Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Seminararbeit

Oliver Köckritz

Verteilter Desktop mit Remote-Windows für Collaborative Workspaces

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Oliver Köckritz

Verteilter Desktop mit Remote-Windows für Collaborative Workspaces

Seminararbeit, 3. Semester im Fach Seminar Ringvorlesung
im Studiengang Master of Computer Science
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Professor : Prof. Dr. Kai von Luck

Abgegeben am 15. Februar 2007

Inhaltsverzeichnis

1 Einleitung

Rechner, die in den letzten Jahrzehnten in unserer Welt immer sichtbarer wurden, werden heutzutage immer unsichtbarer und dennoch immer allgegenwärtiger. Ein einziger Arbeitsplatz besteht heute aus mehreren Rechnern und die einzelnen Arbeitsplätze sind miteinander verbunden. Es wird über sie kommuniziert, Daten ausgetauscht und parallel an den selben Projekten gearbeitet. Eine extreme Form des gleichzeitigen Arbeitens mit verschiedenen Rechnern ist ein Collaborative Workspace. In Collaborative Workspaces, in denen sich alle beteiligten Personen in einem Raum zur gleichen Zeit aufhalten, kommunizieren Menschen mit Hilfe immer „unsichtbarer“ werdender Technologie. So wird die Idee von Terry Winograd und Flores [Winograd], dass Menschen nicht durch Computer ersetzt werden, sondern Menschen und Menschlichkeit unterstützt wird, zunehmend zur Realität.

Nutzen wir die heutige Standardsoftware im Collaborative Workspace, müssen wir ständig Daten kopieren und konvertieren oder müssen sogar auf einige Geräte verzichten. Ein fließendes Arbeiten ist hier bei einem Wechsel von einem zum anderen Rechner nicht möglich. Zwei in Kapitel 2 genannte Ansätze, Roomware, ein Framework und iROS, eine Betriebsystemerweiterung zur Entwicklung von Anwendungen in Collaborative Workspaces, versuchen eine Grundlage zu schaffen, um diesem Problem zu begegnen.

Abbildung 1: Die zweite Generation von Roomware [chi2002]

In dieser Arbeit werden in Kapitel 2 verschiedene Ansätze, in Bezug auf Topologie in Collaborative Workspaces, betrachtet. Einhergehend zeigen sich Problematiken und Anreize zur Entwicklung neuer Strategien. In Kapitel 3 werden zwei Techniken und deren neue Einbindung in ein Konzept für Collaborative Workspaces vorgestellt. Kapitel 4 geht konkret auf das Vorhaben in der Masterarbeit und deren Risiken ein. Zum Schluss werde ich einen möglichen Blick und Ausblick auf ein erfolgreiches Projekt beleuchten.

2 Collaborative Workspace Topologie

Ein Collaborative Workspace (CW) ist ein zeitlich und räumlich gebundener elektronischer Arbeitsplatz. Auch die Bezeichnungen synchron und lokal sind dafür gebräuchlich. [vgl. Nastansky]

Wir können zwischen einem statischen und einem dynamischen CW unterscheiden. In einem statischen CW sind alle beteiligten elektronischen Bestandteile vorher festgelegt. Alle Bestandteile sind fest integriert und können vorher klar dimensioniert werden. Das bedeutet, dass jedes Einzelteil des CWs genau angepasst werden kann. Einen dynamischen CW zu integrieren, ist wesentlich komplexer, da hier Benutzer mit ihrem persönlichen und heterogenen Equipment integrierbar sein müssen.

