Autorenversion für die Online-Veröffentlichung auf der Website der AG Mensch-Computer Interaktion, FB Informatik & Informationswissenschaft, Universität Konstanz (http://hci.uni-konstanz.de).

Hans-Christian Jetter, Michael Zöllner, Harald Reiterer – *Universität Konstanz*

Gestaltung und Programmierung von interaktiven Räumen mit dem ZOIL-Paradigma

Rainer Groh, Martin Zavesky (Hrsg) – *TU Dresden*

Wieder mehr Sehen! Aktuelle Einblicke in die Technische Visualistik.

TUDpress, Dresden, Germany, Okt 2011. Seiten 105-115.

http://mg.inf.tu-dresden.de/kategorie/art-des-forschungsbeitrags/buch

Hans-Christian Jetter, Michael Zöllner, Harald Reiterer

Gestaltung und Programmierung von interaktiven Räumen mit dem ZOIL-Paradigma

Zusammenfassung

Interaktive Räume, also physische Arbeitsumgebungen mit einer Vielzahl kooperierender Ein- und Ausgabegeräte, sind eine gestalterische und technologische Herausforderung: Wie können interaktive Wände, Tische oder Mobilgeräte für die effiziente Zusammenarbeit mehrerer Benutzer kombiniert werden? Welche Visualisierungs- und Bedienkonzepte mit Multi-Touch, Gesten oder (be)greifbaren Gegenständen sind dafür geeignet? Das ZOIL-Paradigma stellt hierfür Lösungsansätze bereit: Die ZOIL-Designprinzipien beschreiben neue Gestaltungsrichtungen für die Visualisierung und Interaktion in interaktiven Räumen. Das ZOIL-Software-Framework ermöglicht deren technische Realisierung. Beide Instrumente werden hier für die Nachnutzung in Forschung und Praxis vorgestellt.

1 Einleitung

In interaktiven Räumen wie i-Land [13] oder WeSpace [14] werden die physischen und logischen Grenzen einzelner interaktiver Geräte überwunden, um einen nahtlosen Verbund eng kooperierender Hard- und Software zu erschaffen. Dieser Verbund dient dabei in seiner Gesamtheit der computerunterstützten Kooperation zwischen mehreren Benutzern, zum Beispiel in Studios, Ateliers, Besprechungs- oder Kontrollräumen (Abb. 1). Neben den traditionellen WIMP[1]-PC mit Maus und Tastatur treten dabei insbesondere post-WIMP Geräte und Interaktionstechniken, die sich durch ihre natürliche Bedienung nahtlos in unsere physische und soziale Arbeitswelt einfügen sollen. Typische Vertreter sind große hochauflösende Displaywände mit Stift- und Touch-Eingabe oder interaktive Tische, die mit Multi-Touch oder (be)greifbaren Gegenständen – sogenannten *tokens* – bedient werden. Mit Tablet-PCs, Pads, Smartphones oder digitalen Stiften werden dabei auch tragbare persönliche Oberflächen auf Displays oder Papier realisierbar. Oft dienen Kameras zusätzlich zur Gesten- und Positionserkennung. Während diese Basistechnologien bereits weit verbreitet sind, stellen sich immer noch grundlegende Fragen zu deren Integration in ein raumweites Bedienkonzept: Wie müssen Funktionalität und Inhalte gestaltet und programmiert werden, damit Benutzer diese auf „natürlichem“ Wege nutzen, im Raum teilen und verteilen können? Wie kann eine flexible und effiziente *ad-hoc*-Kooperation ermöglicht werden, die ohne eine aufwändige vorherige Planung von Arbeitsschritten, Geräten, Speicherorten und -formaten auskommt?

Das hier vorgestellte ZOIL-Paradigma (**Z**oomable **O**bject-Oriented **I**nformation **L**andscape) stellt einen Ansatz zur Beantwortung dieser Fragen aus der Sicht der technischen Visualistik, der Mensch-Computer-Interaktion und der Softwaretechnik bereit: Die sechs ZOIL-Designprinzipien beschreiben neue Gestaltungsrichtungen für die Visualisierung und Interaktion. Deren softwaretechnische Realisierung wird durch das ZOIL-Software-Framework ermöglicht. Beide Instrumente werden hier vorgestellt und sind für die Nachnutzung in Forschung und Praxis quelloffen im Internet verfügbar.

[1] Als WIMP („**W**indows **I**cons **M**enus **P**ointer“) wird hier das heute dominante Design für das Graphical User Interface (GUI) des PCs bezeichnet, das durch Mausbedienung, Anwendungsfenster und die Desktop-Metapher mit „Dateien“ und „Ordnern“ gekennzeichnet ist.

