State-of-the-Art Report (STAR)

Titel:	STAR
Kontakt:	Jens Neumann (Loewe Opta GmbH, jneumann@loewe-komp.de)
Datum/Rev.:	22.10.2003 / Version 1.0: Erstellung 31.12.2004 / Version 2.0: Erste vollständige Version 14.1.2004 / Version 2.1.- 2.4: Korrekturen und Vereinheitlichung 2.2.2004 / Version 3.0: Ergänzungen und Veränderungen nach Review 13.8.2004 / Version zum Download, d.h. ohne viele Bilder um die Größe zu reduzieren.

ABBILDUNGSVERZEICHNIS

1. Szenarien / Visionen

1.1. Pebbles

Im Projekt Pebbles [>NMH02] wird eine personalisierte, universelle Fernbedienung realisiert (Personal Universal Controller, PUC). Der PUC besteht aus einem PDA (Pocket PC oder Palm), der drahtlos mit Geräten wie einer Stereoanlage oder einem PC verbunden wird. Den Geräten wird ein spezieller Hardware-Adapter vorgeschaltet (z.B. Kontrolle der Stereoanlage über Infrarot-Schnittstellen).

Jedes Gerät liefert eine auf XML basierende Selbstbeschreibung seiner Funktionen. Diese wird auf dem PDA dazu verwendet dynamisch ein Sprach-Graphik-Interface zu realisieren. Das Graphik-Interface besteht aus einer auf PersonalJava basierenden GUI, die dynamisch entsprechend der Funktionen des PDAs generiert wird. Das Sprach-Interface besteht aus einer Spracherkennung und einem Parser, die dynamisch an die Gerätefunktionen angepasst werden. Die im Demonstrator angebundenen Geräte sind eine Stereoanlage mit Infrarot-Schnittstelle, ein Sony Camcorder mit IEEE 1394 Schnittstelle und ein Mitsubishi Videorekorder, der über HAVi angesprochen wird. Außerdem können der WinAmp und der Windows Media Player angesteuert werden.

In Pebbles wird keine Selbstorganisation von Geräteensembles betrachtet. Die PUC dient lediglich zur Fernsteuerung von einzelnen Geräten. Eine Kombination von verschiedenen Gerätefunktionalitäten (z.B. den Videostream auf dem PDA-Bildschirm abspielen und den Audiostream auf dem Lautsprecher) ist nicht vorgesehen. Aufbauend auf einer intelligenten Selbstorganisation von Geräteensembles in Dynamite könnte eine Pebbles-ähnliche dynamische Generierung von Sprach-/Grafik-Interfaces dazu verwendet werden, um multimodale Eingaben für Dynamite-Geräteumgebungen zu realisieren. Weitere relevante Überschneidungen mit Dynamite existieren nicht.

1.2. Oxygen

Das Projekt Oxygen [>Oxygen2003] vom MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, USA, verfolgt die Vision des human-centered Computing. Der Name Oxygen leitet sich tatsächlich aus dem Wort Sauerstoff her, denn genauso wie Sauerstoff den Menschen überall umgibt, soll Oxygen pervasive Computing möglich machen. Und zwar ohne dass der Mensch es unmittelbar merkt oder kompliziert bedienen muss. Die Visionen sind weitgesteckt:

- Konfigurierbare generische Device (z.B. Handheld oder embedded Devices)
- Zielorientiert, benutzerorientiert und präferenzorientiert (Zitat: „...helping us to do more while doing less“)
- Natürliche Kommunikationsmöglichkeiten mit Sprache und Gestik, wobei Sprache und Gestik dazu dienen, die individuellen Ziele auszudrücken
- Allg. Produktivitätssteigerung: Ausführung wiederholter Arbeiten, Informationssuche, Perfektionierung von Zusammenarbeit
- „Computer“ folgen dem Menschen auf seinem Weg. Das ist absolut wörtlich gemeint, d.h. die Arten der Computer werden immer dort ausgeführt, wo er sich gerade befindet und ändern den Ort, wenn der Benutzer seinen Ort ändert.

Das Projekt Oxygen ist genaugenommen kein integriertes Projekt. Unter der Bezeichnung Oxygen laufen verschiedene Forschungsaspekte und Arbeitsgruppen beim MIT zusammen. Die gemeinsame Klammer der hierbei entwickelten Technologien stellen die Szenarios dar, die von den einzelnen Gruppen in Teilen bearbeitet werden. Ob es eine Gesamtintegration geben wird, lässt sich beim bisherigen Stand nicht beurteilen. Einen guten Überblick über die Szenarios, beteiligte Abteilungen und die Technologien von Oxygen finden sich im Vortrag von Victor Zue, Direktor MIT Laboratory for Computer Science unter [>Oxygen_Overview2002].

1.2.1. Die Herausforderungen und der Ansatz von Oxygen

Zur Verwirklichung dieser visionären Ziele wurden mehrere Teilziele herauskristallisiert, die das Gesamtsystem Oxygen beschreiben:

- *Pervasive* – Das System muss überall sein
- *Embedded* – es muss in die Gegenstände des Alltags integriert sein
- *Nomadic* – es muss Benutzern und Systemteilen möglich sein, sich frei zu bewegen
- *Adaptable* – es muss sich auf veränderbare Benutzeranforderungen anpassen können

- *Powerful*, *efficient* – Benutzeranforderungen dürfen die einzigen Grenzen sein, keine Hardwareressourcen
- *Intentional* – Benutzer müssen Dingen Namen und eigene Bezeichner geben können, um dann z.B. „den nächstgelegenen Drucker“ sagen zu können
- *Eternal* – es muss immer „on“ sein. Es darf kein Shutdown und kein Reboot geben (Ewigkeit). Komponenten dürfen kommen und gehen, aber das Gesamtsystem muss immer an sein.

Der Ansatz von Oxygen will spezifische Benutzer- und Systemtechnologien vereinen, wobei die Benutzertechnologien sich auf die Sprach- und Visualisierungsinteraktionen konzentrieren. Die Zugangstechnologien sind dabei aufgeteilt (siehe Abbildungen auf http://oxygen.lcs.mit.edu/Overview.html):

- E21s – Enviro21s können die direkte Umgebung manipulieren (Lautsprecher, Lichtsteuerung, Ventilatoren etc.)
- H21s – Handy21s, ermöglicht Kommunikation mit dem System über Handhelds etc.
- N21s – Networks21s, zur Lokalisierung von Personen, Geräten und Ressourcen
- O21s – Software, die sich auf Änderungen der Umgebung oder der Benutzervariablen adaptieren kann

Eigenschaften der Geräte in Oxygen (sowohl der mobilen als auch stationären):

- Verfügen über Mechanismen zur Kommunikation
- Verfügen über „Eigenintelligenz“ (they are computation appliances)
- Bewahren die Anonymität ihrer Benutzer

1.2.2. Netzwerktechnologien in Oxygen

Durch dynamisch veränderbare Konfigurationen aufgrund der mobilen und stationären Geräte, die sich selbst identifizieren können, werden Regionen der Zusammenarbeit geschaffen. Dabei unterstützt das Netzwerksystem (N21s) verschiedene Kommunikationsprotokolle wie point-to-point, gebäudeweite Kommunikation und feldübergreifende Kommunikation. Zugleich unterstützt N21s einen komplett dezentralen Mechanismus zur *Naming*, *Location* und *Resource Discovery* sowie den sicheren Zugang zu Daten.

1.2.3. Schlüsselszenarios in Oxygen

Business Conference:
Kurz zusammengefasst, sollen in diesem Szenario Projektmitarbeiter unterschiedlicher Länder und unterschiedlicher Muttersprachen. Dabei sind die Telefoneinrichtungen in der Lage, in der jeweiligen Muttersprache das Anrufers Auskunft über An- oder Abwesenheit des gewünschten Gesprächspartners zu geben. Wollen die Projektmitarbeiter ein gemeinsames Meeting verabreden gleicht Oxygen die verschiedenen Kalender ab, schlägt mögliche Termine vor und erledigt die nötigen Hotel- und Flugreservierungen. Die persönlichen Handhelds sind danach in der Lage, den jeweiligen Mitarbeitern den Weg zum richtigen Gebäude und innerhalb des Gebäudes zum richtigen Raum zu weisen (→ H21 Visitors Guide). Gleichzeitig sind die Handhelds in der Lage, die nötigen Dateien für das Meeting auf Bedarf zu organisieren. Die Interaktion mit den Handhelds findet in natürlicher Sprache statt.

Guardian Angel:
Dieses Szenario beschreibt vielfältige Erleichterungen im Haushalt. Telefonanrufe werden auf die vorhandenen Lautsprecher umgeleitet, ebenso die Gegensprechanlage der Türe. Vielfache Sensoren verhindern das Überlaufen der Badewanne oder die Badetemperatur. Der intelligente Fernseher weist auf Vorlieben im aktuellen Fernsehprogramm hin. Ebenso kann Oxygen die Einnahme von Medikamenten und durch Videoaufzeichnungen und Vergleich von persönlichen Videos Schwankungen im Gesundheitszustand erkennen, der dann von einem Arzt interpretiert werden kann. Mittels Oxygen kann in Notfällen direkt Kontakt mit der Notfallzentrale aufgenommen werden. Das Szenario Guardian Angel beschreibt das Leben von älteren Menschen, denen mit technischen Möglichkeiten das autonome Leben in der eigenen Wohnung erleichtert werden soll.

1.2.4. Bisher realisierte Anwendungen

Unter [>Oxygen2003] lassen sich diverse Videos betrachten, die bisher realisierten Anwendungen demonstrieren. Alle weisen sehr hohe Tendenzen zur Interaktion mit Sprache auf. Die bisher realisierten

Demonstratoren sind stand-alone Lösungen. Daher scheint das Projekt Oxygen sich in mehrere (voneinander unabhängige) Teilprojekte sich aufzugliedern:

1.2.4.1. Device Technologies

Es wurde ein Handheld (siehe auch Abbildungen auf http://oxygen.lcs.mit.edu/H21.html) entwickelt, welches Interaktionen mittels Mikrophon, Lautsprecher, Kamera und Display erlaubt. Zugleich wurde auf Optimierung des Stromverbrauchs geachtet.

1.2.4.2. Network Technologies

Analog zu GPS wurde ein System (Cricket Location Support System) entwickelt, welches die Lokalisation von Menschen innerhalb eines Gebäudes ermöglicht (siehe auch Abbildungen auf http://oxygen.lcs.mit.edu/Network.html).

Zusammen mit dem Intentional Naming Service ist es dadurch möglich, den gewünschten Dienst (z.B. Musik abspielen, Datei ausdrucken etc.) immer an dem Gerät auszuführen, das dem Benutzer örtlich am nächsten ist (Szenario: Musik, die dem Benutzer beim Gang durch die Räume folgt). Ebenso ist die Führung einer Person durch Gänge des Gebäudes zu einem bestimmten Raum möglich. Beide Szenarios werden durch Videos demonstriert. Diese Ziele werden von der Gruppe „Networks and Mobile Systems“ [>NMS2003] verfolgt.

Intentional Naming Service

Namen in INS beschreiben mehr als nur Netzwerklokalisierungen, sondern auch Eigenschaften und die Attribute von Ressourcen und Daten. Hierzu gibt es auf Applikationsebene sog. Intentional Name Resolvers, die Informationen über Ressourcen speichern und dadurch Anfragen auflösen können. Die gespeicherten Informationen liegen in sog. Twine-Routern [>Twine2003] und [>Balazinska2002], von denen immer mehrere verfügbar sind und somit eine gewisse Art an Dezentralisierung erreicht wird. Die einzelnen Knoten sind in der Lage über benachbarte Knoten zu lernen und die verfügbaren Ressourcen in der Nachbarschaft zu lernen. Auf [>Twine2003] steht eine Open-Source Version von Twine für Java in der Version 1.4.0 zur Verfügung, zusammen mit ausführlichen Dokumentation und einigen Beispielanwendungen.

1.2.4.3. Software-Technologien

Auch der zugrundeliegenden Software Technologie liegen verschiedene Visionen zu Grunde:

- Adaptierung auf Benutzer, auf die Umgebung, auf deren Veränderungen und Fehler
- Computerdienste folgen dem Benutzer (im direkten Wortsinne, siehe das Beispiel, in dem die Musik jeweils aus den Lautsprechern kommt, die dem Benutzer am nächsten stehen)
- Netzwerke adaptieren sich auf Änderungen der beteiligten Geräte
- Geräte adaptieren ihre Kommunikationsprotokolle auf Geräte der näheren Umgebung auf den Benutzer und seine Anforderungen und auf die gegenwärtigen Umgebungszustände
- Softwaresysteme sind in der Lage auf Fehler zu reagieren, indem sie eigene Diagnosemittel besitzen und alternative Ansätze wahrnehmen können

Zusätzlich soll die in Oxygen eingesetzte Software adaptierbar sein, d.h. sie soll schnell anpassbar sein, an neue Benutzeranforderungen sowie an neu hinzukommende Software. Hierzu soll es auch möglich sein, Software just-in-time auf Handhelds zu bringen sowie die automatische Installation von neuen Softwarekomponenten und deren Updates.

Die in Oxygen verwendete Architektur wird in drei Grundsätze aufgeteilt:

Abstraction

Abstraktionen werden eingesetzt um Komponenten zu charakterisieren, die Berechnungen ausführen und die dazu gehörigen Objekte besitzen. Hierzu sollen Abstraktionen helfen, um adaptierbare Komponenten und deren Objekte zu benutzen, indem sie:

- Den Zugang zu Komponenten erlauben
- Schnittstellen anbieten zur Objektbenutzung, zur Ergänzung oder zu Ersetzung von Objekten

- Schnittstellen anbieten, die Sprache, Bilder und sensorische Daten unterstützen
- Bedingungen und Ereignisse abstrahieren, um Berechungen zu separieren (Eventmodell)
- Definition von sog. „cutpoints“ erlauben, um das System fehlerreaktiv zu machen

Specifications

Die oben beschrieben Abstraktionen werden in den Spezifikationen explizit gemacht. Hier werden:

- Systemkonfigurationen, verfügbare Module und Funktionalitäten definiert
- Repositories für Code (der über das Network anderen Geräten wie Handhelds verfügbar gemacht werden soll) angelegt
- Gegenseitige Abhängigkeiten von Modulen festgelegt
- Verhalten von Modulen festgelegt

Persistent object store with transactional semantics

In einem gemeinsam verwalteten Object-Store werden Code, Datenobjekte und Spezifikationen hinterlegt, die von allen Oxygen-Technologien verwendet werden können. Durch diese zentrale Stelle wird die Integrität von Updates etc. gewährleistet. Hierzu wurde ein Semantik entwickelt, die den Zugang und das Auffinden von Daten erleichtert.

1.2.4.4. Software

Als Software werden unterschiedliche Systeme von unterschiedlichen Arbeitsgruppen des MIT verwendet.

MetaGlue

Zitat [>Coen1999]: „*Distributed modular systems need computational glue*“.

MetaGlue ist eine Erweiterung der Java Programmiersprache, die eine neue Klasse (*Agent*) einführt. Mit dabei ist eine MetaGlue Virtual Machine, die ebenso eine Erweiterung der Java Virtual Machine darstellt. Auf ihr laufen die Agenten. Agenten kommunizieren miteinander, indem sie ihre Methoden gegenseitig aufrufen. Dadurch soll es sehr einfach sein, bereits bestehende Java-Programme in MetaGlue-Programme zu erweitern (agent wrapping). Auf jedem Rechner, der im System ist, wird die MetaGlue Virtual Machine gestartet. Es handelt sich somit wirklich um ein verteiltes System.
Folgende Eigenschaften werden von MetaGlue unterstützt:

- Konfigurationsmanagement: hierzu verfügt MetaGlue über eine eigene SQL-Datenbank, auf die mittels Attributwerten zugegriffen werden kann. Agenten müssen Werte somit nicht selber dauerhaft speichern, sondern können dies mittels der Meta Virtual Machine in einer SQL-Datenbank tun.
- Agentenkonfiguration: Agenten können spezifische Anforderungen an die Virtual Machine stellen, auf der sie laufen. Können diese Anforderungen nicht erfüllt werden (z.B. durch teilweisen Systemausfall, sorgt MetaGlue dafür, dass der Agent auf einer anderen Maschine (auf einem anderen Rechner) ausgeführt wird.
- Agentenverbindungen: Wenn Agenten einen Dienst von einem anderen Agenten benötigen, besorgen sie sich seine Referenz (Agent speechSynthesizer = reliesOn(speech.Synthesizer)) und rufen die dazu nötige Methode auf (speechSynthesizer.say(„Hello“)). Die MetaGlue Virtual Machine versucht einen Agenten der zugehörigen Klasse zu finden bzw. selbst zu starten.
  MetaGlue verwaltet also Agentenfunktionalitäten und keine direkten Bezeichner. Über Auswahlprozesse in MetaGlue ist nichts bekannt. Es gibt offensichtlich, dass der erste gefundene Agent den Zuschlag des Auftrages erhält.
- Agentenzustand: Mittels Java-Methoden können Agenten ihren aktuellen Status in der SQL-Datenbank speichern und zu gegebener Zeit wieder laden.
- Management verteilter Ressourcen: MetaGlue stellt einen Satz von sog. Dealer-agents zur Verfügung, die verantwortlich sind, die Ressourcen des restlichen Systems zu verteilen.
- Event Broadcasting: Agenten können sich auf Events registrieren, die sie interessieren (das sind Nachrichten, die von anderen Agenten zu anderen Agenten gehen). Man bekommt also somit eine Kopie als Event.

Die Arbeitsgruppe stellt eine Version von MetaGlue für das JDK V1.3.x inklusive Beschreibung der API und ausführlicher Dokumentation [>Metaglue2003] zur Verfügung.

Es ist jedoch zu sagen, dass die nächste Version von MetaGlue für den Herbst 2002 angekündigt ist, und noch nicht erschienen ist. Es ist unklar, ob an diesem System noch weiter gearbeitet wird.

CORE

Zu CORE finden sich leider keine publizierten Papers der beteiligten Arbeitsgruppe (Oxygen Research Group LCS beim MIT [>Oxygen ResearchGroup2003]).
Bei CORE scheint es sich um einen Router zu handeln, der es „pervasive computing devices“ erlaubt miteinander zu kommunizieren. Es ist plattform-unabhängig in Java implementiert und bildet die Schicht über der Netzwerkschicht ab (ähnlich EMBASSI-Router, JATLite u.a.). Dabei gestattet CORE den aktuellen Zustand zu beobachten. Auch scheinen gewisse Kommunikationsregeln definiert werden zu können. Der Hintergrund von CORE ist aber ein anderer. CORE dient dem Debugging der angeschlossenen Geräte, deren Verhalten und deren Kommunikationsarten (siehe: [>CORE2003]). CORE wird oft zu Studentenprojekten und Schulungen eingesetzt, so findet sich z.B. ein Vortrag über den Aufbau eines Präsentations-Managers auf Basis des CORE-Systems: [>CORE_PresentationManager2001].

GOALS

GOALS ist ein Planungssystem für die automatische Konstruktion von Komponenten, in dem eine oder mehrere Technologien ausgewählt werden, die als „Rezept“ dienen können, um eine Klasse von Zielen zu erfüllen. Ziele werden aufgelöst, indem Systemobjekte, die den aktuellen Kontext widerspiegeln und ein Satz von anwendbaren Technologien gegeneinander abgewogen werden. Hierzu wird ein sog. Zielbaum erstellt und daraufhin der am meisten erfolgversprechende Ast zur Implementierung empfohlen.
Mit dem Projekt EMBASSI verglichen, scheint es sehr stark verwandt mit der Idee zur Assistenzentwicklung von GUIDEAS zu sein. Die Ergebnisse des GOALS-Mechanismus sind Pebbles.

Pebbles

Pebbles (nicht zu verwechseln mit dem „The Pittsburgh Pebbles PDA Project“, siehe Kapitel 1.1 in diesem STAR) sind plattform-unabhängige Softwarekomponenten, die vom Planungsmechanismus von GOALS dynamisch zusammengebaut werden. Die dadurch entstandenen Beschreibungen enthalten eine Mischung aus formalen Schnittstellenbeschreibungen, informellen Beschreibungen (User Manuals) und anderen Informationen zusammen mit Code für Tests.

Click

Click ist eine Software-Architektur, um flexible und konfigurierbare Router zu erstellen. Ein Click-Router ist aufgebaut aus Modulen, die Pakete verarbeiten können. Diese werden „Elements“ genannt. Dazu gibt es Routerfunktionen, wie Klassifikation von Paketen, Queueing, Scheduling und das Bereitstellen von Schnittstellen mit Geräten im Netzwerk. Eine Konfiguration eines Routers ist ein gerichteter Graph, der an seinen Eckpunkten solche Elemente hat. Ein Standard Click-Router hat 16 Elemente, die sich auch um Ethernet Switches und IP Tunneling kümmern.

1.2.5. Zusammenfassung / Einschätzung

Oxygen ist ein Mantel, der unterschiedliche Tätigkeiten und Projekte innerhalb des MIT zusammenfasst. Verschiedenste Projekte, Abteilungen und Technologien beschäftigen sich mit der Mensch-Technik-Interaktion und dem Pervasive Computing. Die hier erzielten Ergebnisse und Technologien werden unter dem Oberbegriff Oxygen zusammengefasst. Die Heterogenität belegt schon die Tatsache, dass unterschiedliche Systeme (siehe z.B. Click, MetaGlue oder Twine) eingesetzt werden, um Komponentenkommunikation zu ermöglichen.

Für DynAMITE scheint folgendes wichtig zu sein:
Es ist dem MIT mit der Initiative Oxygen gelungen, den Begriff Pervasive Computing und damit verbundene „Keywords“ wie Embedded, Nomadic oder Adaptable definiert und bekannt gemacht zu haben. Ein Projekt, das sich der Weiterentwicklung dynamischer Systeme und der Selbstorganisation widmet, wird an Oxygen nicht vorbeikommen. Oxygen belegt hier somit eine wichtige, wenn nicht die wichtigste Referenz.
Technologisch scheint Metaglue am interessantesten zu sein (siehe auch eigener Abschnitt in diesem STAR). Hier wurde interessante Ansätze wie das Einfrieren von Agentenzuständen, die selbstständige Migration von

Agenten und das Auffinden von Diensten eingeführt. Ein Agent und dessen Funktionen sind also nicht an einen spezifischen Ort gebunden, oder in anderen Worten: ein Agent ist nicht lokalisiert. Sondern er wird an einem Ort gestartet, dessen Ressourcen diese Möglichkeit zulassen. Metaglue, aber auch anderen Realisationen von Oxygen, liegt keine Topologie zugrunde. Es ist also nicht so, dass erkannte Zustände oder Benutzeräußerungen auf Ebenen nächster Verarbeitung gepostet werden (z.B. durch Eventmechanismen). Agenten / Komponenten rufen Methoden unbekannter Agenten auf (was in sich schon einen Fortschritt gegenüber anderen Ansätzen, wie KQML oder FIPA bedeutet). Dahingehend ist Metaglue mit der Open Agent Architecture vergleichbar. Über Konfliktlösungsmechanismen oder Auswahlverfahren möglicher Zielagenten ist in Metaglue aber nichts bekannt. Hier scheinen der Ort der Ausführung oder die Metapher des „first comes, first serves“ zu bestimmen. Dahingehend ist die Intelligenz der verteilten Ausführung auf den auftraggebenden Agenten (d.h. derjenige, der einen Dienst sucht und die entsprechenden Methoden des Zielagenten aufruft) verlagert zu sein.

Es ist unklar, in wie weit das MIT die Metaglue-Technologie noch weiter verfolgt. Die nächste Version ist für Herbst 2002 angekündigt und bisher nicht erschienen. Die Initiative dahingehend scheint erlahmt zu sein.

Literaturreferenzen zu [Kapitel 1.2]

[>Oxygen2003] MIT Project Oxygen, Pervasive, Human-centered Computing, http://oxygen.lcs.mit.edu/

[>Oxygen_Overview2002] Vortrag von Victor Zue, Direktor MIT, Projektübersicht Oxygen http://akseli.tekes.fi/Resource.phx/tivi/nets/netsvuosiseminaariesitykset140502.htx.liite.liitteet.4.pdf

[>Oxygen ResearchGroup2003]] Oxygen Research Group http://www.org.lcs.mit.edu/

[>NMS2003] Die Arbeitsgruppe Networks and Mobile Systems beim MIT: http://nms.lcs.mit.edu/

[>Twine2003] Resource Discovery System Twine: http://nms.lcs.mit.edu/projects/twine/

[>Balazinska2002] Magdalena Balazinska, Hari Balakrishnan, and David Karger**,** INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery, Pervasive 2002 - International Conference on Pervasive Computing, Zurich, Switzerland, August 2002 (zu finden unter: http://nms.lcs.mit.edu/papers/)

[>Metaglue2003] MetaGlue: http://www.ai.mit.edu/projects/iroom/metaglue/

[>CORE2003] CORE-Projektseite: http://org.lcs.mit.edu/projects/?action=viewProject&projectID=3&projectGroup=CORE).

[>CORE_PresentationManager2001] Vortrag über einen Präsentations-Manager auf Basis von CORE, http://www.org.lcs.mit.edu/6.964/course-core-overview.pdf

[>Coen1999] Michael Coen, Brenton Phillips, Nimrod Warshawsky, Luke Weisman, Stephen Peters, and Peter Finin. Meeting the Computational Needs of Intelligent Environments: The MetaGlue System. In Proceedings of MANSE'99. Dublin, Ireland. 1999

[>Click2000] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, M. Frans Kaashoek, The Click Modular Router, ACM Transactions on Computer Systems **18**(3), August 2000, pages 263-297

1.3. EasyLiving

EasyLiving ist ein Projekt, das an Microsoft Research durchgeführt wurde. Ziel von EasyLiving ist die Entwicklung von Architektur und Technologien für intelligente Umgebungen, in dennen verschiedene zusammenarbeitende Geräte dem Benutzer Zugriff auf Informationen und Dienste bereitstellen.

Dabei konzentriert sich EasyLiving hauptsächlich auf die Entkopplung des Benutzers vom PC-Desktop bei der Durchführung seiner „PC-focused" Aktivitäten. Um dies zu erreichen, trennt EasyLiving „hardware device control", „internal computation logic" und „user interface presentation". Vielmehr kann der Benutzer in seinem „intelligenten Raum" an mehreren Orten und unter verschiedenen Situationen mit seinen Applikationen (und auch mit sonstigen computer controllable devices wie Licht) interagieren. Dabei erlaubt die intelligente Umgebung geeignete Ein- und Ausgabegeräte für eine bestimmte Situation des Benutzers zu benutzen (siehe Braumitt2000).

Weiter erlaubt EasyLiving (Room Controller) dem Benutzer die physische Umgebung direkt zu manipulieren. Hierzu werden die in einem Raum befindlichen Geräte und deren räumliche Anordnung und Beziehungen untereinander visualisiert. Der Benutzer kann ein bestimmtes Gerät (z.B. Lampe) auswählen – ohne den Netzwerknamen oder Adresse zu kennen – und die Angebotenen Dienste benutzen (z.B. ausschalten) (Braumitt2000).

EasyLiving Architektur (siehe Braumitt2000) besteht aus den Komponenten ***EasyLiving Geometric Model, Perception, Devices*** (I/O, devices dedicated to providing computational capabilities), ***Service Description*** sowie das ***InConcert Middleware.*** (Braumitt2000)

Der entwickelte Prototyp unterstützt einen Raum (Zimmer) in denen bis zu 3 Personen arbeiten können (Braumitt2000).

1.3.1. EasyLiving Geometry Model

The EasyLiving Geometric Model (EZLGM) provides a general geometric service for ubiquitous computing, focusing on in-home or in-office tasks in which there are myriad I/O, perception, and computing devices supporting multiple users. The EZLGM is designed to work with multiple perception technologies and abstract the application (including its user interface) away from any particular sensing modality. It is aimed at geometry "in the small", that is, for applications which may require sub-meter localization of entities in the environment. Integrating this model with localization services for larger scales remains an open issue. The base item in the EZLGM is an entity, which represents the existence of an object in the physical world. Measurements are used to define geometric relationships between entities. In particular, a measurement describes the position and orientation of one entity's coordinate frame, expressed in another entity's coordinate frame. Since objects in the physical world (as well as virtual objects like service regions) have some physical expanse, this can also be expressed as an extent in the geometric model using a polygon described in the coordinate frame of the entity. Additionally, measurements can have an uncertainty associated with them, expressed as a covariance matrix on parameters of the transformation.
Once a set of measurements has been provided to the geometric model, the model can be queried for the relationships between entities' frames. The measurements describe an undirected graph, where each vertex is the coordinate frame of an entity, and each edge is a measurement, as described above. If at least one path exists between two frames, then the graph can be processed to produce a single geometric relationship between the frames. Since a particular queried relationship may not have been previously directly measured, the response typically involves the combination of multiple measurements; uncertainty information is used to properly merge multiple redundant measurements as needed. Region-intersection queries are also supported, allowing, for example, selection of all devices within a particular radius of a user. In the example scenario, the person tracking software continuously updates the measurement which describes the geometric relationship between Tom/Sally and the coordinate frame of the sensor which is observing them. Whatever process is responsible for keeping Tom's session available on nearby devices can query EZLGM for all devices that have service areas that intersect with his location. The process first looks at types and availability to determine the set of devices which could provide the display, text input, pointing, etc. It then further prunes the list by considering the physical constraints (e.g. visibility) and electronic constraints (e.g. availability), in order to reach a set of usable, available, and physically-appropriate devices.Visibility can be checked by examining all entities along the line of sight between the user and the device and ensuring none have an extent present which represents something that physically blocks Tom's view. Then, once Tom's location is stable with respect to a set of devices, the session can be directed to automatically move. (Braumitt2000)

1.3.2. Relevanz für Dynamite

Das EasyLiving System erlaubt es dem Benutzer in seinem Smart Environment an mehreren Orten und unter verschiedenen Situationen mit seinen Applikationen (und auch mit sonstigen computer controllable devices wie Licht) zu interagieren. Dabei konzentriert sich das System hauptsächlich auf die Entkopplung des Benutzers vom PC-Desktop bei der Durchführung seiner „PC-focused" Aktivitäten konzentriert. Um dies zu erreichen, trennt EasyLiving „hardware device control", „internal computation logic" und „user interface presentation".

Ein Schlüsselkonzept von EasyLiving ist das „EasyLiving Geometric Model" (s. **Fehler! Verweisquelle konnte nicht gefunden werden.**), welche die physische Anordnung und die relative räumliche Beziehungen von sogenannten Entities (People, Places, Things, I/O Devices), die sich in einem (geschlossenen) Raum befinden, repräsentieren kann.
Diese Weltmodell wird ständig von „Sensing Devices" mit „perceptual information" versorgt, welche Informationen über „the state of the World, such as location of people" liefern (Braumitt2000). Das EZLGM definiert weiter Service-Bereiche für jades Objekt. Benötigt der Benutzer einen bestimmten Service (Präsentation Service), so wird das Objekt (Monitor) ausgewählt in dessen Service-Bereich die Aktion ausgeführt werden soll bzw. der Benutzer sich befindet (d.h. der für Benutzer am besten sichtbare Monitor). Das Konzept des Raummodell (mit relativen Koordinatensystem und Service-Bereich) ist für das DynAMITE-Projekt interessant.

Nachfolgend werden einige Abgrenzungen zur DynAMITE skizziert bzw. einige für DynAMITE interessante Eigenschaften von EasyLiving hervorgehoben:

- EasyLiving unterstützt keine zielbasierte Interaktion und auch keine dynamische Ensembles im Sinne von EMBASSI und DynAMITE, die Nutzerziele umsetzen können.
- EasyLiving hat eine zentralistische Architektur. Es werden sogenannte Room Server eingesetzt.
- EasyLiving ist dynamisch erweiterbar, da die dynamische Hinzunahme von neuen Eingabe-Ausgabegeräten sowie Positionierungssystemen unterstützt wird. Allerdings sind die funktionalen Rollen dieser Geräte fest vorgegeben. Es können also nur Geräte von bestimmten Typen hinzukommen, welche das System bereits kennt und für die Realisierung der Szenario benötigt (z.B. Display, Monitor, Beamer für Output devices).
- EasyLiving ist eine Anwendung für die Unterstützung des Benutzers in einer bestimmten Umgebung (context-bound; also nur zu Hause oder nur bei der Arbeit). EasyLiving kann nicht umgebungsübergreifend eingesetzt werden, wie etwa personal assistant systems es ermöglichen.
- EasyLiving unterstützt hauptsächlich Aktivitäten, die man unter „Environment Control" klassifizieren würde
- EasyLivings Interaktionsparadigma ist Augmentation
- Im Gegensatz zu EasyLiving konzentriert sich DynAMITE auf die Bereitstellung von generischen Frameworks und Middleware, welche die ad-hoc Realisierung von solchen Systemen wie EasyLiving unterstützen.

Literaturreferenzen zu [Kapitel 1.3]

[Braumitt2000] B. Braumitt, B. Meyers, J. Krumm, A. Kern and S. Shafer (2000). „EasyLiving: Technologies for Intelligent Environments". In Handheld and Ubiquitous Computing, September 2000. http://www.research.microsoft.com/easyliving/Documents/2000%2009%20Barry%20HUC.pdf}

[MicrosoftResearch2003] Microsoft research, EasyLiving Web site, Retrieved on 07.12.2003 from http://www.research.microsoft.com/easyliving/

1.4. Ambiente / Roomware

Das Projekt Roomware vom Fraunhofer/GMD [>SGH99] beschäftigt sich mit Multi-User Interaktionen mit Hilfe von großen Displays im Kontext eines Meeting-Szenarios. Der Begriff Roomware bezeichnet die Verbindung von Möbeln wie Tischen oder Stühlen mit Computern. Innerhalb des Projektes wurden drei Typen von Geräten realisiert.

Die DynaWall ist eine großer Touchscreen-Display mit den Abmessungen 4.5mx1.1m. Aufgrund der Größe können Interaktionstechniken wie Drag&Drop nur schwer angewendet werden. Stattdessen wurden neue Konzepte definiert wie das „aufnehmen und hinlegen" von Objekten oder das „werfen" von Objekten. Der

InteracTable ist ein Tisch, auf dessen Oberfläche ein großer Display sitzt. Der Touchscreen unterstützt die Erkennung von Gesten, wie zum Beispiel das drehen des Bildes in die Perspektive des jeweiligen Benutzers. Der CommChair ist ein Sessel, der durch einen Laptop ergänzt wurde. Der CommChair unterstützt das Arbeiten mit „geshareten“ Arbeitsumgebungen. Weiterhin können Daten auf die im Raum befindlichen DynaWalls oder InteracTables übertragen werden, die sich zur Zeit vor dem Sessel befinden. Der CommChair ist ebenso wie der InteracTable vollständig mobil.

Der Schwerpunkt von Roomware ist die intelligente Interaktion über „gesharete“ Informationen mit Hilfe von großen Displays und einer verteilten Geräteinfrastruktur. Die Funktionalitäten der einzelnen Geräte werden dazu nicht explizit modelliert. Ebenso werden keine sich gegenseitig ergänzenden Gerätefunktionalitäten betrachtet. Das Projekt Roomware betrachtet keine Dynamik innerhalb der Geräteensembles. Allerdings können die innerhalb des Projektes untersuchten Techniken für Interaktionen mit mehreren Geräten für Dynamite relevant sein.

1.5. Embassi (Elektronische Multimediale Bedien- und Service Assistenz)

EMBASSI war eines der 6 Leitprojekte die von 1999 bis 2003 unter der Förderung des BMB+F in Deutschland zur Entwicklung neuer Mensch-Technik-Interaktionsmodelle liefen (neben MAP, Arvika, SmartKom, Morpha und Invite). EMBASSI war in mehrere Teilprojekte aufgeteilt. Unter anderem wurden Szenarios für den Privathaushalt, für Kraftfahrzeuge und für Terminalsysteme entwickelt. Da die Themenstellungen den im Projekt DynAMITE beteiligten Partnern weitestgehend bekannt sind, sollen hier nur die für DynAMITE relevanten Forschungspunkte besprochen werden. Diese wären die Entwicklung einer generischen Architektur, die auf KQML basierenden Kommunikationssysteme und deren in EMBASSI eingesetzten Lösungen, die Ontologien in EMBASSI sowie die Fortentwicklung der generischen Architektur zu Soda-Pop.

1.5.1. Architektur in EMBASSI

Die generische Architektur [>Kirste2001] der unterschiedlichen autonomen Komponenten in EMBASSI sind in einer zielbasierten auf den Nutzer ausgerichteten Topologie angeordnet. Diese Topologie folgt der sogenannten Kanalflusstopologie, in der Komponenten derselben Ebene für die Umwandlung der einzelnen Ontologien zuständig sind (siehe Abbildung 1).

Die Architektur ist grob in zwei große Blöcke geteilt. Die MMI-Ebenen wandeln die atomaren Benutzeräußerungen in Ziele des Benutzers um, während die ASSIST-Ebenen diese Ziele dazu benutzen, mit Hilfe der vorhandenen Geräte die Umgebung zu verändern, um diesen Zielen gerecht zu werden. Dies entspricht somit Blöcken zur Interpretation und Zielfindung und Blöcken zur Strategiebildung.

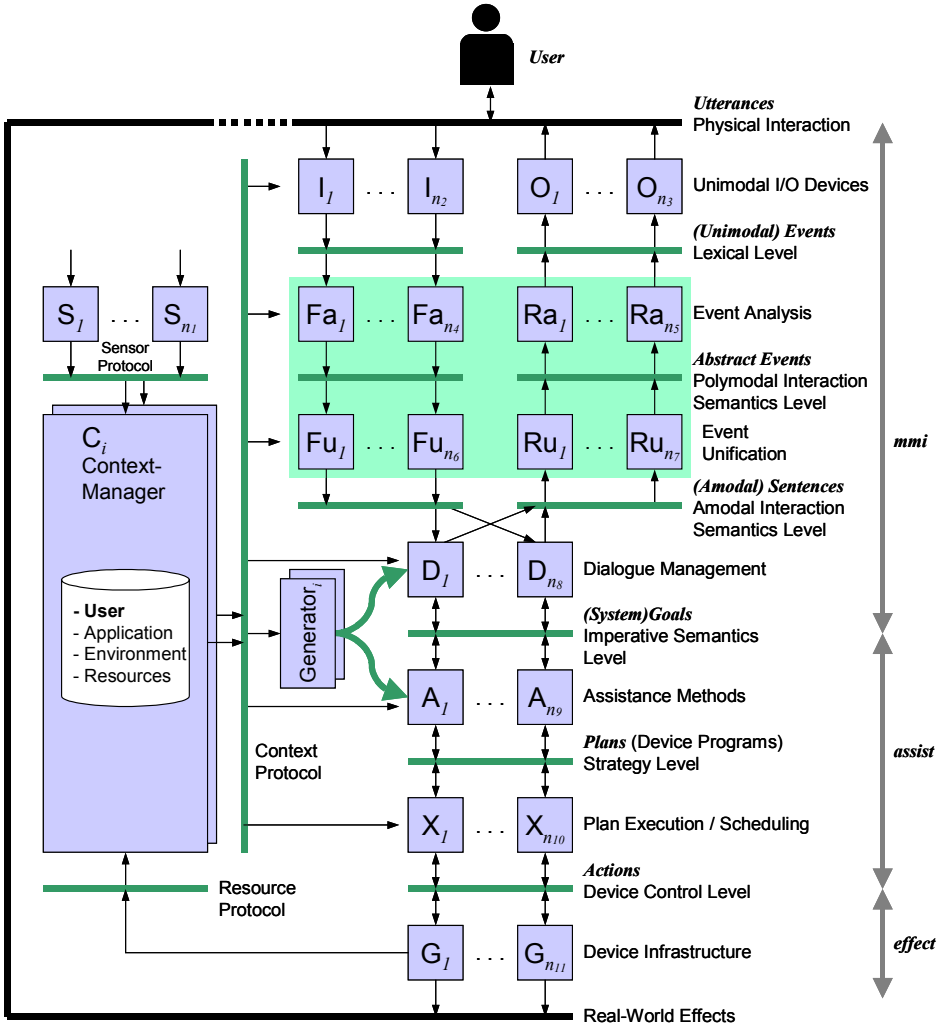

Abbildung 1 Embassi Systemarchitektur.

Die generische Architektur verfolgt den Pipeline-Ansatz, um die Äußerungen des Benutzers auf Ziele auf das Ändern von Umgebungsvariablen abzubilden. Jede Ebene („Level“) der Architektur repräsentiert eine Funktion innerhalb dieser Pipeline, während die Ebenenschnittstellen die verschiedenen Ontologien voneinander separieren. Diese Ontologien (das ist der Satz on Objekten über die auf jeder Ebene diskutiert werden) werden sichtbar auf den Schnittstellen. Diese Schnittstellen formen zusammen das EMBASSI-Protokoll. Jede Ebene besteht aus einer Anzahl an Prozessen (Komponenten, engl. „components“), die in Zusammenarbeit die Funktionalität jeder Ebene bilden. Prozesse können dynamisch hinzugefügt oder entfernt werden, passende Mechanismen der Koordination auf jeder Ebene sind für die Interaktionen zwischen den Einzelprozessen verantwortlich. Offensichtlich gibt es keine zentrale Kooperationsstelle.
Die Entscheidung für dieses aufgeteilte Ebenenmodell in Zusammenwirken mit der Eigenschaft der Dynamik erlaubt es Systeme zu entwerfen, die schrittweise ad-hoc aufgebaut und erweitert können. Im besonderen ist es möglich vollständig kompatible Systeme aus Komponenten von unterschiedlichen Herstellern aufzubauen, wo Komponenten hinzugefügt und entfernt werden können, und das über die gesamte Laufzeit durch den Endbenutzer.
Im besonderen verfügt die EMBASSI Topologie über die MMI-Ebenen (Multimodale Interaktionen) und die Assistentenebene. Die MMI-Ebenen sind verantwortlich für die Abbildung von multimodalen Benutzeräußerungen zu Benutzerzielen, mit denen dann die Strategiekomponenten weiterarbeiten können. Zuerst werden die physikalischen Interaktionen in atomare Interaktionsevents übersetzt (lexikalische Ebene). Dann werden die atomaren Interaktionsevents von den Filterkomponenten weiterverarbeitet. Diese Komponenten sind verantwortlich für die Vereinigung der atomaren Events zu amodalen Sätzen (syntaktische Ebene). Diese Sätze werden dann von den Dialogkomponenten weiterverarbeitet. Diese Komponenten sind verantwortlich diese Sätze in Ziele zu übersetzen.
Die Assistentenebene arbeitet mit den Zielen die von den MMI-Ebenen identifiziert wurden. Hierzu werden die Ziele zu Änderungen der Umgebung gemäß folgender Schritte abgebildet. Die Assistentenkomponenten benutzen die Ziele und entwickeln Strategien um diese Ziele zu erfüllen. Diese Strategien werden Pläne

(plans) genannt. Die Pläne werden an die „execution control“-Komponenten weitergereicht, die dafür verantwortlich für die verteilte Vorbereitung und Ausführung der Pläne sind. Zuletzt werden individuelle Action requests an die Geräte geschickt.

Parallel (aber zur Topologie gehörig) verwaltet ein Context-Manager alle kontextrelevanten Daten, die unter anderen von verschiedenen Sensorkomponenten stammen.

Ziel der generischen Architektur ist die völlige Vermeidung von zentralen Komponenten. Dies unter anderem um Informationsengpässe zu vermeiden, aber auch, um beim Ausfall einer einzigen Komponente die Arbeitsleistung des Gesamtsystems nicht zu gefährden.

Realisierung im EMBASSI-Projekt:

- Die F-Ebene bzw. auch die R-Ebene bestehen aus jeweils nur einer Komponenten, die für die Medienfusion bzw. die Mediendiffusion zuständig ist (PMI und PMO [>Elting2003]).
- Das Dialog-Management wurde von einer Komponente (der Dialog-Manager) übernommen.
- Die Organisation der Komponenten untereinander erfolgte über eine zentrale Routingkomponente, die auf der Agenten-Kommunikationssprache KQML aufsetzte.

1.5.2. Routing in EMBASSI

Im EMBASSI-Projekt verhalten sich alle Komponenten wie KQML-Agenten (siehe Abschnitt über KQML in diesem STAR). Durch diesen Lösungsansatz verschwindet leider die Art der Anordnung der Komponenten wie in der generischen Architektur angedacht war. Die EMBASSI-Komponenten ordnen sich bei KQML quasi in einem Kreis um den Router, der die Nachrichten von einer Komponente zur anderen durchreicht.

Im EMBASSI-Projekt wurden drei verschiedene Router entwickelt, um den unterschiedlichen Projektanforderungen gerecht zu werden. Auf diese soll hier kurz eingegangen werden:

1.5.2.1. KQML-Router in Privathaushalt und Kfz-Systeme:

Da diese beiden Szenarios jeweils auf einer Agentenplattform aufsetzen, wurde hier ein zentraler Router eingesetzt. In OpenEMBASSI sind zwei Versionen dieses Routers verfügbar (jeweils in C und in Java). Eine Besonderheit des Routers, der hier eingesetzt ist, ist seine Fähigkeit, die Inhalte der Nachrichten, welche in XML formuliert sind, gegen eine angegebene DTD zu checken ([>Forkl2002]).

1.5.2.2. A-Term-Router

Für die Anforderungen des Terminalszenarios, wo sowohl auf der EMMA (so heißt das persönliche Bediengerät, vergleichbar mit einem PDA in EMBASSI) und auf den Terminals eine eigenständige Plattform läuft, wurde eine Routingkomponente entwickelt (siehe Abbildung 2), die die Metapher der direkten Adressierung der Agenten überwindet. Agenten einer Plattform können – per Definition – die Namen von Agenten auf einer unbekannten anderen Plattform nicht kennen.

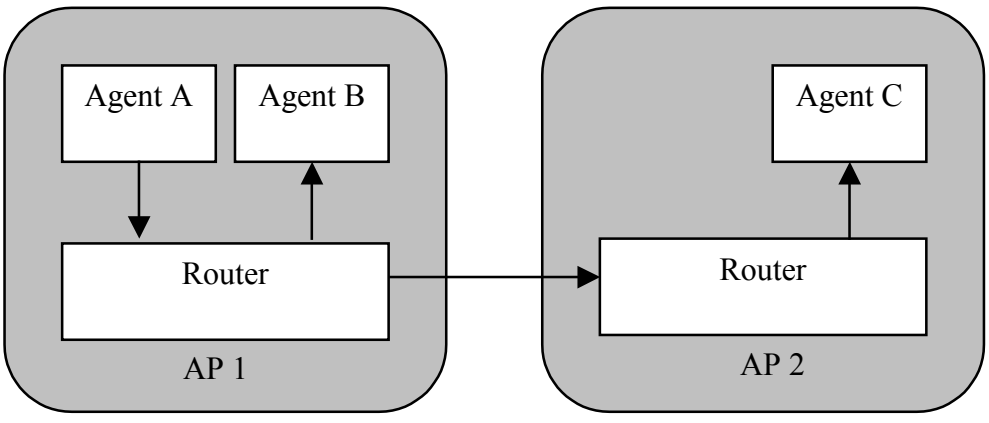

Abbildung 2: A-Term Router.

Diese Lösung basiert schon auf den Grundzügen von Soda-Pop in der nach außen hin anonyme Agenten in Ebenen organisiert sind, die ihrer jeweiligen Funktionalität entsprechen. In den Terminalszenarios wurde jedoch auf konkurrierende Agenten verzichtet, d.h. auf jeder Ebene gibt es jeweils nur einen Agenten, so dass hier vor allem das Prinzip der Anonymisierung verwirklicht wurde. Ebenso findet sich hier ein Mechanismus, um andere Plattformen zu entdecken und sich automatisch zu verbinden. Der Server einer EMMA fungiert hier dann als ein Agent auf dem Server des Terminals.

1.5.2.3. Meta-Router

Der Meta-Router (siehe Abbildung 3) verbindet über das Internet die verschiedenen Einzelagentenplattformen miteinander. Hierzu können die einzelnen Router auch hinter Firewalls geschützt sein. Als Kommunikationsprotokoll wird zwischen den Routern und dem Meta-Router HTTP eingesetzt. Der Meta-Router ist hierbei in der Lage, alle auf den verschiedenen Einzelplattformen verbundenen Agenten zu verwalten, sowie die Nachrichten für die einzelnen Router vorzuhalten.

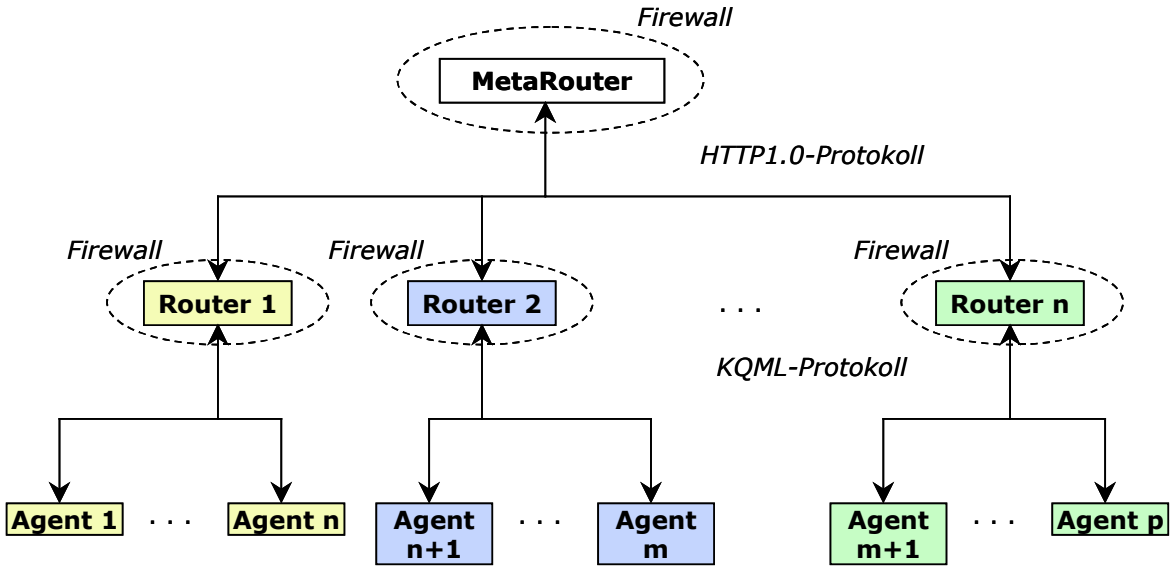

Abbildung 3: Meta-Router in Embassi.

Sendet ein Agent eine Nachricht an einen anderen, wird dieser Adressagent erst auf der eigenen Plattform gesucht (FIPA-Prinzip) und erst im Negativfalle wird die Nachricht dem Meta-Router übergeben, der sie dann dem Router übergibt, der diesen Adressagent verwaltet.

1.5.3. Kommunikation in EMBASSI

Die Kommunikation in EMBASSI unter den verschiedenen Komponenten basierte auf der Anwendung der Agentenkommunikationssprache KQML. Als Beschreibungssprache für die Inhalte wurde XML eingesetzt.

1.5.3.1. KQML

Zur Verwendung der Agentenkommunikationssprache KQML soll an dieser Stelle nicht mehr viel erwähnt werden. Die im EMBASSI-Projekt am häufigsten verwendeten Performatives sind: register (analog zu connect), unregister, tell, ask-one, ask-all und achieve. Ebenso wird sorry und error verwendet. Auf den subscribe und advertise-Mechanismus, den der Context-Manager im Prinzip unterstützt, wurde verzichtet.

1.5.3.2. XML

Die Inhalte (content) von Nachrichten wurden in XML beschrieben. Hierzu tauschen Komponenten, die in der generischen Architektur benachbart liegen (d.h. parallel oder in übergeordneten Ebenen) geeignete XML-Nachrichten aus, die der Ontologie auf der jeweiligen Ebene entsprechen. Eine übergeordnete Ontologie, die alle Sprechakte zusammenfassen würde, existiert in EMBASSI nicht.

1.5.3.3. sem.dtd auf Dialogmanager-Ebene

Die sem.dtd fasst die Inhalte der PMO / PMI, Dialog-Managerebene und der Ebene der Assistenten zusammen. Die sem.dtd beschreibt somit alle Sprechakte, die den Dialog-Manager direkt betreffen. Auf dieser

Ebene der Ontologie (und damit auch der Topologie der generischen EMBASSI-Architektur) fand somit eine Standardisierung statt.

1.5.4. Soda-Pop

Soda-Pop entwickelt die Idee der generischen EMBASSI-Architektur weiter [>Heider2002]. Das SODA-POP Model führt zwei grundlegende Organisationsebenen ein:

- Grobkörnige Selbstorganisation basierend auf der Aufteilung in Datenflüsse
- Feinkörnige Selbstorganisation für funktionell ähnliche Komponenten basierend auf einer Art von „Pattern-Matching“

Ein SODA-POP System besteht aus zwei Typen von Komponenten:

Channels (Kanal): liest eine einzelne Nachricht und bildet sie auf vielfache Nachrichten ab, die ohne Verzögerung zu den Komponenten geschickt werden. Channels besitzen keinen Speicher, können verteilt sein und müssen jede Nachricht akzeptieren. Oder in anderen Wörtern: Channels sorgen für räumliche Verteilung von Events an mehrere Transducer.

Transducer (Wandler)[1]: lesen eine oder mehrere Nachrichten und bilden sie auf eine (oder mehrere) Ausgangsnachrichten ab. Transducer sind nicht verteilt, können Speicher haben und müssen nicht jede Nachricht akzeptieren. Transducer sorgen für die zeitliche Vereinigung von mehreren Events zu einem einzigen Output. Ein Transducer kann mehrere Eingangs- und Ausgangs-Kanäle haben.

Channels akzeptieren und übergeben Nachrichten eine bestimmten Typs, während Transducer Nachrichten von einem Typ zu einem anderen Typ umwandeln (siehe Komponenten in EMBASSI). Ein gesamtes System wird definiert als ein Satz an Channels und ein Satz an Transducer, die mit diesen Channels verbunden sind. Im Vergleich zu EMBASSI sind die EMBASSI-Busse die Channels und die Komponenten / Agenten die Transducer.
Channels werden mittels Channel-Deskriptoren gekennzeichnet. Diese verschlüsseln die Ontologie des Channels, so dass Transducer automatisch mit Channels verbunden werden, dessen Sprache sie verstehen. Die Middleware für multimodale Event Verarbeitung sollte zumindest die beiden Kommunikationsmuster der Events und der Remote Procedure Calls (RPCs) unterstützen. Events werden in der Art des Datenflusses von Transducer zu Transducer durchgereicht. Ein Event erwartet keine Antwort. Aber ein Transducer, welches ein Event verschickt, erwartet dass es im folgenden auch verarbeitet werden kann. Ein RPC folgt der Metapher der RPCs. Ein Transducer, der einen RPC verschickt, erwartet eine Antwort. Im Gegensatz zu den Events bestimmt der initiierende Transducer den Weiterführungsschritt.
In SODA-POP wird somit zwischen Event und Remote Procedure Channels unterschieden (die selbstverständlich über dieselbe Ontologie verfügen können). Jeder Event Channel verfügt gleichzeitig über einen Channel für inverse Remote Procedure Calls. Das ist leicht verständlich. Events folgen der push-Metapher. Es kann jedoch nötig sein, dass ein „Event-Verbraucher“ nach Events fragt. Also von sich aus einen inversen RPC initiiert. Ein Beispiel hierfür ist der Dialog-Manager in EMBASSI, der von der PMI / F – Ebene Events verbraucht und im Gegensatz zu Nachfragen RPC verschickt.

Events und RPCs werden ohne spezielle Angabe von Adressaten gemacht. Es liegt am verantwortlichen Channel, die Nachricht zu zerlegen und an einen Satz von Empfängern weiterzuleiten.

1.5.5. Zusammenfassung / Einschätzung

EMBASSI kann wohl zurecht als Vorläuferprojekt von DynAMITE angesehen werden. EMBASSI führte eine Topologie zur Verarbeitung von multimodalen Interaktionen des Benutzers ein, die dann auf verschiedenen Ebenen der Verarbeitung amodal weiterverarbeitet wird und letztendlich den Geräten ein Benutzerziel zur Ausführung gegeben wird. Hierzu führte EMBASSI Ebenen ein, an denen Komponenten angehängt sind. Die Komponenten dienen sozusagen als „Ontologieumwandler“, z.B. amodale Interaktionsinformationen zu Zielinformationen, während Ebenen Informationen gleicher Ontologie behandeln. In EMBASSI fand diese Topologie sich jedoch in der Implementation und damit den Demonstratoren nicht wieder. Auf Basis der Topologie wurde dann der SodaPop-Ansatz entwickelt, der in Komponenten und Kanäle unterscheidet.

Für DynAMITE bildet der SodaPop-Ansatz die Basis, um das Ziel der Selbstorganisation von Geräten (in SodaPop lassen sich Konfliktlösungsalgorithmen etc. definieren, die anders als in anderen Lösungen (z.B.

[1] Im folgenden werden die Begriffe Komponente, Transducer und Agent als Synonyme behandelt.

OAA)) keine ontologischen Kenntnisse voraussetzen) weiterzuentwickeln. Hierzu bilden die Erkenntnisse aus EMBASSI und die Grundlage von SodaPop eine gute Basis.

Literaturreferenzen zu [Kapitel 1.5]

[>Embassi1999] Die EMBASSI-Projektseite, http://www.embassi.de

[>Kirste2001] Kirste Th, Rapp St: Architecture for Multimodal Interactive Assistant Systems, Vortrag und in Proceedings der Statustagung der Leitprojekte "Mensch-Technik-Interaktion", S. 111-115, Saarbrücken, 26./27.10.2001

[>Elting2003] Elting C, Rapp S, Möhler G, Strube M: Architecture and Implementation of Multimodal Plug and Play. ICMI-PUI `03 Fifth International Conference on Multimodal Interfaces, November 2003, Vancouver B. C., Canada

[>Forkl2002] Forkl Y, Hellenschmidt M: Mastering Agent Communication in EMBASSI on the Basis of a Formal Ontology, ISCA Tutorial and Research Workshop, Multi-Modal Dialogue in Mobile Environments, June 17-21, 2002 Kloster Irsee, Germany

[>Heider2002] Heider Th., Kirste Th.: Architecture considerations for interoperable multi-modal assistant systems, PreProceedings of the 9th International Workshop on Design, Specification, and Verification of Interactive Systems; DSV-IS 2002, 6,2002, Rostock, S.403 – 417

1.6. Ambient Intelligence

„Ambient Intelligence“ ist eine Initiative zur Integration von diversen Forschungs- und Entwicklungsaktivitäten von Phillips Research. Ziel von dieser Initiative ist es die Auswirkungen von *„Ambient Intelligence“* auf das Alltagsleben des Menschen sowie deren Interaktionsverhalten mit der Technik zu untersuchen. Als Testbett und Integrationsplattform dient das ***HomeLab*** (Phillips2003a).

HomeLab ist ein zweistöckiges Familienhaus ausgestattet mit Ambient Intelligence Technologien, das von freiwilligen Drittpersonen bewohnt werden kann. Ein breites Spektrum von in Phillips entwickelten Technologien und Konzepte können im HomeLab integriert werden, um dann im Rahmen einer Gesamtszenario unter realen Lebensbedingungen evaluiert zu werden. Eine Besonderheit von HomeLab ist es, dass es „more a home that is also a Lab“ ist, wodurch dann eine angenährt echte Alltagsumgebung geschaffen wird. Unter realen Bedingungen werden Interaktionsverhalten der (echten) Bewohner mit der (intelligenten) Umgebung, Realisierbarkeit der Technologien sowie das Vertrauen der Benutzer in Ambient Intelligence Anwendungen durch Beobachtung untersucht. HomeLab soll Phillips erlauben praktische, psychologische und soziale Implikationen von Ambient Intelligence zu erforschen, sowie „***to better understand the moments people touch technology***” (Phillips2002a).

Das HomeLab hat folgende Eigenschaften:

- Rs ist ein „balanced environment in which people feel both psychologically and physically at home“
- Hat zwei Stockwerke mit Wohnzimmer, Küche, zwei Schlafzimmer, Badeszimmer und Arbeitszimmer
- Überall Displays und Visualisierungsgeräte (Display-lastig)
- Kameras (34) und Mikrophone verteilt über das ganze Haus, wobei die Kameras und Mikrophone “versteckt” werden
- Ethernet
- 270 GB Speichernetz
- Gigaport-Breitband-Internet
- Video- und HF-Anlage zum Identifizieren und Verfolgen
- Anlagen zur Personenidentifikation
- Sprach- und Gestenerkennung
- Kontrollsystem für die Fernsteuerung von Licht- und Strom
- hat ein Kontrollsystem für die Kameras und A/V Signalverarbeitung
- Hat 2 Überwachungsräume mit 20 Monitoren, von den aus das ganze Haus durch halbdurchsichtige Spiegel und über Videokameras observiert werden kann
- hat Multidisziplinäre Forschungsgruppen, welche die Bewohner beobachten, die Videoausschnitte annotieren und später alles analysieren
- Die Geräte (Fernsehgerät, Lichtschalter, etc.) sind im Hintergrund verschwunden. Die Funktionalitäten (z.B. Film anschauen) sind aber in allen Räumen verfügbar
- Erlaubt natürliche Interaktion
- Kann die Identität und Position von Menschen und Objekten bestimmen
- Hat **„ubiquitous sound and vision“** *(FollowMe mäßig!)*
- Die Umgebung ist mehrbenutzerfähig und kann personalisiert werden
- Die Umgebung kann die **Menschen wiedererkennen** (Gedächtnis!) und von deren Verhalten **lernen**
- HomeLab konzentriert sich sehr auf **Unterhaltungselektronik und Heimszenarien**
- Enge Kooperation mit **MIT Oxygen**, **ITEA AMBIENCE**[2]**, ITEA ROBOCOP**[3] und langzeit Forschungskooperation **AIR&D**[4] zusammen mit INRIA und Thomson Multimedia

1.6.1. Szenario

Bob and Linda have been invited to HomeLab to experience how a future living room might support the planning of their yearly summer holiday. After a tour around HomeLab, which surprisingly well matched the picture of what they would consider an ‘ordinary’ home, they settle at the dinner table and start to browse the

[2] http://www.extra.research.philips.com/euprojects/ambience/
[3] http://www.extra.research.philips.com/euprojects/robocop/
[4] http://www.air-d.org/

holiday brochures lying around. After a while, they begin to realize that the room seems to listen to their conversations, as pictures and movie clips started to appear on the wall that matches the actual holiday destinations they are discussing. Things get even more interesting when the colour of the lighting of the environment adjusts to the Tuscany pictures they are looking at. The experience is complete when they notice the soft Italian music that was playing in the background. After Bob and Linda selected their dream holiday, they join the research team to discuss the way they experienced the Ambient Intelligence room. They both tell the team they are impressed by the smartness of the room and the smooth way interesting media were displayed in the room only when it seemed appropriate and meaningful. At the end of their visit to HomeLab, Bob and Linda are given a tour backstage.

This is where they learn about the Wizardof-Oz type of experiment in which they had been involved. It turned out that the intelligent behaviour of the room was not generated by an electronic system but by a group of researchers who had been watching, listening and interpreting their activities to select, present and direct the actual media interaction experience in which Bob and Linda had been immersed. After some initial disappointment that the system was not yet for sale, their enthousiasm returns: Bob and Linda would love to live in the Ambient Intelligence environment they had been experiencing at HomeLab (Phillips2002a).

1.6.2. Anwendungen in HomeLab

Seinen Bewohnern bietet HomeLab einige Anwendungen, die nicht in erster Linie für die Realisierbarkeitsstudie selbst oder Haussteuerung (Positionierungssystem, Videoüberwachungssysteme, Multimodale Interaktion, Lichtsteuerung, Power Management, etc.) benötigt werden.

Im Folgenden einige Beispiele der neuen Technologien, die gegenwärtig im HomeLab erforscht werden (Phillips2002b):

- POGO – ein interaktives Spiel für Kinder, mit dem sie ihre Vorstellungskraft durch die Verschmelzung von Fantasie und Realität auf ein Umfeld beziehen können, in dem Kinder spielen und gemeinsam Geschichten aufführen können.
- Connected Planet – Szenarios und drahtlose Vernetzungslösungen auf der Basis eines Netzes von IP-Schnittstellen, die in echte kommunikative Landschaften eingebettet sind und einen Wissensaustausch in realen Gemeinschaften ermöglichen.
- WWICE – eine interaktive Benutzeroberfläche namens „Window on the World of Information, Communication and Entertainment“, die verschiedene Vorrichtungen im Haus in ein einziges System zusammenfasst, so dass typische digitale Vorgänge wie das Aufzeichnen von Voicemail, das Anschauen eines Videobandes oder das Anhören von Musik von jedem Zimmer im Haus möglich ist.
- Health Coach – ein auf LCD-Technologie basierender „intelligenter Spiegel“, der ein Umfeld für die komplette Körperpflege bereitstellt und Benutzer in verschiedenen Bereichen der Gesundheits- und Körperpflege, wie zum Beispiel Gewicht, Zahnpflege etc., überwachen und anleiten kann.

1.6.3. Eingesetzte Middleware

Das WWICE (Window on the World of Information, Communication and Entertainment) von Phillips bildet die Basis für ihre verteilte **In-Home Digital Networks (IHDN)** Anwendungen, die auch in HomeLab eingesetzt werden.

Ziel von WWICE ist es, Middleware und Infrastruktur für intuitiv zugreifbare und bedienbare A/V Geräte und Anwendungen bereitzustellen. Folgende Eigenschaften besitzt WWICE (Baldus2000):

- **Gradually increasing software and hardware:** durch die offene Schnittstellen können dynamisch neue Geräte und Funktionalitäten dazu kommen: *„the user can gradually increase the functionality of his IHDN and the supported hardware“*
- **Ubiquitous Media Access:** der Zugriff auf Multimedia Content ist unabhängig von „storage location“, benutztes Gerät und der Ort des Benutzers möglich
- **Network and device transparency:** The user is not aware of any details of network topology, device allocation and availability or necessary format conversions. As long as suitable devices and the required bandwidth suitable for a task are available somewhere they will be found and allocated by the system.
- **Location independence:** The user can move around the house and take with him anything he is currently using the system for (as long as the technical preconditions are fulfilled).

- **Uniform way of handling:** The user is presented a coherent user interface on all environments. While different capabilities of these environments might imply a different look, the UI concepts have to remain the same to flatten the learning curve for the user.

Das Schlüsselkonzept von WWICE ist die Einführung von „Activity“, „Content“ und „Place“ Konzepten (s. Abbildung 4). Hierbei geht es darum eine Aufgabe oder Aktivität von den für dessen Erfüllung notwendigen Content, Methoden und Geräten zu trennen. Dem Benutzer soll es transparent sein, wie eine Aktivität und mittels welchen Ressourcen durchgeführt wird. Vielmehr soll der Benutzer sich Überlegungen, **was** er **wo** machen möchte (Baldus2000).

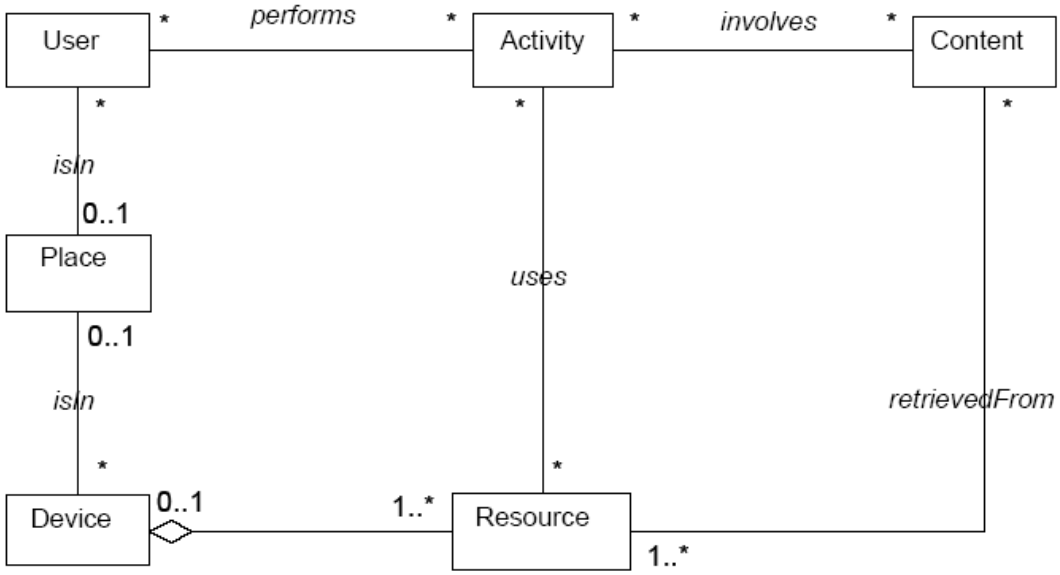

Abbildung 4 WWICE Domain Model

*„An **Activity** represents a specific kind of 'task' a user wants to execute, for example watching a TV. Each Activity contains information about the devices that are necessary for its task, methods for moving or copying it, and a way to get hold of a list of possible Contents.*

***Content** objects 'parameterize' Activities: While Activities specify the operations needed to execute a general task in an abstract manner, a Content object is needed to 'instantiate' this abstract description into the specific performance intended by the user, i.e. different TV channels. Content objects are actually provided by Content Managers. They have the task of hiding the devices in the network, which provide the content. For this they retrieve content information from all eligible devices (e.g. all tuners or all VCRs) and merge them into one list.*

*In order to deal with locations, the concept of **Place** is introduced. A Place typically denotes a room but could also mean a part of a room. If the user wants to know what is going on in his system, he is shown Activities associated with Places. Therefore also any manipulation of Activities takes place on the basis of Places, not devices."*

Die Geräte in einem Raum sind durch IEEE1394 vernetzt, wobei die Rechenleistung von Set Top Boxes oder Digital TVs erbracht wird.