2.1 Verschiedene Ansätze zur Verteilung

Die hier vorgestellten verschiedenen Ansätze werden nur kurz umrissen und auch nicht vollständig klassifiziert. iROS z.B. ist technisch eher als Untermenge von Roomware zu sehen, womit Roomware einen größeren Handlungsspielraum hat. Allerdings ist der Schwerpunkt der Beispielszenarien, in denen es zur Anwendung kommt, eben in die hier gezeigten Klassifikationen einzuordnen, womit die Entwickler eine Tendenz in Form und Konzept vorlegen. Es werden also nicht die eventuell möglichen Techniken betrachtet, sondern die realitätsbestimmenden dargestellten Überlegungen und Szenarien. Es werden auch nur drei Beispiele gezeigt, da andere vorhandene Konzepte komplett oder partiell unter die hier gezeigten subsumiert werden können

2.1.1 Spezielle Anwendung in fester Umgebung

Roomware von der IPSI[1] Gruppe der Fraunhofer Gesellschaft aus Darmstadt beinhaltet eine feste Reihe von Ein- und Ausgabegeräten, die sich alle im selben Raum befinden, alle miteinander vernetzt sind und auf den gleichen Datenbestand zugreifen. Das Framework Beach [Tandler], ist für die Displaysteuerung sowie für die Objektdarstellung und deren jeweiliger Transformation zuständig. Für die synchrone Nutzung der Datenobjekte ist das Framework COAST [Schümmer] zuständig. Die Gruppe der Ein- und Ausgabegeräte besteht aus dem ViewPort, dem CommChair, dem ConnectTable, der DynaWall und dem InteracTable. Die definierte Hardware und COAST sowie BEACH bieten Entwicklern eine

[1] IPSI http://www.ipsi.fraunhofer.de/ambiente/projekte/projekte/roomware.html (Zugriffsdatum: 15.2.2007)

fest kalkulierbare Basis. Zur Demonstration wurde die Beispielanwendung MagNet mit den Erweiterungen Palmbeach und BeachMap entwickelt.

Damit stellt Roomware eine Grundlage zur Entwicklung von parallel nutzbaren verteilten Anwendungen dar. Anwendungen, die speziell für ein festes Repertoire von Hardware unter zu Hilfenahme von COAST und BEACH entwickelt werden, können hier zur Anwendung kommen. Um die Topologie im Raum zu ändern, wurden feste Schnittstellen definiert. Beispielsweise verbinden sich zwei ConnectTable automatisch, wenn man sie aneinander stellt. Dadurch bilden sie miteinander eine gemeinsame Arbeitsfläche.

2.1.2 Flexible Anwendungen in fester Umgebung

Xinerama ist zum Ansteuern einer Powerwall erdacht worden. Es verteilt X-Window-Events und X-Atome auf die verschiedenen X-Server, welche die einzelnen Ein- und Ausgabe Geräte des vorstrukturieren Environments verteilt. Es ist basiert auf dem X-Window-System von Linux und ist damit eine Erweiterung des Window-Systems unter Linux. Da die Verteilung auf dem X-Windows-System basiert und nur eine feste Topologie möglich ist, ist diese Technologie zwar sehr interessant, aber nicht portierbar oder direkt nutzbar.

2.1.3 Flexible Umgebung mit fester Anwendung

iRoom von der Forschungsgruppe HCI der Stanford University besteht ebenfalls aus einer Reihe von Ein- und Ausgabegeräten, die sich alle im selben Raum befinden, alle miteinander vernetzt sind und auf den gleichen Datenbestand zugreifen. Das eigene Betriebssystem iROS unterstützt das gemeinsame Benutzen von Displays, orts- und typenunabhängige Datenspeicherung und die Koordination von Anwendungen. Die Koordination zwischen den Anwendungen wird durch einen so genannten EventHeap realisiert. Die Datenspeicherung ist durch einen DataHeap implementiert, der durch ein Transformationsframework Typenunabhängigkeit realisiert und die Daten durch Indizes auf die typenunabhängigen Metadaten zugreifbar macht. Die Ein- und Ausgabegeräte des iRoom bestehen aus einem hochauflösenden Display, drei Smartboards und einem tableTop mit Touchfunktion. Weitere mobile Geräte wie Notebooks werden über ein Wlan eingebunden.

