Ludwig-Maximilian-Universität München
Sommersemester 2004
Hauptseminar „Augmented and Virtual Reality“
Betreuer: Dipl.-Inf. Arnd Vitzhum

Architektur und Entwicklung von Augmented Reality Anwendungen; AR Frameworks

-Eine Hausarbiet von Branimir Mirtchev-

Inhaltsangabe

Kurzzusammenfassung

Augmented Reality ist ein neues Forschungsgebiet und befindet sich noch in der Entwicklungsphase. Deshalb ist die Entwicklung von Augmented Reality Anwendungen eine komplexe und schwierige Aufgabe. Man sollte auf vorhandenen Erfahrungen aufbauen. Bis jetzt konnten sich keine Standards etablieren, die eine allgemeine Vorgehensweise beim Entwurf von AR Software definieren. Diese Seminararbeit gibt einen Überblick über das Thema Architektur und Entwicklung von Augmented Reality Anwendungen und insbesondere AR Frameworks. Um dem Leser die Grundlagen der Architektur von AR Anwendungen näher zu bringen, werden zwei Notationen zur Beschreibung von AR Systemen vorgestellt. Nach einer kurzen Definition von Frameworks werden dann 5 wichtige Projekte aus dem Themenbereich genauer erläutert.

1. Einleitung

Lange Zeit war die Entwicklung und Nutzung von Augmented Reality eine kostspielige Angelegenheit. Die benötigte Hardware wird aber immer billiger und mittlerweile stehen einige grundlegende Entwicklungswerkzeuge als Software zur Verfügung.
Die Projekte, die hier vorgestellt werden, sind experimentell und unvollständig. Die Dokumentation, die angeboten wird, ist noch lückenhaft. Es gibt immer noch viele Entwickler von AR Anwendungen, die ihre eigenen Algorithmen und Techniken bauen. Die Wiederverwendung ist nicht das Hauptziel.
Ohne die Wiederverwendung von Code und Algorithmen ist die Entwicklung einer AR Anwendung viel zu aufwendig. Die meisten Projekte basieren auf zwei grundlegende Bibliotheken: OpenGL und ARToolkit. OpenGL ist weitverbreitet und wird für die 3-D-Darstellung von Objekten benutzt. ARToolkit ist eine Open-Source-Softwarebibliothek, die Funktionen für Tracking zur Verfügung stellt und genügend schnell ist, um in Echtzeitanwendungen benutzt zu werden. Immer mehr Entwürfe benutzen und erweitern diese und ähnliche Software, um letztendlich Gesamtlösungen für spezifische oder allgemeine Probleme beim Entwurf und der Entwicklung von AR anzubieten. Es werden nicht nur Bibliotheken entwickelt, sondern auch Tools, Frameworks und vollständige Entwicklungsumgebungen, die dann von AR Experten für die schnelle Prototypenbildung eingesetzt werden können.

2. Beschreibungssprachen

Bis jetzt gibt es keine bekannten Designtools oder Methoden der Systementwicklung, die die typischen Eigenschaften von Augmented Reality in Betracht ziehen. Die traditionellen Methoden sind nicht dazu geeignet, physische Objekte einzubeziehen und zu charakterisieren. Die bewährten Modellierungssprachen, wie UML, modellieren die Funktionalität von Anwendungen sehr gut, aber die einzigen physischen Objekte, die charakterisiert werden können, sind die Benutzer des Systems.
ASUR und UMLi sind zwei Notationen, die benutzt werden können, um designspezifische Eigenschaften von Augemented Reality Anwendungen zu erfassen.

2.1 ASUR und ASUR++

Die ASUR Notation wurde ursprünglich an der Universität Glasgow entwickelt. ASUR hilft beim Kombinieren der physischen und digitalen Welt, indem es die physischen und digitalen Objekte des Systems identifiziert. ASUR++ ist eine Weiterentwicklung dieser Notation. Die Mobilität des Benutzers in einem AR System erfordert die Beachtung der räumlichen Beziehungen zwischen den Usern und den Objekten des Systems. Die neuen Features von ASUR++ ermöglichen es dem Designer, solche Beziehungen auszudrücken. Man kann z.B. mit Hilfe von ASUR++ die Bedingung ausdrücken, dass der Abstand zwischen zwei Objekten kleiner als 2 Meter sein muss.