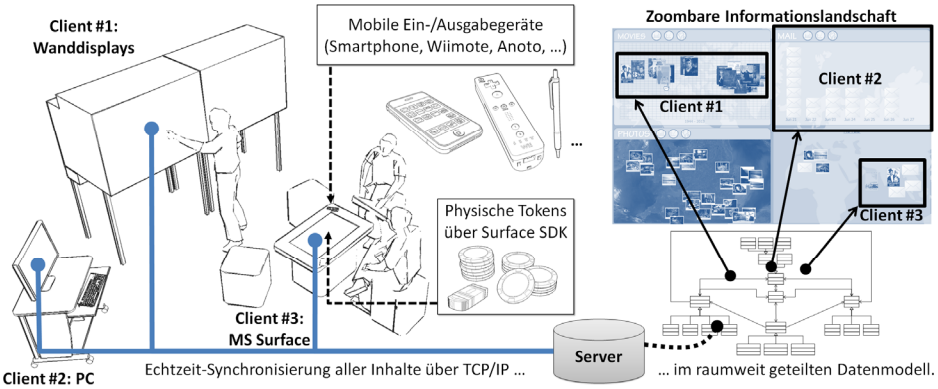

Abbildung 1: Hard- und Software-Architektur eines auf ZOIL basierenden interaktiven Raumes: einzelne Clients (links) und deren Sichten auf die gemeinsame Informationslandschaft (rechts).

2 Das ZOIL-Paradigma

Kernkomponente des ZOIL-Paradigmas für die Gestaltung und Programmierung interaktiver Räume ist ein gemeinsamer virtueller Arbeitsbereich, der über eine Client-Server-Architektur von allen Geräten im Raum gleichzeitig und in Echtzeit zugreifbar und manipulierbar ist. Dieser Arbeitsbereich dient als Rückgrat der benutzer- und geräteübergreifenden Kooperation und wird als zoombare objekt-orientierte Informationslandschaft bezeichnet. Ihre visuelle Erscheinung und Logik folgt dabei den Ansätzen von Perlin & Fox [10], Raskin [11] und Reiterer et al. [12]: Sie ist eine aus der Draufsicht betrachtete Ebene, die einer virtuellen Pinnwand unendlicher Größe und Auflösung ähnelt. Alle Inhalte, zum Beispiel Objekte wie Dokumente, Fotos, Diagramme oder Karten, und die notwendigen Werkzeuge zu deren Bearbeitung sind visuell in die Landschaft integriert. Sie können dazu an beliebigen Orten und in beliebiger Größe in der Landschaft platziert sein. Alle Objekte können per Zoom vergrößert und dann direkt an Ort und Stelle betrachtet oder bearbeitet werden, ohne Umwege über Applikationen, Anwendungsfenster oder Dateihierarchien.

Der Zugriff auf die Objekte und deren Funktionen kann von jedem Gerät oder Client aus erfolgen, indem der dargestellte Landschaftsausschnitt mittels gängiger Maus-, Stift- oder Multi-Touch-Interaktion vergrößert und verschoben wird (*zooming* und *panning*). Alle im Raum integrierten Geräte fungieren also als eine Art freibewegliche Kamera, die einen gewünschten Ausschnitt der Landschaft aus der Draufsicht darstellt. Abb. 1 (rechts) illustriert, wie drei Clients unterschiedliche Ausschnitte aus der gemeinsamen Informationslandschaft darstellen. Diese Kamera-Ausschnitte können dabei vom Benutzer für jeden Client völlig frei gewählt werden. Werden aber vom Benutzer Veränderungen an den eigentlichen Positionen, Inhalten oder Zuständen von Objekten innerhalb der Landschaft vorgenommen, dann werden diese in Echtzeit über den Server mit allen anderen Clients beziehungsweise Benutzern synchronisiert. Diese Eigenschaft ist von zentraler Bedeutung, da sie die Nutzung der Landschaft als kooperatives Medium erst ermöglicht. Aufbauend auf diesem Grundgedanken wird im Folgenden die weitere Gestaltung eines interaktiven Raums mit ZOIL anhand der sechs ZOIL-Designprinzipien präzisiert.

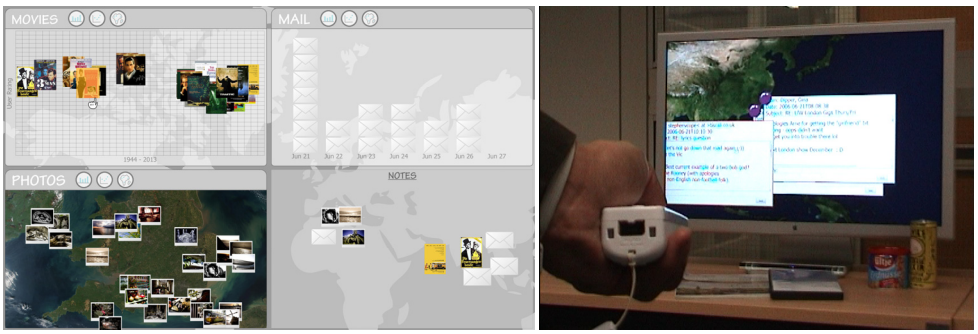

Abbildung 2: ZOIL für E-Mails, Fotos und Filme auf HDTV-Geräten mit Wiimote Bedienung [6].

