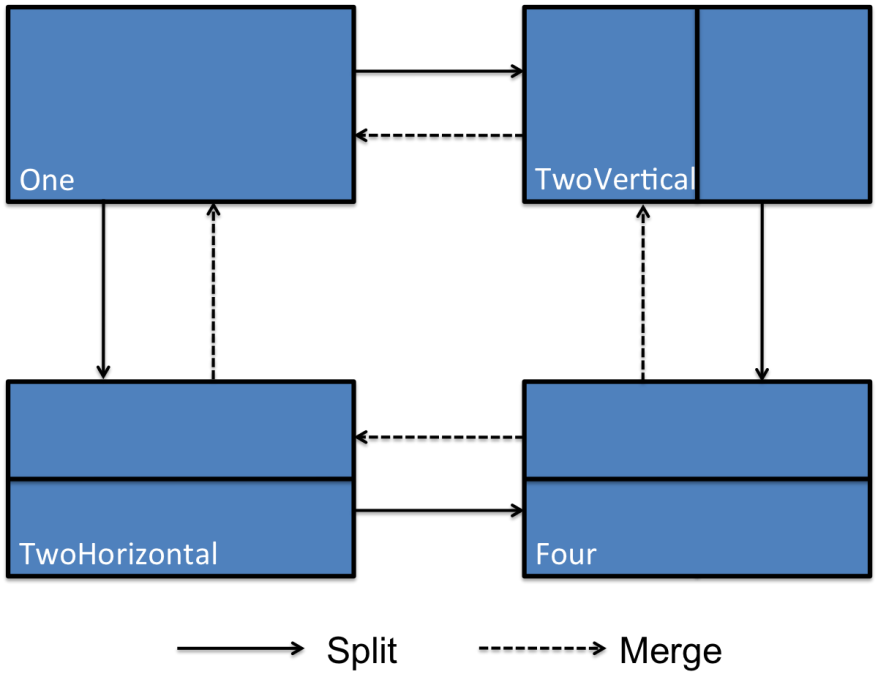


Abbildung 19: *Situationsabhängige Operation. Je nach Layout wird eine Interaktionsgeste als* Split *oder* Merge *interpretiert.*

Interaktion Wie oben erläutert wurde, muss dem Benutzer eine Interaktionsgeste zur Verfügung gestellt werden. Diese Geste kann prinzipiell von verschiedenen Eingabegeräten ausgeführt werden. Im Rahmen dieser Arbeit wurden zwei Gesten umgesetzt:

Tastaturgeste Die Tastaturgeste wurde durch eine Tastenkombination realisiert. Somit wird die Interaktion an einem klassischen Desktop-PC unterstützt. Mit Auslösen der Kombination **STRG + C** kann der Benutzer eine horizontale Aktion auslösen. Die vertikale Kombination lautet **STRG + V**.

[12] Der Begriff Geste beinhaltet in diesem Zusammenhang nicht nur Handgesten, sondern auch Tastatur- oder Mausgesten.

Multitouchgeste Um dem Benutzer eine intuitive Geste anzubieten, wurde auf dem Surface[13] eine Handgeste implementiert.

Auch hierbei wird zwischen einer vertikalen und einer horizontalen Geste unterschieden. Dabei handelt es sich nicht um eine Freihandgeste, sondern die Geste passiert auf den sogenannten *Handles* (vgl. Abbildung 20a). Die *Handles* sind Ausgangs- und Endpunkt für die Geste.

Die Umsetzung mit Hilfe der *Handles* hat zwei grundsätzliche Vorteile:

1. Der Benutzer bekommt durch farbliche Hervorhebungen ein direktes visuelles Feedback über die möglichen Interaktionen.
2. Mit Hilfe der *Handles* erhält der Benutzer eine Handlungsaufforderung (affordance), die verdeutlicht, dass überhaupt die Möglichkeit einer Interaktion besteht.

Um die Geste auszuführen, muss der Benutzer seine Hand von einem *Handle* zum einem gegenüberliegenden *Handle* ziehen. Er kann sozusagen den Bildschirm durchtrennen und damit eine Aktion auslösen.

Mit Hilfe der Taxonomie für Surface-Gesten (vgl. Kapitel 2.2) lässt sich die Geste näher kategorisieren. Die Geste wird in vier Kategorien untersucht:

Form Die Form lässt sich in die Kategorie „Statische Pose und ein Pfad“. Das bedeutet die Pose der Hand ändert sich nicht während der Bewegung und die Hand folgt einem Pfad.

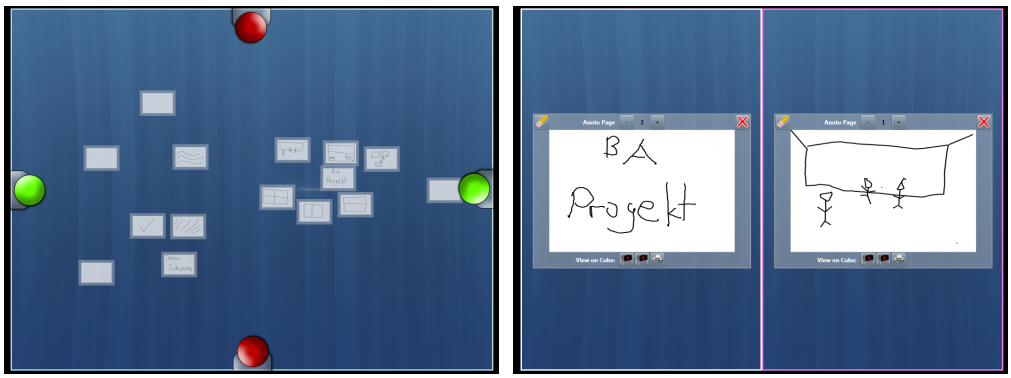
Natur Die Geste ist methaphorischer Natur. Mit der Geste ist die Metapher des Durchtrennens verbunden. Der Benutzer „teilt“ den Bildschirm.

Bindung Die Geste bindet sich an die „Welt“, d.h. sie ist abhängig vom gesamten Bildschirm und nicht nur von einem bestimmten Objekt.

Ablauf Der Ablauf ist diskret. Die Reaktion auf die Geste (der geteilte Bildschirm) erfolgt nach Abschluss der Geste.

Der *Split Screen* ermöglicht dem Benutzer mehr als nur ein Objekt anzuzeigen und bearbeiten zu können (vgl. Abbildung 20b). Damit ist eine Lösung für das erste Feld in der Taxonomie vorhanden. Um diese Aktion noch direkter und intuitiver zu gestalten, wird im folgenden Kapitel eine weitere Interaktionsmöglichkeit vorgestellt, die dem Benutzer das Fokussieren von mehr als einem Objekt ermöglicht.

[13] Bei dem sogenannten Surface handelt es sich um einen von der Firma Microsoft hergestellten Multitouch-Tisch, der im Rahmen dieser Arbeit als Eingabegerät verwendet wurde.



(a) *Darstellung der Handles für die Splitgeste an einem Multitouch-Tisch*

(b) *2 Objekte im näheren Fokus des Betrachters*

Abbildung 20: *Handles und Fokusse im Split Screen.*

4.1.2 Mehrfacher Click Zoom

Möchte der Benutzer auf einem Tabletop ein Objekt näher betrachten, so hat er die Möglichkeit einen *Click Zoom*[14] zu initiieren. Im ZOIL Framework ist dafür eine Tab-Geste[15] vorgesehen. Daraufhin wird das Objekt herangezoomt und im jeweiligen *Ausschnitt* angezeigt.

In einer Situation, in welcher der Benutzer mehrere Objekte fokussieren möchte, bietet es sich an, dass er das gleiche Verhalten auf mehr als ein Objekt anwenden kann. Der Benutzer kann gleichzeitig mehrere Objekte (derzeit maximal 4) berühren und löst mit dem Anheben eines Fingers einen *mehrfachen Click Zoom* aus. Das Konzept bietet eine weitere Methode für die Anzeige und Manipulation von mehreren Objekten im Single-Display Kontext.

Grundsätzlich hat der Interfacedesigner die Möglichkeit die Reaktion auf einen *mehrfachen Click Zoom* selbst zu bestimmen. Er muss dafür lediglich auf das entsprechende Event reagieren. D.h. die Interaktion mit dem System und die Option, mehrere Objekte zu fokussieren sind unabhängig von der Visualisierung, die sich daraus ergibt.

Visualisierung des mehrfachen Click Zoom In dieser Arbeit wurde die Visualisierung an den *Split Screen* aus Kapitel 4.1.1 angepasst.

Hat ein Benutzer mehrere Finger auf Objekten positioniert und ein Finger wird angehoben, so muss der Bildschirm in so viele Bereiche aufgeteilt werden, dass alle getappten Objekte angezeigt werden können. Anschließend müssen die Objekte sinnvoll gezoomt

[14] Ein Click Zoom beschreibt das Verhalten, dass ein Objekt nach auslösen eines „Clicks“ herangezoomt und groß angezeigt wird. Die Begrifflichkeit ist im Rahmen des ZOIL Framework entstanden. Ursprünglich sprach man von einem *Goal-directed Zoom* [Woodruff et al., 1998].

[15] Bei einer Tab-Geste berührt der Benutzer den Bildschirm an einer Stelle, nicht länger als einen Augenblick und lässt dann direkt los. Er „tappt“ also nur einmal kurz auf die Oberfläche.

werden. Es kommt dabei darauf an, dass der Benutzer trotz dieses kombinierten Aufteilens und Zoomings nicht die Orientierung verliert.

Aus diesem Grund laufen alle Zoomaktionen animiert ab, so dass der Benutzer ein kontinuierliches, visuelles Feedback erhält und dadurch in der Lage ist, den Endzustand nachvollziehen zu können und sich so besser im Raum orientieren zu können [Bederson and Boltman, 1999]. Die Ausgangssituation für einen *mehrfachen Click Zoom* ist ein Screen mit nur einem *Ausschnitt*, der eine Sicht auf die Informationslandschaft ermöglicht.

Für den Übergang von einer Ansicht zu mehreren Fokussen wurde folgender Algorithmus entwickelt (vgl. Abbildung 21):

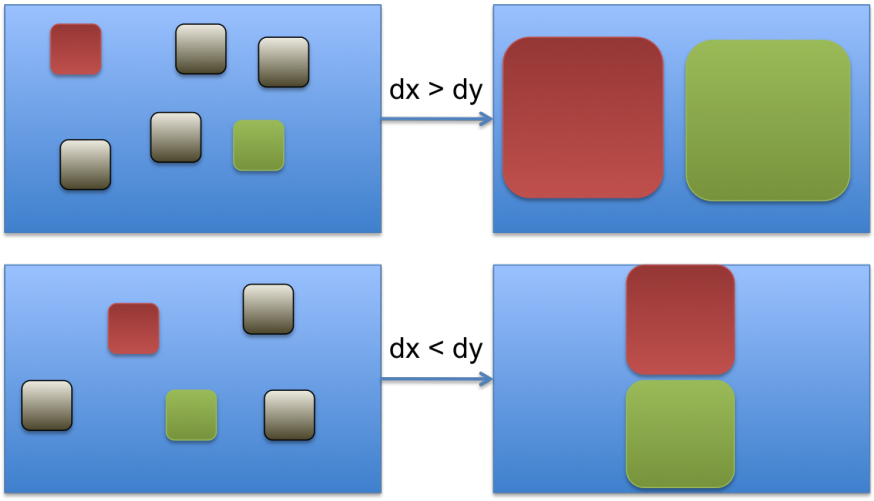
1. Wenn genau 2 Objekte getappt wurden, dann
   (a) Berechne vertikalen und horizontalen Abstand
   (b) Ist der vertikale Abstand größer, so wird der Bildschirm horizontal geteilt und das obere Objekt wird im oberen Bereich angezeigt
   (c) Ist der horizontale Abstand größer, so wird der Bildschirm vertikal geteilt und das linke Objekt wird im linken Bereich angezeigt
2. Wenn mehr als 2 Objekte getappt wurden, dann
   (a) Berechne euklidische Distanz der Objekte zum Nullpunkt
   (b) Sortiere Objekte nach Distanz
   (c) Ordne die Objekte aufsteigend den Bereichen zu
   (d) Falls 3 Objekte fokussiert wurden, so wird ein Ausschnitt auf „Home“ gezoomt

Mit Hilfe der vorgestellten Technik, kann der Benutzer direkt mehrere Objekte fokussieren, in dem er sie mit einem *mehrfachen Click Zoom* auswählt. Die Visualisierung ist in diesem Fall der *Split Screen* aus Kapitel 4.1.1. Wie der Benutzer nun Objekte zwischen den Fokussen bewegen kann, wird im folgenden Kapitel näher erläutert.

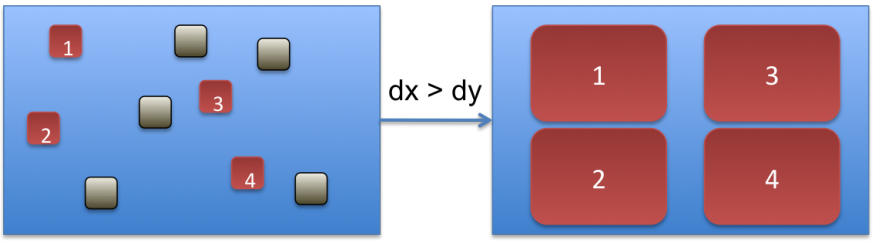
4.2 Objekte verschieben im Single-Display Kontext - Drag&Drop

Bisher gibt es die Möglichkeit Objekte in einem *Ausschnitt* zu verschieben. Das bedeutet, die Objekte werden mit Hilfe der Maus oder dem Finger auf dem Multitouch von einer Position, durch Bewegen auf eine andere Position gelegt. Dieses Prinzip des Draggens ist in heutigen PC-Anwendungen sehr geläufig und ermöglicht die individuelle Anordnung der Objekte auf der Informationslandschaft.

Im Rahmen dieser Arbeit wurde die Technik erweitert. Gerade während der Nutzung einer *Split Screen* Ansicht ist es notwendig, Objekte von einem *Ausschnitt* auf den anderen bewegen zu können.



(a) *Position abhängig von der relativen Position der Objekte*



(b) *Erhaltung der räumlichen Relationen*

Abbildung 21: *Berechnung der Position beim mehrfache Click Zoom.*

Um das Szenario der multifokalen Interaktion an einem Multitouch-Tisch zu unterstützen, lag der Schwerpunkt dieser Arbeit auf der Umsetzung des *Drag&Drop*-Verhaltens auf dem Surface.