2.1.1 Die ASUR++ Notation

Die ASUR++ Notation unterstützt die Beschreibung der physischen und digitalen Entitäten, die ein AR System ausmachen. Diese umfassen die Benutzer, andere Artefakte und ihre informationsspezifischen Beziehungen. Die Abhängigkeiten zwischen den Komponenten stellen gleichzeitig die Eigenschaften der statischen Struktur und des dynamischen Verhaltens des Systems und der damit verbundenen realen Objekten dar.
ASUR beschreibt ein interaktives System als eine Menge von vier Typen von Entitäten, so genannte „components“:

- Komponente S: das Computer System
- Komponente U: User des Systems
- Komponente R: reales Objekt, das an der Anwendung als Tool (R_{tool}) oder als das Objekt der Anwendung (R_{object}) beteiligt ist
- Komponente A: Outputadapter(A_{out}) und Inputadapter (A_{in}) überbrücken die Lücke zwischen den Komponenten. A_{in} repräsentiert alle möglichen Sensoren, wie zum Beispiel optische, haptische, Schallsensoren oder Drucksensoren. Outputadapter können alle Geräte sein, durch die der User Informationen wahrnehmen kann (z.B. Bildschirm, Lautsprecher, HMDs usw.)

Es existieren 3 Typen von Beziehungen zwischen ASUR Komponenten:

- Datenaustausch: wird durch ein Pfeil repräsentiert (A->B). Symbolisiert den Transfer von Informationen zwischen zwei ASUR Komponenten

- Physische Aktivität, die eine Aktion auslöst: ein Doppelpfeil (A=>B) zeigt, dass wenn die Komponente A eine räumliche Bedingung bezogen auf eine Komponente B erfüllt, Daten über eine andere spezifische Beziehung (C->D) ausgetauscht werden.
- Dauerhafte physische Verbindung: dargestellt durch doppelte Linie (A=B). In ASUR++ Diagramme wird diese Beziehung mit einer Kontur um die Objekte zusätzlich verstärkt.

Die Interaktion mit dem System wird folglich durch eine Menge von Relationen dargestellt, die mit der Komponente U verbunden sind. [1]

2.1.2 Veranschaulichungsbeispiel (Augmented Museum Scenario)

Die Aufgabe ist, eine AR Anwendung für ein Museum zu modellieren. Das System muss den Besucher des Museums mit digitalen Informationen versorgen, die an seine aktuelle Position angepasst sind. Es wird davon ausgegangen, dass die Bestimmung der Lage des Benutzers über ein auf Radiofrequenzen basierendes Lokalisierungssystem ermöglicht wird. Die Anwendung kennt, wo sich der User in Bezug auf die Exponate befindet. In diesem Scenario wird der User mit Hilfe von AR auf einem vordefinierten Pfad, bestehend aus einer geordneten Menge von Exponaten, geführt. Dabei bekommt er eine Textbeschreibung des zu folgenden Pfades und der Exponate, die er gerade betrachtet. Die Aufgabe des Computersystems ist es, den Besucher die richtige Information in Abhängigkeit von seinem Standort und die Ausstellungsstücke zu geben.
Eine ganz einfache Modellierung ist durch Abbildung Nr. 1 veranschaulicht. Der Besucher ist die Komponente U, das Exponat ist die Komponente R_{object}, das von dem Benutzer betrachtet wird (R_{object}->U). Die Komponente S ist unter anderem eine Datenbank, die Besucherpfade und Daten über die Museumsstücke enthält.

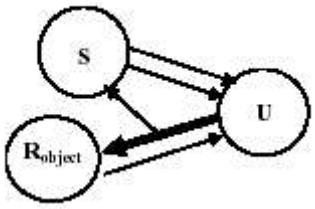


Abbildung 1: Abstrakte Beschreibung des Scenarios in ASUR++ **[1]**

Das System versorgt den Besucher mit Informationen, verbunden mit dem Pfad und dem Exponat: $S_{(path)}$ ->U, $S_{(exibit)}$->U. Diese Informationen werden angezeigt, wenn der User in der Nähe des Exponats kommt: (R_{object}=>U) löst aus (R_{object}=>U)->U und (R_{object}->U).
In Abbildung Nr. 2 werden die Beziehungen zwischen den Komponenten verfeinert. Hier wird der Datenfluss vom System zum Benutzer über einen Outputadapter(A_{out}) ermöglicht. A_{out} bildet mit dem User eine physische Einheit (A_{out}=U) und versorgt ihn, wie bereits erwähnt, mit zwei Typen von Informationen (A_{out}(path)->U und A_{out}(exibit). Die dafür benötigten Daten bekommt er vom System S (S->A_{out}).