2.1 Objekt-orientierte Modellwelten für die direkte Manipulation

Um das Erlebnis einer möglichst direkten und natürlichen Interaktion mit den angebotenen Inhalten und Funktionen zu erzielen, beruft sich das erste ZOIL-Prinzip auf die bekannten kognitiven Modelle der *direct manipulation* [4] und wendet diese auf post-WIMP Interaktion an [8]: Prinzipiell ist die Benutzungsschnittstelle als eine Modellwelt zu gestalten, in der alle Inhalte und Funktionen der Anwendungsdomäne über visuelle und (be)greifbare Objekte kontinuierlich repräsentiert werden. Die Funktionen und Zusammenhänge aus der Anwendungsdomäne werden direkt in diesen Objekten und in deren Erscheinung, Verhalten und logischen Beziehungen zueinander in der Modellwelt abgebildet. Die Benutzung erfolgt nicht durch eine formale Konversation zwischen Benutzer und System entlang festgelegter Pfade aus Hyperlinks, Menüs, Seiten und Dialogfenstern, sondern durch unmittelbares und direktes Eingreifen des Benutzers in die Modellwelt. Er gibt dem System keine indirekten Anweisungen, sondern handelt selbst. Dies führt zur kognitiven Entlastung der Benutzer, da der *gulf of evaluation* (der Abgleich des aktuellen Systemzustands mit den Benutzerzielen) und der *gulf of execution* (die Entscheidung für die nächsten Handlungen und deren Formulierung und Ausführung) stark vereinfacht werden [4].

Anders als bei WIMP-Systemen ist bei ZOIL diese Modellwelt nicht auf den Bildschirm beschränkt, sondern umfasst den gesamten interaktiven Raum mit virtuellen visuellen Objekten auf den Bildschirmen (zum Beispiel Piktogramme oder Miniaturansichten) und physischen Objekten vor den Bildschirmen (zum Beispiel digitale Stifte oder *tokens* auf Tabletops und RFID-Lesern). Die Interaktion erfolgt also nicht nur im virtuellen, sondern auch im physischen Raum durch Bewegung und Gesten [6] oder direkte und kontinuierliche Manipulation der Positionen und Konfigurationen physischer Gegenstände [7,8]. Dabei wird die Systemreaktion unmittelbar sicht- und spürbar und die eigentliche Benutzungsschnittstelle tritt gegenüber der erlebten Modellwelt in den Hintergrund. Natürliches Interaktionsdesign fokussiert den Benutzer so auf seine Inhalte und Ziele anstatt auf Schnittstellen. Unter dem Motto „the content is the interface“[2] wird daher auch eine Abkehr vom administrativen „Schutt“ oder Ballast heutiger GUIs gefordert, also eine Abkehr von der Vielzahl administrativer Schaltflächen und Kontrollelemente, zum Beispiel für Fenstermanagement, Navigation oder Modusauswahl.

[2] “The idea is that the content is the interface, the information is the interface – not computer administrative debris.“, Edward Tufte „iPhone interface design“, http://www.edwardtufte.com.

Die Regeln der Modellwelt und das Verhalten ihrer physischen und virtuellen Objekte müssen sich dabei nicht zwangsweise an realen Vorbildern orientieren, wie es der *desktop* heutiger GUIs mit der Abbildung physischer Gegenstände aus dem Büroalltag versucht (Schreibtisch, Ordner, Papierkorb). Solche überkonkreten Metaphern sind sogar häufig problematisch, da sie beim Benutzer falsche Erwartungen und Vermutungen gegenüber der Funktionsweise auslösen. Kognitiv sinnvoll ist es aber, dass sich elementarere Erfahrungen der Benutzer aus dem nicht-digitalen Alltag in der Modellwelt wiederfinden. Beispielsweise betont die *reality-based interaction* [5] den Nutzen der Imitation von vertrauten physikalischen Konzepten wie Masse, Reibung oder Objektpermanenz für eine direktere und effizientere Interaktion. Auch die Verwendung von 2D- oder 3D-Räumen kann von Vorteil sein, um unsere natürlichen Fähigkeiten zur räumlichen Navigation und Erinnerung nutzbar zu machen. Naiver Realismus ist dabei aber kein Allheilmittel, sondern muss vorsichtig abgewogen werden. Beispielsweise dürfen Benutzer während der Kooperation nicht im Interesse des Realismus von ihrer eigentlichen physischen und sozialen Arbeitsumgebung isoliert werden, zum Beispiel durch besonders immersive VR-Helme oder CAVE-Systeme.