Die Implementierung von *Drag&Drop* auf einem Multitouch-Tisch stellt die Herausforderung, dass gleichzeitig mehrere Personen, mehrere Objekte verschieben können. Auf einem herkömmlichen Desktop-PC interagiert nur ein Benutzer mit dem System.

Ähnlich der Kopplung zwischen Eingabe und *Ausschnitt* beim *Split Screen*, wird auch bei der *Drag&Drop*-Implementierung die Eingabe direkt auf das Objekt gemappt. Die Interaktion mit einem Objekt tangiert dabei weder die Landschaft noch weitere Objekte.

Um dem Benutzer Feedback über seine *Drag&Drop*-Aktion zu geben, wird das Objekt dupliziert und unter dem Finger wird ein visueller Cursor angezeigt (vgl. Abbildung 22). Dieser Cursor ist unabhängig von der zugeordneten Landschaft. Die Kombination aus Input und Unabhängigkeit von der Landschaft ermöglichen das Verschieben von mehr als einem Objekt gleichzeitig. Der Benutzer erhält über den Cursor Feedback, welches Objekt er verschiebt.

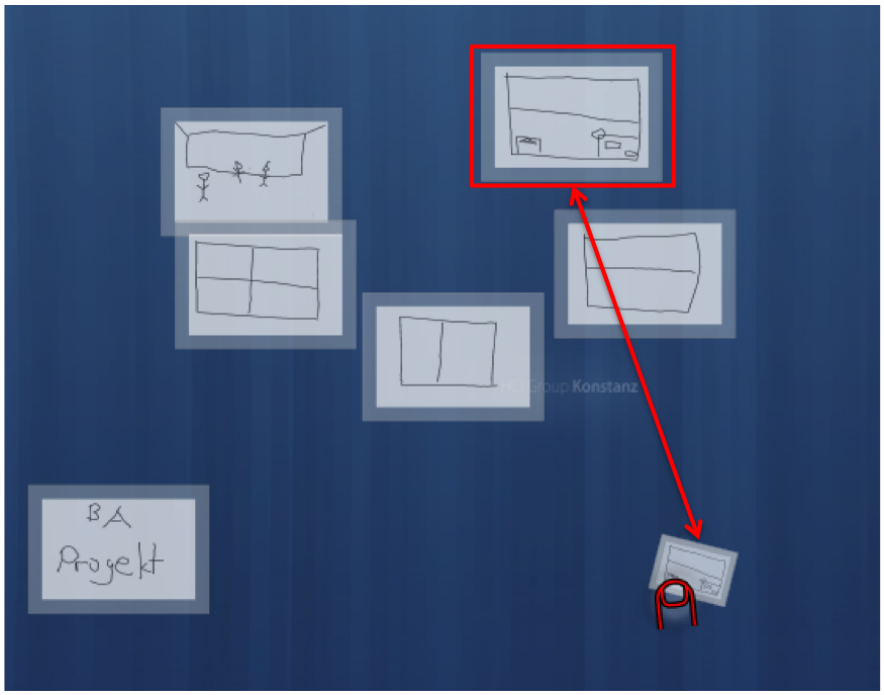


Abbildung 22: *Visuelles Feedback über bewegtes Objekt bei einer Drag&Drop-Operation.*

Mit Hilfe dieser *Drag&Drop*-Unterstützung kann der Benutzer nun Objekte zwischen den *Ausschnitten* verschieben und erhält dadurch eine Option für die „Verschieben“-Aktion im Single-Display Kontext.

Bevor Lösungen vorgestellt werden, die Aktionen in einem Multi-Display Kontext unterstützen, folgt die Beschreibung einer Zusatz-Komponente, die den Benutzer in seiner räumlichen Orientierung unterstützen soll.

4.3 Ein Metaelement - die interaktive Übersicht

Die Einführung des *Split Screen* (vgl. Kapitel 4.1.1) bietet neue Möglichkeiten für Anzeige und Manipulation von Objekten im Single-Display Kontext. Um dem Benutzer zusätzlich eine Orientierungsmöglichkeit zu geben, wurde im Rahmen dieser Arbeit ein Konzept für eine *interaktive Übersicht* erarbeitet. Die Übersicht unterstützt den Benutzer in der räumlichen Orientierung und verhindert die sogenannte „desert fog“ [Jul and Furnas, 1998, König, 2006]. Die *interaktive Übersicht* ist keiner expliziten Kategorie der Taxonomie zugeordnet. Sie unterstützt vor allem die Aktion *Anzeigen* im Single- und Multi-Display Kontext.

Zunächst wird beschrieben, wie sich die Landschaften (oder im *Split Screen* die *Ausschnitte*) bei der *interaktiven Übersicht* melden, bzw. wie sie dort repräsentiert sind. Die Interaktion innerhalb der *interaktiven Übersicht* folgt im Anschluss.

Die visuelle Darstellung der *interaktiven Übersicht* zeigt immer die ganze *Informationslandschaft* aus der Vogelperspektive. Durch die Synchronisation aller Objektzustände und -eigenschaften zeigt die Übersicht den aktuellen Stand der *Informationslandschaft*, so dass sich der Benutzer gut orientieren kann.

Während König eine Übersicht beschreibt, die einen Fokus in der *Informationslandschaft* abdeckt [König, 2006], muss in einer *multifokalen Arbeitsumgebung* die Möglichkeit gegeben sein, dass der Benutzer auch mehrere Fokusse in einer *interaktiven Übersicht* visualisiert bekommt. Die *interaktive Übersicht*, die im Rahmen dieser Arbeit vorgestellt wird, unterstützt diese Option. Sie erweitert zudem die Unterstützung mehrere Geräte und deren Displays. Die Umsetzung der Kommunikation über das Netzwerk ermöglicht diese Erweiterung.

Innerhalb der Übersicht sind die verschiedenen visualisierten *Ausschnitte* farblich gekennzeichnet. Jeweils ein halbtransparentes Rechteck mit einem Rahmen visualisiert einen *Ausschnitt* (vgl. Abbildung 23). Hier kommt die Verwendung der farblichen Unterscheidung, die in 4.1.1 vorgestellt wurde erneut zum Tragen. Diese visuelle Repräsentation durch Rechteck und farblichen Rahmen wird *Overview Shape* genannt. Das Zusammenspiel aus *Ausschnitt*, *interaktiver Übersicht* und *Overview Shape* wird mit Abbildung 24 deutlich.

Die *Ausschnitte* registrieren sich bei der *interaktiven Übersicht*. Dort werden sie mit Hilfe einer *Overview Shape* visualisiert. Die Interaktion in der *interaktiven Übersicht* und die Anpassung der *Ausschnitte* übernimmt jeweils eine *Overview Shape* für die zugeordnete Landschaft.

Abbildung 23: *Visuelle Darstellung des Split Screen mit einer interaktiven Übersicht (rechts unten im Bild).*

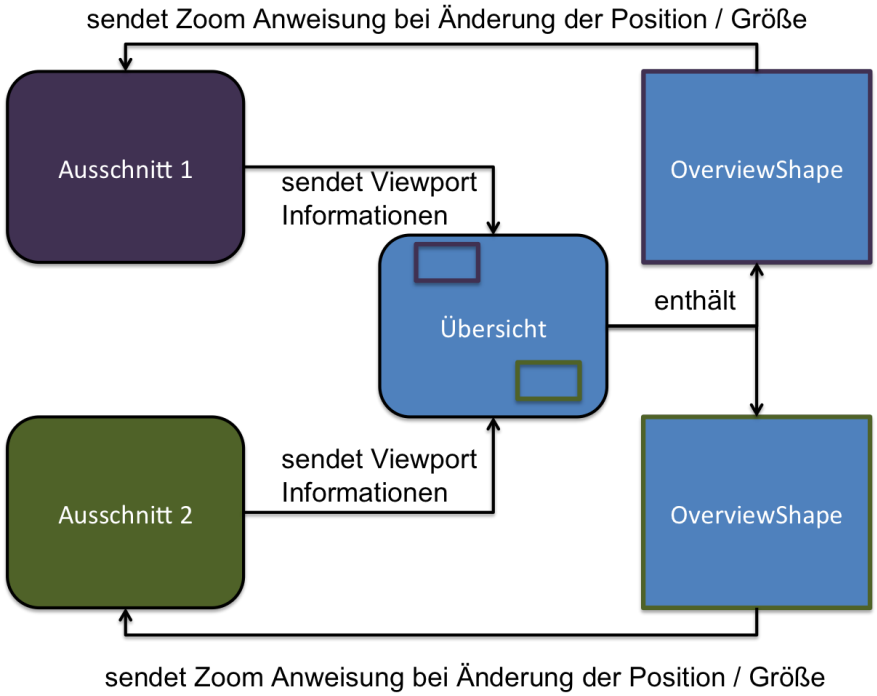


Abbildung 24: *Zusammenspiel zwischen den Landschaften, der Übersicht und den OverviewShapes.*

Die Interaktion innerhalb der *interaktiven Übersicht* bietet dem Benutzer die Möglichkeit, die Referenzen der *Ausschnitte* zu bewegen. Der referenzierte *Ausschnitt* wird an den neuen *Viewport* angepasst.

Die *interaktive Übersicht* ist nicht an ein bestimmtes Bildschirm Layout gekoppelt. Vielmehr kann die Übersicht je nach Anwendung in unterschiedlichen Größen und Positionen dargestellt werden, auch unabhängig auf einem eigenen Display. Im Rahmen dieser Arbeit wird die Übersicht als Element auf dem Display angezeigt. Um eine zu große Überdeckung von Displayfläche zu verhindern, kann die *interaktive Übersicht* ein- und ausgeblendet werden.

Mit Hilfe der vorgestellten *interaktiven Übersicht* erhält der Benutzer ein Element an die Hand, welches ihn bei der räumlichen Orientierung im Informationsraum unterstützt.

Im Folgenden werden nun die einzelnen Lösungen beschrieben, die den Benutzer bei der Interaktion in einem Multi-Display Kontext unterstützen.

4.4 Manipulation im Multi-Display Kontext - Synchronisierte Ansicht

Der Benutzer benötigt laut Taxonomie eine Möglichkeit, in einem Multi-Display Kontext, Objekte bearbeiten zu können, die auf einem *Remote Display* zu sehen sind. Wie schon in 3.3 kurz erwähnt, bietet es sich hier an, eine visuelle Kopie des entfernten Objekts auf dem *Current Display* zu erzeugen. Der Benutzer kann mit dieser Kopie das Objekt auf dem *Remote Display* manipulieren.

Um diese *synchronisierte Ansicht* zu erzeugen benötigt der Benutzer eine Interaktionsmöglichkeit. Hierfür ist ein Synchronisationsmodus vorgesehen. Ist dieser Modus aktiviert, so kann der Benutzer den *Viewport* eines entfernten *Ausschnitts* lokal steuern. Dies ermöglicht ihm, das entfernte Objekt auf dem *Current Display* zu manipulieren. Alle Manipulations- und Navigationsschritte am *Current Display* werden auf das *Remote Display* übertragen.

Verlässt der Benutzer den Synchronisationsmodus nach Abschluss der Manipulation, so kehrt die Ansicht wieder zum ursprünglichen Zustand des *Current Display* zurück. Die Synchronisation der *Ausschnitte* wird deaktiviert und der Benutzer kann normal weiterarbeiten.

4.5 Anzeige im Multi-Display Kontext - Remote Click Zoom

Um die Anzeige eines Objekts auf einem *Remote Display* zu steuern, wurde der *Remote Click Zoom* konzipiert. Damit wird eine weitere Kategorie der Taxonomie unterstützt, die Anzeige von mehreren Objekten in einem Multi-Display Kontext.

Der *Remote Click Zoom* ist eine Erweiterung des *Click Zoom*. Wie in 4.1.2 bereits angesprochen, wird bei einem *Click Zoom* das Objekt herangezoomt und kann so direkt fokussiert werden. Der *Remote Click Zoom* überträgt diese Funktionalität auf ein *Remote Display*. Möchte der Benutzer ein lokales Objekt auf einem *Remote Display* anzeigen, so kann er einen *Remote Click Zoom* ausführen. Das Objekt wird dann animiert auf dem *Remote Display* gezoomt und ist im Anschluss fokussiert zu sehen.

Die Implementierung dieser Funktionalität ermöglicht verschiedene Arten von Ziel-Displays. Jede Art von Display kann in diesem Szenario zu einem *Remote Display* werden. Dazu muss lediglich der nötige Input-Handler [16] am Remote Client registriert werden. Diese flexible Handhabung ermöglicht das schnelle Einbinden von neuen Geräten. Beispielsweise wären mobile Geräte in einem Besprechungsraum dadurch direkt als neues Geräte integrierbar.

Um den *Remote Click Zoom* steuern zu können, bekommt der Benutzer in jedem Objekt Interaktionsmöglichkeiten, die einen *Remote Click Zoom* auslösen. Dies wird im Folgenden am Beispiel der Sketch-Komponente[17] des ZOIL Framework (vgl. Abbildung 25).

Die Komponente steuert den *Remote Click Zoom* über zwei Buttons. Die Zuordnung erfolgt durch visuelle Icons der *Remote Displays*, die links und rechts positioniert sind. Der Benutzer erhält dadurch eine eingeschränkte Möglichkeit der räumlichen Zuordnung. Der *Remote Click Zoom* folgt einem offenen Kontrollparadigma (vgl. Kapitel 2.2). Einmal ausgelöst, wird das entfernte Objekt animiert gezoomt und groß angezeigt. Der Benutzer besitzt dadurch ständiges Feedback, aber keine Kontrolle über den weiteren Verlauf der Aktion. Er kann jedoch durch Auslösen einer weiteren Aktion auf das selbe *Remote Display* die bisherige Interaktion unterbrechen.

Mit Hilfe des *Remote Click Zoom* kann der Benutzer ein ausgewähltes Objekt schnell und effizient auf einem *Remote Display* anzeigen. Damit erhält er eine Möglichkeit für die „Anzeige im Multi-Display Kontext“.

Im nächsten Kapitel wird das letzte Konzept vorgestellt, das zur Vervollständigung der Taxonomie noch fehlt.

4.6 Objekte verschieben im Multi-Display Kontext - Shared Clipboard

Das *Shared Clipboard* ermöglicht das Verschieben von Objekten zwischen verschiedenen Displays.

[16] Im ZOIL Framework spricht man bei einem Input Handler von einer Klasse, die sich um die Behandlung bestimmter Eingabegeräte kümmert.

[17] Mit Hilfe dieser Komponente können schnell Zeichnungen erstellt werden, die im Rahmen des „Blended Interaction Design“-Projekts verwendet werden.