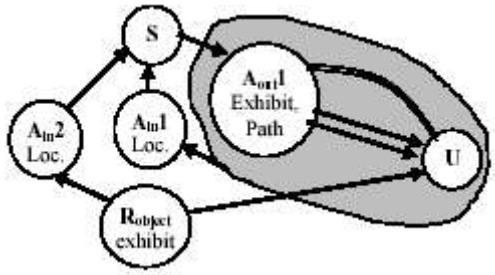
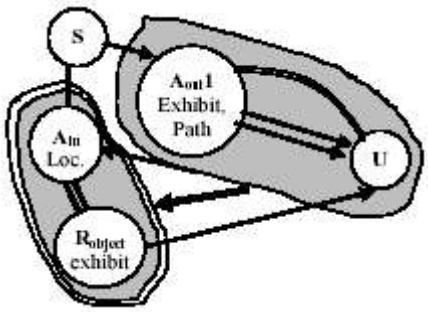


Abbildung 2: ASUR++ Beschreibung des Scenarios mit zwei Inputadapter**[1]**

In Abbildung 2 ist eine Lösung der Aufgabe mit zwei Inputadapter dargestellt: A_{in1} und A_{in2}. Der erste Inputadapter wird benötigt, um die Position des Besuchers zu bestimmen (U->A_{in1}) und diese dann an das Computersystem zu übergeben. Ähnlich bestimmt der zweite Sensor die Lage des Exponats (R_{object}->A_{in2}) und leitet es weiter (A_{in2}->S). Dieser Ansatz ist keine gute Modellierung, denn die Anwendung muss zwei Inputs auswerten.
Eine bessere Lösung ist in Abbildung 3 dargestellt. Hier wird ein Gruppierungsmechanismus implementiert, bei dem der Inputadapter(z.B. Bewegungssensor), der den User lokalisiert, in der Nähe des Ausstellungsstücks installiert ist. Diese Beziehung wird durch R_{object}=A_{in} dargestellt. Auf diese Weise ist die relative Position des Besuchers zum Exponat leicht zu bestimmen. Wenn der Benutzer in der Nähe des Exponats kommt, löst das den Transfer von Daten von dem Besucher zum Inputadapter. Die folgende Beziehung entsteht: U=>A_{in} löst U->A_{in} und logischerweise R_{object}->U.



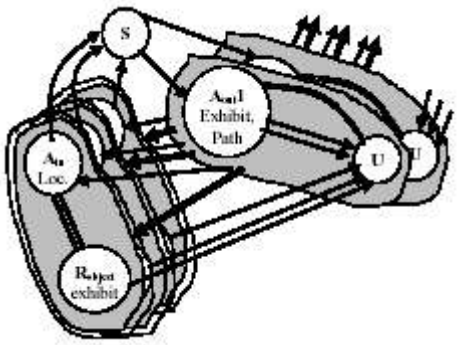


Abbildung 4 [1]

Abbildung 3: Vollständige Beschreibung mit einem Inputadapter **[1]**

In Abbildung 4 ist der Vollständigkeit halber eine Modellierung mit mehreren Besuchern und Exponaten gezeigt. [1]

2.2 Unified Modeling Language for Interactive Applications

UML ist eine Notation für die Modellierung von vollständigen Objektorientierten Softwaresystemen. UML ist für die Entwicklung von Benutzerschnittstellen für AR Anwendungen nicht geeignet, weil interaktive Objekte (Widgets) nicht problemlos modelliert werden können. Model-based user interface development environments (MB_UIDEs) ist eine moderne Herangehensweise zur systematischen Schnittstellenentwicklung. Diese Technologie kann nur für die Konstruktion von UIs benutzt werden und nicht für Anwendungen. UMLi ist ein Forschungsprojekt, entstanden 1998 in der Information Management Group der Universität Manchester. UMLi benutzt eine Kombination von UML und MB-UIDE Techniken. Dadurch wird die nötige Integration zwischen UIs und der Anwendung erreicht und man kann Benutzeroberflächen systematisch nach bekannten Verfahren entwickeln.