Von entscheidender Bedeutung für den Benutzer ist dabei nicht nur die Konsistenz und Kompatibilität der Modellwelt mit realweltlichen Erfahrungen, sondern auch die interne logische Konsistenz der Modellwelt während der Interaktion. Hassenzahl bezeichnet die „Glaubwürdigkeit“ der Modellwelt durch „Konsistenz und Aufmerksamkeit für Details“ als ein entscheidendes Prinzip, das Gestalter von Computerspielen erlernen können [3]. Das Verhalten von Objekten muss innerhalb der gesamten Modellwelt für die Benutzer vorhersehbar und logisch sein. Dies erlaubt eine effizientere Ausführung bereits erlernter Arbeitsschritte, aber insbesondere auch eine schnellere Übertragung von erlerntem Wissen auf neuartige Aufgabenstellungen und Inhalte. Interne Konsistenz erleichtert also den Benutzern die Einarbeitung und Improvisation. Zur Erreichung von interner Konsistenz bedient sich ZOIL einer systematischen Herangehensweise bei der Gestaltung der Modellwelt, die auf den heute in Vergessenheit geratenen Prinzipien der objektorientierten Benutzungsschnittstellen (OOUIs) der 1990er Jahre basiert [9]. In [8] wird dargelegt, wie die Techniken und Notationen der objekt-orientierten Programmierung (zum Beispiel Vererbung, Polymorphismus, UML) zur Gestaltung konsistenter Verhaltensweisen und Erscheinungsbilder von Modellwelten angewendet werden können. Dabei sind OOUIs im Sinne von ZOIL nicht mehr auf die GUI- und WIMP-Interaktion der 1990er beschränkt, sondern umfassen auch moderne Formen der post-WIMP Interaktion.

2.2 Semantisches Zooming

Zweites Prinzip von ZOIL ist die Verwendung von *zoomable user interfaces* (ZUI) mit *semantic zooming* [10]. Gerade vor dem Hintergrund der vorher erwähnten *reality-based interaction*, bieten ZUIs eine sehr eingängige und direkte Form der Informationspräsentation und Navigation, die durch elementare Alltagserfahrungen des Benutzers getragen wird: Auch im Alltag bewegen wir uns im Raum und nähern uns Objekten, um sie zu erkennen und zu nutzen. Zur Orientierung dienen uns dabei visuelle Landmarken und zuvor erlernte räumliche Strukturen. Anders als bei den „labyrinth-artigen“ [11], flüchtigen und überlappenden Anordnungen der Fenster und Seiten in heutigen GUIs, etablieren ZUIs eine bedeutungsvollere und stabilere räumliche Konfiguration ähnlich einer Landkarte. In Kombination mit einem semantischen anstelle eines rein geometrischen Zooms – also durch Optimierung und Austausch der Inhalte und

Funktionen nach verfügbarem Bildschirmplatz (Abb. 3) – gelingt es ZUIs über die zwei Handlungsprimitive *zooming* und *panning* selbst komplexe Informations- und Funktionsräume auf natürliche Weise navigierbar und erfassbar zu machen. Werden mehr Informationen oder Funktionen zu einem Objekt gewünscht, genügt ein Zoom bis das Objekt den Bildschirm füllt und vollständig les- und bearbeitbar wird. Die Inhalts- und Funktionsangebote passen sich dabei dem stetig wachsenden Darstellungsplatz an. In Abb. 3 kann beim Filmobjekt über Zwischenstufen bis in den eigentlichen Videostream gezoomt werden. Beim Powerpoint-Objekt erscheinen nach und nach farbige Annotationen und Werkzeugpaletten. Aufgrund dieses stufenlosen Übergangs von Piktogrammen über Metadaten in Volltexte und Bearbeitungsfunktionen sind ZUIs besonders geeignet für eine natürlichere Interaktion ohne den administrativen Mehraufwand heutiger GUIs wie Fenster- und Dateimanagement. Weiterhin entspricht der semantische Zoom grundlegenden OOUI-Prinzipien, da die Funktionalität nicht über rigide Strukturen und Dialogfolgen einer Applikation vordefiniert wird, sondern an wahlfrei verwendbare Objekte angeheftet ist („flexible structure-by object, instead of rigid structure-by function“, [9]).

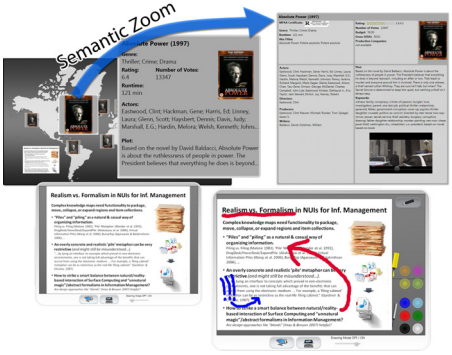

Abbildung 3: Semantischer Zoom in ein Film-Objekt (oben) und ein Powerpoint-Objekt (unten).

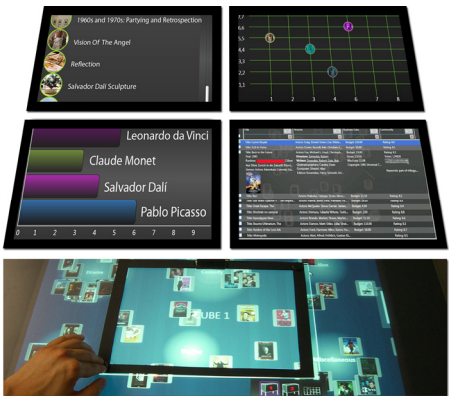

Abbildung 4: Visualisierungen wie *bar charts*, *scatter plot*, HyperGrid [12] (oben). Physische *magic lens* auf MS Surface (unten).