Als Beispielanwendung wurde ein Präsentationsprogramm entwickelt, welches die Darstellung von Objekten auf mehreren Displays koordiniert. Mit dem eigenständigen Programm PointRight können virtuelle Bildschirmtopologien festgelegt werden, damit der Redirect der Eingabeinstrumente beim Wandeln über die Bildschirme möglichst realitätsnah und damit intuitiv verstehbar ist.

2.2 Bewertung der Ansätze

IROS und Roomware legen Wert darauf, die gängigen drei Betriebsysteme Linux, MS-Windows und MacOS zu unterstützen. Dies kann allerdings hauptsächlich damit begründet werden, mögliche Entwicklungen ins eigene Projekt mit einzubeziehen und keine potenziellen User oder Entwickler auszuschließen. Der Mehraufwand kann nicht damit begründet werden, dass die vorhandene Software auf den verschiedenen Betriebsystemen wiederverwendet werden kann, da in beiden Systemen die vorhandenen Programme an iROS oder Roomware angepasst werden müssten, welches nicht so einfach möglich ist. Wenn sich z.B. iROS als Standard durchsetzen sollte, könnten alle Softwarehersteller ihre Software an iROS anpasse.

Beide Ansätze integrieren und propagieren ein globales Datenmodell. Benutzer sollen nicht mehr Daten in verteilten Systemen hin und her kopieren und sich Gedanken um ein Versionsmanagement machen müssen. Dies ist sinnvoll, wenn es weniger Verantwortungsbereiche geben soll. Bei einer sozial interaktiven Verteilung von Verantwortungsbereichen kann es auch genau diese Verantwortung sein, in einer Struktur die gemeinsames Arbeiten unterstützt, die dies übernimmt und den jeweiligen Personen einen eigenen Handlungsspielraum gibt.

iROS beinhaltet mit PointRight den einzigen expliziten Ansatz, in dem sich das Gesamtsystem, in diesem Fall das Eingabemedium, an eine veränderte Topologie dynamisch anpassen kann. Dieses und mehr zeigt auf, mit welchem konzeptuellen Denken an die Problematik Collaborative Workspace herangegangen wird. Dies soll nicht heißen, dass diese Ansätze falsch sind, ganz im Gegenteil. Das ist wahrscheinlich ein solider strukturierter Weg.

Gehen wir etwas spielerischer an die Problematik heran: Nehmen wir Abstand von der datenbasierten Gruppensicht und nähern uns der interaktiven Sicht des Individuums an, mit dem Vertrauen, dass sich die Territorien eines Individuums, die sich z.B. beim gemeinsamen Arbeiten an einem Tisch einstellen, auch hier ergeben. Dann brauchen wir eine verteilte Arbeitsfläche, die aber, um dem interaktiven intuitiven Arbeiten gerecht zu werden, wie eine einzige betrachtet werden kann. Dies ist mit beiden Systemen möglich, mit dem Hindernis, dass bei jedem Wechsel des Anzeigemediums eine Transformation der Anwendung stattfinden muss. Jede Software müsste dann auch genau diesen Arbeitstatus transformierbar machen. Also muss das laufende Programm als Objekt in dem jeweiligen Status serialisierbar sein.

Ähnlich einer Beispielanwendung von iRoom, bei der beim Übertritt der Anwendung von einem Anwendungsraum (Bildschirm) in den nächsten der Programmkontext als Datei übertragen wird und im neuen Kontext neu gestartet wird, wird zur Zeit an der HAW-Hamburg von Michael Gottwald eine Anwendung entwickelt. Diese ermöglicht es, im Präsentationskontext eine Datei auf dem lokalen Rechner mit einem „Griff“ auf dem Präsentationsbildschirm zu präsentieren. Gegebenenfalls kann diese auch „ferngesteuert“

verändert werden. Der Ansatz, die Anwendungspräsentation mit dem dazugehörigen Rechner-Display Paar zu verbinden, ist im Bereich Collaborative Workspace weit verbreitet.