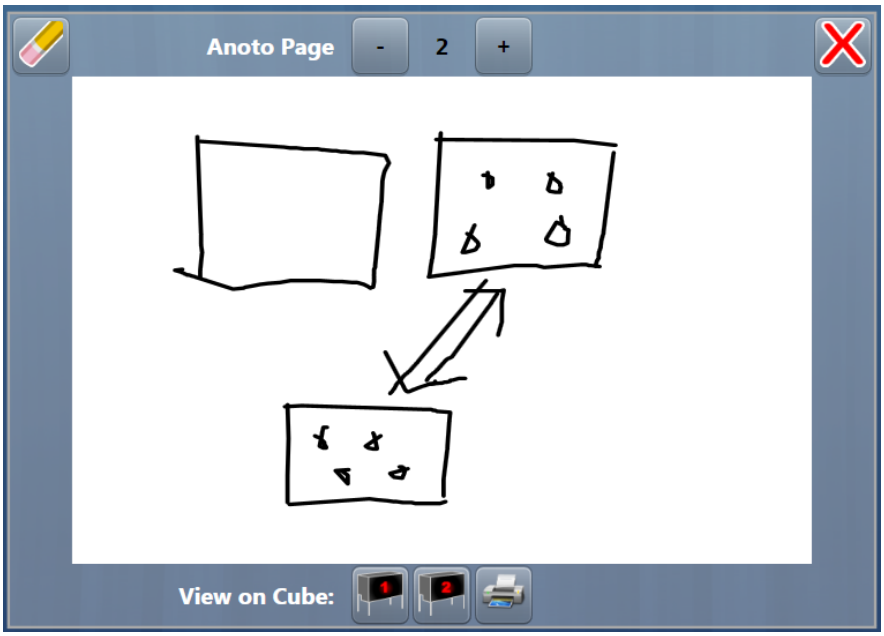


Abbildung 25: *Darstellung der Sketch-Komponente des* ZOIL Framework. *Die Buttons in der Mitte des unteren Rahmens bieten die Möglichkeit einen* Remote Click Zoom *auszulösen.*

Das Konzept eines Clipboards ist aus allen aktuellen Betriebssytemen bekannt. In ein Clipboard, zu deutsch Zwischenablage, kann man unterschiedliche Formen von Informationen ablegen. Dies geschieht meist per Tastenkombination oder Kontextmenü.

Diese klassische Clipboard-Idee wurde im Rahmen dieser Arbeit erweitert. Sie wurde sowohl auf ein OOUI angepasst, als auch auf die Anforderungen in einer *multifokalen Arbeitsumgebung*.

Das *Shared Clipboard* ist über das Datenbackend [Zöllner, 2009] organisiert. Dies erweitert die Möglichkeiten, die ein Benutzer mit dem *Shared Clipboard* hat. Zunächst werden Informationen im *Shared Clipboard* persistent gespeichert. Der Benutzer kann wichtige Objekte ablegen und zu jeder Zeit wieder auf sie zugreifen, auch nach einem Neustart der Anwendung. Hinzu kommt, dass das *Shared Clipboard* auf allen Geräten des Benutzers verfügbar ist, unabhängig vom Rest der Informationslandschaft.

Um dem Benutzer eine Möglichkeit zum Verschieben von Objekten zwischen verschiedenen Displays zu ermöglichen, wird lediglich ein *Shared Clipboard* eingebunden, das die Verschiebung der Objekte ermöglicht. Der Benutzer kann so beliebige Objekte in das *Shared Clipboard* bewegen und an anderer Stelle darauf zugreifen.

Das vorgestellte *Shared Clipboard* Konzept ermöglicht dem Benutzer Objekte auf einem lokalen System als Favoriten zu verwalten, eine persistente Speicherung interessanter Objekte und das Verschieben von Objekten zwischen verschiedenen Displays.

Die Interaktion mit dem *Shared Clipboard* funktioniert über *Drag&Drop*. Auch wenn

der Benutzer mit verschiedenen Eingabemodalitäten arbeitet, so ist das Prinzip von *Drag&Drop* bekannt. Er kann so an verschiedenen Geräten mit der gleichen Interaktion das *Shared Clipboard* bedienen. Für die visuelle Repräsentation wurde ein Realweltobjekt verwendet (siehe Abbildung 26). Der Benutzer soll durch die Metapher eines Klemmbretts schnell erfassen können, wozu das *Shared Clipboard* dient und wie er es verwenden kann.

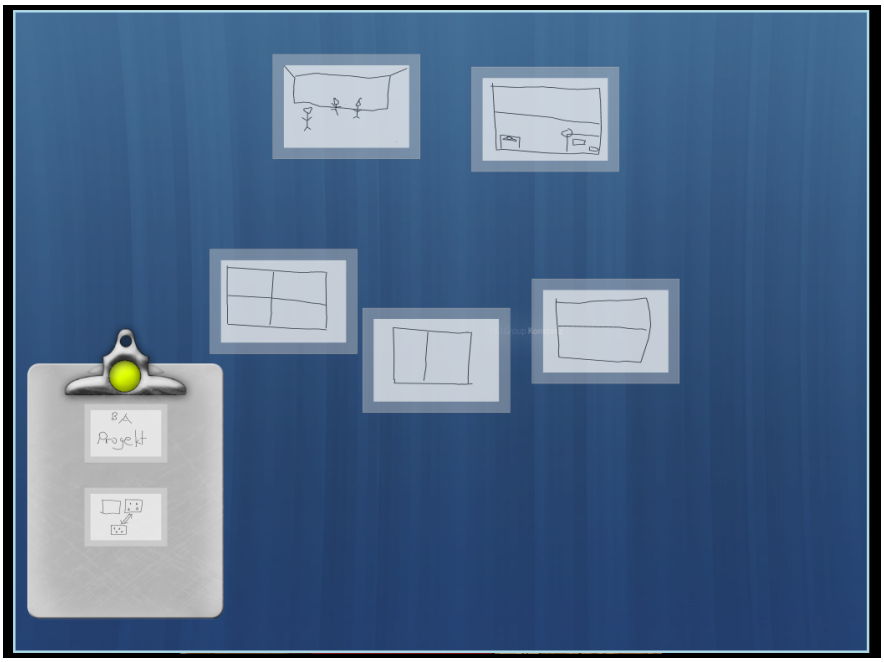


Abbildung 26: *Das Shared Clipboard mit zwei Elementen.*

Es ist dabei darauf zu achten, dass das *Shared Clipboard* nicht zu viel Displayfläche überdeckt. Aus diesem Grund wird das *Shared Clipboard*, wenn nicht benötigt, ausgeblendet und kann bei Bedarf erneut aufgerufen werden. Zieht ein Benutzer ein Objekt auf den sichtbaren Teil eines geschlossenen *Shared Clipboard*, so wird dieses automatisch geöffnet und der Benutzer kann das Objekt ablegen.

Auf der Basis der Analyse aus Kapitel 3 wurden in diesem Kapitel für die einzelnen Kategorien der Taxonomie Lösungskonzepte vorgestellt. Der Benutzer erhält damit für die verschiedenen Kontextsituationen Methoden an die Hand, mit denen er die in Kapitel 3 erarbeiteten Aktionen ausführen kann. Diese Konzepte müssen nun sinnvoll in das ZOIL Framework integriert werden.

5 Frameworkintegration

Wie man in Kapitel 3.3 gesehen hat, gibt es für die verschiedenen Kategorien der Taxonomie verschiedene Lösungen. Bedenkt man die unterschiedlichen Anwendungsdomänen, in denen multifokale Sichten eingesetzt werden und zudem verschiedene Benutzerszenarien und Interaktionstechniken, so wird schnell klar, dass es kein allgemeingültiges Konzept geben kann. Die behandelten Konzepte, die im Rahmen dieser Arbeit entwickelt wurden, bieten eine Basismenge an Funktionen, die in multifokalen Umgebungen verwendet werden können.

Aus diesem Grund war es Teil der Arbeit, dass die bisher umgesetzten Ideen in das vorhandene ZOIL Framework sauber eingepflegt werden. Die Funktionen sollen leicht weiterverwendet und -entwickelt werden können. Neben dieser grundlegenden Einbettung der Konzepte gab es zusätzliche Anforderungen an die softwaretechnische Integration. Diese werden im Folgenden kurz dargelegt, bevor anschließend detailliert erläutert wird, wie die Komponenten und weitere technische Grundlagen in das ZOIL Framework integriert wurden.

5.1 Anforderungen an die Implementierung

Meyers et al. beschreiben in ihrer Analyse von bestehenden und zukünftigen Interface Software Tools verschiedene Aspekte, die ihrer Meinung nach wichtig sind für ein erfolgreiches Tool [Myers et al., 2000]. Zwei dieser Aspekte haben nicht nur bei der Entwicklung des ZOIL Framework Priorität, sondern waren auch bei der Entwicklung und Integration der hier vorgestellten Konzepte wichtig:

Threshold behandelt die Frage, wie schwer es ist, die Benutzung des Systems (hier das Framework) zu erlernen.

Ceiling sagt aus, wie viel man mit dem System erreichen kann.

Höchstes Ziel wäre ein niedriger *Threshold*, verbunden mit einem hohen *Ceiling*. Meyers beschreibt zugleich, dass heutige Systeme meist einen niedrigen *Threshold* mit niedrigem *Ceiling* oder einen hohen *Threshold* mit hohem *Ceiling* besitzen. D.h. heutige Frameworks sind eher leicht zu erlernen, aber stoßen schnell an ihre Grenzen. Oder es benötigt einiger Einarbeitungszeit und man hat dann ein sehr mächtiges Tool an der Hand.

Unter Beachtung dieser beiden Aspekte sollen die beschriebenen Konzepte sinnvoll in das ZOIL Framework integriert.

5.2 Softwaretechnische Integration in das ZOIL-Framework

Die Integration der Konzepte aus Kapitel 4 soll die einfache Verwendung und Erweiterung der Ideen im Rahmen des ZOIL Framework ermöglichen.

Aus den vorgestellten Konzepten wurde ein Paket geschnürt, das alle nötigen Klassen und Funktionen bereit hält, um die Ideen praktisch zu verwenden. Im Folgenden wird zunächst der derzeitige Stand des ZOIL Framework kurz erläutert. Anschließend werden grundlegende Designentscheidungen dargelegt und das Paket in den Gesamtkontext des ZOIL Framework eingeordnet. Im weiteren Verlauf des Kapitels wird das erarbeitete Paket detailliert beschrieben.

Zum Abschluss des Kapitels werden einige kritische Beobachtungen erörtert, die während der Arbeit mit dem Framework und bei der Integration der Komponenten festgestellt wurden.

5.2.1 Architektur des ZOIL Framework

Das ZOIL Framework ist eine Ansammlung von Paketen, sogenannten Assemblies[18]. Jedes dieser Pakete unterstützt verschiedene Teile des Frameworks. Das Framework lässt sich in drei größere Bereiche unterteilen (vgl. Abbildung 27):

Framework Core Im Framework Core befinden sich Klassen und Methoden, die sich um das Zooming, Panning und das damit verbundene Rendering kümmern. Wichtigste Klasse ist die *ZInformationLandscape*.

Datenbackend Das von Michael Zöllner entwickelte Datenbackend hält Funktionen bereit, welche die Persistierung und Synchronisation von Objekten über eine Datenbank ermöglichen [Zöllner, 2009].

Bibliotheken Verschiedene Bibliotheken, die mit dem ZOIL Framework geliefert werden, ermöglichen den Zugang zu weiteren wichtigen Funktionen. Derzeit besitzt das ZOIL Framework bspw. Bibliotheken für Eingabegeräte, Domänenobjekte und Visualisierungen.

Das Paket, das im Rahmen dieser Arbeit integriert wurde (*Library.MultiFoci*) setzt auf dem bisherigen Framework auf und wird in 5.2.3 näher erläutert.

Der Framework Core stellt grundlegende Funktionen bereit, welche für ZUI-Anwendungen benötigt werden. Doch erst die Erweiterungen, in Form von Bibliotheken ermöglichen

[18] Eine Assembly ist eine Datei, die den gesamten Code eines Pakets in kompilierter Form speichert. Diese Datei kann dann in anderen Projekten eingebunden werden, um die enthaltenen Funktionen zu nutzen.

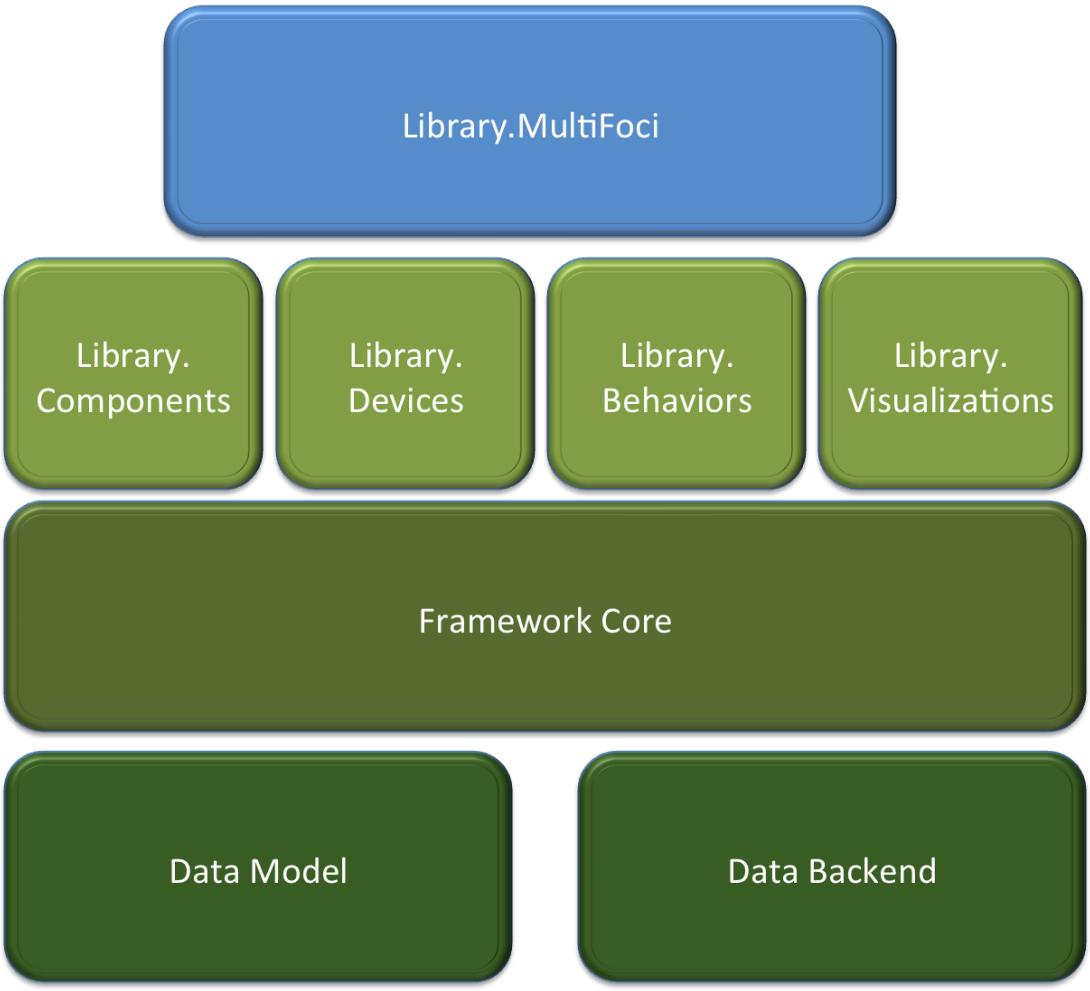


Abbildung 27: *Die Multi-Foci Bibliothek setzt auf dem Framework auf und bildet eine eigenständige Einheit.*

den schnellen und effizienten Einsatz des Frameworks. Neben den Basisfunktionalitäten kann sich der Anwendungsentwickler, ausgehend vom Core, nur die benötigten Klassen-Bibliotheken laden.