Die Architektur von WWICE ist z.B. in www.research.philips.com/ Assets/Downloadablefile/w102-1129.pdf dargestellt. Es ist eine multi-layer Architektur, die es dem Benutzer erlaubt, auf verschiedene A/V Endgeräte einheitlich zuzugreifen:

*The **Platform layer** introduces abstractions for the used devices. The classes Unit and Subunit both provide standard methods for allocation, initialization and connection. Whereas a Unit is a physical device including connector, location etc., a Subunit is a functional component within a device. For each new hardware to be integrated into the system a subclass of these has to be included.*

*The **Middleware layer** provides all functionality that is necessary to allow easy realization of applications which follow the user interface concepts. The Registry contains static information about the system and its environment such as references to the other services and the Units/Subunits available in the system. The Resource Managers control the allocation of local and remote Subunits and network bandwidth. The Graph Mapper maps abstract representations (i.e. graphs) of device networks needed by Activities onto the devices available in the system. This mapping process allows Activities to concentrate on the service they provide to the user instead of on how to find, allocate and connect devices. The Factories are used to create remote instances of a class and the BootManagers control the start-up phase of the system. None of these services needs extensions when adding new user-functionality or hardware.*

*In the **Application layer** the implementation of all user-functionality is located. For each new functionality mainly three classes have to be inserted: Activity, View and Activity-Icon. An Activity has to provide by default methods for initializing, starting, stopping, moving, copying and extending itself.*

*A View provides a GUI to its related Activity. The ActivityIcon (not shown in the architecture) provides another View of the Activity. It displays an icon and a menu for the Activity and thereby allows more specific manipulations of the Activity using the UI-applications MenuBar and Housemap (**Fehler! Verweisquelle konnte nicht gefunden werden.**).*

*The **UI layer** is again a complete layer. It consists of the UIMS and several UI-'applications'. The UIMS provides basic GUIelements and control over overlay and PiP-functionality. The UI-applications can be considered special management applications that – in contrast to the Activities – are not concerned with any Content but with the Activities themselves. The Finder lists kinds of Activities and related Content and allows the user to start them. The MenuBar gives an overview of the local Activities and provides control over their appearance. Finally, the Housemap shows the Activities in the whole house grouped according to the defined Places. This allows movement of Activities and also some remote control.*

Eine Schlüsselkomponente ist der ***Graph Mapper***, welche die Elemente einer als Graph beschriebenen „Activity" auf physische Units und Subunits (z.B. Fernsehgerät, Tuner) und Netzwerkpfade mappt. Hierzu muss der GraphMapper Geräte finden, die zu der in der *Activity* beschriebenen *Subunits* passen und grade verfügbar sind.

"For each type of Activity, a Graph is defined. Its nodes (vertices) represent entities that control and process data, while the edges represent data connections between these entities. An Activity Graph is directed and has one or more source nodes (ones that only have edges pointing away from them) and one or more sink nodes (ones that only have edges pointing towards them). For example, television watching requires a Graph consisting of a tuner, a demultiplexer, a decoder, and a screen. These Subunits have to be connected to each other, with connections leading from the tuner (source node) to the demultiplexer, from the demultiplexer to the decoder and from the decoder to the screen (sink node). The connection between Subunits may be either within a Unit or via the network.

In order to find the requested Subunits, the Graph Mapper contacts the registry to find out which Subunits exist in the system. It then contacts the overall Resource Manager to know whether these Subunits are available. The Graph Mapper also contacts the network to get a free channel, and asks the Units to connect the Subunits to each other or to the network. Possible restrictions when choosing Subunits will be discussed in the following paragraph."

Der Graph Mapper erlaubt einige Auswahlkriterien, um für einen Subunit ein Gerät zu finden:

- ***Location**: Some Subunits (e.g. a tuner) can be equally used regardless of their location, while those that interact directly with the user (screen, camera, microphone, speakers) have to be in the desired Place.*
- ***Content**: Not all instances of a Subunit are able to provide the same content, e.g. a satellite tuner offers different channels than a cable tuner.*
- ***Quality of Service**: Different instances of a Subunit may provide different levels of quality, e.g. a camera offers a certain resolution and frame rate.*

- ***Supported Stream Formats****: Audio and video streams can be provided in different formats and obviously combining different devices only makes sense if they support the same format or if a transcoder Subunit can automatically be included.*

- ***Network Capacity:*** *If there is not enough bandwidth available in the network, either a different device has to be chosen (e.g. one that is locally connected to the other within a Unit), or a different stream format with less bandwidth demand has to be used.*

Um eine ***Activity Relocation*** durchzuführen, muss lediglich der Graph erweitert werden. Möchte man z.B. einen Film in einem anderen Raum verfolgen (Follow Me), so muss der Graph lediglich um neuen Subunits, also decoder und screen erweitert werden: *"For moving an Activity, the sink node (which is usually defined as one specific device, e.g. the living room screen) has to be replaced by another one (e.g. the kitchen screen), which has to be connected to the source and those intermediate nodes which can remain the same."*

1.6.3.1. Zusammenfassung

WWICE fokussiert auf IHDN A/V Anwendungen. Es erlaubt Geräte verschiedener Hersteller miteinander zu kombinieren sowie Geräte- Orts- und Netzwerkunabhängige Anwendungen zu definieren. WWICE benötigt keinen zentralen Server und erlaubt dem Programmierer sehr leicht seine IHDN-Anwendungen um neue Geräte und Funktionalitäten zu erweitern. Hierzu muss der Programmierer lediglich die Beschreibung der Activity erweitern, welcher nur den benötigten Graphen und die im Falle von FollowMe-Operationen zu verlagernde Teile (Subunits) definiert.

Die Ressourcen und Geräte werden statisch im Registry eingetragen.
WWICE unterstützt jedoch keine (zielbasierte Interaktion und auch keine) dynamische Ensembles und Strategien um Ziele mit vorhandener Infrastruktur umzusetzen, wie es im EMBASSI vorgesehen ist.

1.6.4. Relevanz für DynAMITE

HomeLab ist eine Initiative zu Integration von diversen Forschungs- und Entwicklungsaktivitäten von Phillips Research. Ziel von dieser Initiative ist es die Auswirkungen von *„Ambient Intelligence"* auf das Alltagsleben des Menschen sowie deren Interaktionsverhalten mit der Technik zu untersuchen. HomeLab beschäftigt sich somit an erster Instanz mit Feasibility Studies und Interaktionsfragen. HomeLab beschäftigt sich stark mit *„**Haushalt-Szenarien**".*

- Das aus DynAMITE-Sicht am meisten Interessante Szenario ist das „Window on the World of Information, Communication and Entertainment" (WWICE) , die die verschiedenen Vorrichtungen im Haus in ein einziges System zusammenfasst, so dass typische digitale Vorgänge wie das Aufzeichnen von Voicemail, das Anschauen eines Videobandes oder das Anhören von Musik von jedem Zimmer im Haus möglich ist.
- Die raumübergreifende Ausführung von Aktivitäten, die auch durch mittels Trennung I/O Geräten und den Einsatz von EZLGM in ***EasyLiving*** möglich war, basiert hier auf das Konzept von „***Activity-Content-Place***" und das *In-Home Digital Networks* ***(IHDN)***. Bei diesem Konzept sagt der Benutzer lediglich „was" er „wo" haben möchte und und nicht „wie" und mit „welchen" Ressourcen die Aufgabe durchgeführt werden soll. Solch eine Aufschlüsselung von Aufgaben (wurde weiter oben näher beschrieben) erlaubt eine elegante Implementierung von ***„FollowMe"*** Applikationen sowie eine gewisse *ubiquitous data access* und *device and location transparency*.
- Das IHDN bzw. das Activity-Modell von HomeLab erlaubt zwar die Streuung von Aufgabenausführung über Räumen hinaus und die dynamische Allokation von Geräte und Content. Das Modell unterstützt jedoch keine zielbasierte Interaktion und auch keine dynamische Ensembles im Sinne von EMBASSI und DynAMITE.
- Ähnlich zu EasyLiving sind auch HomeLab Anwendungen (, die auf IHDN und Activity Modell basieren) dynamisch erweiterbar, da die dynamische Hinzunahme von neuen Eingabe-Ausgabegeräten unterstützt wird. Allerdings sind die funktionalen Rollen dieser Geräte fest vorgegeben. Es können also nur Geräte von bestimmten Typen hinzukommen, welche das System bereits kennt und für die Realisierung der Szenario benötigt (z.B. Display, Monitor, Beamer für *Output devices*). Im Gegensatz zu EasyLiving können bei HomeLab auch ***dynamisch neue Vorgänge***

(„Tasks“ wie „Aufzeichnen von TV-Sendungen) definiert und modifiziert werden. Hietz dient der Schlüsselkomponente Graph Mapper.

- HomeLab kann auch (per Definition) als eine Context-bound Anwendung gesehen werden. HomeLab zielt hauptsächlich auf Home Szenarien ab.
- HomeLab unterstützt verschiedenartige Aktivitäten, die man unter „***Environment Control***“, ***„Media Management“, „Communication and Cooperation“*** klassifizieren könnte.
- HomeLab’s Interaktionsparadigma ist ***Augmentation***

Literaturreferenzen zu [Kapitel 1.6]

[Phillips2003a] N.N. (2003). “Ambient Intelligence - changing lives for the better”. Phillips Research, Eindhoven, Netherlands, http://www.research.philips.com/Assets/Downloadablefile/343.04_BLZ04-07-2722.pdf, October 2003.

[Phillips2002a] Berry Eggen et al (2002). „Ambient Intelligence in HomeLab”. Royal Phillips Electronics, ISBN 90-74445-55-1 2002, http://www.philips.com/Assets/Downloadablefile/ambientintelligence-1503.pdf

[Phillips2002b] „Philips HomeLab – Hintergrundinformationen”. Philips GmbH, Unternehmenskommunikation, Journalisten-Hotline, 6.1.2002, Hamburg. http://medienservice.philips.de/apps/n_dir/e1231501.nsf/searchalle/F160784771B1C48041256CCC004899C7?opendocument

[Baldus2000] H. Baldus, M. Baumeister, H. Eggenhuissen, A. Montvay, W. Stut (2000). „WWICE: an architecture for in-home digital networks”. Proceedings of SPIE, Vol. 3969, Januar 2002.

1.7. Pervasive Computing

Pervasive computing ist personalisierter Informationszugriff, überall und zu jeder Zeit [>IBM]. Der Grundgedanke hinter diesem Begriff ist es, dass normale Arbeitsplatzrechner zunehmend durch die Steigerung der Rechenleistung und die gleichzeitige Kostenreduzierung und Minimalisierung von kleinen, intelligenten und spezialisierten Geräten (z.B. PDA oder intelligentes Mobiltelefon) verdrängt werden. Diese kleinen Geräte und die drahtlosen Netzwerktechnologien befreien Informationsbenutzer von ihrem Schreibtisch. Ausgestattet mit verschiedenen kleinen Geräten können die Benutzer von ihrem Zuhause, unterwegs oder ihrem Arbeitsplatz aus auf die Informationen zugreifen. Dennoch ist pervasive computing keinesfalls das einfache Zusammenstecken von verschiedenen Geräten. Die Integration von Geräten und Anwendungen, die nicht für ihre Zusammenarbeit entwickelt worden sind, erfordert verschiedene neue Technologien. Im Folgenden wird die IBM Software zum pervasive computing, das *WebSphere Everyplace Suite* [>IBM2000], kurz vorgestellt.

WebSphere Everyplace Suite stellt eine Infrastruktur dar, die es ermöglicht, von verschiedenen Geräten aus über das Netzwerk auf Daten und Anwendungen zuzugreifen. WebSphere Everyplace Suite ist eine Brücke zwischen den zugreifenden Endgeräten und den Daten und Anwendungen auf der anderen Seite. Die Infrastruktur umfasst sechs Funktionalitäten: Konnektivität, Inhaltsmanagement, Sicherheit, Optimierung, Benutzermanagement, Gerätemanagement sowie einige Kerndienste, die der Installation, Konfiguration und Administration des Systems dienen. Diese Funktionalitäten werden jeweils durch von verschiedenen existierenden IBM-Produkten implementiert.

- Konnektivität: *Everyplace Wireless Gateway* stellt sichere, sowohl drahtgebundene als auch drahtlose Verbindungen zwischen einem unternehmensinternen Netzwerk und beliebigen externen Netzwerken zur Verfügung. Eine Konvertierung zwischen den Netzwerkprotokollen ermöglicht eine einheitliche Programmierschnittstelle für die Benutzer. Beispielsweise wird eine TCP/IP Programmierschnittstelle zur Verfügung gestellt.

- Inhaltsmanagement: Da verschiedene Geräte sich in ihrer Größe und ihre Darstellungsfähigkeiten unterscheiden, müssen Inhalte dementsprechend angepasst werden, damit sie auf diesen Geräten angezeigt werden können. *WebSphere Transcoding Publisher* transformiert automatisch den Inhalt zwischen verschiedenen Formaten, so dass das Speichern eines gleichen Inhalts in mehreren Versionen überflüssig wird. Ein Beispiel ist das Transformieren eines HTML-Textes, der ursprünglich für die Darstellung auf einem Arbeitsplatzrechner geschrieben wurde, in einen WML-Text für das Anzeigen auf einem WAP-fähigen Mobiltelefon.

  Pervasive-Geräte sind meistens nur temporär mit einem Server verbunden, eine permanent verfügbare Netzverbindung kann nicht vorausgesetzt werden kann. *MQSeries Everyplace* ermöglicht asynchrone Kommunikation, d.h. das Verschicken und das Empfangen von Nachrichten wird zeitlich entkoppelt. Ein Benutzer verschickt die Daten nur einmal und kann davon ausgehen, dass sie genau einmal an den Empfänger weitergeleitet werden.

  Mit Hilfe von *Everyplace Synchronization Manager* können Benutzer offline an ihrem Kleingerät arbeiten und von Zeit zu Zeit die Ergebnisse mit einer Serverdatenbank synchronisieren, so bald eine Verbindung zum Server verfügbar ist.

- Sicherheit: *Everyplace Authentication Server* ist für die Authentifikation der Benutzer und Geräte zuständig. Er bietet einen einheitlichen, geräteunabhängigen Authentifikationsmechanismus an und leitet die Authentifikationsdaten an die Unternehmensserver. *Everyplace Wireless Gateway* unterstützt das Virtual Private Network (VPN), so dass das sichere unternehmensinterne Intranet über öffentliche Netzwerke sicher benutzbar wird.

- Optimierung: Die Optimierungskomponenten haben das Ziel, durch die lokale Speicherung (engl. Caching) und Komprimierung von Daten sowohl die Kosten der Netzwerkbandbreite als auch die Antwortzeit beim Laden von Daten von einem Datenserver zu reduzieren. Zusätzlich wird mittels ständiger Überwachung eine Lastbalancierung der Server erzielt.

- Benutzer- und Gerätemanagement: Die Dienste des *Tivoli Personalized Services Manager* beinhalten die Personalisierung der Inhalte, die Eintragung der Dienstbenutzer, die Kundenbetreuung, die automatische Generierung von Berichten, die Schnittstelle zu externen Buchungssystemen, die

Distribution der Software und Daten an die Geräte sowie ihre spätere Aktualisierung und die Bereitstellung des Systemstatus.

Zu den ambitionierten Anwendungsbereichen der WebSphere Everplace Suite zählen Telekommunikation, Finanzdienste, Einzelhandel, Gesundheitswesen und Tourismus, in denen mobile Zugriffe auf Unternehmensdaten und –Anwendungen von essentieller Bedeutung sind.

Literaturreferenzen zu [Kapitel 1.7]

[>IBM] IBM Pervasive computing, http://www-306.ibm.com/software/pervasive/index.shtml.

[>IBM2000] Juan R. Rodriguez, Richard Appleby, Bernt Bisgaard, Hao Wang, Adrienne McGrory, Abdulamir Mryhij, Amy Patton, Muhammed Omarjee, An Introduction to IBM WebSphere Everyplace Suite Version 1.1 Accessing Web and Enterprise Applications, http://www.redbooks.ibm.com, 2000.

1.8. TeCO / ETHZ

Das Telecooperation Office (TeCO) [>TECO1993] wurde 1993 an der Universität Karlsruhe gegründet mit dem Ziel der Forschung und Entwicklung im Bereich der Telematik in enger Zusammenarbeit mit der Industrie. Der Kompetenzschwerpunkt ist das Themenumfeld des *Ubiquitous Computing*. Hauptrichtungen der aktuellen Forschung sind Digital Artefact Computing, Netzwerke, Handheld Computing, Mensch-Technik-Interaktion und Kontextbewusstsein. TeCO ist eine Gruppe des Instituts für Telematik, grenzt sich davon aber durch Nähe zur Industrie und ihr Finanzierungskonzept ab.

Die Verteilte Systeme Gruppe [>ETHZ1999] der ETH Zürich wurde 1999 gegründet und von Prof. Friedemann Mattern geleitet. Neben allgemeinen Projekten die in den klassischen Bereichen der verteilten Systeme angesiedelt sind, widmet sich die Gruppe um Prof. Mattern besonders dem Bereich des „ubiquitous computing“. Das Hauptziel ist hierbei Rechenkapazität in Alltagsdinge zu integrieren, und „clevere Dinge“ (smart things) zu erschaffen. Die Vision des „ubiquitous computing“ beruht auf der Annahme, dass Prozessoren und Sensoren bald so klein und kostengünstig sein werden, dass sie in nahezu alles eingebettet werden können. Um die Kommunikation zwischen diesen kleinen Geräten zu ermöglichen werden neuartige Informationsinfrastrukturen benötigt. Diese Infrastrukturen müssen dann mit hochdynamischen Umgebungen zurechtkommen, und unter anderem Ortsinformationen, Kontextinformationen und verlässliche Dienstbereitstellung ermöglichen. Diesen genannten Herausforderungen hat sich die Verteilte Systeme Gruppe verschrieben.

Beide Gruppen arbeiten an einer Vielzahl von Projekten in den oben erwähnten Feldern, wovon im folgenden „Media Cup“ und „Smart-Its“ näher vorgestellt werden.

1.8.1. MediaCup

Die MediaCup [>CUP1999] ist eine mit Rechnertechnologie ausgestattete Kaffeetasse. Eine in den Tassenboden eingelassene Elektronik erlaubt es, die verschiedenen Zustände einer Tasse (etwa ob jemand trinkt) zu ermitteln und drahtlos zu übertragen. Damit können Kontextinformationen einer Umgebung ermittelt und kommuniziert werden. Die MediaCup ist als Beispiel und Forschungsgegenstand entwickelt worden. Sie zeigt, wie Objekte des Alltags mit Computer- und Kommunikationstechnologie ausgestattet werden können. Der MediaCup-Demonstrator, der aus mehreren Tassen sowie weiteren Objekten besteht, erlaubt es, Erfahrungen im Umgang mit solchen neuartigen Formen des Computers und Erfahrung im Aufbau solcher Computersysteme zu sammeln. Dieser Demonstrator ist inzwischen seit September 1999 in dauerhaftem Alltagseinsatz.

Der Boden der MediaCup besteht in einem aus Gummi gefertigten abnehmbaren Überzieher. In diesen Boden ist die Elektronik der Tasse eingelassen. Wenn die Tasse gespült werden soll, kann der Boden abgenommen werden, um die Elektronik nicht zu beschädigen. Die Elektronik der Tasse wird kabellos mit Energie versorgt. Dazu wurde eine spezielle, ebenfalls elektronisierte Untertasse entwickelt. Die Energie wird von der Tasse in speziellen Akkus zwischengespeichert. Ein 15 Minuten langer Aufladevorgang kann die Tasse etwa 10 Stunden mit Energie versorgen.

Die in der Tasse eingelassene Elektronik erkennt den Bewegungszustand der Tasse (zum Beispiel, ob jemand aus der Tasse trinkt) sowie die Temperatur. Diese Informationen werden von der Tasse in den Raum kommuniziert. Objekte, die sich ebenfalls in der Umgebung befinden (zum Beispiel Kaffeemaschinen oder

einem Web-Server) können diese Informationen empfangen und verwerten, um den Benutzer zu unterstützten und Vorgänge zu automatisieren.

1.8.2. Smart-Its

Das Smart-Its Projekt ist eine gemeinsame Unternehmung von Forschungsinstituten in Finnland, Deutschland, Schweden, der Schweiz und England. Es ist Teil der „Disappearing Computer“ Initiative der Europäischen Union. Die Grundidee der Smart-Its ist es, Dinge des alltäglichen Lebens um Berechnungsfähigkeiten und Sensorik auszustatten, um diesen gegenseitige dynamische digitale Bindungen zu ermöglichen. Der Name Smart-Its lehnt sich an die gängigen Post-Its an. Genau wie diese Informationen durch Anheften an andere Objekte hinzufügen sollen Smart-Its andere Objekte um Fähigkeiten der Sensorik, Wahrnehmung und Kommunikation ergänzen. Die entwickelten Smart-Its dienen der Erforschung der sich entwickelnden Technologie und der sich ergebenden Anwendungsfelder.

Die Vision von TecO ist das bereitstellen einer generischen Plattform (bestehend aus Hard- und Software) um alltägliche Dinge mit kleinen Computern anzureichern. TecO entwickelt eine „Rapid Prototyping“ Umgebung, die es in ca. einer Stunde erlauben soll, ein Objekt mit einem Smart-It auszustatten und in Betrieb zu nehmen. TecO hat ca. 160 Sensoren und ca. 140 Sensor boards entwickelt.

Literaturreferenzen zu [Kapitel 1.8]

[>CUP1999] MediaCup, http://mediacup.teco.edu

[>ETHZ1999] Verteilte Systeme Gruppe, ETH Zürich, http://www.inf.ethz.ch/vs/

[>ITS2003] Smart-Its Project, http://smart-its.teco.edu/

[>TECO1993] Telecooperation Office (TecO), http://www.teco.edu

1.9. Northwestern University / Intelligent Classroom

Intelligent Classroom der Northwestern University (Abteilung Intelligent Information Laboratory) möchte Professoren, Geschäftsleute oder allgemein Menschen, die eine Präsentation zu halten haben unterstützen. Das Projekt unterstellt dabei, dass es eine sehr entmutigende Aufgabe sein kann, an einem fremden Ort eine Präsentation halten zu müssen (aus technischer Ausführungssicht).

Nötig sei hierzu eine Umgebung, die dazu in der Lage ist so viele Arten an Medien zu behandeln wie möglich und soll dem Sprecher so viel Arbeit wie möglich abnehmen. Diese Thematik adressiert besonders das Themenfeld „Cooperative Behavior between Computers“. Intelligent Classroom meint nicht direkt einen Raum, sondern eine intelligente Umgebung.
Intelligent Classroom benutzt Videokameras und Mikrophone um, die Bewegungen, Gesten und Wörter des Sprechers wahrzunehmen. Das Intelligent Classroom übernimmt dann die Kontrolle der Beleuchtung, die Kontrolle der Folien und das Abspielen von Videos etc. Es arbeitet quasi mit dem Sprecher Hand-in-Hand, indem der Sprecher ausdrückt, was er zu tun wünscht und die Umgebung die Ausführung übernimmt.
Das Intelligent Classroom verfügt über Videorekorder, Projektoren, Fernseher und Lichtsteuerungen. Die einzelnen Aufgaben, die das Intelligent Classroom bisher übernehmen kann sind [Flachsbart2000]:

- Einen Film über die gesamte Vorlesung produzieren
- Eine PowerPoint-Präsentation ablaufen und steuern lassen
- Eine Videorekorder zum Abspielen von Filmen steuern

Besondere Arbeitsschwerpunkte im Projekt Intelligent Classroom liegen auf der Ausführung von Plänen und der Unterstützung von Multi-Modalität.

1.9.1. Kurzer Überblick

Der Intelligent Classroom weiß über folgende Dinge innerhalb seiner Umgebung Bescheid:

- die persönlichen Präsentationsstrategien des Sprechers und
- die derzeitigen physikalischen Ereignisse, wie Ortsveränderungen, bestimmte Zeigegesten und Schlüsselwörter, die einen bestimmten Zustandsübergang im der Präsentation erfordern

Daraus ist der Classroom in der Lage (Kenntnis der Präsentationsstrategien und die damit verbundenen Interaktionen des Sprechers) die aktuellen Präsentationsziele zu inferieren und die Umgebung daraufhin gezielt zu manipulieren, z.B. das Heranzoomen der Kamera and die Stelle, auf die der Zeigefinger zeigt. Eine explizite Interaktion des Sprechers ist nicht mehr erforderlich. Das System „weiß“, dass Zeigen auf etwas gleichbedeutend ist mit dem Wunsch des Heranführens der Kamera.

1.9.2. Repräsentation im Intelligent Classroom

Um von der Metapher der Eingabe von Kommandos und der simplen Kommandoausführung durch den Computer zu der Metapher „the person physically does something“ und der Computer „senses the person´s actions, figures out the person´s underlying interntions, figures out what to do about it and then does it“ zu kommen, wird ein Modell der Umgebung und der möglichen Interaktionen (und der möglichen Schlüsse daraus) entwickelt [Franklin1998_1]. Hierbei ist folgendes wichtig:

- Der Kontext (gleiche Gesten etc. können in verschiedenen Kontexten unterschiedliches meinen)
- Die möglichen Ziele und der Schluss auf mögliche Aktionen des Systems (plan recognition)
- Und die passenden Strategien um die Ziele umzusetzen (plan execution)

1.9.2.1. Lokalisierung und Tracking des Benutzers

Die Lokalisierung und das Tracking des Benutzers wird im Intelligent Classroom mit einer stationären Kamera gemacht. Dabei wird der Benutzer mittels der Technik der „Static Background Subtraction“ (dabei muss das „Bild“ ohne Sprecher bekannt sein) lokalisiert und seine Gesten interpretiert.

1.9.3. Planausführungen im Intelligent Classroom

Zur Interpretation des aktuellen Geschehens und die Schlüsse daraus wird im Intelligent Classroom der Ansatz der deklarativen Präsentation gewählt (im Gegensatz zum prozeduralen Ansatz, was unübersichtliche Entscheidungsbäume und damit die Erfordernisse des häufigen Re-engineering mit sich bringen würde).
Der deklarative Ansatz (Expertensysteme, Produktionssysteme) eigne sich zum Herausfinden von Ambiguitäten und lasse sich leicht durch neue Regeln erweitern.
Deklarative Repräsentation werden dazu benutzt, die von den Gestenerkennern und Mikrophonen aufgenommenen Aktionen des Sprechers zu interpretieren und (unter Einbeziehung des Kontextes) Schlüsse auf sein zukünftiges Handeln zu ziehen (z.B. wenn der Sprecher in Richtung Tafel geht und nach unten greift (Richtung Kreide) wird er jetzt gleich was an die Tafel schreiben.

Im Intelligent Classroom findet somit eine dreifache Trennung an Daten statt, an denen per Planungskomponenten eingegriffen werden kann.

- Aufnahme von Gestik, Video etc. –Daten (z.B. nach Kreide greifen)
- Interpretation auf zukünftiges Handeln des Sprechers bzw. seine Intentionen (z.B. will an die Tafel schreiben)
- Planausführung, um die Intention zu unterstützen (bessere Ausleuchtung der Tafel)

D.h. aber auch, dass die Entwicklung von intelligenten Komponenten an zwei Stellen der Ausführung angreifen kann: an der Interpretation der Sprecheraktionen und an der Ausführung der Sprecherintentionen. Hierdurch können gezielt Mehrdeutigkeiten behandelt werden, z.B. können zwei völlig unterschiedliche Handlungen des Sprechers dennoch dieselbe Intention bedeuten und damit auch dieselben Ausführungen des Intelligent Classroom bedingen. Durch die Interpretationstrennung (wo oben beschrieben) ist diese Datenbehandlung leicht möglich.
Dieser Ansatz ist grundsätzlich vergleichbar mit dem Ansatz von EMBASSI: Die Eingabekomponenten nehmen die Benutzerintentionen auf und zusammen mit dem Dialog-Management wird das Ziel des Benutzers erkannt (plan / goal recognition). Dieses Ziel wird dann von den Strategiekomponenten erfüllt . Auch in EMBASSI können unterschiedliche Benutzeraktionen dieselbe Intention ausdrücken.

1.9.4. Multi-Modalität im Intelligent Classroom

Zwei unterschiedliche visuelle Aufgaben werden vom Intelligent Classroom unterstützt, das eine ist die Erkennung von Aufgaben, die der Classroom für den Benutzer ausführen soll, das andere behandelt das Thema User Tracking [Flachsbart2000]:
Die Erkennung von Aufgaben untergliedert sich hier in die folgenden Punkte:

- Plan Recognition (wie oben bereits beschrieben)
- Eine Vorlesung filmen (hierbei soll nicht einfach die Vorlesung in der Totalen abgefilmt werden, sondern immer mit dem Sprecher im Mittelpunkt bzw. mit Heranzoomen auf die Dinge, die er zeigt)
- Das Benutzer von Control Icons (hiermit sind Icons gemeint, die mittels Projektor auf die Wand geworfen werden, oder vom Sprecher auf die Tafel gemalt werden).

Die gerade beschriebenen Aufgaben können nur dann ausgeführt werden, wenn dem System zu jeder Zeit bekannt ist, wo sich der Sprecher befindet. Dies wird, wie oben bereits erwähnt, mittels der „Background Substraction“ Technik ermittelt.

Die unterschiedlichen Aufgaben des visuellen Systems werden in sog. Input-Moden beschrieben. Das System hat laut Intelligent Classroom die Fähigkeit die Wahrscheinlichkeit der Aktivität der verschiedenen Input-Moden in Betracht zu ziehen (z.B. kann das System (aus Auslastungsgründen) nicht andauernd das Geschehen auf das Betätigen bzw. Erkennen von Control Icons überwachen). Aus diesem Grunde ist Intelligent Classroom in der Lage aus den verschiedenen Input-Modes jeweils den für die jeweilige Situation wahrscheinlichsten genauer zu überwachen. Neue Tätigkeiten des Benutzers erzwingen die Änderung des aktuellen Input-Modes zu einem anderen, genauso wie die Tatsache, dass ein aktuellen Input-Mode zur Bewältigung einer aktuellen Aufgabe nicht die richtigen Informationen liefern kann. In beiden Fällen überprüft das System die anderen Input-Nodes auf deren Wirksamkeit auf die aktuellen Situation.

Parallel dazu gibt es im Intelligent Classroom auch die Möglichkeit der Steuerung mittels Sprache. Hierbei setzt jedoch nicht so ein detaillierter Interpretationsprozess an, da per Sprache die Aktionen vom Sprecher meistens explizit vorgeben werden (z.B. Bitte die letzte Folie noch mal).

1.9.5. Zusammenfassung / Einschätzung

Der Intelligent Classroom ist im Grunde weniger als Infrastruktur als als Anwendung zu sehen. Ein spezifisches Szenario (Vortragssaal) wurde mit Mitteln der Multimodalen Interaktion, Planungskomponenten auf Basis von Produktions- und Expertensystemen und Trackingkomponenten umgesetzt.
Es ist daher weniger die technologische Basis von Intelligent Classroom, die für DynAMTE von Interesse ist, daher mehr oder weniger bekannte Technologien miteinander verknüpft wurden. Jedoch aus einem anderen Grund ist dieses Projekt interessant: Es liefert ein spannendes Szenario, welches eine große Herausforderung für selbstorganisierte Systeme darstellt und sollte aufgrund dessen von DynAMITE beobachtet werden. Und zwar im Hinsicht auf die Frage, ob eine DynAMTE-Infrastruktur, sollte sie über die technologischen Komponenten verfügen, in der Lage wäre adäquates zu leisten.

Literaturreferenzen zu [Kapitel 1.9]

[IntelligentClassroom2003] Intelligent Classroom, Project Homepage: http://dent.infolab.nwu.edu/infolab/projects/project.asp?ID=11

[Franklin1998_1] Franklin, D., and Flachsbart, J. (1998). All gadget and no representation makes Jack a dull environment. In Proceedings of AAAI 1998 Spring Symposium on Intelligent Environments. AAAI TR SS-98-02.

[Franklin1998_2] Franklin, D. (1998). Cooperating with people: The Intelligent Classroom. In Proceedings of the Fifteenth National Conference on Artifical Intelligence (AAAI-98)

[Flachsbart2000] Flachsbart, J., Franklin, D. and Hammond, K. (2000). Improving Human Computer Interaction in a Classroom Environment using Computer Vision. In Proceedings of the Conference on Intelligent User Interfaces (IUI-2000)

1.10. Aware Home / Abowd GVU

Im Rahmen des AwareHome Projekts [>AWA03, >SAN00] wurde ein dreistöckiges Haus in ein Labor für ubiquitäre Sensortechnologien und Interaktionsparadigmen umgebaut. Das Projekt AwareHome besteht aus mehreren Teilprojekten. Der Schwerpunkt aller Projekte ist die Erkennung und Überwachung von Benutzeraktionen mit Hilfe von fusionierten Sensordaten. Innerhalb des Teilprojektes „SmartSpaces“ werden Techniken untersucht, mit denen sich Kameradaten und Audiodaten fusionieren und interpretieren lassen (z.B. zur automatisierten Erstellung eines Meeting-Protokolls). Im Teilprojekt „Ubiquitous Audio and Video“ werden Techniken zur multiplen Personenerkennung untersucht. Das Teilprojekt „Support for Aging in Place”

beschäftigt sich mit Techniken, wie Senioren durch ein AwareHome bei Alltagsaufgaben unterstützt werden können. Im Teilprojekt „Context Toolkit“ werden Möglichkeiten der Kontextadaptivität im AwareHome untersucht. So werden Benutzer z.B. mit Buttons ausgestattet, die eine drahtlose Benutzererkennung zulassen. Weiterhin beschäftigen sich alle Projekte auch mit Fragen der Sicherheit von privaten Daten.

Da innerhalb des Projektes keine Fragen bzgl. Selbstorganisation oder Interaktionen mit mehreren Geräten untersucht werden, ist das Projekt nur von geringer Relevanz für Dynamite.

Literaturreferenzen zu [Kapitel 1.10]

[>AWA03] [AHRI] - Aware Home Research Initiative, http://www.cc.gatech.edu/fce/ahri/projects/index.html, 2003.

[>SAN00] Sanders, J., Sensing the Subtleties of Everyday Life, Research Horizon Magazine, http://gtresearchnews.gatech.edu/reshor/rh-win00/main.html, 2000.

[>WJHo97] Andy Ward, Alan Jones, Andy Hopper. A New Location Technique for the Active Office. IEEE Personal Communications, Vol. 4, No. 5, October 1997, pp. 42-47.

2. Multimodale Ausgabekoordination

Klassische multimodale Präsentationssysteme (z.B. [>AMR96,>FM98,>KCRM97,>ZF98]) waren auf statische Ausgabeumgebungen beschränkt. Die Systeme konnten zwar für verschiedene Ausgabegeräte konfiguriert werden (z.B. bzgl. der Auflösung der Displays), aber es wurde keine Dynamik bei den Geräten selber berücksichtigt. Im folgenden werden exemplarisch vier Arbeiten vorgestellt, die multimodale Ausgabekoordination in dynamischen und/oder heterogenen Geräteumgebungen durchführen.

2.1. Fluidum

Das 2003 gestartete Projekt Fluidum der Universität Saarbrücken betrachtet multimodale Interaktionen in Räumen und Gebäuden. Das Ziel des Projektes ist die Realisierung von Benutzerschnittstellen für einen Pool von Geräten, die über ein Gebäude hinweg verteilt sind. Dies geschieht auf der Basis von Repräsentationen der Interaktionsfähigkeiten von Geräten wie Laptops oder PDAs sowie von Sensoren. Dabei soll ein übergeordneter Gebäude-Manager die Interaktionen mit den Geräten steuern. In [>BKW02] wird ein mobiles Indoor-Navigationssystem präsentiert, das in den Benutzer mit Hilfe von Infrarot-Barken durch ein Gebäude führt. Die Präsentationen werden angepasst an die Display-Größe des mobilen Gerätes sowie an die Signalstärke. Weiterhin können an bestimmten Positionen Info-Terminals bedient werden, auf denen komplexere Präsentationen darstellbar sind. In [>KK03] werden Interaktionsparadigmen zwischen PDAs und großen Displays untersucht. So lässt sich ein PDA zum Beispiel als Lupe verwenden, um bestimmte Teile eines großen Displays zu bearbeiten. Bisher wurden innerhalb des Projektes noch keine Mechanismen für die Koordination von multimodalen Ausgaben in der beschriebenen dynamischen Geräteumgebung realisiert.

Die innerhalb des Projektes untersuchten Interaktionsformen in einer dynamischen Geräteumgebung sind für Dynamite relevant. Jedoch scheinen keine Fragen bzgl. Selbstorganisation bearbeitet zu werden.

2.2. SmartKom

Das Projekt SmartKom betrachtet multimodale Dialoge in mehreren Szenarien. Die Ausgabe setzt sich je nach Szenario zusammen aus geografischen Karten, dem animierten Assistenten des Systems sowie Sprachausgabe [>MPT02]. In SmartKom wird ein plan-basierter Ansatz auf der Basis eines Multiagentensystems verwendet zur Generierung der Präsentationen. Die Präsentationen in SmartKom werden angepasst an die Größe des Displays des Ausgabegerätes. Allerdings werden weder mehrere parallele Ausgabegeräte noch das dynamische Hinzufügen/Entfernen von Ausgabegeräten betrachtet.

Die in SmartKom realisierte planbasierte Generierung von multimodalen Ausgabe ist auch für Dynamite relevant, da hier ebenfalls kombiniertes Sprach-Grafik-Feedback mit Hilfe von animierten Charakteren untersucht werden wird. Allerdings werden in Dynamite Interaktionen über mehrere Ausgabegeräte hinweg untersucht werden.

2.3. Embassi

Innerhalb des Embassi Projektes wurde eine multimodale Ausgabe realisiert, die sich aus einem animierten Assistenten, einer grafischen Benutzeroberfläche und einer Sprachausgabe zusammensetzte [>EM01]. Als Ausgabegerät stehen der Display eines TV-Gerätes sowie die Lautsprecher des TV-Gerätes zur Verfügung. Weiterhin wurde ein Modell zur Dienstbeschreibung von Ausgabeagenten entwickelt, das das dynamische Hinzufügen und Entfernen von Ausgabegeräten ermöglicht [>ERMS03].

Aus demselben Grund wie bei SmartKom sind auch die in Embassi geleisteten Arbeiten im Bereich der multimodalen Ausgabeplanung für Dynamite relevant.

2.4. PREMO

Die PREMO Architektur [>BFM97] definiert ein generisches Framework für die automatisierte Generierung von multimodalen Präsentationen. So werden Präsentationen wissensbasiert auf verschiedenen semantischen Ebenen generiert und koordiniert. Dabei werden explizite Wissensmodelle über die Applikationen, den Kontext, den Benutzer und die Präsentationsstrategien verwendet. Das Ziel des Standards ist es eine gemeinsame Terminologie für multimodale Präsentationssysteme zu schaffen und deren Vergleichbarkeit anhand klar definierter Kriterien zu ermöglichen. Die PREMO-Architektur bietet ebenfalls eine Taxonomie für multimodale Ausgaben („primitive Hierarchy“), die eine Vorraussetzung für die Selbstbeschreibung von Ausgabeagenten ist. Diese Hierarchie beschreibt multimodale Ausgaben allerdings nur in sehr technischer

Weise, was für eine detaillierte Beschreibung multimodaler Ausgabefähigkeiten nicht genügt (vgl. [>B01]). Aus diesem Grund ist die PREMO Architektur nicht für Dynamite relevant.

2.5. Situated Computing Framework

Das Situated Computing Framework [>PSG00] ermöglicht die drahtlose Steuerung eines Pools von Windows-PCs mittels eines PDAs. Dazu nähert sich der Benutzer einem Pool von PCs, auf die über die Infrarotschnittstelle des PDAs eine Verbindung aufgebaut wird. Diese PCs bieten ihre Fähigkeiten über eine Http-Schnittstelle an. Dabei kann die Wiedergabe von Medien an die aktuelle Ausgabegeräteumgebung angepasst werden. Verfügt der PDA z.B. nicht über Lautsprecher, so kann die Audio-Ausgabe auch auf andere Lautsprecher in der Nähe des Benutzers umgeleitet werden. Der Nachteil des Systems ist, dass es nur mit bereits vorgenerierten Medien arbeitet. So ist es z.B. nicht möglich fehlende Grafikinformationen (z.B. aufgrund der geringen Display-Größe des PDAs) durch eine komplexere akustische Ausgabe auszugleichen. Das Streaming von Mediendaten in Abhängigkeit der Geräteressourcen ist relevant für Dynamite, da dieser Fall bei der Darstellung von Video-/Sound-Daten auf PDAs ebenfalls in Dynamite betrachtet werden muss.

Literaturreferenzen zu [Kapitel 2]

[>AMR96] André, E., Müller J., Rist T., WIP/PPP: Automatic Generation of Personalized Multimedia Presentations, 4th ACM International Multimedia Conference, Boston, MA, November 1996.

[>BFM97] Bordegoni, M. , Faconti, G., Maybury, M., Rist, T. , Ruggieri, S., Trahanias, P., Wilson, M., A Standard Reference Model for Intelli- gent Multimedia Representation Systems, International Journal on the Development and Applications of Standards for Computers, Data Communications and Interfaces, 1997.

[>B01] Bernsen, N. O.: Multimodality in language and speech systems - from theory to design support tool, Granström, B. (Ed.): Multimodality in Language and Speech Systems, Dordrecht: Kluwer Academic Publishers 2001.

[>BKW02] Baus, J., Krüger, A., Wahlster W., A resource-adaptive mobile navigation system, IUI2002: International Conference on Intelligent User Interfaces, New York, 2002.

[>EM01] Elting C., Michelitsch G., A Multimodal Presentation Planner for a Home Entertainment Environment, Workshop on Perceptual User Interfaces PUI01, Orlando, FL, November 15-16, 2001.

[>ERMS03] Elting, C., Rapp, S., Möhler, G., Strube, M., Architecture and Implementation of Multimodal Plug and Play, International Conference on Multimodal Interfaces ICMI03, Vancouver, Canada, November 5-7, 2003.

[>FM98] Feiner, S. K., McKeown K. R., Automating the Gener-ation of Coordinated Multimedia Explanations, in: Mark Maybury & Wolfgang Wahlster (eds.), Readings in Intelligent User Interfaces, pp. 89-97, Morgan Kaufman Publishers, 1998.

[>F03] FLUIDUM – Flexible User Interfaces Distributed Ubiquitous Machinery, http://www.fluidum.org, 2003.

[>KCRM97] Kerpedjiev, S., Carenini, G., Roth, S, Moore, J., Integrating Planning and Task-based Design for Multi-media Presentation, International Conference on Intelligent User Interfaces IUI'97, pp.145-152, Orlando, FL, 1997.

[>KK03] Kruppa, M., Krüger, A., Concepts for a Combined Use of Personal Digital Assistants and Large Remote Displays, Proceedings of SimVis 2003, Magdeburg, March 2003.

[>MPT02] Müller, J., Poller, P., Tschernomas, V., Situated Delegation-Oriented Multimodal Presentation in SmartKom, Workshop Intelligent Situation-Aware Media and Presentations ISAMP, Edmonton, 2002.

[>NMH02] Nichols, J., Myers, B. A., Higgins, M., Hughes, J., Harris, T. K., Rosenfeld, R., Litwack, K., Personal Universal Controllers: Controlling Complex Appliances with GUIs and Speech, ," In Extended Abstracts of CHI'2003, April 5-10. Ft. Lauderdale, FL. pp. 624-625.

[>PSG00] Pham, T. L., Schneider, G., Goose, S., A Situated Computing Framework for Mobile and Ubiquitous Multimedia Access Using Small Screen and Composite Devices, ACM International Conference on Multimedia, Los Angeles, CA, 2000.

[>SGH99] Streitz, N., Geißler, J., Holmer, T., Konomi, S., Müller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., Steinmetz, R., i-LAND: An Interactive Landscape for Creativity and Innovation, CHI'99, Pittsburgh, PA, 1999.

[>ZF98] Zhou, M. X., Feiner S. K., Efficiently Planning Coherent Visual Discourse, Journal of Knowledge-based Systems, 10(5) pp. 275-286, 1998.

3. Agentenplattformen und Agenteninteroperabilität

3.1. The Open Agent Architecture

Laut der Open Agent Architecture erfordert die sog. Post-PC-Bewegung (oder auch ubiquitious oder pervasive Computing genannt), wo verbundene Computer in fast allen Tagesarbeiten mit hineinwirken (Levy1999) ein System, welches flexibel genug ist, auf die hohe Dynamik neu hinzukommender und entfernter Komponenten zu reagieren. Mit der Open Agent Architecture soll das Paradigma der fest definierten Agentensysteme (siehe KQML oder FIPA) hin zu flexiblen Systemen verschoben werden, die flexibel auf die Anforderungen der dynamisch sich verändernden Ensembles reagieren können.

Die Open Agent Architecture hat sich zum Ziel gesetzt die Eigenschaften, der Methoden, der

- Distributed Objects: basiert auf Methodenaufrufen ferner Objekte (vgl. CORBA, DCOM). Nachteile: fest kodierte Objekt und Methodenaufrufe.
- Conversational Agents: erlaubt die Kooperation unter Agenten durch den Austausch eines mächtigen Wortschatzes an Speech Acts (vgl. KQML, FIPA). Nachteile: Interaktion findet häufig hartcodiert und damit statisch statt.
- Blackboards: hier findet Zusammenarbeit unter Agenten statt, indem Agenten, die nach Diensten anfragen, diese an ein sog. Blackboard posten und andere Agenten von diesem Blackboard ständig Anfragen „absaugen“. Mit diesem Ansatz sei Flexibilität (d.h. Dynamische Ensembles) möglich, aber die Kontrolle über das Gesamtgeschehen und damit die kontrollierte Zusammenarbeit des Agentenensembles ist nicht möglich.

zu vereinen.

3.1.1. Das Delegated Computing Model

Das Delegated Computing Model erlaubt es, sowohl den Menschen (über User Interfaces) als auch anderen Komponenten ihre Anfragen in der Form des „Was ist zu tun ?“ mitzuteilen, ohne spezifizieren zu müssen, wer es auszuführen hat und wie es auszuführen ist. Agenten können also die Aufgabe ein Ziel zu erreichen und die Kontrolle, wie und von wem dieses Ziel zu erreichen ist, an eine zentrale Stelle delegieren.
Hierfür verwendet die OAA einen speziellen zentralen Server-Agent (Facilitator), der die Fähigkeiten und die Anfragen verwaltet und entsprechend verteilen kann.

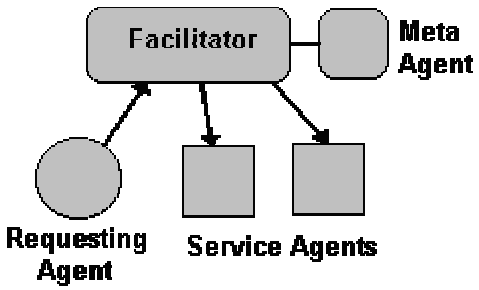

Abbildung 5: Komponententypen der Open Agent Architecture

Dabei sind die Fähigkeiten von OAA an 4 verschiedenen Typen von Komponenten lokalisiert (siehe Abbildung 5):

- Requesting Agent (oder Requester): spezifiziert ein Ziel für den Facilitator und gegebenenfalls Hinweise, wie es erfüllt werden soll.
- Providers: die ihre Fähigkeiten beim Facilitator anmelden (Service Agents)
- Facilitator: der eine Liste an verfügbaren Providers verwaltet und einen Satz an globalen Strategien besitzt, wie Ziele umgesetzt werden können
- Meta-Agents: die über Domänen- oder Zielspezifische Kenntnisse und Strategien verfügen. Diese Kenntnisse können vom Facilitator als Hilfe verwendet werden.

Der Facilitator ist hier also zuständig, die Anfragen zu verwalten, die Leistungen der Provider-Agenten zu koordinieren und die Antworten an den anfragenden Agenten zurückzusenden. Gleichzeitig (in der Analogie zu den Distributed Objects und den Conversational Agents) ist auch die direkte Adressierung von Agenten

oder Broadcasts möglich. Agenten werden in der OAA auch in Applikationsagenten und als User-Interface-Agenten unterschieden.

Dieses Delegationsmodell soll die Agenten von der Verantwortlichkeit der Aufgabenplanung (auch Aufgabenzerlegung) und der Ausführungsbeobachtung und Kontrolle entlasten. Die Vorteile seien:

- Reduziere die Komplexität für Benutzer und Agenten, da sie nur das Ziel und einige Hinweise bereitstellen müssen. Danach können sie sich wieder anderen Aufgaben widmen und müssen keine Erfolgskontrolle leisten.
- Garantiere eine höhere Flexibilität, in der Agenten „on the fly“ hinzugefügt oder weggenommen werden können, ohne eine Umprogrammierung der nötigen Kontrollintelligenzen nötig zu machen.
- Ermögliche Wiederverwendbarkeit von Agenten, da ihre Kommunikationswege nicht hardcodiert sind.

3.1.2. Kooperation von Agenten in der OAA

Laut der OAA gibt es unbedingte Voraussetzungen für effektives Kommunizieren zwischen den einzelnen unabhängigen Akteuren:

- Ein Transport-Mechanismus, der die Nachrichten asynchron übermittelt
- Ein Interaktionsprotokoll, der die unterschiedlichen Arten an Nachrichtenaustausch definiert (semantisches Protokoll)
- Eine Content-Sprache die Ausdruck und Interpretation von Äußerungen zulässt
- Eine Ontologie

Die Mechanismen der Kooperation laufen in der OAA in drei Schritten ab:

- Provider von Diensten registrieren die Spezifikationen ihrer Fähigkeiten beim Facilitator
- Diensteanfrager konstruieren Ziele und leiten sie an den Facilitator weiter
- Der Facilitator koordinierte die Leistungen der passenden verfügbaren Serviceprovider um das übertragene Ziel zu erfüllen.

Hierzu ein Auszug aus der Beschreibung der OAA:
„An agent requests services of the community by delegating tasks or goals to its facilitator. Each request contains calls to one or more agent solvables, and optionally specifies parameters containing advice to help the facilitator determine how to execute the task. It is important to note that calling a solvable does not require that the agent specify (or even know of) a particular agent or agents to handle the call. While it is possible to specify one or more agents using an address parameter 8and there are situations in which this is desirable), in general it is advantageous to leave this delegation to the facilitator. Programming in this style greatly reduces the hard-coded dependencies among components that one often finds in other distributed frameworks.”

Wie oben beschrieben, kann der Serviceanfrager dem Facilitator Hinweise zur Erfüllung des Zieles mitgeben. Diese betreffen Angaben, z.B. über Zeitlimits, ob die Anfrage / das Ziel von einem einzelnen oder im Verbund erfüllt werden sollen / können und Angaben, wie mögliche Antworten der Provider-Agenten gefiltert oder miteinander kombiniert werden können. Oder auch, ob der Serviceanfrager glaubt, einen gut geeigneten Serviceprovider bereits zu kennen. Ebenso ist es möglich, dem Facilitator einen Vorschlag zu machen, in welche Teilziele das gegebene Gesamtziel zerlegt werden kann und ob diese Teilziele zugleich oder nacheinander ausgeführt werden sollen.

3.1.2.1. Die Interagent Communication Language (ICL)

Als Nachrichten zwischen Agenten dienen sog. Events. Diese entsprechen den Zielen, die Agenten erreichen sollen. Jedes Event besteht aus einer Typenbezeichung, einem Satz an Parameter und einem Content.
Zur Beschreibung von Fähigkeiten (capabilities declarations / solvables) wird ebenso die ICL eingesetzt. Hierbei wird unterschieden, zwischen Fähigkeiten, die ein Ziel erreichen (procedure solvables) und Fähigkeiten, die Zugang zu Daten / Datensammlungen (data solvables) bieten.

3.1.3. Anwendungen

Auf der Webseite lassen sich verschiedene bereits realisierte Applikationen als Video oder als veröffentlichtes Paper oder als Beschreibung ansehen (http://www.ai.sri.com/~oaa/applications.html). Einige seien hier genannt (Auszüge aus der Webseite):

3.1.3.1. Surf

Surf provides an OAA-based interface for your TV, where you can task your community of agents (home appliances and information services) by speaking into your remote control. Events (e.g. the phone ringing) can also trigger informational updates on the TV screen (e.g. the phone number using caller-id).

3.1.3.2. TravelMATE And CARS

TravelMATE uses GPS, a compass, a 3D virtual reality terrain database and a smart windshield to augment a tourist's driving experience with context-relevant information. CARS provides a speech-enabled interface to systems and services provided by the car and by the Internet.

3.1.4. OfficeMATE

The objective is to give SRI visitors a wireless tablet computer that augments their experience during the day. Position tracking helps the user navigate SRI hallways and resources, and information is updated on the display based on the user's location and interests.

3.1.5. Verfügbare Distributionen

Auf der offiziellen Webseite ist bereits in der Version OAA 2.3 ein Facilitator in verschiedenen Sprachen und für die verschiedensten Betriebssysteme verfügbar (siehe Tabelle 1).

Quintus Prolog	SunOs/Solaris, Windows 9x/NT/2000, other Quintus-supported platforms
Sicstus Prolog	SunOs/Solaris, Linux, Windows 9x/NT/2000, other Sicstus-supported platforms
ANSI C/C++ (Unix, Microsoft, Borland)	SunOs/Solaris, Linux, Windows 9x/NT/2000
Java	Any Java platform
Compaq's Web Language	Any Java platform

Tabelle 1: Verfügbare Distributionen der Open Agent Architecture nach Programmiersprache und Betriebssystem

Die Version OAA 2.3 beinhaltet folgende Teile (Auszüge aus der entsprechenden Webseite):

- OAA 2.3 Facilitator
- OAA Monitor: graphically trace and profile agent community
- OAA Start-It: launches a community of agents, and restarts agents if they unexpectedly terminate
- OAA Debug Interface: Send queries and messages to the agent community or to an individual agent, using English or the Interagent Communication Language (ICL) (Note: two similar versions of Debug are provided, one written in Java and one in C. They are functionally equivalent)
- OAA Shell Agent: Send queries and messages to the agent community from the (Unix or DOS) command line.
- OAA-WebL Library: Rapidly define OAA agents in Compaq's Web Language (previously known as WebL) to extract information from the web.
- Sample Application including:
    - DCG-NL Natural Language Parser
    - WebL script for finding weather from cities
    - WebL script for extracting employee info from simple web database
- Eliza agent and text-based client interface:Ask questions of a phony, annoying psychotherapist

Es sind sowohl Sourcen für Windows als auch Sourcen für UNIX-Systeme verfügbar. Die Sourcen liegen sowohl in C++ (cpp-Files) als auch in Java (für die Version JDK 1.4) vor.

3.1.6. Zusammenfassung / Einschätzung

Die Open Agent Architecture bietet sehr interessante Ansätze und wird nicht zuletzt deswegen oftmals als wichtige Referenz angesehen. Als einer der ersten Agentenplattformen wird hier die Kommunikation von Agenten untereinander nicht mehr auf Basis von Agentennamen, sondern mittels Events realisiert. Die

Weiterverarbeitung der Events (z.B. mögliche Empfänger etc.) geschieht durch den Router / Facilitator auf Basis des Wissens von Meta-Agenten. Die Unterschiede des Ansatzes der OAA zu den Zielen von DynAMITE sind aber offensichtlich:

- OAA besitzt zentrale Router und ist somit per se zentralisiert.
- Die Intelligenz über Auftragsausführungen ist sowohl in den Agenten (die mögliche strategische Hilfen mitsenden) als in den Meta-Agenten, als auch im Facilitator lokalisiert. Dies mag ein Ansatz sein, um das System flexibel zu gestalten, erschwert jedoch die Transparenz und auch die Entwicklung solcher Agenten.
- Auch die Frage, woher die Intelligenz der Meta-Agenten kommt ist nicht hinreichend beantwortet. Im „schlimmsten“ Fall basieren die Regeln auf Expertensystemen, deren Erweiterung im Falle neu hinzukommender Agenten schwer ist.
- Der Ansatz der Events anstatt direkter Kommunikation und der Ansatz Intelligenz zur Lösung von Konflikten etc. auszulagern, entspricht jedoch dem Ansatz von SodaPop (hier Events und RPCs und verteilte Strategien auf dem Channels) und scheint zukunftsträchtig zu sein. In so fern sollte DynAMITE die Entwicklungen der Open Agent Architecture auf alle Fälle weiterverfolgen.

Literaturreferenzen zu [Kapitel 3.1]

[OAA2001] The Open Agent Architecture Homepage: http://www.ai.sri.com/~oaa/

[Martin1999] Martin, David L. and Cheyer, Adam J. and Moran, Douglas B. *The Open Agent Architecture: A Framework for Building Distributed Software Systems*. Applied Artificial Intelligence, vol. 13, no. 1-2, pp. 91-128, January-March 1999.

[OAA_Whitepaper2003] Open Agent Architecture: Technical White Paper, http://www.ai.sri.com/~oaa/whitepaper.html

[Levy1999] Levy, Steven, "The New Digital Galaxy," Newsweek Magazine, May 31,1999, pp. 57-62

3.2. Fipa

FIPA (Foundation for Intelligent Physical Agents) ist eine internationale Organisation, die im Jahr 1996 gegründet wurde, um Softwarestandards für heterogene und interagierende Agenten sowie agentenbasierte Systeme zu definieren (http://www.fipa.org). Im Jahr 2002 waren bereits über 60 universitäre und industrielle Mitgliedsorganisationen in FIPA vertreten.

Die *FIPA-Spezifikationen* sind eine Sammlung von Standards, die der Interoperabilität zwischen Agenten und zwischen agentenbasierten Diensten dienen. Die Spezifikationen wurden und werden in Form von *FIPA-Agentenplattformen* implementiert. Einige davon sind öffentlich verfügbar (http://www.fipa.org/resources/livesystems.html). Ein Agentenplattform gilt als *FIPA-kompatibel*, wenn es mindestens die Spezifikationen zum Agentenmanagement [>FIPA00023] und zur Agentenkommunikation [>FIPA00061,>FIPA00086] implementiert. Im Folgenden werden diese Spezifikationen kurz erläutert.

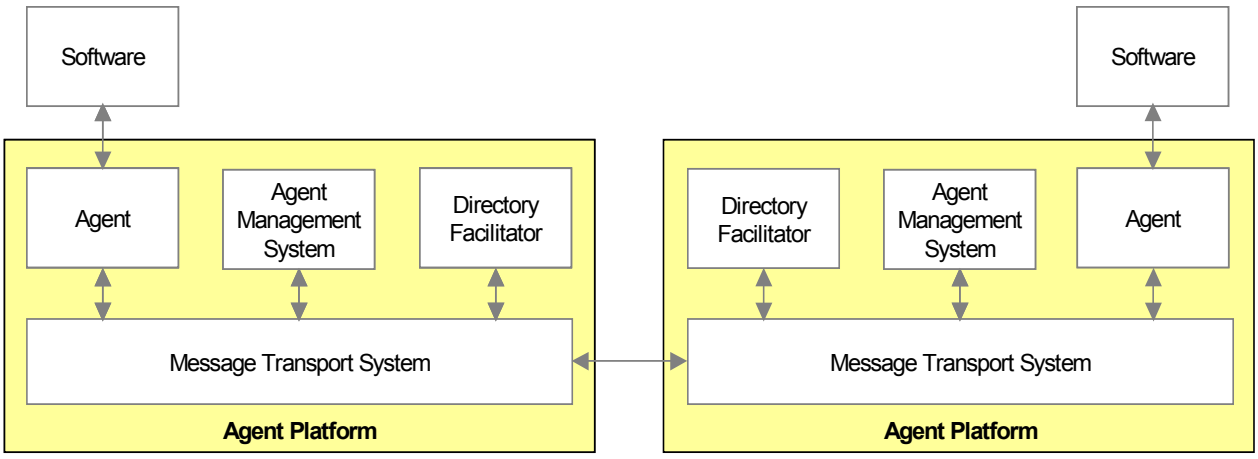

Abbildung 6: Referenzmodell des Agentenmanagements

Agentenmanagement bildet ein Framework, innerhalb diesem FIPA-Agenten existieren und operieren. Es definiert ein logisches Referenzmodel für die Generierung, Registrierung, Lokalisierung, Kommunikation, Migration und Ausscheiden der Agenten. Das Referenzmodel (siehe Abbildung 6) beinhaltet folgende logische Komponenten mit ihren jeweiligen Fähigkeiten:

- Ein *Agent* ist ein Prozess, der die autonome und kommunikative Funktionalität einer Applikation implementiert. Agenten kommunizieren über eine Agentenkommunikationssprache (engl. *Agent Communication Language*, kurz ACL). Ein Agent stellt einen oder mehrere Dienste zur Verfügung, die durch Dienstbeschreibungen (engl. *Service Description*) bekannt gegeben werden. Jeder Agent muss mindestens einen Besitzer haben und hat eine eindeutige Kennung (engl. *Agent Identifier*, kurz AID). Jeder Agent kann über mehrere Transportadressen verfügen, über die er kontaktiert werden kann.
- Ein *Agentenplattform* (engl. *Agent Platform*) stellt eine physikalische Infrastruktur zur Verfügung, in der Agenten sich entfalten können. Ein Agentenplattform setzt sich aus einem oder mehreren Rechnern, Betriebssystemen, Agentensoftware und FIPA-Agentenmanagementkomponenten (AMS, DF, MTS, siehe unten) zusammen. Agenten einer Agentenplattform müssen nicht auf demselben Hostrechner beheimatet sein. Stattdessen kann eine Agentenplattform verteilt sein, d.h. die Agenten einer Agentenplattform können auf mehrere Rechner verteilt sein.
- Ein *Agentenmanagementsystem* (engl. *Agent Management System*, kurz AMS) ist eine obligatorische Komponente jeder Agentenplattform. Jede Agentenplattform, auch eine verteilte Agentenplattform, verfügt über genau ein AMS. Ein Agent meldet sich beim AMS an und erhält seine eindeutige Kennung AID, die u.a. die Transportadresse des Agenten beinhaltet. Das AMS verwaltet ein Verzeichnis der AIDs für registrierte Agenten und stellt anderen Agenten sogenannte „white pages services“ zur Verfügung.
- Ein *Directory Facilitator* (DF) ist eine optionale Komponente einer Agentenplattform. DF stellt anderen Agenten sogenannte „yellow pages services” zur Verfügung. Agenten registrieren sich beim DF mit ihren Diensten und können über den DF herausfinden, welche Dienste andere Agenten anbieten. Mehrere DF können auf einer Agentenplattform existieren und bilden einen Verbund.
- Ein Nachrichtentransportdienst (engl. *Message Transport Service*, kurz MTS) implementiert die Standardmethode zur Interplattformkommunikation, d.h. Kommunikation zwischen Agenten auf verschiedenen Agentenplattformen.
- Eine *Software*-Komponente ist eine nichtagentenbasierte und ausführbare Einheit, auf die Agenten zugreifen können. Ein Agent greift auf Softwarekomponenten zu, um beispielsweise neue Dienste anbieten zu können bzw. neue Protokolle, Algorithmen oder Tools zu erwerben.

Die Spezifikationen der FIPA-Agentenkommunikation befassen sich mit der Kommunikationssprache, den Kommunikationsakten (engl. *communicative acts*), die auf der Sprechakttheorie basieren, den Interaktions-protokollen für den Nachrichtenaustausch sowie der Repräsentation der Inhalt einer Nachricht.

Jede FIPA-Nachricht setzt sich aus mehreren Parametern zusammen, die den Kommunikationsakt, die Teilnehmer der Kommunikation und den Inhalt der Nachricht bestimmen, den Nachrichteninhalt beschreiben (beispielsweise der Sprache, der Kodierung und der Ontologie des Nachrichteninhalts) und die Konversation, zu der die Nachricht gehört, kontrollieren. Eine Konversation ist eine Sequenz von korrespondierenden Nachrichten. Die Parameter, die eine Konversation kontrollieren, bestimmen beispielsweise die eindeutige Konversationskennung, die Interaktionsprotokolle, die die zugelassenen Kommunikationsakte der Nachrichtensequenz vorschreiben (z.B. auf einer Anforderungsnachricht folgt eine Informations- bzw. Fehlermeldungsnachricht) und die Bezugsinformationen (z.B. Parameter *in-reply-to*, oder *reply-with*).

Ein bekanntes Beispiel des FIPA-kompatiblen Netzwerks ist das Agentcities-Netzwerk [>Agentcities], an dem derzeit über 160 Agentenplattformen partizipieren. Insbesondere basieren die meisten Agentenplattformen auf den weit verbreiteten FIPA-kompatiblen Agentenplattformen JADE [>JADE] und FIPA-OS [>FIPA-OS].

Literaturreferenzen zu [Kapitel 3.2]

[>Agentcities] Project Agentcities, http://www.agentcities.org.
[>FIPA00023] FIPA Agent Management Specification, http://www.fipa.org/specs/fipa00023.
[>FIPA00061] FIPA ACL Message Structure Specification, http://www.fipa.org/specs/fipa00061.
[>FIPA00086] FIPA Intology Service Specification, http://www.fipa.org/specs/fipa00086.
[>JADE] JADE-Java Agent Development Framework, http://sharon.cselt.it.
[>FIPA-OS] FIPA-OS, http://fipa-os.sourceforge.net.

3.3. KQML

KQML (The Knowledge Query and Manipulation Language) ist eine Sprache, die die Kommunikation zwischen Software-Agenten ermöglicht. KQML bietet eine Auswahl an Nachrichten-Typen (in KQML Performatives) genannt an, die die Stellung des Inhaltes der ausgetauschten Nachricht beschreiben. So zum Beispiel ob es sich bei der Nachricht um eine Frage handelt oder eine Anweisung. Daneben stehen noch Parameter bereit um die Nachricht näher zu beschreiben, so zum Beispiel den Absender, den Empfänger oder Nachrichtennummern. KQML definiert im wesentlichen drei Komponenten innerhalb eines Agentensystems: Die Agenten, die White-Page-Dienste und die Yellow-Page-Dienste. Zusätzlich zu KQML existiert eine Erweiterung um einige zusätzliche Performatives, die Diskussionen und Verhandlungen ermöglichen soll.

3.3.1. Architektur in KQML

KQML definiert im wesentlichen drei Arten von Komponenten innerhalb einer Agentenplattform:

- Agenten; sind fähig zur Kommunikation untereinander; arbeiten zusammen, um komplexe Ziele zu erreichen; handeln auf Eigeninitiative hin und nutzen lokale Informationen und Ressourcen (um diese wiederum zu manipulieren). Außerdem sind sie in der Lage Anfrage anderer Agenten zu beantworten.
- White-Page-Dienst: Agenten melden sich am White-Page-Dienst (zugleich der Router) an und können fortan ohne Kenntnis der aktuellen IP-Adresse der anderen Agenten mit den anderen Agenten Nachrichten austauschen. Der White-Page-Dienst übernimmt das Routing. FIPA definiert den White-Page-Dienst ähnlich, jedoch kann hier von ihm auch nur die IP des angefragten Agenten erfragt werden: FIPA erlaubt es somit auch, Nachrichten direkt auszutauschen.
- Yellow-Page-Dienst: Die gelbe Seiten agieren als ein Art Vermittlungsstelle. Über sie können die Namen von Agenten erfragt werden, die einen bestimmten Dienst ausführen können. Die KQML Spezifikation erlaubt auch eine Vermittlungsstelle, die zugleich sog. Mapping oder Übersetzungen von Nachrichten anbietet.

Neben der Definition der drei Basiskomponenten macht KQML auch grobe Angaben, wie Nachrichten zu übermittelt sind. Dabei soll gelten:

- Agenten sind unidirektional verbunden verknüpft
- Wenn ein Agent eine Nachricht erhält, weiß er auch von wem
- Nachrichtenempfang findet nach dem First-In-First-Out-Prinzip statt, das bedeutet schlicht, dass keine Zeitverzögerung beim Nachrichtenaustausch stattfinden darf
- Die Sendung von Nachrichten muss zuverlässig sein

3.3.2. Nachrichten in KQML

Einzelne Agenten tauschen in KQML Nachrichten aus, indem sie die Nachricht mit einem Performative spezifizieren, einige Parameter bzgl. Absender, Empfänger, verwendete Sprache oder Antwortnummern angeben und danach den eigentlichen Inhalt der Nachricht beschreiben. Das Vorgehen hier entspricht der Brief-Briefumschlag-Metapher. Der eigentliche Inhalt steckt im Briefumschlag, während auf dem Briefumschlag alle nötigen Angaben zum Brief gemacht werden können.

Eine Beispielnachricht zur Verdeutlichung:

```
(tell
        :sender AgentA
        :receiver AgentB
        :in-reply-to REPLY-NUMERx
        :reply-with REPLY-NUMBERy
        :language XML
        :ontology ONTOLOGY
        :content (<xml>…</xml>))
```

In dieser Nachricht schickt somit AgentA an AgentB eine Nachricht mit dem tell-Performative. Es handelt sich daher um eine rein informative Nachricht, womöglich um die Antwort auf eine Frage, die vorher von AgentB an AgentA gestellt wurde. Um Antworten zu vergangenen Fragen zuordnen zu können, können die Agenten bei den Parameters in-reply-to bzw. reply-with Nachrichtennummern spezifizieren. Mit dem Parameter language wird angegeben, dass der content der Nachricht in der Sprache XML formuliert wird. Der Slot ontology gibt einen Hinweis auf die verwendete Ontologie im content.

Die folgenden Parameter sind für eine KQML-Nachricht definiert:
performative, sender, receiver, from, to, in-reply-to, reply-with, language, ontology, content

Hinweis: der Parameter performative muss gegen eines der Performatives ausgetauscht werden und dient in der Definition der Parameter nur als Platzhalter.
Auf die Beschreibung der einzelnen Parameter und er einzelnen Performatives soll an dieser Stelle verzichtet werden. Diese sind in der angegebenen Literatur, die im Internet frei verfügbar ist, nachzulesen. KQML war der Nachrichtenaustauschstandard in EMBASSI und dürfte daher allen beteiligten DynAMITE-Partner geläufig sein, so dass auf eine Detaillierung der einzelnen Definitionen verzichtet werden kann.