3 Transformation vorhandener Technologie in den CW-Kontext

In diesem Kapitel werde ich zwei ineinandergreifende Ansätze vorstellen, mit denen ein flüssiges intuitives gemeinsames Arbeiten in einem Collaborative Workspace möglich gemacht werden soll. Mit dem zweiten Teil „Remote-Windows“ stelle ich eine Möglichkeit vor, einfach von der heutigen Arbeitsumgebung in die des Collaborative Workspace migrieren zu können. Mit dem ersten Teil, dem „verteilten 3D-Desktop“, stelle ich eine neue Variante zur Verteilung vor, die ein Arbeiten und Entwickeln ohne Barrieren in einem Collaborative Workspace mit dynamischer Topologie unterstützt.

3.1 Desktopverteilung mit Szenengraphen

Bei der verteilten Anzeige eines größeren Systems graphischer Objekte bietet sich eine Umsetzung mit Szenengraphen[2] an. Hier werden nur die sichtbaren Teile gerendert und in der Struktur des Graphen sind implizit die Abhängigkeiten der zu verteilenden veränderten Graphenteilstücke effektiv integriert.

Jedes Window, der Desktophintergrund und die Mauszeiger werden zu 3D-Objekten und werden dann wie auf dem ursprünglichen Desktop übereinandergelagert. Die Oberfläche des jeweiligen 3D-Objekts wird mit dem Bildinhalt des dazugehörigen Window-Objekts gefüllt. Alle Veränderungen des 3D-Objekts müssen auf das jeweilige Window-Ponton gemapt werden, so dass die Manipulation der höheren Ebene auf die darunter liegende Window-Ebene durchgereicht wird. Dieser Vorgang mit Vor- und Nachteilen wird explizit in Kapitel 3.2 besprochen.

Da der Szenengraph leicht über ein Netzwerk verteilt werden kann und die Projektionspositionen veränderbar sind, können nun die jeweiligen Objekte, die auf die verschiedenen Window-Objekte referenzieren, in einem neuen 3-dimensionalen Raum miteinander verbunden und neu arrangiert werden. Das heißt, dass alle Objekte des Gesamtsystems, also der verschiedenen Rechner innerhalb eines Collaborative Workspace,

[2] Szenengraphen werden hier nicht explizit erklärt und können unter www.OpenSG.org nachgelesen werden.

in diesem Szenengraphen integriert sind. Dieses Arrangement kann z.B. dem realen physikalischem Arrangement nachgeahmt werden oder einer bestimmten funktionalen Denkweise folgen.