Bibliotheken enthalten in der Regel generische Funktionalitäten. Das heißt, dass bspw. eine spezielle Visualisierung in verschiedenen Anwendungen nützlich sein kann. Die Auslagerung in eine Bibliothek ermöglicht eine flexible Verwendung der Visualisierung und erleichtert die Wartung und Weiterentwicklung der Komponente.

Die Strukturierung in Bibliotheken hat weitere Vorteile. Die einzelnen Pakete bilden einen organisatorischen Rahmen. Sowohl für Entwickler, als auch für Anwender ist eine klare Struktur mit verschiedenen Teilen einfacher zu durchschauen und anzuwenden. Zudem lassen sich die Zugriffsrechte, bspw. im Rahmen einer OpenSource-Entwicklung, individuell gestalten. Im Vordergrund einer Implementierung mit Klassen-Bibliotheken steht die lose Kopplung der einzelnen Pakete. Es ist wichtig darauf zu achten, dass die einzelnen Pakete nicht zu viele Verweise aufeinander haben und dadurch die Architektur hinfällig machen.

5.2.2 Grundlagen

Ausgehend von diesem Stand werden wichtige Design-Entscheidungen erläutert, die während der Realisation der Arbeit hinsichtlich der Software-Architektur getroffen wurden.

Kommunikation Um die Kommunikation zwischen verschiedenen Geräten zu ermöglichen wird das OSC[19]-Framework der Firma Bespoke eingesetzt. Es stellt Funktionen bereit, mit der einfach und effizient Nachrichten zwischen vernetzten Geräten versendet und interpretiert werden können. OSC bietet einen guten Tradeoff zwischen Mächtigkeit und Komplexität. Zudem ist es gegenüber einer Kommunikation mit einer Datenbank performanter.

Die Kommunikationsanforderungen, die im Rahmen dieser Arbeit gestellt wurden, die Übermittlung von Koordinaten und einzelnen Identifikations-Strings, werden durch OSC voll abgedeckt. Während ein weiterer Kandidat, die Windows Communication Foundation[20] (WCF) eine Fülle an Funktionen bereitstellt, hätte dieser Ansatz erhebliche Nachteile gehabt. Neben der komplexeren Implementierung, machte auch die Installation und Konfiguration der Komponenten den Einsatz uninteressant. Das verwendete OSC-Framework kommuniziert über UDP[21] und ist daher in jedem Netzwerk

[19] OSC steht für Open Sound Control und bezeichnet ein nachrichtenbasiertes Kommunikationsprotokoll für verschiedene Geräte, wie bspw. Computern und Multimedia-Geräten.

[20] Die Windows Communication Foundation ist eine Kommunikationsplattform für verteilte Anwendungen und ist Teil des .NET Frameworks der Firma Microsoft

[21] Das User Datagram Protocol, kurz UDP, dient der Übermittlung von Datenpaketen in einem Netzwerk.

einsetzbar. Darüber hinaus bieten sich durch UDP auch Möglichkeiten einer geräteunabhängigen Kommunikation. Es bietet das Versenden von Nachrichten an Multicast-Adressen. Dabei wird die Nachricht nicht an eine einzige IP-Adresse versendet, sondern an eine Gruppe von Adressen. Multicast macht es möglich, dass neue Geräte in ein bestehendes Netz integriert werden können, ohne dabei Änderungen am Programm-Code zu machen.

Um das Verwenden von OSC im Framework zu erleichtern gibt es in der *Library.Devices* zwei Klassen, die das Versenden und Empfangen von Nachrichten ermöglichen (*OscSender* und *OscListener*). Da die Funktionalitäten dieser Klassen für die Konzepte dieser Arbeit nicht ausreichen, wurde eine weitere Klasse implementiert. Der *ViewportSender* stellt Methoden bereit, welche die Kommunikation in multifokalen Szenarien erleichtert.

Zusammengefasst bietet OSC alle Funktionen, die benötigt werden. Es ist einfach zu verwenden, performant in der Ausführung und flexibel nutzbar.

Ereignisgetriebene Steuerung Eine heute weit verbreitete Technik der Software-Architektur basiert auf Ereignissen. Dabei sieht das grundlegende Modell drei Stufen vor: Aktion, Ereignis und Reaktion (vgl. Abbildung 28). Aufgrund einer Aktion (bspw. „Drücken einer Keyboard-Taste“) wird ein Ereignis ausgelöst. Auf dieses Ereignis kann eine Anwendung bei Bedarf reagieren. Um über Ereignisse informiert zu werden, muss sich die Anwendung bei bestimmten Ereignissen als „Interessent“ registrieren.

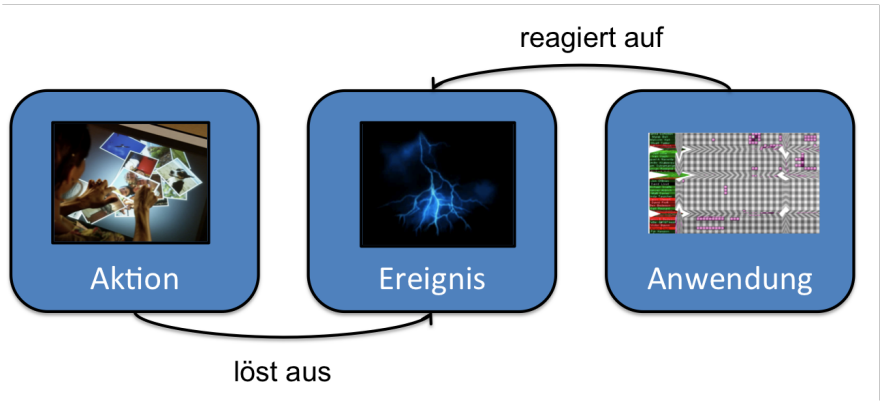


Abbildung 28: *Abstraktion von Aktion und Reaktion durch Ereignisse.*

Durch diese Entkopplung von Aktion und Reaktion werden häufig grundlegende Mechanismen, wie z.B. Interaktion oder Änderungen in Datenräumen von der Anwendunglogik getrennt. Die Anwendung reagiert dabei nur auf spezielle Ereignisse. Im ZOIL Framework wird diese Methode verwendet, beispielsweise um Selektionsverhalten, Interaktionen oder Synchronisation der Daten zu kommunizieren.

Darauf aufbauend, wurde auch im Rahmen dieser Arbeit darauf geachtet, stets eine lose Kopplung zur Anwendung zu ermöglichen. Aus diesem Grund basieren Benachrichtigungen über neue OSC-Nachrichten und viele Teile der Interaktion auf Ereignissen.

Interaktion Wichtig für die Unterstützung der erarbeiteten Konzepte sind neue Interaktionsmöglichkeiten. Bestimmte Interaktionen können dabei für verschiedene Anwendungen interessant sein. Es ist daher sinnvoll, diese generischen Interaktionen, wie bspw. Zooming und Panning, für das ganze Framework bereit zu stellen.

Im Rahmen dieser Arbeit sind verschiedene Interaktionskonzepte vorgestellt worden. Teile dieser Interaktionsmöglichkeiten sind generischer Natur und werden im Anschluss vorgestellt. Interaktionen, die näher an der konkreten Anwendung hängen werden im Anwendungskontext vorgestellt.

Mehrfacher Click Zoom Um dem Benutzer die Möglichkeiten für den *mehrfachen Click Zoom* (vgl. Kapitel 4.1.2) bereit zu stellen, bedarf es einer Möglichkeit mehrere Objekte zu fokussieren. Wie schon im Konzept erläutert, ist die Interaktion dabei zunächst unabhängig und kann durchaus auch in weiteren Szenarien eine Rolle spielen. Aus diesem Grund wurde das Interaktionsverhalten nicht in eine spezielle Bibliothek ausgelagert, sondern in die vorhandene *Library.Behaviors* integriert. Die *Library.Behaviors* kapselt Klassen zur direkten Objektinteraktion. Auch der *mehrfache Click Zoom* passiert direkt auf Objekten und wurde aus diesem Grund in diese Bibliothek eingepflegt.

Wie im vorherigen Abschnitt beschrieben, bietet sich für eine Abstraktion der Interaktion eine Implementierung mit Ereignissen an. Die Klasse *MultipleClickZoomBehavior* kapselt die Ereignisfunktionalität. Der *ZObjectSurfaceHandler* reagiert auf Fingerinteraktion mit dem Surface und löst bei Erkennen eines *mehrfachen Click Zoom* das nötige Ereignis aus. Die Anwendung kann nun darauf reagieren.

Drag&Drop Die Interaktion bei *Drag&Drop* verläuft generisch, während die Reaktion auf Ereignisse stark von den beteiligten Komponenten und der Intention des Benutzers abhängt. Ein simples Beispiel erläutert dieses Problem. Zieht ein Benutzer ein Dokument von einem Ordner in einen anderen, weiß das System nicht ob er es kopieren möchte oder verschieben. Dieses Verhalten zeigt, dass bei *Drag&Drop*-Verhalten die beteiligten Komponenten semantisch entscheiden müssen, wie die Aktion abläuft. Die Interaktion, also das Verschieben an sich, ist in allen Fällen gleich.

Die Klasse *SurfaceDragDropBehavior*, die im Rahmen dieser Arbeit umgesetzt wurde, bindet alle nötigen Ereignishandler an die *Informationslandschaften*. Die spezielle Anforderung, dass dieses *Drag&Drop*-Verhalten auch über mehrere Landschaften hinweg

funktioniert (siehe Kapitel 3.3) ist zunächst nur im Multi-Fokus Kontext besonders wichtig. Die Komplexität einer generischen *Drag&Drop* Implementierung, evtl. auch als Verbindung zwischen ZOIL und der WIMP-Welt, erfordern eine komplette Überarbeitung der bisherigen Implementierung. Die *Drag&Drop*-Unterstützung, welche im Rahmen dieser Arbeit umgesetzt wurde, ist deshalb zunächst in das ausgelagerte Paket integriert und beeinflusst den Rest des Frameworks nicht. Im Anschluss an dieses Kapitel werden die Probleme und Möglichkeiten einer ganzheitlichen *Drag&Drop*-Unterstützung ausführlicher diskutiert.

Die Klasse *SurfaceDragDropBehavior* ermöglicht die Aufnahme, das Verschieben und die Ablage von Objekten in unterschiedlichen *Informationslandschaften*. Für die *Informationslandschaften* ist es dabei nicht relevant, von welcher Quelle das Objekt kommt. Die Implementierung überprüft lediglich, ob das übergebene Datenobjekt einem validen Typ entspricht. Ist dies der Fall, kann das Objekt hinzugefügt werden.

Die visuelle Repräsentation ist über die Möglichkeit realisiert, mehrere Sichten auf die gleichen Datenobjekte zu erhalten [Zöllner, 2009]. So können für die Domänenobjekte, die in der *Library.Components* beinhaltet sind, sogenannte „Thumbnail Views“ kreiert werden, die sowohl für das *Drag&Drop*-Verhalten, als auch für weitere Elemente verwendet werden können, um eine Voransicht auf das Objekt zu ermöglichen.

Zusammenfassend lässt sich sagen, dass die Entkopplung des *mehrfachen Click Zoom* dem ereignisbasierten Konzept des ZOIL Framework entspricht. Weiter wurde ein generisches *Drag&Drop*-Verhalten für den Surface umgesetzt, das aufgrund von komplexeren Zusammenhängen zunächst nicht in das Gesamtframework eingeordnet wurde, aber auf Basis der *Informationslandschaften* für das *Drag&Drop*-Verhalten im Allgemeinen verwendet werden kann.

Es folgt die Beschreibung des Pakets, das im Rahmen dieser Arbeit umgesetzt wurde.

5.2.3 Modulare Implementierung der Library.MultiFoci

Die *Library.MultiFoci*, die im Rahmen dieser Arbeit entstanden ist, ergänzt das vorhandene Framework um Funktionen, die multifokales Arbeiten ermöglichen.

Einordnung in das Gesamtframework Aus den bereits beschriebenen Gründen, wird auch die *Library.MultiFoci* als Bibliothek in das ZOIL Framework integriert. Die *Library.MultiFoci* beinhaltet einzelne Komponenten, Klassen für die Interaktion und den *Split Screen*.

Die Auslagerung in eine Bibliothek beinhaltet einige Herausforderungen. Entscheidend ist hierbei vor allem, dass keine Abhängigkeiten entstehen, die das Framework direkt an die Bibliothek binden. In anderen Worten ausgedrückt: „Man muss die Bibliothek entfernen können und der Rest muss immer noch laufen“.

Die Implementierung der *Library.MultiFoci* setzt auf dem bisherigen Framework auf und ist lose an das Framework angebunden (vgl. Abbildung 27). Das bedeutet, dass man die *Library.MultiFoci* aus dem Projekt entfernen kann und die restlichen Pakete weiterhin funktionieren.

Neben dieser losen Kopplung der Bibliothek an das Framework wurden weitere Aspekte bei der Architektur beachtet. Die Wiederverwendbarkeit und Erweiterbarkeit der einzelnen Entitäten standen dabei im Vordergrund. Wie dies für die *Library.MultiFoci* realisiert wurde, wird im Folgenden dargelegt.