Folgende Performatives sind in KQML definiert, um einen Diskurs zu führen:
ask-if, ask-all, ask-one, stream-all, eos, tell, untell, deny, insert, uninsert, delete-one, delete-all, undelete, achieve, unachieved, advertised, unadvertised, subscribe

Die folgenden Performatives sind definiert, um Interventionen und sog. Mechanische Eingriffe zu beschreiben:
error, sorry, standby, ready, next, rest, discard

und zuletzt die Performatives zum Interaktion mit den Yellow-Page und White-Page Funktionalitäten eines Facilitators und eines Routers:
register, unregister, forward, broadcast, transport-address, broker-one, broker-all, recommend-one, recommend-all, recruit-one, recruit-all

Zusätzlich zu KQML existiert eine Spezifikation einer Erweiterung um Diskussionen und Verhandlungen zwischen Agenten zu ermöglichen. Hierzu sind zusätzliche Performatives definiert worden, die an dieser Stelle näher betrachtet werden sollen:

- *propose*: This is used to propose to an agent a sub goal to achieve
- *counter-propose*: a counter proposal is another sub goal that partially satisfies the initial goal of a propose. The use of this speech act can result in a sequence of counter-proposals from both the origin proposer and the respondent.
- *accept*:
- *reject*: accept and reject are used to signal acceptance, respectively rejection of a proposal or counter-proposal. Rejection starts a new negotiation phase.
- *cancel*: this cancels a previously accepted proposal or counter-proposal
- *commit*: positive confirmation that an agent puts itself in a state that will satisfy a proposal; also ends a negotiation or renegotiation phase
- *de-commit*: cancellation of a previous commit
- *satisfy*: an agent announces that a requested goal has been achieved
- *fail*: an agent informs that execution of a committed goal has failed

3.3.3. Zusammenfassung / Einschätzung

Wie schon die Erfahrungen aus EMBASSI zeigten, basiert KQML auf dem Ansatz jeder Komponente einen festen Bezeichner zu geben. Dieser ist direkt (über TCP/IP) mit seiner IP-Adresse verknüpft. Dynamisierung ist daher schlecht möglich. Es besteht der Weg der Dynamisierung, indem

- Die Agenten ihre Fähigkeiten beim Facilitator anmelden
- Bei jeder Anfrage, den Facilitator fragen, welcher Agent einen bestimmten Dienst anbieten kann
- Und diesen dann kontaktieren

Die Erfahrung von KQML-Projekten zeigt aber, dass entweder kein Facilitator existiert (KQML selber bietet keine Software frei verfügbar an) oder dieser nicht genutzt wird und die Komponenten mit festen Kommunikatioswegen erstellt werden.
Zumal auch der Umweg über den Facilitator einige Probleme nicht löst, diese seien hier kurz angeführt:

- Lösung von Konkurrenzsituationen (Fall: Facilitator liefert mehr als einen möglichen Addressaten)
- Problem Decomposition (Fall, Facilitator liefert keinen möglichen Addressaten, dann muss die Nachricht unter Umständen in mehrere Teilprobleme zerlegt werden, die dann gelöst werden können)
- KQML hat mehrere zentrale Komponenten (hohes Ausfallrisiko, Bottleneckprinzip)
- KQML weist zwar den Slot Ontology aus, um anzudeuten mehrere Ontologien parallel zu unterstützen, diese können architektonisch jedoch nicht wiedergefunden werden

- Die Sprache COOL stellt einen interessanten Ansatz dar, Verhandlungen zwischen Agenten zu ermöglichen, jedoch beruhen auch diese Verhandlungen auf einer bidirektionalen Verbindung
- Bidirektionale Verbindungen bedeuten immer, dass jede mögliche Intelligenz im System in den verschiedenen Agenten implementiert ist und somit keine Systemleistung darstellt.

Die KQML Initiative ist schon recht alt und wird offenbar auch nicht weiterentwickelt. KQML ist es zu verdanken, explizite Sprechakte eingeführt zu haben (ähnlich der FIPA-Ansatz) und erste Lösungen für Service Discovery vorgestellt zu haben. Leider hat KQML keinerlei Ansätze bezüglich Konfliktlösungen anzubieten. Hier ist die Intelligenz jedem einzelnen Agenten überlassen. Gleichwohl soll erwähnt werden, dass Sprechakte benutzt werden können, um Events von RPCs oder ähnlichen zu unterscheiden und deswegen wird die DynAMITE Vorversion (die auf Arbeiten im Projekt EMBASSI basiert, in dem KQML als unterlagerte Kommunikation eingesetzt wurde) auf ein paar Sprechakte von KQML basieren. Dies wird jedoch im weiteren Projektkontext von DynAMITE nicht weiter verfolgt werden.

Literaturreferenzen zu [Kapitel 3.3]

[Finin, Fritzxon, McKay, 1992] Tim Finin, Rich Fritzson, Don McKay. A Language and Protocol to Support Intelligent Agent Interoperability. Proceedings of the CE & CALS Washington ´92 Conference, Juni 1992

[Mayfield, Labrou, Finin, 1995] James Mayfield, Yannis Labrou, Tim Finin. Desiderata for Agent Communication Languages, Proceedings of the AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments, AAAI-95 Spring Symposium, Stanford University, Stanford, CA. March 27-29, 1995

[McKay, Pastor, McEntire, Finin, 1996] Donald McKay, Jon Pastor, Rbon McEntire,Tim Finin. An architecture for information agents, Proceedings of the Third International Conference on Artificial Intelligence Planning Systems (ARPI Supplement), (AIPS-96), AAAI Press, May 1996

[Barbuceanu, Fox] COOL - A Language for Describing Coordination in Multi Agent Systems, Mihai Barbuceanu and Mark Fox, Enterprise Integration Laboratory, University of Toronto

[Labrou, Finin, 1997] Yannis Labrou, Tim Finin A Proposal for a new KQML Specification. TR CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250, February 1997

[http://www.cs.umbc.edu/kqml/, 2003] http://www.cs.umbc.edu/kqml/ Offizielle Webseite von KQML (maintained at the UMBC Lab for Advanced Information Technology)

3.4. Galaxy Communicator Infrastructure

Galaxy Communicator Infrastructure [Galaxy2001] ist eine Initiative der Spoken Language Systems Group [SLS2001] am MIT Laboratory for Computer Science. Galaxy definiert eine Infrastruktur, die entwickelt wurde um Sprachtechnologien zu integrieren. Dazu benutzt Galaxy eine Client-Server-Technologie, mit der Benutzer mit dem Gesamtsystem kommunizieren können indem sie kleine (light-weight) Klienten benutzen, während spezielle Server die schweren Aufgaben, wie Spracherkennung, Sprachverständnis, Datenbankzugriffe und Sprachsynthese erledigen. Die wichtigste Komponenten innerhalb der Galaxy-Architektur ist ein zentraler programmierbarer HUB, der den Datenfluss zwischen den verschiedenen Clients und den Servern kontrolliert und dabei den aktuellen Status und die Historie des aktuellen Gesprächs verwaltet.
Die Galaxy-Architektur ist die Referenzarchitektur des DARPA Communicator Programms. Diese Zusammenarbeit wurde im September 2003 beendet aber in Zusammenarbeit mit der MITRE-Corporation hat die SLS-Gruppe ein Open-Source Packet freigegeben, welches unter SourceForge.net verfügbar ist [Galaxy@SourceForge2003]. Eine komplette Dokumentation der Client-Server-Technologie und der frei verfügbaren Software ist unter [Galaxy_Documentation1_2003] einzusehen. Eine andere Dokumentation ist unter [Galaxy_Documentation2_2003] verfügbar.

Im Rahmen des Galaxy-Projektes sind im Zeitraum von 1995 bis 2003 viele Dissertationen und Veröffentlichungen entstanden. Diese sind komplett einsehbar unter [Galaxy_Publications2003]. Eine komplette Zusammenfassung aller Forschungsergebnisse des Projektes Galaxy ist unter [Galaxy_Summary2003] als PDF einzusehen. Die Rollenverteilung im Projekt war in etwa wie folgt: Das MIT (die SLS-Gruppe) stellte das Entwicklungsteam, DARPA war der Auftraggeber und leitete das Projekt und MITRE stellte die Aufsicht und testete die jeweils aktuellen Entwicklungen.

3.4.1. Die Galaxy Architektur

Die Galaxy Communicator Software-Infrastruktur ist eine verteilte, Nachrichten-basierte Hub-and-Spoke-Infrastruktur.
Die aktuelle Architektur des Galaxy Communicators basiert auf der Client-Server-Architektur von Galaxy aus dem Jahre 1994. Die Ziele der Neuentwicklung zusammen mit DARPA waren die Folgenden:

- Möglichkeit Dialogstrukturen leicht zu visualisieren
- Flexibel die Entscheidungsprozesse auf der obersten Ebene zu manipulieren
- Entwicklung einer Script-Sprache
- Schnelle Rekonfigurierungsmöglichkeiten (und dadurch Adaptivität auf neue Anwendungsszenarios) ohne Implementierungsaufwand

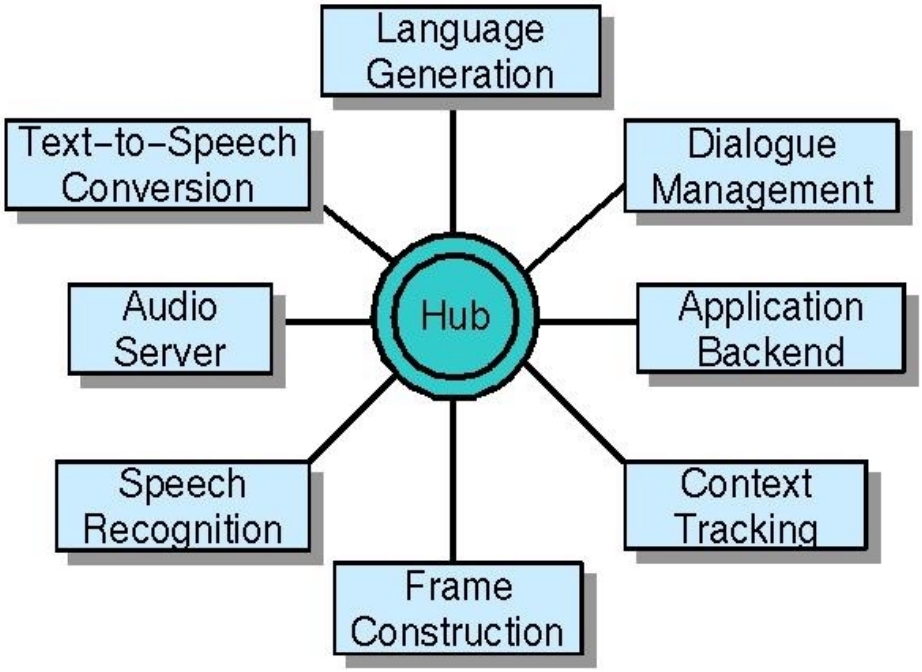

Abbildung 7: Galaxy Communicator Software-Infrastruktur

Die Architektur stellt verschiedene Server zur Verfügung, die jeder für sich für komplexe Aufgaben innerhalb der Gesamtarchitektur verantwortlich sind. Diese sind: Spracherkennung, Server für natürliches Sprachverständnis, natürliche Spracherzeugung und Sprachsynthese. Die Architektur gestaltet sich somit wie in der Abbildung 7 gegeben. Verschiedene Server, die verantwortlich sind für bestimmte Aufgabenbereiche sind mittels eines HUBs miteinander verbunden.
Vor dem Starten einer Anwendung werden mittels eines Skriptes die Liste der Server (Host, Port, Satz an verfügbaren Operationen) dem HUB mitgeteilt. Zugleich werden eine Menge an Regeln mitgegeben, wann bei welchen verfügbaren Variablen welche Nachricht an welchen Server zu schicken ist. Somit sind komplette Dialogprozeduren möglich, die der Benutzer mittels graphischen Frontend (auch über eine Webseite), mittels Spracheingabe oder speziellen Desktop-Agenten initiieren kann.

3.4.2. Nachrichten und Nachrichtentypen in Galaxy

Fast alle Kommunikationen in Galaxy werden in der Form sog. Frames beschrieben (siehe Abbildung 8):

Ein Frame ist eine Attribut-Wert-Struktur, die aus einem Namen besteht, einem Frametyp (der ist immer „c“) und einem gewissen Satz an Key-Value-Pairs. Die Keys beginnen hierbei mit einem Doppelpunkt.
Werden diese Frames von einem Server zum HUB oder vom HUB zu einem Server geschickt, werden sie Nachrichten genannt.

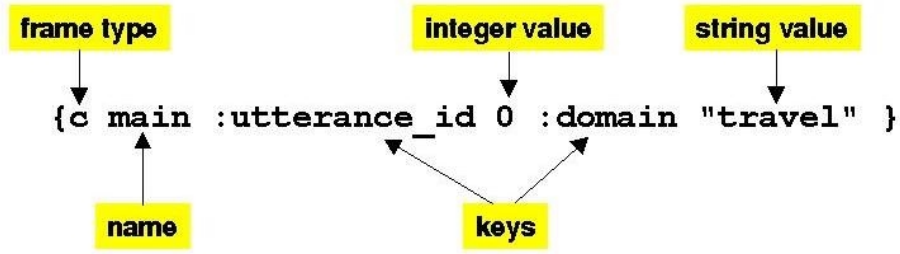

Abbildung 8: Kommunikationsframe in Galaxy

Wenn ein HUB eine Nachricht an einen Server schickt, ruft dieser eine sog. Dipatch-Funktion auf, die Teil der Nachricht ist. Diese Information wurde dem Server vorher beim Startskript mitgegeben (siehe Abbildung 9)

Die Bezeichnung Recognizer.Recognize im Name-Slot der Nachricht bezeichnet also den Speech-Recognizer als Empfänger der Nachricht und die Funktion Recognize als aufzurufende dispatch-Funktion.

Auch Server können Nachrichten an den HUB senden. Hierzu spezifizieren sie in ihrer Nachricht eine Funktion (oder Regel) die der HUB nach Erreichen der Nachricht ausführen kann. Findet der HUB diese Funktion nicht in seiner eigenen Datenbasis, so sucht er einen Server der diese Funktion unterstützt. Gibt es auch einen solchen Server nicht, wird die Nachricht ignoriert (siehe Abbildung 10).

Hier sendet also der Audio-Server eine Nachricht an den HUB mit dem Funktionsaufruf „FromAudio“. Der HUB findet eine entsprechende Regeln in seiner Datenbasis, die ihn anweist eine Nachricht an den Recognizer-Server zu der dispatch-Funktion Recognize zu senden.
Weitere Regeln und Erläuterungen zu der Galaxy Nachrichtenstruktur sind unter [Galaxy_Documentation1_2003] zu finden.

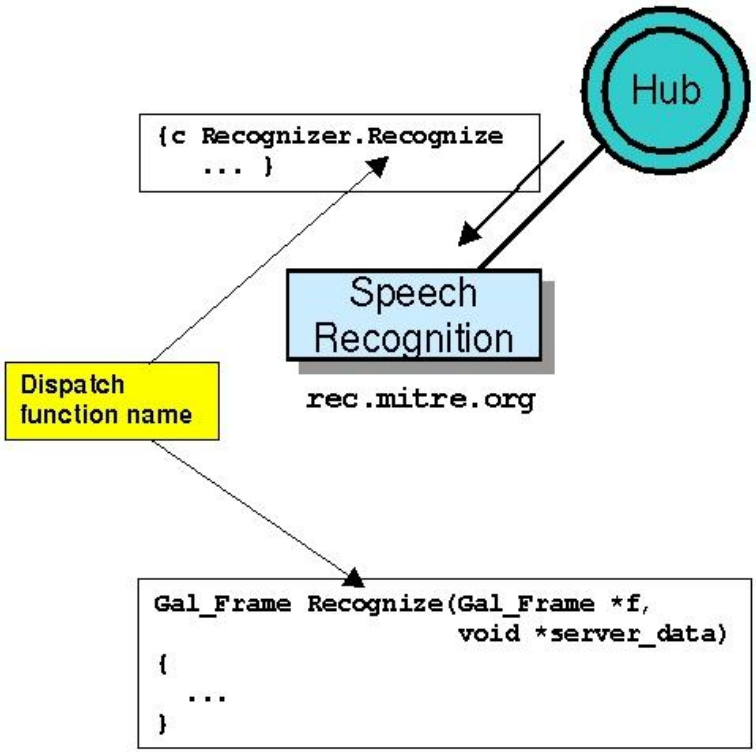

Abbildung 9: Nachricht von Hub an Server in Galaxy

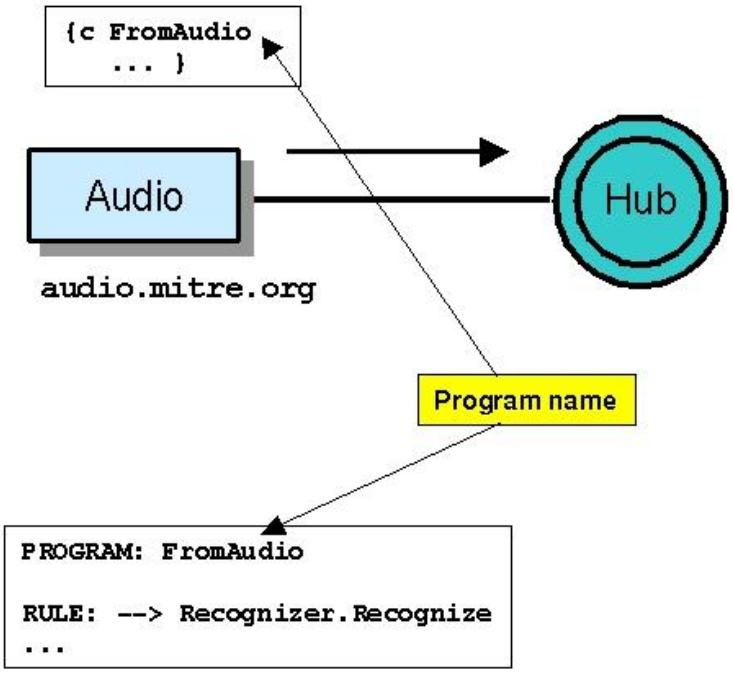

Abbildung 10: Nachricht von Server an Hub in Galaxy.

3.4.3. Multimodalität in Galaxy

Die Galaxy-Infrastruktur unterstützt Variationen der Multimodalität bezüglich der Interaktion mit graphischen Userinterfaces und Sprache. Die Interaktion mit dem System ist sowohl nur in Sprache, nur mittels graphischen Userinterfaces als auch in einer Kombination daraus möglich.
So sind z.B. Interaktionen wie „Ruf mit unter der folgenden Nummer an“ und ein Mausklick auf die Telefonnummer möglich. Zeitliche Korrelationen werden hierbei nicht berücksichtigt.

3.4.4. Realisierte Anwendungen

Mehrere verschiedene Anwendungen wurden bereits mit Galaxy realisiert. Diese sollen hier nur kurz aufgezählt werden:

- Jupiter: weltweite Wetterinformationen
- Pegasus: Abflug-, Ankunftsinformationen über Flüge der wichtigsten US-Städte
- Voyager: Assistenz bei der Navigation und über das Verkehrsgeschehen in Boston

Die Anwendungen sind in der Sprache Englisch realisiert. Japanisch und Mandarin sei während der Projektlaufzeit auch realisiert worden.

3.4.5. Das Open-Source Packet

Unter [Galaxy@SourceForge2003] kann eine Distribution des Galaxy Open Source Toolkits (OSTK) heruntergeladen werden.
Für die folgenden Betriebssysteme ist eine Version verfügbar:

- Sparc Solaris version 2.7 (sparc-solaris)
- Red Hat Linux for x86 6.2 (x86-linux)
- Win32 (x86-nt) (NT 4.0; also tested on Windows 2000 Professional)

Und für die Folgenden verfügbar, aber laut Webseite nicht ausgetestet:

- SGI IRIX 32- and 64-bit (mips-irix, mips64-irix)
- FreeBSD 4.3 (x86-FreeBSD)
- MacOS X 10.1/Darwin 1.4 (powerpc-Darwin)

Verfügbar gemacht wurden:

- Ein HUB, der in C implementiert wurde, der die Verbindungen zwischen den Servern (die im herkömmlichen Sinne ja Clienten sind) ermöglicht
- Server Library, um Galaxy-kompatible Server in C, C++, Java, Phytion and Allegro Common LISP zu implementieren
- Beispiele, die grundlegende und fortgeschrittene Funktionalitäten aufzeigen, um eigene Server zu programmieren und den HUB zu konfigurieren
- Dokumentationen

Es gibt aber auch vieles, was nicht verfügbar gemacht wurde:
Das komplette End-to-End-Kommunikationssystem wurde nicht verfügbar gemacht, Teile daraus sind verfügbar, andere muss man sich, ja nach Bedarf, selber erstellen.
Während Galaxy Communicator die unteren Ebenen zum Nachrichtentransport bereitstellt, werden aber keine semantischen Standards mitgeliefert. Es wurde kein Standard für Spracherkenner, oder Ausgabegeräte innerhalb einer API realisiert, so dass dieser bei der Benutzung von Galaxy zusätzlich zu entwickeln ist. Das gleiche gilt für die Konfiguration der entsprechenden Server. Auch hierfür wird kein Standard mitgeliefert.

3.4.6. Zusammenfassung / Einschätzung

Gleichwie einige Eigenschaften von Galaxy schon beschrieben wurden, sollen hier doch noch ein paar grundsätzliche Eindrücke geschildert werden:

- Galaxy scheint eine „gewöhnliche“ Client-Server-Struktur zu haben (hier jedoch Server-HUB genannt) in der Teile der sonst verwendeten Yellow-Pages als Regeln in den HUB gewandert sind.
- Galaxy scheint sehr statisch zu sein. Zum Zeitpunkt des Starts des HUBs müssen alle Server (Clients !!) schon gestartet sein und deren Eigenschaften in Konfigurationsdateien vorhanden sein.
- Die Dokumentation ist sehr ausführlich, leider an vielen (konkurrierenden) Stellen (siehe Literaturliste) zu finden, wiederholt sich öfters
- Die frei verfügbare Software stellt hohe Ansprüche sowohl an die bereits vorhandene Software, als auch and das Konfigurationsverständnis des Benutzers. Zudem ist sie nicht vollständig. Man ist so darauf angewiesen, selber Server zu implementieren.
- Zudem liegen für die in Galaxy realisierten Anwendungen die Kommunikationsschnittstellen nicht offen, so dass Galaxy eigentlich zu einem regelgestützten Server und einer API für Clienten (auf Java oder C-Basis) limitiert ist.
- Unter [Galaxy_Toolkit2003] sind einige Implementationen von Servern verfügbar, teilweise mit Sourcen, teilweise ohne und teilweise ohne Dokumentation.

Die Galaxy Communicator Infrastructure ist stark unter dem Aspekt der Sprachanwendung gewachsen. In diesem Sinne kann hier sogar von einer Topologie gesprochen werden, da unterschiedliche Clienten mit unterschiedlichen Aufgabenbereichen einer gewissen Position der Architektur zugeordnet werden können. Jedoch ist der Router stark regelbasiert und schon aufgrund dieser Entscheidung schwer erweiterbar oder gar dynamisierbar. Dies zeigt schon die Tatsache, dass verschiedenste Server implementiert wurden.

Für DynAMITE ist Galaxy jedoch eine starke Referenz. Kann ein System, welches selbstorganisiert ist und keine ontologischen (anwendungsbezogenen) Regeln zentralisiert vorhalten soll, einen ähnlichen Leistungsumfang bereitstellen, wie die verschiedenen Realisationen, die mit Galaxy bisher erreicht wurden ? Aus diesem Grund sollte der Fortschritt von Galaxy weiterbeobachtet werden.

Literaturreferenzen zu [Kapitel 3.4]

[SLS2001] The Spoken Language Systems Group: http://www.sls.csail.mit.edu/sls/sls-orange-nospec.html

[Galaxy2001] Galaxy Overview: http://www.sls.csail.mit.edu/sls/technologies/galaxy.shtml

[Galaxy@SourceForge2003] Galaxy Communicator auf SourceForge: http://communicator.sourceforge.net/

[Galaxy_Publications2003] Publikationen von Galaxy: http://www.sls.csail.mit.edu/sls/publications/

[Galaxy_Summary2003] 2003 Research Summary:
http://www.sls.csail.mit.edu/sls/publications/2003/ResSum2003.pdf

[Galaxy_Documentation1_2003] Dokumentation von Galaxy auf den SourceForge-Seiten
http://communicator.sourceforge.net/sites/MITRE/distributions/GalaxyCommunicator/docs/manual/index.html

[Galaxy_Documentation2_2003] Dokumentation von Galaxy auf den SLS-Seiten
http://www.sls.csail.mit.edu/documentation/galaxy/index.html oder
http://www.sls.csail.mit.edu/documentation/galaxy/tutorial.pdf

[Galaxy_Toolkit2003] Available Components of the Toolkit
http://communicator.sourceforge.net/sites/MITRE/distributions/OSTK-20021004/index.html

3.5. MetaGlue

Das System MetaGlue wurde im Rahmen des Projektes „Intelligent Room“ des MIT Artificial Intelligence Laboratory entwickelt. MetaGlue erweitert die Programmiersprache Java um eine kleine Menge von Primitiven, die die Programmierung von Agenten erleichtern [>CPWW99]. Eine neue Klasse namens „Agent“ wurde eingeführt. Ein Postcompiler wird verwendet, um aus kompilierten Java-Classfiles Metaglue-Agenten zu generieren, die auf einer sogenannten MetaGlue Virtual Machine (MVM) ablaufen. Im Folgenden werden die wesentlichen Funktionalitäten des MetaGlue-Systems, die im DynAMITE-Kontext relevant sind, vorgestellt:

- Kontext/Konfigurationsmanagement: Eine integrierte verteilte SQL-Datenbank wird verwendet, um modifizierbare Parameter der Agenten (z.B. ihre Ausführungsorte) und ihre persistenten Zustände zu speichern. Der Einsatz einer Datenbank erlaubt zum einen schnellere Zugriffe im Vergleich zu Java-Filezugriffen und zum anderen die Vereinfachung der Zugriffe auf Daten migrierender Agenten. MetaGlue verfügt über eine webbasierte Schnittstelle zur Modifizierung der Parameter, die auch während der Ausführung eines Agenten möglich ist.
- Agentenkonfiguration: Mittels der Primitive *tiedTo()* spezifiziert ein Programmierer die Hardware- und Softwareanforderungen zur Ausführung eines Agenten, die das System erfüllen muss. Falls ein Agent auf einer Maschine gestartet wird, die diesen Anforderungen nicht genügt, verschiebt das MetaGlue-System die Ausführung der Agenten automatisch zu einer alternative Maschine. Die angegebenen Anforderungen dienen auch bei einem Restart der Agenten (siehe unten).
- Verbindungen zwischen den Agenten: Mittels der Primitive *reliesOn()* wird eine Verbindung zu einem anderen Agenten, der ein bestimmten Dienst anbietet, hergestellt. Das System liefert eine Referenz auf den dienstanbietenden Agenten zurück. Da die Agenten mittels ihren Diensten statt mittels ihren Namen miteinander verbunden sind, können neue Agenten, die den gleichen Dienst anbieten, in das System hinzugefügt werden, ohne die Agenten modifizieren zu müssen, die diesen Dienst in Anspruch nehmen. Die Abhängigkeit zwischen den Agenten ist transitiv. Das Starten eines Agenten kann aufgrund der Abhängigkeitskette zur Ausführung einer gesamten Applikation führen.
- Laufzeitzustände der Agenten: Zustände eines Agenten werden benötigt, um ihn zur Laufzeit anhalten (z.B. für die Modifizierung bzw. Debugging) und anschließend wieder integrieren zu können. Mittels der Primitiven *freeze()* und *defrost()* können Laufzeitzustände eines Agenten in die SQL-Datenbank eingefroren bzw. wieder ausgelesen werden.
- Modifizierung eines laufenden Systems: Eine Änderung einzelner Komponente zur Laufzeit soll nicht dazu führen, dass das gesamte System heruntergefahren und anschließend neu gestartet werden muss. Das MetaGlue System verfügt dafür über folgende Mechanismen:
    - Falls ein Agent angehalten wird (beispielsweise fürs Debugging), können die Agenten, die von ihm abhängig sind, entweder darauf warten, bis er wieder lauffähig ist, oder temporär einen anderen Agenten aussuchen, der den gleichen Dienst anbietet.
    - Beim Wiederstarten eines Agenten werden mittels der Primitive *defrost()* seinen eingefrorenen Zustand aus der SQL-Datenbank herausgeholt. Der Agent behandelt zunächst die offenen Anforderungen, die während seiner Abwesenheit eingetroffen sind.
    - Im Fall eines unerwarteten Hardware- oder Softwarefehlers kann die Zustandsinformation ggf. nicht rechtzeitig vollständig eingefroren werden. Das MetaGlue-System leitet dieses Problem an menschliche Benutzer, die es dann beheben.
- Ressourcenmanagement: Ressourcen sind nicht in beliebigem Umfang verfügbar sein. Falls mehrere Anwendungen gleichzeitig auf die gleichen Ressourcen (z.B. Display oder Mikrophone eines Gerätes) zugreifen, entstehen Konflikte. Das Ressourcenmanagement des MetaGlue-Systems erlaubt den Agenten, die Ressourcen auf einem hohen Abstraktionsniveau (statt auf einem maschinennahen Niveau) anzufordern. Es bietet eine Menge hierarchisch angeordneter und vom Programmierer erweiterbarer *Dealer-Agenten* an, die dafür zuständig sind, die Ressourcen zu verteilen. Ein Dealer-Agent kann nicht

nur die Ressourcen verteilen. Er kann auch eine verteilte Ressource zurück verlangen, falls sie von einem Agenten mit einer höheren Priorität angefordert wird.

Event-Mechanismen: MetaGlue-Agenten können sich bei anderen Agenten oder dem MetaGlue-System für bestimmte Ereignisse registrieren. Beim Eintreffen der Ereignisse werden die registrierten Agenten mittels Nachrichten informiert. Ereignis-Broadcasting wird verwendet, um eine Gruppe von Agenten beispielsweise über die Kontextänderung zu informieren.

Literaturreferenzen zu [Kapitel 3.5]

[>CPWW99] Michael Coen, Brenton Phillips, Nimrod Warshawsky, Luke Weisman, Stephen Peters, Peter Finin. Meeting the Computational Needs of Intelligent Environments: The Metaglue System. Proceedings of the first International Workshop on Managing Interactions in Smart Environments (MANSE'99), 1999.

4. (Multimedia-) Middleware

4.1. Jini

Jini [>SUN1999, >COLLAB1999] ist eine Technologie der Firma SUN (auf Basis der Java 2 Plattform), die den Aufbau von Netzwerken durch Bereitstellung einer eigenen Infrastruktur fördern soll. Die Jini Technologie reiht sich in eine Reihe anderer verteilter Systemarchitekturen, wie zum Beispiel CORBA oder DCOM ein. Sie grenzt sich von diesen Architekturen durch einige Eigenschaften ab, die sie durch ihre Java Basis hat. Jini wurde 1999 erstmalig veröffentlicht und stellt Schnittstellen und Protokolle zur Verfügung, die die verteilte Programmierung in Netzwerken unterstützt.

Die Elemente dieser Netzwerke sind Software-Dienste und Hardware-Geräte oder beides zusammen, die spontan zu sog. Gemeinschaften (federations) zusammengefasst werden können. Die Hardware-Elemente müssen nicht zwangsläufig vollwertige PCs sein, es kann sich durchaus auch um kleine, weniger leistungsstarke Geräte (wie z.B. Digitalkameras) handeln. Außerdem stellt Jini eine Art Selbstheilungs-Mechanismus bereit, der das sichere Entfernen von Geräten aus solchen Gemeinschaften handhabt. Jegliche Software- und Hardware-Dienste werden durch Java-Objekte repräsentiert und das Auffinden dieser sowie der Zugriff auf diese geschieht über Java-Interfaces. Ein wesentliches Merkmal dabei ist, dass Java Programmcode über das Netzwerk übertragen wird und nicht ein festgelegtes Protokoll vereinbart werden musste. Für den Programmierer entsteht dadurch kein Bruch in der Entwicklung, da die gewohnte Java Programmierung nicht unterbrochen wird um zum Beispiel eine Abbildung von Programmzustand auf ein Netzwerkprotokoll zu realisieren.

4.1.1. Architektur

Die Jini Infrastruktur setzt das Vorhandensein einer Java 2 Plattform voraus. Dabei macht Jini insbesondere Gebrauch von der Serialisierung von Objekten und RMI (Remote Method Invocation). Das Ziel von Jini ist es, die Vorteile der objektorientierten Programmierung auf das Netzwerk zu übertragen. Anstatt ein Protokoll auf dem Netzwerk zu definieren, erlaubt es die Programmierung unter Verwendung von Objekt-Schnittstellen. Jini definiert eine Laufzeitumgebung, die es dem Programmierer erlaubt, Dienste dem Netzwerk hinzuzufügen, zu entfernen, diese zu suchen und zu verwenden.

Jini Services	▪ ***JavaSpaces*** ▪ ***Transaction Managers*** ▪ ***Printing, Storage, Databases,...***
Jini Infrastructure	▪ ***Discovery*** ▪ ***Lookup Service***
Jini Programming Model	▪ ***Leasing*** ▪ ***Distributed Events*** ▪ ***Transactions***
Java 2 Platform	▪ ***Java RMI*** ▪ ***Java VM***

Abbildung 11: Jini Architektur

Abbildung 11 zeigt, wie sich Jini in die Java 2 Architektur eingliedert. Auf der Jini Infrastruktur bauen die Jini Services auf. Die Infrastruktur von Jini (Abbildung 11) beschreibt den Kern von Jini selbst und umfasst im Wesentlichen ein verteiltes Sicherheitssystem basierend auf Java RMI, das Discovery-/Join-Protokoll und den Lookup-Service.

Um einer Jini-Einheit (Dienst oder Anwendung) die Teilnahme an einem Netzwerk zu ermöglichen, muss diese zunächst eine Jini-Gemeinschaft finden, die diese aufnehmen kann (Discovery). Als Ergebnis einer Suche nach einer Jini-Gemeinschaft erhält eine Einheit eine oder mehrere Referenzen auf verfügbare Lookup-Services. Die Anmeldung an einem oder mehreren dieser Lookup-Services wird als Join bezeichnet.

Sobald ein oder mehrere Lookup-Services gefunden wurden, kann ein Client damit beginnen, Dienste der Jini Gemeinschaft in Anspruch zu nehmen. Dazu übergibt er eine Liste an gesuchten Dienstschnittstellen an eine Suchmethode der Lookup-Services und erhält eine Menge an Dienstobjekten, die der Anfrage entsprechen.

Diese Dienst-Objekte wurden über das Netzwerk von den Diensten geladen und enthalten allen notwendigen Programmcode, um die Dienste anzusprechen.

4.1.2. Programmiermodell

Wie bereits erwähnt ist es ein Merkmal von Jini, das Konzept der objektorientierten Programmierung auf das Netzwerk auszudehnen. Um dies zu erreichen, trennt Jini die Schnittstelle eines Dienstes von seiner Implementierung. Ein Client sucht beispielsweise nach einer bestimmten Dienstschnittstelle und erhält mehrere Implementierungen dieser Schnittstellen als Ergebnis seiner Anfrage. Diese Dienst-Objekte können nun entweder die volle Funktionalität des Dienstes beinhalten oder reine „Proxies“ für den Netzwerkdienst darstellen. Die Kommunikation des Dienst-Objektes mit dem eigentlichen Dienst ist eine private Angelegenheit der beiden. Die kann im Jini Sinne mittels RMI geschehen, aber auch jede andere Form der Kommunikation ist hierbei denkbar (Webservice, CORBA, DCOM, usw.).

Daher können verschiedene Dienst-Objekte desselben Interfaces vollkommen verschiedene interne Implementierungen besitzen. Diese Art des Dienstzugriffs erlaubt es einem Client ohne jegliche Installation von Treibern mit verschiedensten Geräten und Diensten in Verbindung zu treten, da die Implementierung von dem Dienst selbst geliefert wird.

Ferner bietet Jini auch einige Funktionalitäten, um das Entwickeln verteilter Dienste zu vereinfachen. Hierzu zählt auch die Behandlung von Ausfällen und Fehlern wie das Versagen oder Entfernen eines Gerätes. Ein „Leasing“ Mechanismus erzwingt die regelmäßige Erneuerung einer Dienstnutzung.

Um Java-Objekte (insbesondere Jini-Dienste) über externe Zustandsänderungen in einem Netzwerk zu informieren, werden Remote-Ereignisse verwendet. Dieses ist eine Erweiterung der Java-Standard-Ereignisse, welche den Anforderungen in verteilten Systemen gerecht werden sollen. Bei verteilten Systemen können erheblich mehr Fehler und Schwierigkeiten auftreten als bei lokalen Systemen (Verzögerung, Vertauschung der Reihenfolge, Verlust von Events im Netzwerk). Das Jini-Ereignismodell stellt Werkzeuge bereit, um diese Probleme zu handhaben.

Um das System nicht in einem möglicherweise inkonsistenten Zustand zu belassen, bietet Jini auch eine Transaktionsschnittstelle an, die das Ausführen verteilter Transaktionen anbieten. Eine Transaktion ist die Gruppierung von mehreren zusammenhängenden Operationen, deren Ausführung lediglich zwei Resultate zulassen: entweder werden alle Operationen erfolgreich ausgeführt oder alle schlagen fehl. Jini verwendet bei Transaktionen das sogenannte Two-Phase-Commit Protokoll.

4.1.3. Jini Dienste

Jini Dienste verwenden das Jini Programmiermodell und die Jini Infrastruktur, um sich gegenseitig bekannt zu machen, sich zu finden und schlussendlich Methoden aufzurufen. Dienste erscheinen als normale Java Objekte und können wiederum aus einer Vielzahl anderer Java Klassen bestehen. Ein Dienst implementiert eine bestimmte Schnittstelle, die die Operationen des Dienstes definiert. Ein Beispiel für einen Jini Dienst ist ein Druckservice, der eine bestimmte Druckerhardware kapselt.

Literaturreferenzen zu [Kapitel 4.1]

[>COLLAB1999] Jini Community, http://www.jini.org

[>SUN1999] Sun Microsystems, http://www.sun.com/software/jini

4.2. UPnP

4.2.1. Idee

Die Technologie "Universal Plug and Play" (UPnP, [>UPNP2000]) wurde von Microsoft mit dem Ziel begründet, das Konzept "Plug & Play" vom einzelnen PC auf Heimnetzwerke zu übertragen. Universal Plug and Play soll Benutzern die Installation und Bedienung vernetzter Geräte vereinfachen durch eine

- automatische Erkennung und möglichst konfigurationslose Installation
- in Verbindung mit der Interoperabilität und komfortablen Bedienung dieser Geräte.

UPnP zielt dabei auf den privaten Heimbereich und schließt neben Geräten aus dem Bereich

- Computer, wie z.B. PCs, Drucker, Scanner, Speichermedien,

auch Geräte aus dem Bereich

- Audio/Video, wie z.B. TVs, DVD-, MP3, CD-Spieler, Boxen, Videokameras

und aus der

- Heimautomatisierung, wie z.B. Lampensteuerungen, Überwachungskameras, Uhren,

mit ein. Durch das Zusammenspiel der Geräte aus unterschiedlichen Anwendungsbereichen soll für den Benutzer ein deutlicher Mehrwert entstehen.

4.2.2. Technologie

4.2.2.1. Komponenten

UPnP unterscheidet grundsätzlich drei Arten von Komponenten ([>UPNP2003], [>MNTW2001], siehe Abbildung 12):

1. **Device** (Gerät): Geräte sind Container für Services (s.u.) und eingebettete Geräte. Alle Eigenschaften und Services sind in standardisierten Gerätebeschreibungen in XML beschrieben.
2. **Service**: Ein Service bietet Benutzern den Zugriff auf eine Gerätefunktionalität. Er modelliert seinen Zustand über Statusvariablen und bietet Aktionen an, um den Zustand zu beeinflussen und abzufragen. Aktionen und Statusvariablen sind in Servicebeschreibungen standardisiert.
3. **Control-Point** (CP): Ein CP steuert Geräte über Services entweder automatisch oder bietet einem Anwender die Möglichkeit der Steuerung.

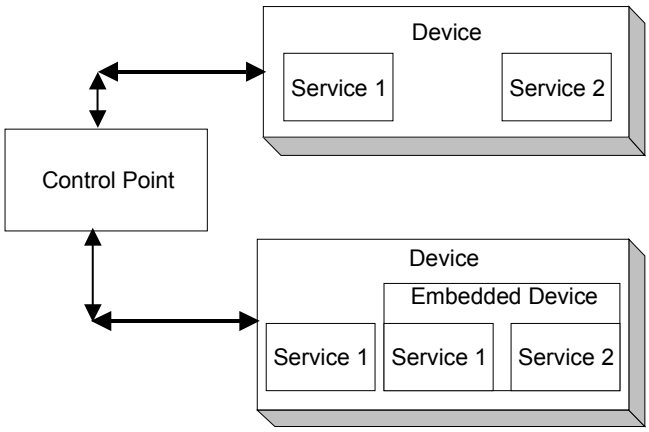

Abbildung 12 UPnP-Komponenten: Control-Points, Devices und Services.

4.2.2.2. Protokolle

UPnP basiert auf standardisierten TCP/IP Protokollen. Abbildung 13 zeigt die Protokolle. Zwischen verschiedenen Geräten werden nur Daten und keine Programme ausgetauscht. Dadurch ist UPnP, im Gegensatz z.B. zu HAVi (siehe Kapitel 0), unabhängig von der Programmiersprache.

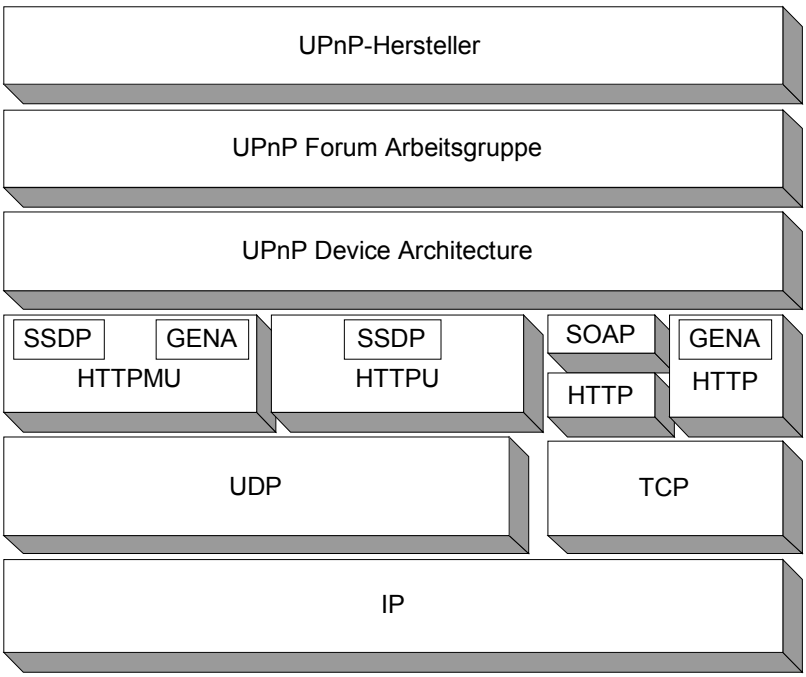

Abbildung 13 UPnP-Protokolle

SSDP (Simple Service Discovery Protocol)

SSDP ([>IETF 3]) basiert auf HTTPU (Unicast) und HTTPMU (Multicast) ([>IETF99])und wird in UPnP für das Auffinden von Devices und Services benutzt.

GENA (Generic Event Notification Architecture)

GENA ([>IETF2000a]) definiert ein Konzept über Subscriber und Publisher von Ereignissen (engl. Events). GENA Formate werden innerhalb von UPnP für die SSDP „Presence Announcements“ (über HTTPMU) und die Signalisierung von Zustandsänderungen (über HTTP, [>RFC 2616]) benutzt.

SOAP (Simple Object Access Protocol)

SOAP ([>W3C2001]) definiert die Benutzung von XML und HTTP für RPCs.
UPnP benutzt SOAP für Kontrollnachrichten und entsprechende Antwortnachrichten. Die Kontrollnachricht enthält die auszuführende Aktion mit den zugehörigen Parametern. Die Antwortnachricht enthält den Status, den Rückgabewert und eventuelle Rückgabeparameter.

4.2.2.3. UPnP Device Architecture

Beim Betrieb einer UPnP-Komponente wird zwischen folgenden fünf Phasen unterschieden ([>UPNP2003]):

1. **Adressierung**: Jedes Gerät muss über einen DHCP-Client verfügen. Nach dem Start beziehen das Gerät über DHCP ([>RFC 2131]) eine IP-Adresse. Falls im Netz kein DHCP Server vorhanden ist, wird Auto-IP ([>IETF 1]) verwendet, d.h. vereinfacht, es wird willkürlich eine freie IP-Adresse im Bereich 169.254/16 benutzt.
2. **Discovery**: Je nachdem ob ein Gerät oder ein CP dem UPnP-Netzwerk beitritt, kann das Discovery über Advertisement oder Search stattfinden ([>IETF 3]).
   - Search: Ein neuer CP sucht nach dem Start über eine Broadcast-Adresse nach interessanten Geräten indem er ein SSDP request (HTTPMU) sendet. Er gibt dabei an, für welche Geräte oder Services er sich interessiert. Devices lauschen auf der standardisierten Multicast-Adresse und antworten mit SSDP-Nachrichten, falls eine Service oder eingebettetes Device auf die Beschreibung passt.
   - Advertisement: Geräte geben sich im Netz durch Broadcast-Nachrichten bekannt. Dazu senden sie in regelmäßigen Abständen SSDP-Discovery Nachrichten aus. In diesen Nachrichten geben Geräte sich selbst, jedes eingebettete Device sowie jeden Service bekannt. Diese Nachrichten haben den gleichen Inhalt wie die Antworten auf ein Search-Request. CPs lauschen auf der standardisierten Multicast-Adresse.

   Wenn ein Gerät das Netz verlässt, z.B. abgeschaltet wird, dann sollte es sich über SSDP abmelden.

3. **Description**: Hat ein CP während der Phase Discovery ein Gerät einer interessanten Kategorie gefunden, dann kann er anschließend genauere Geräte- und Service-Beschreibungen über HTTP abrufen. Dabei werden vom UPnP-Forum genormte XML-Dokumente ausgetauscht. Der CP fragt zunächst die Gerätebeschreibung über HTTP von einer URL ab, die er während der „Discovery“ ermittelt hat. In dieser Gerätebeschreibung sind wiederum URLs für die Servicebeschreibungen enthalten. Diese können anschließend auf die gleiche Weise über HTTP abgefragt werden.
   Gerätebeschreibungen enthalten herstellerspezifische Informationen, Definitionen der eingebetteten Geräte, URLs für HTML-Präsentationen ([>W3C99]), eine Aufzählung aller Services mit den zugehörigen URLs für die Steuerung und Events. Hersteller können durch UPnP-Arbeitsgruppen standardisierte Beschreibungen erweitern. Die Hauptarbeit des UPnP-Forums besteht in der Standardisierung verschiedener Gerätebeschreibungen um eine weitreichende Interoperabilität zu gewährleisten.
   Alle Beschreibungen sind versioniert mit Haupt- und Unter-Version. Eine neue Unter-Version muss ein kompatibles Superset der Vorgängerversion sein.
4. **Control**: Diese Phase ist unabhängig vom **Eventing** (nächster Abschnitt) und komplementär zur **Präsentation** (übernächster Abschnitt).
   Durch die Phase Description besitzt der CP das notwendige Wissen für die Steuerung eines Gerätes, d.h. die Informationen über die möglichen Kommandos, deren Parameter, die Zustandsvariablen inkl. Datentyp, Wertebereich und deren Ereignischarakteristik.
   Zur Steuerung wird das Protokoll SOAP verwendet ([>W3C2001]). CPs senden SOAP-Requests an eine bestimmte, in der Gerätebeschreibung enthaltene, URL. Als Antwort sendet das Gerät Rückgabewerte, Rückgabeparameter oder Fehlercodes. Es muss innerhalb von 30 Sekunden antworten, kann in der Antwort auch nur signalisieren, dass die Ausführung noch nicht beendet ist.
   UPnP definiert die Aktionsnamen, Parameternamen und Variablen in den Nachrichten. Die Werte sind herstellerspezifisch.
5. **Eventing** dient dazu, alle interessierten CPs ständig über den aktuellen Gerätezustand, genauer den Zustand eines Service, zu informieren. Dazu meldet ein CP sein Interesse für einen bestimmten Service an und wird anschließend eine herstellerspezifische Zeit lang über alle Änderungen spezieller, in der Service-Beschreibung festgelegter Statusvariablen benachrichtigt. CPs müssen die Anmeldung vor dem Ende der Gültigkeitsdauer erneuern.
6. **Präsentation**: Zusätzlich zu den beschriebenen Mechanismen Control und Eventing kann ein Gerät in der Gerätebeschreibung die URL einer HTML-Seite angeben. Diese HTML-Seite kann ein CP mittels HTTP in einen Browser laden und so einfach eine Gerätesteuerung über eine HTML-Bedienoberfläche ermöglichen. Mögliche Inhalte der HTML-Seite sind nicht näher spezifiziert, sondern völlig herstellerspezifisch. Definiert durch UPnP ist nur das Format HTML der Version 3.0 oder höher. Empfohlen wird die Unterstützung der Lokalisierung in HTTP.

4.2.3. Geräte- und Servicebeschreibungen

Für die verschiedenen Anwendungsbereiche entwickeln Arbeitsgruppen Geräteprofile, die von UPnP-Geräten erfüllt werden müssen. Diese Arbeitsgruppen spezifizieren neben den Geräteprofilen, Services, Statusvariablen und Aktionen auch deren Semantik. Es existieren inzwischen unter anderem entsprechende Standards für Drucker, Scanner, Router sowie Heizungs- und Klimaanlagen. Von besonderem Interesse für das Projekt DynAMITE sind die Spezifikationen aus dem Bereich Audio und Video (AV).

4.2.3.1. UPnP AV-Spezifiaktionen

Die speziellen UPnP AV-Spezifiaktionen bestehen aus einer allgemeinen Architektur ([>RK2002]) sowie verschiedenen Service- und Geräte-Spezifikationen. Diese Spezifikationen zielen speziell auf den Markt der Unterhaltungselektronik-Geräte. Darunter versteht das UPnP-Forum allgemein alle Geräte die Audio- oder Video-Inhalte verarbeiten, d.h. neben TVs, DVD-Spielern oder HiFi-Anlagen auch PCs. Die entwickelte Architektur unterscheidet zwischen sogenannten AV-Medien-Servern ([>R2002a]) und AV-Medien-Renderern ([>R2002b]). So wird ein herkömmlicher TV in UPnP als eine Kombination dieser Funktionalitäten angesehen. Der TV-Tuner ist ein AV-Medien-Server und das Display und die Boxen bilden den AV-Medien-Renderer. Zusätzlich kann ein TV einen Control-Point zur Steuerung fremder Geräte oder eigener Funktionalitäten beinhalten.
Weitere Beispiele für AV-Medien-Server sind Digital-/Videokameras, DVD-/MP3-Spieler und Festplattenspeicher. Musikanlagen und Monitore stellen AV-Medien-Renderer dar.

Die Spezifikationen beschreiben die Steuerung und Interaktionen der Geräte. So soll die AV-Architektur den Netzwerk-transparenten Zugriff von allen CPs auf die Inhalte aller AV-Medien-Server ermöglichen, um diese Inhalte auf möglichst vielen AV-Medien-Renderern im Heimbereich anzeigen zu können.

Die Protokolle für die Übertragung und das Format von Nutzdaten wie Filme, Musikstücke oder Fotos sind jedoch nicht Gegenstand von UPnP. Wenn z.B. ein Gerät nur das Protokoll HTTP zur Datenübertragung unterstützt, dann kann es mit einem anderen Gerät, das nur das Protokoll IEEE1394 unterstützt, nicht kooperieren.

4.2.4. Marktstellung und Zukunft

UPnP konkurriert mit anderen Technologien für die Spontanvernetzung wie z.B. Jini von Sun Microsystems, Rendezvous von Apple oder HAVi. Für Endgeräte-Hersteller erweist es sich als besonderer Vorteil, dass UPnP im Gegensatz zu HAVi oder Jini keine Java VM voraussetzt. Endgeräte wie z.B. TVs oder Hifi-Anlagen verfügen i.A. über beschränkte Ressourcen, auf denen sich keine performanten Java-Anwendungen realisieren lassen. Im Gegensatz zu Rendezvous schließt UPnP die Gerätesteuerung mit ein. In Anwendungsgebieten, in denen schon zahlreiche und etablierte Steuerungsprotokolle existieren, fand das einfacher zu implementierende Rendezvous deshalb schon Anwendung. Beispiele sind hier der Bereich Drucken und Datenbanken. Allerdings muss durch die Unterstützung von UPnP durch den Monopolisten Microsoft auch in diesen Bereichen mit einer Verbreitung von UPnP gerechnet werden.
In Märkten ohne verbreitete Steuerungsprotokolle, insbesondere in dem Markt Heimmultimedia, ist UPnP dank der integrierten Gerätesteuerung besonders interessant. Rendezvous kann hier nur als Grundlage für proprietäre Lösungen dienen. Aus diesem Grund sind inzwischen alle namhaften Hersteller von Unterhaltungselektronik, wie z.B. Bang & Olufsen, Bose, Hitachi, LG, Loewe, Matsushita, Mitsubishi, Nokia, Onkyo, Panasonic, Phillips, Roxio, Sanyo, Samsung, Sharp, Sony, Thomson oder Toshiba dem UPnP-Forum beigetreten ([>UPNP2003b]).

Auch die DHWG (Digital Home Working Group) entwickelt Standards für kabellose und kabelgebundene Netzwerke von PCs, Unterhaltungselektronik und mobilen Geräten, wie z.B. Handys, die auf UPnP basieren. Es soll eine nahtlose geräteübergreifende Nutzung und Erzeugung von multimedialen Inhalten ermöglicht werden. Auch dieser Allianz sind alle großen Hersteller von Unterhaltungselektronik und viele bedeutende PC und Software-Firmen beigetreten, wie z.B. Microsoft, IBM, Fujitsu und Intel.

UPnP scheint sich zum Industriestandard in der Unterhaltungsindustrie zu entwickeln und muss daher in DynAMITE unbedingt berücksichtigt werden.

Literaturreferenzen zu [Kapitel 4.2]

[>IETF 1] IETF draft., Auto-IP - Automatically Choosing an IP Address in an Ad-Hoc IPv4 Network. http://search.ietf.org/internet-drafts/draft-ietf-dhc-ipv4-autoconfig-05.txt.

[>IETF2000a] GENA - General Event Notification Architecture. IETF draft, 2000, http://www.upnp.org/draft-cohen-gena-client-01.txt

[>IETF 3] IETF draft, SSDP- Simple Service Discovery Protocol. http://www.upnp.org/download/draft_cai_ssdp_v

[>IETF99] HTTPMU/HTTPU - HTTP Multicast over UDP, HTTP Unicast over UDP, IETF draft, November 1999, ftp.ics.uci.edu/pub/ietf/http/ draft-goland-http-udp-01.txt.

[>MNTW2001] B. Miller, T. Nixon, C. Tai, M. Wood, Home networking with universal plug and play, IEEE Communication Magazine, Vol. 39, No. 12, December 2001.

[>RFC 2616] RFC 2616, HTTP: Hypertext Transfer Protocol 1.1. IETF request for comments. http://search.ietf.org/rfc/rfc2616.txt?number=2616

[>RFC 2131] RFC 2131 Dynamic Host Configuration Protocol. IETF request for comments. http://search.ietf.org/rfc/rfc2131.txt?number=2131 1_03.txt

[>RK2002] J. Ritchie, T. Kuehnel, UPnP AV Architecture 0.83, Juni 2002, http://www.upnp.org/download/UPnPAvArchitecture%200.83.prtad.pdf

[>R2002a] J. Ritchie, Media Sever 1.0. Juni 2002, http://www.upnp.org/download/MediaServer%201.0.pdf

[>R2002b] J. Ritchie, Media Renderer 1.0. Juni 2002, http://www.upnp.org/download/MediaRenderer%201.0.pdf

[>UPNP2000] UPnP™ Forum, http://www.upnp.org

[>UPNP2003] Contributing Members of the UPnP™ Forum. UPnP™ Device Architecture1.0.1, Mai 2003, http://www.upnp.org/download/Clean%20UPnPDA101-20030506.doc

[>UPNP2003b] UPnP Forum, Membership List: http://www.upnp.org/membership/members.asp, Dezember 2003.
[>W3C2001] World Wide Web Consortium, Simple Object Access Protocol version 1.2, W3C Technical report, 2001. www.w3.org/TR/soap12/.
[>W3C99] HTML-HyperText Markup Language, W3C Recommendastion, Dezember 1999, http://www.w3.org/TR/html4

4.3. JXTA

Das JXTA Projekt [>JXTA2001] wurde im Mai 2001 von Sun Microsystems in einem *Open Source* Forschungsprojekt gestartet. Das Ziel des Projektes war es grundlegende Mängel und Schwächen der zurzeit existierenden oder zu entwickelnden Peer-to-Peer Systemen zu beheben. Das wesentliche Ziel dabei war es eine Plattform zu entwickeln, die insbesondere Interoperabilität, Plattformunabhängigkeit und Geräteunabhängigkeit unterstützt.

Interoperabilität: Die JXTA Technologie wurde entwickelt, um es miteinander verbundenen Peers zu ermöglichen, ihre Verbundpartner zu lokalisieren, mit ihnen zu kommunizieren und an gemeinsamen Aktivitäten ihres Peer-to-Peer Systems teilzunehmen, wie beispielsweise dem Dateiaustausch. Zusätzlich sollten sie sich gegenseitig Dienste über die Grenzen von Peer-to-Peer Systemen anbieten können.

Plattformunabhängigkeit: Die JXTA Technologie wurde entwickelt, um unabhängig von einer Programmiersprache, wie beispielsweise C oder JAVA, unabhängig von einem Betriebssystem, wie beispielsweise Microsoft Windows oder UNIX und unabhängig von einer Netzwerkplattform, wie beispielsweise TCP/IP oder Bluetooth zu sein.

Geräteunabhängigkeit: Die JXTA Technologie wurde entwickelt, um für jedes digitale Gerät, einschließlich Sensoren, mobile Telefone, drahtlose PDAs, PCs, usw. implementiert werden zu können und auf diesen zu laufen.

Die JXTA Technologie besteht aus einer Menge offener Protokolle, die es jedem verbundenen Gerät eines Netzwerkes möglich macht mit anderen verbunden Geräten zu kommunizieren und zu kooperieren. Die grundlegenden Konzepte von JXTA um Interoperabilität zu erreichen sind demnach genau spezifizierte Protokolle und Nachrichtenformate. Zusätzlich nutzt JXTA XML als Nachrichtenformat und Beschreibungsformat seiner Bausteine. Wegen der Einfachheit und allgemeinen Verfügbarkeit der XML Technologie können JXTA Anwendungen auf fast jeder Plattform, die XML Nachrichten generieren und parsen kann, entwickelt werden.

Die Unterstützung von Plattformunabhängigkeit erreicht die Technologie dadurch, dass die JXTA v2.0 Protokoll Spezifikation [>JXTA2003] die Protokolle und die grundsätzlichen Bausteine, aus denen die JXTA Technologie aufgebaut ist, definiert, sie aber keine konkrete Implementierung der Protokolle vorschreibt. JXTA liegt zurzeit unter anderem in einer Java Referenz Implementierung [>PLATFORM2003] von Sun Microsystems vor.

Ein JXTA Netzwerk besteht aus einer Menge miteinander verbundener Peers. Die JXTA Plattform unterstützt dabei unter anderem das Auffinden von Peers, die Selbstorganisation von Peers in Peergruppen, das Veröffentlichen und Lokalisieren von Ressourcen und die Kommunikation zwischen den Peers.

4.3.1. Kernbausteine der Plattform

Dieser Abschnitt beschreibt die wesentlichen Konzepte und die Bausteine aus denen die JXTA Technologie aufgebaut ist.

4.3.1.1. Peers

Ein JXTA Peer ist eine beliebige Netzwerkressource, die ein oder mehrere JXTA Protokolle realisiert. Insbesondere muss ein Peer nicht alle JXTA Protokolle verstehen. Ein Peer kann u.a. ein Prozess, ein Gerät oder ein Benutzer sein. Beispielsweise kann ein Peer ein Sensor, ein mobiles Telefon oder ein PDA sein. Jeder Peer arbeitet unabhängig und asynchron von allen anderen Peers und wird durch eine ID eindeutig identifiziert. Zur Kommunikation untereinander u.a. für den Austausch von Protokollnachrichten veröffentlichen

Peers eine oder mehrere Netzwerkschnittstellen sogenannte Peer Endpunkte (*peer endpoints*). Diese Endpunkte werden für den Aufbau von Punkt-zu-Punkt Verbindungen zwischen Peers verwendet. Die Netzwerkschnittstellen bieten Zugriff auf beliebige Protokolle wie beispielsweise TCP/IP, HTTP oder SMS. Peers müssen keine direkten Punkt-zu-Punkt Verbindungen haben, um miteinander kommunizieren zu können; JXTA unterstützt die Weiterleitung (*Routing*) von Nachrichten über zwischengeschaltete Peers. Ein wesentliches Merkmal eines Peers ist die Möglichkeit andere Peers spontan aufzufinden, um beispielsweise eine Peer Gruppe zu bilden.

4.3.1.2. Peergruppen (*Peer Groups*)

Eine Peergruppe ist eine Sammlung von Peers. Die Peergruppe ist ein zentrales Konzept von JXTA, denn Peergruppen unterstützen eine sinnvolle Aufteilung von Peers um miteinander zu kooperieren und zu kommunizieren. Die JXTA Spezifikation schreibt nicht vor wo, wann und warum eine Peergruppe gebildet wird. Die Spezifikation beschreibt nur wie eine Gruppe gebildet, veröffentlicht und gefunden wird. Peergruppen können somit von Peers dynamisch gebildet und wieder aufgelöst werden, um anwendungsspezifische Bedürfnisse zu erfüllen. Die Peers organisieren sich somit selbständig in Peergruppen und können gleichzeitig Mitglied mehrerer Gruppen sein. Jede Peergruppe wird durch eine eindeutige Peergruppen ID identifiziert. Grundsätzlich ist die erste Gruppe, die in einem JXTA Netzwerk gebildet wird die *NetPeerGroup.* Diese Gruppe fungiert als eine Art Obergruppe, der jeder Peer zunächst angehört.

4.3.1.3. Kanäle (*Pipes*)

JXTA Peers benutzen Kanäle für den Austausch von Nachrichten. Ein JXTA Kanal ist ein asynchroner unidirektionaler Nachrichtentransfermechanismus. Kanäle unterstützen den Transport beliebiger Daten; dies schließt auch binäre Daten, beliebige Zeichenketten oder Java Objekten ein. Da die Kanäle nur Daten in eine Richtung transportieren, werden zwei Endpunkttypen unterschieden. Ein Endpunkt ist die Quelle oder das Ziel eines beliebigen Datenelementes, welches über das Netzwerk übertragen wird. Es gibt Ausgabekanäle (*output pipes*) für das Senden von Nachrichten und Eingabekanäle (*input pipes*), die Nachrichten empfangen können. Die Kanalendpunkte werden zur Laufzeit dynamisch an Peer-Endpunkte gebunden. Dies impliziert, dass die JXTA Kanäle dynamisch verlegt werden können, d.h. an unterschiedliche Peers zu unterschiedlichen Zeiten gebunden sind oder nicht gebunden sind.

Kanäle sind virtuell und verbinden Peers miteinander, die nicht notwendigerweise eine physikalische Verbindung haben müssen. In diesem Fall werden ein oder mehrere zwischengeschaltete Peer-Endpunkte zur Weiterleitung von Nachrichten zwischen zwei Kanal Endpunkten verwendet.

Kanäle unterstützen die Kommunikation von Punkt-zu-Punkt oder die Propagierung von Nachrichten zu mehreren Peers. Ein Punkt-zu-Punkt Kanal verbindet genau zwei Peers miteinander. Ein Eingabekanal eines Peers erhält eine Nachricht von einem Ausgabekanal eines anderen. Im zweiten Fall wird ein Ausgabekanal mit mehren Eingabekanälen verbunden und die Nachricht wird propagiert. Diese Nachrichtenverteilung ist allerdings auf Mitglieder einer Gruppe beschränkt.

4.3.1.4. Nachrichten (*Messages*)

Nachrichten sind die Einheiten in JXTA, die über Kanäle zwischen Peers verschickt werden. Die JXTA Nachrichten können über asynchrone, unzuverlässige, unidirektionale Transportsysteme übertragen werden. Eine JXTA Nachricht wurde als Datagramm spezifiziert, welche aus einem Umschlag und Protokollkopfzeilen mit einem Hauptteil besteht. Der Umschlag enthält eine Kopfzeile, einen Auszug aus der Nachricht und optional Quell- und Zielendpunkte. Ein Endpunkt ist ein logisches Ziel in Form einer URI (*Uniform Resource Identifier*) jedes Netzwerktransports, welcher Daten übertragen kann. Die Endpunkte werden typischerweise vom Laufzeitsystem auf physikalische Adressen abgebildet. Jeder Hauptteil eines Protokolls enthält eine variable Anzahl von Bytes.

In der Java Referenzimplementierung ist eine Nachricht ein Objekt, welches zwischen Peers übertragen wird. Die Nachricht ist eine geordnete Abfolge von Nachrichtenelementen und ihres Inhalts. Ein Nachrichtenelement hat einen Namen und einen Typ. Der Inhalt der Nachrichtenelemente kann beliebig sein. Die JXTA Referenzimplementierung stellt zwei Nachrichtendarstellungen zur Verfügung: XML oder binär.

4.3.1.5. Visitenkarten (*Advertisements*)

Eine Visitenkarte ist eine sprachenneutrale Metadatenstruktur, welche durch ein XML Dokument dargestellt wird. JXTA Netzwerkressourcen wie beispielsweise Peers, Peergruppen oder Kanäle werden durch

Visitenkarten benannt, ihre Eigenschaften werden beschrieben und die Existenz der Ressourcen wird öffentlich gemacht. Peers finden Ressourcen durch die Suche nach den entsprechenden Visitenkarten. Beispielsweise kann ein Peer, der eine Visitenkarte eines anderen Peer gefunden hat, versuchen eine direkte Verbindung zu diesem Peer aufzubauen oder ein Peer mit der Visitenkarte einer Peergruppe kann Mitglied in der entsprechenden Gruppe werden. JXTA definiert eine grundlegende Menge an Typen von Visitenkarten. Diese sind mit Hilfe von XML Schemata in Subtypen erweiterbar.

4.3.1.6. Eindeutige Bezeichner (*Identifier*)

Peers, Peergruppen, Kanäle oder andere JXTA Ressourcen müssen im JXTA Netzwerk eindeutig identifizierbar sein, damit beispielsweise Peers über Netzwerkgrenzen hinweg oder die möglicherweise zeitweise nicht mit anderen Peers verbunden sind, trotzdem adressierbar durch andere Peers sind. Jede Netzwerkressource im JXTA Netzwerk wird mit einer eindeutigen ID versehen. Das JXTA Adressiermodel basiert auf einem einheitlichen und ortsunabhängigen logischen Adressiermodel. Die JXTA ID ist im URN (*Uniform Ressource Name*) Format und wird in eine Visitenkarte eingebettet. URN sind in Form einer URI und dienen als dauerhafte, orttunabhängige Bezeichner von Ressourcen. Die JXTA Referenzimplementierung nutzt generierbare zufällige 128-bit UUIDs, welche jeder Einheit erlauben ihre eigene ID zu generieren. Die lokale Eindeutigkeit der UUID ist leicht zu garantieren. Global eindeutig wird UUID in Verbindung mit beispielsweise einem Namen oder einer Netzwerkadresse.

4.3.2. JXTA Protokolle

Die JXTA Spezifikation enthält sechs Protokolle, die in diesem Abschnitt beschrieben werden. Zum besseren Verständnis werden die englischen Bezeichnungen für die Protokolle verwendet. Alle JXTA Protokolle basieren auf einem asynchronen Anfrage/Antwort Modell; In JXTA wird jedes Protokoll durch eine oder mehrere Nachrichten definiert, welche von den Teilnehmern am Protokoll ausgetauscht werden. Alle Protokolle sind unabhängig voneinander und können auch einzeln in P2P Systemen verwendet werden. Durch die Zusammenarbeit der Protokolle wird unter anderem das dynamische und transparente Auffinden von Ressourcen, die Organisation und die Kommunikation der Peers unterstützt.

4.3.2.1. Peer Discovery Protokoll (PDP)

Dieses Protokoll wird von Peers verwendet, um ihre eigenen Ressourcen wie beispielsweise Peers, Peergruppen und Kanäle zu veröffentlichen und um andere Ressourcen des JXTA Netzwerks zu finden. Das Auffinden von Ressourcen durch PDP ist gruppenspezifisch, d.h. jede Anfrage eines Peers einer Gruppe A auf eine Ressource wird nur an Mitglieder der Gruppe A weitergeleitet. Das PDP unterstützt lokale und entfernte Anfragen auf Ressourcen. Die Visitenkarten gefundener Ressourcen werden vom Peer lokal gespeichert und können jederzeit über das PDP lokal angefragt werden, das bedeutet, dass häufig angefragte Visitenkarten stark im JXTA Netzwerk repliziert werden. Das PDP nutzt für entfernte Anfragen das *Peer Resolver Protokoll*, welches nachfolgend erklärt wird. Das PDP arbeitet nicht deterministisch, d.h. es können mehrere Visitenkarten einer Ressource als Antwort auf die Suche erhalten werden.

4.3.2.2. Peer Resolver Protokoll (PRP)

Dieses Protokoll wird von den Peers verwendet, um beliebige (generische) Anfragen zu einem oder mehreren Peers zu senden und eine oder mehrere Antworten auf die Anfrage zu erhalten. Die Anfragen können an alle Peers einer Peergruppe oder zu einem spezifischen Peer innerhalb einer Gruppe gerichtet werden. Im Gegensatz zum PDP, das genutzt wird, um Anfragen auf spezifische vordefinierte Information zu stellen, können über dieses Protokoll beliebig definierte Nachrichten ausgetauscht werden. Das PRP ist unzuverlässig, d.h. es garantiert keine sichere Auslieferung der Nachrichten. JXTA stellt keine Unterstützung für Zuverlässigkeit beim Nachrichtenaustausch zur Verfügung.