UMLi ist eine Erweiterung von UML. Das bedeutet, dass UMLi mehr Konsturktoren als UML hat. Es gibt 6 Konstruktoren für UI Diagramme, die verschiedne Rollen von Interaktionsklassen repräsentieren:

- Freie Kontainer (gestrichelte Würfel) sind „top-level" Interaktionsklassen, die nicht in anderen enthalten sein können
- Kontainer (gestrichelte Zylinder) bieten ein Gruppierungsmechanismus, dass Interaktionsklassen zusammenbringt
- Inputters (abwärts zeigende Dreiecke) empfangen Informationen vom User
- Displayers (aufwärts zeigende Dreiecke) schicken Informationen zu den Usern
- Editoren (nach oben zeigende Rhomben) ermöglichen den Informationsaustausch in zwei Richtungen
- ActionInvokers (nach rechts zeigendes Pfeil) bekommen Instruktionen von den Usern

[3],[4]

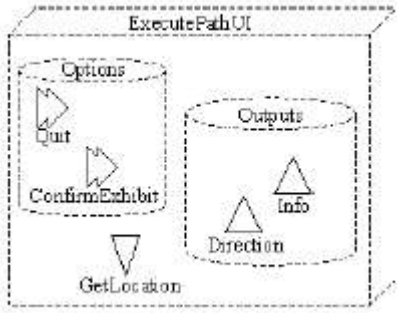


Abbildung 5: Mögliche Modellierung der Benutzerschnittstelle aus Abschnitt 2.1.2. Auf diese wird nicht näher eingegangen.[3]

3. Frameworks

Nach der Definition von Pomberger/Blaschek ist ein „framework" (Rahmenwerk, Anwendungsgerüst) eine Menge von zusammengehörigen Klassen, die einen abstrakten Entwurf für eine Problemfamilie darstellen. Die Vorteile der Benutzung von Frameworks sind die Wiederverwendung von Code, Architektur und Entwurfsprinzipien und auch die Wiederverwendung von Verhaltensschemata einer Gruppe von Klassen. Prof. Hußmann unterteilt in [4] Frameworks in architektur-getriebene und daten-getriebene. Ein architektur-getriebenes Framework kann durch Vererbung und Überdefinieren angepasst werden und ist aus komplexen Klassenhierarchien und Muster aufgebaut. Die Einarbeitung ist aufwendig und es ist viel neuer Code zu schreiben. Datengetriebene Frameworks sind hingegen einfach durch Objektkonfiguration und Parametereinstellung anzupassen, haben aber eine eingeschränkte Flexibilität. Ein Kompromiss ist eine Zwei-Schichten-Architektur, die die Vorteile der beiden Typen kombiniert.

4. AMIRE (Authoring Mixed Reality)

AMIRE gehört zu den größten europäischen Projekten, die sich mit AR beschäftigen. Es wird von dem EU IST Programm gefördert. Das Hauptziel und Idee von AMIRE ist ein

Softwaresystem zu definieren und implementieren, das Experten das leichte Design und Implementierung von Mixed Reality Anwendungen ohne detailliertes Wissen über die Basistechnologien von MR ermöglicht. Es beschreibt ein generisches Framework, das eine einfache Kommunikation zwischen den Objekten (MR Komponenten) ermöglicht, die für die Entwicklung von AMIRE Anwendungen verwendet werden. Das Framework benutzt eine auf Komponenten basierte Technologie und besteht aus eine Menge von Komponenten, die für den Entwurf von Demonstratoren nötig sind, eine Sammlung von wiederverwendbaren GEMs und ein visuelles Tool für die Konstruktion von MR Anwendungen.

4.1 Der AMIRE Produktionsprozess

Der Produktionsprozess umfasst viele Aufgaben: von dem Erkennen des Problems bis zur Endlösung in Form einer AR Anwendung.