2.3 Visuelle analytische Werkzeuge

Angesichts des Umfangs heutiger Informationsräume sind Suche, Filterung und Analyse auf großen Mengen von Objekten unentbehrlich. Zu diesem Zweck fordert das dritte ZOIL-Prinzip das Angebot visueller Werkzeuge zur Informationsvisualisierung, die analytische Sichten auf die darunterliegenden Objekte innerhalb einer Region der Informationslandschaft ermöglichen (Abb. 4). Visualisierungen wie Diagramme, Landkarten oder zoombare Tabellen [12] verwenden dazu die Metadaten der enthaltenen Objekte (zum Beispiel Autor, Datum, geografische Position), um die Darstellung und räumliche Verteilung der Objekte nach den gewünschten Gesichtspunkten oder Filterkriterien zu verändern. Diese Visualisierungen können dabei temporär in frei beweglichen *magic lenses* [2] erzeugt werden, die per Multi-Touch oder mit physischen Objekten steuerbar sind (Abb. 4, unten). Hilfreiche Visualisierungen können als visuelle Einstiegspunkte für die spätere Verwendung auch direkt in die Informationslandschaft integriert werden (Abb. 2 und 5). ZOIL vereinigt so die gezielte Navigation mit eher interessengeleitetem Stöbern und analytischer Informationsvisualisierung und -suche.

Abbildung 5: Eine Landkarte in ZOIL als ortsbezogener Einstiegspunkt in eine persönliche Fotosammlung [6].

2.4 Raum für Analyse & Annotation

Andrews et al. haben die Nutzung von Raum beziehungsweise Bildschirmplatz durch den Benutzer während komplexer Aufgaben analysiert [1]. Sie beobachteten, dass Raum eine vielfältig eingesetzte Ressource während kognitiv-belastender Aufgaben wie Analyse, Reflexion oder *sensemaking* ist. Beispielsweise werden zentrale Arbeitszonen etabliert, in denen Objekte verwendet, gelesen oder verändert werden, während periphere Zonen eher der Zwischenspeicherung von Objekten dienen. Raum wird auch als externes Gedächtnis verwendet, um Objekte durch Position und Größe hervorzuheben und den Zugriff darauf zu beschleunigen. Eine weitere Nutzung von Raum sind „inkrementelle Formalismen“ [1]: Zunächst informell oder chaotisch anmutende Verteilungen von Objekten werden im Verlauf des *sensemaking* durch die Benutzer in zunehmend formalere Ordnungen überführt, beispielsweise durch fortschreitende Sortierung. Der Raum dient hier als Medium in dem informelle provisorische Anordnungen iterativ zu sinnhaften visuellen Formalismen weiterentwickelt werden, zum Beispiel zu Zeitstrahlen, Hierarchien oder Netzwerken. Das vierte ZOIL-Prinzip fordert daher genug frei verfügbaren Raum: In der Informationslandschaft sollen Inhalte und Funktionalitäten jederzeit frei platziert und arrangiert werden können.

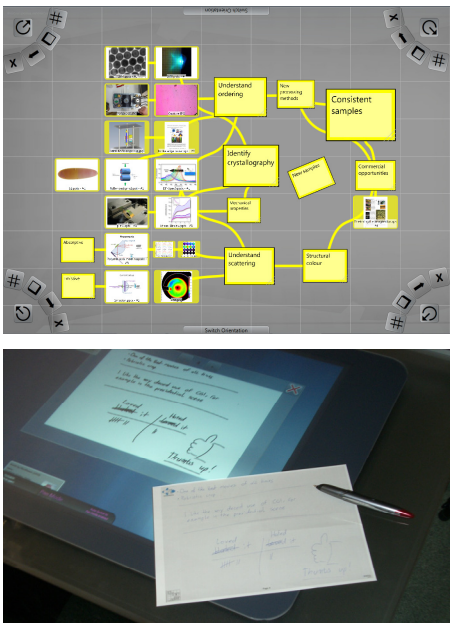

Abbildung 6: Sensemaking für wissenschaftliche Inhalte in ZOIL mit „DeskPiles“[3] (oben). Digitale Anoto-Stifte übertragen Notizen auf Papier in virtuelle Post-It-Notes (unten).

Die Informationslandschaft soll dabei auch Raum für Annotation und informelle Kommentare bieten. Prinzipiell sind daher alle Regionen der Landschaft wie bei einem realen Whiteboard mit Zeichnungen, Handschrift oder virtuellen Post-It-Notes annotierbar. Annotationen können dabei direkt auf der Landschaft, aber auch auf Objekten stattfinden (Abb. 3, unten) und nutzen Eingabemodalitäten wie Touch, Stylus auf Tablet-PCs oder digitale Anoto-Stifte auf Papier (Abb. 6). Durch Handschrifterkennung wird dabei auch die maschinelle Verarbeitung ermöglicht, zum Beispiel die Stichwortsuche in Annotationen.