4.3.2.3. Peer Information Protokoll (PIP)

Dieses Protokoll wird von den Peers verwendet, um Statusinformationen wie beispielsweise Lebensdauer, Zustand und Möglichkeiten, usw. von anderen Peers zu erhalten.

4.3.2.4. Pipe Binding Protokoll (PBP)

Dieses Protokoll wird von den Peers verwendet, um virtuelle Kommunikationskanäle zwischen einem oder mehreren Peers aufzubauen. Voraussetzung für die Kommunikation ist die Bindung einer oder mehrerer Kanalendpunkte an physikalische Peer-Endpunkte. Das PBP spezifiziert diesen Prozess. Das PBP nutzt das PRP, um die protokollspezifischen Nachrichten auszutauschen. Durch die Verwendung des *Endpoint Routing*

Protokolls (nachfolgend erklärt), bietet das PBP die Möglichkeit verschiedene Netzwerkschnittstellen wie beispielsweise zu TCP/IP oder HTTP an die Kanäle zu binden.

4.3.2.5. Endpoint Routing Protokoll (ERP)

Dieses Protokoll wird von Peers dazu verwendet, Wege zu Netzwerkschnittstellen anderer Peers zu finden, um diesen eine Nachricht zu senden. Möchte beispielsweise Peer A Peer C eine Nachricht senden und es gibt keine direkte Verbindung zwischen beiden Peers, wie beispielsweise wenn sie nicht dasselbe Netzwerktransportprotokoll verwenden oder durch eine Firewall getrennt sind, müssen zwischengeschaltete Peers gefunden werden, welche die Nachricht von A nach C weiterleiten. Mit Hilfe des ERPs kann die von Peer A angefragte Weginformation gefunden werden. Ändert sich die Netzwerktopologie und der Weg nach C ist nicht mehr vorhanden, können Peers das ERP verwenden, um einen neuen Weg nach C zu finden. Das ERP verwaltet Information über die Topologie des JXTA Netzwerks und bestimmt die gewünschte Weginformation. Diese setzt sich aus der Peer ID des nach dem Weg suchenden Peers, der Peer ID des Zielpeers, einem Lebenszeitparameter (TTL) der Route und einer geordnete Liste von Peer IDs, die Wegpunkte auf dem Weg der Nachricht von der Quelle zum Ziel darstellen.

4.3.2.6. Rendezvous Protokoll (RVP)

Das RVP ist dafür verantwortlich, dass Nachrichten an alle Mitglieder einer Peergruppe gesendet werden. Sind alle Mitglieder einer Gruppe aus dem gleichen lokalen Netzwerk können ,,einfache“ Mechanismen wie beispielsweise Multicast verwendet werden. In dem Szenario einer Gruppe von Peers aus verschiedenen Netzwerken nutzt das RVP einen spezifischen Peer der Gruppe, den Rendezvous Peer, um die Nachrichten an alle Mitglieder der Gruppe weiterzuleiten. Der Rendezvous Peer nimmt alle Nachrichten von Peers aus der Gruppe an und leitet sie an die anderen Gruppenmitglieder innerhalb und außerhalb des Netzwerks weiter. Dieser nutzt dazu alle verfügbaren Möglichkeiten, wie beispielsweise Unicast, Broadcast und Multicast. Voraussetzung dazu ist, dass ein Peer der Gruppe die Funktion des Rendezvous Peers übernimmt und alle anderen Mitglieder der Gruppe sich mit diesem bekannt machen. Das RVP wird vom PRP und dem PBP genutzt, um Nachrichten zu propagieren.

Literaturreferenzen zu [Kapitel 4.3]

[>JXTA2001] Projekt JXTA, http://www.jxta.org

[>PLATFORM2003] JXTA platform infrastructure and protocols for Java 2 SE, http://platform.jxta.org/servlets/ProjectHome

4.4. HAVi

4.4.1. Überblick

HAVi [1] ist die Abkürzung für Home Audio Video interoperability. Hinter diesem Begriff verbirgt sich eine Initiative von führenden CE-Geräteherstellern, die sich zum Ziel gesetzt hat, einen zukunftssicheren De-facto-Standard für die Vernetzung und Kommunikation von Geräten auf Basis von IEEE 1394 [2] und IEC 61883 [3] in der Unterhaltungselektronik zu schaffen.
HAVi ist eine offene, skalierbare, plattform-unabhängige und programmiersprachen-neutrale Middleware, mit anderen Worten, HAVi kann in irgendeiner beliebigen Programmiersprache und auf ein x-beliebiges Betriebssystem implementiert werden.
Die HAVi-Architektur spezifiziert einen Satz von APIs (Application Programming Interfaces), die den Geräteherstellern die Entwicklung von interoperablen HAVi-Geräten ermöglicht, und zusätzlich Dritt-Entwicklern („third parties developer“) die Erstellung von Java-Applikationen (z.B. Benutzeroberflächen) für diese Geräte erlaubt.

Ein HAVi-System basiert auf dem IEEE 1394 Standard (auch Firewire genannt), welcher hohe Datenraten, einfache Konfiguration und Plug-and-Play bietet. In einem HAVi-Netzwerk wird zwischen Kontrollgeräten und Zielgeräten unterschieden, die wiederum, abhängig von der Ausprägung (einfach bis hoch kompliziertes CE-Gerät), in vier Kategorien/Geräteklassen unterteilt werden:

Full AV Device (FAV)
Ein FAV ist ein Kontrollgerät mit einem vollständige HAVi-Stack (Middleware) und einer Java-Laufzeitumgebung (inklusive Java UI-API) zum Ausführen von Java-Bytecode (DCM / Havlet). Beispiele hierfür wären digitale Fernsehgeräte oder Set-Top-Boxen.

Intermediate AV Device (IAV)
Kontrollgerät mit einem kompletten HAVi-Stack, aber keiner Java-Laufzeitumgebung.

Base AV Device (BAV)
BAVs sind zu kontrollierende Geräte (Zielgeräte) mit einem IEEE 1394 Anschluss, die ihre Software zur Ansteuerung (Java-Bytecode) bereitstellen. Ein typischer Vertreter für solch ein Gerät ist z.B. ein Videorecorder.

Legacy AV Device (LAV)
Diese Kategorie repräsentiert die derzeitigen Geräte der Unterhaltungselektronik: Geräte, die HAVi nicht kennen bzw. unterstützen. LAVs benötigen einen FAV oder IAV als Gateway, um ihre Dienste in einem HAVi-Netzwerk anbieten zu können.

In der HAVi-Architektur stellen Objekte, sogenannte Softwareelemente, ihre Dienste über wohldefinierte Schnittstellen (APIs) zur Verfügung. Die Softwareelemente können prinzipiell in vier Gruppen eingeteilt werden: Netzwerkdienste, Systemdienste, Gerätedienste und Dienste für die Benutzeroberfläche. Nachfolgend werden kurz die Aufgaben der einzelnen Softwareelemente aus Abbildung 14: HAVi-Architektur beschrieben.

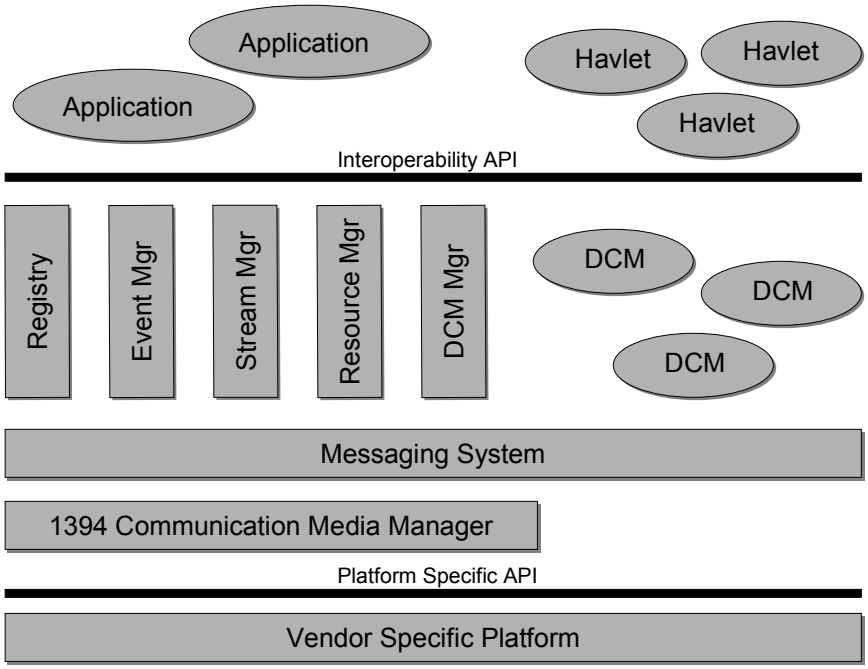

Abbildung 14: HAVi-Architektur

Der Communication Media Manager bietet die Schnittstelle für die Kommunikation in einem HAVi-System.
Das *Messaging System* ist für den Nachrichtenaustausch zwischen den Softwareelementen verantwortlich.
Die *Registry* stellt Dienste zum Registrieren und Suchen von Softwareelementen im Netzwerk zur Verfügung.
Der *Event Manager* informiert über Statusänderung eines Softwareelements oder Änderungen im Netzwerk.
Der *Stream Manager* ist für die Verwaltung von isochronen Datenverbindungen zuständig.
Der *Resource Manager* bietet Methoden für das Reservieren und Freigeben von Resourcen an.
Der *DCM Manager* ist für das Installieren und Entfernen von DCMs zuständig.
Ein *DCM* (Device Control Module) stellt die Schnittstelle für die allgemeine Kontrolle eines Gerätes zur Verfügung. Zu einer DCM gehören FCMs (Functional Component Modules), die verschiedene funktionale Komponenten eines Gerätes repräsentieren (Tuner, Amplifier, etc.).

Damit Geräte unterschiedlicher Hersteller untereinander Informationen austauschen und anzeigen können, bietet HAVi zwei Arten von Benutzerschnittstellen:
Level 1 UI, genannt Data Driven Interaction (DDI), und *Level 2 UI*, eine Java-UI-Umgebung basierend auf Java AWT 1.1 und Personal Java.
Mittels DDI bzw. mit dem entsprechenden Protokoll kann eine Applikation oder ein DCM sein User Interface im beschränkten Maße beschreiben. Wie sich die einzelnen Oberflächenelemente jedoch präsentieren bleibt

dem darzustellenden Gerät überlassen bzw. ist von Art und Größe des Displays abhängig.
Eine Level 2 UI – Anwendung ist eine Java-Applikation (Havlet), die nur auf einem FAV mit der entsprechenden HAVi-UI-Umgebung (Level 2 UI) gestartet werden kann, und die Erstellung von komfortablen Bedienoberflächen erlaubt.

4.4.2. Ausblick / Relevanz

Im Jahre 1997 trafen sich acht führende Hersteller von Unterhaltungselektronik um einen Branchen-Standard für die Interoperabilität von Geräten zu schaffen – dies war die Grundsteinlegung für HAVi. Die erste offizielle Veröffentlichung der Spezifikation HAVi 1.0 folgte im Herbst 1999. Im Sommer 2001 wurde die Spezifikation durch die bis dato geltende Version 1.1 abgelöst. Mit dem Konzept und durch das Mitwirken großer und wichtiger CE-Gerätehersteller schien HAVi eine große Zukunft bevorzustehen, doch bis heute sind nur eine Handvoll HAVi-fähige Geräte (FAVs) am Markt platziert. Es gibt viele mögliche Gründe, warum HAVi bisher der große Durchbruch in der Heimvernetzung verwehrt geblieben ist:

- fehlende aggressive Umsetzung durch führende CE-Hersteller
- embedded-Lösungen, begrenzte Resourcen, fehlende Java-Umgebung (JVM)
- die Komplexität von HAVi (Regelwerk umfasst mehr als 500 Seiten)
- fehlende Referenzlösungen bzw. -geräte
- Lizenzgebühren – HAVi ist ein kommerzieller Standard
- kein einheitliches Look-and-Feel, da jedes Gerät in einem HAVi-Netzwerk seine eigene Bedienoberfläche mitbringt (Havlet)
- HAVi-Spezifikation lässt Spielraum für Fehlinterpretationen
- IAV/BAV-Geräte unterliegen einem harten Wettbewerb/Preiskampf (zusätzliche Kosten durch IEEE 1394 bzw. Resourcen)
- Hersteller bieten proprietäre Systemlösungen (bewusster Ausschluss von „Fremdgeräten“)
- Standardisierung von neuen funktionalen Komponenten (FCMs)

Der HAVi-Standard setzt sich prinzipiell aus zwei Teilen zusammen: aus dem HAVi-Stack (Middleware) und aus dem HAVi-UI (Level 2 User Interface, Java-APIs). Die HAVi-Organisation hat allerdings mit dem HAVi-UI eine herstellerunabhängige Benutzerschnittstelle geschaffen, die nicht nur in HAVi eingesetzt wird, sondern auch von MHP (Multimedia Home Plattform) [4] und anderen Konsortien wie ATSC (Advanced Television Systems Committee) [5] und OpenCable [6] übernommen wurde.

HAVi wurde als Netzwerktechnologie für Audio und Video im Heimbereich für das Anwendungsgebiet von DynAMITE entwickelt. Da sich HAVi jedoch am Markt nicht durchsetzten konnte ist diese Technologie nicht relevan

Literaturreferenzen zu [Kapitel 4.4]

[1] www.havi.org
[2] IEEE 1394 : Standard für einen seriellen Hochgeschwindigkeits-Bus
[3] IEC 61883, Teil 1-5: Standard für die isochrone Übertragung über IEEE 1394
[4] www.mhp.org
[5] www.atsc.org
[6] www.opencable.com

4.5. Mobile Corba

Die Common Object Broker Architecture (CORBA) bietet eine programmiersprachen-unabhängige Schnittstelle und Modelle zur Entwicklung portierbarer, verteilter und objektorientierter Anwendungen. CORBA erlaubt es einem Client Operationen auf verteilten Objekten auszuführen, ohne sich um den Ort, die Programmiersprache, das Betriebssystem, oder Kommunikationsprotokolle kümmern zu müssen. CORBA trifft einige Annahmen über das Netzwerk. Es wird davon ausgegangen, dass ein verbindungsorientierter, verlässlicher Transport vorhanden ist, der eine Bytestrom-Abstraktion bietet. Diese Annahmen treffen üblicherweise auf ein kabelgebundenes Netzwerk zu. Damit stellt sich die Frage, welche Performanz CORBA in drahtlosen Netzwerken aufweisen kann. Probleme, die sich bei einem CORBA Betrieb in drahtlosen Netzen ergeben, sind Interoperabilität, Handover zwischen Mobilitätsdomänen und Mechanismen für den Zugang zu neuen mobilen Domänen. Um diese Anforderungen in CORBA zu integrieren, sind umfangreiche Ergänzungen notwendig, die jedoch die Interoperabilität zu anderen CORBA Implementierungen schwierig machen. Ein Vorschlag um *Mobile CORBA* zu implementieren, ohne die Interoperabilität zu beeinträchtigen, ist der Einsatz von *Mobile IP* als Unterbau zu verwenden. *Mobile IP* behandelt alle Probleme der Mobilität schon in der Netzwerkschicht, so dass keine Änderungen an CORBA nötig sind.

Im Jahr 1998 hat die Object Management Group (OMG) [>OMG1997] eine Initiative gestartet, um drahtlosen Zugriff und Mobilität in CORBA zu integrieren [>Black2001]. Ziele waren unter anderem eine einfache clientseitige Implementierung und eine Transparenz für Object Request Broker (ORB), die nicht für Mobilität ausgelegt sind. Alle Vorschläge, die eine Veränderung an der bestehenden ORB Struktur erfordert hätten, wurden von dem Konsortium abgelehnt.

Die von der OMG akzeptierte und standardisierte Architektur von Mobile IP besteht im Wesentlichen aus folgenden neuen Elementen:

- Mobile Interoperable Object Reference (MIOR)
- Home Location Agent (HLA)
- Access Bridge (AB)
- Terminal Bridge (TB)
- GIOP Tunnelling Protocol (GTP)

Die MIOR ist eine ortsvariante Objektreferenz, die das Terminal auf dem sich das Zielobjekt befindet identifiziert und zeigt zusätzlich noch auf die *Access Bridge* zeigt, bei der das Terminal zuletzt angemeldet war. Der Home Location Agent verwaltet ständig die aktuelle Adresse des mobilen Terminals und stellt Operationen zum Suchen und Aktualisieren der Adresse und zum Erhalten von initialen Dienstreferenzen.

Eine *Access Bridge* ist der netzwerkseitige Endpunkt eines GIOP (General Inter-ORB Protocol) Tunnels, eine Terminal Bridge ist das Terminalseitige Ende eines GIOP Tunnels. Beide kapseln GIOP Nachrichten mit Hilfe des *GIOP Tunnelling Protocol (GTP)*. Die Terminal Bridge kann auch noch einen Kanal für Mobilitätsereignisse bereitstellen. Mit Hilfe eines GIOP Tunnels werden GIOP Nachrichten zwischen einer Terminal und einer Access Bridge transportiert. Die Unterstützung des mobilen Terminals ist optional. Denn ein denkbares Szenario ist ein Arbeiter, der sich im Hotel an einen Mobile CORBA Dienst konnektiert. Solch ein Nutzer benötigt keine Mobilität, Portabilität reicht völlig aus.

Literaturreferenzen zu [Kapitel 4.5]

[>OMG1997] Object Management Group (OMG), http://www.omg.org

[>Black2001] Wireless Access and Terminal Mobility in CORBA, Kenneth Black and Jon Currey and Jaakko Kangasharju and Jari Länsiö and Kimmo Raatikainen, http://citeseer.nj.nec.com/black01wireless.html

4.6. MHP

4.6.1. Technologie

Die Multimedia Home Platform (MHP) wurde von ETSI standardisiert und erstmals im Februar 2000 als „MHP 1.0“-Standard veröffentlicht.
MHP kann als Erweiterung zum bestehenden DVB-System gesehen werden, die es Sendeanstalten und anderen Gesellschaften ermöglichen soll, Inhalte in grafischer Form an den Fernsehzuschauer zu bringen, interaktive Anwendungen zu übermitteln und Systeme leicht an das Internet anzubinden.
MHP wurde nach und nach in verschiedenen Profilen um die dargestellten Einsatzmöglichkeiten erweitert:

- Profil 1: Enhanced Broadcast Profile (grafische Inhalte) (MHP 1.0)
- Profil 2: Interactive TV Profile (interaktive Anwendungen) (MHP 1.0)
- Profil 3: Internet Access Profile (MHP 1.1)

Hintergrund der Entwicklung des offenen Standards MHP war der Grundgedanke, einen horizontalen Markt für die neue Art der Inhaltsübermittlung zu schaffen. Dieser Markt war bisher vertikal ausgelegt und wurde hauptsächlich von Pay-TV-Anbietern genutzt, wobei Inhaltsangebot, Middleware-Entwicklung und Set-Top-Boxen-Verkauf nach dem vertikalen Prinzip in der Hand einer einzigen Gesellschaft lagen.
Mit der Schaffung eines einheitlichen Standards sollte dies geändert werden. Für weitere Information siehe Quelle [1].

Um in den oberen Softwareschichten (betrifft vor allem die Anwendungsentwicklung) plattformunabhängig zu sein, wurde die Programmiersprache Java gewählt.
Das Modell zeigt hierbei gewisse Ähnlichkeiten mit dem Prinzip der Applets im Internet. Anwendungen in MHP, „Xlets“ genannt, implementieren alle die „javax.tv.xlet.Xlet“-Schnittstelle mit Methoden zum initialisieren, starten, pausieren und zerstören einer Applikation. Die Steuerung der Anwendungen erfolgt durch

implementierungsabhängige Management-Klassen, die sich im Rahmen des MHP-Software-Stacks auf der Set-Top-Box befinden. Die Übertragung der Anwendungen innerhalb eines Transportstroms erfolgt nach dem DSM-CC-Prinzip (Datenkarussell).

Der MHP-Software-Stack besteht neben den nicht zugänglichen Management-Klassen und nativen Implementierungen zur Hardware hin (Sicherheitsaspekt) im wesentlichen aus mehreren Basis-APIs, die den MHP-Anwendungen folgende Funktionalität zur Verfügung stellen:

- „Section Filtering API“: Möglichkeit, MPEG-Sektionen zu filtern, um Zugriff auf das Datenkarussell und SI-Tables zu erhalten
- „Tuning API“: Möglichkeit, auf verschiedene Transportströme mit verschiedenen Frequenzen zuzugreifen
- „Service Selection API“: Möglichkeit, verschiedene Services innerhalb eines Transportstroms auszuwählen
- „JMF API“: basierend auf dem „Java Media Framework“ zur Präsentation und Kontrolle sowie Manipulation von Video und Audio
- „Conditional Access API“: Schnittstelle zu CA-Systemen
- „Java AWT API“: Möglichkeit zur Darstellung von grafischen Elementen, komplette Funktionalität dieser Standard-Java-Klassen sind gewährleistet
- „HAVi API“: zum Teil aufbauend auf dem AWT, um zusätzliche Funktionalität zu gewährleisten, u. a. bessere Unterstützung von reinen Remote-Control-Systemen; gleichzeitig Bereitstellung von Schnittstellen zum Java Media Framework und Kontrolle des gesamten Grafiksystems
- „org.dvb.ui-API“: Möglichkeit des Alpha Blendings
- „Event-Manager-API“
- „Return Channel API“: Kontrolle von vorhandenen Rückkanälen verschiedenster Technologie
- APIs für Inter-Xlet-Kommunikation

Nähere Hinweise zum Aufbau des MHP-Software-Stacks sind unter Quelle [2] – Tutorials – MHP Software Stack zu finden.

4.6.2. Technische Rahmenbedingungen

Da die MHP-Anwendungen in Java geschrieben sind, besteht für sie keinerlei Plattformabhängigkeit.
Dies gilt für den gesamten MHP-Software-Stack natürlich nicht. Wenngleich auch die oberen Schichten des Stacks nach der Spezifikation noch Java-Implementierungen sind (teilweise aber auch hier schon abhängig vom Anbieter), so geht doch in aller Regel spätestens mit der nativen Schnittstelle zur Hardware eine starke Plattformabhängigkeit einher. Diese Abhängigkeit offenbart sich sowohl zum Betriebssystem und Grafiksystem an sich, als auch zu Hardwarekomponenten, v. a. der DVB-Empfangshardware. Ebenso wird natürlich eine lauffähige VM auf der Zielplattform benötigt.
Ein bekannter MHP-Software-Stack ist die Referenzimplementierung des Instituts für Rundfunktechnik (IRT) in München [3]. In der ursprünglich vorgestellten Version ist diese Implementierung beispielsweise an das Betriebssystem „Microsoft Windows“, „DirectX“ und an die Treiber der Technotrend-DVB-Karte gebunden. Bekanntgaben des Instituts zufolge soll aber eine Implementierung für Linux in naher Zukunft folgen.

4.6.3. Nachrichtensystem und Vernetzung

Mit Spezifikation des zweiten Profils wurde die Kommunikation nach außen definiert. Durch das „org.dvb.net.rc“-Package wird den Anwendungen bzw. Komponenten innerhalb des Stacks die Möglichkeit gegeben, jegliche Netzwerkhardware auf dem Zielsystem anzusprechen und zu aktivieren bzw. mit Zustimmung des Benutzers eine Verbindung aufzubauen, sei es über Modem, ISDN-Karte, LAN oder andere Hardware.
Verpflichtend nach MHP-Spezifikation ist hierbei die Unterstützung des TCP/IP-Protokolls. Durch Nutzung des Standard-Java-Packages „java.net“ kann z. B. eine Netzwerkverbindung hergestellt werden.
Höhere Protokolle wie etwa HTTP usw. sind nicht verpflichtend vorgeschrieben, wenngleich das dritte MHP-Profil gewisse Unterstützung dieser Protokolle nötig machen wird.

4.6.4. Zukunftsaussichten

Der Grundgedanke von MHP, einen horizontalen Markt für interaktive Dienste im Rahmen von DVB zu schaffen, wird von mehreren Gesellschaften und Organisationen unterstützt, die die Entwicklung und

serienmäßige Einführung von MHP vorantreiben wollen. Mit ganz oben steht die ETSI [4], worüber auch die MHP-Spezifikationen bezogen werden können.

Als großen Meilenstein auf dem Weg zur Einführung von MHP ist die „Mainzer Erklärung“ vom September 2001, in der sich die Rundfunkanstalten ARD, ZDF und RTL, die damalige Kirch-Gruppe und die Direktorenkonferenz der Landesmedienanstalten zu MHP bekannten und sich zur uneingeschränkten Unterstützung verpflichteten.

Tatsächlich strahlen diese Sender inzwischen seit längerem MHP-Anwendungen aus. Einen kleinen Überblick bietet Quelle [5].

Die Frage nach dem letztlich fehlenden Durchbruch von MHP kann mit dem sog. „Henne-Ei-Problem“ beantwortet werden. Sowohl die aufwändige Entwicklung von Anwendungen auf Senderseite als auch die hohen Initialkosten auf Seite der Set-Top-Box-Hersteller schrecken davon ab, als Vorreiter nach vorne zu preschen. Die Folge davon ist, dass Set-Top-Boxen mit MHP-Unterstützung kaum zu finden sind und die MHP-Anwendungen, die von den Sendern ausgestrahlt werden, von teilweise sehr dünnem Inhalt.
Ein Fortschritt ist somit aus jetziger Sicht nicht zu erzielen und die Zukunft von MHP hängt daher am seidenen Faden. Im Zuge zunehmender breitbandiger Anbindung der Haushalte und damit auch der TV-Geräte an das Internet stellt sich auch die Frage, ob MHP mittelfristig überhaupt konkurrenzfähig zum Übertragungsweg Internet sein kann, zumal dessen Technik schon bei weitem ausgereifter und kostengünstiger ist.

4.6.5. Relevanz für DynAMITE

MHP ist von geringer Relevanz für DynAMITE, da zum eine die Anwendungsgebiete nur zu kleinen Teilen überlappen. MHP ist keine Vernetzungstechnik und bietet keine Lösungen zur Selbstorganisation bzw. Konfliktlösung.

Literaturreferenzen zu [Kapitel 4.6]

[1] www.mhp.org
[2] www.mhp-interactive.org
[3] www.irt.de/IRT/mhp/mhp.htm
[4] www.etsi.org
[5] www.mhp-forum.de

4.7. ZeroConf /Rendezvous

4.7.1. Einleitung und Ziele

Das Ziel von Rendezvous, alias ZeroConf („zero configuration networking“), ist die konfigurations- und administrationslose Vernetzung von Geräten und Diensten [ZERO]. ZeroConf wurde als IETF Working Group [IETFC1] im September 1999 gegründet. Damals schon vorhandene proprietäre Technologien, wie z.B. AppleTalk für Macintosh und NetBEUI für Windows sollten durch eine offene, plattform-unabhängige und auf den verbreiteten TCP/IP-Protokollen basierende Technologie ersetzt werden.
Heute steht ZeroConf in direkter Konkurrenz zu anderen auf TCP/IP-Protokollen basierenden Technologien für die „Plug & Play“-Vernetzung, wie z.B. UPnP (siehe Abschnitt 4.2) und Jini (siehe Abschnitt 4.1).
Anwendungsgebiete und Zielmärkte für ZeroConf finden sich überall dort, wo für die Netzwerk-Konfiguration kein Administrator zur Verfügung steht, z.B.

- in den für DynAMITE interessanten Heimnetzwerken,
- in eingebetteten Netzwerken (z.B. im KFZ),
- in kleinen Büronetzwerken oder
- bei (mobilen) ad-hoc Netzwerken (z.B. E-Kiosk, Netzwerkspiele oder andere Konferenzen).

4.7.2. Technologie

Die Grundlage für die Entwicklung der Technologie ZeroConf bildete ein Anforderungskatalog [IETFD1], der aus den oben genannten Zielen und Anwendungsgebieten abgeleitet wurde. Um mit anderen Geräten im Netzwerk kommunizieren zu können, benötigen Geräte spezielle Informationen, wie z.B. eine IP-Adresse, eine „Netmask“, DNS-Adressen, einen DNS-Namen und konfigurierte Routen. Da in kleinen oder „ad-hoc“ Netzwerken normalerweise weder Systemadministrator noch für diese Zwecke konfigurierte Dienste, wie z.B. DHCP [RFC2131], DNS [RFC1034], Directory Service und MADCAP[RFC2730] zur Verfügung stehen, muss

ZeroConf für diese Dienste konfigurationslose Ersatzlösungen anbieten. ZeroConf unterteilt die Anforderungen in zwei Hauptaufgaben:

1. Automatische Konfiguration des Netzwerks
2. Dynamisches Auffinden von Diensten („Service-Discovery“)

Zur Bewältigung dieser Aufgaben sieht die ZeroConf-Spezifikation verschiedene Technologien und Protokolle vor [IETFD2], die in den nächsten Abschnitten vorgestellt werden.
Im Gegensatz zu UPnP ist die Steuerung von Geräten nicht Gegenstand der Technologie ZeroConf.

4.7.2.1. Automatische Konfiguration des Netzwerks

ZeroConf-Dienste bzw. Geräte finden ihre IP-Adresse in IPv4-Netzwerke über die Technologie „Link-Local-Adressing“ [IETFD3]. Beim „Link-Local-Adressing“ wählt das Gerät willkürlich eine IP-Adresse aus dem Bereich 169.254.xxx.xxx und überprüft anschließend, ob diese Adresse bereits von einem anderen Gerät belegt ist. Dieser Vorgang wiederholt sich so lange, bis das Gerät eine freie Adresse findet.
Dieses „Link-Local-Adressing“ findet nicht notwendigerweise Verwendung. Wird einem ZeroConf-Dienst/Gerät eine IP-Adresse über DHCP oder von einem Administrator zugewiesen, dann behält das Gerät diese Adresse. Diese Ermittlung einer IP-Adresse ähnelt dem Verfahren von UPnP [IETFD4]. Es ist grundsätzlich kompatibel zu diesem, d.h. UPnP- und ZeroConf-Geräte und Dienste können miteinander kommunizieren.

4.7.2.2. Service-Discovery

Durch die IP-Adresse kann das ZeroConf-Gerät im TCP/IP-Netzwerk kommunizieren. Anschließend muss das ZeroConf-Gerät eigene Dienste im Netzwerk bekannt geben, sowie von anderen Geräten angebotene Dienste im Netzwerk finden können. Für dieses „Service-Discovery“ verwendet ZeroConf eine Variante Domain Name System, den sogenannten „Multicast DNS“ (mDNS) [MDNS] in Verbindung mit „DNS-Service Discovery“(DNS-SD) [DNSSD]. MDNS dient wie DNS der Übersetzung zwischen Netzwerkadressen und Namen. Namen sind nicht nur benutzerfreundlicher sondern auch konstanter, da sie bei Netzwerkänderungen und veränderten Adressen beibehalten werden können. DNS-SD wird durch die beiden Arbeitsgruppen „ZeroConf“ und „DNS Extensions“ [DNSEXT] der IETF spezifiziert.

Tritt das Gerät in ein Netzwerk ein, dann gibt es seinen Service über eine „mDNS-SD notification“ bekannt. Diese Nachricht an eine Multicast-Adresse enthält neben dem

- Typ des Dienstes, z.B. IPP (Internet Printing Protocol)- Drucker,
- einen Namen, z.B. „Zentrale“,
- die IP-Adresse inklusive Port sowie
- optionale Informationen, im Falle eines Druckers z.B. die PPD-Datei (Postscript Printer Definition).

Jedes ZeroConf-Gerät im Netz erhält diese Nachricht und wird so über jeden neuen Dienst informiert.

Zusätzlich können ZeroConf-Geräte, die einem Netzwerk beitreten, bestimmte Dienste abfragen. Sie senden dazu sogenannte „DNS-SD query“-Nachrichten aus. Diese Mulitcast-Nachrichten enthalten den gesuchten Dienst-Typ, z.B. IPP Drucker. Alle entsprechenden Geräte beantworten diese Anfrage mit einer „DNS-SD response“, die den eigenen Namen enthält. Anschließend ermittelt das interessierte Gerät weitere Informationen zu einem bestimmten Dienst über eine DNS-Anfrage mit dem erhaltenen Namen.

4.7.3. Marktposition und Zukunft

Die Technologie ZeroConf wird hauptsächlich von Apple entwickelt und unter dem Namen Rendezvous vermarktet [APPLE1]. Neben der allgemeinen Integration in das Betriebssystem MacOS-X bietet Apple Rendezvous-Anbindungen für Anwendungen wie z.B. iTunes, iChat und den Browser Safari. Um die Verbreitung auch in anderen Systemumgebungen zu fördern, veröffentlichte Apple Implementierungen von Rendezvous im Quellcode für unterschiedliche Betriebssysteme, z.B. Windows, Windows Ce, Pocket PC, Linux und vxWorks [APPLE2].

Durch die frühe Marktpositionierung konnte ZeroConf zudem Erfolge in einigen Märkte erzielen [APPLE1,CHE03]:

- Alle namhaften Drucker-Hersteller unterstützen die Technologie ZeroConf, z.B. Lexmark, Canon, Brother, Epson und Hewlett Packard (HP). HP hat diese Technologie z.B. in den Druckern LaserJet 1300, LaserJet 2300, ColourJet 4600, ColourJet 5500 und DesignJet 120 integriert.

- In einigen Computer-Spielen, z.B. Nascar-Racing 2002, F1 Championship und iConquer, wurde ZeroConf zwecks automatischer Konfiguration von Netzwerkspielen integriert..
- Renommierte Datenbank-Hersteller, z.B. Oracle und Sybase, unterstützen die Netzwerkfähigkeit ihrer Produkte durch ZeroConf.

Da ZeroConf keine besonderen Steuerungsprotokolle vorsieht, lässt sich diese Technologie vergleichsweise leichter implementieren als die Konkurrenztechnologie UPnP. Aus diesem Grund finden sich Produkte mit ZeroConf hauptsächlich in Anwendungsgebieten mit verbreiteten und standardisierten Steuerungsprotokollen wieder, wie z.B. bei Druckern und Datenbanken.

In Anwendungsgebieten ohne verbreitete Steuerungsprotokolle, wie z.B. im A/V-Bereich, wird ZeroConf keine große Rolle spielen. Deshalb ist diese Technologie für DynAMITE nur von geringer Bedeutung.

Literaturreferenzen zu [Kapitel 4.7]

[>APPLE 1] http://www.apple.com/macosx/features/rendezvous/
[>APPLE 2] http://www.opensource.apple.com/projects/rendezvous
[>CHE03] Stuart Cheshire, "Zero Configuration Networking with Rendezvous", O'Reilly Emerging Technology Conference, Santa Clara, USA, April 2003.
[>DNSSD] http://www.dns-sd.org/
[>DNSEXT] http://ietf.org/html.charters/dnsext-charter.html
[>IETFC 1] http://www.ietf.org/html.charters/zeroconf-charter.html
[>IETFD 1] http://www.ietf.org/internet-drafts/draft-ietf-zeroconf-reqts-12.txt
[>IETFD 2] http://files.zeroconf.org/draft-ietf-zeroconf-host-prof-01.txt
[>IETFD 3] Cheshire, S., IETF draft: "Dynamic Configuration of IPv4 link-local addresses", http://www.ietf.org/internet-drafts/draft-ietf-zeroconf-ipv4-linklocal-04.txt, Work in progress.
[>IETFD 4] IETF draft., Auto-IP - Automatically Choosing an IP Address in an Ad-Hoc IPv4 Network. http://search.ietf.org/internet-drafts/draft-ietf-dhc-ipv4-autoconfig-05.txt.
[>MDNS] http://www.multicastdns.org/
[>RFC 2131] RFC 2131 Dynamic Host Configuration Protocol. IETF request for comments. http://search.ietf.org/rfc/rfc2131.txt?number=2131
[>RFC 1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.
[>RFC 2730] Hanna, S., Patel, B. and M. Shah, "Multicast Address Dynamic Client Allocation Protocol (MADCAP)", RFC 2730, December 1999.
[>ZERO] www.zeroconf.org

5. „Information Appliances“

5.1. Einführung

Nach einer Definition von Neoware auf linuxdevices.com ist ein Information Appliance

“[...] one of a new generation of smart computing products that is designed to be flexible and to perform dedicated tasks extremely well. An information appliance has the speed and power of a personal computer, but is simpler to use, easier to manage, and it costs less.” [>NEOWARE_2000]

Nach Donald Norman ist ein Information Appliance: „An appliance specializing in information: knowledge, facts, graphics, images, video, or sound. An information appliance is designed to perform a specific activity, such as music, photography, or writing. A distinguishing feature of information appliances is the ability to share information among themselves.” [>WOLF_2001]

Etwas fraglich ist der Punkt “speed and power of a personal computer”, da dies nicht unbedingt auf ein Smartphone zutrifft, diese Geräte jedoch auch als Information Appliances bezeichnet werden. Auch der Preis eines Handhelds kann den eines billigen PCs übersteigen. Strittig könnte auch die Bezeichnung „non-PC“ sein, da in einigen Geräten, wie z.B. Set-Top Boxen sich durchaus zum PC kompatible Hardware befinden kann.

Zusammenfassend lassen sich für Information Appliance folgende grobe Merkmale finden:

1. Ein Information Appliance ist ein „computing device“ für den Endanwender („consumer device“) mit (annähernd) der Leistungsfähigkeit eines PCs.
2. Ein Information Appliance ist leichter zu bedienen als ein PC [>NEOWARE_2000].
3. Ein Information Appliance enthält Funktionalität, die dem Anwender sofort nach Einschalten zugänglich sein sollte (ohne umfangreiche Konfigurationen oder Installationen) [>NEOWARE_2000].
4. Ein Information Appliance ist in der Regel kein „Alleskönner“ wie ein PC sondern auf dedizierte Aufgaben beschränkt [>WOLF_2001].
5. Ein Information Appliance hat in der Regel die Fähigkeit, auf ein Netzwerk zuzugreifen (insbesondere auf ein Netzwerk von Information Appliances), typischerweise auch auf das Internet. [>NEOWARE_2000].
6. Ein Information Appliance ist robuster als ein PC [>NEOWARE_2000].

Viele Information Appliances basieren dabei auf einer Konvergenz von Anwendungen, z.B. Telekommunikation und Internet in Smartphones [>WOLF_2001].

Ein weiterer Aspekt der Information Appliances ist die Dezentralisierung von Aufgaben auf verschiedene vernetzte Geräte, d.h. Funktionalität, die früher den Computern vorbehalten war, wird in Geräte wie Mobiltelefone, Fernseher, Waschmaschine integriert und vernetzt (pervasive / ubiquitous computing) [>WOLF_2001], s. auch Kapitel 1.7.

Beispiele für Information Appliances sind nach oben genannter Auffassung:

- PDAs (werden in Kapitel 6.2.2 beschrieben)
- Smartphones (werden in Kapitel 6.2.3 beschrieben)
- Webphones
- Email Appliances
- Set-top Boxen
- Webterminals
- ebooks
- Kühlschränke, Mikrowellen, Waschmaschinen mit Internetzugang

Literaturreferenzen zu [Kapitel 5.1]

[>ODLYZKO_1999] A. Odlyzko, The Visible Problems of the Invisible Computer: A skeptical Look at Information Appliances, 1999, http://www.firstmonday.org/issues/issue4_9/odlyzko/

[>TECHTARGET] http://whatis.techtarget.com/definition/0,,sid9_gci914561,00.html

[NORMAN_1998] D. A. Norman, The Invisible Computer: Why Good Products Can Fail, the Personal Computer Is So Complex, and Information Appliances Are the Solution, MIT Press, 1998

[>NEOWARE_2000] What is an Information Appliance, Neoware Systems, Inc., 05.04.2000, http://www.linuxdevices.com/articles/AT6664732218
[>INFOAPPS] about this site, Infoapps, http://www.infoapps.com/about/index.html
[>WOLF_2001] L. Wolf, M. Beigl, Ubiquitous Computing, 2001, http://www.teco.edu/lehre/ubiqws0102/03appliances_4.pdf

5.2. Digitale Bilderrahmen und Accessoires

Digitale Bilderrahmen empfangen die Bilder drahtlos (z.B. Infrarot) oder lesen sie direkt von SmartMedia- oder CompactFlash-Karten [>HEISE_25092003], [>HEISE_29032002].

5.2.1. Nokia SU-4

Der digitale Bilderrahmen SU-4 von Nokia kann bis zu 50 Bilder speichern, die über Infrarot (IrDA, Protokolle IrOBEX v1.3 und IrTranP) in das Gerät eingespeist werden. Eine Versendung von dem Bilderrahmen an ein anderes kompatibles Gerät ist ebenfalls über Infrarot möglich. Die Fotos können entweder einzeln auf dem Farbdisplay dargestellt werden oder als Diashow ablaufen. Das Farbdisplay hat eine Auflösung von 320 x 240 Pixel, misst 5,1 Zoll und kann bis zu 4.096 Farben darstellen. Unterstützt werden Bilder im JPEG- und GIF-Format (keine animierten GIFs) mit einer Auflösung von bis zu 1.024 x 768 Pixel und einer Größe von bis zu 1 MByte. Die Bilder werden auf die Auflösung des Displays skaliert. Die Bedienung erfolgt über Tasten an dem Gerät. Folgende Funktionen werden angeboten:

- Ein-/Aus-Taste
- Helligkeit regulieren
- Rotationstaste: Dreht das Bild um 90 Grad im Uhrzeigersinn
- 3-Stufen-Schalter für Diashow-Modus, Einzelbild-Modus, Bearbeitungsmodus
- Auswählen von Bildern
- Bilder für die Diashow markieren
- Infrarot-Übertragung aktivieren
- Bilder löschen

[>NOKIA_SU4], [>HEISE_25092003].

5.2.2. Nokia SU-7

Das Modell SU-7 von Nokia kann außerdem über eine SIM-Karte Bilder von jedem MMS-fähigen Mobiltelefon empfangen [>HEISE_25092003], [>NOKIA_SU7].

5.2.3. Nokia Medaillon

Von Nokia werden außerdem Ketten mit digitalen Medaillons, die über Infrarot Bilder empfangen, angeboten [>HEISE_25092003].

5.2.4. Digital Picture Portrait

Das „Digital Picture Portrait“ vom Georgia Institute of Technology, das im Rahmen der „Aware Home Research Initiative“ entwickelt wurde, stellt nicht nur Bilder dar sondern visualisiert symbolisch folgende Aspekte einer überwachten Person:

1. Gesundheit, z.B. Schlaf und Essenszeiten der Person, Wetter, Zustand der Wohnung
2. Interaktionen mit anderen Personen
3. physische Aktivitäten
4. spezielle Ereignisse

Die Informationen werden über das Internet empfangen und über Kontaktsensoren gewonnen. Z.B. lassen sich über Bodensensoren die Bewegungen der Person erfassen, Sensoren in der Toilette messen den Flüssigkeitsverlust [>HEISE_02032001].

Literaturreferenzen zu [Kapitel 5.2]

[>HEISE_25092003] Bildbetrachter für Regale und Decolletés: Digitale Bilderrahmen von Nokia, Heise News, 25.09.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/dz-25.09.03-001/default.shtml
[>HEISE_29032002] Digitale Bilderrahmen, Heise News 29.03.2002, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/ciw-18.03.02-000/default.shtml
[>HEISE_02032001] Digitaler Bilderrahmen als Babysitter, Heise News, 02.03.2001, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/hag-02.03.01-001/default.shtml
[>NOKIA_SU4] http://www.nokia.de/de/service/mobiltelefone/zubehoer/bilderrahmen_su-4/displayedItemId=76706/77218.html
[>NOKIA_SU7] http://www.nokia.de/de/service/mobiltelefone/zubehoer/bilderrahmen_su-7/displayedItemId=76770/77242.html

5.3. Smart Pen

5.3.1. C-Pen

Der C-Pen 800C von C-Technologies mit eingebauter Kamera kann Text scannen, abspeichern und übersetzen. Der gespeicherte Text wird z.B. auf den PC übertragen. Es lässt sich außerdem mit dem Gerät Text schreiben, der dann ebenfalls abgespeichert wird [>CPEN].

5.3.2. Logitech io Digital Pen

Der Logitech io Digital Pen erfasst während des Schreibens auf mitgeliefertem Spezialpapier die Stiftbewegung und speichert sie ab. Mit Hilfe einer mitgelieferten Ladestation lassen sich die Notizen über USB in den PC übertragen. Anhand der auf dem Papier markierten Felder werden die Notizen entsprechend ihrer Funktion (z.B. Termine, Todo-Liste) korrekt abgelegt. Mit Hilfe der Software MyScript® Notes wird die Handschrift in Text konvertiert. Auch Tabellen und Zeichnungen können konvertiert werden [>THINKGEEK], [>LOGITECH].

5.3.3. Akademische Projekte

An der Fachhochschule Regensburg wird ein biometrisches Handschriften Erkennungssystem (Biometrischer Smart Pen, BiSP-Projekt) zur Nutzer-Verifikation und Nutzer-Identifikation entwickelt [>BISP] .

Beim 3DSketch Projekt der University of Southern California wird mit Hilfe eines Stiftes die Konturen eines vorhandenen Objektes im dreidimensionalem Raum nachgezogen und am Computer modelliert [>3DSKETCH].

Literaturreferenzen zu [Kapitel 5.3]

[>CPEN] http://www.cpen.com
[>THINKGEEK] Logitech io Digital Pen, ThinkGeek, http://www.thinkgeek.com/gadgets/electronic/5c38/
[>LOGITECH] http://www.logitech.com/index.cfm?countryid=7&languageid=4&page=products/features/digitalwritingtopics&CRID=1551&parentCRID=1545&contentID=7156
[>BISP] http://homepages.fh-regensburg.de/~hoc39055/BiSP/BiSP_Page_hook_8.html
[>3DSKETCH] http://www.uni-mannheim.de/acm97/papers/songhan/3dsketch.html

5.4. Smart Watch

5.4.1. Microsoft MSN® Direct

Über den kostenpflichtigen Service MSN® Direct können einige ab 2004 z.B. von Fossil oder Suunto erhältliche Uhren von einem UKW Sender Nachrichten oder persönliche Mitteilungen über MSN Messenger empfangen. Der Service selbst soll US $60 im Jahr kosten.

[>MS_SWS] Microsoft Announces Wireless Service Plans For Smart Watches Available This Fall, Microsoft, 04.06.2003, http://www.microsoft.com/presspass/press/2003/jun03/06-04SmartWirelessServicePR.asp
[>HEISE_07122003] Microsofts Datenuhr kommt im Januar, Heise News, 7.12.2003, http://www.heise.de/newsticker/data/anw-17.12.03-001/

5.5. Intelligente Haushaltsgeräte

Ein in dieser Hinsicht häufig zitiertes Beispiel ist der Kühlschrank der selbstständig nach Bedarf Waren nachbestellt. Diese Idee scheint sich bislang nicht durchgesetzt zu haben, es wird auch davon gesprochen, dass der Verbrauer dies nicht wolle, so z.B. von Viktor Grinewitschus, technischer Leiter des inHaus [>HEISE_29062003].

Der Trend bei Haushaltsgeräten geht in Richtung Vernetzung von Geräten und Fernsteuerung. Bei der Vernetzung der Geräte gibt es verschiedene Möglichkeiten die Geräte zu verbinden (z.B. drahtlos über Funk, Nutzung des vorhandenen Stromnetzes über Powerline, über zusätzliche Steuerleitungen bei EIB oder z.B. über Ethernet oder IEEE1394) sowie verschiedene Protokolle, die die Kommunikation der Geräte untereinander regeln (z.B. das relativ neue CHAIN Protokoll oder Internet-Protokolle beim inHaus-Projekt [>inHaus]).

5.5.1. Serve@home von Siemens

Siemens bringt über Powerline vernetzte Haushaltsgeräte auf den Markt. Die Technologie Powerline macht es möglich, dass die Geräte einfach an das Stromnetz angeschlossen werden und keine zusätzliche Verkabelung erforderlich wird. Die Geräte können über eine Leitzentrale per Fernabfrage mit dem Mobiltelefon gesteuert werden. Innerhalb des Hauses lassen sich die Geräte über einen Tablet-PC bedienen [>HEISE_17122003], [>SERVE@HOME].

5.5.2. CHAIN vom CECED

CHAIN (CECED Home Appliances Interoperating Network) vom europäischen Verband der Hausgerätehersteller definiert das Datenaustauschprotokoll für vernetzte Haushaltsgeräte verschiedener Hersteller. Die Kommunikation innerhalb des Hauses soll über Stromleitungen (PowerLine) oder Funkverbindungen erfolgen. Unterstützt wird der CHAIN-Standard von folgenden Firmen: Arcelik, BSH Bosch und Siemens Hausgeräte, Elco Brandt, Candy, De'Longhi, Electrolux, Fagor, Gorenje, Liebherr Hausgeräte, Merloni, Miele und Whirlpool Europe. Entsprechende Geräte gibt es bislang noch nicht zu kaufen [>HEISE_11122003], [>GOLEM_11122003], [>ZVEI_10122003].

5.5.3. Internet-Kühlschrank

Einige angekündigte Internet-Kühlschränke sucht man mittlerweile vergeblich auf dem Markt. Der Screenfridge von e2Home (ein Joint-Venture von Ericsson und Electrolux) war ein Internet-Kühlschrank, der in einem Feldversuch von 50 Haushalten getestet dann aber doch nicht auf den Markt gebracht wurde. Ausgerüstet war das Gerät mit einem Internetzugang, einem 13“ LCD Touchscreen und einem Barcode-Lesegerät sowie der Möglichkeit die eingescannten Produkte gleich Online zu bestellen [>HEISE_12021999], [WIRED_12021999]. Ein Problem bei diesem und ähnlichen Produkten ist der relativ hohe Preis [>HEISE_16102002].

Eine Kombination aus Multimedia-PC und Kühlschrank wird unter der Bezeichnung „Internet Refrigerator“ auf der Internetseite von LG dargestellt. Ausgestattet mit 15,1“ TFT Display, Lautsprechern und Digitalkamera, ermöglicht er u.a. das Surfen im Internet, Fernsehen und Abspielen von MP3s. Das Gerät kam Ende letzten Jahres auf den Markt [>HEISE_16102002], [>LG_FRIDGE].

5.5.4. Vernetzte Klimaanlage

Laut einer Pressemitteilung von Heise wurde von einem Joint Venture von IBM und Carrier Corporation eine vernetzte Klimaanlage auf den Markt gebracht. Die Klimaanlage ist über GSM-Modul drahtlos mit dem Internet verbunden und kann somit ferngesteuert werden [>HEISE_09042001].

Literaturreferenzen zu [Kapitel 5.5]

[>HEISE_29062003] Der "Smart Garden" für das "intelligente Haus", Heise News, 29.06.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/jk-29.06.03-004/default.shtml
[>HEISE_11122003] Verband entwickelt Kommunikationsprotokoll für Hausgeräte, Heise News, 11.12.2003, http://www.heise.de/newsticker/data/anw-11.12.03-000/
[>GOLEM_11122003] CHAIN-Standard erlaubt Datenaustausch zwischen Hausgeräten, Golem, 11.12.2003, http://www.golem.de/0312/28900.html
[WIRED_12021999] The Coolest Internet Appliance, Wired News, 12.02.1999, http://www.wired.com/news/technology/0,1282,17894,00.html
[>HEISE_12021999] Online mit dem Kühlschrank, Heise News, 12.02.1999, http://www.heise.de/newsticker/data/fr-12.02.99-000/
[>HEISE_16102002] Multimedia für den Kühlschrank, Heise News, 16.10.2002, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/pmz-16.10.02-000/default.shtml
[>LG_FRIDGE] http://www.lge.com/products/homenetwork/internetproduct/refrigerator/introduction.jsp
[>INHAUS] inHaus, Technologien und Produkte, http://www.inhaus-duisburg.de/projektbeschreibung/tech_produkte.htm
[>HEISE_17122003] Siemens vernetzt Hausgeräte mit Powerline, Heise News, 17.12.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/dab-17.12.03-002/default.shtml
[>SERVE@HOME] http://www.siemens.com/index.jsp?sdc_rh=null&sdc_flags=null&sdc_sectionid=0&sdc_secnavid=0&sdc_3dnvlstid=&sdc_countryid=175&sdc_mpid=0&sdc_unitid=999&sdc_conttype=8&sdc_contentid=1131717&sdc_langid=0&
[>HEISE_09042001] IBM entwickelt Internet-Software für Klimaanlagen, Heise News, 09.04.2001, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/wst-09.04.01-001/default.shtml
[>ZVEI_10122003] Reibungslose Kommunikation im Haus, ZVEI, 10.12.2003, http://www.zvei.org/news/

5.6. Hausautomation

Für die Automatisierung des Heims stehen eine Reihe von Vernetzungstechniken, wie z.B. das EIB (s. Abschnitt 6.1.4 EIB) zur Verfügung, die ausschließlich in der Haustechnik eingesetzt werden. Es besteht auch die Möglichkeit, dass sich eher Lösungen über Powerline, Ethernet, Firewire oder drahtlos durchsetzen werden. Der Trend scheint allgemein in Richtung offener Systeme zu gehen [>ARCH_01042003]. Da es einen Trend zur Integration von Haustechnik und Unterhaltungselektronik gibt [>HEISE_09062002], wäre es sinnvoll, wenn sich in den Bereichen gemeinsame Vernetzungstechniken und Protokolle durchsetzten.

Im Markt für Heimautomation sieht z.B. die Unternehmensberatung Frost & Sullivan in Zukunft ein großes Wachstumspotential ([>ARCH_01042003]).

In diesem Abschnitt werden nicht nur einzelne Produkte genannt, sondern auch Initiativen und Projekte aufgeführt, die als Anlaufstelle zur Information über weitere Entwicklungen dienen können.

5.6.1. inHaus vom IMS

Das inHaus Projekt vom Fraunhofer-Institut für mikroelektronische Schaltungen und Systeme (IMS) in Duisburg umfasst ein Wohnhaus, ein Werkstatthaus, ein vernetztes Fahrzeug und einen vernetzten Garten. Es wird Haustechnik der Zukunft unter besonderer Berücksichtigung der Vernetzung der einzelnen Geräte erprobt. Zu den eingesetzten Technologien und Applikationen gehören ISDN, DSL, GSM einschließlich der SMS-Funktion, GPRS, UMTS, WWW, E-Mail und WAP. Für die interne Datenübertragung werden IEEE1394 (Firewire), EIB und Bluetooth eingesetzt [>inHaus].

5.6.2. Initiative Intelligentes Wohnen

Hersteller verschiedener Branchen haben sich zur Initiative Intelligentes Wohnen zusammengefunden. Ziel der Initiative ist es, den Einsatz von Technologien zur Vernetzung von Geräten und Systemen der Haustechnik voranzutreiben. Dies umfasst insbesondere die Etablierung von Standards und Normen und das Sicherstellen einer Mindestausstattung in der Vernetzung. Mitglieder der Initiative auf Herstellerseite sind bisher ABB Stotz-Kontakt, AEG Hausgeräte, BSH Bosch und Siemens Hausgeräte, Buderus, Dehn + Söhne, Deutsche Telekom, Diehl AKO, Gira, Hager Tehalit, Hirschmann, Honeywell, ise, Kathrein, Kerpen, Loewe, Merten, Miele, Philips, Reichle & De-Massari, Schüco, Siedle, Siemens, Somfy, Stiebel Eltron, TCS, Techem, Viessmann und Winkhaus [>ARCH_14112003].

5.6.3. Gira SmartTerminal

Der Gira SmartTerminal dient sowohl für die Bedienung der EIB Hausinstallation als auch als Internetterminal zum Abruf von Informationen (z.B. Wettervorhersage, Staumeldungen, Nachrichten) oder Emails [ARCH_08122003].

5.6.4. Loewe/ Gira Homeserver

Ein Loewe-Fernseher in Verbindung mit dem OnlinePlus Aufrüstsatz lässt sich als Kommunikationsschnittstelle für den Gira Homeserver nutzen und damit die Hausinstallation von der Lichtanlage bis zum Klimasystem über EIB steuern [>GIRA]. Der auf dem OnlinePlus-Aufrüstsatz installierte Webbrowser wird für die Steuerung der Hausanlage genutzt. Darüber hinaus kann mit dem Fernseher in Verbindung mit dem Loewe OnlinePlus Aufrüstsatz im Internet gesurft werden.

Alternativ dazu lässt sich für die Anbindung an den Gira Homeserver auch ein Notebook, PC, PDA oder WAP-Handy einsetzen.

Literaturreferenzen zu [Kapitel 5.6]

[>INHAUS] inHaus, Technologien und Produkte, http://www.inhaus-duisburg.de/projektbeschreibung/tech_produkte.htm

[ARCH_08122003] Gira SmartTerminal kombiniert EIB und Internet, ARCHmatic, 08.12.2003, http://www.bauzentrale.com/news/2003/1frame.htm?http%3A//www.bauzentrale.com/news/2003/1346.php4

[>GIRA] Gira HomeServer2, http://www.gira.de/data/1886100603.pdf

[>HEISE_09062002] Vom PC zum Heimserver, Heise News, 09.06.2002, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/em-09.06.02-000/default.shtml

[>ARCH_14112003] 'Initiative Intelligentes Wohnen' als unternehmensübergreifende Plattform gegründet, ARCHmatic, 14.11.2003, http://www.bauzentrale.com/news/2003/1frame.htm?http%3A//www.bauzentrale.com/news/2003/1270.php4

[>ARCH_01042003] Steigende Nachfrage nach Hausautomation: zweistelliges Wachstum ab 2006, ARCHmatic, 1.4.2003, http://www.bauzentrale.com

5.7. Set-top Box (STB)

Eine Set-Top Box ist ein Zusatzgerät für den Fernseher, mit dem es dem Benutzer möglich ist, Online-Dienste (Webbrowser, Email) zu nutzen und / oder digitales Fernsehen zu empfangen [>SEARCHNET]. Eine aktuelle Marktübersicht zu geben ist schwierig, da Geräte in einer hohen Frequenz auf dem Markt auftauchen und wieder verschwinden.

Literaturreferenzen zu [Kapitel 5.7]

[>SEARCHNET] http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci212971,00.html

6. Infrastruktur

6.1. Netzwerktechnologien

6.1.1. Bluetooth

6.1.1.1. Entstehung

Die kabellose Übertragungstechnik Bluetooth wurde 1984 von 2 Mitarbeitern der Firma Ericsson entwickelt. 1998 wurde von Ericsson, IBM, Intel, Nokia und Toshiba die Bluetooth Initiative gegründet aber erst Mitte 2000 erschienen erste Bluetooth-Geräte auf dem Markt. Der Bluetooth Standard 1.1 wurde Anfang 2001 verabschiedet. Aus der Bluetooth Initiative wurde die Bluetooth Special Interest Group (BSIG). Im April 2002 wurde Bluetooth als IEEE 802.15.1-2002 Standard verabschiedet. Der Standard 1.2 wurde im November 2003 verabschiedet, er ist abwärtskompatibel zum Standard 1.1. Es gibt bereits erste Geräte, die diesen neuen Standard unterstützen, Chips der Version 1.1 dominieren jedoch noch den Markt.

6.1.1.2. Technologie

Übertragungstechnologie

Bluetooth ist ein auf Funk basierender Standard für die drahtlose Kommunikation. Er wurde entwickelt, um auch ohne Kabel Kommunikation zwischen verschiedenen Geräten zu ermöglichen. Bluetooth nutzt das lizenzfreie 2,4 GHz-ISM-Band (ISM: Industrial Scientific Medical). Innerhalb des verfügbaren Frequenzbereiches gibt es 79 1MHz breite Kanäle, die für Bluetooth genutzt werden können. In einigen Ländern ist jedoch die Anzahl der Kanäle eingeschränkt, da diese bereits für andere Anwendungen eingesetzt werden. Aufgrund des Spread-Spectrum-Frequency-Hopping, das von Bluetooth gewählt wurde, um die Abhörung durch Dritte und die Störungen durch andere Geräte (z.B. Mikrowellen, Garagenöffner, WLAN-Geräte) gering zu halten, wird 1600 mal in der Sekunde die Frequenz gewechselt. Dabei wird jedes Bluetooth-Paket in einer anderen Frequenz übertragen und zwar innerhalb von in der Regel einem (oder bis zu 5) Timeslots von 625 µs Länge.

Die Datenrate beträgt zur Zeit 1MBit/s soll aber in Zukunft weiter steigen. Bluetooth ermöglicht für die Übertragung eine Kombination aus synchroner und / oder asynchroner Übermittlung von Daten und Sprache, wobei bis zu 3 gleichzeitige synchrone Sprachübertragungen möglich sind. Die asynchrone verbindungslose Übertragung steht dabei für die point-to-multipoint Verbreitung von Daten zur Verfügung (ACL). Die synchrone Übertragung (SCO) kann für verbindungsorientierte point-to-point Übermittlung von Audio oder einer Kombination aus Audio und Daten verwendet werden.

Netztopologie

Obwohl es häufig so eingesetzt wird, ist eine Bluetooth-Verbindung nicht nur zwischen zwei Geräten möglich. Bis zu sieben Slaves können in einem Piconetz mit genau einem Master aktiv sein (weitere Slaves in einem „parked“ state). Der Master ist dabei ein Bluetooth-Gerät wie jedes andere auch, er wird dadurch zum Master, dass er die Verbindung initiiert. Er kontrolliert die Nutzung des Kanals und ist an jeglicher Kommunikation beteiligt, d.h. es gibt keine point-to-point Verbindung zwischen zwei Slaves. ACL Pakete, die nicht an einen bestimmten Slave adressiert sind, werden als broadcast-Pakete betrachtet und von allen Slaves empfangen. Bis zu 10 Piconetze können zusammen ein Scatternetz bilden.

Der Aufbau der Netze ist nicht festgelegt, die Verbindungen können auf einer ad-hoc Basis jederzeit neu aufgebaut und wieder beendet werden.

Die Reichweite eines Bluetooth-Gerätes richtet sich danach, für welche Klasse es ausgelegt ist; wie folgende Tabelle zeigt:

Klasse	Reichweite draußen	Reichweite innen	Ausgangsleistung
1	ca. 100-130m	ca. 50-80m	100mW
2	ca. 25-35m	ca. 20-30m	2,5mW
3	ca. 10-18m	ca. 8-12m	1mW

Fehlerkorrektur

Im Bluetooth-Standard ist eine automatische Fehlerkorrektur vorgesehen. Möglich ist hier eine Kombination aus FEC (forward error correction) und ARQ (automatic repeat request). Bluetooth gilt als besonders robust, hat aber in der Vergangenheit z.B. die WLAN Funkübertragung stark gestört. Dies sollte jedoch mit der Bluetooth 1.2 Spezifikation der Vergangenheit angehören (s. 6.1.1.5 Markt und Zukunft).

Sicherheitsmechanismen

Um einen Zugriff unerwünschter Dritter zu verhindern und eine Abhörsicherheit zu gewährleisten sieht die Bluetooth Spezifikation als Sicherheitsmechanismen die Authentifizierung und Verschlüsselung vor. Für die Authentifizierung wird ein 128-Bit Link Key als Schlüssel verwendet. Für die Verschlüsselung der Daten lässt sich dann ein kürzerer Schlüssel einsetzen, der aus dem Link Key generiert wird. Wenn zwei Bluetooth-Geräte noch nicht miteinander kommuniziert haben, wird in der Regel über die Eingabe der gleichen PIN auf beiden Seiten ein Initialisierungsschlüssel generiert. Der Initialisierungsschlüssel wird dann nur für die Aushandlung des später zu verwendenden Link Keys genutzt und danach verworfen. Die beiden Geräte sind in Zukunft einander bekannt und können sich über den Link Key gegenseitig authentifizieren.

Für Bluetooth existieren drei verschiedene Sicherheitsmodi:

- Security Mode 1 (non-secure): das gerät initiiert von sich aus keine Sicherheitsfunktionen
- Security Mode 2 (service level enforced security) : Authentifizierung und Verschlüsselung werden von höheren Übertragungsprotokollen oder Anwendungen durchgeführt
- Security Mode 3 (link level enforced security): Sicherheitsfunktionen werden in der Verbindungsschicht realisiert, zum großen Teil bereits in der Firmware des BT-Moduls

Die Sicherheitsstrategien bei Bluetooth weisen noch einige Lücken auf, die für die 1.1 Spezifikation genauer in [>BTVAINIO], [>BTCT1103], [>BTCT1203] dargelegt werden. Mit der Version 1.2 wird jedoch die Nutzung des Unit Keys als Link Key „deprecated“. Dies bedeutet, dass dies zwar noch implementiert werden darf aber nicht empfohlen wird ([>BTSPEC3], S. 155). Dies hat Auswirkung auf die Kritik, dass aufgrund der symmetrischen Verschlüsselungsstrategie unter Umständen der Schlüssel eines Gerätes weiteren Geräten preisgegeben werden kann ([>BTVAINIO]:Nutzung eines Unit Key als Link Key). Ein weiterer Kritikpunkt ist, dass aufgrund der eindeutigen Adresse eines Bluetooth-Gerätes bei Zuordnung eines Benutzers zu diesem Gerät Bewegungsprofile erstellt werden können. Dies wird sich wohl erst mit der Einführung des Anonymity Modes erübrigen, s. 6.1.1.5.

Stromverbrauch

Um den Stromverbrauch eines Gerätes gering zu halten, existieren verschiedene Modi, in denen sich ein Bluetooth-Gerät befinden kann ([>BTKR]):

Modus	**Beschreibung**	**typische Stromaufnahme (bei 3,3V)**
Sniff-Modus	Default-Modus. In diesem Modus wird in periodischen Abständen nach Nachrichten gehorcht.	ca. 300µA
Master-Modus	Verbindungsaufnahme zu anderen Geräten als Master	ca. 3-30mA
Hold-Modus	Das Gerät bleibt im Piconetz, es findet jedoch keine Datenübertragung statt, das Gerät bleibt aber verbindungsbereit.	ca. 60µA
Park-Modus	Das Gerät nimmt nicht am Datenverkehr teil.	ca. 30µA

Protokollarchitektur

Bluetooth basiert auf einigen Kernprotokollen und ermöglicht ansonsten den Einsatz bereits vorhandener Protokolle, wie z.B. OBEX oder TCP. Weitere Protokolle können auf den oberen Schichten aufsetzen. Das Host Controller Interface (HCI) fungiert als Kommandointerface zu unteren Bluetooth-Layern (Baseband und Link Manager oder L2CAP).

Bluetooth Layer	Protokoll	Beschreibung
Bluetooth-Kernprotokolle	Baseband	Low-Level Kommunikation: setzt Pakete zusammen und synchronisiert sie, regelt Frequenzhopping, Fehlerkorrektur ...
	LMP: Link Manager Protocol	Baut Verbindung auf, regelt Verbindungszustände, Powermodi, zuständig für Verschlüsselung und Authentifizierung
	L2CAP: Logical Link Protocol and Adaption Protocol	Setzt Pakete auf einer höheren Ebene zusammen (bis 64 KByte), stellt Verbindungen für höhere Protokollschichten zur Verfügung
	SDP: Service Discovery Protocol	liefert Geräteinformationen, Verbindungsinformationen
Cable Replacement Protocol	RFCOMM: Serial Cable emulation protocol	emuliert RS-232 Protokoll
Telephony Control Protocols	TCS Binary: Telephony Control Protocol – Binary	bit-orientiertes Protokoll mit Rufkontrolle, Verbindungsaufbau, Sprach- und Datenübertragung
	AT-Commands	AT-Befehle für Mobiltelefone und Modems
Aufgesetzte Protokolle	PPP, UDP/TCP/IP, OBEX, WAP	

Es müssen nicht sämtliche von einem Bluetooth-Stack implementiert werden, bei einigen Geräten reichen die Kernprotokolle.

Bluetooth-Profile

Über spezifische Profile werden für einzelne Anwendungen die Details der Realisierung über Bluetooth spezifiziert. Unterstützen zwei Bluetooth-Geräte (auch unterschiedlicher Hersteller) das gleiche Profil, so sollte für den Anwender der Nutzung dieses Dienstes (z.B. Austausch von Dateien) zwischen den Geräten nichts mehr im Wege stehen. Nicht spezifiziert ist allerdings die Benutzerführung innerhalb des Gerätes.

Es werden nicht alle Profile von allen Geräten unterstützt. Unterschieden wird zwischen Kann- und Muss-Profilen, wobei auch die Muss-Profile nicht von Geräten unterstützt werden müssen, bei denen dieser Dienst Unsinn wäre.

Beispiele:

- OPP: Object Push Profile. Hiermit lassen sich Daten von Organizer-Anwendungen übertragen, zum Beispiel Visitenkarten oder Termine. Es wird zwischen OPP-Server und OPP-Client unterschieden
- FT: File Transfer Profile. Es können Dateien von einem FT-Client zu einem FT-Server übertragen werden.
- Weitere Profile sind in [>BTCT2303], [>BTHEIMO1] aufgeführt und erläutert.

6.1.1.3. Geräte und Anwendungen

Aufgrund der Integration der Bluetooth Technologie in eine Single-Chip Lösung mit geringem Platzverbrauch und Stromaufnahme lässt sich Bluetooth auch in kleinste Geräte einbauen, z.B. Mobiltelefone. Typische Größen liegen bei 7mm x 7mm Abmessungen (z.B. Infineon BlueMoon), auch kleinere Chips sind auf dem Markt. Bluetooth ist bereits in eine Vielzahl von Geräten wie Tastaturen, Mäuse, Headsets, Mobiltelefone (Handys), PDAs, Notebooks, Drucker, Festplatten, DSL-Modems, Digitalkameras, Fernbedienungen, MP3-Player, Koffer (z.B. Samsonite "Intelligent Baggage"), Camcorder, Kühlschränke, Mikrowellen und Wäschetrockner integriert worden. Ein PC (oder Notebook) lässt sich über USB-Module, PC-Card, CF (Compact Flash), oder PCI-Einsteckkarten nachrüsten.