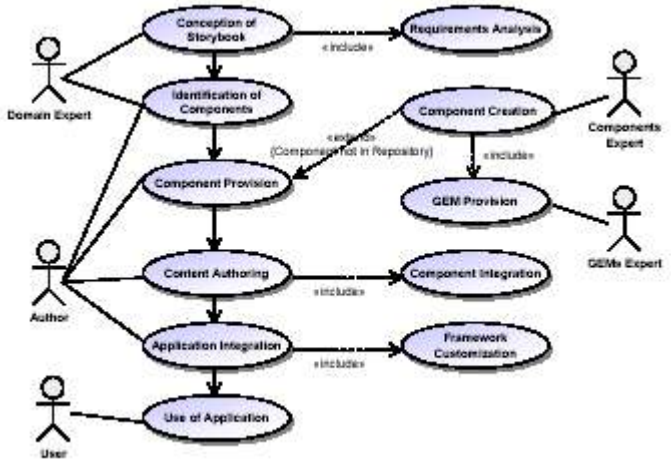


Abbildung 6: Das Produktionsprozess [6]

Im ersten Schritt „Conception of Storybook“ wird eine gründliche Anforderungsanalyse vorwiegend von dem Domain Experten durchgeführt. Im zweiten Schritt „Identification of necessary Components“ werden die Komponenten, die für die Implementierung nötig sind, genau beschrieben. Diese werden bei „Content Provision“ geliefert. Der Autor sucht nach bereits implementierten Komponenten. Diejenigen, die noch nicht existieren, werden von Komponentenexperten entworfen, der eng mit dem GEMS Experten zusammenarbeitet. Wenn alle Komponenten fertig sind, beginnt der Autor mit „Content Authoring“. In diesem Schritt werden die Komponenten angepasst und kombiniert. Bei der „Application Integration“ muss die Anwendung dem Benutzer innerhalb des Laufzeitframeworks verfügbar gemacht werden. Im letzen Schritt wird die Software während der Bedienung gepflegt und verbessert.[5],[6]

4.2 MR GEMs

Viele kleine Tricks, Ideen und Lösungen von Problemen, die immer wieder bei der Entwicklung von MR Anwendungen auftauchen, sind bereits in anderen Projekten implementiert worden. Leider werden solche Lösungen nicht veröffentlicht und es werden keine spezifischen Bibliotheken dafür entwickelt. Deshalb müssen sie mehrmals aufs Neue entdeckt werden. GEMs bieten ein „low-level“ Mechanismus für Wiederverwendung und Einbindung solcher Techniken und Algorithmen in das AMIRE Framework.[5],[6]

4.3 AMIRE Components

Mixed Reality Komponenten in AMIRE sind essentielle Elemente für die Entwicklung der Autorensoftware. Sie repräsentieren Lösungen für spezielle domainspezifische Probleme und kombinieren und erweitern typischerweise GEMs, um eine „high-level“ API zu erhalten. AMIRE Komponenten sollten strukturiert, wieder verwendbar für verschiedene Versionen und Anwendungen, flexibel und erweiterbar sein.[6]

4.4 Das AMIRE Framework

Das Framework ist die Verbindung zwischen Komponenten und GEMs. Komponentenentwickler müssen ein Interface verwenden, das so genannte „generic uniform component interface“, welches Teil des Frameworks ist. Diese Schnittstelle ermöglicht die Konfiguration und Kommunikation der Komponenten. GEMs können auch leicht mit Hilfe von Richtlinien und Schnittstellen in das Framework integriert werden.[6]

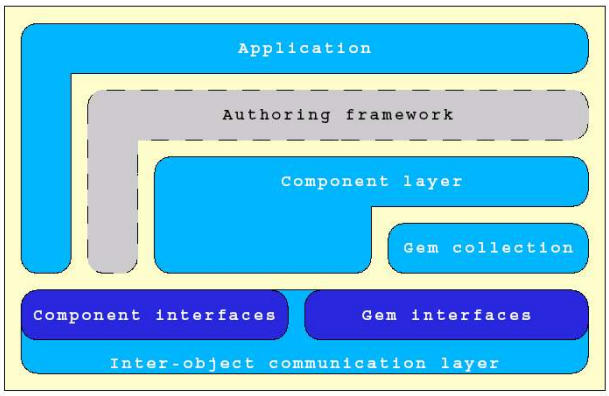


Abbildung 7: Zusammenarbeit aller Teile von AMIRE, um die Anwendung zu entwickeln.[7]

5. DART

Das Designers Augmented Reality Toolkit wurde entworfen, um die schnelle Prototypenbildung von AR Anwendungen zu unterstützen. DART ist als eine Sammlung von Erweiterungen von Macromedia Director aufgebaut, das fast ein Standard für die Entwicklung von Multimediaanwendungen ist. DART ist auch für Benutzer geeignet, die keine Erfahrung mit Augmented Reality haben.