[3] DeskPiles ist ein Kooperationsprojekt mit Microsoft Research Cambridge und der Universität Cambridge. http://research.microsoft.com/en-us/projects/deskpiles

Abbildung 7: DeskPiles in einer Multi-Geräte Umgebung mit Tabletop und Wanddisplays.

2.5 Erlaube Multi-Geräte-Interaktion

Theoretisch ermöglichen die vielen Bauformen heutiger Rechner von Tablet-PC bis Tabletop eine spontane aufgabenspezifische Wahl der zu verwendenden Hardware. In der Praxis ist jedoch der Wechsel zwischen verschiedenen Geräten oder deren gleichzeitige Nutzung aufwändig und verlangt insbesondere die zeitintensive Portierung, Übertragung und Installation von Inhalten und Funktionen. Um eine nahtlose Integration verschiedenster Geräte in den Arbeitsprozess zu ermöglichen, fordert das fünfte ZOIL Prinzip daher eine Client-Server-Architektur mit einer Echtzeit-Synchronisierung über Gerätegrenzen hinweg (Abb. 1 und 7). Dabei findet diese Synchronisierung nicht auf der „Pixel-Ebene" statt, wie zum Beispiel durch Duplizierung von Bildschirminhalten zwischen Geräten mittels Videokabeln oder Remote-Desktop-Protokollen. Stattdessen erfolgt sie auf der Ebene eines gemeinsamen Datenmodells der Informationslandschaft, das von allen Clients geteilt, aber einzeln visualisiert wird.

Diese Architektur eröffnet zahllose Möglichkeiten für die Arbeit in interaktiven Räumen. Individuelle Geräteeigenschaften, wie zum Beispiel die Mobilität und die hochauflösende drucksensitive Stifteingabe von Tablet-PCs, können gezielt eingebracht werden (Abb. 8). Dabei kann jedes Gerät als Kamera dienen, die einen individuellen Ausschnitt aus der Informationslandschaft darstellt und zur Bearbeitung mit den geräteeigenen Möglichkeiten bereitstellt. Einzelne Geräte können sowohl der freien Navigation dienen, als auch eine bestimmte Region fixieren, beispielsweise die „Todo"-Liste in Abb. 8 (oben), die als zentraler Ort für die Sammlung von Gruppenergebnissen oder Protokollen dient. Neben der Nutzung als individuelle Kameras, können mehrere Geräte auch eng aneinander gekoppelt werden: So kann zum Beispiel die physische *magic lens* in Abb. 4 auf dem Tabletop auch als Fernsteuerung eines weiter entfernten großen hochauflösenden Wanddisplays dienen. Der Tabletop fungiert dabei als *overview*, das Wanddisplay als *detail view*, das immer den Ausschnitt innerhalb der *magic lens* hochauflösend darstellt. Die ZOIL-Architektur ermöglicht somit also flexible Konfigurationen von Geräten, um individuelle Arbeitsstile und Benutzerziele besser zu unterstützen.

Abbildung 8: Wanddisplay als „Todo"-Liste (oben). Tablet-PCs dienen als persönliche Arbeitsbereiche um einen Tabletop (unten).

Abbildung 9: Multi-User-Interaktion mit Facet-Streams [7].

2.6 Erlaube Multi-User-Interaktion

Durch die Client-Server-Architektur können Benutzergruppen mit mehreren persönlichen Geräten in der Informationslandschaft kooperieren. Anders stellt sich die Situation dar, wenn mehrere Benutzer gleichzeitig mit einem Gerät interagieren möchten, wie dies typischerweise bei großen Wanddisplays oder Tabletops der Fall ist (Abb. 9). Facet-Streams ist ein Beispiel für ein ZOIL-basiertes System, das speziell für die Kooperation um einen Tabletop entworfen wurde [7]. Die Benutzer können facettierte Boolesche Suchanfragen gemeinsam formulieren. Dazu dient eine visuelle Filter-Flow-Repräsentation der Suchanfrage, die über Multi-Touch-Interaktionen und physische *tokens* aus Glas von jedem Benutzer direkt manipulierbar ist. Durch die Darstellung der Suchanfrage als Netzwerk aus gläsernen „Spielsteinen“ können Teilaspekte jederzeit von einzelnen Benutzern oder kleinen Gruppen isoliert und weiterbearbeitet werden. Basierend auf den Erfahrungen mit Facet-Streams, wurden drei Gestaltungsziele für die Multi-User-Interaktion in ZOIL formuliert: 1.) Hohe Flexibilität in den unterstützen Arbeitsstilen durch nahtlose Wechsel zwischen Phasen eher lose-gekoppelter paralleler Arbeit und Phasen eng-gekoppelter Kooperation. 2.) Erhöhte *awareness* in der Benutzergruppe durch eine gemeinsame physische und/oder visuelle Externalisierung der gemeinsamen Aufgabe. 3.) Gleichberechtigte und parallele Interaktion durch die simultane Verarbeitung mehrere Eingaben mit Multi-Touch oder *tokens* und eine Gestaltung ohne feste Vorzugs- oder Leserichtungen. Weitere Details hierzu werden in [7] diskutiert.