Eine typische Anwendung für Bluetooth ist der Datenaustausch. Dazu gehört z.B. die Synchronisation zwischen PDA und PC, der Austausch von Visitenkarten, das Transferieren von Bildern auf den Drucker, die Übertragung von Dateien und die Übertragung von Audio. Zu weiteren Anwendungen zählen die Steuerung von Geräten, z.B. die Bedienung des DVD-Spielers aus einem anderen Raum heraus, die „drahtlose“ Maus oder Tastatur, der schnurlose Zugang zum Internet, z.B. über Handy vom PDA oder über ein DSL-Modem vom PC/Notebook, die Nutzung eines Bluetooth-Geräts als Zugangskontrolle sowie der Zugriff auf Informationen wie Fahrpläne oder Touristeninformationen an öffentlichen Plätzen.

6.1.1.4. Vergleich mit anderen Technologien

Zu den Vorteilen von Bluetooth gehören der verhältnismäßig geringe Preis, die hohe Verbreitung in verschiedensten Geräten, die Unterstützung verschiedenster Anwendungen, eine niedrige Emissionsrate und Stromaufnahme sowie die geringe Störempfindlichkeit (z.B. im Vergleich zu WLAN). Als Nachteile können die relativ geringe Reichweite (WLAN bis zu 500m), die jedoch für typische Anwendungen ausreichen dürfte, sowie die geringere Übertragungsgeschwindigkeit (WLAN 11 Mbit/s), die jedoch in der Version 2.0 weiter aufgestockt werden soll, insbesondere gegenüber WLAN gelten. Des Weiteren ist die Anzahl der Netzteilnehmer bei Bluetooth begrenzt. Ein Vorteil gegenüber IrDA besteht darin, dass kein Sichtkontakt zwischen den kommunizierenden Geräten erforderlich ist.

6.1.1.5. Markt und Zukunft

Zur Zeit sind insgesamt über 1200 Bluetooth-Geräte zertifiziert worden, insbesondere bei Handys besteht eine hohe Verbreitung von Bluetooth. 2002 wurden insgesamt 35 Millionen Chipsätze verkauft, wobei der Markt 2003 noch weiter angewachsen ist.

In der gerade verabschiedeten Spezifikation 1.2 sind einige Kritikpunkte von Bluetooth ad acta gelegt worden([>BTHEI1], [>BTSPEC2]):

- Schnellerer Verbindungsaufbau
- Enhanced Voice Processing :Verbesserung bei der Sprachübertragung, unter anderem sollen hier Störung durch Fehlererkennung beseitigt werden
- Enhanced Quality of Service: hiermit lässt sich die Priorität der Dienste eines Piconetzes regeln (z.B. Musikübertragung sollte eine hohe Priorität bekommen, um Unterbrechungen zu vermeiden)
- bessere Rücksichtnahme gegenüber WLAN-Funknetzen durch Adaptive Frequency Hopping (AFH)

Für die Zukunft sind höhere Datenraten geplant. Auch ein Anonymity Modus steht in der Diskussion. Des weiteren soll die Master-Slave Beziehung innerhalb eines Piconetzes wegfallen und anders geregelt werden ([>BTYOSH]). Zur Zeit besteht das Problem, dass beim Wegfall des Masters das komplette Piconetz zusammenbricht.

Für bessere Benutzerfreundlichkeit durch einfache Bedienung und Wahlfreiheit was die eingesetzte Technologie anbetrifft sowie Interoperabilität zwischen den Geräten ist anzustreben, dass eine Kommunikation zwischen verschiedenen Technologien und Geräten möglich wird. Es sollte keine Einschränkung der Nutzung durch ausschließlichen Zugriff auf Bluetooth-Geräte von Bluetooth-Geräte bestehen. Denkbar ist hier eine Interoperabilität zwischen z.B. WLAN, Bluetooth, ISDN und Ethernet-Netzen. Aufgrund der Schichten-Architektur von Bluetooth ist diese Möglichkeit grundsätzlich gegeben, in den Profilen sind auch einige Schnittstellen vorgesehen (z.B. CIP: Common ISDN Access). Eine Technologie, die das Auffinden und Nutzen von verschiedensten Geräten vorsieht, ist UPnP (s. 4.2). Es gibt auch bereits einen Ansätze, UPnP über Bluetooth einzusetzen, der sich jedoch noch nicht flächendeckend in Endgeräten wiederfinden lässt. Im Extended Service Discovery Profile (ESDP), das zur Zeit noch im Draft vorliegt, ist diese Möglichkeit vorgesehen [>BTESDP], in der Suche nach Unterstützung dieses Profils in der Heise Bluetooth Datenbank (http://www.bluetooth-db.de) sind jedoch z.Z. noch keine Geräte gefunden worden.

Literaturreferenzen zu [Kapitel 6.1.1]

[>BTCHON] Y. K. Choong et. al, WPAN using UPnP over Bluetooth, Nanyang Technological University, Singapore, 2001, http://www.computer.org/CSIDCArchive/2001ProjectReports/Nanyang.pdf
[>BTCT1103] M. Schmidt, Blauzahnlücken, c’t 11, 2003
[>BTCT1203] M. Schmitdt, Maskerade, c’t 12, 2003
[>BTCT2303] D. Zivadinovic, Firstclass Luftverkehr, c’t 23/2003
[>BTESDP] A. Ayyagari, Bluetooth ESDP for UPnP, 0.95a Draft, 2001

[>BTHEI1] Bluetooth 1.2 verabschiedet, heise news, 6.11.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/dz-06.11.03-000/default.shtml&words=Bluetooth
[>BTHEIMO1] M. Özkilic, D. Zivadinovic, Vertikale Schnitte, Heise mobil, http://www.heise.de/mobil/artikel/2003/04/18/bluetooth-profile/default.shtml
[>BTKR] A. Kral, H. Kreft, Wireless LANs Networker's Guide, Markt+Technik, 2003
[>BTPROT] R. Mettala, Bluetooth Protocol Architecture (Whitepaper), Version 1.0, Bluetooth SIG, 25.08.1999
[>BTSPEC1] „Specification of the Bluetooth Sytem" Volume 1,Version 1.1 (22.02.2001), Bluetooth SIG, http://www.bluetooth.org
[>BTSPEC2] Bluetooth Core Specification 1.2, http://www.bluetooth.com/dev/spec.v12.asp
[>BTSPEC3] Specification of the Bluetooth System 1.2, 05.11.2003, http://www.bluetooth.com
[>BTVAINIO] Vainio, Juha: "Bluetooth Security", Helsinki University of Technology -Department of Computer Science and Engineering, 2000, http://www.niksula.cs.hut.fi/~jiitv/bluesec.html
[>BTYOSH] J. Yoshida, Scientist tips features of Bluetooth 2.0, EE Times, 17.06.2002, http://www.eetimes.com/story/OEG20020611S0033

6.1.2. WLAN

6.1.2.1. Entstehung

1985 wurde vom IEEE (Institute of Electrical and Electronics Engineers, Inc) eine Arbeitsgruppe 802.11 eingesetzt. 1997 bestätigte dann das IEEE den Standard IEEE 802.11. Darauf folgten einige Verbesserungen in Form von Erweiterungen des Standards, die bereits verabschiedet wurden:

- 1999: 802.11b („Higher-Speed Physical Layer Extension in the 2.4 GHz Band")
- 2002: 802.11a („High-speed Physical Layer in the 5 GHz Band")
- Juni, 2003: 802.11g ("Further Higher Data Rate Extension in the 2.4 GHz Band")
- September. 2003: 802.11h ("Spectrum and Transmit Power Management Extensions in the 5 GHz Band in Europe")

Am weitesten verbreitet ist bei der Implementierung in den Geräten zur Zeit noch der 802.11b Standard.

6.1.2.2. Technologie

Der IEEE 802.11 Standard ist so konzipiert, dass er sich in die bereits bestehenden IEEE 802.x Standards für Netzwerke einfügt. Dazu gehören z.B. 802.3 (Ethernet) und 802.5 (Token-Ring). In Abbildung 15 ist zu erkennen, dass vom IEEE 802.11 Standard nur die untersten Protokollschichten abgedeckt werden.

Die verschiedenen IEEE 802.x Standards für LAN beschreiben die physikalische Ebene (Physical Layer) und die Zugriffskontrollebene (Medium Acess Control). Dabei entspricht die physikalische Ebene dem entsprechenden Layer im ISO/OSI Modell, die Zugriffskontrollebene dagegen deckt nur einen Teil der Sicherungsschicht (Data Link Layer) des OSI-Modells ab.

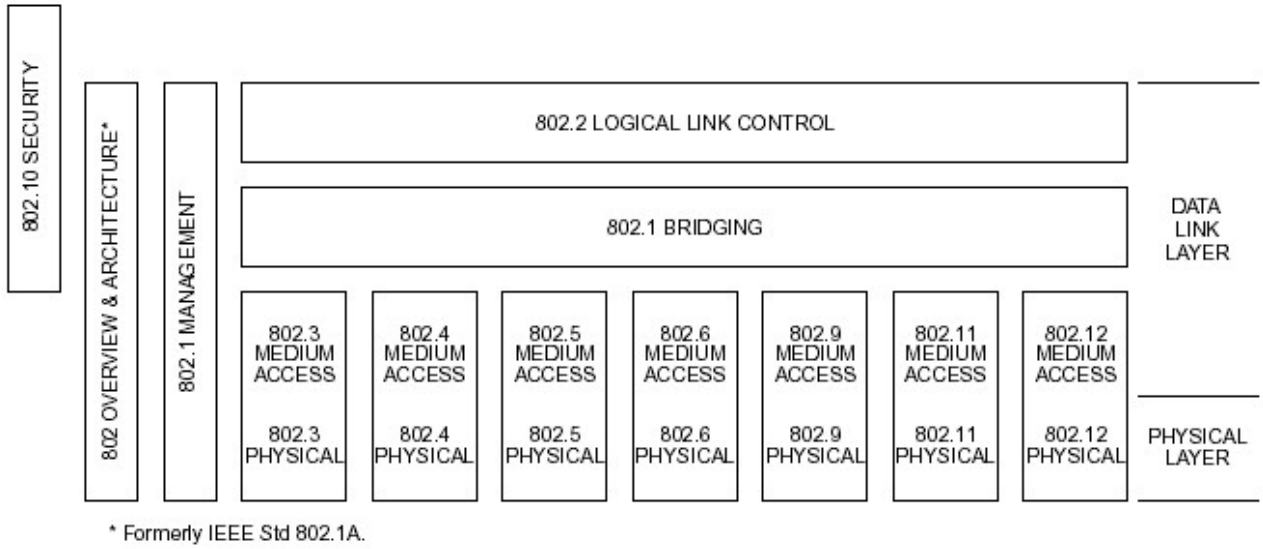

Abbildung 15: Einbindung der 802-Standards in das OSI-Modell, aus [>WLSPEC1]

Übertragungsverfahren

Physikalische Ebene

Für WLAN ist sowohl eine Infrarotübertragung als auch eine Funkübertragung vorgesehen. Im Folgenden wird jedoch ausschließlich die Funkübertragung betrachtet werden. Als Medium wird beim ursprünglichen 802.11 Standard wie bei Bluetooth das 2,4 GHz ISM Band (s. 6.1.1.2) genutzt und die Signale mit einer Spread-Spectrum Technologie übermittelt. Für die Erweiterungen 802.11a und 802.11h sind Bereiche des 5 GHz Spektrums vorgesehen.

Als Spread Spectrum Technologie können FHSS oder DSSS (Direct Sequence Spread Spectrum) eingesetzt werden. Die FHSS-Technologie wurde bereits im Abschnitt Bluetooth (s. 6.1.1.2) beschrieben. Bei DSSS wird das ursprüngliche Signal mit einem Bitmuster (Chipping-Seqenz, Pseudo-Noise-Code) XOR-verknüpft und somit gespreizt und in ein breitbandiges Signal mit geringerer Intensität umgewandelt. Der Empfänger benötigt das gleiche Bitmuster, um das empfangene Signal zu dechiffrieren. Bei DSSS sind höhere Datenraten möglich als bei FHSS (bis zu 11MBit/s). Die 802.11b Erweiterung nutzt DSSS.

Als weitere Übertragungstechnologie ist OFDM (Orthogonal Frequency Division Multiplexing) vorgesehen. Dabei werden die Daten parallel auf mehreren sich überschneidenden Unterkanälen übertragen. Hierdurch sind höhere Datenraten möglich und Störungen, die sich auf kleine Frequenzbereiche beschränken, lassen sich durch Fehlerkorrektur beheben.

Zugriffskontrollebene (Media Access Layer)

Die Zugriffskontrollebene regelt das Versenden von Nachrichten zwischen verschiedenen Stationen, wozu unter anderem die kollisionsfreie Übertragung, der eigentliche Aufbau der Pakete, die Adressierung, Verschlüsselung und Authentifizierung sowie Mechanismen zum Stromsparen gehören.

Für die (weitgehend) kollisionsfreie Übertragung wird das CSMA/CA- (Carrier Sense Multiple Access/ Collision Avoidance) Verfahren verwendet. Bevor gesendet wird, wird überprüft, ob bereits ein anderes Gerät sendet. Aus übertragungstechnischen Gründen ist es bei WLAN nicht möglich, gleichzeitig zu senden und auf Kollisionen zu horchen. Das Auftreten von Kollisionen wird daher nicht erkannt (wie bei Ethernet: Collision Detect) sondern versucht, zu vermeiden (Collision Avoidance).

Beim Senden einer Nachricht wird in jedem Frame die voraussichtliche Dauer der Übertragung mit gesendet. Jede Station trägt diesen Wert in ihren Network Allocation Vector (NAV) ein, der als Timer heruntergezählt wird. Das Senden ist für diesen Zeitraum für andere Stationen gesperrt. Ähnlich wird auch das Hidden-Station Problem gelöst. Hier kann eine Station A, die mit B kommunizieren will nicht feststellen, dass gleichzeitig C an A senden will, wenn sich C nicht innerhalb der Reichweite von A befindet. C sendet nun ein RTS- (Request to

send) Frame an B, das die Übertragungsdauer der Datennachricht enthält. B antwortet wiederum mit einem CTS-(Clear to send) Frame, welches ebenfalls die Dauer enthält und auch von A empfangen wird. A erkennt somit, dass das Medium noch belegt ist.

Kollisionen werden mit diesem Verfahren nicht vollständig ausgeschlossen. Daher wird für jeden Frame vom Empfänger ein Acknowledge gesendet. Wenn dies ausbleibt, wird das Senden des Frames wiederholt.

Für das zeitkritische Versenden eignet sich das CSMA/CA-Verfahren nicht. Für diese Zwecke steht optional ein zusätzliches Übertragungsverfahren, das PCF (Point Coordination Function) zur Verfügung welches jedoch nicht in ad-hoc Netzwerken (s. nächster Abschnitt) verwendet werden kann. Durch das zur Verfügung stellen von Zeitabschnitten durch einen als PC (Point coordinator) agierenden Access Point wird gewährleistet, dass immer nur eine Station gleichzeitig sendet. Dies ist somit eine CF (Contention Free, d.h. wettbewerbsfreie) Übertragungsmethode. Durch das periodische Wechseln von CFP (Contention Free Period) und CP (Contention Period) lassen sich beide Zugriffsverfahren gleichzeitig einsetzen.

Netztopologie

Im 802.11 Standard ist eine Station ein Gerät, das über einen drahtlosen Zugang entsprechend des 802.11 Standards verfügt. Mehrere Stationen können sich zu einem Ad-hoc-Netzwerk (Independent Basic Service Set) zusammenschließen, sofern sie sich innerhalb Reichweite (d.h. in einer Zelle) befinden, um miteinander drahtlos kommunizieren zu können. Ein ad-hoc Netzwerk kann spontan und schnell aufgebaut werden. Ad-hoc Netzwerke sind einfach zu konfigurieren und sie benötigen nichts außer den beteiligten Stationen. Der Nachteil ist, dass sie eine begrenzte Reichweite haben und keinerlei Verbindung zu bestehenden Netzwerken.

Ein Infrastruktur-Netzwerk (Extended Basic Service Set) kann aus mehreren Zellen bestehen, die jeweils von einem Access Point verwaltet werden. Die Zellen sind über ein weiteres Netzwerk, das Verteilungssystem (DS: Distribution System), miteinander verbunden. Dies kann z.B. ein Ethernet-Netzwerk sein, so dass sich die kabellose Geräte leicht in ein bestehendes Netzwerk einfügen lassen. In einem Infrastruktur-Netzwerk stehen Dienste zur Verfügung, die ein Ad-hoc Netzwerk nicht bietet (Distribution System Services). Dazu gehört das An- und Abmelden bei einem Access Point, Roaming und die Verteilung von Nachrichten zwischen den Access Points.

In einer Zelle muss nicht jede Station mit jeder anderen Station kommunizieren können. Gleichzeitig müssen nicht Stationen, die sich gegenseitig in Reichweite befinden, einer Zelle angehören. Zellen können sich überlappen, was einer Station das Wandern von Zelle zu Zelle erleichtert (Roaming). Hierfür meldet sich die Station bei ihrem Access Point ab und beim neuen Access Point an.

Um einen Access Point zu finden bietet sich entweder aktives oder passives Scanning an. Beim aktiven Scanning sendet die suchende Station Probe-Frames, die vom Access Point beantwortet werden. Beim passiven Scanning wartet die Station darauf, ein Beacon-Frame vom Access Point zu empfangen.

Sicherheitsmechanismen

Der 802.11 Standard bietet, wie Bluetooth auch, Mechanismen zur Authentifizierung und Verschlüsselung. Die im 802.11 Standard vorgesehene Wired Equivalent Privacy (WEP) ist mittlerweile aus verschiedenen Gründen kritisiert worden:

1. Ein (symmetrischer) Schlüssel wird über längere Zeit im gesamten Netz verwendet: fehlendes Key Management (s. [>BTKR]).
2. Der eingesetzte RC4-Verschlüsselungsalgorithmus erzeugt bei 1 von 256 Schlüsseln sogenannte weak keys, die ein Angriffspotential bieten (genaueres s. [>WLROOS])

Die Angriffsmöglichkeit wurde nicht nur theoretisch nachgewiesen sondern auch praktisch erprobt (s. [>WLFLMASH]).

Um WEP zu ersetzen wurden mehrere Ansätze entwickelt. Einer davon ist die Erweiterung 802.11i, die jedoch bislang noch nicht verabschiedet wurde (vorrausichtlich Ende 2003). Die Wi-Fi Alliance hat dagegen versucht Wi-Fi Protected Access (WPA), das eine Teilmenge von 802.11i darstellt, als Übergangslösung zu etablieren (s. [>WLHEIN1]). WPA nutzt folgende Sicherheitsmechanismen ([>WLWIFI]):

1. Verwendung des Temporal Key Integrity Protocols (TKIP) für die Verbesserung der Verschlüsselung
2. Benutzerauthentifizierung durch 802.1x und das Extensible Authentication Protocol (EAP, s. [>WLEAP]) über einen zentralen Authentication Server (z.B. RADIUS). Dies beinhaltet z.B. die

gegenseitige Authentifizierung. Bei WEP war es so, dass sich eine Station bei einem Access Point authentifizieren musste, aber nicht umgekehrt.

Geräte lassen sich durch ein Softwareupgrade WPA-kompatibel aufrüsten.

Abgesehen von den hier erwähnten Mechanismen gibt es auch die Möglichkeit, von 802.11 unabhängige Sicherheitsmechanismen wie ein Virtual Private Network (VPN) einzusetzen.

Stromsparmechanismen

Der 802.11 Standard sieht vor, dass sich einzelne Stationen in einem Stromsparmodus befinden können. In diesem Fall speichert der Access Point in einem Infrastrukturnetzwerk die Nachrichten, die an diese Station gerichtet wurden zwischen und kann sie zu einem späteren Zeitpunkt an die Station schicken, wenn diese aktiv wird. Diese Möglichkeit steht wegen des fehlenden Access Points in einem Ad-hoc Netzwerk nicht zur Verfügung.

802.11 Erweiterungen

Im Folgenden soll ein kurzer Überblick über die teilweise bereits im Einzelnen erwähnten 802.11 Erweiterungen gegeben werden:

802.11b: Higher-Speed Physical Layer Extension in the 2.4 GHz Band

802.11b, auch WiFi (Wireless Fidelity) genannt, ermöglicht eine Übertragungsrate von bis zu 11 MBit/s mit DSSS im 2,4 GHz Frequenzband. 802.11b ist zur Zeit noch am weitesten verbreitet.

802.11a: High-speed Physical Layer in the 5 GHz Band

802.11a nutzt Teile des 5 GHz Frequenzbandes unter Verwendung von OFDM und erreicht damit eine maximale Bruttoübertragungsrate von 54 MBit/s. Dieser Standard hat eine stärkere Widerstandskraft gegenüber Störungen. 802.11a wurde nach 802.11b verabschiedet und ist nicht abwärtskompatibel.

802.11g: Further Higher Data Rate Extension in the 2.4 GHz Band

802.11g nutzt wie 802.11b das 2,4 GHz Frequenzband allerdings anders als 802.11b unter Verwendung von OFDM oder DSSS. Mit OFDM sind bis zu 54MBit/s möglich. Es hat eine Reichweite von 30-50m und ist abwärtskompatibel zu 802.11b.

802.11h: Spectrum and Transmit Power Management Extensions in the 5 GHz Band in Europe

Dieser Standard erreicht eine maximale Datenrate von 54 MBit/s im Frequenzband 5 GHz mit OFDM und einer Reichweite von 30-50m. Er erweitert 802.11a um Dynamic Frequency Selection (DFS) und Transmit Power Control (TCP). DFS, das auch von HiperLAN/2 eingesetzt wird, ermöglicht das Erkennen und Ausschließen von bereits belegten oder gestörten Frequenzen. Hierdurch soll gewährleistet werden, dass WLAN im 5 GHz Bereich andere Funkdienste (Radar, Erdbeobachtungssatelliten) nicht stört. Dies ist in Deutschland von der Regulierungsbehörde für Telekommunikation und Post (RegTP) in den Nutzungsbedingungen für das 5 GHz Band für die meisten Anwendungen vorgeschrieben. Gleichzeitig vorgeschrieben ist auch TCP, welches eine automatische Sendeleistungssteuerung ermöglicht ([>WLHEIN2]).

802.11i: Security Enhancements

Wie bereits erwähnt soll mit 802.11i WEP durch einen besseren Sicherheitsmechanismus basierend auf TKIP und AES ersetzt werden.

802.11e: Quality of Service Enhancements

Mit 802.11e werden Erweiterungen für Quality of Service (QoS) und Streaming Multimedia Verfahren in den Standard integriert.

6.1.2.3. Geräte und Anwendungen

WLAN findet sich in PDAs, Notebooks, PCs und Druckern, ist als PC-Card oder PCI-Karte oder auch als externer Adapter, der über den USB- oder Ethernet-Stecker angeschlossen wird, erhältlich. Bereits integriert ist ein WLAN Access Point in einigen DSL-Routern / DSL-Modems. Dies vereinfacht den klassischen Aufbau

für den Heimbereich: DSL-Modem, WLAN-Access Point, Desktop-PC sowie weitere Geräte mit WLAN-Karte. Auf dem Markt erhältlich sind auch Geräte für die Vernetzung von WLAN mit Powerline (Ethernet über Stromkabel). Mit der WANDA-Technogie (Wireless Any Network Digital Assistant) von Texas Instruments wird die Integration von WLAN, Bluetooth und GSM in PDAs erleichtert.

Eine Möglichkeit unterwegs ins Internet zu kommen bieten mittlerweile öffentliche Internetzugänge für WLAN, sogenannte Hot Spots, die bereits jetzt in Städten wie Hamburg, Berlin und München zu finden sind.

6.1.2.4. Markt und Zukunft

Mit 8 Millionen verkaufter Chipsätze in 2001 und über 20 Millionen verkaufter Chipsätze 2002 sowie prognostizierten 150 Millionen im Jahr 2007 befindet sich der Markt für WLAN wie auch der Markt für Bluetooth im Wachstum. Zur Zeit sieht es auch nicht danach aus, dass eine dieser Technologien die andere ausstechen könnte. An einigen Stellen wird angemerkt, dass WLAN aber die Verbreitung von UMTS gefährden könnte. WLAN ist im Vergleich mit UMTS zur Zeit schneller und billiger [>WLHEM2].

Literaturreferenzen zu [Kapitel 6.1.2]

[>WLSPEC1] ANSI/IEEE Std 802.11, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999

[>WLNETT] E Nett, M. Mock, M. Gergeleit, Das drahtlose Ethernet, Addison-Wesley, 2001

[>WLLEHN] F. Lehner, Mobile und drahtlose Informationssysteme, Springer, 2003

[>WLROOS] A. Roos, A class of weak keys in the RC4 stream cipher, http://marcel.wanda.ch/Archive/WeakKeys

[>WLFLMASH] S Fluhrer, I. Mantin, A. Shamir, Weaknesses in the Key Scheduling Algorithm of RC4, http://citeseer.nj.nec.com/cache/papers/cs/24059/http:zSzzSzwww.crypto.comzSzpaperszSz.zSzotherszSzrc4_ksaproc.pdf/fluhrer01weaknesses.pdf

[>WLHEIN1] Verbesserung für WLAN-Sicherheit, Heise News, 31.10.2002, http://heise.de/newsticker/result.xhtml?url=/newsticker/data/ea-31.10.02-000/default.shtml&words=Verbesserung%20f%FCr%20WLAN%20Sicherheit

[>WLHEIN2] Konsens für erweitertes 5-GHz-WLAN in USA, 05.02.2003, http://heise.de/newsticker/data/ea-05.02.03-000/

[>WLHEM1] Hamburg startet bundesweit größtes nichtkommerzielles WLAN-Projekt, Heise Mobil, 13.12.2002, http://www.heise.de/mobil/newsticker/meldung/33066

[>WLHEM2] Anwendungsentwickler zweifeln an UMTS-Erfolg, Heise Mobil, http://www.heise.de/mobil/result.xhtml?url=/mobil/artikel/2002/02/18/wlanumts/default.shtml&words=WLAN

[>WLWIFI] Wi-Fi Alliance, Wi-Fi Protected Access, 2002, http://www.wi-fi.org/OpenSection/pdf/Wi-Fi_Protected_Access_Overview.pdf

[>WLEAP] RFC 2284 - PPP Extensible Authentication Protocol (EAP), 1998, http://www.faqs.org/rfcs/rfc2284.html

[>WLRUE] B. Rüdt, Drahtlos glücklich, PCtip, Mai 2003

[>WLCT231] Stromfunkter, c't 23/2003, S. 23

6.1.3. UMTS

6.1.3.1. Entstehung

UMTS (Universal Mobile Telecommunication System), eine Mobilfunktechnologie der dritten Generation (3G), wurde von verschiedenen Firmen und Interessengruppen in Zusammenarbeit mit ETSI (European Telecommunications Standards Institute) und dem 3GPP (3rd Generation Partnership Project) ab 1989 entwickelt. Es stellt eine Weiterentwicklung von Mobilfunksystemen der zweiten Generation (2G), insbesondere von GSM, dar, wobei GPRS (General Packet Radio Services) und EDGE (Enhanced Data Rates for GSM Evolution) als evolutionäre Zwischenschritte angesehen werden. UMTS bietet im Vergleich zu GSM zusätzliche Dienste mit höherer Qualität und verbesserten Sicherheitsmechanismen. 1996 wurde ein „Memorandum of Understanding of the Introduction of UMTS“ definiert. Im März 1999 waren die Standards für UMTS Phase 1 weitgehend fertiggestellt.

Die UMTS-Lizenzen, d.h. einzelne Frequenzbänder, wurden in Deutschland im August 2000 von sechs verschiedenen Telekommunikationsanbietern zur Nutzung ab 2003 erworben. Die flächendeckende Einführung von UMTS wird jedoch innerhalb Deutschlands auch Ende 2004 noch nicht abgeschlossen sein. Vodafone kündigt jetzt an, 2004 mit dem UMTS-Massengeschäft zu starten. O2 nennt als Stichtag für den Start die Cebit 2004 [>UMHN1]. Auch die Verfügbarkeit von UMTS-fähigen Geräten ist noch sehr schwach.

6.1.3.2. Technologie

Übertragungstechnologie

Für UMTS-terrestrisch wurden die Frequenzbereiche 1920-1980 MHz, 2110-2170 MHz (paired) und 1900-1920 MHz, 2010-2025 MHz (unpaired) definiert. Für die Satellitenkomponenten wurden 1980-2010 MHz und 2170-2200 MHz vorgesehen. Weitere Frequenzbereiche wurden definiert, stehen aber bislang in den meisten Ländern noch nicht zur Verfügung.

Für die Übertragungstechnik stehen zwei Technologien zur Verfügung, die nebeneinander existieren können:

1. UTRA-FDD (paired): Es wird FDD zur Richtungstrennung und WCDMA als Zugriffsverfahren eingesetzt. FDD bedeutet Frequency Division Duplex und verwendet verschiedene Frequenzen für unterschiedliche Richtungen im Duplexbetrieb. WCDMA (Wideband-CDMA) ist ein hybrides Verfahren aus einer Kombination von CDMA und FDMA, d.h. die Kommunikation unterschiedlicher Teilnehmer wird durch verschiedene Codes (CDMA) und verschiedene Frequenzen (FDMA) voneinander getrennt.
2. UTRA-TDD (unpaired): Es wird TDD zur Richtungstrennung und TD-CDMA als Zugriffsverfahren verwendet. TDD bedeutet Time Division Duplex und nutzt verschiedene Zeitscheiben für unterschiedliche Richtungen. TD-CDMA ist eine Kombination aus TDMA (Zeitscheibenmultiplex) und CDMA (Codemultiplex).

Datenraten bis zu 2MBit/s sind möglich.

Netztopologie

Bei UMTS ist eine hierarchische Zellstruktur vorgesehen, die sich aus Makrozellen (300m - 10km), Mikrozellen (100m - 300m) und Pikozellen (bis 100m) zusammensetzt.

Architektur

Vereinfacht gesehen besteht UMTS aus drei Kernkomponenten:

1. User Equipment (UE) / mobiles Endgerät: beinhaltet Funktionen zur Funkübertragung sowie Schnittstellen zum Teilnehmer und die USIM (User SIM) mit Informationen zur Verschlüsselung und Authentisierung
2. UMTS Terrestrial Radio Access Network (UTRAN) / Ultra-Netz: ist das Zwischenstück zwischen UE und CN und beinhaltet u.a.
   - Zugriffssteuerung (Admission Control) der Verbindungen zur Vermeidung von Überlast
   - Verschlüsselung des Funkkanals
   - Steuerung der Verbindungsübergabe bei einem Zellwechsel.
3. Core Network (CN) / Kernnetz: ist aufgeteilt in kanalvermittelnde Dienste (Circuit Switched Domains, CSD), welche im wesentlichen den Diensten des GSM-Netzes entsprechen, und paketvermittelnde Dienste (Packed Switched Domains, PSD), welche den Diensten des GPRS-Netzes entsprechen. Die existierenden Netze können mit einigen Erweiterungen für UMTS wiederverwendet werden.

6.1.3.3. Geräte und Anwendungen

Für UMTS sind eine Vielzahl von Anwendungen denkbar. Bereits aus der Möglichkeit eines Internet-Zugangs ergeben sich einige Anwendungsbeispiele, wobei es sich bei einigen Punkten noch um Visionen und nicht um gängige Praxis handelt. Als Geräte bieten sich typischerweise Mobiltelefone an, insbesondere mit Zusatzfunktionalitäten wie eingebaute Kamera oder Organizer („Smartphone“).

- Mobile Multimedia Messaging (MMS): Textnachrichten mit eingebetteten Bildern und Klängen. Lässt sich z.B. auch für das Verschicken von Fotos verwenden. Dies ist besonders interessant für Mobiltelefone mit eingebauter Kamera. Diese Technologie ist bereits relativ weit verbreitet (Beispiel: Nokia 5140, Nokia 7600).
- Zugang zu Informationen wie
  - Wetterinformationen, -vorhersagen
  - Nachrichten
  - Gelbe Seiten
  - Sportergebnisse
  - Fahrpläne, Verkehrsinformationen
  - Lokale Informationen, z.B. Restauranttips, nächste Tankstelle
  - Börsenkurse
  - Zugriff auf persönliche Informationen (z.B. Kalender) aus dem Büro
- Herunterladen / Abspielen von
  - Musik (Beispiel: Siemens U15)
  - Videoclips, z.B.
    - Kinotrailer
    - Sportclips
  - Programmen, z.B.
    - Spielen
- M-commerce
- Verbessertes Telefonieren
  - Videotelefonie (Beispiel: Siemens U15)
  - „Push-to-talk“ (ist bereits bei GPRS verfügbar): Über Push-To-Talk läßt sich eine Verbindung zu einem oder mehreren Push-To-Talk-Handys aufbauen. Die Verbindung bleibt bestehen, es wird jedoch nur die reine Sprechzeit abgerechnet, die durch einen Tastendruck initiiert wird. ([>UMHN2]). (Beispiel: Nokia 5140)
- Vernetzte Applikationen
  - Spiele
  - Chat

6.1.3.4. Markt und Zukunft

2002 waren ca. 60% der Bevölkerung von Deutschland gleichzeitig Benutzer von Mobilfunksystemen, wobei dieser Prozentsatz nach gängigen Prognosen weiter ansteigen wird. Als zur Zeit populärste mobile Technologie hat GSM einen Nutzerstamm von 850 Millionen weltweit. Aus diesem Grund sind die Vorraussetzungen für die Verbreitung von weiterführenden Technologien in diesem Bereich denkbar günstig, vorrausgesetzt das Preis-Leistungsverhältnis wird von den Anwendern akzeptiert.

Bis Mitte 2003 wurden 120 Lizenzen im IMT-2000 Spektrum vergeben, alle bis auf 2 nutzen UMTS/WCDMA-Technologie ([>UMWPME]).

Verbesserungen am UMTS-Standard sind bereits in Arbeit. Dazu gehört die Verbesserung der WCDMA Technologie durch High Speed Downlink and Uplink Packet Access (HSDPA) wodurch Geschwindigkeiten bis 14,2 Mbit/s (oder andere Quellen: 10 Mbit/s) möglich werden. Eine weitere Verbesserung ist das IP Multimedia Subsystem (IMS), welches Echtzeit-Multimediadienste sowie mehrere Dienste per Session ermöglicht.

Längerfristig ist eine größere Interoperabilität mit Digital Video Broadcasting (DVB) sowie Digital Audio Broadcasting (DAB) geplant. An DVB-Handheld (DVB-H) basierend auf DVB-Terrestrisch (DVB-T) wird bereits gearbeitet, es gibt jedoch noch einigen Forschungsbedarf und die Notwendigkeit, den Stromverbrauch weiter zu drosseln [>UMHN3].

UMTS muss sich durchaus in Konkurrenz mit anderen Technologien noch behaupten, so ist der Endkunde bei WLAN im Geschwindigkeitsvorteil gegenüber UMTS, oder im Kostenvorteil beim einfachen Austausch von Daten gegenüber dem Multimedia Messaging Service [>UMCT603]. UMTS wird insbesondere im europäischen und japanischen Markt verbreitet sein. In den USA dominiert dagegen bereits das in Konkurrenz zu GSM stehende IS-95. Als 3G-Technologie wird sich dort voraussichtlich CDMA2000 durchsetzen, welches zu UMTS nicht kompatibel ist [UMCT802].

Literaturreferenzen zu [Kapitel 6.1.3]

[>UMLEHN] F. Lehner, Mobile und drahtlose Informationssysteme, Springer, 2003

[>UMWPME] Mobile Evolution, UMTS Forum, August 2003, http://www.umts-forum.org/servlet/dycon/ztumts/umts/Live/en/umts/MultiMedia_PDFs_UMTS-Forum-White-Paper-1-August-2003.pdf

[>UMFOR] http://www.umts-forum.org

[>UM3GPP] http://www.3gpp.org (enthält ausführliche Liste der Spezifikationen für GSM und UMTS sowie die Spezifikationen selbst)

[>UMHN1] Vodafone verspricht für 2004 UMTS-Massengeschäft in Deutschland, Heise News, 18.11.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/jk-18.11.03-004/default.shtml

[>UMHN2] Push-To-Talk-Handy von Nokia, Heise News, 12.11.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/rop-12.11.03-000/default.shtml

[>UMHN3] DVB-Standard für Handys und PDAs nimmt Formen an, Heise News, 28.08.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/vza-28.08.03-001/default.shtml

[>UMCT603] D. Zivadinovic, Spaßfunker auf dem Vormarsch, c't, 6/2003, Heinz Heise Verlag

[>UMCT802] J. Steuer, M. Meincke, P. Tondl, c't 8/2002, Heinz Heise Verlag

6.1.4. EIB

6.1.4.1. Einleitung

Der EIB (European Installation Bus) ist ein System für die Haus- und Gebäudeautomation. EIB ist eine Technik zum Schalten, Dimmen, Steuern, Regeln, Messen, Überwachen etc. von elektrischen Anlagen und Systemen. Typische Anwendungen sind dezentrale Temperaturregelung für verschiedene Räume zur Energieoptimierung , Steuerung von Lichtszenarien, zeit- und lichtabhängige Jalousiensteuerung, Sicherheits- und Überwachungssysteme.[DIE99,ROS00]

Der EIB trennt klar die Übertragung von elektrischer Energie und Information. Die Verbraucher werden direkt mit elektrischer Energie versorgt. Alle Verbraucher, Sensoren und Aktoren werden über ein zusätzliches informationsübertragendes Netzwerk miteinander verbunden, über das der gesamte Informationsaustausch stattfindet.

Die Zuordnung zwischen Betätigungselement und Verbraucher ist über das Netzwerk programmierbar und beliebig oft veränderbar.
Für den Betrieb wird kein zentrales Busmanagement benötigt. Die Verwaltung und Speicherung aller Daten erfolgt dezentralisiert in den einzelnen Busgeräten. Jedoch stehen sämtliche Informationen des EIB-Netzes bei Bedarf an jedem Punkt lückenlos zur Verfügung, so dass eine Integration der Elektroinstallation in ein zentralisiertes Überwachungs- und Steuerungssystem ohne zusätzliche Installationskosten möglich ist. Der Grundgedanke besteht darin, dass sich das Gesamtsystem für unterschiedliche Konfigurationen frei programmieren lässt und damit für alle Änderungen am Gebäude flexibel ausgelegt ist.

Der EIB ist durch die EIBA (Europäische Installation Bus Association) standardisiert.[EIB03]. Dies ermöglicht die Kombination verschiedener Geräte unterschiedlicher Hersteller.

6.1.4.2. Technologie

Die Technologie des EIB wurde auf die folgenden Anforderungen aus der Gebäudeautomatisierung optimiert:

- Vielzahl an Knoten
- Kurze Reaktionszeiten
- Geringe Nutzdatenmenge
- Wartungsfreundlichkeit

Das EIB-System besteht aus:

- Sensoren (z.B. Taster, Windmesser), die Befehle in Form von Telegrammen erzeugen.
- Aktoren (z.B. Schaltrelais für Licht, Jalousien), welche die empfangenen Telegramme in Aktionen umsetzen und
- einer Busleitung, die alle Sensoren und Aktoren für den Telegrammverkehr miteinander verbindet.

Übertragung

EIB Übertragung ist über folgende Wege möglich [EIB98]:

- „Twisted Pair“ Leitung: Übertragung der Daten über 2 verdrillte Leiter. Die Übertragungsgeschwindigkeit beträgt 9.600 bit/s. die Busleitungen dürfen beliebig verzweigt werden und benötigen keine Abschlusswiderstände. Der EIB wird mit Sicherheitskleinspannung SELV (Safety Extra Low Voltage; ungefährliche Berührungsspannung) über eine EIB-Spannungsversorgung mit integrierter Drossel versorgt. Die Teilnehmer sind bis minimal 21 V betriebsbereit und entnehmen dem Bus 150 mW, bei zusätzlichen Strombedarf im Endgerät (z.B. LEDs) bis zu 200 mW.
- Powerline: Die Übertragung erfolgt durch überlagerte Signale auf der 230-V-Leitung. Die Übertragungsgeschwindigkeit beträgt 1.200 bit/s. bzw. 2400 bps
- Hochfrequenz: Kabellose Übertragung durch Funk der Frequenzen 433MHz und 868MHz.
- EIB.net (10Mbps on Ethernet): Mit diversen Produkten lassen sich EIB-Daten über Ethernet in Netzwerken wie dem Internet oder ISDN übertragen. Die Übertragungsgeschwindigkeit beträgt 10Mbps.

Obgleich alle diese Übertragungsmedien für den EIB seitens der EIBA (European Installation Bus Association) spezifiziert wurden, versteht man generell unter EIB die „Twisted Pair“-Leitung, die parallel zum 230 V Stromnetz verlegt ist. Diese Technologie wird in allen Neuinstallationen verwendet. Die Übertragung per Powerline und Hochfrequenz dient lediglich der kostengünstigen Nachinstallation in bestehenden Gebäuden.

Bei der Twisted-Pair-Verkabelung sind Aktoren und Sensoren mittels dieser Zweitdrahtleitung miteinander verbunden. Diese Leitung überträgt sowohl die Spannungsversorgung für die Elektronik der Teilnehmer als auch die Information. Die Information wird auf dem Adernpaar symmetrisch übertragen. Deshalb dürfen die Adern nicht geerdet werden. Der Teilnehmer wird durch die Spannungsdifferenz an den beiden Adern ausgesteuert. Störeinstrahlungen wirken auf beide Adern mit der gleichen Polarität und beeinflussen daher die maßgebende Differenz der Signalspannung nicht.

Die Sensoren und Aktoren einer EIB-Installation kommunizieren mittels Telegrammen. Beim Drücken eines Lichtschalters (Sensor) wird z.B. ein Telegramm mit dem Befehl "Licht ein" erzeugt und auf den Bus gesendet. Alle Adressaten erhalten und verarbeiten das Telegramm. Z.B. setzt ein Schaltaktor das Telegramm in einen Einschaltbefehl der Leuchte um.
Falls das Telegramm korrekt empfangen wurde, wird eine Quittierung an den Absender geschickt. Falls nicht, wird das Telegramm noch maximal drei weitere Male gesendet, bis der Vorgang beendet ist.

Um die Kompatibilität gleichartiger Geräte verschiedener Hersteller zu sichern, wurden für verschiedene Anwendungen Datentypen und deren Kodierungen spezifiziert, die sogenannten EIS (EIBA Interworking Standards)-Typen (siehe Tabelle 2).[EIB98]

EIS-Typ	Funktion
1	Schalten
2	Dimmen
3	Uhrzeit
4	Datum
5	Zahl mit Nachkommaanteil
6	Relativwert (0-100%)
7	Antriebssteuerung
8	Zwangssteuerung
9	Zahl mit Nachkommaanteil (IEEE Float)
10	Ganze Zahl (0-85535)
11	Ganze Zahl (0-4294967295)
12	Zugangskontrolle
13	Ein ASCII-Zeichen
14	Zähler (0-255)
15	Zeichenkette (max. 14 Zeichen)

Tabelle 2: Datentypen des EIBA Interworking Standards (EIS).

Netztopologie

Eine EIB-Installation besteht aus maximal 15 Bereichen, auch Hauptlinien genannt. Diese Bereiche werden mittels sogenannter Bereichskoppler über ein „Backbone“ miteinander verbunden. Jede Hauptlinie setzt sich aus maximal 12 Linien zusammen. Jede dieser Linien kann 63 Geräte enthalten. Durch Verwendung von Linienverstärkern lässt sich diese Kapazität auf bis zu 255 Geräte aufstocken. Damit sind in einem EIB-System über 14.000 Teilnehmer möglich.[SCH03]

Bei Anlagen mit sehr hoher Belastung des Backbones (z.B. bei Visualisierungen oder Kommunikation mit Managementebenen) ist es möglich, die Hauptlinien mittels entsprechender Gateways auch über Ethernet zu koppeln. So können mehr Daten transportiert werden. Die Übertragungsrate ergibt sich in diesem Fall durch den verwendeten Ethernet-Standard. Auf diese Weise können auch Anlagen mit weit mehr als 14.000 Teilnehmern erstellt werden.

Installation und Kommunikation

Die EIB-Installation erfordert die EIB Tool Software (ETS). Die ETS wird von der EIBA in Zusammenarbeit mit Softwarehäusern und EIB-Herstellern entwickelt und vertrieben.

Jeder EIB-Teilnehmer erhält bei der Installation durch Parametrierung mit der ETS eine eindeutige Adresse im System. Diese besteht aus

1. Bereichsnummer,
2. Liniennummer und
3. Teilnehmernummer.

Diese physikalische Adresse wird ein einziges Mal über den Bus vom PC in den Teilnehmer geladen, wobei beim entsprechenden Teilnehmer eine kleine Programmiertaste gedrückt werden muss. Anschließend können einzelne Geräte, z.B. zur Programmierung, über diese physikalische Adresse adressiert werden.

Im eigentliche EIB-Betrieb erfolgt die Kommunikation jedoch über sogenannte Gruppenadressen. Man kann diese Gruppenadresse auch als Namen einer globalen verteilten Variable verstehen. Gruppenadressen können wie folgt strukturiert werden:

1. Hauptgruppe mit Wertebereich 0-15
2. Mittelgruppe mit Wertebereich 0-7
3. Untergruppe mit Wertebereich 0-255

Aktoren können mehreren Gruppenadressen angehören, Sensoren jedoch nur an eine Gruppenadresse senden. Z.B. sendet ein Lichtschalter bei Betätigung ein Telegramm an eine Gruppenadresse, z.B. 2/0/3. Dieser Gruppenadresse 2/0/3 können mehrere Lampen gleichzeitig angehören, die dann alle geschaltet werden. Man spricht deshalb auch von 1-n- oder Multicast-Kommunikation.

Es ist die Aufgabe des EIB-Installateurs bzw. Systemintegrators, für die jeweilige Installation die optimalen Gruppenadressen festzulegen und zu verteilen.

Neben der Verteilung von physikalischen und Gruppenadressen, dient die ETS der Festlegung von Programmen und Parametern von EIB-Geräten. Sie kann Parameter und Programme in die Geräte laden. Durch diese Programme legt sich häufig erst das Verhalten eines Aktors oder Sensors fest, z.B. ob er als Lichtschalter oder Lichtdimmer agiert.

Protokollarchitektur

Der EIB ist ein vertikales System, dass alle Schichten des OSI-Referenzmodells abdeckt.[EIB98]

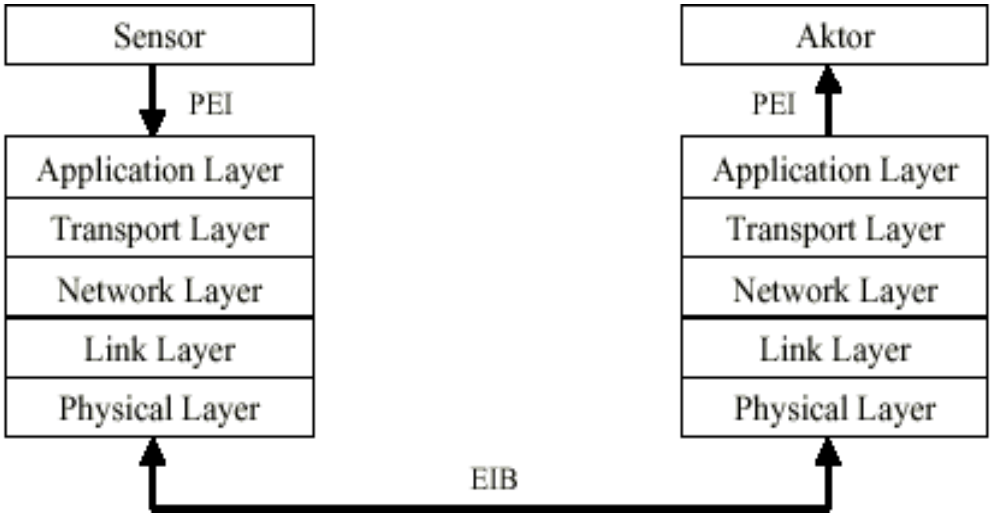

Abbildung 16: Protokollarchitektur des EIB.

6.1.4.3. Marktposition

Die internationale European Installation Bus Association (EIBA) wurde am 8. Mai 1990 als Zusammenschluss von europäischen Herstellern von Installationsgeräten und -systemen gegründet. Der Gründung folgte die Einrichtung einer Geschäftsstelle in Brüssel sowie 15 nationaler EIBA-Gruppen als Kontaktzentren und Interessenvertretungen. Inzwischen besteht die EIBA aus 113 weltweit tätigen Herstellerfirmen.

Seit Gründung der EIBA am 8. Mai 1990 wurden

- weltweit über 100.000 Gebäude mit dem EIB ausgerüstet,
- in über 50 Ländern weltweit über 12.000 EIB Tool Software-Lizenzen vergeben,
- mehr als 6000 Produktgruppen zertifiziert,
- weltweit 70 geprüfte Schulungszentren eingerichtet,
- weltweit mehr als 70.000 Elektroinstallateure in der EIB-Technik ausgebildet,
- über 4.200 EIB-Partner für Planung, Installation und Service im Handwerk gewonnen
- und weltweit über 10 Millionen Knoten installiert.

Zunehmend werden EIB-Anlagen mit anderen Systemen der Gebäudeautomation verbunden. Dies geschieht über sogenannte Bridges, die das EIB-Protokoll und das andere Protokoll gegenseitig übersetzen. Momentan existieren Bridges unter anderem für ISDN, GSM, Ethernet, WLAN, Jini, Bluetooth und UPnP.
Alle wichtigen Hersteller von Gebäudeautomationssystemen bieten heute solche Einbindungen an. Deutliche Zunahme hat auch die Kopplung von EIB zum offenen Standard Building Automation and Control Network BACnet erfahren.
Grosse Bedeutung, gerade auch im Heimbereich, wird die Verbindung zwischen EIB und Internet erlangen. Momentan sind am Markt schon Systeme erhältlich, mit denen man von unterwegs z.B. interessante Daten wie die Raumtemperatur überprüfen und ggf. einstellen kann.

KNX

Im May 1999 gründeten 9 Mitglieder der drei Organisationen

- BatiBUS Club International (BCI) [BCI03],
- European Installation Bus Association (EIBA) und
- European Home Systems Association (EHSA) [EHS03]

die Konnex Association mit dem Ziel einen einzigen Konvergenzstandard „KNX“ als Ersatz für die drei Standards der Ursprungsorganisationen zu begründen [KNX03]. Mittlerweile repräsentiert die Konnex Association mehr als 100 Mitglieder. Im May 2002 wurde die Version 1.0 des Standards KNX fertiggestellt [KNX02]. Der Standard KNX basiert im wesentlichen auf dem EIB und wurde um die Konfigurationsmöglichkeiten und Medien des BatiBUS und EHS erweitert. Momentan wird an einer Erweiterung des Standards zur Version 1.1 gearbeitet, die dann im Leistungsumfang ungefähr dem aktuellen Stand der EIB Spezifikation entspricht und z.B. die Schnittstellen zu IP-Netzwerken behandelt.

Literaturreferenzen zu [Kapitel 6.1.4]

[>BCI03] BatiBUS Club Internatinal: http://www.batibus.com/anglais/gen/index.htm, 2003.

[>DIE99] D. Dietrich, W. Kastner, T. Sauter (Hrsg.): EIB Gebäudebussystem, Hüthig Verlag 1999.

[>EIB98] EIB Association: EIB System Handbook, Brüssel, 1998.

[>EIB03] EIB Association: www.eiba.org, 2003.

[>EHS03] EHS Association: www.ehsa.com, 2003.

[>KNX02] Konnex Association: KNX specification, Brüssel, 2002

[>KNX03] Konnex Association: www.konnex.org, 2003

[>ROS00] M. Rose, W. Kriesel, J. Rennefahrth: EIB für die Gebäudesystemtechnik in Wohn- und Zweckbau, Hüthig Verlag 2000.

[>SCH03] Rainer Schilling, Deutsche EIB/KNX Gruppe: www.konnex-eiba.de, 2003

6.1.5. IEEE1394

6.1.5.1. Rahmenbedingungen

IEEE 1394, auch Firewire (Apple Trademark) oder i.LINK (Sony Trademark) genannt, ist ein serielles Übertragungsmedium, das ursprünglich von Apple entwickelt wurde. 1995 entstand daraus der IEEE 1394 Standard.

IEEE 1394 sieht Geschwindigkeiten von ca. 100, 200 oder 400 MBit/s und eine Kabellänge von bis zu 4,5 m zwischen den einzelnen Geräten vor. Mit der Spezifikation 1394b, die 2002 verabschiedet wurde, sind Geschwindigkeiten von ca. 800 und 1600 MBit/s möglich.

IEEE 1394-Kabel sind entweder 4polig (2x2 Adern für die Datenübertragung) oder 6-polig (2x2 Adern für die Datenübertragung und 2 Adern für die Stromversorgung). Der IEEE 1394b Standard verwendet eine ganz andere Signalkodierung und andere Pegel, den so genannten Beta-Mode. Es gibt Ports, die den neuen Betamodus und den alten Standard unterstützen. Diese Ports heißen bilinguale Ports. Die neuen Beta- und Bilingualen Ports sind 9-polig, bei dem bilingualem Port passen jedoch auch die „alten“ 4- bzw. 6-poligen Stecker.

Die Geräte handeln automatisch aus, welcher Modus verwendet wird.

Der IEEE-1394b Standard erlaubt auch den Einsatz von anderen Kabeln so z.B. CAT-5-UTP-Kabel (mit bis zu 100m Länge und einer Geschwindigkeit von bis zu 100 MBit/s) oder auch Kunststoff- oder Glasfaserkabel.

Medium	**S100**	**S200**	**S400**	**S800**	**S1600**
Geschwindigkeit					
Shielded Twisted Pair	-	-	4,5 m	4,5 m	4,5 m
CAT-5 UTP	100 m	-	-	-	-
Plastic Optic Fiber	50 m	50 m	-	-	-
Hard Polymer Clad Fiber	100 m	100 m	-	-	-
Multimode Fiber	-	-	100 m	100 m	100 m

Tabelle 3: Kabellängen im Beta-Mode, aus [>HEISE_102003]

Der IEEE 1394-Standard sieht sowohl asynchrone als auch isochrone Übertragungen (isochron bedeutet, dass eine bestimmte Datenrate garantiert wird, und somit die Übertragung von zeitabhängigen Daten ermöglicht wird [>IEEE1394_TATT]) vor und ist somit gut für Audio- und Videodatenströmen geeignet.

Während beim IEEE 1394-Protokoll nur eine Halbduplex-Übertragung möglich war, sieht der 1394b-Standard die Übertragung im Vollduplexverfahren vor [>HEISE_102003].

6.1.5.2. Netztopologie

Bis zu 63 Geräte können an einen Bus angeschlossen werden. Die einzelnen Geräte eines Netzwerkes sind dabei als Baum organisiert, d.h. jedes Gerät hat nur einen Parent und es gibt nur einen Wurzelknoten. Die Organisation wird bei jedem Bus-Reset neu ausgehandelt. Dies ist z.B. der Fall, wenn ein neues Gerät eingesteckt wird (Hot-Plugging). Physikalisch werden die Geräte immer Punkt-zu-Punkt verbunden.

Damit es nicht zu Kollisionen kommt, genehmigt der Wurzelknoten immer nur einem Gerät das Senden von Paketen [>HEISE_102003].

6.1.6. Einsatzmöglichkeiten

Eingesetzt wird IEEE 1394 hauptsächlich in Camcordern, Digitalen Kameras, externen Laufwerken (z.B. Festplatten, DVD-Brenner) und PCs.

Unter Windows XP, Linux oder Mac OS X ist es möglich, die IEEE 1394-Adapter als Netzwerkadapter zu konfigurieren und über IP über Firewire zu kommunizieren [>HEISE_192003].

Literaturreferenzen zu [Kapitel 6.1.5]

[>HEISE_102003] H. Bögeholz, J. Schuster, FireWire prescht vor, c't 10/2003, Heinz Heise Verlag, http://www.heise.de/ct/03/10/166/
[>RFC_2734] IPv4 over IEEE 1394, http://www.ietf.org/rfc/3.txt
[>HEISE_192003] O. Lau, Renn-LAN, c't 19/2003, Heinz Heise Verlag
[>IEEE1394_TA] http://www.1394ta.org/
[>IEEE1394_TATT] http://www.1394ta.org/Technology/About/TechTalk.htm

6.2. Geräte

6.2.1. Heim-PC

Auch wenn der Desktop-PC in Visionen wie „Ambient Intelligence“ und „Pervasive Computing“ keine zentrale Rolle mehr spielt, gibt ein STAR über Computer für den Heimbereich wichtige Aufschlüsse darüber, welche Technologien sich in diesem Bereich bereits durchgesetzt haben oder durchsetzen werden.

Tabelle 4 vergleicht beispielhaft zwei aufgrund ihrer großen Bedeutung für den Heimbereich „Volks-PC“ genannte Computer von Plus ([>SCHR03]) und Aldi ([>MEDION03]) mit einem Heim-PC vom weltweiten Marktführer Dell ([>DELL03]) und einem der ersten Media-Center PCs von Fujitsu Siemens ([>MM03]).

Hersteller / Vertrieb	Dell	Fujitsu Siemens / Media Markt	MBO / Plus	Medion / ALDI
Typbezeichnung	Dimension 8300	Scaleo 600 P 306	Plus P.C.V.O.2800+	Titanium MD 8080 XL
Preis / Euro	1099	1099	999	1179
CPU	Intel Pentium 4, 3.0 GHz	Intel Pentium 4, 3.0 GHz	AMD AthlonXP 2800+, 2083 MHz	Intel Pentium 4, 3.0 GHz
Festplatte	120 GByte	200 GByte	160 GByte	160 GByte
Hauptspeicher (RAM)	1024 MByte	512 MByte	512 MByte	512 MByte
Grafikspeicher	128 MByte	128MByte	128 MByte	128 MByte
Netzwerk	10/100 Mbit/s Ethernet	10/100 Mbit/s Ethernet	10/100 Mbit/s Ethernet 22 Mbit/s WLAN	10/100 Mbit/s Ethernet 54 Mbit/s WLAN Bluetooth
Anschlüsse	PS 2 8 x USB 2.0 Firewire optional	PS 2 RS232 4 X USB 2.0 2 x Firewire	PS 2 RS232 5 x USB 2.0 2 x Firewire	PS 2 RS232 7 x USB 2.0 2 x Firewire

	TV-out: S-Video, Composite VGA DVI SPDIF	TV-out: S-VHS TV-in: S-VHS, Composite VGA DVI SPDIF FM Antenne TV Antenne	TV-out: S-Video, Composite VGA SPDIF	TV-out TV-in VGA SPDIF
Laufwerke	Diskette optional CD DVD+R/+RW	Diskette CD DVD+R/+RW/-R/-RW	Diskette CD DVD+R/+RW/-R/-RW	CD DVD+R/+RW/-R/-RW
Betriebssystem	WindowsXP Home Edition	WindowsXP Media-Center-Edition	WindowsXP Home Edition	WindowsXP Home Edition
Sonstige Hardware	Modem optional	Fernbedienung 6-fach Card-Reader (MS/SD/MMC/CF/MD/SM) V9x Modem MPEG2-Encoder	6-fach Card-Reader (MS/SD/MMC/CF/MD/SM) V92/V90 Modem	Fernbedienung V9x Modem

Tabelle 4: Vergleich vier beispielhafter PCs für den Heim-Bereich ([>SCHR03,DELL03, MEDION03, MM03]).

Heute gehören

- 100Mbit/s-Ethernet,
- USB2.0 und
- Firewire

ebenso zur Standardausstattung wie

- CD/ DVD-Brenner und
- Grafikkarten mit TV-Ausgang.

Auch kabellose Netzwerktechnologien

- Bluetooth und
- WLAN

findet man immer häufiger direkt im PC integriert, insbesondere in den sogenannten Volks-PCs. Für den Datenaustausch mit mobilen Geräten wie Digitalkameras oder MP3-Playern stehen häufig Universal-Lesegeräte für Speicherkarten zur Verfügung. Folgende Formate werden oft unterstützt:

- CompactFlash (CF)
- SmartMedia (SM)
- Memory Stick (MS)
- MultiMediaCard (MMC)
- SD Card (SD)
- Micro Drive (MD)
- xD Picture Card

Bemerkenswert ist die zunehmende Ausstattung mit Fernbedienungen, die verpflichtender Bestandteil von Windows-Media-Center PCs sind (siehe Abschnitt 6.3.2.1).

Literaturreferenzen zu [Kapitel 6.2.1]

[>DELL03] Dell, http://commerce.euro.dell.com/dellstore/config/frameset.asp?s=dedhs&l=de&m=eur&c=726&n=4534&cu=dedhs&v=d&cc=&ogn=&kcd=&ad=&mc=&rs=&cuid=&cg=&pch=1&pn=-1&demo=&gc=&sbc=dedhs&co=&b=46942À, Dezember 2003.

[>MEDION03] Medion: http://www.medion.de/content.html, Dezember 2003.

[>MM03] Media Markt: http://shop.mediamarkt.de/webapp/wcs/stores/servlet/ProductDisplay?productId=30512& catalogId=5000&langId=-3&storeId=5000&categoryId=10002, Dezember 2003.

[>SCHR03] Georg Schnurer: Volks-PC IV, ct' 20, S. 22, 2003.

6.2.2. PDA

PDA steht für Personal Digital Assistent. Die Begriffe PDA, Handheld, Palmtop und Pocket Computer werden meist nicht klar voneinander abgegrenzt verwendet. In der Regel handelt es sich um Geräte, in etwa in Größe einer Handfläche, die annähernd die Funktionalität eines PCs bei geringem Ressourcenverbrauch bieten. Der Begriff Pocket PC ist gleichzeitig die Bezeichnung für ein Betriebssystem von Microsoft auf der Basis von Windows CE, das häufig auf diesen Geräten vorinstalliert ist sowie die Bezeichnung für Geräte, die mit diesem Betriebssystem ausgeliefert werden [>WKPP].

An mitgelieferter Software gibt es typischerweise PIM-Applikationen (Personal Information Management) wie Terminplaner, Kalender, Adressbuch etc. Bei einigen „High-End"-Geräten sind auch Office-Applikationen wie Textverarbeitung, Tabellenkalkulation, außerdem PDF-Reader, E-Book-Reader sowie Webbrowser und Emailapplikationen im Lieferumfang enthalten. Eine mitgelieferte Wireless-Anbindung über IrDA, Bluetooth oder WLAN 802.11 gehört auch häufig dazu. Ein komfortabler Austausch der Daten zwischen PDA und PC (Synchronisieren der Daten) wird häufig benutzerfreundlich über eine Docking-Station oder Wireless angeboten. Die Eingabe erfolgte klassischerweise beim Palm mit dem Stift (Eingabe vordefinierter handschriftähnlicher Zeichen), mittlerweile sind auch viele Geräte mit einer kleinen Tastatur anzutreffen.

Sehr weit verbreitet sind die iPAQ-Geräte von HP sowie die PDAs von Palm. Als Betriebssystem wird auf den neueren HP iPAQs „Microsoft® Windows® Mobile 2003 Premium software for Pocket PC" eingesetzt, bei den Palms das gleichnamige Palm-OS. Im zweiten Quartal 2003 machten auf Palm-OS basierende PDAs weltweit 51,4 % aller ausgelieferten PDAs aus, wobei Pocket PC PDAs 35,9 % erreichten. Der Palm Zire 71 war der meistverkaufte PDA im gleichen Zeitraum. In den U.S.A. erreicht Sony nach Palm 12,1 % Markanteil im PDA-Geschäft und ist dort die Nummer zwei [>JS2003]. HP ist nach eigenen Angaben weltweit der Marktführer in Bezug auf Marktanteile im Pocket PC Geschäft [>HP2003] und die Nummer zwei im PDA-Geschäft.

Geräte, die bereits mit Linux ausgeliefert werden, haben eine relativ geringe Marktrelevanz, es sollen jedoch einige hier der Vollständigkeit halber mit aufgeführt werden. Für die Unterstützung von Linux auf den HP iPAQs gibt es ein (oder auch mehrere) Projekt(e) (s.u.), das Gleiche gilt für den Palm. Ein besonderes Augenmerk verdienen im Rahmen von DynAMITE die Entwicklungswerkzeuge. Auch hier werden einige kurz beschrieben, es lässt sich jedoch nicht das gesamte kommerziell und frei verfügbare Angebot komplett abdecken.

6.2.2.1. iPAQ

Vom HP iPAQ Pocket PC, der zu den meistverkauften Pocket PCs gehört, gibt es eine ganze Reihe von unterschiedlichen Modellen mit verschiedenen Features. Die Modelle h5150 sowie h5550 bieten z.B. Biometric Security, d.h. Autorisierung über Fingerabdruck. In der folgenden Tabelle wird der h4355, ein Gerät der mittleren bis höheren Preisklasse, exemplarisch vorgestellt.

Hersteller	HP (http://www.hp.com)
Typbezeichnung	iPAQ h4355
Preis	US-Preis $499,99
CPU	400 MHz Intel XScale
Festspeicher	32 MB Flash ROM
RAM	64 MB SDRAM
Zusatzspeicher	SD memory card slot, SD/MMC
Display	3,5" TFT 64000 Farben
Betriebssystem (ausgeliefert)	Microsoft Windows Mobile 2003 Premium for Pocket PC
weitere Betriebssysteme verfügbar	
mitgelieferte Software	
Wireless	Bluetooth, WLAN 802.11b, Infrarot
Schnittstellen	Secure Digital Card slot, SDIO, SD, MMC
Texteingabe	Tastatur oder Stift
Ausmasse	
Gewicht	5,8 ounces
Stromversorgung	wiederaufladbare 1560mAh Litium-Ionen Batterie, austauschbar
Sonstiges	Mikrofon, Lautsprecher

Quelle: http://www.hp.com

6.2.2.2. Yopy

Der Yopy YP 3700 PDA von G.Mate wird mit der Embedded Linux Distribution "Linupy" ausgeliefert

Hersteller	G-Mate (http://www.gmate.com)
Typbezeichnung	Yopy YP 3700
Preis	
CPU	206 MHz 32-bit Intel StrongARM RISC Prozessor
Festspeicher	32 MB Flash
RAM	128 MB RAM
Zusatzspeicher	Multimedia Card slot (für MMC Karten) Compact Flash II slot
Display	3,5“ TFT, 320x240, 65,536 Farben
Betriebssystem (ausgeliefert)	„Linupy“ Linux Distribution
weitere Betriebssysteme verfügbar	
mitgelieferte Software	PIM suite (Personal Information Manager), beinhaltet Terminkalender, Backupprogramm etc., MP3-Spieler
Wireless	IrDA
Schnittstellen	Lautsprecheranschluss, Mikrofonanschluss, 3.5mm „ear phone plug“, RS-232C/USB (client)
Texteingabe	Miniaturtastatur, Touchscreen-Tastatur sowie Handschriftenerkennung
Ausmasse	2.7“ x 4.1“ x 1.0“
Gewicht	7 ounces
Stromversorgung	
Sonstiges	Client Programm für Windows für Backup etc. (über Docking-Station)

Quelle: http://www.gmate.com

6.2.2.3. Sharp Zaurus

Es sind nicht sämtliche Sharp Zaurus-Geräte in Deutschland erhältlich, der hier aufgeführte sollte es jedoch sein.

Hersteller	Sharp
Typbezeichnung	Zaurus SL-5500G (http://www.zaurus.de)
Preis	
CPU	StrongARM ® SA-1110 206 MHz
Festspeicher	16 MB Flash
RAM	64 MB SDRAM
Zusatzspeicher	Karteneinschübe für SD und CF Typ II Karten
Display	3,5” TFT-Farb-LCD 240x320 Pixel, 65536 Farben mit Vordergrundbeleuchtung
Betriebssystem (ausgeliefert)	Linux ® 2.4 (Embedix™), Qtopia™, Personal Java™ (Jeode)
weitere Betriebssysteme verfügbar	
mitgelieferte Software	PIM-Applikationen (Kalender, Aufgabenliste, Adressbuch etc.), MP-3 Player, MPEG-4 Movie Player, Webbrowser Opera™, Email-Client, Image viewer, Voice Recorder (Headset erforderlich)
Wireless	WLAN, Bluetooth, GSM, GPRS, Infrarot
Schnittstellen	Stereo-Kopfhörer, Audio Input (Mono)
Texteingabe	ausziehbare QWERTY-Tastatur, Touch-Display mit Stift
Ausmasse	74x138x18mm
Gewicht	194g (inkl. Batterie und Stift, ohne Display-Schutz)
Stromversorgung	Lithium-Ionen Batterie 950 mAh (austauschbar), Batterie-Ladegerät optional

Sonstiges	Synchronisation über Docking-Station, Intellisync für MS Outlook® und Palm Desktop

Quelle: http://www.sharp.de/php/td.php?par=17_no_no_107

6.2.2.4. Palm

Die umfangreiche Produktpalette an PDAs von Palm kann hier nicht im Einzelnen ausführlich dargestellt werden. Es werden die wichtigsten Daten eines High-End Gerätes, des Tungsten-C, exemplarisch aufgelistet:

Hersteller	Palm-One (http://www.palmone.com)
Typbezeichnung	Tungsten C
Preis	
CPU	Intel Xscale 400 MHz ARM Prozessor
Festspeicher	16 MB Flash
RAM	64 MB RAM
Zusatzspeicher	SD/MultiMediaCard Erweiterungs-Slot
Display	Transflexiv-Bildschirm, 320x320, 16-bit Farbtiefe
Betriebssystem (ausgeliefert)	Palm OS 5.2.1
weitere Betriebssysteme verfügbar	Unbekannt
mitgelieferte Software	Kalender, Adressbuch, Aufgabenliste, Merkzettel, Notizen, Hotsync (TM), Web Browser, VPN-Client, Word-, Excel- und PowerPoint Dateien editieren, Palm Photos (Photo Viewer), Palm Reader, Sprachnotiz, Acrobat Reader, Netzwerk HotSync, PPTP, Email VersaMail, Synchronisation mit Microsoft Outlook
Wireless	Infrarot, 802.11b
Schnittstellen	2,5 mm Kopfhörer (Mono), Lautsprecher, Palm Universal Connector
Texteingabe / Bedienung	Minitastatur, Buchstabenerkennung Graffiti 2
Ausmasse	12,2 x 7,8 x 1,7 cm
Gewicht	178g
Stromversorgung	Wiederaufladbare Lithium-Ionen-Polymer-Batterie (bis zu 6 Tage Betrieb)
Sonstiges	

Quelle: http://www.palm.com, http://www-5.palmone.com/de/de/products/tungsten-c/Tungsten_C_DS_de.pdf, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/dal-23.04.03-000/default.shtml

6.2.2.5. Microsoft Entwicklungsumgebung

Zwecks Verständnis der unterschiedlichen Microsoft Betriebssystemversionen für PocketPC / Smartphone empfiehlt es sich, folgende Webseite zu studieren:

„Comparison of Windows CE .NET 4.2, Pocket PC 2002, and Windows Mobile 2003 Software for Pocket PCs", Microsoft Corporation, August 2003, http://www.windowsfordevices.com/articles/AT5612706156.html

Als Anlaufstelle bietet sich außerdem die Windows Mobile Developer Home-Seite http://www.microsoft.com/windowsmobile/information/devprograms/default.mspx sowie die Newsgroups microsoft.public.pocketpc.developer.* an.