Die *Library.MultiFoci* ist in drei Bereiche unterteilt (vgl. Abbildung 29). Den ersten Bereich bilden Klassen, die für grundlegende Aufgaben, wie bspw. elementare Interaktion zuständig sind (vgl. Kapitel 5.2.2). Der zweite Bereich enthält Komponenten, die unabhängig voneinander verwendet werden können und im Kontext dieser Arbeit entstanden sind. Ein dritter Bereich kapselt die Implementierung des *Split Screen*.

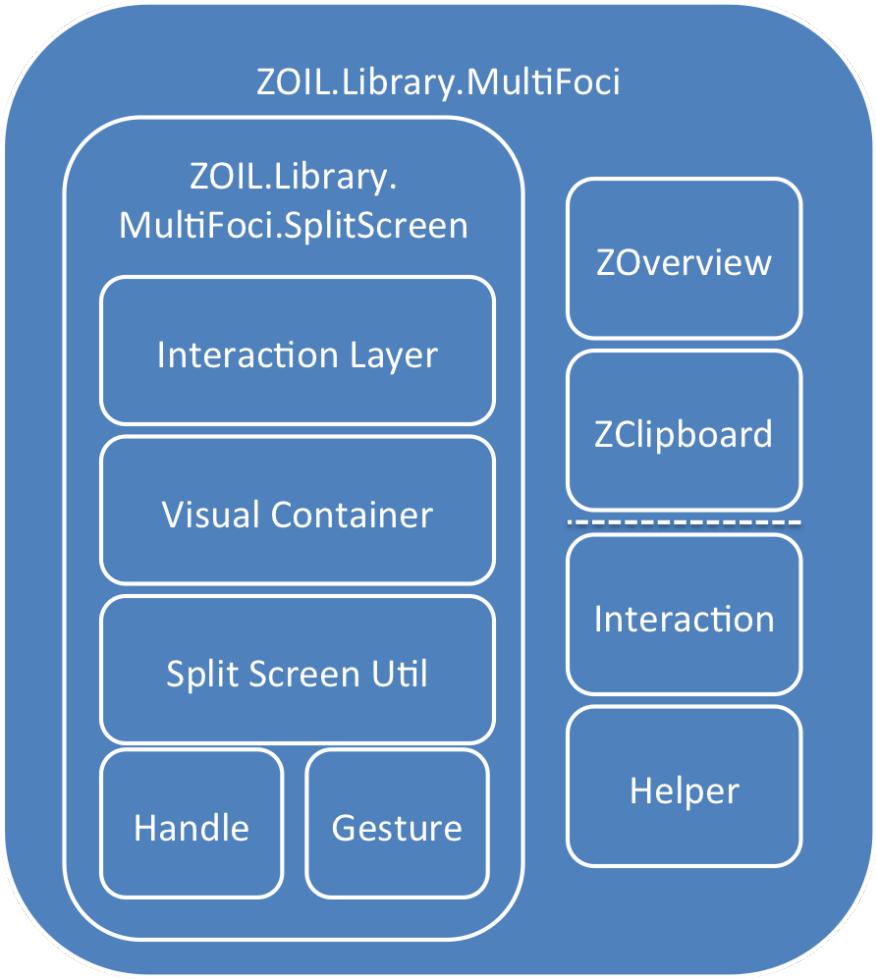


Abbildung 29: *Übersicht über die Library.MultiFoci*

Um in WPF (Windows Presentation Foundation)[22] Klassen zu gruppieren, werden sogenannte *namespaces*[23] verwendet. Für die *Library.MultiFoci* wurden zwei *namespaces* angelegt:

ZOIL.Library.MultiFoci Dieser *namespace* beinhaltet die ersten beiden Bereiche, also Interaktionsklassen, Helferklassen und die eigenständigen Komponenten.

ZOIL.Library.MultiFoci.SplitScreen Dieser erweiterte *namespace* umfasst die Klassen für den *Split Screen*.

Mit Hilfe dieser beiden *namespaces* kann der Anwender auf die volle Funktionalität für multifokale Sichten im ZOIL Framework zugreifen. Die folgende Beschreibung der einzelnen Bereiche der *Library.MultiFoci* erläutert die verschiedenen Module der Bibliothek. Es wird kurz beschrieben, wie sie umgesetzt wurden, wie sie zusammenhängen und wie der Entwickler sie verwenden kann.

Die Komponenten wurden nach dem MVVM-Pattern[24] implementiert. Die Vorteile des MVVM-Pattern liegen in der Unterscheidung zwischen Präsentation und Datendomäne, der Möglichkeit verschiedene Sichten auf die gleichen Daten zu erzeugen und der guten Anbindungsmöglichkeit an eine Datenbank über das darunter liegende Modell. Es ist möglich die Komponente sowohl mit, als auch ohne Datenbackend zu verwenden. Die Komponentenentwicklung innerhalb des ZOIL Framework wird zunehmend mit diesem Pattern betrieben. Diese Argumente sprachen dafür, auch die multifokalen Komponenten mit dem Pattern umzusetzen. Die konkrete Realisierung und notwendige Abwägungen werden nachfolgend beschrieben.

ZOverview Die Umsetzung der *interaktiven Übersicht* besteht aus zwei Teilen. Die *ZOverview* übernimmt die visuelle Erscheinung und das Handling neuer Netzwerk-Ereignisse. Die *ZOverviewShape* übernimmt die visuelle Repräsentation der referenzierten Landschaften in der Übersicht.

Während die *ZOverview* dem MVVM-Pattern folgt, ist die *ZOverviewShape* nur als Hilfsklasse realisiert, benötigt keine persistente Datenhaltung und ist somit als normales *UserControl*[25] implementiert. Die Auslagerung in ein *UserControl* ermöglicht das individuelle Styling der Shapes.

Die Kommunikation zwischen den Landschaften und der *interaktiven Übersicht* ist über OSC realisiert. Die *ZOverview* implementiert einen *OscListener*, empfängt damit neue

[22] WPF ist ein Teil des .NET Frameworks und dient der Anwendungentwicklung unter Windows.

[23] Namespaces sind eine verbreitete Methode um Klassen zu gruppieren, vergleichbar mit den Paketen in Java. In WPF werden sie zusätzlich verwendet um im XAML-Code andere Pakete einzubinden

[24] Das **M**odel-**V**iew-**V**iewmodel Pattern, unterscheidet zwischen Präsentation und Daten, ähnlich dem Model-View-Controller Pattern. Nähere Information beschreibt Zöllner in seiner Projektdokumentation [Zöllner, 2009].

[25] Ein UserControl wird in WPF verwendet um eigene Komponenten umzusetzen.

Nachrichten und passt entsprechend ihre Ansicht an. Diese Aktionen laufen im *ZOverviewViewModel* ab (vgl. Listing 1). Neue Landschaften, Anpassung der *Viewports* oder das Ausblenden von inaktiven Landschaften werden hier gesteuert und an die entsprechenden *ZOverviewShapes* weitergeleitet. Die *ZOverviewShapes* kapseln Informationen über Position und Größe des zugeordneten *Ausschnitts* und können so auf Änderungen reagieren.

```
//Osc Listener erstellen
public static OSCListener _listener = _listener = new OSCListener
   (4496, "/zoil/viewport", MessageReceivedMethod);

//bei Empfangen einer Nachricht werden die OverviewShapes angepasst
void MessageReceivedMethod(object sender, OscMessageReceivedEventArgs
    e)
{
  // Informationen aus Nachricht extrahieren
  ViewportChangedEventArgs args = ...
  // Aenderungen an OverviewShapes uebergeben
  ApplyViewportChanges(GetShape(arg.ID), arg.X, arg.Y, arg.width, arg
     .height, arg.color);
}
```

Listing 1: *Verarbeitung der ankommenden Nachrichten im ZOverviewViewModel*

Die Kommunikation über OSC ermöglicht den Einsatz der *interaktiven Übersicht* im Netzwerk. Landschaften versenden lediglich die entsprechenden Anweisungen über OSC und die *ZOverview*-Komponente reagiert.

Der entgegengesetzte Weg funktioniert über die *ZOverviewShapes* Interaktionen in der *interaktiven Übersicht* werden von den *ZOverviewShapes* verarbeitet. Sie reagieren auf alle Änderungen in Größe oder Position und schicken direkt ein Update an die referenzierte Landschaft. Diese kann dann wiederum ihren *Viewport* anpassen. Die beiden Klassen, welche die OSC-Nachrichten behandeln, sorgen dafür, dass keine Schleifen in der Kommunikation entstehen. Eine Übersicht über diesen Zusammenhang bietet Abbildung 24.

Die *interaktive Übersicht* ist eine Komponente, die in die *Library.MultiFoci* integriert wurde. Sie ermöglicht visuelle Hilfen für mehrere Landschaften. Die Kommunikation, realisiert in OSC, und die Eigenständigkeit der Komponente ermöglichen einen flexiblen Einsatz der *interaktiven Übersicht*. Die Komponente ist nicht abhängig von anderen Elementen, kommuniziert über ein Netzwerk und ist somit auf verschiedenen Geräten, als alleinige Komponente oder in Kombination mit weiteren Elementen verwendbar.

Der Anwendungsentwickler muss lediglich eine Übersichts-Komponente initialisieren

und diese im visuellen Baum der Anwendung einhängen (vgl. Listing 2).

```
ZOverview ovm = new ZOverview();
//Zwischenschritt durch MVVM-Pattern
ZOverviewView ov = MVVMUtil.CreateDefaultView(ovm) as ZOverviewView;
ov.Width = 300;
ov.Height = 200;
Canvas.SetTop(ov, 685);
Canvas.SetLeft(ov, 960);
this.ElementLayer.Children.Add(ov);
```

Listing 2: *Einfügen einer Übersicht-Komponente in einer Beispielanwendung*

ZClipboard Auch das *Shared Clipboard* wurde als eigenständige Komponente in die *Library.MultiFoci* integriert. Genau wie die *ZOverview* wurde das *ZClipboard* mit dem MVVM-Pattern implementiert. Die Implementierung unterscheidet sich jedoch von den restlichen Komponenten, speziell in der Datenhaltung, also der *ZClipboard*-Klasse[26]. *ZClipboard* erbt von *DInformationCollection*. Die *DInformationCollection* ist die Basisklasse zum Erzeugen einer Sammlung von Objekten, die über die Datenbank synchronisiert werden. Sie ist Bestandteil der *ZOIL.DataBackend*-Bibliothek.

Das *ZClipboardViewModel* hält *Commands* bereit zum Hinzufügen und Entfernen von Objekten. Daran kann nun die Interaktion und die visuelle Anzeige geknüpft werden. Beides ist in der *ZClipboardView* realisiert. Das *Drag&Drop*-Verhalten ähnelt dem der Landschaften. Wie aber schon in Kapitel 5.2.2 angesprochen, muss das *Drag&Drop*-Verhalten von speziellen Objekten selbst übernommen werden, da vor allem das Droppen spezifisch auf die Komponente gemappt wird. Die *ZClipboardView* implementiert alle nötigen Ereignishandler, die notwendig sind (vgl. Listing 3). Die *Drag&Drop*- Implementierung wurde nicht in eine eigene Klasse ausgelagert, da sie sehr stark an die visuelle Erscheinung der *View* geknüpft ist.

```
void DragCompleted(object o, SurfaceDragCompletedEventArgs e)...
void DataContextChanged(object o, DependencyPropertyChangedEventArgs
   e)...
void DragSourcePreviewContactDown(object o, ContactEventArgs e)...
void DropTargetDragEnter(object o, SurfaceDragDropEventArgs e)...
void DropTargetDragLeave(object sender, SurfaceDragDropEventArgs e)
   ...
void DropTargetDrop(object sender, SurfaceDragDropEventArgs e)...
void TargetChanged(object sender, TargetChangedEventArgs e)...
```

Listing 3: *Ereignishandler in der ZClipboardView regeln das Drag&Drop-Verhalten*

[26] Die Klasse übernimmt die Rolle des Model im MVVM-Pattern.

Die Daten des *Shared Clipboard* werden in der Datenbank gespeichert, aktuell gehalten und sind auf jedem System verfügbar.

Das *Shared Clipboard* kann entsprechend der *interaktiven Übersicht* sehr leicht in eine bestehende Anwendung integriert werden (vgl. Listing 4). Der Entwickler muss sich nicht um die Datenhaltung kümmern und kann seine individuelle Ansicht und Interaktion entwickeln.

```
ZClipboard cb = loadClipboard();
//Zwischenschritt fuer MVVM-Pattern
ZClipboardView cbView = MVVMUtil.CreateDefaultView(cb) as
   ZClipboardView;
cbView.Width = 250;
cbView.Height = 500;
Canvas.SetLeft(cbView, 10);
Canvas.SetTop(cbView, 630);
this.ElementLayer.Children.Add(cbView);
```

Listing 4: *Hinzufügen eines Clipboard in eine Anwendung*

SplitScreen Die Umsetzung des *Split Screen* wurde wie schon erwähnt in einem eigenen *namespace* ausgelagert, da er aus mehreren Klassen besteht und diese somit als Teilpaket klar getrennt sind.

Bei der Implementierung des *Split Screen* haben andere Faktoren eine Rolle gespielt, als bei den bisherigen Komponenten. Wichtig war die schnelle und flexible Integration mit Hilfe der Extensible Application Markup Language (XAML). Da der *Split Screen* nicht innerhalb einer *Informationslandschaft* verwendet wird und keine Datenhaltung erfordert, wurde auf das MVVM-Pattern verzichtet.

Dennoch sieht auch der *Split Screen* eine Trennung in drei wesentliche Teile vor (vgl. Abbildung 29):

ZSplitScreenContainer Diese Klasse übernimmt die visuelle Anzeige des *Split Screen*. D.h. hier geschieht die Einteilung der *Ausschnitte* (das Layout), die Berechnung der *Split*- und *Merge*-Operationen und die visuelle Darstellung der farblichen Rahmen. Zusätzlich werden Funktionen angeboten, die nach außen hin die einzelnen *Ausschnitte* verfügbar machen. Diese sind so abstrakt, dass sie nach außen keine spezielle Implementierung voraussetzen (vgl. Listing 5).

ZSplitScreenUtil Die Util-Klasse übernimmt im *Split Screen* die Organisation der Landschaften. Dort werden die *Views* initialisiert, die richtigen Daten aus der Datenbank gelesen, die nötigen Device-Handler angemeldet. Darüber hinaus senden die Landschaften ihren aktuellen *Viewport* an die *interaktive Übersicht*. Auch diese Funktionen übernimmt die Klasse *SplitScreenUtil*.