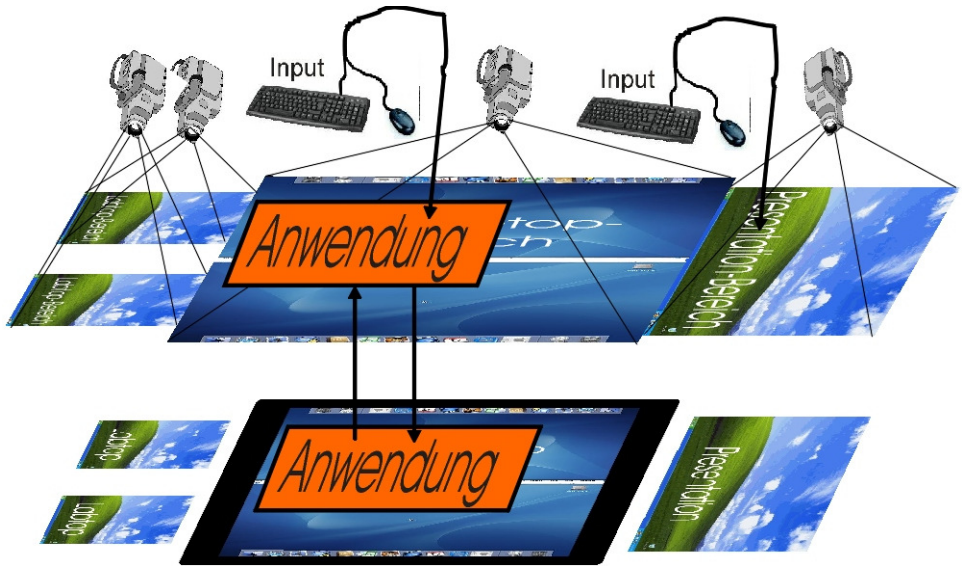

Abbildung 2: Reale und virtuelle Topologie mit Viewports

Ebenfalls können sich Bereiche überlappen, um z.B. ein Laptop-Kontext bei Anwesenheit auf einer Tabletop-Oberfläche zu integrieren, während der Laptop nicht direkt bedient wird. Außerdem ist ein dynamisches Neu-Arrangement der Topologie jederzeit möglich. Zwei Systeme, bei denen die Darstellung auf Szenengraphen basieren sind Croquet[3] und OpenSG[4]. Da es in der Projektgruppe schon Erfahrungshintergrund mit OpenSG gibt und ebenfalls eine Physikengine für OpenSG existiert, ist geplant, die ersten Prototypen mit OpenSG zu realisieren. Croquet sollte parallel evaluiert werden und gegebenenfalls ein Systemwechsel vollzogen werden.

[3] Croquet: http://www.opencroquet.org (gesichtet am 23.11.06)

[4] OpenSG: http://opensg.vrsource.org/trac (gesichtet am 23.11.06)

3.2 Remote-Windows

Remote-Windows können als eine Variante von Remote-Desktop wie zum Beispiel VNC[5] gesehen werden. Der Unterschied besteht darin, dass Remote-Desktop den ganzen Bildschirm in Form des Desktop ausfüllt und damit unflexibel in der Orientierung ist. Es werden außerdem nicht relevante Informationen, wie zum Beispiel nicht benötigte Anwendungen, Icons oder auch das Hintergrundbild, dargestellt. Remote-Windows zeigen nur den Kontext der zu transferierenden Anwendung und machen diese nicht mehr desktopgebunden. Mit einem Remote-Window können auch einfach Betriebssystembeschränkungen übergangen werden, da mit ihnen z.B. unter Linux ohne Emulator die Anwendung eines anderen Windowsrechners angezeigt und genutzt werden kann. Hierbei ist allerdings zu beachten, dass der Programmkontext der des eigentlichen Hosts bleibt. Nur die Anzeige und Eingabe des Benutzers wird transferiert. Bei großen Windows steigt damit der Aufwand des Transfers des Abbildes über das Netzwerk. Ein Vorteil dabei ist, dass der darstellende Rechner keine besondere Rechenleistung bei rechenintensiven Anwendungen zur Verfügung stellen muss. Auch die Problematik des Serialisierens von laufenden Anwendungen tritt hier erst einmal nicht in Erscheinung. Das Programm läuft einfach im eigenen Kontext weiter, nur die Erscheinung muss serialisiert werden.