Information zu den erforderlichen Entwicklungsumgebungen findet sich unter „Developing Applications for Windows Mobile: FAQ", Microsoft Corporation, August 2003, http://msdn.microsoft.com/mobility/default.aspx?pull=/library/en-us/dnppcgen/html/devmobfaq.asp

Für die Entwicklung unter C++ sind folgende Tools erforderlich:

- Microsoft® eMbedded Visual C++® 4.0. Die Entwicklung erfolgt unter Windows 2000 oder Windows XP. Das Paket ist per Download (http://www.microsoft.com) kostenlos erhältlich.
- Microsoft® eMbedded Visual C++® 4.0 Service Pack 2, per Download kostenlos erhältlich
- Pocket PC 2003 Software Development Kit, per Download kostenlos erhältlich

6.2.2.6. Microsoft .NET Compact Framework

Diese Bibliothekssammlung erleichtert und beschleunigt die Entwicklung von insbesondere Benutzungsoberflächen und XML-Web-Services für PDAs und Smartphones. Es unterstützt Pocket PC 2000, Pocket PC 2002, Windows Mobile 2003-basierte Pocket PCs sowie Embedded Systeme unter Windows CE .NET 4.1 und später. Das Framework benötigt 1,35 MByte RAM auf dem Zielsystem.

Weitere Informationen:
http://msdn.microsoft.com/mobility/prodtechinfo/devtools/netcf/
http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/anw-01.04.03-000/default.shtml

6.2.2.7. Palm OS Entwicklungssystem

Palm OS bietet ein kostenloses Entwicklungssystem für die Crossentwicklung für den Palm. Erhältlich sind

- Palm OS® Software Development Kit Version 5
- Palm OS Simulator: Mit dem Simulator läuft das Palm Betriebssystem über einem Device Abstraction Layer (DAL) auf der Entwicklungsumgebung.
- Palm OS Emulator: Der Palm OS Emulator emuliert die Hardware der verschiedenen Palm PDAs auf dem Entwicklungssstem.

Palm OS ist außerdem offen für .NET.

Weitere Informationen:
http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/dal-14.05.02-000/default.shtml
http://www.palmos.com

6.2.2.8. Java

J2ME (Java 2 Platform Micro Edition) ist für einige PDAs verfügbar. Für die Entwicklung steht der Java 2 Platform Micro Edition Wireless Toolkit für Windows, Linux und Solaris zur Verfügung. Sun One Studio, Mobile Edition, eine IDE von Sun enthält ebenfalls den J2ME Wireless Toolkit. Insbesondere Smartphones unterstützen häufig CLDC (Connected Limited Device Configuration), eine J2ME Konfiguration für kleine Handhelds, und MIDP (Mobile Information Device Profile), eine J2ME API, die festlegt, wie die Schnittstelle für Applikationen auf Mobiltelefonen aussieht. Applikationen, die MIDP unterstützen, werden MIDlets genannt ([>WEBO]).

Weitere Informationen:
CLDC: http://www.jcp.org/en/jsr/detail?id=30
MIDP: http://www.jcp.org/en/jsr/detail?id=37
J2ME: http://java.sun.com/j2me/

6.2.2.9. Java für Palm OS

Für einige Palm-Modelle steht eine Java Virtual Machine der WebSpere Serie kostenlos zur Verfügung. Unterstützte Modelle sind zur Zeit der Tungsten C, E, W, T2 und T3. Die Virtual Machine unterstützt die J2ME sowie die CLDC und MIDP Standards ([>HEISE2003]).

6.2.2.10. Metrowerks Entwicklerkit

Von Metrowerks ist ein Entwicklerkit mit Namen OpenPDA erhältlich, das die Applikationserstellung für PDAs, die auf Linux und Java basieren, erleichtern soll. Es unterstützt die Prozessoren ARM, ARM (XScale), MIPS, SH und X86. Als Referenzplattformen werden der Sharp Zaurus SL-5500, Sharp Zaurus SL-A300 Personal Mobile Tool, Sharp Zaurus SL-5600, Sharp C700 sowie der AMD Alchemy™ Solutions Mobile Client RDK aufgeführt. Mit Metrowerks CodeWarrior ist auch die Crossentwicklung unter Windows möglich. Die CodeWarrior Development Tools für Sharp Zaurus basieren ebenfalls auf der OpenPDA Plattform.

Weitere Informationen:
http://www.metrowerks.com

6.2.2.11. Familiar für iPAQ (Linux)

Das Familiar Projekt v 0.7.2 ist eine Linux Distribution für die iPAQ Modelle H3100, H3600, H3700, H3800, H3900, H5400 und H5500.

Weitere Informationen:
http://familiar.handhelds.org

6.2.2.12. Qtopia

Qtopia von der Firma Trolltech ist eine Applikationsplattform basierend auf Qt/Embedded (ein Applikationsentwicklungsframework basierend auf Qt, ebenfalls von Trolltech) für Embedded Linux Geräte wie PDAs, Mobiltelefone und Smartphones. Qtopia enthält ein Windowssystem, ein Entwicklungssystem, ein Synchronisierungsframework, Unterstützung für verschiedene Eingabemethoden, Java Integration, Wireless Support sowie diverse Applikationen. Es wird inklusive Source Code ausgeliefert. Für Applikationsentwickler wird ein Preis von US $195 angegeben (ohne Source-Code, ohne Cross-Compiler), für OEMs gibt es Konditionen auf Anfrage.

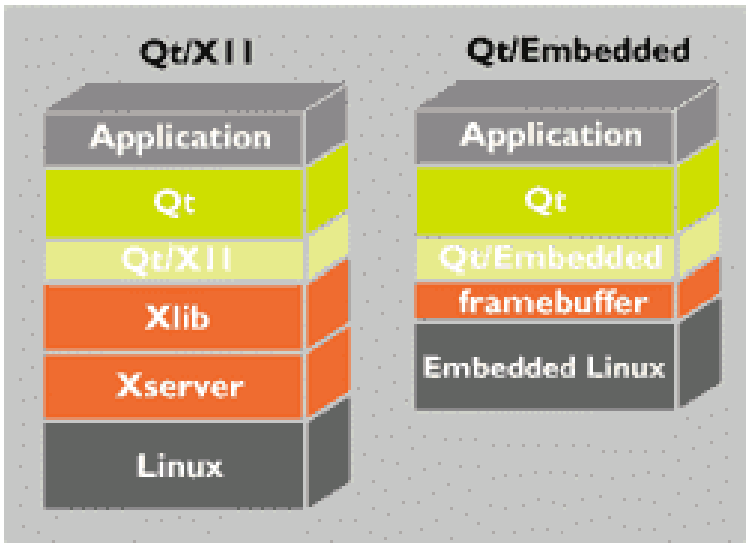

Abbildung 17: Unterschied zwischen Qt undQt/Embedded, aus http://www.trolltech.com

Weitere Informationen:
http://www.trolltech.com

Literaturreferenzen zu [Kapitel 6.2.2]

[>JS2003] J. Sundgot, Palm OS drives handheld market, 19.08.2003, http://www.infosyncworld.com/news/n/3963.html
[>HP2003] HP Comments on Recent Handheld Devices Market Share Data, HP News Release, 27.10.2003, http://www.hp.com/hpinfo/newsroom/press/2003/031027b.html
[>WKPP] Wikipedia, Pocket PC, http://en.wikipedia.org/wiki/Pocket_PC
[>HEISE2003] Java für Palm PDAs, Heise News, 01.10.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/dal-01.10.03-000/default.shtml
[>WEBO] http://www.webopdia.com

6.2.3. Smartphone

Ein Smartphone ist weitestgehend eine Kombination aus Mobiltelefon und PDA. Es beinhaltet in der Regel eine zusätzliche Schnittstelle zu einem PC, z.B. über Bluetooth [>WKSP]. Auf diesen Geräten verbreitete Betriebssysteme sind Symbian, Windows CE, Palm OS und Linux. Symbian wurde 1998 als Firma gegründet

und gehört Ericsson, Nokia, Panasonic, Psion, Samsung Electronics, Siemens und Sony Ericsson. Das erste open Symbian OS basierte Mobiltelefon kam 2001 auf den Markt.

Nach einer Prognose von Gartner Anfang 2003 [>GO2003] wird das Jahr 2003 das Jahr sein, in dem die Betriebsystemkämpfe für Smartphones beginnen. Symbian und Microsoft sollen dabei eine zunehmende Konkurrenz von Linux zu spüren bekommen. Motorolla kündigte im August 2003 an, in den eigenen Smartphones Linux einzusetzen, was einige als schweren Schlag für Symbian und Microsoft werteten ([>HM82003]). Im September 2003 stellten Motorolla und Microsoft jedoch das gemeinsam entwickelte Smartphone MPx200 vor [>HN2003]. Mit dem älteren Microsoft-Smartphone SPV von HTC waren viele Kunden unzufrieden. Zu den Kritikpunkten zählten Abstürze und verzögertes Auslösen der Kamera. Software-Entwickler bemängelten die Beschränkung auf von Microsoft zertifizierte Programme ([>HEISE2002]).

Der Multimedia Messaging Service (MMS) sowie mobile Videoanwendungen sollen sich durchsetzen, was wiederum Endgeräte mit integrierten Kameras zu Gute kommen soll ([>GO2003]).

An Mobiltelefonen sollen nach einer Prognose von Strategy Analytics 2003 insgesamt an die 492 Millionen verkauft werden. Nokia liegt zur Zeit mit einem Marktanteil von 34,5 % ganz vorne, gefolgt von Motorolla mit 15,3 %, Samsung mit 11,4 %, Siemens (8,7 %), LG (5,8 %) und Sony Ericson (5,4%) [>REUTERS2003].

Im europäischen Smartphone-Markt hielt Nokia im zweiten Quartal 2003 78 % Markanteil, gefolgt von Sony Ericsson mit 15 % Marktanteil aber starken Wachstumsraten ([>SMITH2003]). Die Series 60 Plattform für Smartphones (basierend auf Symbian OS) von Nokia ist nach Angaben von Nokia die weltweit führende Smartphone Plattform und wird von Firmen wie Siemens, Samsung, Panasonic, Sendo und Nokia lizensiert.

Im Folgenden werden einige Smartphones sowie verfügbare Entwicklungsumgebungen knapp vorgestellt.

6.2.3.1. Nokia 6600

Hersteller	Nokia (http://www.nokia.com)
Typbezeichnung	6600
Preis	
CPU	
Festspeicher	6 MB verfügbar
RAM	
Zusatzspeicher / Erweiterungskarten	Memory Card Slot
Display	16-bit Farbtiefe TFT, 176x208 Pixel
Betriebssystem (ausgeliefert)	Series 60 Plattform (Symbian OS), Java MIDP 2.0
weitere Betriebssysteme verfügbar	
mitgelieferte Software	RealOne Player (Streaming Video und Audio), MMS, Email (POP3, IMAP4), Kalender, Browser
Kamera	Kamera mit digitalem Zoom, Videoaufnahme
Mobilfunkübertragung	Tri-Band GSM, GPRS, HSCSD
Wireless	Bluetooth, Infrarot
Schnittstellen	
Texteingabe / Bedienung	„5-way joystick navigation“, “predictive text input”
Ausmasse	108.6 x 58.2 x 23.7mm
Gewicht	125g
Stromversorgung	Lithium-Ionen Batterie, im Betrieb 2-4h, Standby 150-240h
Synchronisieren mit PC über	PC Suite, SyncML
Sonstiges	

Quelle: http://www.nokia.com

6.2.3.2. Sony Ericson P900

Hersteller	Sony Ericson
Typbezeichnung	P900
Preis	829 Euro
CPU	
Festspeicher	16 MB verfügbar

RAM	
Zusatzspeicher / Erweiterungskarten	Memory Stick Duo 32 MB liegt bei
Display	208x320 Pixel, 16-bit Farbtiefe
Betriebssystem (ausgeliefert)	Symbian OS 7 mit der UIQ Plattform
weitere Betriebssysteme verfügbar	
mitgelieferte Software	MMS, Video- und Audioplayer (auch MP3), Dokumentenviewer, Organizer, Webbrowser (HTML, cHTML, WML)
Kamera	VGA-Kamera, Fotos und Videos mit Ton (MPEG4)
Mobilfunkübertragung	GSM, GPRS, HSCSD
Wireless	Bluetooth, IrDA
Schnittstellen	
Texteingabe / Bedienung	Touchscreen, Tastenfeld
Ausmasse	
Gewicht	
Stromversorgung	Sprechzeit bis 16 Stunden, Standby bis 450 Stunden
Synchronisieren mit PC über	SyncML (kabellos)
Sonstiges	

Quelle: http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/rop-21.10.03-001/default.shtml

6.2.3.3. Motorolla A760

Hersteller	Motorolla (http://www.motorolla.com)
Typbezeichnung	A760
Preis	
CPU	
Festspeicher	
RAM	
Zusatzspeicher / Erweiterungskarten	
Display	65000 Farben
Betriebssystem (ausgeliefert)	Embedded Linux von Montavista
weitere Betriebssysteme verfügbar	unbekannt
mitgelieferte Software	Verwaltung von Adressen und Terminen, MP3-Player, Multimedia Messenging
Kamera	ja
Mobilfunkübertragung	
Wireless	Bluetooth
Schnittstellen	USB
Texteingabe / Bedienung	Touchscreen, Scrollrad
Ausmaße	
Gewicht	
Stromversorgung	
Synchronisieren mit PC über	secure Over the Air (OTA), Bluetooth, Infrarot oder USB
Sonstiges	

Quelle:http://www.motorolla.com, http://www.heise.de/mobil/newsticker/meldung/39725

6.2.3.4. Treo 600

Hersteller	Handspring (http://www.handspring.com)
Typbezeichnung	Treo 600
Preis	ca. US $600
CPU	144-MHz-ARM-Prozessor OMAP 310 von Texas Instruments
Festspeicher	
RAM	32 MB RAM (24 MB user-available)
Zusatzspeicher / Erweiterungskarten	SD Card, MultiMediaCard
Display	CSTN-Display mit 160 × 160 Pixel, 3375 Farben

Betriebssystem (ausgeliefert)	Palm OS 5.2.1
weitere Betriebssysteme verfügbar	
mitgelieferte Software	Kalender, Organizer, Spiele, Office-Dokument-Leser, Multimedia-Applikationen, POP-3 Email-Applikation, MMS, Webbrowser Blazer™, Bildbetrachter
Kamera	VGA Kamera
Mobilfunkübertragung	Qaud-Band (850/900/1800/1900 MHz) GSM/ GPRS
Wireless	Infrarot
Schnittstellen	
Texteingabe / Bedienung	QWERTY-Tastatur, „5-way button“
Ausmaße	11.2 x 6.0 x 2.2 cm
Gewicht	
Stromversorgung	Wiederaufladbare Lithium Ionen Batterie
Synchronisieren mit PC über	
Sonstiges	

Quelle: http://pressroom.palm.com/InvestorRelations/PubNewsStory.aspx?partner=5150&storyId=98945, http://www.handspring.com, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/anw-23.09.03-000/default.shtml

6.2.3.5. Windows Mobile 2003-based Smartphone Developer Kit

Dieses Entwicklungssystem kostet zur Zeit US $499.00, soll aber nur für eine begrenzte Zeit erhältlich sein. Mitgeliefert wird ein red e SC1100 Smartphone inklusive SIM-Karte, Headset und USB Sync Kabel. Das Entwicklerkit enthält außerdem Visual Studio .NET 2003 Demoversion (auf 60 Tage begrenzt). Für Entwickler, die diese Version noch länger als 60 Tage verwenden wollen, kommen Zusatzkosten dazu:

- Visual Studio .NET 2003 Professional: Preis ca. US $1,079
- Visual Studio .NET 2003 Enterprise Developer: Preis ca. US $1,799
- Visual Studio .NET 2003 Enterprise Architect: Preis ca. US $2,499
-

Des weiteren enthält das Entwicklerkit eine Smartphone 2003 SDK CD (mit eMbedded Visual C++ 4.0) sowie den Microsoft Mobile Development Toolkit. Es beinhaltet außerdem Informationen über das Microsoft Mobile2Market Programm, ein Vermarktungsprogramm für Smartphones und Pocket PCs.

Weitere Informationen:
http://www.microsoft.com/windowsmobile/information/devprograms/smartphonedevkit/default.mspx
http://www.microsoft.com/windowsmobile/information/devprograms/mobile2market/default.mspx
http://msdn.microsoft.com/vstudio/device/embedded/datasheet.aspx

6.2.3.6. Series 60 Plattform

Nach Angaben von Nokia ist Series 60 die führende Plattform für Smartphones. Es basiert auf dem Symbian Betriebssystem und enthält eine Benutzungsoberfläche sowie Programme, die speziell auf Smartphones zugeschnitten sind.

Series 60 beinhaltet die J2ME (Java 2 Platform, Micro Edition) unter Einbeziehung von MIDP und CLDC. Eine Entwicklung in Java oder C++ (Symbian Applikationen) ist möglich.

Weitere Informationen:
http://www.nokia.com/downloads/solutions/mobile_software/Series_60_PO_final.pdf
http://www.nokia.com/downloads/solutions/mobile_software/S60_apps.pdf

6.2.3.7. UIQ

UIQ ist eine Benutzungsoberfläche für das Symbian Betriebssystem. UIQ wird von der Firma UIQ Technology AB entwickelt, die 1999 als Tochterfirma von Symbian Ltd. entstanden ist. Das Sony Ericsson P900 Smartphone wird mit UIQ ausgeliefert. Für Entwickler steht der UIQ SDK kostenlos bereit. Der UIQ Software Development Kit enthält Programme für die Entwicklung von Software für ein UIQ-basiertes Mobiltelefon. Dazu gehört ein Emulator für den PC. Als Programmiersprachen stehen C++ und Java zur Verfügung.

Weitere Informationen:

http://www.uiq.com

6.2.3.8. CodeWarrior Development Studio for Symbian OS

Das CodeWarrior Development Studio for Symbian OS von Metrowerks ist eine Entwicklungsumgebung, mit der Applikationen in C und C++ für das Symbian Betriebssstem entwickelt werden können. Unterstützt werden der Texas Instruments Innovator™, ARM Integrator SPP/2™, Intel® Xscale™ DBPXA250 und einiger Smartphones wie dem Sony Ericsson™ P800.

Weitere Informationen:
http://www.metrowerks.com/MW/Develop/Wireless/Symbian/Default.htm

Literaturreferenzen zu [Kapitel 6.2.3]

[>GO2003] Prognose: 2003 beginnt der Kampf um Smartphone-OS-Herrschaft, 18.02.2003, Golem, http://www.golem.de/0302/24018.html
[>WKSP] Wikipedia, Smartphone, http://en.wikipedia.org/wiki/Smartphone
[>HM82003] Motorola hat sein erstes Linux-Smartphone fertig, Heise Mobil News, 25.08.2003
[>HN2003] Motorola und Microsoft stellen gemeinsam neues Smartphone vor [Update], Heise News, 15.09.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/dz-15.09.03-000/default.shtml
[>HEISE2002] Kunden mit dem Microsoft-Smartphone SPV unzufrieden, Heise Mobil News, 30.11.2002, http://www.heise.de/mobil/newsticker/meldung/32762
[>REUTERS2003] Mobile phone sales near 500 mln in 2003-survey, Reuters Business News, 30.10.2003, http://uk.biz.yahoo.com/031030/80/eckur.html
[>SMITH2003] T. Smith, Euro Q2 smartphone sales sky rocket, The Register, 22.07.2003, http://www.theregister.co.uk/content/68/31880.html

6.2.4. Tablet-PC

Ein Tablet PC ist zum einen ein spezielles Notebook PC, mit dem sich auf dem Bildschirm mit einem Stift schreiben läßt. Der Tablet PC setzt die Schrift mittels Handschrifterkennung um. Des weiteren bezeichnet Tablet PC das Betriebssystem von Microsoft für Tablet PCs [>TPWEBO]. Als Convertibles werden Tablet PCs mit integrierter Tastatur bezeichnet. Durch ein klappbares Display lassen sie sich wie Notebooks (mit Tastatur) oder wie Tablet PCs (mit Stift) bedienen [>HEISE1811_2003]. Der Unterschied zwischen Tablet PCs und den PDAs, die ebenfalls mit einem Stift bedient werden, ist in der höheren Leistungsfähigkeit und Funktionalität, der besseren Handschrifterkennung und auch in Größe und Preis zu sehen. Die Texteingabe per Stift erfolgt in der Regel nicht über einen Touchscreen, sondern über einen elektromagnetischen Digitizer, der nur auf Eingaben des zugehörigen Stifts reagiert. Dies verhindert die versehentliche Eingabe durch Auflegen der Hand. Zudem kann die Eingabe ohne Kontakt des Stiftes mit der Bildschirmfläche erfolgen [>MS_2002].

Im ersten Jahr der Verfügbarkeit von Tablet PCs wurden im Raum Europa, Afrika und Naher Osten fast 100000 Stück verkauft. Damit macht der Tablet PC ca. 1 % des Notebook-Marktes aus. Im Jahr 2007 rechnet Gartner damit, dass 35 % aller Notebooks Pocket-PC Funktionalität haben werden. Allerdings gingen in diesem Jahr die Verkaufszahlen von Tablet PCs vom zweiten zum dritten Quartal um 20 % zurück [>HEISE0611_2003].

Somit reichen die Verkaufszahlen nicht an die Erwartung der Hersteller heran, dass Tablet-PCs einen signifikanten Anteil des Notebook-Verkaufs ausmachen werden. Als ein Hauptgrund hierfür wird der hohe Preis insbesondere im Vergleich mit normalen Notebooks angesehen, z.B. von Zhou Shih-Hsiung, einem Analysten von MIC. Wang Chen-tang, Präsident von Acer, beschuldigt Microsoft dabei einen zu hohen Preis für sein Windows XP Tablet PC Edition zu berechnen [>CNET2310_2003]. Auch Chris Jones von Canalys schlägt vor, dass Microsoft die Preise für das Tablet PC Betriebssystem senken solle [>HEISE3007_2003].

Der führende Tablet PC Hersteller ist (Mitte diesen Jahres) HP mit 30,5 %. An zweiter Stelle liegt Acer mit 22,6 %, gefolgt von Toshiba mit 16,5 % [>HEISE0611_2003].

Viele Hersteller sehen ihre Zielgruppe besonders in Unternehmen und propagieren Anwendungen im Bereich Medizin und Gesundheitswesen.

Zur Zeit dominiert Windows XP Tablet PC Edition als Betriebsystem den Markt der Tablet PCs. Es gibt jedoch auch Linux-Lösungen und Anzeichen dafür, dass Apple mit einem Tablet PC herauskommen wird [>PBS_2003].

6.2.4.1. HP Compaq Tablet PC TC1000

Hersteller	HP (http://www.hp.com)
Typbezeichnung	Compaq Tablet PC TC1000 [DG985A]
Preis	€ 2.573 (ohne Mwst)
CPU	Transmeta Crusoe™ 5800 1 GHz
Festspeicher	Festplatte 30 GB
RAM	256 MB DDR SDRAM (max. 768 MB)
Zusatzspeicher	SODIMM-Steckplatz
Display	10,4" XGA TFT-Weitwinkelbildschirm
Tastatur	ja, abnehmbar
Betriebssystem (ausgeliefert)	Microsoft Windows XP Tablet PC Edition,
weitere Betriebssysteme verfügbar	
mitgelieferte Software	
Wireless	802.11b
Schnittstellen	2x USB 2.0, VGA, RJ-45 NIC, RJ-11 Modem, Stereokopfhörer, Mono-Headset, Mikrofon
Ausmasse	216 x 20 x 274 mm
Gewicht	
Stromversorgung	bis zu 5 Stunden Akku-Laufzeit
Sonstiges	10/100 Mbit Ethernet LAN, Integrierte Stereolautsprecher, integriertes Modem

Quelle: http://www.hp-expo.com/de/ger/commercial/tabletpc/dg985a.html

6.2.4.2. Acer Travelmate C110 Serie

Hersteller	Acer
Typbezeichnung	Travelmate C110 Serie (C111 TCi)
Preis	€ 2499 (inkl. Mwst)
CPU	Intel® Pentium® M 1.0GHz Prozessor
Festspeicher	40GB HDD
RAM	512 MB DDR (aufrüstbar auf max. 2048MB)
Zusatzspeicher	
Display	10.4"" XGA TFT Farbdisplay mit berührungsempfindlicher Oberfläche für Stifteingabe, 1024×768, 16.7M Farben
Tastatur	ja
Betriebssystem (ausgeliefert)	Microsoft® Windows® XP Tablet PC Edition
weitere Betriebssysteme verfügbar	
mitgelieferte Software	
Wireless	1 x FIR (Fast Infrared), Bluetooth (optional)
Schnittstellen	2 x USB 2.0, 1 x IEEE 1394 Firewire, 1 x RJ-11-Anschluß für 56Kbps Modem, 1 x RJ-45-Anschluß für Ethernet, 1 x Mikrofon, 1 x Kopfhörer-out, 1 x CardBus Typ-II Einschub, 1 x Easy-Port-Docking Schnittstelle
Ausmasse	
Gewicht	1.45kg
Stromversorgung	ca. 3 Stunden Laufzeit, ACPI 2.0 Powermanagement
Sonstiges	56Kbps Modem, 10/100Mbps Fast Ethernet, externes DVD-CDRW, 1.44" FDD (extern), Integriertes Mikrofon

Quelle:
http://www.acer.de/acereuro/page4.do?dau22.oid=2963&UserCtxParam=0&GroupCtxParam=0&dctx1=9&ctx1=DE&crc=3811760133

6.2.4.3. Tablet-PCs mit Linux

Element Computer bieten den Helium 2100 Convertable Tablet-PC mit einer Linux Distribution von Lycoris für US $999 an.
Weitere Informationen:
http://www.elementcomputer.com/store/product_info.php?products_id=33
http://zdnet.com.com/2100-1103_2-5112309.html
http://www.lycoris.com/

Von Desktop Evolution gibt es den De-Tablet für US $1900, mit Lycoris Desktop/LX Tablet Edition und basierend auf dem Portege 3500 von Toshiba. Eine Schrifterkennungssoftware soll jedoch erst mit der nächsten Version verfügbar sein.

Weitere Informationen:
http://www.pcworld.com/reviews/article/0,aid,112743,00.asp
http://www.lycoris.com/
http://www.tecchannel.de/news/betriebssystem/13413/

6.2.4.4. Microsoft Windows XP Tablet PC Edition 2004

Microsoft Windows XP Tablet PC Edition ist eine erweiterte Version von Windows XP Professional.

Diese Version beinhaltet laut Microsoft insbesondere eine verbesserte Texteingabe mit dem Stift mit Hilfe des Tablet PC Input Panel (TIP) sowie eine Integration in Office 2003 und Microsoft Office OneNote™ 2003. Es soll auch möglich sein, handschriftliche Emails per Microsoft Outlook zu verschicken.

Die handschriftlichen Eingaben werden als eine Folge von Bezier Kurven abgespeichert. Die Breite und Farbe der „Digital Ink“ Schrift ist auswählbar. Eine Spracherkennung erweitert die Möglichkeiten für den Anwender [>MS_2002].

Weitere Informationen:
http://www.microsoft.com/windowsxp/tabletpc/evaluation/default.asp
http://www.microsoft.com/windowsxp/tabletpc/evaluation/overviews/indepth.asp

6.2.4.5. Microsoft Tablet PC Software Development Kit (SDK)

Diese Entwicklungsumgebung ist in der Version 1.5 zum Herunterladen verfügbar und läuft unter TabletPC, Windows 2000 und Windows XP. Erforderlich ist die Entwicklungsumgebung Microsoft Visual Studio version 6.0 mit Service Pack 5 oder Microsoft Visual Studio .NET.

Weitere Informationen:
http://www.microsoft.com

Literaturreferenzen zu [Kapitel 6.2.4]

[>TPWEBO] Webopedia, http://www.webopedia.com/TERM/t/tablet_PC.html
[>HEISE0611_2003] 100.000 Tablet PCs in Europa verkauft, Heise Mobil News, 06.11.2003, http://www.heise.de/mobil/newsticker/meldung/41785
[>HEISE1811_2003] Der Tablet PC hat Geburtstag, Heise News, 18.11.2003, http://www.heise.de/newsticker/result.xhtml?url=/newsticker/data/jr-18.11.03-000/default.shtml
[>CNET2310_2003] Acer: Tablet PC fees hard to swallow, CNET News, 23.10.2003, http://zdnet.com.com/2100-1103-5095467.html?tag=nl
[>PBS_2003] R. Cringely, Digital Hubris:Apple's Tablet Computer Might Finally Be That Link Between Your PC and TV, PBS, 27.11.2003, http://www.pbs.org/cringely/pulpit/pulpit20031127.html
[>HEISE3007_2003] Marktforscher: Tablet PC braucht mehr Aufmerksamkeit, Heise News, 30.07.2003, http://www.heise.de/newsticker/data/anw-30.07.03-003/

[>MS_2002] Tablet PC: An Overview, Microsoft, 07.11.2002,
http://www.microsoft.com/windowsxp/tabletpc/evaluation/overviews/indepth.asp

6.2.5. A/V Geräte

Die Unterhaltungselektronik vollzieht gerade den Übergang von analogen zu digitalen Technologien. Aus diesem Grund werden sich die erhältlichen A/V-Geräte in der Laufzeit des Projekts DynAMITE erheblich ändern.

Das digitale Speichermedium DVD (MPEG2) hat die analoge Videokassette bereits in den Verkaufszahlen überholt. Auch die digitale DVD- und Festplatten-Rekorder gewinnen zunehmend Marktanteile gegenüber den herkömmlichen analogen Videorekordern. In Berlin wurde 2003 das analoge terrestrische Fernsehen abgeschaltet. Seitdem wird dort terrestrisch nur noch DVB-T (Digital Video Broadcasting - terrestrisch) ausgestrahlt. DVB-T-Inhalte sind ebenfalls in MPEG2 kodiert. Weitere Teile Deutschlands folgen im Jahr 2004. In Kabelnetzen und über Satelliten wird ebenfalls digitales Video ausgestrahlt (DVB-C bzw. DVB-S).

Während die Inhalte also zunehmend in digitaler Form verbreitet werden, nutzen A/V-Geräte untereinander, speziell für die Video-Informationen, fast ausschließlich analoge Übertragungsnormen:

- Das **FBAS Composite** (Farbe Bild Austast Synchron): Diese Standard Übertragungssignal im Videobereich wird von allen A/V-Geräten verstanden. Es ist auch das Signal mit der schlechtesten Qualität, weil hier das Helligkeitssignal Y (Luminance) und das Farbsignal C (Chrominance) zusammen mit dem Syncronisationssignal verschachtelt über eine Leitung übertragen wird. Ausgegeben wird das FBAS Composite Signal entweder über eine gelbe Cinch/RCA Buchse oder über die SCART Verbindungsbuchse.
- Beim **S-Video Y/C** wird Helligkeitssignal Y und das Farbsignal C über zwei Kabel getrennt übertragen. Ausgegeben wird das S-Video Y/C Signal entweder über eine S-Video Y/C-Hosiden Buchse oder über die SCART Verbindungsbuchse.
- Das **RGB** (Rot Grün Blau) überträgt die Farben nach den drei Grundfarben getrennt über 3 Kabel mittels der SCART Verbindungsbuchse, oder in ganz seltenen Fällen zur Projektor Ansteuerung mittels 5 Cinch/RCA oder BNC Buchsen.
- **YUV - YCbCr Component** überträgt die Helligkeit (Y) sowie die Farbdifferenz-Signale U (Rot Differenzsignal: - 0.17 * Rot - 0.33 * Grün + 0.5 * Blau) und V (Blau Differenzsignal: 0.5 * Rot - 0.42 * Grün - 0.081 * Blau) ebenfalls über drei getrennte Kabel. Da die Farbinformation auf der DVD in diesem Farbformat vorliegen, bietet sich diese Art der Übertragung insbesondere für den Anschluss von DVDs an. Ausgegeben wird das YUV Component Signal über drei Cinch/RCA oder BNC Buchsen die dann mit Y, U (Cb/Pb) und V (Cr/Pr) bezeichnet sind.

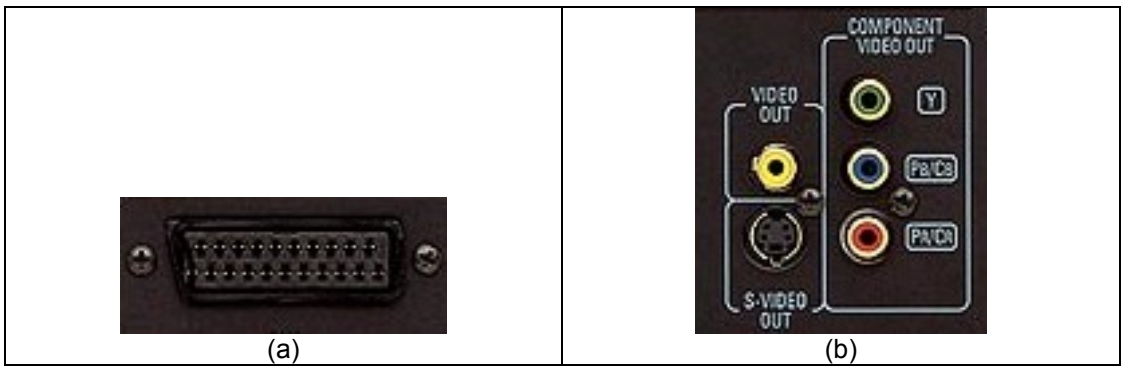

Abbildung 18: (a) Scart-Buchse. (b) S-Video, YUV-Component und FBAS-Composite (VIDEO OUT) Anschlüsse.

Digitale Verbindungstechniken, wie z.B.

- Firewire (IEEE-1394)-Schnittstellen in Verbindung mit DTCP (Digital Transmission Content Protection) als Kopierschutz oder
- das HDMI (High Definition Multimedia Interface) Interface, welches auf der DVI (Digital Visual Interface) Schnittstelle aufsetzt und mit dem HDCP (High-bandwith Digital Content Protection) Kopierschutz ausgestattet ist,

werden aber auch bei der Bild- und Tonübermittlung Einzug halten. Insbesondere HDMI wird aufgrund der Sicherheit und Bandbreite von bis zu 5 GBit/s für die Bild- und Tonübertragung Verwendung finden. Die Bandbreite von Firewire (heute 800 Mbit/s, zukünftig 1600Mbit/s) reicht für unkodierte HDTV(High Definition Television)-Übertragung (2,2 GBit/s) nicht aus. Pioneer und Panasonic bieten bereits erste Fernseher und DVD Player mit HDMI an [>HDMI03].

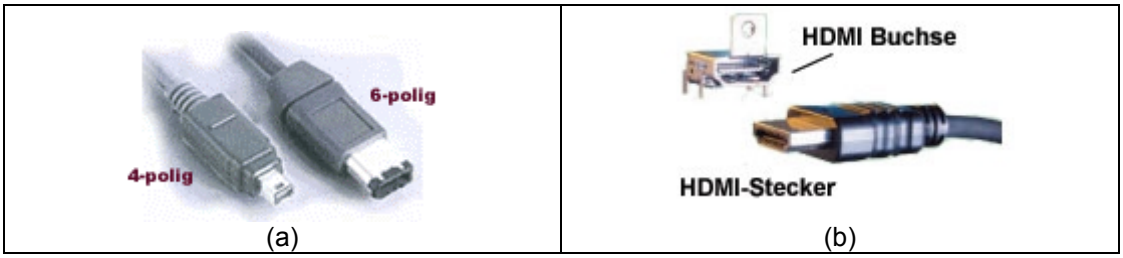

Abbildung 19: (a) Firewire-Stecker mit (6-polig) und ohne (4-polig) Stromübertragung. (b) HDMI-Buchse und Stecker.

Wie die analogen Anschlüsse ist auch HDMI aufgrund der großen Übertragungsbandbreite nicht netzwerkfähig und für DynAMITE wenig interessant.

Mit der zunehmenden Digitalisierung werden auch Netzwerktechnologien während der Projektlaufzeit von DynAMITE zunehmend in A/V-Geräte integriert werden. Durch diese beiden Entwicklungen

- Digitalisierung und
- zunehmende Vernetzung

ändert sich die Projekt-relevante A/V-Infrastruktur.

Zu den Geräten der Unterhaltungselektronik sind inzwischen auch die

- Smartphones,
- Heim-PCs oder
- PDAs

zu zählen, da diese ausgeprägte Multimediafähigkeiten aufweisen und vielfach als Wiedergabegeräte für Audio und Video genutzt werden. Diese Geräteklassen bieten schon jetzt Technologien, wie z.B.

- Verarbeitung verschieden kodierter digitaler Inhalte oder
- Netzwerkunterstützung (Bluetooth, WLAN, Ethernet, Firewire)

die in Unterhaltungselektronikgeräten erst in den nächsten Jahren vermehrt einfließen werden. Deshalb bieten sich diese Geräte für die Realisierung von Prototypen und Demonstrationen im Rahmen von DynAMITE an. Genauer Beschreibungen dieser Geräteklassen findet man in den vorigen Abschnitten dieses Kapitels.

Im Folgenden zählt dieser Abschnitt exemplarisch Unterhaltungselektronik-Geräte auf, die aufgrund ihrer

- netzwerkfähigen Schnittstellen

oder durch ihre

- große Mobilität

den State-of-the-Art bilden, zukünftige Entwicklungen andeuten und interessante Aspekte für das Projekt DynAMITE liefern:

1. **Fernseher mit Netzwerkschnittstelle**: Wenige Hersteller, z.B. Loewe, bieten Fernseher mit Ethernet-Schnittstelle an. Die Fernseher von Loewe können über diese Schnittstellen
   - im Netzwerk gespeicherte Audio- und Video-Dateien abspielen,
   - andere Geräte steuern, oder
   - gesteuert werden.
2. **DVD-Player mit Netzwerkschnittstelle**: Einige Hersteller, z.B. Kiss [>KISS03], bieten DVD-Player mit Ethernet-Schnittstelle an. Der DVD-Player von Kiss kann über diese Schnittstelle Video-, Musik- und Bild-Dateien abspielen bzw. anzeigen, die auf einem PC im Netzwerk gespeichert sind. Allerdings muss dazu auf den PCs spezielle proprietäre Software installiert werden.
3. **Hifi-Anlagen mit Netzwerkschnittstelle**: : Einige Hersteller bieten Hifi-Komponenten mit Ethernetanschluss, z.B. Onkyo einen A/V-Receiver [>ONKYO03]. Diese Geräte können im Netzwerk gespeicherte Dateien oder Internet-Radio abspielen.

Zu diesen drei Gerätetypen sind auch Produkte wie Pinnacle's „Show Center" zu zählen. Dieses verbindet Unterhaltungselektronik-Geräte mit einem PC, indem die mitgelieferte Software Multimedia-Dateien von einem PC über Ethernet oder WLAN zu dem „Show Center" überträgt, das diese dann über analoge Schnittstellen, z.B. Scart oder YUV-Component, auf A/V-Geräten anzeigt.

4. **DVD-Player/Rekorder mit USB oder Firewire-Schnittstellen** werden inzwischen von vielen Herstellern angeboten. Firewire bildet den Vernetzungs-Standard für Digital Video(DV)-Kameras und dominiert deshalb gegenüber USB in der Unterhaltungsindustrie.
5. **Medien Server**, die im Gegensatz zu den bisher genannten Lösungen nicht nur auf im Netzwerk gespeicherte Daten zugreifen können, sondern dem Netzwerk auch Daten zur Verfügung stellen, werden im Audio-Bereich bereits als Produkt angeboten. Im Videobereich befinden sich Systeme für den Heimbereich, im Gegensatz zu Lösungen für professionelle Anwender wie Fernsehstationen [>PANASONIC03], i.A. noch im Stadium eines Prototypen, wurden aber z.B. auf der Cebit oder IFA 2003 demonstriert.
6. **Festplattenrekorder**, wie z.B. der TVTV-Server der Fast-TV-Server AG [>FAST03], können bereits beachtliche Markterfolge erzielen. Diese aus dem PC-Bereich bekannte Technologie wird häufig in DVD-Player oder TVs (z.B. Loewe) integriert angeboten.
7. **Mobiles Video-Abspielgerät mit Festplatte** z.B. von Archos, Sony oder Thomson, speichern Videos aus dem Computer im MPEG-1- ,2 -4-Format auf ihre eingebauten Festplatten. Unterwegs zeigen sie die Videos über eingebaute LC-Displays an und geben den Ton über Ohrhörer aus.
8. **DVB-H (Digital Video Broadcasting - Handheld)-Geräte** werden in naher Zukunft keine Verbreitung finden. Gegenüber DVB-T soll DVB-H speziell die Bedürfnisse an jede Art von Mobilität bedienen - mit geringem Stromverbrauch des Empfängers, Empfang bei hohen Geschwindigkeiten und geringem Datenaufkommen der übertragenen Sendungen. Nokia stellte 2003 zwar einen Prototypen eines mobilen DVB-H Multimedia-Terminals vor, aber mit einer Ausstrahlung von DVB-H ist aufgrund der hohen Kosten in den nächsten Jahren nicht zu rechnen. Zunächst wird in Deutschland (und anderen Ländern) die flächendeckende Versorgung mit DVB-T ausgebaut, die bis 2010 bundesweit abgeschlossenen sein soll. Ende 2004 sollen Pilotprojekte mit DVB-H in einigen europäischen Städten starten, für die Nokia das Gerät einsetzen will. Da gleichzeitig die Fähigkeiten mobiler Geräte schnell zunehmen, werden Geräte wie Handys oder PDAs zunächst DVB-T unterstützen und dadurch zu mobilen Fernsehern. So hat etwa NEC in Japan bereits vor der IFA 2003 einen Prototypen eines Handys mit DVB-T-Empfang vorgestellt.

Literaturreferenzen zu [Kapitel 6.2.5]

[>FAST03] Fast TV Server AG: TV-Server Produktinformation, http://www.tv-server.de/content/tv-server/index.htm, 2003.

[>HDMI03] HDMI Licensing: http://www.hdmi.org/press/press.asp , Mai 2003.

[>KISS03] Kiss Technologies: Produktinformationen, http://www.kiss-technology.com/?p=dvd&v=users, 2003.

[>NOKIA03] Nokia Presseveröffentlichung, Oktober 2003, http://press.nokia.com/PR/200310/922405_5.html.

[>ONKYO03] Onkyo: Produktinformation, http://www.onkyo.net/de/cms/products/home/av_rec/TX-NR801E/, 2003.

[>PANASONIC03] Panasonic: Produktinformation, http://www.panasonic.com/PBDS/subcat/Products/mnu_servers_editing.html, 2003.

6.3. Betriebssysteme

Die große Zahl der existierenden Betriebsysteme lässt sich grob nach Anwendungsgebieten und Zielmärkten in die folgenden drei Klassen unterteilen:

- Server-Betriebssysteme
- Desktop-Betriebssysteme
- Embedded und Echtzeit-Betriebssysteme

Die entsprechenden drei Anwendungsgebiete unterscheiden sich nicht nur in ihren technische Anforderungen an die jeweiligen Betriebssysteme, sondern auch durch verschiedene Geschäftsmodelle und Vertriebsstrukturen.

Der folgende Abschnitt gibt eine kurze Marktübersicht zur Identifikation der für das Projekt DynAMITE interessanten Betriebssysteme. Anschließend wird der aktuelle Stand der Technik der relevanten Betriebssysteme vorgestellt.

6.3.1. Marktübersicht

6.3.1.1. Server-Betriebssysteme

Server-Betriebssysteme zeichnen sich besonders durch ihre hohe Leistungsfähigkeit, Skalierbarkeit und Mehr-Benutzerfähigkeit aus. Der Markt erstreckt sich von den sogenannten Mainframe-Betriebssystemen für Großrechner in Rechenzentren, die mehr als 10000 Benutzer gleichzeitig bedienen können, bis zu kleineren Systemen für bis zu 100 Benutzer [>HINES03, MJEMVNSAMFAB03].
Server-Betriebssysteme stellen hohe Anforderungen an die Hardware. Sie verfügen im Allgemeinen über keine benutzerfreundlichen grafischen Systemschnittstellen und sowohl Installation als auch der Betrieb erfordert geschultes und spezialisiertes Personal. Tabelle 5 zeigt die Marktanteile nach Lizenzen der dominierenden Betriebssystemfamilien für diesen Markt.

Betriebssystem-Kategorie	Marktanteil
Windows	55,1%
Linux	23,1%
Unix	11%
Netware	9,9%

Tabelle 5 Marktanteile der dominierenden Server-Betriebssystemfamilien: Die angegebenen Zahlen beziehen sich auf die verkauften Lizenzen. Stand 2002 nach Quelle [>HINES03, MJEMVNSAMFAB03].

Die Vorteile der Server-Betriebssysteme sind im Heimbereich aufgrund der begrenzten Anzahl von Nutzern und Anwendungen nur eingeschränkt nutzbar und von entsprechend geringer Bedeutung. Sie können hier den Hauptnachteil hoher Kosten der erforderlichen Hardware nicht aufwiegen. Server-Betriebssysteme spielen nicht zuletzt deswegen zur Zeit, und voraussichtlich auch in Zukunft, in dem Anwendungsgebiet von DynAMITE keine Rolle und werden deshalb im Folgenden nicht weiter betrachtet.

6.3.1.2. Desktop-Betriebssysteme

Den größten Markt unter den Betriebssystemen bilden die Desktop-Betriebssysteme. Diese Systeme verfügen i.A. über eine benutzerfreundliche Bedienoberfläche und sind einfach zu installieren. Auch Wartung und Administration vereinfacht sich zunehmend, kann aber den Privatanwender noch überfordern.
Diese Betriebssysteme zeichnet neben einem geringen Preis vor Allem die Unterstützung vielfältiger Peripheriegeräte sowie die große Auswahl an Anwendungsprogrammen aus. Aus diesen Gründen verdrängen Varianten dieser Systeme zunehmend spezialisierte Server-, Embedded und Echtzeit-Betriebssysteme.
Der Markt der Desktop-Betriebssysteme wird vom Monopolisten Microsoft beherrscht (vergleiche Tabelle 6). Zunehmend gewinnt jedoch das Betriebssystem Linux an Bedeutung und konnte seinen Marktanteil in den letzten Jahren um 600% steigern [MJEMVNSAMFAB03]. Ferner ist zu beachten, dass Linux beliebig oft kopiert werden darf und deshalb die Anzahl der tatsächlichen Installationen die Zahl der verkauften Lizenzen deutlich übertreffen kann.

Betriebssystem	Marktanteil
Windows	93,8%
Linux	2,8%
Apple Mac Os	2,3%

Tabelle 6 Marktanteile der Betriebssysteme auf dem Desktop-Markt. Die angegebenen Zahlen beziehen sich auf die verkauften Lizenzen. Stand 2002 nach Quelle [>HINES03, MJEMVNSAMFAB03].

Desktop-Betriebssysteme sind heute in vielen Haushalten vorhanden und so von besonderem Interesse im Projekt DynAMITE. Viele Anwendungen und Produkte im Markt Heimmultimedia basieren auf diesen

Betriebssystemen, wie z.B. das „Windows XP Media Center“ (vgl. Abschnitt 6.3.2.1) von Microsoft oder das „Show Center“ von Pinnacle [WALZ2003].

6.3.1.3. Embedded und Echtzeit-Betriebssysteme

Die meisten verkauften bzw. installierten Betriebssysteme zählen zur Klasse der Embedded und Echtzeit-Betriebssysteme. Sie finden sich in unterschiedlichen Geräten aus praktisch allen Anwendungsgebieten, wie z.B. Handys, PDAs, Routern, Geräten der Unterhaltungselektronik oder in Überwachungs- und Fertigungsanlagen.
Embedded Betriebssysteme sind optimiert bezüglich Programmgröße, verfügbaren Arbeitsspeicher und Stromverbrauch. Echtzeit-Systeme müssen zusätzlich Anforderungen an Reaktionszeiten erfüllen.
Aufgrund der vielen Anwendungsfelder mit vielen unterschiedlichen Anforderungen stellt sich dieser Markt im Vergleich zum Markt der Desktop-Betriebssystemen sehr heterogen dar. Aufgrund der Geschäftsmodelle und Vertriebsstrukturen kann die Zahl der installierten Systeme nicht ermittelt werden. Tabelle 7 listet die bedeutendsten Hersteller von Embedded und Echtzeit-Betriebssystemen nach Marktanteil auf [>LINUXDEVICES03,VDC03]. Dabei ergibt sich der Marktanteil nicht allein aus dem Verkauf der Betriebsysteme sondern bezieht Entwicklungssysteme, Anwendungen und Dienstleistungen mit ein.

Marktposition	Hersteller	Betriebssystem(e)
1.	Microsoft	Windows CE
2.	Wind River	VxWorks, pSOSystem
3.	Symbian	Symbian OS
4.	Palm	Palm OS
5.	QNX	QNX Neutrino
6.	Enea Data	OSE
7.	Green Hills Software	Integrity, ThreadX
8.	LynuxWorks	LynxOs, BlueCat Linux
9.	MontaVista	MontaVista Linux

Tabelle 7 Die bedeutendsten Hersteller von Embedded und Echtzeit-Betriebsystemen sortiert nach Marktanteil. [>LINUXDEVICES03,VDC03]

Laut einer Umfrage der Evans Data Corporation aus dem Jahre 2002 [>LINUXDEVICES02,EDC02] werden momentan die meisten Projekte für Embedded und Echtzeitumgebungen auf den Betriebssystemen Windows Embedded (Windows CE und Pocket PC) sowie Linux geplant (siehe Abbildung 20). Die Anzahl der Projekte entspricht nicht der Anzahl daraus resultierender Systeminstallationen. Das am häufigsten installierte Betriebssystem der Welt ist nach dieser Studie VxWorks der Firma WindRiver [>LINUXDEVICES02,EDC02].

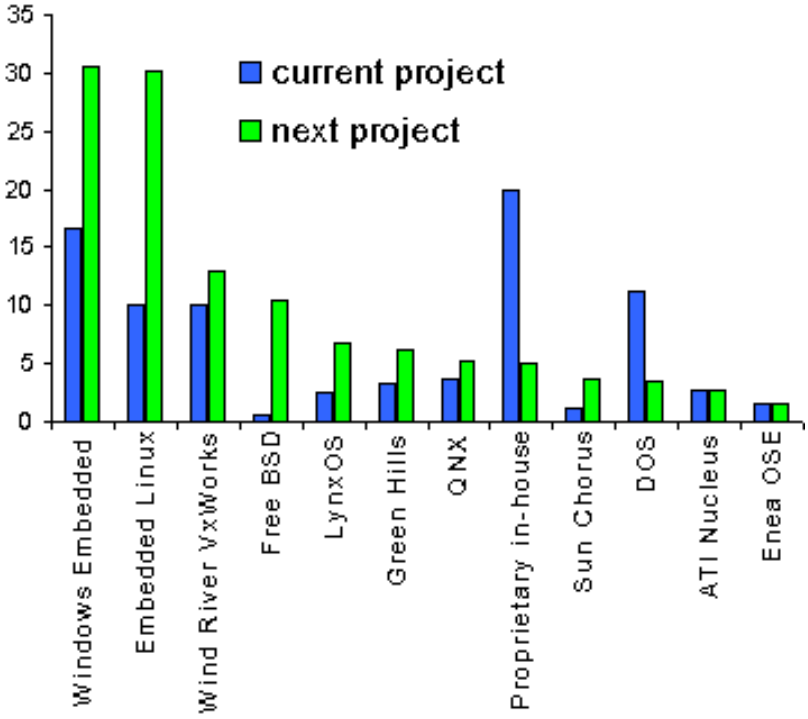

Abbildung 20 Anteil der Betriebssysteme in momentanen und geplanten Projekten im Embedded und Echtzeit-Anwendungen.

Anforderungen an Embedded- und Echtzeit-Betriebssysteme hängen stark vom Anwendungsumfeld ab. Aus diesem Grund haben sich hier viele spezialisierte Systeme für Telekommunikation, Rüstungsindustrie oder Automatisierungstechnik entwickelt. Symbian OS dominiert mit einem Marktanteil von über 90% den Markt der „Smartphones“ (vgl. Abschnitt 6.2.3) und Windows CE zusammen mit Palm OS den Markt der PDAs (vgl. Abschnitt 6.2.2). In Geräten der Unterhaltungselektronik sind neben proprietären Lösungen insbesondere VxWorks und auf Linux basierende Betriebssysteme verbreitet.

Insbesondere die führenden Hersteller von Geräten der Unterhaltungselektronik zeigen ein starkes Interesse an Linux. So gründeten acht der größten Hersteller von Unterhaltungselektronik-Geräten (Matsushita, Sony, Hitachi, NEC, Philips, Samsung, Sharp und Toshiba) 2003 das „Consumer Electronics Linux Forum“(CELF) [CELF03]. Dieses Forum konzentriert sich auf die Etablierung und Promotion von Linux in digitalen Unterhaltungselektronik-Produkten. Dazu sollen formale Anforderungen zur Erweiterung von Linux definiert und besprochen werden. Diese Anforderungen will das CELF veröffentlichen, um dann Open-Source-Lösungen zu akzeptieren und zu evaluieren, die diesen Kriterien entsprechen. Zunächst will das Forum

- das Starten und Beenden von Linux beschleunigen sowie
- die Echtzeit-Fähigkeiten des Systems verbessern,
- den Speicherbedarf senken und
- das Power-Management verbessern.

Dabei will man mit Open-Source-Projekten sowie der Linux-Community zusammenarbeiten.

6.3.2. Windows

Der vorige Abschnitt verdeutlicht die starke Marktstellung von Windows-Betriebssystemen. Aufgrund der großen Bedeutung beschreibt dieser Abschnitt den aktuellen Stand dieser Systeme.

Die Betriebssystemfamilie Windows von Microsoft besteht aktuell aus den Produkten (Stand Dezember 2003)

- Windows XP Professional,
- Windows XP Home Edition,
- Windows 2000 Professional,

- Windows Me,
- Windows NT Workstation 4.0,
- Windows 98 Second Edition,

für den Desktop, sowie den Echtzeit- und Embedded-Betriebssystemen

- Windows CE
- Windows CE 3.0,
- Windows CE Net

und den Server-Systemen

- Windows Server 2003 Standard Edition,
- Windows 2000 Server,
- Windows 2000 Advanced Server,
- Windows NT Server 4.0,
- Windows NT Server, Enterprise Edition und
- SharePoint Portal Server.

Wie bereits erwähnt werden die Server-Betriebssysteme im Folgenden nicht weiter betrachtet.

6.3.2.1. Windows XP

Windows XP ist eine Weiterentwicklung von Windows 2000 bzw. NT Workstation 4.0 für Intel-Prozessoren und löst dieses in der Version "XP Professional" ab. Die Variante "XP Home Edition" ist der Nachfolger von Windows 95, 98, und ME. Sie ist gegenüber "XP Professional" um einige Eigenschaften gekürzt, die hauptsächlich für Unternehmen benötigt werden, wie z. B. Fernverwaltung, Dateiverschlüsselung, zentrale Wartung mittels Richtlinien, Nutzung von mehreren Prozessoren.

Windows XP bietet im Vergleich zu seinen Vorgängern neben verbesserter Zuverlässigkeit, und Benutzerfreundlichkeit [>MS03a]

- Höhere Sicherheit, einschließlich der Möglichkeit, Dateien und Ordner zu verschlüsseln
- Umfassende Unterstützung für digitale Medien
- Tools für Instant Messaging, Audio- und Videokonferenzen
- Unterstützung für drahtlose Netzwerke
- Verbesserte Leistung/Multitasking
- Windows Messenger
- Remotedesktop/ Remoteunterstützung für den Fernzugriff/Fernsteuerung von einem anderen Computer mit Windows
- Erweiterte Laptopunterstützung
- Systemwiederherstellung

Abbildung 21 zeigt den strukturellen Aufbau von Windows XP. Als Ersatz für die DOS-basierten Windows-Versionen 95, 98 und ME mussten Möglichkeiten geschaffen werden, ältere, nicht unter NT lauffähige Programme auszuführen und insbesondere kontrollierte Zugriffe von Anwenderprogrammen auf die Grafik zu ermöglichen. Für Grafikoperationen integrierte Microsoft die Technologie DirectX in das System. Diese ermöglichen durch die Umgehung der Systemdienste und des Mikrokernels einen Direktzugriff insbesondere auf die Grafikhardware. DirectX kennt dabei die Fähigkeiten der Hardware und stellt nicht vorhandene Eigenschaften per Software zur Verfügung. Der Programmierer kann daher Anwendungen, insbesondere Spiele, ohne Geschwindigkeitsverluste trotzdem unabhängig von der eigentlichen Hardware entwickeln. Allerdings werden auch die Sicherheitsvorkehrungen von Windows XP umgangen und einzelne Anwendungen können das System gefährden. Die Ausführung weiterer, für ältere Systeme entwickelte Programme ist durch einen Kompatibilitätsmodus genannt „Personality“ möglich. Dieser emuliert bei Bedarf Routinen aus älteren Systemen. Andere Anwendungen nehmen den vollen Speicherschutz von Windows NT in Anspruch.

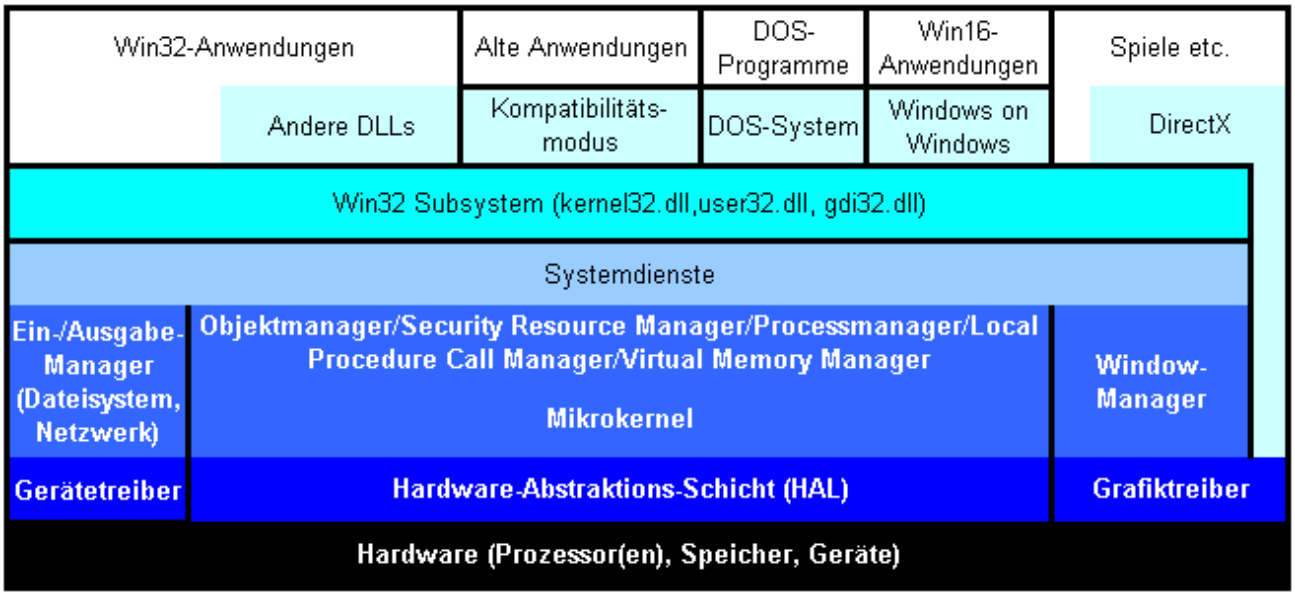

Abbildung 21: Schichten unter Windows XP (etwas vereinfacht).

Windows XP stellt folgende Systemanforderungen:

- Prozessor mit 233 MHz der Intel Pentium/Celeron-Produktfamilie, der AMD K6/Athlon/Duron-Produktfamilie oder ein kompatibler Prozessor
- Arbeitsspeicher: 128 MB RAM oder mehr empfohlen (mindestens erforderlich sind 64 MB, doch kann die Leistung einiger Funktionen eingeschränkt sein)
- Festplatte mit mindestens 1,5 GB verfügbarer Speicher
- CD-ROM- oder DVD-Laufwerk
- Super VGA-Grafikkarte und -Monitor mit einer Auflösung von 800 x 600 oder höher
- Tastatur und Mouse oder kompatibles Zeigegerät

Neben den Varianten „Professional“ und „Home Edition“ ist Windows XP zusätzlich in den Ausprägungen "Tablet PC Edition" und "Media Center Edition" erhältlich.

Windows XP Tablet PC Edition

Die Tablet PC Edition ist als Betriebsystem nur zusammen mit speziellen mobilen Computern erhältlich, die als Tablet-PCs bezeichnet werden (siehe Abschnitt 6.2.4).

Windows XP Media Center Edition

Die Media Center Edition ergänzt die bekannten PC-Funktionen um zahlreiche Funktionen zur Nutzung digitaler Medien wie TV, Personal Video Recording (PVR), Musik, Bilder, Videos oder DVD. Es wird ebenfalls nur auf speziellen sogenannten Media Center PCs vorinstalliert vertrieben. Media Center PC müssen mit folgenden Multimedia-Komponenten ausgestattet sein:

- Windows XP Media Center Edition kompatible Fernbedienung.
- Infrarotsensor (IR) um die Signale der Fernbedienung empfangen zu können und zur Steuerung eines Kabel- und/oder Satellitenreceivers.
- Hochleistungs-Grafikkarte um ein Fernsehsignal in voller Qualität darzustellen.
- TV Tuner für den Empfang von Fernsehsendungen über Kabel, Satellit oder Antenne.
- Hardware Video Encoder für die Aufzeichnung und Kodierung/Komprimierung des Fernsehsignals.
- Video-/Fernsehausgang um Videos, die im Media Center PC gespeichert sind, mit einem Fernsehgerät oder Beamer anschauen oder auf einem externen analogen Gerät aufzeichnen zu können.
- Digitaler Audio-Ausgang um den Media Center PC in ein vorhandenes AV-System zu integrieren.
- WLAN Karte für die kabellose Anbindung in ein Heimnetzwerk und die Verbindung ins Internet.

- DVD-Brenner um Videos und Musik dauerhaft zu speichern.
- Festplatte mit mindestens 100 GB Speicherkapazität.

Für den Einsatz im Wohnzimmer wurde die Oberfläche der Windows XP Media Center Edition so gestaltet, dass sie auch aus größeren Entfernungen leicht erkannt und mit einer Fernbedienung gesteuert werden kann. Media Center PCs werden in 3 Gerätetypen unterteilt:

- Desktop-PC für den Schreibtisch in Arbeits- oder Kinderzimmer
- Hifi-Komponente für das Wohnzimmer
- Notebook für die mobilen Nutzung

6.3.2.2. Windows CE

Windows CE ist ein modulares 32-Bit-, Multitasking-, Multithread-Betriebssystem und wurde vor allem für mobile Endgeräte wie z.B. PDAs, Webpads, Handhelds, Mobilfunkgeräte und Smartphones aber auch für Spielkonsolen entwickelt. Im Gegensatz zu den Desktop-Systemen unterstützt Windows CE verschiedene Prozessor-Architekturen:

- ARM: ARM720T, ARM920T, ARM1020T, StrongARM, XScale
- MIPS: MIPS II/32 mit FP, MIPS II/32 mit FP, MIPS16, MIPS IV/64 ohne FP, MIPS IV/64 ohne FP
- SHx: SH-3, SH-3 DSP, SH-4
- X86: 486, 586, Geode, Pentium I/II/III/IV

Die momentan aktuelle Version Windows CE .Net 4.2, der Nachfolger von Windows CE 3.0, zeichnet sich gegenüber anderen Embedded- und Echtzeit-Betriebssystemen durch seine Multimedia-Fähigkeiten aus und ist dadurch von besonderem Interesse im Projekt DynAMITE. Besonders erwähnenswerte Anwendungen dieses Betriebssystems sind

- der Internet Explorer (Version 5.5),
- der Windows Media Player und die Windows Media Codecs sowie
- das Digital Rights Management (DRM).

Diese Produkte stellen durch die Monopolstellung von Microsoft auf dem Desktop-Markt einen Quasistandard dar, der sich auch auf den Embedded-Markt auswirkt. Z.B. wenden die großen deutschen Video-On-Demand-Anbieter Arcor ([>ARCOR03]) und Deutsche Telekom ([>TCOM03]) die Codecs und das DRM von Microsoft an. Dadurch ist die Nutzung dieser Angebote momentan nur von Windows-Betriebssystemen möglich.

Außerdem erleichtert Microsoft die Entwicklung auf dieser Plattform durch die Integration vieler Funktionalitäten und bietet zahlreiche aus der Desktop-Welt bekannte Schnittstellen an, wie z.B.:

- Active Template Library (ATL)
- C Bibliotheken
- Component Object Model (COM)
- Distributed Component Object Model (DCOM)
- Device Management
- Lightweight Directory Access Protocol (LDAP) Client
- Microsoft Message Queuing (MSMQ)
- Microsoft Foundation Classes (MFC)
- Bluetooth Object Exchange Protocol (OBEX)
- Microsoft Rich Edit Control 2.0
- Microsoft Pocket Outlook® Object Model (POOM) API
- Simple Object Access Protocol (SOAP)
- Standard SDK for Windows CE .NET
- Microsoft .NET Compact Framework
- Microsoft SQL Server 2000 Windows CE Edition (SQL Server CE) Version 2.0
- XML
- DirectX 8

Hersteller wählen zunehmend Windows CE aufgrund integrierter Funktionalitäten und Anwendungen, die potentiellen Kunden bzw. Anwendern aus der Desktop-Welt bekannt und vertraut sind, wie z.B.

- Windows Media Player,
- Internet Explorer,
- Microsoft ActiveSync,
- Dateibetrachter (Microsoft Excel, Image, PDF, Microsoft PowerPoint, Microsoft Word),
- Spiele (Freecell, Solitaire),
- Hilfefunktionen,
- Terminal Emulator,
- Windows Messenger oder
- WordPad

Einen weiteren Vorteil von Windows CE stellt die Verfügbarkeit von leistungsfähigen, ebenfalls aus dem Desktop-Bereich bekannten Entwicklungswerkzeugen dar siehe dazu auch Abschnitt 6.2.2.5 und 6.2.3.5). Außerdem bietet Microsoft kostengünstige und frei Emulationssoftware an, wie z.B. den Microsoft Windows CE .NET 4.2 Device Emulator [>MS03c].

Da der Quellcode von Windows CE nicht frei zugänglich ist, können Entwickler dieses Betriebssystem nicht an Ihre Anforderungen anpassen oder skalieren. Als Ersatz erlaubt ein „Platform-Wizard“ die Erzeugung zwölf vorkonfigurierter Konfigurationen von Windows CE .NET für folgende unterschiedliche Gerätetypen:

- Digital Media Receiver
- Enterprise Terminal
- Enterprise Web Pad
- Gateway
- Industrial Controller
- Internet Appliance
- IP Phone
- Mobile Handheld
- Mobile Phone
- Set-top Box
- Tiny Kernel
- Windows Thin Client

Allerdings stellen diese vorkonfigurierten Systemvarianten nur in sehr seltenen Fällen eine optimale Lösung dar. So können die beiden Profile „Set-Top Box“ oder „Digital media Receiver“ natürlich nicht das ganze Produktspektrum der Unterhaltungselektronik abdecken, vom einfachen MP3-Spieler bis zu einer High-End Heimkino-Installation.

Auf Windows CE basiert auch das Betriebssystem Windows Mobile 2003 in den verschiedenen Ausprägungen Pocket PC, Pocket PC Phone Edition und Smartphone (siehe auch Abschnitt 6.2.3.5).

6.3.2.3. .NET

Aufgrund der Vielzahl eigener Betriebssysteme und in Konkurrenz zu Java entwickelte Microsoft .NET. .NET ist kein Betriebsystem, wird aber trotzdem in diesem Kapitel beschrieben, da es der Vereinheitlichung verschiedener Windows-Versionen dient.

.NET ist eine Betriebssystem-unabhängige Entwicklungsplattform von Microsoft und besteht neben einer virtuellen Laufzeitumgebung aus einem Framework von Klassenbibliotheken und Diensten. Anders als bei Java, kann .NET verschiedene, von der Laufzeitumgebung unterstützte Programmiersprachen ausführen. Hierzu existiert die so genannte Common Language Spezifikation (CLS). Diese Spezifikation definiert einen vereinheitlichten Binärcode, der von der virtuellen Laufzeitumgebung (VM) interpretiert und ausgeführt werden kann. Somit ist es möglich, .NET mit verschiedenen, an die CLR angepassten Sprachen zu programmieren, wie z.B. C# , C++, VisualBasic, VisualJ, Fortran, Haskell, Cobol oder Delphi. Bestehende Programme (z. B. C++-Quelltexte) können aber ohne vorherige Anpassungen nicht kompiliert und ausgeführt werden.

Nachdem ein Programm kompiliert wurde, liegt es gemäß der CLS in einer sprachneutralen Binärform vor. Hiermit wird gewährleistet, dass unterstützte Programmiersprachen auf Bibliotheken oder Pogramme zugreifen können, die in anderen Programmiersprachen geschriebenen wurden. Dieser wiederverwendbare Code wird in so genannten Assemblies zusammengefasst und bereitgestellt (vgl. mit Packages/Paketen in Java oder hierarchischen APIs in C und C++).