Die Hauptidee von DART ist die Zeit zwischen dem Entstehen der Idee für eine AR Anwendung und deren Verwirklichung zu verkürzen. Das Tool unterstützt die iterative Entwicklung durch Testen in der Konstruktionsphase. Auf lange Sicht ist das Ziel der Autoren des Toolkits, Designern die Möglichkeit zu geben, ihre Anwendungen schnell zu entwickeln und in derselben Umgebung, die für den Einsatz des Endprodukts benutzt wird, zu testen. [8]

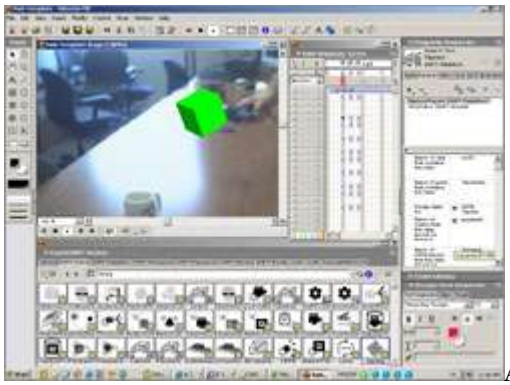


Abbildung 8: Typisches Aussehen des Toolkits[8]

6. ARVIKA

Viele europäische Projekte befassen sich mit der Entwicklung von AR Anwendungen für eine spezielle Domäne. Im Rahmen des Leitprojekts ARVIKA (Augmented Reality für Entwicklung, Produktion und Service) werden Augmented-Reality-Technologien erforscht und realisiert. Das Projekt wird von dem Ministerium für Bildung und Forschung gefördert. Ziel des Projekts ist die benutzerorientierte und anwendungsgetriebene Erforschung und Realisierung von Arbeitsprozessen in Entwicklung, Produktion und Service für komplexe technische Produkte und Anlagen.
Die anwendungsbezogenen Themenschwerpunkte von ARVIKA haben zum Ziel die Erprobung von Augmented Reality in der Entwicklung, in der Produktion/Fertigung und im Service. Die genannten Schwerpunkte sind in eine benutzerdefinierte Systemgestaltung eingebettet, die das Projekt in allen Phasen begleitet. Der Themenschwerpunkt „Basistechnologien“ hat zum Ziel, die notwendigen Augmented Reality-Technologien bzgl. Visualisierung, Tracking, Interaktion, Mobilität etc. zu erforschen und zu entwickeln, so dass eine leistungsfähige und offene Systemplattform zu Verfügung gestellt wird, auf die sich die Anwendungsfelder stützen können.

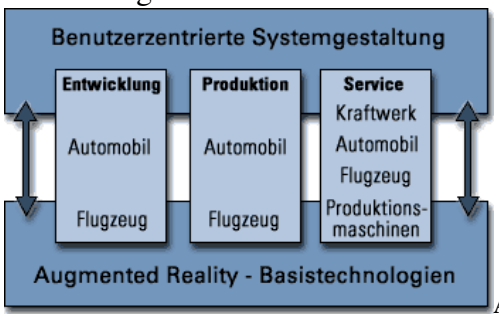


Abbildung 9: Projektstruktur [9]

Konkret erfolgt die Konzentration bei der Entwicklung auf das Automobil, bei der Produktion auf Automobil und Flugzeug und beim Service auf Anlagentechnik, hier Kraftwerke und der für die Produktion/Fertigung notwendigen Werkzeug- und Produktionsmaschinen.
Die Projektideen werden in Anwendungsfeldern wie Automobil- und Flugzeugbau, sowie Maschinen- und Anlagenbau umgesetzt. Damit sind Domänen mit außerordentlicher Relevanz für die deutsche Industrie tangiert.[9]