ZSplitScreenInteractionLayer Neben diesen beiden Klassen wurde ein dritter Teil implementiert, der sich um die Interaktion mit dem *Split Screen* kümmert. Wie in Kapitel 4.1.1 und 4.1.2 beschrieben, werden derzeit drei Arten von Interaktionen unterstützt. Der *ZSplitScreenInteractionLayer* reagiert auf Tastatureingaben, behandelt die Gesteninteraktion mit den *Handles* und auf einen *mehrfachen Click Zoom*. Um die XAML-Definition zu ermöglichen wird die Verbindung zwischen *Container* und *InteractionLayer* mit Hilfe einer *DependencyProperty* realisiert. Die Verbindung kann so über *Data Binding* hergestellt werden.

```
public ZInformationLandscape GetLandscapeFromSection(String
    sectionName)
{
  // intern wird die Landschaft einer Border zugeordnet
  return GetLandscapeFromBorder(b);
}
```

Listing 5: *Kapselung des Zugriffs auf die Informationslandschaften innerhalb des SplitScreenContainer*

Die Trennung zwischen visueller Darstellung und Interaktion kann nicht mit einer Trennung zwischen Anwendungslogik und Präsenation aus dem MVVM-Pattern realisiert werden. Der *InteractionLayer* ist genau wie der *Container* eine visuelle Komponente in WPF, da die Interaktion beispielsweise auf den *Handles* stattfindet.

Die Kommunikation der beiden Komponenten läuft ereignisgetrieben. Die Klasse *ZSplitGestureHandler* kapselt die nötigen Methoden, der *ZSplitScreenInteractionLayer* löst das Ereignis aus und der *ZSplitScreenContainer* reagiert.

Mit Hilfe der Trennung dieser drei Bereiche ist es möglich, dass sowohl der *InteractionLayer*, als auch der *Split Screen* grundsätzlich unabhängig voneinander verwendet und weiterentwickelt werden können.

Für den Anwendungentwickler ist jedoch wieder entscheidend, wie er nun diese Komponente verwenden kann. Listing 6 demonstriert, wie innerhalb von XAML ein *Split Screen* und der dazugehörige *InteractionLayer* implementiert werden. Die beiden Komponenten können beliebig in XAML integriert werden und mit Hilfe des *Data Binding* einander zugeordnet werden. Weitere Code ist nicht notwendig.

In diesem Kapitel wurde die Integration der einzelnen Komponenten in das vorhandene ZOIL Framework näher erläutert. Die Analyse hat gezeigt, dass die Aufteilung in Klassenbibliotheken sinnvoll erscheint im Bezug auf die klare Trennung verschiedener Bereiche und die dadurch entstehende bessere Wartbarkeit.

Das vorgestellte Paket enthält die notwendigen Funktionen, um die in Kapitel 4 beschriebenen Konzepte zu realisieren. Dabei ist die Integration modular aufgebaut, so

```
<s:SurfaceWindow x:Class="Client.MultiFoci.Window1">
...
<p:ZSplitScreenContainer x:Name="SplitScreenGrid" Visibility="Visible
    " />
<p:ZSplitScreenInteractionLayer x:Name="InteractionLayer"
  HandlesEnabled="True"
  SplitScreenContainer="{Binding ElementName=SplitScreenGrid}"
  Visibility="{Binding ElementName=SplitScreenGrid, Path=Visibility}"
      />
...
</s:SurfaceWindow>
```

Listing 6: *Einfache Integration der Split Screen Komponente innerhalb von XAML*

dass eine Weiterverwendung und Weiterentwicklung der einzelnen Teile gut möglich ist. Mit Bezug auf Kapitel 5.1 ist dies ein Indikator für ein gutes *Ceiling*, da die vorhandenen Möglichkeiten kombiniert und leicht erweitert werden können.

Darüber hinaus wurde gezeigt, dass ein Anwendungsentwickler die Komponenten sehr leicht und flexibel einsetzen kann. Die einfache Integration der einzelnen Komponenten lässt auf einen niedrigen *Threshold* schließen (vgl. Kapitel 5.1).

Während der Realisierung im Rahmen dieser Arbeit sind einige Fragen und Probleme aufgetreten, welche die Architektur des Frameworks und die hier vorgestellten Bereiche betreffen. Diese werden zum Abschluss des Kapitels näher ausgeführt.

5.3 Lessons Learned

Die Entwicklung eines Software-Frameworks stellt viele besondere Anforderungen. Klare Struktur, einfache Verwendung und Wiederverwendung, Ausgliederung generischer Teilbereiche und eine lose Kopplung zwischen diesen Bereichen sind nur einige Beispiele.

Im Folgenden werden Erfahrungen ausgeführt, die der Autor während der Arbeit gemacht hat und mögliche Alternativen dargelegt.

Generische Interaktionen Ein wichtiges Element für ein Framework, das auf Rapid-Prototyping und UI-Programmierung ausgelegt ist, bildet die Unterstützung von generischen Interaktionen. Selektionsverhalten, Objektmanipulation und verschiedene Eingabegeräte sind Beispiele dafür.

Im vorliegenden Framework wurden diese Bereiche in zwei Bibliotheken ausgelagert, die *Library.Devices* und die *Library.Behaviors*. Erstere beinhaltet Handler für mögliche Eingabegeräte. Das zweite Paket beinhaltet unterschiedliche Verhaltensweisen, bspw.

verschiedene Arten für Selektion oder Zooming. Darüber hinaus sind Klassen enthalten, die sich um die direkte Manipulation von Objekten kümmern, indem sie Methoden zur Verfügung stellen, die an spezielle Objekte neue Verhaltensweisen anknüpfen lassen.

Diese Aufteilung hat einige Vorteile. Ein Anwendungsentwickler kann schnell neue Geräte als Eingabe verwenden. Dies versetzt ihn in die Lage, sich nicht um die grundlegende Interaktion zu kümmern. Die *Library.Behaviors* bietet ebenfalls die Möglichkeit mit geringem Aufwand neue Features zu integrieren, bspw. das Rotieren von Objekten.

Zwei strukturelle Probleme fallen jedoch auf. Zum Einen sind die beiden Pakete nicht unabhängig. Die *Library.Devices* enthält Verweise auf *Library.Behaviors* und zum zweiten sind beide Pakete an verschiedene Plattformen gebunden (Microsoft Surface und Windows). Problem eins zeigt, dass es Überschneidungen in den Paketen gibt, die zunächst einer Trennung der Pakete widerspricht oder eine Neuordnung erfordert. Problem zwei ist problematisch, da die Bibliotheken nur auf den vorgeschriebenen Plattformen verwendet werden können.

Ein weiteres Beispiel soll die Probleme verdeutlichen. Im Rahmen dieser Arbeit wurde versucht ein neues *Drag&Drop*-Verhalten zu implementieren. Diese Aktion passiert zunächst auf Objekten. Entscheidend ist aber auch die Reaktion der Container. Zudem soll das Verhalten mit verschiedenen Eingabegeräten funktionieren. Die Implementierung eines solchen Verhaltens bedeutet, dass man in verschiedenen Pakete für verschiedene Geräte programmieren muss.

Um dieses Probleme zu lösen, wäre eine Restrukturierung des Frameworks sinnvoll. Im Folgenden wird eine Möglichkeit aufgezeigt, wie die Struktur des Frameworks geändert werden könnte.

Zunächst könnten Klassen, die abstrakte Funktionen zu Interaktion und Navigation bereit halten in einer *Library.Interaction* zusammengefasst werden. Die Bibliothek sollte geräteunabhängig sein. Mit den Klassen *ZLandscapeInputHandler* und *ZBehavior* befinden sich bereits Kandidaten für solche Klassen im Framework. Weiter könnten konkrete Bibliotheken erstellt werden, welche die geforderten Aktionen aus der *Library.Interaction* für jeweils ein Gerät umsetzen (vgl. Abbildung 30).

Ein Vorteil dieser Struktur liegt im Zusammenspiel der Interaktionen. Innerhalb eines konkreten Geräts, bspw. für den Surface könnten alle Möglichkeiten genutzt werden, die ein spezielles Geräte anbietet. Die einzelnen Interaktionen könnten aufeinander abgestimmt werden. Zum Beispiel ist die Verbindung zwischen Tabgeste und *Drag&Drop*-Interaktion ein typisches Problem für den Surface. Dies könnte intern geregelt werden, da beide Implementierungen im gleichen Paket enthalten sind.

Die *Library.Interaction* abstrahiert diese Verhaltensweisen und enthält eine Grundmenge an Möglichkeiten, die der Anwendung zur Verfügung gestellt werden. Der Anwendungsentwickler arbeitet zunächst nur mit der *Library.Interaction*.

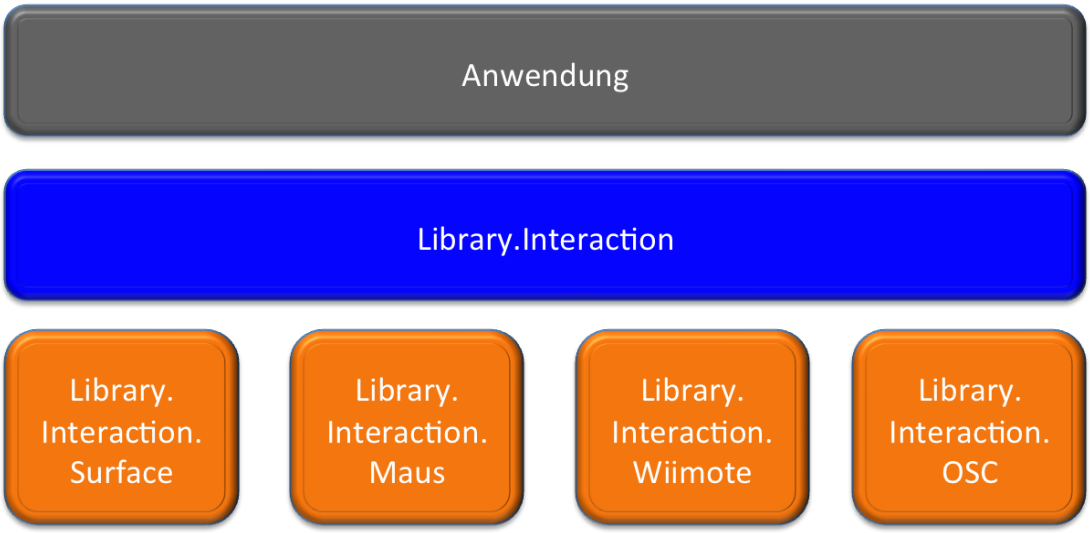


Abbildung 30: *Mögliche Restrukturierung der Interaktionen im ZOIL Framework.*

Die konkreten Implementierungen, also die Gerätebibliotheken *Library.Interaction.-Surface*, müssen die Funktionen der *Library.Interaction* bereitstellen. Der Anwendungsentwickler muss sich nicht darum kümmern, von welchem Gerät die Interaktion ausgelöst wird. Möchte der Entwickler auf spezielle Funktionen, wie Gesten (Surface) oder Druckempfindlichkeit (Stifteingabe) zugreifen, so hat er die Möglichkeit die konkreten Umsetzungen der einzelnen Geräte direkt zu verwenden. Er umgeht bewusst die *Library.Interaction* und nimmt damit in Kauf, dass die Anwendung auf ein bestimmtes Gerät festgelegt ist. Die Struktur erhält aber alle Möglichkeiten der verschiedenen Eingabegeräte.

Mit einer solchen Struktur könnten auch Anwender unterstützt werden, die zum Beispiel keinen Bedarf an Multitouch-Unterstützung oder dergleichen haben. Probleme durch inkompatible Plattformen würden umgangen (Beispiel Surface), da nur die notwendigen Gerätebibliotheken eingebunden werden müssten.

Diese Struktur hat jedoch auch Nachteile. So müssten für jedes Gerät ein Extra-Paket eingebunden werden. Zudem müssen die einzelnen Geräte an die Anwendung gebunden werden. Dies könnte eine globale Klasse übernehmen (z.B. *ZInteraction*). In ihr könnten die notwendigen Informationen zu den Geräten gesammelt werden. Der Anwendungsentwickler kann so verschiedene Features an seine Anwendung binden und bleibt dennoch unabhängig von den Eingabegeräten.

Visuelle Darstellung der Domänenobjekte Ähnliche Probleme treten auf, wenn man sich die Bibliothek ansieht, die verschiedene Domänenobjekte enthält (*Library.-Components*). Derzeit werden in dieser Bibliothek für jeden Typ von Objekt alle Teile des MVVM-Patterns integriert. Dies hat den Nachteil, dass erneut Abhängigkeiten zu be-

stimmten Plattformen entstehen. Ein Beispiel: Verwendet eine bestimmmte Ansicht eine Surface-Komponente, so kann das gesamte Paket nur auf Surface kompatiblen Systemen eingesetzt werden.

Eine ähnliche Restrukturierung wie im vorherigen Abschnitt würde hier Abhilfe schaffen. Die Vorteile des MVVM-Pattern wurden bereits gennant und die Komponentenentwicklung wird auch in Zukunft damit stattfinden. Um das Problem der Inkompatibilitäten zu verhindern, müssten die einzelnen Teile des Pattern getrennt werden.

Zunächst gäbe es ein allgemeines Paket für alle Datenobjekte (*Library.ObjectModels*), bspw. *ZMovie* und *ZPhoto*. Für diese Datenobjekte werden in speziellen Paketen *Views* und dazugehörige *ViewModels* definiert. Diese Pakete werden jeweils für eine bestimmte Plattform erstellt. Ein weiteres Paket (z.B. *Library.Views*) übernimmt die Abstraktion dieser *Views* und gewährleistet eine geräteunabhängige Anbindung an die Anwendung (vgl. Abbildung 31).