Neben dem sprach- und plattformneutralen Binärcode kennt .NET den so genannten "unmanaged Code". Dieser Code liegt in seiner kompilierten Form in einer maschinennahen Form vor und verwaltet die vom Betriebssystem bereitgestellten Ressourcen eigenverantwortlich, ähnlich Java Native Interface (JNI). Ein Programm, das "unmanaged Code" verwendet, ist wie bei JNI i.A. nicht plattformunabhängig.

"Managed code" wird im Vergleich zu "unmanaged code" von einer Laufzeitumgebung verwaltet - der Common Language Runtime (CLR). Diese virtuelle Maschine übernimmt die Anforderung und Freigabe von Ressourcen (Garbage Collection) und stellt sicher, dass geschützte Speicherbereiche nicht direkt angesprochen oder überschrieben werden können. Auch Zugriffe auf Dienste, Dateisystem-Funktionen oder Geräte werden kontrolliert und können, sofern sie gegen Sicherheitsrichtlinien verstoßen, von der CLR abgelehnt werden (Sandbox-Prinzip). Durch die automatische Ressourcenverwaltung und die erhöhte Sicherheit benötigt die Ausführung von "managed code" mehr Zeit und hat einen erhöhten Bedarf an Ressourcen.

Microsoft bezieht die Plattformunabhängigkeit nur auf eigene Betriebssysteme. Eine Laufzeitumgebung (CLR, VM) für Linux steht aber durch das Open-Source-Projekt Mono zur Verfügung , das vom Hersteller Ximian initiiert wurde [>XIMIAN2003]. Dieses Projekt befindet sich noch in der Entwicklung befindet und es stehen nicht alle Komponenten von .NET zur Verfügung. In der nächsten Zeit ist mit einer hohen Verbreitung von .NET-Anwendungen auf Windows-fremden Systemen nicht zu rechnen. Gründe sind fehlende Betriebssystem-Unterstützung von Windows-fremden Systemen durch Microsoft, fehlende Entwicklungsumgebungen sowie lizenzrechtliche Probleme. Unter dem Aspekt der plattformübergreifenden Verfügbarkeit dominiert weiterhin Java.

Für Handhelds und Mobiltelefone unter Windows CE existiert eine abgespeckte Version der .NET Laufzeitumgebung in Form des Compact Frameworks. Die Verbreitung ist aber momentan gering im Vergleich zu Technologien wie der Java 2 Micro Editon (J2ME).

6.3.3. Linux

Der Abschnitt 6.3.1 verdeutlicht die zunehmende Bedeutung des Betriebssystems Linux, speziell auch in der Unterhaltungsindustrie. Aufgrund dieser großen Bedeutung beschreibt dieser Abschnitt den aktuellen Stand dieses Systems.

6.3.3.1. Überblick

Linux bezeichnet genaugenommen den Betriebssystemkern (Kernel) von Linux. Es wird aber allgemein verwendet, um das darauf basierende freie Betriebssystem zu bezeichnen. Im Sprachgebrauch der Free Software Foundation wird dieses Betriebssystem auch als GNU/Linux bezeichnet. Es besteht aus dem Linux Kernel und der GNU-Betriebssystem-Software, wie z.B. Kommandozeile, Shells, Entwicklungswerkzeuge (Compiler, Editoren, Make, ...). Dieses GNU/Linux ist ähnlich wie DOS nur über einen Kommandozeileninterpreter zu bedienen. Eine Linux-Distribution kombiniert deshalb viele, meist freie Software. Sie enthält den

- Linux-Kernel,
- die GNU-Software

und viele darüber hinausgehende Programme wie

- X11-Grafikserver (XFree86),
- Arbeitsoberflächen (z. B. KDE oder GNOME),
- Anwendungen (z. B. OpenOffice und Mozilla) und / oder
- Netzwerkdiensten (z.B. Mailserver und Webserver).

Der Begriff "Linux" wird überwiegend synonym für ganze Linux-Installationen oder Linux-Distributionen verwendet. Notwendig für ein System ist aber nur der Kernel sowie eine Applikation, die eine Systemfunktionalität zur Verfügung stellt.

Der Linux-Kernel bildet die hardwareabstrahierende Schicht und ist zuständig für die Steuerung von Festplatten und Dateisystemen, Multitasking, Lastverteilung und Sicherheitserzwingung.
Er stellt das Basis-System dar, das zwischen dem BIOS der Hardware und der höheren Software liegt und macht die Hardware für die auf dem System laufenden Anwendungen durch ein einheitliches API verfügbar.
Der Kernel enthält nur die Kernkomponenten eines Betriebssystems. Er besteht aus fünf Subsystemen:

1. Prozess-Ablaufsteuerung
2. Dateisystem-Interface
3. Speicherverwaltung
4. Netzwerk-Protokolle
5. Interprozesskommunikation

6.3.3.2. Nachteile

Dieser Nachteil zählt wichtige, für das Projekt DynAMITE relevante Nachteile von Linux auf.

Geringe Verbreitung auf dem Desktop

Nachteile von Linux gegenüber Windows ergeben sich zum Teil aus der Monopolstellung von Windows auf dem Desktop-Markt, wie z.B. die mangelnde Erfahrungen von Benutzern mit Linux. Auch existieren insbesondere Standard-Anwendungen, z.B. MS-Office, Adobe Photoshop, viele Entwicklungsumgebungen oder Spiele aufgrund der geringen Anzahl von Linux-Nutzern nur als Windows-Ausgaben.

Fehlende Multimedia-Schnittstelle

Zudem fehlt unter Linux bislang eine einheitliche Schnittstelle für Multimedia-Anwendungen. Unter Windows bieten DirectX und DirectShow eine einfache Schnittstelle für den Zugriff auf Multimedia-Funktionen. Z.B. übernimmt beim MP3-Abspielen ein DirectShow-Filter die MP3-Dekodierung und die Soundausgabe erledigt eine DirectX-Komponente. Unter Linux hat bisher keine derartig umfassende Multimedia-Architektur Verbreitung gefunden. Linux-Anwendungen implementieren grundlegende Funktionen zur Verarbeitung von Multimedia-Daten immer wieder neu.

6.3.3.3. Vorteile

Modularität & Skalierbarkeit

Während der Kernel immer für ein Linux-System erforderlich ist, können andere Systembestandteile, wie z.B. Gerätetreiber, bei Bedarf geladen bzw. entladen werden. Diese Modularität minimiert den Speicherbedarf und erhöht die Skalierbarkeit. Ein Linux-System kann leicht entsprechend den Anforderungen zusammengestellt werden, da sich Teilkomponenten austauschen lassen. Nur die benötigten Protokoll- und Treiberkomponenten müssen in den Kernel eingefügt werden. Zudem kann die Anzahl der Systemprogramme an die jeweiligen Anforderungen angepasst werden.

Portabilität

Ein wichtiger Vorteil gegenüber Windows ist die Portabilität von Linux. Linux ist inzwischen nach NetBSD das am zweithäufigsten portierte Betriebsystem und läuft z.B. auf folgenden Prozessorarchitekturen:

- ARM (610, 710, 720T, 920T, StrongARM, Intel Xscale)
- AMD64 (x86-64)
- Axis Communications' CRIS
- Compaq Alpha-Prozessor
- Hitachi H8/300
- HP PA-RISC
- IA-64: PCs mit 64bit Intel Itanium-Prozessor
- IBM S/390

- Intel 80386, 80486, und Pentium-Serie, AMD Athlon, Duron, Thunderbird; Cyrix-Prozessoren.
- MIPS (MIPS4k, MIPS64, MIPS5k MIPS10k R2k, R3k, R4k, R4x00,
- Motorola 68020
- NEC v850e
- PowerPC
- Sun SPARC und UltraSparc
- Hitachi SuperH (SH-3, SH-4).

Dieses Kapitel über Linux unterscheidet nicht nach Desktop- und Embedded-Versionen. Für Embedded-Linux-Anwendungen stehen prinzipiell die Befehle aller Linux-Bibliotheken zur Verfügung. Das Format der ausführbaren Dateien ist – sofern der gleiche Prozessor zum Einsatz kommt – bei Linux und Embedded Linux gleich. Alle in diesem Abschnitt über Linux genannten Vor- und Nachteile gelten auch für Embedded-Linux.

Ebenso listet dieser Abschnitt keine Systemanforderungen auf, da Linux aufgrund der Skalierbarkeit und Flexibilität an die vorhandene Hardware angepasst werden kann. Dadurch sind auch verschiedene Distributionen für unterschiedliche Hardware kostenlos erhältlich.

Stabilität & Flexibilität

Die große Zahl der Portierungen bedingt wiederum Rückwirkungen auf Linux in Form von

- massiver Stabilität durch das Testen auf verschiedensten Plattformen,
- einem breiten Spektrum an verfügbaren Komponenten sowie einer
- hohen Flexibilität des Systemaufbaus,

da gemachte Erfahrungen aktiv in die Entwicklung von Linux einflossen und einfließen.

Offenes System

Neben der Modularität, Skalierbarkeit und Stabilität stellen die

- konsequente Nutzung offener Standards sowie
- die Offenheit des Quellcodes

weitere Vorteile gegenüber Windows dar. Aufgrund der Offenheit des Quellcodes fließen in Linux praktisch keine proprietären Lösungen ein. Statt dessen unterstützt Linux zahlreiche Standards für Kommunikation (TCP/IP-Protokolle, WLAN, IEEE1394, USB,...) und APIs (Posix, ANSI-C/C++).

Aufgrund der Offenheit des Quellcodes, und insbesondere der Verpflichtung zur Offenlegung von Änderungen, steht eine große Anzahl an Entwicklern zur Verfügung. So entstehen innerhalb kürzester Zeit Portierungen auch für ausgefallene Plattformen, wie z.B. für Sonys PlayStation oder Microsofts Xbox.

Aufgrund der Offenheit des Systems und durch die Lizenzfreiheit des Betriebssystems bietet sich Linux zudem als Testsystem in Forschung und Wissenschaft an. Speziell Forschungsergebnisse können dadurch als ganzes System kostenlos verbreitet werden, wie z.B. im Fall von OpenEmbassi [>EMBASSI03] geschehen. Gerade in den letzten Jahren nutzen immer häufiger Forschungsgruppen Linux als Betriebssystem. Durch die große Zahl an Forschungsergebnissen, z.B. Software-Bibliotheken und Architekturen für Linux, gewinnt dieses Betriebssystem zusätzlich an Attraktivität. Neben OpenEmbassi sei hier beispielhaft das Projekt „Network-Integrated Multimedia Middleware“ (NMM) der Universität Saarbrücken genannt [>SAAR03, LOHSE03].

Im Rahmen von NMM entstand eine netzwerkfähige Multimedia-Infrastruktur für Linux, die ein einfaches und leicht nutzbares Interface für in Applikationen integrierte Multimedia-Funktionen darstellen soll. Die lokale Basis der netzbasierten Architektur stellt eine Multimedia-Middleware für Linux dar, die sich um Plug-ins und Applikationen erweitern lässt. Darauf aufbauend arbeitet man an einer Architektur für Integration und Kontrolle von Geräten, die im Netz verteilt sind. Dies schließt eine netzbasierte Registry, ein einheitliches Event-System und Daten-Transport ein. Als reines Forschungsprojekt gestartet, ist NMM mittlerweile ein Open-Source-Projekt [>NMM03]. Bei entsprechender Verbreitung lösen die Ergebnisse dieses Projekts das in Abschnitt 6.3.3.2 beschriebene Problem einer fehlenden einheitlichen Multimedia-Schnittstelle in Linux.

6.3.3.4. Fazit

Im Desktop-Bereich stellen Windows-Betriebssysteme ein Monopol dar. Dieses wirkt sich auch auf andere Märkte aus. So gewinnt Microsoft mit Pocket PC immer größere Marktanteile im PDA-Markt. In anderen Anwendungsbereichen, wie z.B. im Mobilfunk oder in der Consumer Elektronik bestimmen andere Betriebssysteme wie Symbian OS oder Linux den Markt.

Ein großer Vorteil von Windows liegt in der Verfügbarkeit von ausgereiften Entwicklungsumgebungen, insbesondere Visual Studio, die Microsoft für alle Windows-Zielplattformen anbietet.

Mit dem .NET-Framework versucht Microsoft Anwendern die Möglichkeit zu geben, Programme Betriebssystem-unabhängig, zumindest unabhängig von der Windows-Version, zu erstellen. Dadurch würde ein Hauptnachteil gegenüber Linux aufgelöst, bei dem diese prinzipielle Unabhängigkeit der Programme von einer Linux-Version oder –Distribution bereits gegeben ist. Die Verbreitung von .NET ist momentan noch gering im Vergleich zu Java. .NET-Programmierung eignet sich, ähnlich wie Java-Programmierung, aufgrund der geringen Komplexität der Sprache für Prototypen und die Erstellung graphischer Benutzeroberflächen. Für Systemprogramme sind Java und .NET, zumindest im Embedded-Bereich, aufgrund der schlechten Performanz bzw. hohen Systemanforderungen ungeeignet.

Windows-Betriebssysteme zeichnen sich insbesondere durch die integrierten Multimedia-Fähigkeiten aus. Speziell die sehr leistungsstarken Windows Media Codecs [>Zota03] finden weite Verbreitung, z.B. im Internet oder bei Video-On-Demand Anbietern. Die Codecs stehen zwar auch für andere Betriebssysteme wie z.B. Linux zur Verfügung, allerdings mangelt es noch an Abspielsoftware, die in kommerziellen Produkten eingesetzt werden kann.

Speziell die Windows-Betriebsysteme ohne große Marktverbreitung sind noch wenig getestet, entsprechend unstabil und führen häufig bei Produktentwicklungen zu Zeitverzögerungen. Ein großer Nachteil ist auch der entweder nicht einsichtige bzw. nicht zu ändernde Quellcode des Betriebssystems. Diese Gründe, in Verbindung mit Vorbehalten gegenüber Geschäftspraktiken von Microsoft, verhinderten bisher die Verbreitung von Windows-Betriebssystemen in den für das Projekt Dynamite interessanten Märkten Heimmultimedia und Mobilfunk.

Durch seine größere Modularität, Skalierbarkeit, Flexibilität, Stabilität und Portabilität im Vergleich zu Windows ist Linux auf deutlich mehr Plattformen verfügbar. In den drei Märkten Server, Desktop und Embedded ist es das Betriebssystem mit den größten Zuwachsraten. Insbesondere die Unterstützung durch alle großen Unterhaltungselektronik-Konzerne lässt erwarten, das sich Linux in der Unterhaltungselektronik durchsetzt.

Durch die Offenheit des Systems wählen viele Forschungsgruppen Linux als Betriebssystem. Die Quellen von Linux und häufig auch Forschungsergebnisse stehen als Quelltexte zur Verfügung und dürfen modifiziert werden. Dadurch stehen viele Bibliotheken und Programme für die Nutzung in anderen Projekten, z.B. DynAMITE, zur Verfügung.

Literaturreferenzen zu [Kapitel 6.3]

[>ARCOR03] Arcor, Video on Demand, http://www.arcor.de/vod/vod_1_0.jsp, 2003

[>CELF03] Consumer Electronics Linux Forum, Juli 2003, http://www.celinuxforum.org/.

[>EDC02] Evans Data Corporation, Embedded Systems Developer Survey, Vol.2, 2002, http://www.evansdata.com/survey_embedded_03_1_topical.shtml.

[>EMBASSI03] Embassi Konsortium: OpenEmbassi 2003, http://www.embassi.de/open_embassi/start_frame.html.

[>HINES03] Matt Hines: Microsoft still rules server OS market, News.som, Oktober 2003, http://news.com.com/2100-7344-5088233.html

[>LINUXDEVICES02] LinuxDevices: Linux, Windows neck-and-neck in embedded, Oktober 2002, http://www.linuxdevices.com/news/NS7483572763.html.

[>LINUXDEVICES03] LinuxDevices: VDC finds Microsoft, Wind River in "dead heat" as #1 embedded OS, März 2003, http://www.linuxdevices.com/news/NS7483572763.html.

[>LOHSE03] Marco Lohse: Ein Knoten für alle Fälle -NMM-Vernetztes Multimedia Home Entertainment mit Linux, ct' 22, Seite 232, 2003.

[>MJEMVNSAMFAB03] Mark Melenovsky, Stephen L. Josselyn, Matthew Eastwood, Thomas Meyer, Ricardo Villate, Masahiro Nakamura, Avneesh Saxena, Rajnish Arora, Roman Maceka, Alan Freedman, Greg Ambrose , Jean S. Bozman: Worldwide and U.S. Server Forecast Update 2002-2007, Oktober 2003, IDC market Analysis, http://www.idc.com/getdoc.jhtml?containerId=30232

[>MS03a] Microsoft GmbH: Windows XP Produktübersicht, http://www.microsoft.com/germany/ms/windowsxp/solution/index.asp, 2003.

[>MS03b] Microsoft GmbH:, Windows CE, http://www.microsoft.com/germany/produkte/druck.asp?siteid=120, 2003.

[>MS03c] Microsoft Corp., Microsoft Windows CE .NET 4.2 Device Emulator, http://www.microsoft.com/downloads/details.aspx?familyid=b670079a-3000-4f3c-a74f-0235a20563cf&displaylang=en, 2003.

[>NMM03] http://sourceforge.net/projects/nmm.

[>SAAR03] Universität Saarbrücken, Computer Graphics Lab: Network-integrated Multimedia Middleware for LINUX, http://graphics.cs.uni-sb.de/NMM, 2003

[>TCOM03] T-online: T-Online Vision, http://www.t-online-vision.de/ ,2003.

[>VDC03] Venture Development Corporation, The Embedded Software Strategic Market Intelligence Program 2002/2003 - Volume II, Part A, Embedded/Real-Time Operating Systems and Toolkits, Studie, März 2003, http://www.vdc-corp.com/embedded/reports/03/br03-10.html.

[>WALZ03] Alexander Walz, Frisch serviert, video, Vol. 12, 2003.

[>XIMIAN2003] Ximian: Mono-Project, http://www.go-mono.com/, 2003.

[>Zota03] Volker Zota: Kompressionisten - Aktuelle Video-Codecs im Vergleich, ct' 10, Seite 146, Mai 2003.

7. Ontologien

Es gibt bereits verschiedene Ontologien und Beschreibungstemplates für Geräte, Dienste, Nutzer, Positionen (location) und Kontext, die in diesem Kapitel näher beschrieben werden. Diese Ontologien sind unabhängig voneinander und von unterschiedlichen Gremien entwickelt worden. Die eingesetzten Beschreibungssprachen sind unterschiedlich und haben vor allem unterschiedliche Ausdrucksmöglichkeiten.

7.1. Umgebungsmodell (Raummodell) / Dynamische Nachführung

Es wurden keinerlei Ontologie für integrierte Weltmodelle für Intelligent Home oder intelligent Office Szenarien gefunden, die man z.B. für zielbasierte Plannung und Service Composition benutzen konnte. Es gibt allerdings einige Arbeiten in den Bereichen autonome Robotersysteme, militärische Einsatzplannungssysteme und Raumfahrt, die aber für DynAMITE Szenarien nicht direkt einsetzbar sind.

7.1.1. Easyliving Geometric Model

Ein erweiterbares Raummodell ist das „EasyLiving Geometric Model“**,** welche die physische Anordnung und die *relative* räumliche Beziehungen von sogenannten ***Entities*** (People, Places, Things, I/O Devices), die sich in einem (geschlossenen) Raum befinden, repräsentieren kann. Diese Weltmodell wird ständig von „Sensing Devices“ mit „perceptual information“ versorgt, welche Informationen über „the state of the World, such as location of people“ liefern. (Braumitt2000)

Hier Originalauszug von (Braumitt2000), der das Model beschreibt:

“The EasyLiving Geometric Model (EZLGM) provides a general geometric service for ubiquitous computing, focusing on in-home or in-office tasks in which there are myriad I/O, perception, and computing devices supporting multiple users. The EZLGM is designed to work with multiple perception technologies and abstract the application (including its user interface) away from any particular sensing modality. It is aimed at geometry “in the small”, that is, for applications which may require sub-meter localization of entities in the environment. Integrating this model with localization services for larger scales remains an open issue. The base item in the EZLGM is an entity, which represents the existence of an object in the physical world. Measurements are used to define geometric relationships between entities. In particular, a measurement describes the position and orientation of one entity’s coordinate frame, expressed in another entity’s coordinate frame. Since objects in the physical world (as well as virtual objects like service regions) have some physical expanse, this can also be expressed as an extent in the geometric model using a polygon described in the coordinate frame of the entity. Additionally, measurements can have an uncertainty associated with them, expressed as a covariance matrix on parameters of the transformation.

Once a set of measurements has been provided to the geometric model, the model can be queried for the relationships between entities’ frames. The measurements describe an undirected graph, where each vertex is the coordinate frame of an entity, and each edge is a measurement, as described above. If at least one path exists between two frames, then the graph can be processed to produce a single geometric relationship between the frames. Since a particular queried relationship may not have been previously directly measured, the response typically involves the combination of multiple measurements; uncertainty information is used to properly merge multiple redundant measurements as needed. Region-intersection queries are also supported, allowing, for example, selection of all devices within a particular radius of a user. In the example scenario, the person tracking software continuously updates the measurement which describes the geometric relationship between Tom/Sally and the coordinate frame of the sensor which is observing them. Whatever process is responsible for keeping Tom’s session available on nearby devices can query EZLGM for all devices that have service areas that intersect with his location. The process first looks at types and availability to determine the set of devices which could provide the display, text input, pointing, etc. It then further prunes the list by considering the physical constraints (e.g. visibility) and electronic constraints (e.g. availability), in order to reach a set of usable, available, and physically-appropriate devices.Visibility can be checked by examining all entities along the line of sight between the user and the device and ensuring none have an extent present which represents something that physically blocks Tom’s view. Then, once Tom’s location is stable with respect to a set of devices, the session can be directed to automatically move.”

Literaturreferenzen zu [Kapitel 7.1]

[Braumitt2000] B. Braumitt, B. Meyers, J. Krumm, A. Kern and S. Shafer (2000). „EasyLiving: Technologies for Intelligent Environments”. In Handheld and Ubiquitous Computing, September 2000. http://www.research.microsoft.com/easyliving/Documents/2000%2009%20Barry%20HUC.pdf}

7.2. Gerätebeschreibung

Es gibt verschiedene Standards für Gerätebeschreibung. W3C CC/PP, UPnP Device Description, FIPA Device Ontology, WAP-Forum's UAProf (UAProf99) sind die bekanntesten. UAProf wird hier nicht weiter beschrieben, weil W3C CC/PP weitestgehend kompatibel zu UAProf ist und weil CC/PP und UAProf relativ leicht ineinander konvertiert werden können.

7.2.1. FIPA Device Ontology

FIPA device ontology (FIPA-DevOnto2001) spezifiziert Eigenschaften von Geräten, die von Agenten benutzt werden, wenn diese sich über Geräte unterhalten. FIPA device ontology beinhaltet Angaben über *Hardware, Product Info, Connection Type, User Interface, Screen, Resolution, Memory, Memory Type und Software properties*.

Folgender Profil beschreibt einen hypothetisches SmartPhone xyz:

Profile	fipa.profiles.device.smartphonexyz
Ontology	Fipa-Device

Parameter						**Value**
info-description	name					SmartPhone
	vendor					Smartphones Ltd.
	version					xyz
type						mobile-phone PDA GPS
agent-compliancy						true
hw-description	connection-description	info-description	name			Bluetooth
			version			x.x
	connection-description	info-description	name			Infrared Data Association
			version			y.y
	connection-description	info-description	name			High Speed Circuit Switched Data
			version			z.z
	ui-description	screen-description	width			500
			height			800
			unit			mm
			resolution-description	width		1024
				height		768
				unit		pixels

				bpp	32
				graphics	true
			color		true
		audio-input			true
		audio-output			true
	memory-description	memory-type-description		amount	8
				unit	MB
				usage-type	storage
		memory-type-description		amount	3856
				unit	KB
				usage-type	storage
	cpu				64-bit ARM9-based RISC
sw-description	info-description			name	SmartOS abc
				vendor	ABCVendor Corp.
				version	8.1
	agent-platformhttp://www.fipa.org/specs/fipa00091/PC00091A.html - _ftn13			name	FIPA-OS v2.1.1
				dynamic	true
				mobility	true

Die Werte der äusserst rechten Spalte können sich ändern. Wenn z.B. extra Speicher verfügbar ist, oder anderer Version von Betriebssystem installiert ist, können sich die dazugehörige Werte ändern.Die Parameter an sich sind aber statisch, da sie in in Fipa-Device ontology spezifiziert sind, welcher unabhängig von diesm konkreten Profil ist

7.2.2. W3C Composite Capability/Preference Profiles (CC/PP)

CC/PP ist ein RDF-basiertes Framework für Beschreibung und Management von Software- und Hardwareprofilen. Ein CC/PP Profil ist eine Beschreibung von Geräteeigenschaften und Nutzerpräferenzen, die für Web content delivery and content adaptation von Bedeutung sind. CC/PP (ccpp03) soll benutzt werden, um die exakte Fähigkeiten eines Clients zu beschreiben, um raffinierte content negotiation techniques zwischen Web-Servern und Web-Clients zu ermöglichen. Ziel ist es, eine optimierte Erzeugung von Markups für die Benutzung und Präsentation von Content auf eine Vielzahl von Agenten zu erlauben.

In einem CC/PP Profile werden im wesentlichen eine Reihe von Attributen mit zugehörigen Werten beschrieben, wobei das Vokabular standardisiert ist bzw. dem Server bereits bekannt ist.
CC/PP ist für eine Vielzahl von webfähigen Geräten (PDA, Desktop, laptop, WAP phones, web television, spezial browsers for disable users, etc.) konzipiert worden. CC/PP ist ein W3C Standards, in der namhafte Firmen wie Ericsson, Nokia, T-Mobile, IBM, Sun, SAP, Fujitsu, etc. beteiligt sind.

Man kann CC/PP RDF Schema benutzen um spezifische Vokabular zu definiere, die dann spezielle Eigenschaften von speziellen Gerätekategorien beschreiben (etwa vergleichbar mit ***UPnP* Gerätetemplates**). Solche Vokabular gibt es für ***„Print and Display“***-Geräte, das auf Arbeiten von IETF media feature registration Working Group (RFC2534) basiert. Darin *werden type, schema, charWidth, charHeight, charset, pix-x, pix-y, und color* beschrieben. Auf die Semantik wird hier nicht eingegangen. Ein Vokabular gibt es auch für ***Mobil Phones*** und spezielle Profile davon gibt es für ***Ericsson T68, , Ericsson T39 sowie für Mitsubishi Trium.*** Es gibt viele ***UAProf-Profile für fast alle gängigen Mobile Phones*** (UAProofRepository).

7.2.3. UPnP™ Device Architecture

Ein UPnP (UPnP2003) Gerät kann als ein Container für Dienste und weitere verschachtelte logische Geräte angesehen werden. Logische Geräte können als *single root device* mit integrierten Diensten oder als *multiple root device* modelliert werden.

UPnP Gerätebeschreibung ist in XML ausgedrückt und wird auf der Basis von *UPnP Device Templates* erstellt. UPnP Device Templates werden von *UPnP Working Commitees* definiert, welche auch Kategorien von Geräten mit zugehörigen allgemeingültigen Gerätebeschreibungen definieren. Diese Gerätetemplates werden hier NICHT weiter beschrieben. Es ist aber zu bemerken, dass ***MediaServer, MediaRenderer, LightControls****,* für DynAMITE von besonderer Bedeutung sind.

1. **Internet Gateway Device (IGD) V1.0:** Zu dieser Kategorie gehören u.a. WANConnectionDevice, WANEthernetLinkConfig, WANPOTSLinkConfig, WANDSLLinkConfig, WANDevice, LANDevice, LANHostConfigManagement, WANPPPConnection, WANIPConnection. Für detailierte Informationen sei auf (UPnP-IGD01) verweist.

2. **MediaServer V1.0 and MediaRenderer V1.0:** dieses Template definiert einen general-purpose Gerät geeignet für Beschreibung jäglicher Consumer Electronic (CE) Geräte, die AV content für andere UPnP Geräte in Home Networks anbieten. Beispiele für MediaServer sind: VCRs, CD Players, DVD Players, audio-tape players, still-image cameras, camcorders, radios, TV Tuners, and set-top boxes. Andere Beispiele sind MP3 servers, PVRs, Home Media Servers, PCs. MediaServer (via the Content Directory) ist in der Lage verschiedene contents (MPEG2 video, CD audio, MP3 and/or WMA audio, JPEG images) in einheitliche und konsistente Art für Home Networks zur Verfügung zu stellen. The MediaServer ist sowohl für low-resource devices wie MP3 players als auch für high-end Home Media Servers geeignet. Alle full-featured MediaServer Geräte bieten folgende Funktionalitäten an (UPnPMedia03):
    - Enumerate and query any of the content that the MediaServer can provide to the home network.
    - Negotiate a common transfer protocol and data format between the MediaServer and target device.
    - Control the flow of the content (e.g. FF, REW, etc).
    - Copy (import) content to the MediaServer from another device.
3. **Printer Device and Print Basic Service V1.0**
4. **Scanner (External Activity V1.0, Feeder V1.0, Scan V1.0, Scanner V1.0)**
5. **Basic Device V1.0**
6. **HVAC V1.0: Ein HVAC-System beinhaltet alle Heiz- und Kühlsysteme, welche für unabhängige Klimaregelung eines ganzen Hauses oder Teile eines Hauses notwendig sind.**
7. **WLAN Access Point Device V1.0**
8. **Device Security V1.0 and Security Console V 1.0**
9. **Lighting Controls V1.0:** es werden die Beleuchtungsgerätetypen BinaryLight V1.0, DimmableLight V1.0, Dimming V1.0, SwitchPower V1.0 beschieben.

UPnP Device Templates sowie UPnP Device Description sind beide mashine-readable. Es gibt Tools für automatische type checking (ob “required elements are correctly nested and have values of the correct data types”). Eine UPnP Gerätebeschreibung hat zwei logische Teile:

1. device description: beschreibt die physische und logische Kontainer sowie integrierte (logische) Geräte
    a. vendor-specific manufacturer information
        i. model name
        ii. serial number
        iii. URLs to vendor specific Web site
        iv. Etc.

b.

2. service description: beschreibt die Fähigkeiten (Capabilities) des Geräts in Form von eine Liste von Services
    a. service type
    b. service name
    c. URL to a service description
    d. URL for control
    e. URL for eventing
3. URL for presentation of the aggregate

Folgende XML Beschreibung ist ein Template für UPnP Device Description:

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>urn:schemas-upnp-org:device:deviceType:v</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service:serviceType:v</serviceType>
        <serviceId>urn:upnp-org:serviceId:serviceID</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      Declarations for other services defined by a UPnP Forum working
      committee (if any) go here
      Declarations for other services added by UPnP vendor (if any) go here
    </serviceList>
    <deviceList>
      Description of embedded devices defined by a UPnP Forum working
      committee (if any) go here
      Description of embedded devices added by UPnP vendor (if any) go here
    </deviceList>
    <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```

Folgende XML-Beschreibung ist ein Template für UPnP Service Description:

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
    <name>actionName</name>
      <argumentList>
        <argument>
          <name>formalParameterName</name>
          <direction>in xor out</direction>
          <retval />
          <relatedStateVariable>stateVariableName</relatedStateVariable>
        </argument>
        Declarations for other arguments defined by UPnP Forum working
        committee (if any)go here
      </argumentList>
    </action>
    Declarations for other actions defined by UPnP Forum working
    committee (if any)go here
    Declarations for other actions added by UPnP vendor (if any) go here
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="yes">
      <name>variableName</name>
      <dataType>variable data type</dataType>
      <defaultValue>default value</defaultValue>
      <allowedValueList>
        <allowedValue>enumerated value</allowedValue>
        Other allowed values defined by UPnP Forum working committee (if
        any) go here
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>variableName</name>
      <dataType>variable data type</dataType>
      <defaultValue>default value</defaultValue>
      <allowedValueRange>
        <minimum>minimum value</minimum>
        <maximum>maximum value</maximum>
        <step>increment value</step>
      </allowedValueRange>
    </stateVariable>
    Declarations for other state variables defined by UPnP Forum working
    committee(if any) go here
    Declarations for other state variables added by UPnP vendor (if any)
    go here
  </serviceStateTable>
 </scpd>
```

7.2.4. Relevanz für DynAMITE

Es wurden keinerlei Ontologie für integrierte Weltmodelle für Intelligent Home oder intelligent Office Szenarien gefunden, die man z.B. für zielbasierte Plannung und Service Composition benutzen konnte. Es gibt allerdings einige Arbeiten in den Bereichen autonome Robotersysteme, militärische Einsatzplannungssysteme und Raumfahrt, die aber für DynAMITE Szenarien nicht direkt einsetzbar sind.

Die meisten Geometriemodelle (Raummodelle) sind aus Anwendungsgebieten, die ausserhalb von DynAMITE Hauptszenarien liegen. Jeeoch eignet sich das EasyLiving Raummodell für DynAMITE, da es besonders erweiterbar ist und auch weil EasyLiving Geometriemodell für Home/Office Szenarios ausgelegt ist. Besonders interessant ist es, dass in diesem Modell die Positionen von Objekten relativ zu dem Objekt selbst festgehalten werden.

Es existieren auch viele Gerätebeschreibungssprachen und –ontologien wie CC/PP, UAProf, FIPA Device Architektur, UPnP Device Templates. UAProf und CC/PP beschränken sich lediglich auf Beschreibung von „Web-Geräten". Fokus ist also die Unterstützung von optimierte (HTML) Content-Adaptation und (HTML) Content Selection. Da aber das WAP Standard UAProf einsetzt, gibt es fast für jedes Mobiltelefon ein UAProf. Dies ist ein klarer Vorteil für UAProf. Trotzdem eignet sich CC/PP und UAProf **nicht** für DynAMITE, weil dann entweder der Standard angepasst bzw. erweitert werden müsste (!) oder zusätzliche Gerätebeschreibungen benutzt werden müssten.

Die am fortschrittlichsten Ontologien und Standards für Geräte Service–Beschreibungen bietet UPnP an. So sind UPnP Device Templates und UPnP Device Description beide mashine-readable. Es gibt auch Tools für automatische type checking. UPnP bietet nicht nur die Beschreibungssprache für Geräte. UPnP Working Groups bieten für viele Gerätekategorien (MediaServer, LightControls) standardisierte Beschreibungen an. UPnP ist für DynAMITE sehr interessant.

Literaturreferenzen zu [Kapitel 7.2]

[UpnP2003] "Universal Plug and Play". 2003. URL: http://www.upnp.org

[UpnPDevice03] "UpnP Device Architecture 1.0". UpnP Forum, Version 1.0.1, 2. 12. 2003. URL: http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf

[UpnP-IGD01] "Internet Gateway Device". UPnP, Version 1.0, 19.11.2001. URL: http://www.upnp.org/standardizeddcps/igd.asp

[UPnPMedia02] "MediaServer V1.0 and MediaRenderer V1.0". UpnP Forum, Version 1.0, 24.06. 2002. URL http://www.upnp.org/standardizeddcps/mediaserver.asp

[CCPP03] Composite Capabilities/Preferences Profiles. URL: http://www.ccpp.org

[FIPA-DevOnto2001] „FIPA Device Ontology Specification". Document Number XC00091C, Genve, Switzerland,10.05.2002. URL: http://www.fipa.org/specs/fipa00091/PC00091A.html

[UAProf99] „User Agent Profile Specification". WAP-Forum, Wireless Application Group, Version 10-Nov-1999. URL: http://www.wapforum.org/what/technical/SPEC-UAProf-19991110.pdf

[UAProfRepository] http://w3development.de/rdf/uaprof_repository/

7.3. Beschreibungssprachen

7.3.1. DAML/S

DAML-S (DAMLS-2003) bietet eine Ontology und Beschreibungssprache für *Services,* um eine autmatische discovery, composition und monitoring von Services zu ermöglichen. Die Ontology besteht aus drei Teilen:

- „Service Profile" ermöglicht Dienste zu inserieren und zu finden (service lookup and discovery)
- „Process Model" beschreibt detailiert die von dem Service ausgeführten Operationen
- „Grounding" beschreibt, wie mit dem Service (basierend auf Messages) interagiert werden soll.

Definition of Service: **"Web sites that do not merely provide static information but allow one to affect some action or change in the world, such as the sale of a product or the control of a physical device…. Services can be complex, composed of multiple primitive services."** (DAMLS-Coalition03)

DAML-S erlaubt also Services Maschienen-lesbar zu beschreiben und zu spezifizieren, wie ein Servic benutzt werden soll.

7.3.1.1. Was kann man mit DAML-S machen?

- **Automatic Web service discovery**: ein Agent kann ein bestimmte Service im Web finden, das bestimmte Beschreibungsmuster, Verhalten oder Funktionalität anbietet. *Beispiel: ein Service um Flug von Berlin nach Paris am 07.01.2004 für 500,- € zu finden.* Hierzu bietet DAML-S „*declaraive advertisements of service properties and capabilities"*.
- **Automatic Web service invocation:** ein Agent kann einen bestimmten discovered service automatisch ausführen lassen. Die „automatic execution of a Web service" kann als eine Sammlung von Funktionsaufrufen betrachtet werden. Mittels DAML-S kann solch eine API *deklarative und maschienen-interpretierbare* beschrieben werden. Ein Agent muss aus ddiesen (DAML-S) Markup interpretieren und verstehen können, welche Input für den Funktionsaufruf notwendig ist und was der Aufruf bewirkt (output und Seiteneffekte).
- **Automatic Web service composition and interoperation:** um eine bestimmte Aufgabe durchzuführen, kann ein Agent mehrere Web Services automatisch finden, aggregieren und verstehen, wie die einzelnen Services miteinander interagieren sollen. Hierfür ermöglicht DAML-S *„prerequesities and consequences of individual service use that are necessary for automatic service composition and interoperation"* deklarativ zu beschreiben.
- **Automatic Web service execution monitoring:** ein Agent kann den Status seines Requests abfragen bzw. die Ausführung eines Services beeinflussen. Dies erfordert Diskriptoren für Service execution. Dies wird momentan **NICHT** von DAML-S angeboten, ist aber für die Zukunft vorgesehen.

7.3.1.2. Top-Level Service Ontology

Die Ontologie SERVICE strukturiert sich in drei Bereiche (DAMLS-Coalition2003) wie es in der **Fehler! Verweisquelle konnte nicht gefunden werden.** zu sehen ist:

1. ***What does the service require of the user(s), or other agents, and provide for them?***

Die Antwort hierzu wird in "Profile" beschrieben (Klasse SERVICEPROFILE). Es beschreibt, „what the service does". Das Service Profil wird dazu benötigt, zu bestimmen, ob ein Service die Anforderungen von Service-Requester erfüllt. Allgemein wird der SERVICEPROFILE für *Discovery* benötigt. Danach ist es quasi nutzlos. DAML-S Profile beschreibt einen Service als "function of three basic type of information":

- What organization provides the service: Diese sind Informationen für den Menschen wie. Z.B. kontaktinformationen
- What function the service computes: dies wird in Form von Daten-Transformation und Statuswechsel beschrieben. Es werden die erwartete **input**, ausgegebene **output**, **preconditions** die von Service vorrausgesetzt warden, und die **expected effects,** welche nach der Ausführung der Service als Ergebnis auftreten. **Example: "***a selling service may require as a precondition a valid credit card and as input the credit card number and expiration date. As output it generates a receipt, and as effect the card is charged."*. Preconditions sind logische Bedingungen. Effects sind die Ergebnisse eines erfolgreichen Service execution. Die Repräsentation von Precondition und Effects hängt von der Repreäsentation von „rules" in der Sprache DAML ab. Momentan gibt es ***keine Empfehlungen***. Man beachte folgende Beschreibung für Input, Output, Precondition und Effects:

    - **Input** specifies one of the inputs of the service. It takes as value an instance of "ParameterDescription" that specifies an id of the input, a value and a reference to the corresponding input in the process model.
    - **Output** specifies one of the outputs of the service. It takes as value an instance of "ParameterDescription" that specifies an id of the output, a value and a reference to the corresponding output in the process model.
    - **Precondition** specifies one of the preconditions of the service. It takes as value an instance of "ParameterDescription" that specifies an id of the precondition, a value and a reference to the corresponding precondition in the process model.
    - **Effect** specifies one of the effects of the service. It takes as value an instance of "ParameterDescription" that specifies an id of the effect, a value and a reference to the corresponding effect in the process model.

- A host of features that specify charachteristics of the service: dies beinhaltet die Kategorie eines Services innerhalb eines bestimmten Klassifizierungssystems sowie Qualitätsbewertung (z.B. „very gut“, qucik to respond“, „sluggish“) innerhalb eines bestimmten ranking systems. Der Service-Requester muss nicht diese Daten benutzen.

2. ***How does it work?***

Dies wird in „model“ beschrieben (Klasse SERVICEMODEL). Es wird beschrieben, was beim Ausführen des Services passiert und wie der Service arbeitet. Ausserdem wird diese Beschreibung dazu benötigt, genauer zu analysieren, ob ein Service die Anforderungen des Service-Requester erfüllt. Weiter wird diese Beschreibung benötigt, um die Aktivitäten verschiedene Teilnehmer eines composed services zu koordinieren sowie die Ausführung eines Services zu beobachten (monitoring). Die Beschreibungen in Profile und Model können u.U. ***inkonsistent*** sein, ohne „affecting the validity of the DAML expression“. Folgende Besonderheiten hat die Service-Modellierung unter DAMlL-S:

- Services werden als Prozesse aufgefasst
- Ein Prozess beschreibt einen Service im Sinne von *inputs, (conditional) outputs,participant, precondition, and (conditional) effects*: “A process can have **any number of inputs**, representing the information that is, under some **conditions**, required for the execution of the process. It can have **any number of outputs**, the information that the process provides, **conditionally**, after its execution. Besides inputs and outputs, another important type of parameter specifies the **participants in a process**. A variety of **other parameters** may also be declared, including, for physical devices, such things as rates, forces, and knob settings. There can be **any number of preconditions**, which must all hold in order for the process to be invoked. Finally, the process can have **any number of effects**. **Outputs and effects can have conditions associated with them**.”
- Es gibt drei Sorten von Prozessen:
  - ***Atomic*** haben keine Subprozesse und sind unmittelbar und in einem Schritt (aus der Sicht des Service-Requesters) ausführbar.
  - ***Simple*** sind nicht unmittelbar ausführbar. Wie Atomic processes müssen sie in einem Schritt ausgeführt werden. Die Simple Processes dienen zu Kappselung oder Abstraktion von Atomic oder Composite processes.
  - ***composite*** Prozesse sind zerlegbar in anderen composite oder noncomposite Prozessen. Die Zerlegung kann mittels Kontrolstrukturen wie Sequence oder If-Then-Else spezifiziert werden. Jeder composite process hat einen ***CONTROLCONSTRUCT,*** der wiederum ein ***PROCESSCOMPONENT*** besitzt, welcher (COTROLCONSTRUCT) „the ordering and conditional execution of the subprocesses“ spezifiziert.
    - **Beispiel:** *“the control construct, SEQUENCE, has a components property that ranges over a PROCESSCOMPONENTLIST (a list whose items are restricted to be PROCESSCOMPONENTs, which are either processes or control constructs).”*
- Der Inhalt („the range“) dieser Parameter ist nicht eingeschränkt. Domainspezifische Ontologien sollen diese Wertebereiche einschränken oder detailierte spezifizieren
- Die Klasse PROCESSMODEL – auch ***Process Ontology*** genannt – kann für die Repräsentation von eine Vielfalt von Services dienen.
- Ein *process* erlaubt die Plannung, Komposition und Agent/Service Interoperation.
- PROCESS CONTROL MODEL – auch ***Process Control Ontology*** genannt – erlaubt das Monitoring von Service-Ausführung.
- Die beiden Ontologien setztn auf Arbeiten in standardisation of *planning languages*, work in *programming languages*, *distributed systems*, *process modelling*, *workflow automation* wie Process Specification Language (PSL) und Workflow Management Coalition effort (WFMC), work on *modelling verb semantics and event structure*, *action-inspired Web service markup*, *AI modelling complex actions*, *agent communication language*
- Für die Unterstützung von „Process“ und „Process Control“ definiert DAML-S *„ontology of resources“* und *„ontology of time“*.
- Folgende Kontrolstrukturen gibt es:
  - **Sequence**, A list of Processes to be done in order. We use a DAML+OIL restriction to restrict the components of a Sequence to be a List of process components -- which may be either processes (atomic, simple and/or composite) or control constructs.
  - **IF-Then-Else**, The IF-THEN-ELSE class is a control construct that has properties *ifCondition*, *then* and *else* holding different aspects of the IF-THEN-ELSE. Its semantics is intended as “Test *If-condition*; if True do *Then*, if False do *Else*.” (Note that the class CONDITION, which is a placeholder for further work, will be defined as a class of logical expressions.)

- **Repeat-Until**, The REPEAT-UNTIL class is similar to the REPEAT-WHILE class in that it specializes the IF-THEN-ELSE class where the *ifCondition* is the same as the *untilCondition* and different from the REPEAT-WHILE class in that the *else* (compared to *then*) property is the repeated process. Thus, the process repeats until the *untilCondition* becomes true.
- **Iterate**, ITERATE is a control construct whose *nextProcessComponent* property has the same value as the current process component. REPEAT is defined as a synonym of the ITERATE class. The repeat/iterate process makes no assumption about how many iterations are made or when to initiate, terminate, or resume. The initiation, termination or maintainance condition could be specified with a *whileCondition* or an *untilCondition*
- **Split**, The components of a *Split* process are a bag of process components to be executed concurrently. No further specification about waiting or synchronization is made at this level.
- **Split+Join**, Here the process consists of concurrent execution of a bunch of process components with barrier synchronization. With SPLIT and SPLIT+JOIN, we can define processes that have partial synchronization (e.g., split all and join some sub-bag).
- **Unordered**, Allows the process components (specified as a bag) to be executed in some unspecified order, or concurrently. All components must be executed. As with Split+Join, completion of all components is required. Note that, while the unordered construct itself gives no constraints on the order of execution, nevertheless, in some cases, there may be constraints associated with subcomponents, which must be respected.
- **Choice,** is a control construct with additional properties *chosen* and *chooseFrom*. These properties can be used both for process and execution control (e.g., choose from *chooseFrom* and do *chosen* in sequence, or choose from *chooseFrom* and do *chosen* in parallel) as well for constructing new subclasses like “choose at least n from m”, “choose exactly n from m”, “choose at most n from m”

- DAML-S hat ***lediglich eine propritäre Lösung für Einsatz von Variablen*** bzw. für Parameter Binding sowie Datenflussbeschreibung zw. Prozessen. Diese Lösung ist offenbar nur für diese Version entwickelt.
  - Oft ist es notwendig, verschiedene Properties bzw. Elemente eines Prozessmodells miteinander zu vergleichen oder aufeinander zu referenzieren (ähnlich wie Variablen in Programmiersprachen). Solche Fälle treten z.B. auf bei
    - In relating process inputs to the process' conditions, outputs or effects, including its preconditions, the conditions governing conditional effects and conditional outputs, and (properties of) the effects and outputs themselves.
    - In relating the inputs and outputs of a composite process to the inputs and outputs of its various component subprocesses.
    - In relating the inputs and outputs of elements of a composite process definition to parameters of other process components. For example, when a composite process is defined as a sequence of subprocesses, the output of one component of the sequence may well be an input to a subsequent component of the sequence.
  - DAML+OIL unterstützt solche Variablen nicht. DAML-S hat folgenden primitiven Ansatz:
    - “The intent is to capture, purely as a set of process annotations, this critical information about how processes are to be instantiated and information shared between process elements. We have extended our process ontology with the classes and properties used in this notation. The use of this notation in a process definition will enable a **specialized DAML-S process reasoner** to use this information to determine which properties should have “the same value” in any coherent instance of the process being defined.“
    - “In this notation, an instance of the class VALUEOF, with properties *atClass* and *theProperty* denotes the object (value) of the specified property at the specified class. **This style of reference is intended to be used only within the context of a process being annotated using the property *sameValues*, which relates a process class to a collection of VALUEOF objects.** The set of referenced *ValueOf* elements are considered to share the same information, as if their values were represented by a single variable.“
- DAML-S sieht ***keine Definitionen für Process Control Ontology*** in der jetzigen Version vor, jedoch wird sie für die Zukunft geplannt.

3. ***How is it used?***

Dies wird in “grounding” (Klasse SERVICEGROUNDING) beschrieben. Service Grounding beschreibt, wie ein Agent den Service benutzen soll. Hauptsächlich geht es um Protokolle und Nachrichtenformat, „serialization“, Transport und Adressierung. Service Grounding kann als ***„mapping from an abstract to a***

concrete specification" von Servicebeschreibungselemente (z.B. Input, Output von atomaren Prozessen) verstanden werden.
DAML-S benutzt das Web Service Definition Language (***WSDL***) und SOAP für concrete message specification.

7.3.1.3. Resources

Prozesse benötigen Ressourcen. DAML-S bietet eine Resource Ontology an. Es bietet Grundelemente wie *„resource type, such as fuel", „resource tokens, such as the fuel in the gas tank of a particular car"* und *„capacity, such as the five gallons of fuell in the car's tank right now"*an. Ressource können

- be consumed
- be replenished
- be locked, and
- released

Ressourcenn werden von Activities oder Prozessen alloziert. Der Status einer Ressource nach seiner Allokation und Benutzung gibt seinen *„**AllocationType"*** an. Wenn eine Ressource nach der benutzung „alle" ist, wird die als *„**ConsumableAllocation**"* und sonst als *„**ReusableAllocation**"* genannt. Beispiele dafür sind jeweils „Batterie" bzw. „Belegung eines Ortes". Ressourcen können nicht benutzt werden, wenn Sie „locked" sind.

Preconditions von Prozessen können oft als die Verfügbarkeit von einige Ressourcen angesehen werden (z.B. location precondition, access precondition). Im Allgemeinen kann man das Produkt oder die Ausführung eines Prozesses als eine Ressource ansehen, wenn der Prozess als precondition eines anderen Prozesses ausgeführt wurde.

Eine Ressource hat eine ***"CapacityType"***, die entweder ***„DiscreteCapacity"*** oder ***"ContinousCapacity"*** ist:

> *Capacity can be related to various other resource-theoretic predicates. In the following rules, for future incorporation into the DAML ontology, R stands for a resource, A for an activity, T for a time interval, and t for a time instant. The expression use(A,R,T/t) means that activity A uses resource R over time interval T or for time instant t. The expression capacity(R,T/t) refers to the capacity of resource R over time interval T or for time instant t. The capacity of a persistent resource at the beginning of its use is the same as at the end.*
>
> $$reusable(R)\ \&\ use(A,R,T) \Rightarrow capacity(R,start(T)) = capacity(R,end(T))$$
>
> *The quantity of a consumable resource at the beginning of its use is more than at the end.*
>
> $$consumable(R)\ \&\ use(A,R,T) \Rightarrow capacity(R,start(T)) > capacity(R,end(T))$$
>
> *When an agent replenishes a resource during period T, there is more after the replenishment.*
>
> $$replenish(A,R,T) \Rightarrow capacity(R,start(T)) < capacity(R,end(T))$$
>
> *When a reusable resource is used for period T, it is locked at the beginning of T and released at the end.*
>
> $$reusable(R)\ \&\ use(A,R,T) \Rightarrow lock(A,R,start(T))\ \&\ release(A,R,end(T))$$
>
> *Capacities of resources can also have a capacityGranularity, that is, the units in terms of which the capacity is measured.*

Eine Ressource kann atomar oder aggregiert sein. Deshalb gibt es ***„AtomicResource"*** und ***„AggregateResource"*** als Subklassen von Resource Klasse.

DAML-S unterscheidet zwischen "***unit capacity atomic resources"***, welcher entweder für eine Aktivität zu verfügung steht oder nicht, sowie ***„batch capacity atomic resources"***, die vmultiple Aktivitäten synchron unterstützen können. *UnitCapacityResource* und *BatchCapacityResource* sind Subklassen von *AtomicResource*.

Aggregationen können conjunctive oder disjunctive sein. Bei ***conjunctive aggregates*** müssen alle Elemente für eine Aktivität alloziert warden. Bei ***disjunctive aggregate*** können nur eine Teilmenge der Elemente

alloziert werden. Ein Beispiel für disjunctive resource ist ein Prozess, dass 3 benachbarte Stühle von 100 Stühlen in einem Raum benötigt. *ConjunctiveAggregateResource* und *DisjunctiveAggregateResource* sind subclasses von *AggregateResource*.

7.3.1.4. Zusammenfassung

DAML-S ermöglicht eine semantische Beschreibung von Services. Services werden als Prozesse aufgefasst, die atomat oder composite sein können. Wie ein Service von atomaren oder anderen composite services zusammengebaut wird, ist in Process Model Ontology beschrieben. Vor allem wird der composite Prozess im Sinne von *inputs, (conditional) outputs,participant, precondition, and (conditional) effects* in der Form von „information transformation and state changes" beschrieben. Hierzu bietet DAML-S Process Control Ontologien und Konstrukte wie SEQUENCE, ITERATE, REPEAT-UNTIL, IF-THEN-ELSE, ORDERED, CHOICE, etc. an. **Example: "***a selling service may require as a precondition a valid credit card and as input the credit card number and expiration date. As output it generates a receipt, and as effect the card is charged.".* Preconditions sind logische Bedingungen. Effects sind die Ergebnisse eines erfolgreichen Service execution. Die Repräsentation von Precondition und Effects hängt von der Repreäsentation von „rules" in der Sprache DAML ab.
Die Sprache und Ontologien sind noch im entstehen. Einige Features werden gar nicht oder nur als vorübergehende properitäre Lösung angeboten. Hierzu gehören vor allem

- Keine Empfehlung für Repräsentation von *„rules"* (logischen Ausdrücken) in DAML-S
- „Process Control Ontology",
- Benutzung von Variablen bei der Beschreibung von Prozessmodellen (Input, Output, precondition, effect),
- Beschreibung von KontrollKonstrukte sind nicht vollständig definiert (DAMLS-Coalition03). Das Element CONDITION z.B., die auch in IF-THEN-ELSE vorkommt, ist noch nicht definiert: („Note that the class CONDITION, which is a placeholder for further work, will be defined as a class of logical expressions.")
- Die Beschreibungen in ServiceProfile und ServiceModel können u.U. ***inkonsistent*** sein, ohne „affecting the validity of the DAML expression".

Literaturreferenzen zu [Kapitel 7.3.1]

[DAMLS-2003] "DAML-based Web Service Ontology". Version 0.9 , Mai 2003. Retrieved on 09.12.2003 from DAML Web site: http://www.daml.org/services/

[DAMLS-Coalition03] The DAML Service Coalition (2003). "DAML-S: Semantic Markup or Web Services". White Paper. Retrieved on 09.12.2003 from DAML-S Web site: http://www.daml.org/services/daml-s/0.9/daml-s.pdf

7.3.2. PDDL

7.3.2.1. Was kann man mit PDDL machen?

Mit PDDL kann man eine Umgebung („the physics of a domain") beschreiben. Zur Beschreibung der Umgebung zählen (PDDL1998)

- Die vorhandenen Prädikate („predicates")
- Mögliche Operationen ("Basic STRIP Style actions") sowie "hierarchical actions", die aus "subactions" and "subgoals" bestehen
  - Ein action ohne Expansion ist ein primitive action.
  - Ein Action kann durch eine ***series-parallel*** Kombination oder eine beliebige ***partial order*** aus anderen Actions zusammengesetzt werden.
  - Ein action muss entweder einen **„Effect"** haben **oder** eine **„expansion".** Nicht aber beides.
  - Expansions können mittels einfachere Aktionen und Kontrollstrukturen beschrieben werden
  - **Constraints**: Man kann „Goal Definitions" angeben, die während der Ausführung eine Aktion gelten müssen.
  - Folgende Kontrollstrukturen gibt es:

- ***Choise,***
- ***Foresome,***
- ***Series,***
- ***Parallel,***
- ***Foreach,***
- ***In-context***: This construct is used to declare preconditions and maintenance conditions of actions that are due purely to their occurring in the context of this expansion. For example, to indicate a plan to evacuate an area of friendly forces and then shell it, one might write

  (series (clear ?area)
  (in-context (shell ?area)
  :precondition (not (exists (?x - unit)
  (and (friendly ?x) (in ?x ?area))))))
- ***Constrained***: The *(constrained A C*)* syntax allows fairly arbitrary further conditions to be imposed on an action spec, with labels standing in for actions and their endpoints. The labels are defined by the *(tag labels action)* construct. A label stands for the whole action (occurrence) unless it is qualified by "<" or ">", in which case it stands for the beginning or end of the action. Inside *C*, *(series l_1 l_2 ... l_k)* imposes an additional ordering requirement on the time points tagged $l_1,\ldots, l_k$. *(in-context (series l_1 ... l_k) -conditions-)* can be used to impose extra conditions (or announce extra effects) of the interval corresponding to such an additional ordering.
  For example, to expand an action into four subactions (A), (B), (C), and (D), such that (A) precedes (B) and (D), and (C) precedes (D), with condition (P) maintained from the end of (A) until the end of (D), write

  :expansion (constrained ((series (tag A (> end-a)) (B))
  (series (C) (tag (< beg-d) (D) (> end-d))))
  (in-context (series end-a beg-d end-d)
  :maintain (P)))
- ***only-in-expressions***: Solche Actions können nur dann in einem Plann aufgenommen warden, wenn sie Teil einer Expansion eines anderen Action sind. Dadurch kann man spezifizieren, dass man nicht alle *preconditions* dieser Action nicht kennen. Wir kennen aber einige *standard contexts,* in dennen diese Aktion ausgeführt werden kann, wo offenbar die *preconditions* gegeben sein zu sein scheinen.

- „Effects" von Operationen
  - „Effects list the changes which the action ompose on the current state of the world"
  - **Effects** may be universally quantified or conditional, but **full first order sentences. Disjunctions and Skolem functions are not allowed.**
  - man kann **NICHT** zw. Primary effects und Side-effects unterscheiden.
- Preconditions,
  - die als goal description beschrieben werden
  - **„arbitrary function-free first order logical sentence"**
- Goals
  - Wird benutzt um die Ziele einer Plannung oder die preconditions einer Operation zu spezifizieren
  - Erlaubt sind function-free first order predicate logic (including nested quantifiers)
  - As in STRIPS, the truth value of predicates are assumed to persist forward in time.
  - Unlike STRIPS, PDDL has no delete list --- *instead of deleting **(on a b)** one simply asserts **(not (on a b))***. If an action's effects does not mention a predicate ***P*** then the truth of that predicate is assumed unchanged by an instance of the action. The semantics of ***(when P E)*** are as follows: If ***P*** is true before the action, then effect ***E*** occurs after. ***P*** is a *secondary precondition*. The action is feasible even if ***P*** is false, but the effect ***E*** occurs only if ***P*** is true.
- Dynamic axioms over satisfied theories
- Specification over safety constraints [z.B. Objekt x darf nie in Location z existieren.]
  - Sind back-ground goals
  - A plan is allowed only if at its end none of these background goals are false.
- Probleme
  - Problem ist das, was der Planner zu lösen versucht
  - Ein Problem spezifiziert ***„Initial Situation"*** und ein zu erreichendes „***Goal"***

- A *PROBLEM* definition must specify either an initial situation by name, or a list of initially true literals, or both.
- The goal of a problem definition may include a goal description or an expansion, or both. A solution to a problem is a series of actions such that
  - the action sequence is feasible starting in the given inital situation situation;
  - the ***:goal***, if any, is true in the situation resulting from executing the action sequence;
  - the ***:expansion***, if any, is satisfied by the series of actions.
  - For instance, in the transportation domain, one might have the problem

```
(define (problem transport-beans)
    (:domain transport)
    (:situation standard-network)
    (:init (beans beans27)
        (at beans27 chicago))
    (:expansion (constrained (tag (carry-in-train
                                beans27 chicago newyork)
                            (> end))
                    (in-context end
                        :precondition (not (spoiled beans27))))))
```

- Management of multiple problems in multiple domains using different subsets of language features (to support sharing of domains across different planners that handle varying levels of expressiveness).

7.3.2.2. Beispiel

In diesem Beispiel können Objekte zw. „Home“ und „Office“ mittels eines „briefcase“ transportiert werden. Es gibt hierbei „effects“ als „universal quantification“ (alle Objekte sind entfernt worden) und „conditional effects“ (PDDL1998):

```
(define (domain briefcase-world)
  (:requirements :strips :equality :typing :conditional-effects)
  (:types location physob)
  (:constants (B - physob))
  (:predicates (at ?x - physob ?l - location)
               (in ?x ?y - physob))

  (:action mov-b
    :parameters (?m ?l - location)
    :precondition (and (at B ?m) (not (= ?m ?l)))
    :effect (and (at b ?l) (not (at B ?m))
                 (forall (?z)
                    (when (and (in ?z) (not (= ?z B)))
                         (and (at ?z ?l) (not (at ?z ?m)))))) )

  (:action put-in
    :parameters (?x - physob ?l - location)
    :precondition (not (= ?x B))
    :effect (when (and (at ?x ?l) (at B ?l))
                  (in ?x)) )

  (:action take-out)
    :parameters (?x - physob)
    :precondition (not (= ?x B))
    :effect (not (in ?x)) )
```

Literaturreferenzen zu [Kapitel 7.3.2]

[PDDL1998] “PDDL-The Planning Domain Definition Language”. Version 1.2, October 1998. Retrieved on 09.12.2003 from PDDL Web site: ftp://ftp.cs.yale.edu/pub/mcdermott/software/pddl.tar.gz

7.3.3. Relevanz für DynAMITE

Es ist schwierig einer der beiden Sprachen (PDDL und DAML-S) für DynAMITE zu favorisieren.

- Zunächst eignen sich beide Sprachen für DynAMITE, wenn es um die Realisierung dynamischer Planung von Strategien sowie „goal description“ geht.
- Einige DAML-S Features sind noch im entstehen oder sind noch nicht standardisiert. Auf der anderen Seite ist DAML-S die Standard für Semantic Web (W3C Standard). Damit hat DAMl-S ein Potential sich durchzusetzen. Außerdem werden viele Tools für DAML-S geben. Man muss aber berücksichtigen, dass W3C einen neuen Standard (OWL) raus bringt, das DAML-S, DAML+OIL etc. vereint und ergänzt. Dieser Standard dürfte sich durchsetzen.
- Auf der anderen Seite hat DynAMITE Konsortium eine positive Erfahrung mit PDDL. Außerdem ist PDDL die Sprache für Planungssysteme. Ferner erfordert DAML-S Kenntnisse von vielen XML Technologien (RDF, DAML, XSLT, XML), bevor man irgend etwas in DAML-S anfängt. Dagegen ist PDDL relativ leicht zu verstehen und lehnt sich an allgemein bekannter Lisp Syntax an.
- **Alles in einem sollte PDDL bevorzugt werden**.

Hier einige für DynAMITE interessante Aussagen über DAML-S:

DAML-S ermöglicht eine semantische Beschreibung von Services. Services werden als Prozesse aufgefasst, die atomar oder composite sein können. Wie ein Service von atomaren oder anderen composite services zusammengebaut wird, ist in Process Model Ontology beschrieben. Vor allem wird der composite Prozess im Sinne von *inputs, (conditional) outputs,participant, precondition, and (conditional) effects* in der Form von „information transformation and state changes“ beschrieben. Hierzu bietet DAML-S Process Control Ontologien und Konstrukte wie SEQUENCE, ITERATE, REPEAT-UNTIL, IF-THEN-ELSE, ORDERED, CHOICE, etc. an.
Example: “*a selling service may require as a precondition a valid credit card and as input the credit card number and expiration date. As output it generates a receipt, and as effect the card is charged.”*.

Preconditions sind logische Bedingungen.
Effects sind die Ergebnisse eines erfolgreichen Service execution. Die Repräsentation von Precondition und Effects hängt von der Repräsentation von „rules“ in der Sprache DAML ab.
Die Sprache und Ontologien sind noch im entstehen.
Einige Features werden gar nicht oder nur als vorübergehende proprietäre Lösung angeboten. Hierzu gehören vor allem

- Keine Empfehlung für Repräsentation von *„rules“* (logischen Ausdrücken) in DAML-S
- „Process Control Ontology“,
- Benutzung von Variablen bei der Beschreibung von Prozessmodellen (Input, Output, precondition, effect),
- Beschreibung von Kontrollkonstrukten sind nicht vollständig definiert (DAMLS-Coalition03). Das Element CONDITION z.B., die auch in IF-THEN-ELSE vorkommt, ist noch nicht definiert: („Note that the class CONDITION, which is a placeholder for further work, will be defined as a class of logical expressions.“)

Die Beschreibungen in ServiceProfile und ServiceModel können u.U. ***inkonsistent*** sein, ohne „affecting the validity of the DAML expression“.

7.4. Content-Modell

7.4.1. Einleitung

Heutzutage werden Inhalte von Multimediadaten auf unterschiedliche Art beschrieben. Z.B. findet man in Peer-to-Peer Netzwerken häufig viele Informationen zu Multimedia-Dateien in deren entsprechend langen Namen kodiert. Der Titel einer Musikdatei kann sich so aus dem Interpreten, dem Album, dem Titel und der Bitrate zusammensetzen. Die Art der Kodierung ist unstandardisiert.
Als Lösung werden zunehmend Header von Dateien für Informationen über die enthaltenen Multimedia-Inhalte genutzt. Die darin enthaltenen Informationen beschreiben hauptsächlich technische und kaum semantische Aspekte. Entsprechende Standards sind meist für eine Anwendungsdomäne wie Musik, Video oder Literatur. spezifiziert und nicht interoperabel. Außerdem sind die Standards für viele Anwendungen zu unflexibel [>WSI01]. Trotz der großen Verbreitung von MP3-Dateien gibt es heute noch keine kommerziellen Anwendungen, die die darin enthaltenen Informationen für einen automatisierten Zugriff auf Dateien nutzen. Da die Daten in der Datei kodiert sind, wären Suchfunktionen zu aufwändig. Bestenfalls könnten die Informationen automatisch extrahiert werden und in veränderter Form in Datenbanken oder anderen Strukturen für Suchfunktionen zur Verfügung stehen. Auch für Präsentationszwecke sind derartige

Informationen häufig nicht zu gebrauchen. Zwar nutzen z.B. MP3-Player häufig Informationen in MP3-Headern für Anzeigen auf einem Display, aber für andere Anwendungen, wie z.B. den Internetvertrieb, sind die entsprechenden Standards aufgrund ihrer mangelnden Flexibilität und fehlenden Erweiterbarkeit unbrauchbar. Hier werden zusätzliche Informationen wie semantische Beschreibungen, Kritiken, Bilder, Kurzauszüge und Verweise auf ähnliche Daten benötigt. Den entsprechenden Standards fehlen insbesondere die Möglichkeit der Verweise und Relationen.
Außerdem eigen sich in Multimedia-Dateien enthaltene Informationen nicht zur Beschreibung von Kompositionen aus mehreren Dateien, wie z.B. Musik- und Fotoalben oder Web-Seiten.

7.4.2. DIDL

Als Lösung der beschriebenen Probleme hat die Moving Picture Expert Group (MPEG) die „Digital Item Declaration Language“ (DIDL) spezifiziert [MPEG02,CP01].
Das Hauptziel der DIDL ist die Etablierung

- eines einheitlichen erweiterbaren und flexiblen Standards für die Beschreibung digitaler Inhalte.

Folgende Anforderungen soll die DIDL erfüllen:

- Separierbarkeit der digitalen Ressourcen von den zugehörigen beschreibenden Informationen.
- Unabhängigkeit von den Datentypen der Ressourcen.
- Unabhängigkeit von Beschreibungsschemas unterschiedlicher Anwendungsgebiete.
- Unterstützung von Kompositionen.
- Effizientes Suchen und Browsen.
- Identifizierbarkeit digitaler Inhalte.

DIDL führt dazu abstrakte Konzepte ein, die zusammen ein wohldefiniertes Datenmodell für komplexe digitale Objekte bilden. Basierend auf diesem „Digital Item Declaration Model" definiert DIDL ein W3C XML-Schema. Dieses Schema soll aufgrund der Flexibilität, Erweiterbarkeit und der hohen Abstraktionsstufe die Basis für Anwendungen aus vielfältigen Anwendungsdomänen bilden können.

7.4.2.1. MPEG-21

DIDL ist Teil des MPEG-21 Standards [MPEG01, BWHBP03], der ein ganzheitliches Multimedia-Framework für digitale Inhalte definieren soll. Ziel von MPEG-21 ist die Abdeckung der gesamten Verarbeitungskette und aller Aspekte digitaler Inhalte, von der Erzeugung und Produktion über die Verarbeitung und Verbreitung bis zur Nutzung.
MPEG-21 besitzt eine modulare Architektur und teilt sich unter anderem auf in folgende Teile („Parts“), von denen sich noch einige im Spezifikationsprozess befinden [MPEG01]:

- Part 1: Architektur des Frameworks und funktionale Anforderungen an MPEG-21.
- Part 2: Die DIDL dient zur Beschreibung komplexer digitaler Objekte
- Part 3: Die „Digital Item Identification Language“ (DIIL) dient der Identifikation komplexer digitaler Objekte.
- Part 4: „Intellectual Property Management and Protection“ (IPMP) soll ein Framework zum Schutz intellektuellen Eigentums an digitalen Inhalten beschreiben.
- Part 5: Die „Rights Expression Language“ (REL) dient der Beschreibung von Rechten an digitalen Inhalten.
- Part 7: Die „Digital Item Adaption“ (DIA) beschreibt die Transkodierung und Anpassung digitaler Inhalte.
- Part 10: Das „Digital Item Processing“ (DIP) behandelt die Verbindung von Verarbeitungsmethoden mit digitalen Inhalten.

7.4.2.2. Schema

Die folgenden Elemente bilden den Kern des DIDL-Schemas [MPEG02,CP01b]:

- <CONTAINER>: Ein Container setzt sich aus Containern und/oder Items zusammen. Er dient zur Gruppierung und erlaubt so z.B. den Aufbau logischer Pakete.
- <ITEM>: Ein Item besteht aus Items und/oder Components. Ein Item, dass keine anderen Items enthält, kann als logisch unteilbares digitales Objekt aufgefasst werden
- <COMPONENT>: Eine Component enthält eine oder mehrere Ressourcen. Sie dient der Gruppierung dieser Resourcen nach entsprechenden Eigenschaften, die in dem <DESCRIPTOR> der Component aufgezählt sind (siehe unten).

- <RESOURCE>: Eine Ressource ist ein individuell identifizierbarer Datenbestand, z.B. ein Bild, ein Text, eine Audio-Datei oder ein Video-Stream. Eine Ressource kann auch ein physikalisches Objekt sein, wie z.B. ein Buch. Ressourcen müssen über eine eindeutige Adresse lokalisierbar sein. Normalerweise werden Ressourcen per Referenz über diese eindeutige Adresse definiert. Eine Ressource kann allerdings alternativ die Daten auch direkt als <CDATA> enthalten.
- <DESCRIPTOR>: Zusätzliche Informationen zu Containern, Items oder Components können durch Descriptoren ausgedrückt werden. Zum Beispiel können in Components enthaltene Descriptoren Bit-Raten oder Verschlüsselungs-Informationen beschreiben, die auf alle Ressourcen der Component zutreffen.

Abbildung 22 verdeutlicht die Hierarchie dieser Elemente des DIDL-Schemas. Zusätzlich kennt DIDL folgende Elemente: <DIDL>, <DECLARATIONS>, <STATEMENT>, <ANCHOR>, <CHOICE>, <SELECTION>, <CONDITION>, <OVERRIDE>, <REFERENCE>, <ANNOTATION> und <ASSERTION>. Eine Beschreibung dieser Elemente findet man in [MPEG02].

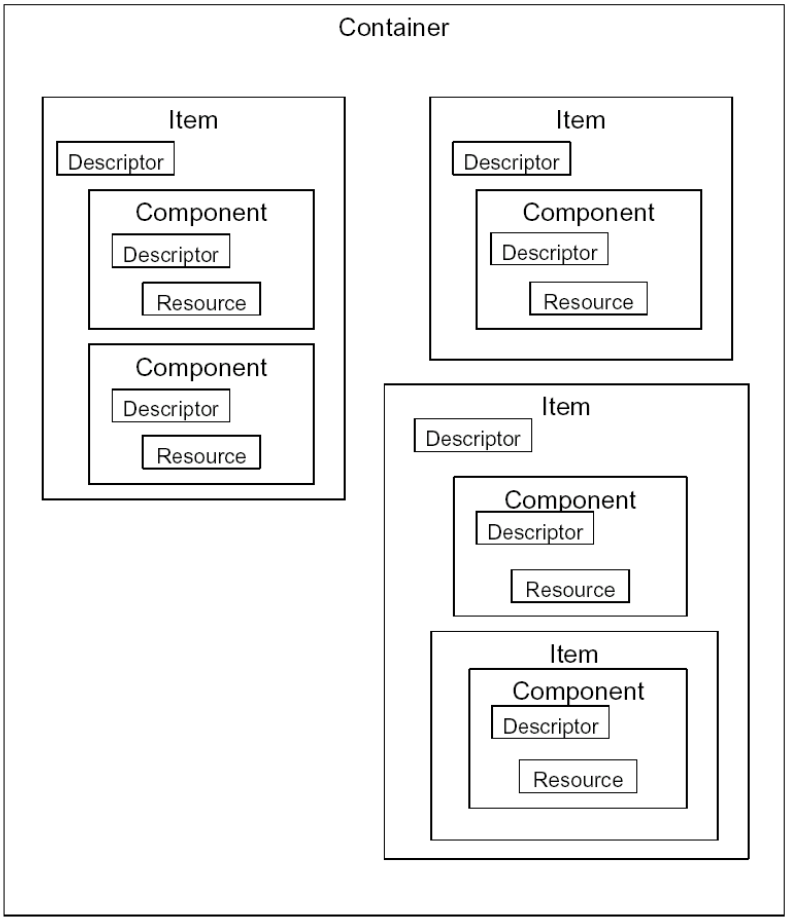

Abbildung 22: Schematische Hierarchie des Datenmodells der DIDL, bzw. der Elemente des entsprechenden XML-Schemas.

Abbildung 23 verdeutlicht den Aufbau einer DIDL-Beschreibung am Beispiel einer Sammlung von Bildern, die in verschiedene Foto-Alben aufgeteilt sind. DIDL erlaubt unterschiedliche Darstellungen des gleichen digitalen Inhalts. So könnte jedes einzelne Fotoalbum im Beispiel der Abbildung 23 anstatt als <ITEM> auch als <CONTAINER> definiert werden. Man beachte die Möglichkeit einem digitalem Item mehrerer Ressourcen („myFirstPicture.jpg“ und „myFirstPicture.bmp“) zuzuordnen. Die <COMPONENT> dient dazu diese verschiedenen Ressourcen einer Beschreibung, hier „Auf der Fahrt in den Urlaub!“ zuordnen.