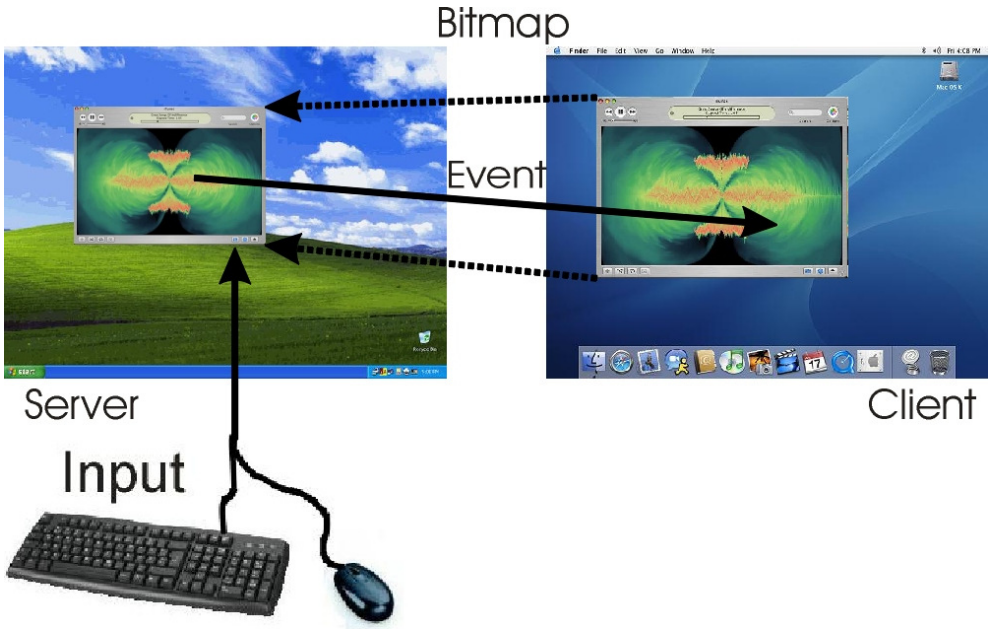

Abbildung 3: Remote-Windows zum „befreien“ von Anwendungen

[5] VNC (Virtual Network Computing): http://ultravnc.sourceforge.net (gesichtet am 23.11.06)

Da jedes der drei großen Betriebsysteme, Linux, Windows und MacOs das eigene Window-System mit dem Konzept des Window-Managers und Event-Manager integriert haben, können hier die jeweiligen Events einfach transformiert und an das zuständige System weitergeleitet werden. Der Window-Manager liefert die Bilddaten, welche zum Remote-System weitergeleitet werden. Das Vergrößern bzw. Verkleinern wird auf dem Host umgesetzt. Um die Ausrichtung und die relative Größe auf dem Remote-System verändern zu können, sollte im Remote-Status des Window ein Extrarahmen zur Manipulation dieser Parameter implementiert werden. Wenn das Window den eigentlichen Host verlässt, verbleibt zwingend das eigentliche Window auch auf dem Host. Dies würde dazu führen, dass zwei Windows existieren, was aber nicht gewollt ist. Hier muss das remotete Window auf dem Host überdeckt werden oder in einem nicht sichtbaren Bereich geparkt werden. Bei einer Überdeckung des Window müssen die jeweiligen Events abgefangen und umgeleitet werden. Dies kann zu komplexen Zuordnungen der Events führen, wenn sich remotete und nicht remotete Windows auf dem Host überdecken. Deshalb sollte hier die Wahl auf das Parken im „Nirwana“, falls möglich, bevorzugt implementiert werden.

Verschiedene Problematiken stellen sich bei Remote-Windows. Alle Datentransfers zwischen System und Anwendung, die sonst durch das jeweilige Betriebsystem umgesetzt werden, können nun nicht mehr so einfach angewendet werden, da die remoteten Anwendungen weiterhin im eigenen Betriebsystemkontext hängen. Hier müssen dann nicht, wie in den herkömmlichen Anwendungen, der Zustand der Anwendung serialisierbar gemacht werden, sondern der Datentransfer zwischen System und Anwendung. Betriebssystemeigene Implementierungen wie Drag&Drop und ein Clipbord, die systemübergreifend aber personengebunden arbeiten sollen, müssen für ein freies Funktionieren der Anwendungen integriert werden.

4 Aufbau und Risiken

4.1 Vorgehen und Aufbau der praktischen Arbeit

Geplant ist, möglichst schnell kleine Prototypen der Betriebsystemerweiterung zu implementieren. Aufbauend auf die vorangegangenen Prototypen werden weitere entwickelt und nützliche Teile wieder verwertet. Da auf existierende Betriebs- und Windows-Systeme aufgebaut wird, und damit nicht garantiert werden kann, dass alles so funktioniert, wie es in der Theorie entworfen wurde, sollen möglichst früh Irrläufer erkannt und nach neuen Möglichkeiten gesucht werden.