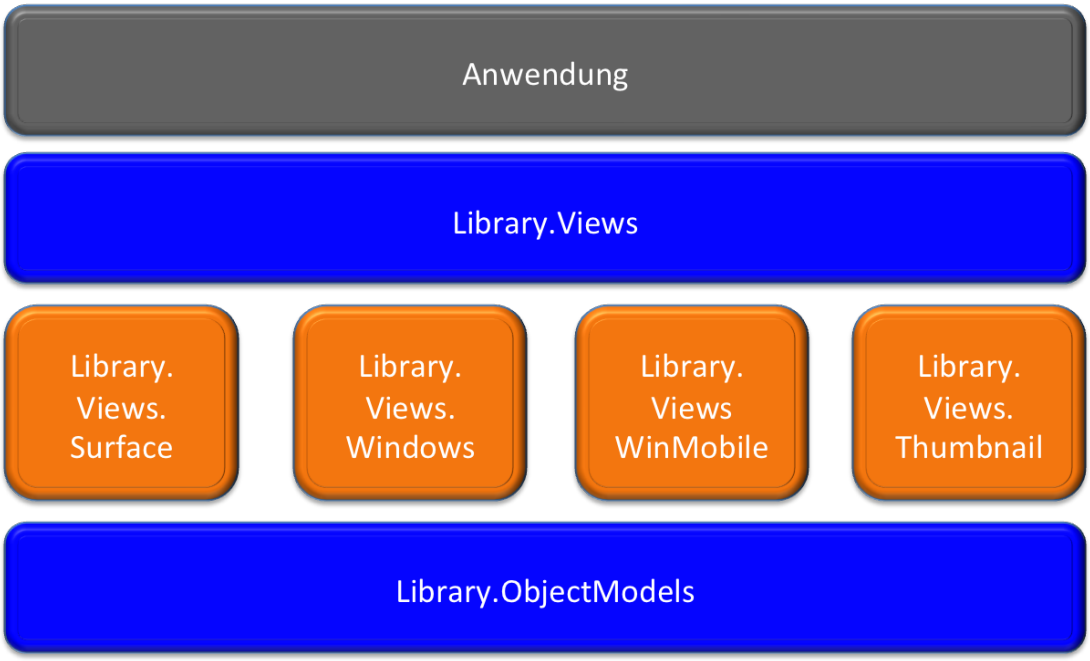


Abbildung 31: *Mögliche Restrukturierung der Domänenobjekte im ZOIL Framework.*

Die Anwendung kann nun auf die *Library.Views* zugreifen und diese in die Anwendung übernehmen. Im Idealfall bekommt der Anwender immer die richtige *View* zugeteilt, je nachdem auf welchem Gerät er arbeitet. Die Datenobjekte und damit die Datenzustände bleiben auf allen Geräten gleich.

Ein Beispiel verdeutlicht die Zusammenhänge. Ein Benutzer benutzt eine ZOIL Anwendung sowohl auf dem Surface, als auch am Laptop. Der Laptop hat kein Surface SDK[27]

[27] Ohne das Surface Software Development Kit können keine Surface Elemente angezeigt werden.

installiert. Wenn die Anwendung nun gestartet wird, fragt sie lediglich nach einer Ansicht für ein Objekt (z.B. *ZView.getView(Movie)*) und erhält die richtige *View* für das verwendete Gerät.

Diese Aufteilung ermöglicht geräteunabhängige Darstellungen indem das Framework den Gerätekontext abstrahiert.

MVVM und XAML Im Rahmen dieser Arbeit wurden zwei Komponenten mit dem MVVM-Pattern realisiert. Um diese in eine Anwendung zu integrieren musste immer der Umweg über prozeduralen Code gegangen werden (siehe Listing 2 und 4). Die einzelnen Teile des Pattern müssen verbunden werden. Dies erhöht die Einstiegshürde (Threshold) und unterbricht den Workflow zwischen Entwickler (C#) und Designer (XAML).

Um dies zu umgehen, wäre eine direkte Integrationsmöglichkeit in XAML für Objekte mit MVVM-Pattern sinnvoll. Dies könnte beispielsweise funktionieren, in dem eine Klasse, beispielsweise *MVVMObject* die Verbindung der einzelnen Teile übernimmt, und je nach Typ und Art der *View* dann die richtige Ansicht erzeugt. In XAML wäre somit die direkte Deklaration möglich (vgl. Listing 7).

```
...
<MVVMObject Type="Movie" View="Default" />
...
```

Listing 7: *Deklaration eines MVVM-Objekts in XAML*

Das ZOIL Framework bietet viele Möglichkeiten ZOIL basierte Anwendungen zu entwickeln. Die schnelle Umsetzung einer *Informationslandschaft*, die Möglichkeit der Navigation auf verschiedenen Geräten und das Angebot an Visualisierungen, Domänenobjekten und weiteren Features machen das Framework zu einem wichtigen Tool zur Entwicklung von Highfidelity-Prototypen. Schnell und effektiv sind neue Anwendungen realisiert.

In diesem Kapitel wurde anhand von Erfahrungen, die während dieser Arbeit gemacht wurden, gezeigt, in welchen Bereichen noch Optimierungsmöglichkeiten für das Framework liegen. Diese würden, vor allem in Richtung Plattformunabhängigkeit und multimodaler Interaktion, Erleichterungen für Anwender und Entwickler mit sich bringen.

Im Hinblick auf die Unterstützung vieler Geräte und der Masse an neuen Interaktionsmöglichkeiten sind solche Schritte wichtig für die Zukunft des ZOIL Framework.

6 Zusammenfassung und Ausblick

Diese Arbeit entwickelt eine grundlegende Systematik für die Gestaltung neuer Benutzerschnittstellen in *multifokalen Arbeitsumgebungen*. Die Analyse relevanter Forschungsarbeiten lieferte den begrifflichen Rahmen für die Konstruktion einer Taxonomie von Benutzerinteraktionen in *multifokalen Arbeitsumgebungen*.

Die Taxonomie schafft ein Grundverständnis der verschiedenen Ausrichtungen, die im Umfeld von *multifokalen Arbeitsumgebungen* angesiedelt sind.

Die Taxonomie abstrahiert von konkreten Aufgaben und ermöglicht die Unterstützung generischer Aktionen in einer *multifokalen Arbeitsumgebung*. Dieses Verständnis versetzt den Entwickler neuer Konzepte in die Lage zu prüfen, welche Aktionen er unterstützen will.

Hat man im Rahmen der Taxonomie, für die verschiedenen Kategorien gute Lösungen gefunden, so können diese auf verschiedene Anwendungsfelder übertragen werden. Ziel ist es, für die einzelnen Felder gute Lösungen zu identifizieren, zu evaluieren und diesen Satz an guten Werkzeugen für gezielte Anwendungen zu verwenden.

Neben diesen theoretischen Grundlagen, wurden für die einzelnen Kategorien der Taxonomie erste Lösungskonzepte entwickelt. Die Konzepte unterstützen alle notwendigen Aktionen in der *multifokalen Arbeitsumgebung*. Die Implementierungen wurden in ein bereits bestehendes Framework so integriert, dass sie möglichst leicht verwendet und weiterentwickelt werden können.

Die so entstandene Klassenbibliothek *Library.MultiFoci* dient als Ausgangspunkt für die Realisierung weiterer Konzepte in ZOIL. Neben den konkreten Komponenten enthält sie technische Grundlagen für Interaktions- und Kommunikationsaufgaben, die innerhalb von *multifokalen Arbeitsumgebung* benötigt werden.

Ausgehend von dieser Basis, können nun weitere Forschungen in diesem Gebiet betrieben werden, die ein tieferes Verständnis kognitiver und sozialer Aspekte in *multifokalen Arbeitsumgebungen* ermöglicht. Nach Meinung des Autors sind vor allem folgende Schwerpunkte lohnende Forschungsperspektiven:

Optimierung der vorgestellten Konzepte Die Konzepte, welche im Rahmen dieser Arbeit vorgestellt wurden, bieten die notwendigen Funktionen, um den Einsatz von ZOIL in einer *multifokalen Arbeitsumgebung* zu ermöglichen. Dennoch bedarf es einer tiefer gehenden Analyse, um einzelne Aspekte zu optimieren.

Um den *Split Screen* flexibler zu gestalten, sollten mehr Layouts möglich sein und eine dynamische Anpassung der Bereiche ermöglichen. Weiter gibt es im Bereich des *mehrfachen Click Zoom* Optimierungsmöglichkeiten. Vor allem die Zuordnung der fokussierten Objekte zu einem bestimmten Bereich ist ein nicht triviales Problem. Mögliche

Ansätze wären eine *optimale Ausnutzung von Bildschirmfläche* und die *die Beachtung des Seitenverhältnisses der Objekte*. Für den *Remote Click Zoom* könnten weitere Interaktionskonzepte erörtert werden, die evtl. eine räumliche Zuordnung ermöglichen. Neue Geräte könnten sich an einer Datenbank anmelden und die Interaktionsmöglichkeiten des *Remote Click Zoom* könnten sich daran anpassen. Die Idee des *Shared Clipboard* könnte durch ein physisches Objekt erweitert werden. Denkbar wäre eine Klemmbrett, das über ein Display verfügt und die Objekte, welche im Clipboard liegen, visualisiert.

Es gilt, diese Ideen zu analysieren und zu verfeinern.

Erweiterung der Taxonomie Die Taxonomie aus Kapitel 3 ordnet drei generische Aufgaben und zwei verschiedene Kontextsituationen in einen Gesamtrahmen ein. Der Rahmen ist über diese Arbeit hinaus erweiterbar. Es könnten sowohl die bestehenden Dimensionen erweitert und darüber hinaus weitere Dimensionen in die Taxonomie integriert werden.

Bei der Erweiterung der generischen Aufgaben ist darauf zu achten, ob die zusätzlichen Aktionen allgemein gültig sind. D.h. es müssten Fragen gestellt werden, wie zum Beispiel: Lässt sich Aktion A vielleicht aus anderen Aktionen ableiten? Gibt es die Möglichkeit Aktion A als Komposition aus B und C zu verwirklichen? Gibt es für die Aktion Unterschiede in einer *multifokalen Arbeitsumgebung* und einer normalen Arbeitsumgebung?

Beispiele könnten das *Erstellen von Inhalt*, das *Gruppieren von Objekten* oder die *Selektion von Objekten* sein.

Neben diesen generischen Aktionen müssten Aktivitäten und Aufgaben analysiert werden, die in *multifokalen Arbeitsumgebungen* eine Rolle spielen. Diesen Aufgaben könnten dann Aktionen zugeordnet werden, aus denen sich die Aufgaben zusammensetzen. Ergebnis könnte eine hierarchische Baumstruktur sein, die in den oberen Ebenen verschiedene Aufgaben definiert und diese gruppiert. Dieser Baum von Aktivitäten endet in den Blättern mit generischen Aktionen, für welche wiederum Lösungen gestaltet werden könnten. Zusätzlich lässt sich eine solche Menge an Aufgaben daraufhin untersuchen, ob sie in verschiedenen Anwendungsdomänen auftauchen und wenn ja, wie sie sich dann systematisieren lassen.

Der Ausschnitt der Taxonomie, die in dieser Arbeit vorgestellt wurde, kann darüber hinaus mit neuen Dimension erweitert werden.

Konfiguration einer multifokalen Arbeitsumgebung Um dem Benutzer ein flexibles und dynamisches Arbeitsumfeld zu schaffen, muss er Möglichkeiten erhalten, die vorhandene Arbeitsumgebung individuell zu konfigurieren. Ein solches Konfigurationstool

muss Funktionen bereitstellen, die es dem Benutzer ermöglichen, für alle Felder der Taxonomie einfache Lösungswege zu gestalten.

Beispielsweise wären für die *synchronisierte Ansicht* oder den *mehrfachen Click Zoom* verschiedene Konfigurationsmöglichkeiten denkbar. Im Rahmen dieser Arbeit wird die *synchronisierte Ansicht* über einen Modus gesteuert. Ebenso könnte der Benutzer eine visuelle Oberfläche erhalten, in denen er die verschiedenen Clients direkt manipulativ verbindet und diese automatisch ihr Verhalten anpassen. Diese würde auch den Einsatz in einem kollaborativen Raum zusätzlich vereinfachen.

Für den *mehrfachen Click Zoom* gilt es weitere Wege zu finden, welche das dynamische Hinzufügen von Geräten und das direkte Verknüpfen mit Personen ermöglichen. Gerade die Einbindung von mehreren Personen stellt einen interessanten Forschungsschwerpunkt dar.

Unterstützung kollaborativer Arbeit Die Unterstützung kollaborativer Arbeit in *multifokalen Arbeitsumgebungen* ist ein breites Feld und hat Überschneidungen mit Disziplinen der Arbeits- und Kognitionspsychlogie oder den Sozialwissenschaften. Die Dimension „Multi-User“ bringt neue Anwendungsfelder und weitere Fragestellungen mit sich.

Zunächst muss ein gemeinsamer Arbeitsbereich geschaffen werden, der sowohl individuelle, wie auch kollaborative Arbeit unterstützt. Dabei ist es nicht ausreichend, dass alle Benutzer auf einen gleichen Datenbestand zugreifen, vielmehr muss der Zugriff einzeln geregelt werden. Es bedarf entsprechender Manifestierung in der visuellen Darstellung solch logisch zusammenhängender Arbeitsbereiche.

Zusätzlich muss die Möglichkeit bestehen, personelle Zusammenhänge im System zu verankern. Wer ist online? Wer ist anwesend? Wer möchte mit Kollegen zusammenarbeiten? Wo ist eine andere Person und wie kann man mit ihr zusammenarbeiten? Wer arbeitet mit welchen Objekten? All dies sind Fragestellungen, die durch die Unterstützung einer Benutzerverwaltung vom System beantwortet werden könnten.

In *multifokalen Arbeitsumgebungen* könnten darüber hinaus Sichten auf spezielle Inhalte realisiert werden. Beispielsweise könnten periphere Displays immer einen gemeinsamen Arbeitsbereich, ein schwarzes Brett oder ähnliches repräsentieren. Kollaborative Sichten, bspw. eine Kalenderansicht liesen sich in den Arbeitsbereich integrieren.

Bei diesen Möglichkeiten spielt ein wichtiger Aspekt eine Rolle. In *multifokalen Arbeitsumgebungen* und insbesondere im Feld von CSCW hat die Wahrnehmung von Aktivitäten anderer Benutzer einen hohen Stellenwert. Mit Hilfe dieser Informationen könnten komplexere Szenarien unterstützt werden. Wenn mehrere Benutzer gleichzeitig an verschiedenen Systemen auf die gleichen Daten Zugriff haben, ist es wichtig, dass sie sich nicht gegenseitig in die Quere kommen. Situationen, in denen Benutzer realisieren,

wer sich noch für bestimmte Objekte interessiert, wären ein weiteres spannendes Anwendungsfeld in diesem Bereich.

Identifizierung verschiedener Display Rollen In den bisher genannten Schwerpunkten haben die Rollen von Displays eine untergeordnete Rolle gespielt. Wie aber schon in Kapitel 2 kurz angesprochen, wurde in verschiedenen Arbeiten festgestellt, dass in heutigen MDEs genau diese Zuteilung in Rollen sehr wichtig ist. So könnte die Taxonomie nicht nur mit verschiedenen Kontexten arbeiten, sondern durch verschiedene Rollen erweitert werden.