3 Architektur des Frameworks

Nach dieser Betrachtung der gestalterischen Aspekte von ZOIL folgt hier ein Einblick in deren softwaretechnische Umsetzung. Das ZOIL-Framework ist eine Sammlung von Klassen in C# und XAML mit ca. 4.000 *lines of code* für Microsofts .NET Plattform und die Windows Presentation Foundation (WPF). Die Kernaufgaben von ZOIL sind eine effiziente ZUI-Programmierung und eine Client-Server-Architektur für die Synchronisierung und Persistierung von ZUIs in Echtzeit. Weiterhin vereinfacht ZOIL die Verwendung der Multi-Touch- und Objekt-Erkennnung des Microsoft Surface SDK und nutzt Open Sound Control für die Client-Client-Kommunikation. Das Framework ist dabei quelloffen im Internet unter *http://zoil.codeplex.com* für eine beliebige Nachnutzung verfügbar. Da hier nicht alle Aspekte der ZOIL Implementierung vorgestellt werden können, beschränkt sich diese Einführung auf die ZUI-Programmierung und die Client-Server-Architektur. Weitere Informationen sind auf der Webseite des Open-Source-Projekts einsehbar.

(1) Die verteilte zoombare Informationslandschaft wurde aufgrund zweier Anforderungen mit WPF realisiert: Wie in Abb. 10 deutlich wird, ist ein vektor-basiertes Rendering von großer Bedeutung für die Lesbarkeit von ZUIs. Dies ist insbesondere der Fall, wenn die physischen Auflösungen (*pixels per inch*) im Raum stark voneinander variieren, beispielsweise bei der gleichzeitigen Verwendung von großen Wandprojektionen und hochauflösenden Tablet PCs. Ebenfalls kritisch ist die Rendering-Leistung: Ohne Hardwareunter-

stützung erscheinen Zooming-Animationen nicht flüssig. Dies beeinträchtigt deren Nachvollziehbarkeit und damit auch die visuell-räumliche Orientierung in der Landschaft. Das ZOIL- Framework greift daher auf die Palette vektor-basierter und hardware-beschleunigter Steuerelemente von WPF zurück. Neben einfachen Schaltflächen stehen so auch komplexe Kontrollelemente für Videowiedergabe oder für Zeichnen oder Texteingabe mittels Stift zur Verfügung. Weiterhin kann das Surface SDK nahtlos in WPF integriert werden.

```
<ZEMail x:Class="ZComponent">
 DragDropBehavior.IsDraggable="True"
 ResizeBehavior.IsResizable="False"
 RotateBehavior.IsRotatable="True"
 ZInfLandscape.ZoomTarget="True"
 ZInfLandscape.ZoomMargin="10">
<!-- define content of ZEMail here-->
</ZEMail>
```

```
<ZEmail x:Class="ZComponent" … >
 <ZComponentFrames>
  <ZComponentFrame WidthNeeded="0">
   … content at 1st zoom level …
  </ZComponentFrame>
  <ZComponentFrame WidthNeeded="150">
   … content at 2nd zoom level …
  </ZComponentFrame>
  … content for further zoom levels …
 </ZComponentFrames>
</ZEmail>
```

Abbildung 10: Bitmap-Zoom vs. Vektor-Zoom (oben). Verhaltensweisen eines E-Mail Objektes (mitte). Semantischer Zoom (unten).

Ein weiterer Vorteil von WPF liegt in der Definition von Aussehen und Verhalten des ZUI mit dem XML-Dialekt XAML und dem *Attached Behavior Pattern*[4]. Damit kann das interaktive Verhalten eines Objektes definiert werden, indem ihm vordefinierte Verhaltensweisen oder Eigenschaften aus einer Bibliothek zugewiesen werden. Objekte können so mit minimalem Aufwand als „vergrößerbar", „verschiebbar" oder „rotierbar" definiert werden (Abb. 10, oben). Auch für die Definition des semantischen Zooms stehen deklarative Ansätze bereit. In Abb. 10 (unten) wird die visuelle Erscheinung als Funktion des verfügbaren Bildschirmplatzes modelliert.