Ein weiteres Kriterium ist, dass andere Studenten schon im Entwicklungsstadium den verteilten 3D-Desktop zur Entwicklung eigener Software nutzen können, um diesen zu erweitern oder um erste Tests und Thesen mit den eigenen Versuchen zu verifizieren.

Folgende Reihenfolge von Schwerpunkten, die sich während der Abarbeitung ändern kann, sollen als Prototypen implementiert werden.

Implementierung von:

1. Einzelrechner: Oberfläche mit festem Viewport, beweglichem Dummywindow und Mausinteraktion.
2. Multirechner: Verteilte Oberfläche mit fester Anzahl von Rechnern mit festem Viewport, beweglichen Dummywindows und einfacher Mausinteraktion.
3. Multirechner: Verteilte Oberfläche mit variabler Anzahl von Rechnern mit festem Viewport, beweglichem Dummywindow und Multimausinteraktion.
4. Integration von dynamischer Topologie mit variablen Viewports.
5. Entwicklung der Remote-Window Technologie.(inklusive clipboard,drag&drop)
6. Integration der Remote-Window Technologie.
7. Usability-Test des derzeitigen Standes.
8. Gegebenenfalls Portierung auf andere Betriebssyteme.

Als Zwischenschritt zwischen 1. und 2. könnte ein verteiltes Pong zur Präsentation und Aufheiterung implementiert werden. Nach dem 3. Schritt können andere Studenten diese Software nutzen, um eigene Tests mit ihren Entwicklungen durchzuführen. Der 4. Schritt kann als Versuchsphase eingestuft werden. Hier ist noch nicht klar, welchen Sinn eine dynamische Topologie haben kann. Ein Beispiel wäre ein beweglicher großer Screen, über den mithilfe eines Lokalisationssystems Ausrichtung und Position bekannt sind. Dieser würde sich dann dynamisch in die Topologie einpassen, so dass sich die 3-dimensionale Anordnung im System verändert und damit auch die Nutzung des Raumes.

Die Integration der Remote-Windows macht das Projekt zu einem Projekt mit realem Gegenwert. Mit den Remote-Windows ist das System real nutzbar. Zur Bewertung des Erreichten ist Punkt 7 zuständig. Bei positivem Ergebnis sollte der verteilte 3D-Desktop auch auf anderen Betriebsystemen lauffähig sein, um es variabel nutzbar zu machen.

4.2 Risiken

Das größte Risiko bei diesem Projekt ist die Komplexität. Es gibt viele Hürden und Ebenen, die überwunden werden müssen. Deshalb kann der Zeitaufwand stark unterschätzt werden. Folglich ist es sinnvoll, ein Minimalziel zu definieren, dass der oben genannten Liste ohne die Punkte 4, 5, 6 und 8 entspricht. Mit diesem Ergebnis wäre ein Stand erreicht, an dem sich einschätzen lässt, welchen Wert dieses Projekt und damit der Ansatz hat. Weitere Arbeiten können dann darauf aufbauend das Projekt weiterbringen.

Zwei große Risiken liegen in den verwendeten Systemen OpenSG und Windows-XP. An diesem Punkt kann noch nicht eingeschätzt werden, in wieweit das Windows-System die

Daten so zur Verfügung stellt, dass sie gut nutzbar sind. Ein Beispiel: Der Window-Manager von Microsoft verändert das Aussehen des Windows während das Remote-Window-System auf dieses zugreift. In der SDK sind zwar Funktionen implementiert, die zurückliefern, ob ein Window fertig bearbeitet ist. Dies ist aus eigener Erfahrung leider nicht zuverlässig. Dieses Risiko würde nur zur Einschränkung des Projektes führen.