Interessant wäre hier der Zusammenhang zwischen einer Display-Rolle und den auftretenden Aktionen. Die Frage lautet: Kann man bestimmten Rollen eine Menge an Aktionen zuweisen? Falls es solche Zusammenhänge gibt, so könnte man diese systematisch einander zuordnen. Gegebenenfalls würden sich diesen Rollen auch Aufgaben oder Benutzertypen zuordnen lassen. Das Resultat wäre ein Netz von Relationen zwischen Aktionen, Aufgaben, Rollen und Benutzern. Eine derartige allgemeine Kategorisierung würde die Analyse *multifokaler Arbeitsumgebungen* zukünftig erleichtern.

Die Rolle eines Displays hat darüber hinaus Einfluss auf die visuelle Darstellung der Informationen. Ein Beispiel wäre eine Informationstafel, die für viele Personen zugänglich ist. Ist es erwünscht, dass auf einer solchen Tafel keine oder nur eingeschränkte Interaktion stattfindet, so könnte die visuelle Darstellung auf die Rolle des Displays angepasst werden. Funktionen und Interface-Elemente würden bei Bedarf reduziert.

Die vorgestellte Restrukturierung in Kapitel 5.3 könnte nicht nur Pakete für spezielle Plattformen, sondern auch Pakete für die Unterstützung verschiedener Rollen oder Benutzertypen bereit halten.

Dieser Ausblick auf mögliche weitere Arbeiten zeigt, wie viel Potenzial in *multifokalen Arbeitsumgebungen* steckt. Die große Anzahl an bestehenden Arbeiten belegen diese Tatsache. Dennoch ist zu wenig bekannt, mit welchen Mitteln diese Umgebungen besonders gut unterstützt werden könnten.

ZOIL ist bereits ein Konzept, mit dem intuitiv große Informationsräume visualisiert und durchsucht werden können. Das Konzept von ZOIL mit weiteren multifokalen Konzepten anzureichern erscheint als sinnvolle und notwendige Erweiterung, um ZOIL für zukünftige Entwicklungen im Bereich von multifokalen Umgebungen zu rüsten.

Aus Sicht des Autors müssten dafür zwei Wege parallel beschritten werden. Auf der einen Seite wäre es notwendig, ein tieferes Verständnis von Anforderungen, Aufgaben und Anwendungsfeldern zu erlangen und dieses zu systematisieren. Recherchen haben gezeigt, dass viele Nischen-Lösungen erforscht wurden. Neuere Arbeiten deuten jedoch an, wie wichtig es ist, systematische Zusammenhänge zu erkennen und zu dokumentieren [Tang and Fels, 2008, Naacenta et al., 2009, Dearman and Pierce, 2008]. Auf der

anderen Seite ist es aber unerlässlich, praktische Umsetzungen in realistischen Szenarien zu schaffen, welche die Überprüfung der erarbeiteten Konzept ermöglichen. Nur mit Hilfe von Evaluationen in konkreten Settings lässt sich untersuchen, welche Konzepte in der Praxis tauglich sind und einen Gewinn an Effizienz bringen.

Es gilt die bereits bestehenden Erkenntnisse zu bündeln, umzusetzen und am lebenden Objekt auszuprobieren.

A Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.

Konstanz, den 27.07.2009

Unterschrift
(Jonas Schweizer)

B Videomaterial

Dieser Arbeit liegt eine CD-ROM bei, welche ein Demonstrationsvideo der Konzepte, die in dieser Arbeit vorgestellt wurden, enthält.

Es sei darauf hingewiesen, dass die Version der Software, die im Video verwendet wird nicht der finalen Version entspricht. Aus diesem Grund weichen manche visuelle Darstellungen von den in der Arbeit gezeigten Screenshots ab.

Das Video ist im .mov Dateiformat gespeichert und ist vorzugsweise mit dem Quicktime Video Player abzuspielen. Getestet wurden darüber hinaus der VLC-Player unter Windows Vista und Mac OS X. Mit dem notwendigen Codec (MPEG-4 Video) können auch weitere Player eingesetzt werden.

Abkürzungsverzeichnis

CSCW	Computer Supported Cooperative Work
DDE	Distributed Display Environment
MDE	Multi-Display Environment
OOUI	Object-oriented User Interface
OSC	Open Sound Control
WIMP	Windows, Icons, Menus, Pointing Device
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language
ZOIL	Zoomable Object-oriented Information Landscape
ZUI	Zoomable User Interface

Glossar

Computer Supported Cooperative Work
Das Forschungsgebiet der Computer Supported Cooperative Work befasst sich mit Gruppen- und Zusammenarbeit und behandelt gleichzeitig die Kommunikations- und Informationstechnologie, die ein gemeinsames Arbeiten unterstützen.

Distributed Display Environment
Im Rahmen dieser Arbeit werden die Begriffe Multi-Display Environment und Distributed Display Environment synonym verwendet.

Extensible Markup Language
Dieser XML-Dialekt bietet in WPF die Möglichkeit deklarativ Benutzerschnittstellen zu programmieren. D.h. ohne Kenntnisse einer üblichen Programmiersprache können Interfaces gestaltet werden. Dies hat zudem den Vorteil, dass ein guter Designer-Entwickler-Workflow möglich ist.

Multi-Display Environment
Unter einer Multi-Display Environment versteht man ein oder mehrere Displays, die an verschiedenen Systemen angeschlossen sind. Die Systeme können untereinander über ein Netzwerk verbunden sein, so dass dadurch eine logisch zusammenhängende Umgebung wird.

Open Sound Control
OSC ist ein nachrichtenbasiertes Kommunikationsprotokoll. Es wird sowohl für Computer, als auch für unterschiedliche Multimediageräte verwendet. Als Netzwerkprotokoll sind UDP und TCP möglich.

System
Die Bezeichnung *System* steht im Rahmen dieser Arbeit für ein Computersystem, bestehend aus Recheneinheit und Display.

WIMP
Heutige Benutzerschnittstellen folgen dem WIMP-Paradigma. Die Oberflächen von Windows, Mac OS X oder Linux bestehen in der Regel aus Fenstern, Icons, Menüs und Eingabegeräten.

Windows Presentation Foundation
WPF ist eine Softwareframework innerhalb des .NET Frameworks der Firma Microsoft. WPF integriert XAML und übliche Programmiersprachen des .NET Framework. Das ZOIL Framework ist mit der WPF entstanden.

ZOIL Framework
Das ZOIL Framework wurde im Rahmen des *permaedia*-Projekts von der Arbeitsgruppe Mensch-Computer-Interaktion an der Universität Konstanz entwickelt. Es bietet die Möglichkeit, schnell High-Fidelity Prototypen zu erstellen, die auf dem grundlegenden Design-Paradigma ZOIL basieren.

Abbildungsverzeichnis

Listings

Literatur

[Balakrishnan and Baudisch, 2009] Balakrishnan, R. and Baudisch, P. (2009). Introduction to this special issue on ubiquitous multi-display environments. *Human-Computer Interaction*, 24:1–8.

[Baudisch et al., 2004] Baudisch, P., Cutrell, E., Hinckley, K., and Gruen, R. (2004). Mouse ether: Accelerating the acquisition of targets across multi-monitor displays. In *CHI*, pages 1379–1382. ACM Press.

[Baudisch et al., 2003] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., T, P., Bederson, B., and Zierlinger, A. (2003). Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems. pages 57–64.

[Bederson and Boltman, 1999] Bederson, B. B. and Boltman, A. (1999). Does animation help users build mental maps of spatial information? In *INFOVIS '99: Proceedings of the 1999 IEEE Symposium on Information Visualization*, page 28, Washington, DC, USA. IEEE Computer Society.

[Bederson et al., 2000] Bederson, B. B., Meyer, J., and Good, L. (2000). Jazz: an extensible zoomable user interface graphics toolkit in java. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 171–180, New York, NY, USA. ACM.

[Biehl et al., 2008] Biehl, J. T., Baker, W. T., Bailey, B. P., Tan, D. S., Inkpen, K. M., and Czerwinski, M. (2008). Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 939–948, New York, NY, USA. ACM.

[Bier et al., 1993] Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. (1993). Toolglass and magic lenses: the see-through interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, New York, NY, USA. ACM.

[Buxton, 1997] Buxton, W. (1997). Living in augmented reality: Ubiquitous media and reactive environments. In *Video Mediated Communication*, pages 363–384. Hillsdale.

[Cockburn et al., 2008] Cockburn, A., Karlson, A., and Bederson, B. B. (2008). A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41(1):1–31.

[Dearman and Pierce, 2008] Dearman, D. and Pierce, J. S. (2008). It's on my other computer!: computing with multiple devices. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 767–776, New York, NY, USA. ACM.

[Donelson, 1978] Donelson, W. C. (1978). Spatial management of information. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 203–209, New York, NY, USA. ACM.

[Duden, 2006] Duden (2006). *Duden - Deutsches Universalwörterbuch*. Bibliographisches Institut, 6. edition.

[Elmqvist et al., 2008] Elmqvist, N., Henry, N., Riche, Y., and Fekete, J.-D. (2008). Mélange: space folding for multi-focus interaction. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1333–1342, New York, NY, USA. ACM.

[Elmqvist et al., 2009] Elmqvist, N., Henry, N., Riche, Y., and Fekete, J.-D. (2009). Mélange: Space folding for visual exploration. *IEEE Transactions on Visualization and Computer Graphics*.

[Engl, 2008] Engl, A. (2008). A framework for an infinitely zoomable information landscape. Bachelor Projekt Dokumentation, University of Konstanz.

[Forlines et al., 2006] Forlines, C., Esenther, A., Shen, C., Wigdor, D., and Ryall, K. (2006). Multi-user, multi-display interaction with a single-user, single-display geospatial application. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 273–276, New York, NY, USA. ACM.

[Grudin, 2001] Grudin, J. (2001). Partitioning digital worlds: focal and peripheral awareness in multiple monitor use. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 458–465, New York, NY, USA. ACM.

[Hutchings et al., 2005] Hutchings, D. R., Stasko, J., and Czerwinski, M. (2005). Distributed display environments. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 2117–2118, New York, NY, USA. ACM.

[Johanson et al., 2002] Johanson, B., Fox, A., and Winograd, T. (2002). The interactive workspaces project: Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1(2):67–74.

[Jul and Furnas, 1998] Jul, S. and Furnas, G. W. (1998). Critical zones in desert fog: aids to multiscale navigation. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 97–106, New York, NY, USA. ACM.

[König, 2006] König, W. A. (2006). *Referenzmodell und Machbarkeitsstudie für ein neues Zoomable User Interface Paradigma*. mastersthesis, University of Konstanz.

[Myers et al., 2000] Myers, B., Hudson, S. E., and Pausch, R. (2000). Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact.*, 7(1):3–28.

[Naacenta et al., 2009] Naacenta, M. A., Gutwin, C., Aliakseyeu, D., and Subramanian, S. (2009). There and back again: Cross-display object movement in multi-display environments. *Human-Computer Interaction*, pages 70–229.

[Perlin and Fox, 1993] Perlin, K. and Fox, D. (1993). Pad: an alternative approach to the computer interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 57–64, New York, NY, USA. ACM.

[Perry and O'Hara, 2003] Perry, M. and O'Hara, K. (2003). Display-based activity in the workplace. In *INTERACT*.

[Plumlee, 2002] Plumlee, M. (2002). Zooming, multiple windows, and visual working memory. In *In Proceedings of AVI 2002*, pages 59–68. ACM Press.

[Reetz et al., 2006] Reetz, A., Gutwin, C., Stach, T., Nacenta, M., and Subramanian, S. (2006). Superflick: a natural and efficient technique for long-distance object placement on digital tables. In *GI '06: Proceedings of Graphics Interface 2006*, pages 163–170, Toronto, Ont., Canada, Canada. Canadian Information Processing Society.

[Rekimoto, 1997] Rekimoto, J. (1997). Pick-and-drop: a direct manipulation technique for multiple computer environments. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 31–39, New York, NY, USA. ACM.

[Rekimoto and Saitoh, 1999] Rekimoto, J. and Saitoh, M. (1999). Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 378–385, New York, NY, USA. ACM.

[Roberts, 2007] Roberts, J. C. (2007). State of the art: Coordinated & multiple views in exploratory visualization. In *CMV '07: Proceedings of the Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71, Washington, DC, USA. IEEE Computer Society.

[Schweizer, 2009a] Schweizer, J. (2009a). Navigation und management multifokaler sichten auf eine informationslandschaft. Bachelor Projekt Dokumentation, University of Konstanz.

[Schweizer, 2009b] Schweizer, J. (2009b). Seminararbeit im bereich des bachelorprojekts state-of-the-art analyse multifokaler sichten auf eine informationslandschaft. Seminararbeit, University of Konstanz.

[Shoemaker and Gutwin, 2007] Shoemaker, G. and Gutwin, C. (2007). Supporting multi-point interaction in visual workspaces. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 999–1008, New York, NY, USA. ACM.

[Stefik et al., 1987] Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S., and Suchman, L. (1987). Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *Commun. ACM*, 30(1):32–47.

[Streitz et al., 1999] Streitz, N. A., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., and Steinmetz, R. (1999). i-land: an interactive landscape for creativity and innovation. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 120–127, New York, NY, USA. ACM Press.

[Tang and Fels, 2008] Tang, A. and Fels, S. (2008). Four lessons from traditional mdes. In *ACM CSCW'08 Workshop: Beyond the Laboratory: Supporting Authentic Collaboration with Multiple Displays*, page electronic proceedings.

[Ware and Lewis, 1995] Ware, C. and Lewis, M. (1995). The dragmag image magnifier. In *CHI '95: Conference companion on Human factors in computing systems*, pages 407–408, New York, NY, USA. ACM.

[Weiser, 1999] Weiser, M. (1999). The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3(3):3–11.

[Wigdor and Keywords, 2009] Wigdor, D. and Keywords, A. (2009). Wespace: The design, development, and deployment of a walk-up and share multi-surface collaboration system. *Media*, pages 1237–1246.

[Wobbrock et al., 2009] Wobbrock, J. O., Morris, M. R., and Wilson, A. D. (2009). User-defined gestures for surface computing. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1083–1092, New York, NY, USA. ACM.

[Woodruff et al., 1998] Woodruff, A., Landay, J., and Stonebraker, M. (1998). Goal-directed zoom. In *CHI '98: CHI 98 conference summary on Human factors in computing systems*, pages 305–306, New York, NY, USA. ACM.

[Zöllner, 2009] Zöllner, M. (2009). Datenbackend für zoil im kontext von pim. Bachelor Projekt Dokumentation, University of Konstanz.