7. DWARF

Distributed Wearable Augmented Reality Framework (DWARF) wurde am Lehrstuhl für Angewandte Software Entwicklung der Technischen Universität München entwickelt. Das Hauptziel des Projekts ist es, eine Lösung zu finden, die die schnelle Prototypenbildung mit mehreren Entwicklern an verschiedenen Rechnern ermöglicht. Das Ziel ist eine heterogene Infrastruktur aufzubauen, die auf verschiedenen Plattformen laufen kann. Bei DWARF gibt es keine zentrale Komponente, Middleware ist grundsätzlich CORBA.[10]

7.1 Architektur

Das DWARF Framework basiert auf kollaborativen verteilten Services. Diese sind unabhängig und veröffentlichen ihre Anforderungen, so genannte „Needs“ und ihre Angebote („Abilities“) mit Hilfe von Dienstverwalter(„Service Mangers“). In der Architektur gibt es keine zentrale Komponente. An jedem Knoten des Netzwerks befindet sich ein Dienstverwalter. Die „Service Mangers“ bauen die Verbindungen zwischen den Services auf. Jedes „Ability“ hat eine Menge von Attributen, die dessen Qualität beschreiben. Genauso ist es bei den „Needs“. Diese Eigenschaften werden von dem Dienstverwalter benutzt, um Angebote zu finden, die über genügende Qualität verfügen, um eine Anforderung zu erfüllen. Das Bild zeigt, wie Services über ihre Anforderungen und Angebote verknüpft werden.[10]

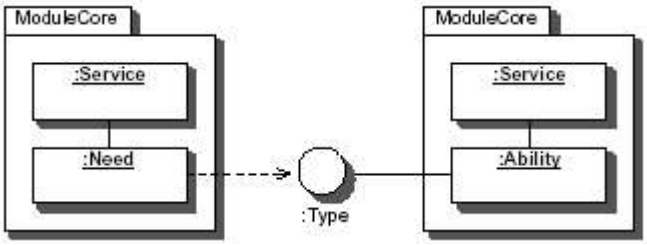


Abbildung 10: Die Services werden mit Needs und Abilities verbunden [11]

7.2 Implementierung

Die stationären Rechner befinden sich in einem 100 megabit LAN, die mobilen Computer sind mit Hilfe von wireless Ethernet verbunden. Die DWARF Services sind als separate Prozesse oder als Threads eines Prozesses realisiert. CORBA und verschiedene Service Manager verbinden die Dienste.
Beim Start wird jedes Service über CORBA bei seinem Servicemanager, der auf ankommende Verbindungen wartet, registriert. Die Dienstverwalter, die auf verschiedene Rechner laufen, kommunizieren miteinander mit Hilfe von SLP(Service Location Protocol) und CORBA und verknüpfen die Dienste. Diese können dann selbst mittels CORBA kommunizieren.[10]

7.3 Tool Support

Als universelles Überwachungs- und Debuggingtool wurde DWARF Interactive Visuallisation Environment implementiert. Es bietet eine graphische Veranschaulichung des

Netzwerks, das die Services verbindet und sich dynamisch verändert, wenn sich die Konfiguration ändert.

7.4 Vorteile

Dank CORBA kann man die Programmiersprache frei wählen. Die meisten Services wurden in C++, Java und Python implementiert. CORBA läuft auf fast jeder Plattform oder Hardware. Die Entwickler waren gezwungen Komponenten zu benutzen, die als eigene Prozesse liefen. Das hat dazu geführt, dass die Kommunikation ganz klar definiert und eine schnelle Prototypenbildung möglich war.
Die Integration von Hard- und Software anderer Forschungsgruppen ist ganz leicht, denn es können um diese Komponenten herum „Wrappers" programmiert werden. Diese sind in derselben Programmiersprache geschrieben wie die Komponenten und werden dann mittels CORBA eingebunden.
Nicht zuletzt ist „Remote monitoring" mit Hilfe von DWARF Interactive Visualisation Environment Tools möglich.[10]

8. Studierstube

Studierstube ist eine Umgebung für die Entwicklung von kollaborativen Augmented Reality Anwendungen. Das Projekt ist an der Technischen Universität Wien entstanden. Der Name macht auf einen Raum aufmerksam, in dem spezielle Computertools vorhanden sind, um AR zu realisieren. Das Konzept umfasst sowohl Hardwareeinstellungen von z.B. Display und Geräte für die Interaktion mit dem System, als auch ein Softwareframework, das so genannte Studierstube Application Programmer's Interface (StbAPI). StbAPI ist realisiert als ein Set von C++ Klassen, aufbauend auf Open Inventor (OIV).
Studierstube unterstützt eine große Auswahl von Hardware und Software:

- Input: Trackers, Arttoolkit, Desktophardware
- Output: diverse HMDs, Vitual Table, Virtual Showcase usw.