Es kann sein, dass die implementierten Funktionen von OpenSG nicht ausreichen, um das Ziel zu erreichen. Aufgrund dessen müsste OpenSG von Grund auf erweitert werden, was wiederum einen sehr großen Zeitaufwand darstellen würde. Dadurch würde wahrscheinlich die Projektzeit überschritten werden. In beiden Fällen müssten Abstriche gemacht werden, die zum Scheitern des Projektes führen könnten. Hier ist der Übergang zwischen Schritt 2 und 3 entscheidend. Ein Scheitern an diesem Punkt würde ein Umsteigen auf das Croquet-System bedeuten, in dem diese Funktionen schon getestet wurden. Das hat zur Folge, dass parallel laufende Projekte auch umsteigen müssten. Deshalb sollte dieser Schritt möglichst im ersten Drittel der Projektphase erreicht werden. Da ein Umsteigen auf Croquet möglich ist, führt dieses Risiko zwar zu einem enormen Zeitverlust aber nicht zum Scheitern.

5 Fazit und Ausblick

Wird das Projekt erfolgreich beendet, so haben wir eine einfach zu implementierende Betriebssystemerweiterung, die das alte Betriebsystem überdeckt und sich mit anderen so verbindet, als würde es eins sein. Das einfache Installieren und Starten des Programms würde hierfür reichen. Es muss keine Standardsoftware angepasst und keine neu implementiert werden. Ebenfalls können Entwickler schnell ein Gefühl für die Nutzbarkeit eines Collaborative Workspace bekommen und dieses in Neuentwicklungen mit einfließen lassen. Durch die einfache Veränderung der Topologie können dann auch sehr einfach verschiedene Anordnungen innerhalb eines Collaborative Workspace ausprobiert werden.

Da das System der Realität angenähert wurde, um fließendes intuitives Arbeiten zu ermöglichen, liegt es nahe, den verteilten 3D-Desktop mit physikalischen Eigenschaften auszustatten. Deshalb ist während oder am Ende des Projekts eine Migration der Ergebnisse eines parallel laufenden Projektes mit dem Titel „Physikbasierte Interaktion im Collaborative Workspace“ von Philipp Rossberger [Rossberger] sinnvoll.

Eine Erweiterung des Systems um persistente Datenverwaltung und ein Wechseln der Remote-Anwendungen auf einen anderen Host mittels Serialisierbarkeit der Anwendung würde das System gut erweitern. Um ein unbeabsichtigtes Verschwinden von Anwendungen zu verhindern, wäre eine Erweiterung, die ich als „Stickable-Window“ bezeichne, ebenfalls sinnvoll. Letztere beiden Erweiterungen basieren darauf, dass es möglich ist, eine Anwendung wie bei iROS von einem Rechner auf einen anderen während des laufenden Betriebs zu transferieren.

6 Literaturverzeichnis

6.1 Veröffentlichungen

[chi2002] Norbert Streitz, Thorsten Prante, Christian Müller-Tomfelde, Peter Tandler, Carsten Magerkurth; Roomware – The Second Generation, CHI 2002, April 20-25, Minneapolis, Minnesota, USA, 2002

[Nastansky] L. Nastansky, T. Bruse, P. Haberstock, C. Huth, S. Smolnik; Büro-informations- und Kommunikationsystem:Groupware, Workflow Management, Organisationsmodellierung und Messaging-Systeme, Erich Schmidt Verlag, Berlin 2002

[Rossberger] Phillip Rossberger, Physikbasierte Interaktion im Collaborative Workspace, Seminararbeit an der HAW-Hamburg Februar 2007, noch nicht erschienen.

[Schümmer] Jan Schümmer, Till Schümmer, Christian Schuckmann, COAST – Ein Anwendungsframework für synchrone Groupware, Conference Proceedings for the „Net.ObjectDays 2000", http://www.opencoast.org (zugriff 15.7.2006)

[Tandler] Peter Tandler, The BEACH Application Model and Software Framework for Synchronous Collaboration in Ubiquitous Computing Environments, Submitted to JSS, special issue on UbiTools v4.2, 26.02.2003

[Winograd] Terry Winograd, Fernando Flores, Erkenntnis Maschinen Verstehen, Rotbuch Verlag, Berlin 1989