(2) Für die Client-Server-Architektur mit Echtzeit-Synchronisierung ist eine Trennung der individuellen *view* eines einzelnen Clients von dem gemeinsamen *model* der Informationslandschaft auf dem Server notwendig. Zu diesem Zweck werden zwei Entwurfsmuster eingesetzt. Erster Schritt ist die Trennung der visuellen Instanzen oder *views* der Landschaftsobjekte von ihrem lokalen Datenmodell im Hauptspeicher eines Clients. Dazu wird das *Model-View-ViewModel* Pattern[5] verwendet, das in der Tradition des *Model-View-Controller* Pattern steht. Es trennt die Daten und prozedurale Anwendungslogik eines Objektes in C# von dessen visuellem Erscheinungsbild und Verhalten in XAML. Im zweiten Schritt muss das im lokalen Hauptspeicher vorgehaltene *model* bidirektional mit dem auf dem Server und allen anderen Clients synchronisiert werden. ZOIL nutzt dazu einen eigenen Server auf der Basis der Objektdatenbank db4o und deren client-seitiges *Transparent Persistence* Pattern. Alle an den lokalen Objekten im Hauptspeicher gemachten Ände-

[4] The Attached Behavior Pattern. http://blogs.msdn.com/johngossman/archive/2008/05/07/the-attached-behavior-pattern.aspx

[5] Introduction to Model/View/ViewModel pattern for building WPF apps. http://blogs.msdn.com/johngossman/archive/2005/10/08/478683.aspx

rungen werden dabei automatisch und für den Entwickler völlig transparent an den Server gesendet, dort persistiert und dann mit allen angeschlossenen Clients synchronisiert.

4 Fazit

Anhand der sechs ZOIL-Gestaltungsprinzipien und der Architektur des ZOIL-Software-Frameworks wurde illustriert, welche neuen Ansätze für die Gestaltung und Programmierung interaktiver Räume existieren und in Zukunft weiterverfolgt werden können. Das ZOIL-Paradigma demonstriert dabei die Realisierung neuer post-WIMP Arbeitsumgebungen durch die Kombination von zoombaren Benutzungsschnittstellen, Informationsvisualisierung und post-WIMP Eingabemodalitäten bei konsequenter Ausnutzung neuer Geräteformen und aktueller Entwurfsmuster der Softwaretechnik. ZOIL ist dabei nicht nur ein Modell und „graue Theorie“: Es ist in seinen Ansätzen durch Anwendungsbeispiele bestätigt worden und ist als quelloffenes Werkzeug, das in Zukunft noch weiter verbessert und erweitert werden wird, für die Nachnutzung in Forschung, Lehre und Industrie verfügbar. ZOIL kann somit einen Beitrag zu einer direkteren, natürlicheren und aufgabenangemessenen Interaktion und Kooperation in den interaktiven Räumen der Zukunft leisten.

Literaturverzeichnis

[1] C. Andrews, A. Endert, und C. North. „Space to think: large high-resolution displays for sensemaking.” in *Proc. CHI '10*. 2010, S.55-64.

[2] E. A. Bier, M. C. Stone, et al. „Toolglass and magic lenses: the see-through interface.” in: *Proc. SIGGRAPH '93*, 1993, S.73-80.

[3] M. Hassenzahl. „Attraktive Software – Was Gestalter von Computerspielen lernen können“ in *User Interface Tuning*. Frankfurt: Software & Support Verlag, 2003, S.27-45.

[4] E. L. Hutchins, J. D. Hollan und D. A. Norman. „Direct manipulation interfaces”. *Hum.-Comput. Interact.* 1, 4, S.311-338, 1985.

[5] R. J. Jacob, A. Girouard, L.M. Hirshfield, M.S. Horn et al. „Reality-based interaction: a framework for post-WIMP interfaces.” in *Proc. CHI ‘08*, 2008, S.201-210.

[6] H.-C. Jetter, A. Engl, S. Schubert und H. Reiterer. „Zooming not Zapping: Demonstrating the ZOIL User Interface Paradigm for ITV Applications.” in *Adj. Proc. Euro-ITV ’08*, 2008, S.239-240.

[7] H.-C. Jetter, J. Gerken, M. Zöllner, H. Reiterer und N. Milic-Frayling. „Materializing the Query with Facet-Streams – A Hybrid Surface for Collaborative Search on Tabletops.“ in *Proc. CHI '11*, 2011.

[8] H.-C. Jetter, J. Gerken, M. Zöllner und H. Reiterer. „Model-based Design and Prototyping of Interactive Spaces.” in *Proc. of HCSE 2010*, 2010. S.22-37.

[9] T. Mandel. *The GUI-OOUI War, Windows vs. OS/2: the designer's guide to human-computer interfaces*. New York: Van Nostrand Reinhold, 1994.

[10] K. Perlin und D. Fox. „Pad: an alternative approach to the computer interface.” in *Proc. SIGGRAPH 93*, 1993, S.57-64.

[11] J. Raskin. *The Humane Interface: New Directions for Designing Interactive Systems*. Addison-Wesley, 2000.

[12] H. Reiterer, H.-C. Jetter, W. König, C. Grün. „Zoomtechniken zur Exploration komplexer Informationsräume am Beispiel HyperGrid.“ in *Mensch und Computer 2005*, 2005, S.143-153.

[13] N.A. Streitz, J. Geißler, et al. “i-LAND: an interactive landscape for creativity and innovation.” in *Proc. CHI '99*. 1999, S.120-127.

[14] D. Wigdor, H. Jiang, C. Forlines, M. Borkin, und C. Shen. “WeSpace: the de-

sign development and deployment of a walk-up and share multi-surface visual collaboration system.” in *Proc. CHI '09*, 2009, S.1237-1246.