Zu der Hardware gehört auch das Personal Interaction Panel, das speziell für Studierstube entwickelt worden ist. Es ist ein zweihändiges Tool für die Interaktion mit virtuellen Objekten und besteht aus einem flachen Brett und einem Stift (mit 1 oder 2 Knöpfe).[12]

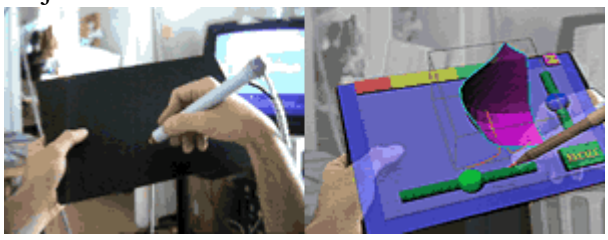


Abbildung 11: Überlagerung des Panels mit Graphiken [12]

9. Ausblick

Der Entwurf von objektorientierter Software ist auch für Anfänger einfach. Es gibt sehr viel Dokumentation, Tutorials, Bücher und weitere Informationsquellen. Augmented Reality hingegen ist ein neues Gebiet und ist von einer Standardisierung noch weit entfernt. Informationen sind schwer zu finden.
Die meisten aktuellen Augmented Reality Anwendungen konzentrieren sich auf der Entwicklung von AR Basistechnologien. Viele europäische Projekte entwickeln AR Software für spezielle Domänen, wie zum Beispiel ARVIKA, STAR, usw. Die Zukunft sind Projekte, die allgemeine Frameworks anbieten. Die Frameworks sind eine gute Basis von AR Software für verschiedene Anwendungsgebiete. DWARF und Studierstube sind solche Projekte. Aktuelle Forschungen die DWARF benutzen sind z.B. Ubiquitous Tracking, Ski, AiRcraft usw. Beispiele für AR die Studierstube benutzen sind Mobile Colloborative Augmented Reality, Construct3D, Virtual Dressmaker und weitere. Das AMIRE Projekt versucht zusätzlich zur Hilfssoftware und Frameworks eine allgemeine Herangehensweise zur Entwicklung von AR Anwendungen zu entwickeln. In Zukunft könnten solche und ähnliche Forschungen sich als Standards etablieren, um so die Entwicklung von AR Anwendungen weiterhin zu erleichtern und dadurch auch einem größeren Publikum zugänglich zu machen.

10. Quellen

[1] Emmanuel Dubois, Philip Gray, Laurence Nigay – ASUR++: a Design Notation for Mobile Mixed Systems**. Proceedings of MobileHCI'2002.**

[2] Emmanuel Dobois, Paulo Pinheiro da Silva, and Philip Gray – Notational Support for the Design of Augmented Reality Systems**. Proceedings of DSV-IS'2002.**

[3] Thema UMLi http://www.cs.man.ac.uk/img/umli/index.html

[4] Prof. Hußmann – 8. Digitale Filmverarbeitung. In Vorlesung Medientechnik

[5] Projekt Amire http://www.amire.net/index.html

[6] Paul Grimm, Michael Haller, Volker Paelke, Silvan Reinhold – AMIRE – Authoring Mixed Reality**. The First IEEE International Augmented Reality Toolkit Workshop, 2002.**

[7] Ralf Dörner, Christian Geiger, Michael Haller, Volker Paelke – Authoring Mixed Reality A Component and Framework- Based Approach**. IWEC 2002.**

[8] Projekt DART http://www.cc.gatech.edu/projects/ael/projects/dart.html

[9] Projekt ARVIKA http://www.arvika.de

[10] Projekt DWARF http://wwwbruegge.in.tum.de/DWARF/WebHome

[11] Prof. Bernd Bruegge Ph.D., Prof. Gudrun Klinker, Ph.D. – DWARF Distrubuted Wearable Augmented Reality Framework**. White Paper.**

[12] Projekt Studierstube http://studierstube.org/doc/stb/