```
<DIDL>
  <CONTAINER>
    <DESCRIPTOR>
      <STATEMENT TYPE=”/text/plain”>My Photo Albums</STATEMENT>
```

```
    </DESCRIPTOR>
    <ITEM>
      <DESCRIPTOR>
        <STATEMENT TYPE=”/text/plain”>Photo Album 1</STATEMENT>
      </DESCRIPTOR>
      <ITEM>
        <DESCRIPTOR>
          <STATEMENT TYPE=”/text/plain”>Auf der Fahrt in den Urlaub!</STATEMENT>
        </DESCRIPTOR>
        <COMPONENT>
          <RESOURCE REF="myFirstPicture.jpg" TYPE="<some preamble>/image/jpg" />
          <RESOURCE REF="myFirstPicture.bmp" TYPE="<some preamble>/image/bmp" />
        </COMPONENT>
      </ITEM>
      <ITEM>
        <DESCRIPTOR>
          <STATEMENT TYPE=”/text/plain”>Im Urlaub am Strand!</STATEMENT>
        </DESCRIPTOR>
        <COMPONENT>
          <RESOURCE REF="mySecondPic.bmp" TYPE="<some preamble>/image/bmp" />
        </COMPONENT>
      </ITEM>
    </ITEM>
    <ITEM>
      <DESCRIPTOR>
        <STATEMENT TYPE= <some preamble>/text/text >Photo Album 2</STATEMENT>
      </DESCRIPTOR>

    ...

    </ITEM>
  </CONTAINER>
</DIDL>
```

Abbildung 23 DIDL-Repräsentation eines digitalen Inhalts am Beispiel einer Sammlung von Bildern.

7.4.3. DIDL-Lite

Die UPnP-Arbeitsgruppe AV hat für die Beschreibung multimedialer Inhalte das W3C-Schema DIDL-Lite entwickelt [UPNP02]. DIDL-Lite basiert auf einer Teilmenge der Elemente der DIDL (vergleiche Abschnitt 7.4.2.2). Das Schema importiert die Namespaces „Dublin Core (dc)“ und „UPNP (upnp)“ für zusätzliche Elemente, wie z.B. <dc:title> oder <upnp:class> [DC03, UPNP01].
DIDL-Lite kennt folgende Elemente:

- <DIDL-Lite>: Entspricht dem Root-Element <DIDL>.
- <desc>: Entspricht dem DIDL-Element <DESCRIPTOR> und kann beliebige Metadaten enthalten.
- <res>: Entspricht dem DIDL-Element <RESOURCE> und kennzeichnet eine Multimedia-Objekt, wie z.B. ein Foto, Lied oder Video.
- <item>: Entspricht dem DIDL-Element <ITEM>. Es kann verschiedene 1. Dublin Core (<dc>), 2. <upnp>, 3. <res> und 4. <desc>- Elemente beinhalten. Im Unterschied zu DIDL darf ein DIDL-Lite <item> kein weiteres <item> und keinen <container> enthalten. Es repräsentiert also eine nicht weiter zerlegbare Entität.
- <container>: Entspricht dem DIDL-Element <CONTAINER>. Es kann verschiedene 1. Dublin Core (<dc:*>), 2. <upnp:*>, 3. <res>, 4. <ref>, 5 <item>, 6. <container> und 7. <desc>- Elemente beinhalten.

Eine genaue Beschreibung der Elemente, sowie die möglichen Attribute kann man dem Schema und seine Beschreibung entnehmen [UPNP01].

7.4.4. Aussicht

Durch die große Marktrelevanz bisheriger MPEG-Standards finden auch die verschiedenen Aktivitäten im Rahmen von MPEG-21 in Wissenschaft und Industrie große Beachtung. Es ist zu erwarten, dass auch die

MPEG-21 Standards inklusive DIDL einen erheblichen Einfluss auf den Multimedia-Markt haben werden. Nicht zuletzt hat dieses Argument z.B. den Ausschlag gegeben, dass DIDL-konforme Dokumente als Standard für die Beschreibung komplexer digitaler Objekte in der Los Alamos National Laboratory Digital Library gewählt wurden [BHS03].

Literaturreferenzen zu [Kapitel 7.4]

[>BHS03] Jeroen Bekaert, Patrick Hochstenbach, Herbert Van de Sompel: Using MPEG-21 DIDL to represent complex digital objects in the Los Alamos National Laboratory Digital Library, D-Lib Magazine, Vol. 9, No. 11, November 2003, http://webdoc.gwdg.de/edoc/aw/d-lib/dlib/november03/bekaert/11bekaert.html.

[>BWHBP03] I. Burnett, R. Van de Walle, K. Hill, J. Bormans, F. Pereira: MPEG-21: Goals and achievements. IEEE Multimedia, pp. 60-70, Oct-Dec 2003.

[>CP01] Cover Pages: MPEG-21 Part 2: Digital Item Declaration Language (DIDL), Technology Report, Juni 2001, http://xml.coverpages.org/mpeg21-didl.html.

[>CP01b] Cover Pages: W3C XML Schema for DIDL XML Document Type. Extracted from Section 6 of [MPEG02], http://xml.coverpages.org/didl-schema2001.txt.

[>MPEG01] ISO/IEC: MPEG-21 Overview, October 2002, http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm.

[>MPEG02] ISO/IEC: MPEG-21 Digital item Declaration, July 2001, http://ipsi.fraunhofer.de/delite/Projects/MPEG7/Documents/n4248.html.

[>UPNP02] UpnP Forum: ContentDirectory:1 Service Template Version 1.01, Juni 2002, http://www.upnp.org/download/ContentDirectory 1.0.prtad.pdf.

[>WSI01] Mark Walker, Todd Schwartz, Vaughn Iverson: DIDL: Packaging Digital Content. XML.com, Mai 2001, http://www.xml.com/pub/a/2001/05/30/didl.html.

[>DC03] Dublin Core Metadata Element Set, Version 1.1: Reference Description, 2003, http://dublincore.org/documents/dces/

8. Spezifikations- und Modellierungswerkzeuge

8.1. UML

Die Unified Modeling Language (UML, [>PS99] [>FS00], [>RJB99], [>JRHZQ03]) ist eine (grafische) Modellierungssprache zur

- Visualisierung,
- Spezifizierung,
- Konstruktion und
- Dokumentation

von Modellen für Software-Systeme. Sie liefert Notationselemente für die

- dynamischen und
- statischen

Modelle aus Analyse, Design und Architektur und unterstützt insbesondere objektorientierte Vorgehensweisen.

Die UML basiert auf einer Reihe von objektorientierten Modellierungsansätzen für die Entwicklung von Softwaresystemen mit komplexen Datenstrukturen, die Anfang der 90er Jahre entwickelt wurden. Wesentliche Einflüsse hatten "Object-Oriented Software-Engineering" (Jacobson's Objectory), "Object-Modeling Technique" (Rumbaugh's OMT) und "Object-Oriented Analysis and Design" (die Booch-Methode), die 1995 in einem gemeinsamen Ansatz "Unified Modeling Language" mündeten. Zahlreiche weitere Einflüsse, wie zum Beispiel SDL (für die Telekommunikation) oder „Statecharts“ (für eingebettete Steuerungssysteme), fanden konzeptuellen Eingang in die UML.

Im Jahr 1997 wurde die UML von der Object Management Group (OMG [>OMG97]), dem für die Standardisierung von UML verantwortlichen Gremium, zum internationalen Standard erhoben. UML hat sich seitdem in wenigen Jahren als Standardsprache für die Modellierung von Software durchgesetzt. Seit ihrem ersten Erscheinen wurde UML mehrmals revidiert. Die bisher letzte Version trägt die Nummer 1.5. Aktuelle Bücher und Tools basieren größtenteils auf den Versionen 1.3 und 1.4. Mit der Integration des XML Metadata Interchange Format (XMI) zur UML 1.3 wurde die zuvor ausschließlich graphische Elemente umfassende Modellierungssprache um eine textuelle Notation gleicher Mächtigkeit ersetzt [>JRHZQ03]. Die Versionen 1.1 bis 1.5 sind jedoch alles kleine Revisionen, welche die Substanz der Sprache nicht verändert haben.
Die von der OMG zur Weiterentwicklung von UML eingesetzte UML Revision Task Force (RTF) arbeitet momentan an der nächsten umfangreicheren Revision von UML in der Version 2.0. Nach einem mehrjährigen Konsensbildungsprozess liegt eine erste vom OMG „Board of Directors“ angenommene Spezifikation vor. Dieser offizielle Standard ist aber noch änderbar und wird voraussichtlich erst durch eine weitere Bestätigung des OMG „Board of Directors“ im Herbst 2004 eine gültige Spezifikation [>RUPP03].
UML 2.0 bringt in einzelnen Modellierungsbereichen substanzielle Änderungen und Verbesserungen gegenüber UML 1.x. Insbesondere der Bereich der Architekturmodellierung wurde komplett überarbeitet. Ferner gibt es erhebliche Änderungen im Bereich der Modellierung von Abläufen und Verhalten, welche insbesondere eine bessere und umfassendere Generierung von Code aus UML-Modellen ermöglichen und das Konzept der Model-Driven Architecture (MDA) unterstützen. Auch das UML-Metamodell wurde erheblich überarbeitet.

Weil in technischen Applikationen wie z.B. der Unterhaltungselektronik der Softwareanteil rapide steigt, wird auch dort der Einsatz der UML immer interessanter. Um den Anforderungen dieses Anwendungsbereichs gerecht zu werden, wurde unter dem Namen UML-RT eine Erweiterung der UML entwickelt. Das Sprachprofil UML-RT [>SELIC99] ermöglicht eine klare Beschreibung der Architektur technischer Anwendungen und bietet einfache Mittel zur Modellierung von Echtzeit an. Obwohl sie keine ausgeprägte Unterstützung für spezielle Anforderungen wie z.B. regelungstechnische Aufgaben bietet, reicht ihre Ausdrucksmächtigkeit für viele Anwendungen aus. UML-RT zeigt, dass die UML durch ihre Erweiterbarkeit ausreichend Potential bietet, sich auch in technischen Anwendungsfeldern als standardisierte Modellierungssprache zu etablieren. Auf Basis der heutigen Form der UML-RT ist die Definition weiterer, spezifischer Anwendungsprofile denkbar. So wird derzeit intensiv an Anwendungsprofilen der UML für die Automobil-Industrie gearbeitet [>DC03].

UML hat sich als Standardsprache zur Modellierung von Software durchgesetzt und entsprechend viele Bücher sind veröffentlicht. Nahezu alle neuen Lehrbücher im Bereich Softwareentwicklung nutzen die Notation der UML. Auch alle relevanten Softwareentwicklungswerkzeuge unterstützen in ihren aktuellen Versionen die UML oder bescheinigen Kompatibilität. Aus diesem Grund ist eine genaue Beschreibung der UML und ihrer Notationselemente hier überflüssig. Interessierte seien auf die Fachliteratur, z.B. [>PS99] [>FS00], [>RJB99],

[>JRHZQ03], und insbesondere auf die ausführlichen Informationen auf der offiziellen Webseite http://www.omg.org/uml/ verwiesen. Der folgende Abschnitt 8.1.1 gibt nur einen kurzen Überblick über die UML 2, als Grundlage für die nachfolgende Diskussion der Stärken und Schwächen der UML in Abschnitt 8.1.2. Dabei legt Abschnitt 8.1.1 einen Schwerpunkt auf die Änderungen gegenüber der UML 1.x.

Die Standardisierung und kontinuierliche Weiterentwicklung sichert der UML eine breite Akzeptanz in Forschung und Industrie, insbesondere auch durch Werkzeughersteller, welche die für eine derartige Sprache notwendigen Editoren, Animations-, Analyse-, Transformations- und Generierungswerkzeuge entwickeln. Abschnitt 8.1.3 führt bedeutende Werkzeuge vergleichend auf.

8.1.1. Übersicht

Klassendiagramme der UML sind heute in jedem Buch über objektorientierte Programmierung zu finden, da Objekte bzw. deren Abstraktion zu Klassen das zentrale Konzept objektorientierter Modellierung sind. Die UML abstrahiert das objektorientierte Grundkonzept Klasse jedoch noch einen Schritt weiter zu dem sogenannten Classifier. Abbildung 24 zeigt alle Classifier der UML 2.0 und deren Hierarchie. Eine Beschreibung der einzelnen Classifier findet man z.B. in [>JRHZQ03].

Viele Eigenschaften von Modellelementen, sind bereits auf der obersten abstrakten Ebene „Classifier“ definiert und stehen damit einheitlich für alle abgeleiteten Arten von Classifiern zur Verfügung. Beispiele für Eigenschaften, die allgemein von Klassen bekannt sind, mit der UML 2.0 aber auch anderen Classifiern zugeordnet werden können, sind

- Zuordnungsbeziehungen (Assoziationen) ,
- Charakteristika (Attribute),
- Verhaltensspezifikationen (Operationen),
- Generalisierungen (Ableitungen) und
- Abstraktionen.

So kann z.B. ein Classifier mit einem anderen assoziiert werden, wie z.B. Verhalten, das sich nach Abbildung 24 in Aktivitäten, Interaktionen und Zustandmaschinen aufteilt, mit Klassen, Interfaces oder Devices.

Konkrete Ausprägungen dieser Classifier können in verschiedenen Diagrammtypen dargestellt werden. Diagramme sind das zentrale Instrument zur Systemmodellierung mit der UML. Sie illustrieren bestimmte Teile oder Aspekte eines Systems durch die Anordnung von Modellierungselementen. **Fehler! Verweisquelle konnte nicht gefunden werden.** zeigt einen Überblick über die Diagramm-Arten in der UML 2.0.

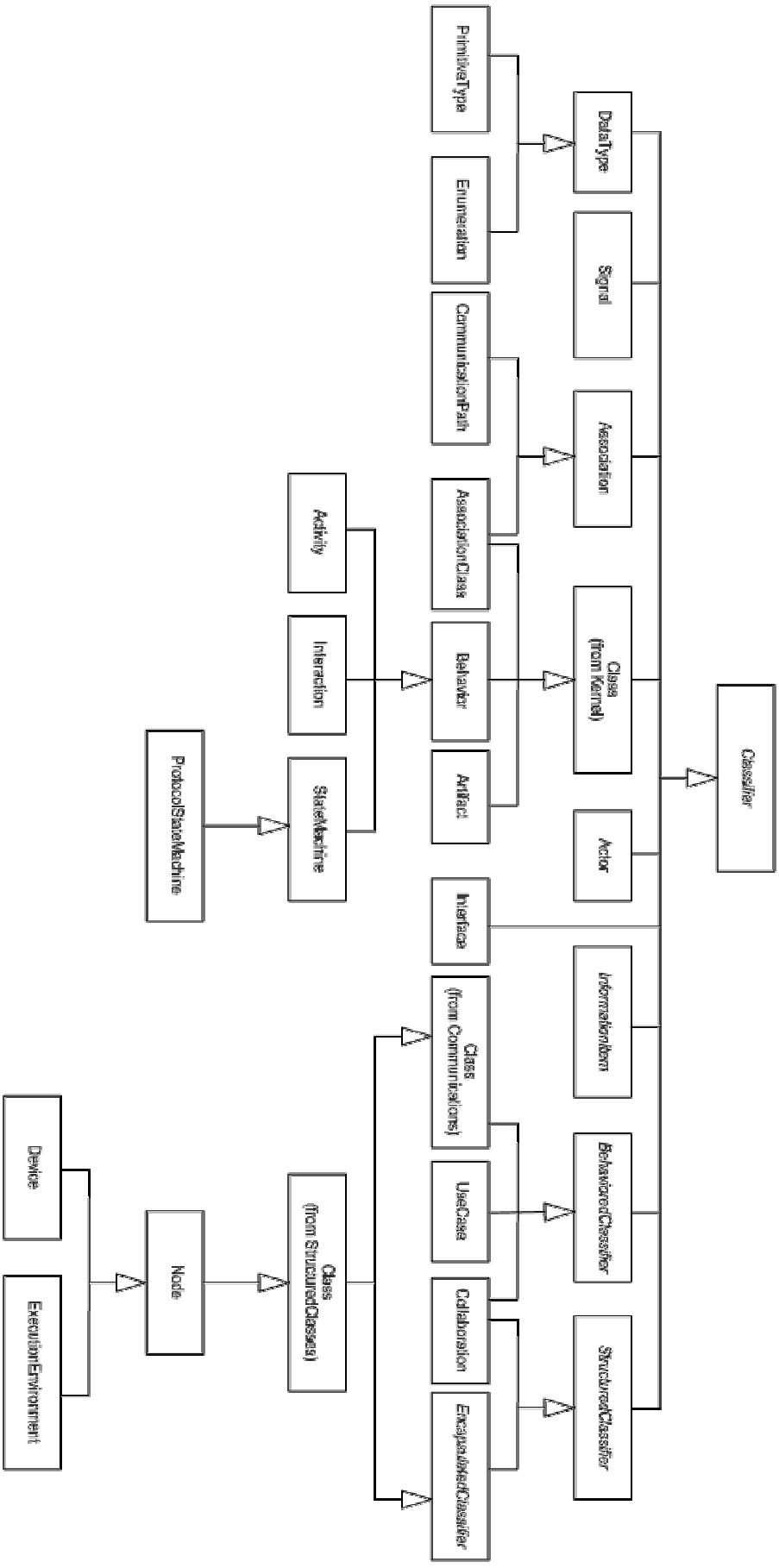

Abbildung 24 Hierarchie der Classifier in der UML 2.0.

UML 2.0 unterscheidet 13 Diagrammtypen, die im Folgenden kurz vorgestellt werden [>JRHZQ03]:

1. **Use-Case-Diagramme** geben einen Überblick über die Funktionalität des zu entwickelnden Softwaresystems in Form von Anwendungsfällen aus der Sicht des Benutzers. Von den Details des Systemverhaltens, z.B. welche Objekte interagieren, wird dabei abstrahiert.

Statische Aspekte des zu entwickelnden Systems werden in Strukturdiagrammen beschrieben.

2. Als wichtigstes und zentrales Diagramm zeigt das **Klassendiagramm** das Softwaresystem in Form von Klassen, Objekten, Schnittstellen, Paketen und deren Beziehungen zueinander. Es enthält alle wichtigen Strukturzusammenhänge und ist normalerweise unverzichtbar. Das Klassendiagramm wurde durch die UML 2 nur unwesentlich geändert.
3. Das **Objektdiagramm** (Instanzdiagramm) modelliert Ausprägungen von Klassen, Assoziationen und Attributen. Es besteht ausschließlich aus Objekten und deren Beziehungen und stellt die innere Struktur eines Systems zu einem bestimmten Zeitpunkt dar. Es kann insbesondere Mengenverhältnisse von Objekten veranschaulichen. Das Objektdiagramm wurde durch die UML 2 nur unwesentlich geändert.
4. **Kompositionsstrukturdiagramme** sind neu in der UML 2 eingeführt und beschreiben das Innenleben von Klassen, Komponenten und Systemteilen sowie deren Interaktionsbeziehungen zu anderen Systemteilen. Sie eignen sich für die „Top-Down“-Modellierung von Systemen, zur Veranschaulichung von Architekturen und Entwurfsmustern.
5. Ein **Komponentendiagramm** verdeutlicht, wie Klassen zu wiederverwendbaren und verwaltbaren Komponenten zusammengefasst werden, und wie diese Komponenten untereinander in Beziehung stehen. Es zeigt die Implementierungsabhängigkeiten von sogenannten Komponenten untereinander. Die kleinen Änderungen zur UML 2.0 betreffen die grafische Darstellung von Komponenten sowie einige Stereotype, wie z.B. die Umbenennung von „*implement*“ in „*manifest*“.

Der Komponentenbegriff in der UML ist sehr implementierungsnah und unterscheidet sich wesentlich von dem weiter gefassten Begriff einer Komponente als wiederverwendbares Teil im Rahmen der komponentenorientierten Softwareentwicklung.

6. Ein **Verteilungsdiagramm** beschreibt das Einsatzumfeld des Systems in Form von Geräten (Hardware) und Ausführungsumgebungen (Software), deren Verbindung (Netzwerk) und den zugeordneten Komponenten, die auf den jeweiligen Knoten ausgeführt werden. Änderungen durch die UML 2 betreffen vor Allem die bessere Unterscheidung von Software und Hardware-Ausführungsumgebungen und die Verwendung von Artefakten.
7. **Paketdiagramme** erlauben eine Übersicht über komplexe Systeme, indem sie Modellelemente logische zu Paketen zusammenfassen und Abhängigkeiten zwischen diesen Paketen modellieren. Als Beispiel können z.B. einzelne Laxer einer Softwarearchitektur in der UML durch Pakete definiert werden. Gegenüber den Versionen 1.4 und 1.5 haben sich in der UML 2.0 keine Änderungen ergeben.

Für die Verhaltensspezifikation stehen in der UML Zustands-, Aktivitäts- und vier Interaktionsdiagrammtypen zur Verfügung. Während Zustands- und Aktivitätsdiagramme Intraobjektverhalten modellieren, beschreiben Interaktionsdiagramme das Interobjektverhalten,

8. **Zustandsdiagramme** (Zustandsautomaten) zeigen das erlaubte Verhalten der Objekte einer Klasse in Form von möglichen Zuständen und Zustandsübergängen, die durch interne oder externe Ereignisse initiiert werden können. Mit der UML 2.0 wurden Ein- und Austrittspunkte sowie sogenannte Terminatoren eingeführt. Außerdem kann das ererbet Verhalten von generalisierten Zustandsautomaten mit einem Zustandsautomaten dargestellt werden. Regeln zur Ergänzung und Ersetzung von Transitionen bei vererbten Zustandsautomaten wurden hinzugefügt.
9. Das **Aktivitätsdiagramm** erlaubt die detaillierte Visualisierung von Abläufen (mit Bedingungen, Schleifen, Verzweigungen), Datenflüssen sowie Parallelisierung und Synchronisation. Es beschreibt das Zusammenspiel einzelner Aktivitäten, d.h. deren Kontroll- und Objektfluss, um eine konkrete Aufgabe oder Algorithmus zu erfüllen. Es wird vor allem als Alternative zu Interaktionsdiagrammen für die Spezifikation von Anwendungsfällen und zur Modellierung von Geschäftsprozessen verwendet. Das Aktivitätsdiagramm wurde am stärksten durch den Übergang zur UML 2.0 verändert Die folgende Liste enthält einige der zahlreiche Änderungen:

- Aktivitätsdiagramme sind keine Sonderform der Zustandsdiagramme mehr.
- Das komplette Diagramm bezeichnet eine Aktivität und die einzelnen Schritte Aktionen.
- Aktionen können mir Vor- und Nachbedingungen verknüpft werden.
- Die Möglichkeit mehrerer Startknoten erlaubt z.B. parallele Abläufe.
- Parallele Abläufe müssen nicht zusammengeführt werden
- Es gibt Endknoten für die gesamte Aktivität und für einzelnen Abläufe innerhalb der Aktivität
- Aktivitätsbereiche können hierarchisch und multidimensional sein.
- Aktivitäten können Ein- und Ausgabeparameter enthalten.
- Neue Notationselemente für Entscheidungen, Schleifen, Mengenverarbeitungen, Parameter, Unterbrechungen, Datenspeicher und –Puffer, und strukturierte Knoten.

10. **Sequenzdiagramme** zeigen den Informationsaustausch zwischen beliebigen Kommunikationspartnern innerhalb eines Systems oder zwischen Systemen generell. Sie beschreiben die notwendigen Interaktionen zwischen Objekten, um eine konkrete Aufgabe (z.B. einen Anwendungsfall) zu erfüllen. Dabei steht die zeitliche Ordnung der Nachrichten, die zwischen den Objekten ausgetauscht werden, im Vordergrund. Auch Sequenzdiagramme wurden in der UML 2.0 wesentlich verändert:
    - Innerhalb von Sequenzdiagrammen kann auf andere Interaktionen verwiesen werden und so Diagramme auf unterschiedlichen Abstraktionsebenen hierarchisch gegliedert werden. Dazu wurden neue Notationselemente eingeführt, wie z.B. Interaktionsreferenzen und Verknüpfungspunkte (Gates) für Nachrichten.
    - Nahezu alle Kontrollflussmöglichkeiten höherer Programmiersprachen sind nun durch kombinierte Fragmente) modellierbar.
    - „Lost“ und „found“-Nachrichten wurden eingeführt für Nachrichten an/von nicht modellierten Empfängern/Sendern.
    - Sprungmarken erlauben eine übersichtliche Gestaltung von Sequenzdiagrammen.
    - Lebenslinien können mit Zustandsinvarianten (Bedingungen für die Interaktion) versehen werden.
11. Das **Kommunikationsdiagramm** entspricht dem Kollaborationsdiagramm aus UML 1.x und stellt die Kommunikation im System dar und zeigt, wer mit wem Informationen austauscht. Dabei steht der Überblick im Vordergrund. Details und zeitliche Abfolgen sind weniger wichtig.
12. Durch das neue **Timingdiagramm** ermöglicht die Modellierung zeitkritischer Anwendungen. Es visualisiert das exakte zeitliche Verhalten von Klassen, Schnittstellen, und Objekten. Es beschreibt die Zustände verschiedener Interaktionspartner zu bestimmten Zeitpunkten.
13. Die ebenfalls neuen **Interaktionsübersichtsdiagramme** beschreiben auf hohem Abstraktionsniveau das zeitliche Verhalten des Systems. Sie zeigen, wann welche Aktionen ablaufen und verbinden verschiedene Interaktionsdiagramme (Sequenz-, Kommunikation- und Timingdiagramme) auf Top-Level-Ebene.

Durch die Anzahl der zur Verfügung stehenden Diagrammarten und die Reichhaltigkeit der darin angebotenen Konzepte entsteht anfangs der Eindruck großer Komplexität der UML. Es ist jedoch zur effektiven Nutzung der UML nicht alle Diagrammtypen nötig. Vielmehr erlaubt die UML, ähnlich einer natürlichen Sprache oder vielen DIN/ISO-Normen, die Konzentration auf eine für die jeweilige Aufgabenstellung hilfreiche Teilmenge der Gesamtsprache. Dadurch kann zum einen die Lernkurve für die UML deutlich verbessert werden. Zum anderen verbessert die Restriktion auf einen begrenzten Sprachkern der UML häufig, etwa innerhalb eines Projekts oder einer Firma, um die Kommunizierbarkeit von Entwürfen zwischen Entwicklern.
Ein solcher Sprachkern besteht beim Einsatz der UML in der Praxis zumeist aus den folgenden Modellierungsnotationen oder sogar nur einer Teilmenge davon [>OES98]:

- Use-Case-Diagramme zur Darstellung des Zwecks des Systems,
- Klassendiagramme zur Strukturdarstellung,
- Zustanddiagramme zur Verhaltensbeschreibung, und
- Sequenzdiagramme zur Darstellung von Interaktion.

Ebenso unterstützen die meisten UML-Werkzeuge (siehe Abschnitt 8.1.3) aufgrund der Komplexität der UML nur eine Teilmenge der UML. Je nach Anwendungsdomäne des Werkzeuges, z.B. Unternehmensmodellierung oder Echtzeit/Embedded-Systeme, werden i.A. unterschiedliche Teilmengen unterstützt.

Dieses allgemein praktizierte Vorgehen der Aufteilung des Sprachumfangs standardisierte das OMG in der UML 2.0 durch die Einführung von Erfüllungsebenen, Erfüllungspunkten und Erfüllungsgrade.

Die UML wird in 3 Erfüllungsebenen (Basic, Intermediate, Complete) bestehend aus jeweils mehreren Erfüllungspunkten aufgeteilt.

Herstelle von Werkzeugen müssen neben den Erfüllungspunkte den Grad der Erfüllung angeben:

1. nicht erfüllt.
2. teilweise erfüllt.
3. vollständig erfüllt.
4. austauschbar (vollständig erfüllt und das Werkzeug unterstützt das Standardaustauschformat XMI).

Die exakte Zuteilung der Notationselementen zu Erfüllungspunkten stand zum Zeitpunkt der Erfüllung dieses STAR noch aus. Bei abgeschlossener Standardisierung könnte im Verlauf des Projektes Dynamite eine Einigung der Projektpartner auf bestimmte Erfüllungsebenen, -grade und -punkte sinnvoll werden.

8.1.2. Vor- und Nachteile

Für den Einsatz als Spezifikations- und Modellierungswerkzeug bietet die UML neben ihrer großen Verbreitung viele Vorteile:

- Die UML unterstützt alle Phasen im Entwicklungsprozess. UML Konstrukte können in der Anforderungsanalyse, Analyse, Design, Implementierung und bei der Testspezifikation eingesetzt werden.
- Die grafische Notation der UML erleichtert den Zugang zu komplexen Systemen. Mit den verschiedenen Diagrammarten unterschiedlicher Abstraktionsebenen können auch komplexe Softwaresysteme leicht übersichtlich dokumentiert werden.
- Die UML ist einfach und wenig formell und dadurch relativ schnell zu erlernen.
- Zahlreiche Entwicklungsprozesse basieren auf der UML und beschreiben den Einsatz der Methoden der UML [>HR02,FOR02,JBR98,KRU99].
- Eine große Anzahl von Fachliteratur beschreibt „Best Practices“, Analyse- und Design-Patterns in der Notation der UML
- Die Unterstützung der UML durch Softwareentwicklungswerkzeuge ist sehr gut (siehe Abschnitt 8.1.3).

Die Hauptschwäche der UML sind mangelnde Möglichkeiten der Verhaltensspezifikation:

- Bei der Modellierung von Aktivitäten erlaubt die UML nicht, oder nur sehr umständlich und unübersichtlich die Spezifikation der Zustandstransformation. Z.B. kann man die Aktivität „Mehrwertsteuer addieren“ mit der UML nur schwer so modellieren, dass sich die tatsächliche Berechnung offenbart, d.h. in diesem Beispiel „Multiplikation mit güterspezifischem Prozentsatz“.
- Die UML kann Abhängigkeiten zwischen bzw. unter Attributen und Objekten schlecht spezifizieren. Z.B. kann man mit der UML die Nebenbedingung, dass Werte von Attributen in verschiedenen Objekten übereinstimmen oder wohldefiniert voneinander abhängen, nur schwer modellieren.

UML bietet große Flexibilität in der Verwendung der Notation. Der dadurch entstehende Verlust an Präzision hat geringere Analysierbarkeit bzw. Simulierbarkeit der Modelle zur Folge. Missverständnisse über die Bedeutung einzelner Diagramme sind Ursache für Fehler.

Formale Spezifikationsmethoden, wie z.B. Z, können die Schwächen der UML ausgleichen und die UML sinnvoll ergänzen [>HV01].

Andere häufig bemängelte Schwächen, wie z.B. fehlende Beziehungen zwischen statischen und dynamischen Diagrammen sind mit der UML 2.0 verbessert worden.

8.1.3. Werkzeuge

Die UML ist eine grafische Modellierungssprache. Aus diesem Grund ist der Einsatz von Werkzeugen unverzichtbar. Modellierungswerkzeuge ermöglichen die einfache Erzeugung von Grafiken, die der Notation der UML. entsprechen. Außerdem überprüfen sie mehr oder weniger die Syntax und können Dokumentationen erstellen. Zusätzlich erlauben immer mehr Werkzeuge die Erzeugung von Quellcode aus UML-Modellen, oder andersherum die Erzeugung von Modellen aus Quellcode.

Der Markt bietet zahlreiche freie und kommerzielle UML-Werkzeuge für unterschiedliche Zwecke. Der Standard XMI soll den Austausch von UML-Modellen zwischen unterschiedlichen Tools ermöglichen. Allerdings wird von vielen unterschiedlichen XMI-Formaten berichtet, die die Interoperabilität erschweren [>TIGRIS03].

Die Tabelle 8 gibt einen Überblick über einige bekannte UML-Werkzeuge. Dabei erstreckt sich die Auswahl von kostenlosen einfachen Programmen bis zu aufwendigen und mächtigen Werkzeugen, wie z.B. den Produkten Rational Rose und Together. Insbesondere Borland und IBM bieten viele Varianten dieser Produkte für unterschiedliche Anwendungsgebiete, Programmierumgebungen und Betriebssysteme an. Verweise zu weiteren UML-Werkzeugen findet man auf http://www.omg.org/uml/.

Produkt	Dia	ArgoUML	Magic Draw	Rational Rose	Together
Hersteller	Alexander Larsson	University of California	No Magic	IBM	Borland
Internet	www.lysator.liu.se/~alla/dia	www.argouml.org	www.magicdraw.com	www.rational.com	www.borland.de/together/index.html
Preis (ca.)	Kostenlos	Kostenlos	1000	10000	5000-9000
Diagramme:					
-Use case	+	+	+	+	+
-Klassen	-	+	+	+	+
-Verteilung	-	+	+	+	+
-Aktivität	-	+	+	+	+
-Kollaboration	-	+	+	+	+
-Sequenz	+	-	+	+	+
-Komponenten	-	-	+	+	+
Code-Generierung	C++, Java (über Dia2code	Java	C++, Java, C#, WSDL	C++, IDL, Java, Ada 83, Ada 95	C++, Java, IDL
Reverse Engineering	-	-	+	+	+
Roundtrip-Engineering	-	-	+	+	+
XMI	-	+	+	+	+
Sonstiges	Kein eigentliches UML-Werkzeug, sondern ein Grafikprogramm.	-	Java-Anwendung für Linux, Windows, und Unix	- Verschiedene Versionen für Unix, Linux, Windows. -Realtime-version	- Java-Anwendung für Linux, Windows, Unix - Viele Editionen - Audits & Metriken - Hohe System-Anforderungen

Tabelle 8 Einige bedeutende UML-Werkzeuge im Vergleich.

Erfüllungsebenen, -punkte und –grade können von den Herstellern heute aufgrund der ausstehenden Standardisierung noch nicht angegeben werden. In naher Zukunft könnte jedoch eine Einigung der Projektpartner auf eine entsprechende Erfüllung der UML zwecks Interoperabilität der eingesetzten Werkzeuge bzw. Austauschbarkeit der erzeugten Modelle sinnvoll werden.

8.1.4. Zusammenfassung und Fazit

UML hat sich als Standard-Modellierungssprache für Softwaresysteme in Industrie und Forschung durchgesetzt. Sie ist unverzichtbar um Softwarearchitekturen und die Kommunikation in Softwaresystemen zu kommunizieren.

Die UML befindet sich momentan im Übergang zur nicht vollständig abwärtskompatiblen Version 2. Dadurch werden sich in der Projektlaufzeit von Dynamite neben der UML selbst auch einzusetzende Tools ändern.

Speziell in der Verhaltensspezifikation bietet die UML nur eingeschränkte Modellierungsmöglichkeiten. Dort können formale Spezifikationssprachen wie Z die UML sinnvoll ergänzen.

Der Standard XMI ermöglicht den Austausch von UML-Modellen zwischen unterschiedlichen Tools. In Projekten können die einzelnen Partner dadurch unterschiedliche, auf Ihre Anforderungen abgestimmte, Tools einsetzen.

Literaturreferenzen zu [Kapitel 8.1]

[>BRJ98] Booch, G., Rumbaugh, J. und Jacobson I.: The Unified Modeling Language User Guide. Addison-Wesley, 1998.

[>DC03] DaimlerChrysler: Automotive-UML, http://www.dc-cc.com/de/forschung/automotive-uml.html, 2003

[>FS00] Martin Fowler und Kendall Scott: UML konzentriert, Addison Wesley, 2000.

[>FOR02] Peter Forbig: Objektorientierte Softwareentwicklung mit UML, Fachbuchverlag Leipzig, 2002

[>HAU01] Frank Haubenschild: Krisenbewältiger - CASE-Tools im Vergleich, Linux-Magazin, 2001.

[>HK02] Martin Hitz und Gerti Kappel, UML@Work - Von der Analyse zur Realisierung, dpunkt.verlag, 2002.

[>HR02] Peter Hruschka, Chris Rupp: Agile Softwareentwicklung für Embedded Real-Time Systems mit der UML, Hanser Fachbuch, 2002

[>HV01] Hvannberg, Ebba Thora, Combining UML and Z in a Software Process, Proceedings of Informatik, GI Jahretagung 2001, Wien Österreich.

[>JBR98] I. Jacobson, G. Booch, J. Rumbaugh: The unified Software Development Process, SEI series in Software Engineering, 1998.

[>JRHZQ03] Mario Jeckle, Chris Rupp, Jürgen Hahn, Barbara Zengler, Stefan Queins: UML 2 glasklar, Hanser, 2003, www.uml-glasklar.de.

[>KRU99] Philippe Kruchten: Der Rational Unified Process, Addison-Wesley, 1999.

[>KRU00] Kruchten, P.: The Rational Unified Process. An Introduction. Second Edition. Addison-Wesley, 2000.

[>OES98] Bernd Oestereich: Objektorientierte Softwareentwicklung – Analyse und Design mit Unified Modeling Language, Oldenbourg, 1998.

[>OMG97] OMG - Object Management Group: www.omg.org.

[>OMG01] OMG - Object Management Group: Unified Modeling Language Specification. V1.4. March 2001.

[>OMG03] OMG: XML Metadata Interchange (XMI) Versionen 1.2. und 2.0, http://www.omg.org/technology/documents/formal/xmi.htm, 2003.

[>PS99] Rob Pooley, Perdita Stevens: Using UML, Addison Wesley, 1999.

[>RJB99] Rumbaugh, J., Jacobson, I. und Booch, G.: The Unified Modeling Language Reference Manual. Addison-Wesley, 1999.

[>RUPP03] Chris Rupp: UML 2 – Ballast oder Befreiung, Vortrag auf den Agility Days in Langen, Oktober 2003, http://www.jeckle.de/uml-glasklar/AgilityDays2003.pdf

[>SELIC99] B. Selic: UML-RT: A profile for modeling complex real-time architectures. ObjectTime Limited, Dezember 1999.

[>TIGRIS03] Tigris.org: http://argouml.tigris.org/faqs/users.html#xmib_linkOMG, 2003.

8.2 Formale Spezifikationsmethoden

8.2.1 Vorüberlegungen

8.2.1.1 Was ist formale Spezifikation?

Aufgabe jeder Spezifikation ist, die *relevanten Eigenschaften* eines Dings *eindeutig* zu beschreiben, ohne die *Implementierung* dieser Eigenschaften festzulegen. Eine Spezifikation *abstrahiert* damit von den nicht-relevanten Eigenschaften des Dings – zu diesen gehört auch die Implementierung der relevanten Eigenschaften.

Die Spezifikation „ein Zylinder 10 cm lang mit einem Durchmesser von 2 cm und einer Längstragfähigkeit von 2000kg“ abstrahiert von der Implementierung (Stahl?) und irrelevanten Eigenschaften (Farbe?).

Eine *formale Spezifikation* verwendet zur Beschreibung der relevanten Eigenschaften eine Sprache mit einer wohldefinierter Syntax, Semantik und Beweisregeln. Dadurch lassen sich rigorose mathematische Verfahren für die Analyse einer solchen Spezifikation einsetzen; weiterhin ist die Formalisierung die Basis für den Einsatz automatisierter Analyse- und Beweismethoden.

8.2.1.2 Wozu formale Spezifikation?

Es gibt eine Vielzahl von Motivationen für den Einsatz formaler Spezifikationsmethode (siehe z. B. [16]). Für DynAMITE stehen jedoch die folgenden Aspekte im Vordergrund:

- Um komplexe Systemeigenschaften eindeutig zu kommunizieren, ohne die Ambiguitäten der natürlichen Sprache, wird ein standardisiertes Vokabular benötigt.
- Um mathematische Beweismethoden einsetzen zu können, mit deren Hilfe sich relevante implizite Eigenschaften des spezifizierten Systems nachweisen lassen.
- Um Einsichten über die Problemdomäne zu gewinnen. Formale Methoden zwingen üblicherweise zu einer eindeutigeren Spezifikation, da sie natürlichsprachlichen Mehrdeutigkeiten weniger Platz lassen. Dies führt regelmäßig zu einem besseren Verständnis der Problemdomäne, da gerade hinter diesen Mehrdeutigkeiten interessante Erkenntnisse über die Natur des Problems (und damit einer effizienten bzw. vollständigen Lösung) verbergen. (Siehe Abschnitt 8.2.3)

Nachteil einer formalen Spezifikation ist der Aufwand, der für die Einarbeitung in die jeweilige Sprache erforderlich ist. Zusätzlich entsteht Aufwand für die Erstellung der Spezifikation selbst, da dieser regelmäßig höher ist, als bei nichtformalen Spezifikationsmethoden. Vor dem Einsatz einer formalen Spezifikationsmethode muss daher immer eine Kosten / Nutzen-Betrachtung stehen. Dabei sollte jedoch bereits der Nutzen eines besseren Verständnisses der Problemdomäne nicht zu gering eingeschätzt werden.

8.2.1.3 Was soll spezifiziert werden?

Spezifikationsmethoden werden unter anderem für die folgenden Aufgabenstellungen eingesetzt, wobei jeweils eine Reihe von Spezifikationsparadigmen zur Verfügung steht:

1. Semantik einer formalen Sprache. Paradigmen:
   - Operationale Semantik.
     Definition einer abstrakten Maschine und die Übersetzung von Sprachkonstrukten in Operationen dieser abstrakten Maschine.
   - Denotationale Semantik (Scott-Strachey-Ansatz).
     Identifikation eines „Wertebereichs“ D für Sprachonstrukte und Angabe einer Übersetzungsfunktion $E : S \longrightarrow D$, mit der die Abbildung von Termen auf diesen Wertebereich festgelegt wird.
   - Axiomatische Semantik (Hoare-Kalkül).
     Definition der Bedeutung von Sprachkonstrukten durch die Angabe von Vor- und Nachbedingungen. ($\{P\}S\{Q\}$: „wenn vor der Ausführung von S die Aussage P wahr ist, gilt nach der Ausführung Q.“)
2. Zustandsraum eines Systems und mögliche Zustandsveränderungen. Paradigmen:
   - Modellbasierter Ansatz (z. B. VDM, Z)
   - Eigenschaftsbasierter Ansatz (z. B. algebraische Spezifikation)
3. Dynamische Entwicklung eines Systems. Paradigmen:
   - Modellbasierter Ansatz (z. B. CSP)

- Eigenschaftsbasierter Ansatz (z. B. temporale Logik)

Für die Anwendung in DynAMITE steht die Spezifikation eines Systems im Vordergrund. Methoden zur Spezifikation einer Sprachsemantik werden deshalb folgend nicht weiter betrachten.

8.2.1.4 Anforderungen an formale Spezifikationsmethoden für DynAMITE

Das Ziel von DynAMITE ist nicht die Entwicklung neuer Spezifikationsmethoden, sondern die Definition einer Middleware für selbstorganisierende Appliance-Ensembles und den Aufbau einer User-Community. Ein „State-of-the-Art“ zum Thema (formale) Spezifikationsmethoden hat daher die Aufgabe, verfügbare Methoden auf Basis der folgenden Kriterien zu betrachten:

- Angemessenheit: Lassen sich die systemrelevanten Fragestellungen mit der Methode beschreiben?
- Reifegrad: Ist die Methode ausgereift? Liegen möglicherweise sogar Standards für ihre Beschreibung vor?
- Akzeptanz: Wird die Spezifikation von den Adressaten verstanden? Gehört die Methode zum typischen Ausbildungsumfang der Informatik? Wird die Methode in der Praxis eingesetzt?
- Werkzeugunterstützung: Stehen Werkzeuge für die Erzeugung und automatische Analyse der Spezifikation zur Verfügung?
- Zugangskosten: Wie aufwändig / kostenintensiv ist es, die notwendige Dokumentation über die Spezifikationsmethode und Zugang zu Unterstützungswerkzeugen zu erhalten?

Im Fall von DynAMITE gehört zur „Angemessenheit“ auch die Nutzbarkeit als Sprache für die mathematische Analyse und die Beschreibung von Lösungskonzepten, so dass derselbe Formalismus sowohl für die Systemspezifikation, wie auch als standardisierte mathematische Sprache für die Konzeptentwicklung eingesetzt werden kann. (Die zum Spezifikationsformalisums gehörigen Verifikationswerkzeuge können dann auch für die Analyse der unterlagerten Konzepte und Beweise eingesetzt werden.)

8.2.1.5 Spezifikationsparadigmen

Wie in Abschnitt 8.2.1.3 dargestellt, lassen sich im wesentlichen zwei große Spezifikationsfragestellungen identifizieren:

- Wie sieht der Systemzustand aus? Welchen Einfluss haben Systemoperationen auf den Systemzustand?
- Welche dynamischen Eigenschaften hat das System? Welche möglichen Lebensläufe des Systems sind zulässig?

Für die Spezifikation des Systemzustands und der darauf möglichen Operationen existieren mindestens zwei grundlegende Ansätze: die *modellbasierte Spezifikation* (teilweise auch als axiomatische Spezifikation bezeichnet) und die *algebraische Spezifikation*.

8.2.1.5.1 Zustand: Modellbasierte Spezifikation. Bei der *modellbasierten Spezifikation* wird zunächst ein abstraktes Zustandsmodell des Systems definiert (zusammen mit den invarianten Zustandseigenschaften) und danach werden die Operationen des Systems in der Form von Vorbedingungen und Nachbedingungen formuliert. Bei dem modellbasierten Ansatz (wie etwa in Z oder VDM) ist die Menge der möglichen Zustände explizit gegeben, während die Menge der möglichen Operationssequenzen implizit vorliegt.

Um etwa ein Dictionary zu beschreiben, das ein Verzeichnis von Namen und Werten darstellt, würde man für die modellbasierte Spezifikation zunächst den abstrakten Systemzustand modellieren:

$$Dictionary == Name \pfun Value$$

und danach die Operationen auf diesem Systemzustand beschreiben:

$$\begin{array}{l} put : Name \times Value \longrightarrow Dictionary \longrightarrow Dictionary \\ get : Name \longrightarrow Dictionary \longrightarrow Value \\ \hline \forall\, d : Dictionary;\ n, n' : Name;\ v, v' : Value \mid n \neq n' \bullet \\ \quad put(n, v)\ d = d \oplus \{n \mapsto v\} \land \\ \quad get\ n\ d = d\ n \end{array}$$

8.2.1.5.2 Zustand: Algebraische Spezifikation. Die *algebraische Spezifikation* (manchmal auch als „eigenschaftsbasierte Spezifikation“ bezeichnet) ist der hierzu duale Ansatz: Hier wird das Systemverhalten durch die Äquivalenz von Operationssequenzen beschrieben. Der Systemzustand (und die *Menge* der möglichen Zustände) wird also implizit, durch seine *Eigenschaften* charakterisiert, nicht durch ein Modell[1]. Ein Verzeichnis von Namen und Werten lässt sich algebraisch etwa in folgender Form beschreiben:

$$
\begin{array}{|l}
put : Name \times Value \longrightarrow Dictionary \longrightarrow Dictionary \\
get : Name \longrightarrow Dictionary \longrightarrow Value \\
\hline
\forall\, d : Dictionary;\ n, n' : Name;\ v, v' : Value \mid n \neq n' \bullet \\
\qquad (get\ n \circ put(n, v))\ d = v \ \land \\
\qquad (get\ n \circ put(n', v))\ d = get\ n\ d
\end{array}
$$

Wobei hier der Typ $Dictionary$ opak ist und nicht weiter detailliert wird. Algebraische Spezifikation ist (wie modellbasierte Spezifikation) als *Paradigma* zu betrachten, das nicht zwingend an eine bestimmte Notation gekoppelt ist. So wurden beide Beispiele in der Sprache Z modelliert, die primär für die modellbasierte Spezifikation vorgesehen ist.

Algebraische Spezifikationen sind aus Sicht eines IT-Ingenieurs (der vor dem Hintergrund der von-Neumann-Architektur ein zustandsbasiertes Denken gewohnt ist) konzeptionell abstrakter als modellbasierte Spezifikationen.

8.2.1.5.3 Dynamik. Für die Spezifikation des *dynamischen Verhaltens* eines Systems steht die Beschreibung der möglichen Lebensläufe eines Systems (ggf. bestehend aus mehreren parallelen Prozessen) im Vordergrund. Hier sind zwei Ansätze zu nennen:

- Communicating Sequential Processes (CSP) [9, 3]. Das dynamische Verhalten von kommunizierenden Prozessen in CSP wird durch die Definition der erlaubten Ereignissequenzen (*traces*) für diese Prozesse spezifiziert. D.h., es wird ein explizites Modell für das Verhalten von Prozessen angegeben. Dagegen ergeben sich Aussagen über bestimmte *Eigenschaften* dieses Verhaltens (etwa Abwesenheit von Deadlocks) nur implizit (z. B. via Beweis).

  Petri-Netze und automatenbasierte Spezifikation von dynamischem Systemverhalten können ebenfalls als modellbasierte Spezifikationsmethoden für Systemdynamik betrachtet werden.

- Temporale Logik (TL). TL erweitert die Prädikatenlogik um zusätzliche temporale Operatoren wie $\Diamond$ („eventually“) und $\Box$ („always“), mit der Bedeutung

  $\Diamond p \Leftrightarrow$ es gibt mindestens einen möglichen zukünftigen Zustand, in dem p gilt

  und

  $\Box p \Leftrightarrow$ in allen möglichen zukünftigen Zuständen gilt p.

  Die Semantik von TL-Ausdrücken wird über Kripke-Strukturen definiert – gerichtete Graphen deren Knoten Zustände und deren Kanten mögliche Zustandsübergänge sind.

  Wie im Fall der algebraischen Spezifikation sind bei der TL-basierten Beschreibungen der Systemdynamik die möglichen Sequenzen von Prozessaktionen implizit gegeben, dagegen sind die Eigenschaften des Verhaltens explizit. Deadlockfreiheit kann beispielsweise einfach gefordert werden ($\Box\Diamond p$, „always eventually something will happen“).

Es ist möglich, die Spezifikation dynamischer Eigenschaften in zustandsorientierte Beschreibungsmethoden zu integrieren (siehe Abschnitt 8.2.2.2).

8.2.1.6 Relevante Methoden

Insgesamt existiert eine sehr große Anzahl von Spezifikationsmethoden, für die unterschiedlichsten Anwendungsfelder[2]. Eine systematische Gegenüberstellung der Methoden und ihrer Anwendungsbereiche ist – zumindest auf Basis oberflächlicher Recherchen – nicht verfügbar; Sammlungen von Methoden und Links finden sich auf verschiedenen Web-Seiten, besonders hervorzuheben sind [2], [7] und [8].

[1] Inkonsistente algebraische Spezifikationen liefern Systeme mit nur einem Zustand, da sich Inkonsistenz durch die Äquivalenz aller Operationssequenzen ausdrückt.

[2] [2] zählt 93 Methoden.

Die im Rahmen dieses STAR durchgeführte informelle und unvollständige Recherche erweckt jedoch den Eindruck, dass VDM [6, 10, 12] und Z [18, 11, 14] die populärsten und am häufigsten genutzten formalen Spezifikationsmethoden sind. Dieser Eindruck wird von [13] bestätigt. Laut [19] ist Z die heute „in der Praxis verbreitetste formale Spezifikationssprache“. Aus diesem Grund werden wir uns folgend auf die Betrachtung von VDM und Z beschränken.

8.2.2 Analyse von VDM und Z

8.2.2.1 Unterschiede zwischen VDM und Z

Für den „normalen Nutzer“ sind VDM („Vienna Definition Method“) und Z bis auf die Unterschiede in der Syntax zunächst recht ähnlich; immerhin basieren beide auf dem Paradigma der modellbasierten Spezifikation. Allerdings scheint die semantische Basis der beiden Methoden unterschiedlich.

Z basiert auf der axiomatischen Mengetheorie (Zermelo-Fraenkel Mengentheorie) und bietet damit den direkten und interoperablen Zugang zu grundlegenden mathematischen Konzepten wie Mengen, Tupel, Relationen, Funktionen, etc.. Insbesondere sind Funktionen Mengen von Paaren, ein Spezialfall der Relation, und stellen damit die Vereinigung der VDM-Kategorien „Function“ und „Map“ dar. [14] legt nahe, dass die Semantik rekursiver Ausdrücke über Tarskis Fixpunkt-Theorem definiert wird (für eine präzise Auskunft hierzu sollte jedoch [11] betrachtet werden).

Die Semantik von VDM basiert auf der Logik Partieller Funktionen (LPF); die Semantik rekursiver Funktionen wird ebenfalls über ein Fixpunkt-Theorem definiert. Vermutlich setzt VDM letztendlich auf Scotts Theorie reflexiver Bereiche auf, die auch für Denotationale Semantik eingesetzt werden; die Ressourcen im Internet und auch [12] machen hier leider keine klare Aussage[3] (der VDM-Standard [10] dürfte präziser sein). Vor diesen Hintergrund wird die in VDM bestehende Trennung der Konzepte „Funktion“ (als etwas, das durch einen – möglicherweise nicht-terminierenden – Berechnungsprozess definiert wird) und „Map“ (eine endliche Menge von Argument / Wert-Paaren) einsichtig.

Beide Sprachen bieten spezifische Mechanismen für die Strukturierung einer Spezifikation. VDM besitzt ein Modulkonzept, Z verfügt mit dem Schema-Kalkül über einen sehr leistungsfähigen Mechanismus für den inkrementellen Aufbau einer Spezifikation.

Insgesamt ist auf der fachlichen Ebene eine Entscheidung für VDM oder Z mehr eine Frage persönlicher Vorlieben als harter inhaltlicher Kriterien. Z spricht eher den Anwender mit mathematisch / mengentheoretischen Hintergrund an („functions as graphs“), während VDM für die Liebhaber des Lambda-Kalküls attraktiver ist („functions as computations“).

8.2.2.2 Angemessenheit

VDM und Z sind beides Spezifikationsformalismen, die besonders für einen *modellbasierten* Ansatz zur Spezifikation des Systemzustands geeignet sind.

Beide Sprachen bieten per se keine Mechanismen für die Beschreibung der Systemdynamik. Allerdings ist es möglich, durch die explizite Modellierung des Lebenslaufs des Systems (in dem die „History“ ein Teil des Systemzustands darstellt) Anforderungen an den Lebenslauf zu modellieren. Für die Sprache Z gibt es beispielsweise Ansätze auf der Basis von temporaler Logik ([4, 5]), auf der Basis von CCS (Calculus of Communicating Systems) sowie Vorschläge für eine Integration mit CSP (TCOZ, CSP-OZ, CSP-Z). CSP und Z sind beide an der Universität Oxford in der von C. A. R. Hoare geleiteten Programming Research Group[4] entstanden. Dadurch besteht eine recht gute Übereinstimmung zwischen den Notationskonventionen, die eine Integration auch optisch leicht macht. Ähnliche Aktivitäten für VDM sind nicht bekannt, allerdings wurden Projekte durchgeführt, in denen VDM und CCS gemeinsam genutzt wurden.

Z bietet darüber hinaus durch seine visuelle Syntax und seine mengentheoretische Basis die eine recht große Nähe zur üblichen Exposition von mathematischen Überlegungen. Z kann daher auch für die mathematischen Analysen und Beweisführungen im Rahmen der Konzeptentwicklung eingesetzt werden, für die dann zumindest eine Überprüfung der statischen Semantik möglich wird.

Die Angemessenheit von VDM ist damit als befriedigend Anzusehen, für Z erscheint sie als gut.

8.2.2.3 Reifegrad und Akzeptanz

Beide Methoden haben einen hohen Reifegrad und besitzen ISO-Standards. Arbeiten an VDM begannen in 1973, die Entwicklung von Z in 1979; hinter beiden Methoden steht daher eine langjährige Entwicklungs- und

[3] Laut [13] setzt VDM ebenfalls auf Mengentheorie und Prädikatenlogik auf. Dies wäre aber nicht mit LPF vereinbar.

[4] Diese Arbeitsgruppe wurde von Christopher Strachey gegründet, der seinerseits gemeinsam mit Dana Scott das Spezifikationskonzept der Denotationalen Semantik entwickelt hat.

Erprobungszeit. VDM und Z sind die populärsten formalen Spezifikationsmethoden in Praxis und Lehre, siehe auch Abschnitt 8.2.1.6.

8.2.2.4 Werkzeugunterstützung

Für die Sprache Z stehen mit den Public-Domain-Tools „fuzz“ [15] und „ZTC“ [17] Type-Checker zur Verfügung, mit deren Hilfe sich die Korrektheit der Syntax und der statischen Semantik einer Z-Spezifikation verifizieren lässt. Auf der Z-Webseite findet sich eine Liste mit ca. 30 Werkzeugen, vom Browser bis hin zum Theorembeweiser.

Für VDM bietet das „SpecBox“-System von Adelard [1] einen Syntax-Checker und einen rudimentären Type-Checker. Einige weitere Werkzeuge werden auf der VDM-Website gelistet.

Für VDM und Z sind LaTeX-Style-Files und TrueType-Fonts verfügbar, mit deren Hilfe eine Spezifikation gesetzt werden kann.

Die Werkzeugunterstützung von Z kann damit als sehr gut betrachtet werden; im Fall von VDM ist sie befriedigend bis gut.

8.2.2.5 Zugangskosten

Für Z sind das Referenzhandbuch [14], verschiedene Type-Checker (fuzz, ZTC) sowie Theorembeweiser etc. frei verfügbar. Im Fall von VDM gibt es ein Lehrbuch [12] und einen rudimentären Typechecker. Für beide Sprachen sind auch Draft-Versionen des ISO-Standards verfügbar.

Durch die gute Werkzeugunterstützung in der Public Domain positioniert sich Z etwas besser als VDM.

8.2.2.6 Interoperabilität mit UML und anderen Methoden

Für die Integration von Z und UML gibt es prototypische Werkzeuge auf der Basis von Rational Rose (public domain: RoZ, kostenpflichtig: RoZeLink), für VDM existiert das VDM++-Rose-Link.

Für beide Methoden existieren verschiedene Versuche zur Integration objektorienterter Konzepte (VDM++; Z++, OOZE, Object-Z). Für Z gibt es Experimente zur Integration mit Haskell (FunZ) und Mathematica. Auch zur VDM/Haskell-Integration gibt es Ansätze.

8.2.2.7 Schlußfolgerung

Z und VDM stellen sich recht gleichmäßig dar; Unterschiede zwischen beiden Methoden sind sowohl im Hinblick auf die Ausdrucksstärke, wie auch in Bezug auf die verfügbare Infrastruktur eher klein. Insgesamt scheint aber der Einsatz von Z aus den folgenden Gründen die bessere Wahl zu sein:

- Nutzbarkeit auch für die mathematische Analyse und Konzeptentwicklung.
- Geringere Zugangskosten durch größere Werkzeugbasis.
- Aktivere Community.

8.2.3 Erkenntnisgewinn durch formale Spezifikation

In Abschnitt wurde die Behauptung aufgestellt, dass formale Spezifikation das Verständnis der Problemdomäne verbessern kann. Dies soll folgend illustriert werden.

Gegeben sei beispielsweise die natürlichsprachliche Spezifikation einer Programmiersprache, speziell die Definition des Funktionsaufrufs, etwa in folgender Form:

> Der Ausdruck „f`(`a_1`,`a_2`,`$\ldots$`,`a_n`)`“ beschreibt einen Funktionsaufruf. Die Werte der Ausdrücke a_i werden an die Funktion f übergeben, der Wert des Gesamtausdrucks ist das Funktionsergebnis.

Hinter dieser hinreichend klar erscheinenden Spezifikation verbergen sich erhebliche Mehrdeutigkeiten sobald die hier beschriebene Programmiersprache Seiteneffekte in Ausdrücken zulässt. Möglicherweise steht an anderer Stelle des Textes:

> Der Ausdruck „a`++`“, wobei a der Name einer Speicherzelle ist, liefert den Wert dieser Speicherzelle und erhöht ihren Inhalt danach um 1.

Im Ausdruck `f(a,a++)` kann dann `f` mit gleichen, aber auch mit unterschiedlichen Werten aufgerufen werden, je nach der Reihenfolge, in der die Funktionsargumente ausgewertet werden. Die Sprachdesigner retten sich in solchen Fällen damit, dass sie die Verwendung von Seiteneffekte in Funktionsargumenten als Programmierfehler deklarieren (vgl. C).

Bei einer *formalen* Spezifikation ist es dagegen *nicht möglich* den Wert eines Ausdrucks zu beschreiben, ohne den Seiteneffekt zu berücksichtigen, etwa durch folgende Spezifikation ausgedrückt

$$[Expr, Value, State]$$

Es gibt Ausdrücke ($Expr$), deren Werte ($Value$) und einen (Speicher-)zustand $State$. Da der Wert eines Ausdrucks vom Speicherzustand abhängt und diesen auch verändern kann, muss die Funktion für die Auswertung von Ausdrücken in folgender Form spezifiziert werden:

$$eval : Expr \longrightarrow State \longrightarrow Value \times State$$

Mit dieser Definition von $eval$ ist der Designer der Sprache jedoch *gezwungen* sich darüber Gedanken zu machen, wie er den sich verändernden Zustand bei der Auswertung einer Argumentliste weiterreicht. Denkbar wäre Auswertung von links nach rechts:

$$\begin{array}{l} argl : \operatorname{seq} Expr \longrightarrow State \longrightarrow \operatorname{seq} Value \times State \\ \hline \forall\, a : Expr;\ al : \operatorname{seq} Expr;\ \sigma : State \bullet \\ \quad argl\, \langle\rangle\, \sigma = (\langle\rangle, \sigma) \land \\ \quad argl\, (\langle a \rangle \frown al)\, \sigma = \\ \qquad (\mu\, v : Value;\ \sigma' : State \mid (v, \sigma') = eval\, a\, \sigma \bullet \\ \qquad\quad (\mu\, vl : \operatorname{seq} Value;\ \sigma'' : State \mid (vl, \sigma'') = argl\, al\, \sigma' \bullet \\ \qquad\qquad (\langle v \rangle \frown vl, \sigma''))) \end{array}$$

oder Auswertung von rechts nach links:

$$\begin{array}{l} argr : \operatorname{seq} Expr \longrightarrow State \longrightarrow \operatorname{seq} Value \times State \\ \hline \forall\, a : Expr;\ al : \operatorname{seq} Expr;\ \sigma : State \bullet \\ \quad argr\, \langle\rangle\, \sigma = (\langle\rangle, \sigma) \land \\ \quad argr\, (\langle a \rangle \frown al)\, \sigma = \\ \qquad (\mu\, vl : \operatorname{seq} Value;\ \sigma' : State \mid (vl, \sigma') = argr\, al\, \sigma \bullet \\ \qquad\quad (\mu\, v : Value;\ \sigma'' : State \mid (v, \sigma'') = eval\, a\, \sigma' \bullet \\ \qquad\qquad (\langle v \rangle \frown vl, \sigma''))) \end{array}$$

die Auswertung in einer *beliebigen* Reihenfolge:

$$\begin{array}{l} argx : \operatorname{seq} Expr \longrightarrow State \longrightarrow \operatorname{seq} Value \times State \\ \hline \forall\, al : \operatorname{seq} Expr;\ \sigma : State \bullet \exists\, order : \operatorname{dom} al \rightarrowtail\!\!\!\!\!\rightarrow \operatorname{dom} al \bullet \\ \quad argx\, al\, \sigma = (\mu\, vl : \operatorname{seq} Value;\ \sigma' : State \mid (vl, \sigma') = argl\, (al \circ order)\, \sigma \bullet (vl \circ order^{\sim}, \sigma')) \end{array}$$

und so weiter. Wie die Argumentlistenauswertung konkret definiert wird, ist tatsächlich gar nicht mehr wirklich von Bedeutung. Wichtig ist, dass die explizite Behandlung von Seiteneffekten in Funktionsaufrufen nicht mehr *übersehen* werden kann. Die formale Spezifikation hat also bei diesem Beispiel zu einem besseren Verständnis der Problemdomäne geführt: zur Feststellung, dass Seiteneffekte in Ausdrücken auch die explizite Festlegung der Auswertungsreihenfolge von Funktions- und Prozedurargumenten erfordern.

Eine Ursache dieses Erkenntnisgewinns ist die referentielle Transparenz mathematisch-fundierter Spezifikationssprachen. Referentielle Transparenz heisst, dass die Bedeutung (der „Wert“) $\mathcal{B}(a)$ eines Ausrucks / Terms a unabhängig vom Kontext (vom umgebenden Spezifikationstext) ist. Daraus folgt, dass a beliebig durch andere Ausrücke a' ersetzt werden kann, solange $\mathcal{B}(a) = \mathcal{B}(a')$. Für (prozedurale) Programmiersprachen ist dies offensichtlich nicht der Fall. Folgendes C-Programm:

```
foo(x) {static y = 0; return x + y++;}

main() {
    auto d = foo(1);
    printf(d == foo(1) ? "true\n" : "false\n");}
```

liefert beispielsweise die Ausgabe `false`. Offensichtlich kann der Ausdruck `foo(1)` im Allgemeinen nicht einfach durch einen Ausdruck mit „gleichem“ Wert – in diesem Fall `d` – ersetzt werden. Dass dies der Fall ist,

kann aber nicht dem Ausdruck `foo(1)`angesehen werden – es ist notwendig, die Funktionsdefinition selbst zu kennen.

Sprachen sind genau dann nicht referentiell transparent, wenn die Bedeutungen von Ausrücken von Zustandswerten abhängig sind, die *nicht* explizit im Ausdruck selbst vorkommen. Es ist dem „Funktionsterm“ `foo(1)` nicht anzusehen, dass sein Wert von einer versteckten (und sich durch Seiteneffekte verändernden) Zustandsvariable (hier: `y`) abhängt.

Mathematische Notationen (und an mathematischen Notationen angelehnte Spezifikationssprachen) sind referentiell transparent; sie bieten keine Möglichkeit einen versteckten Zustand einzubeziehen. Soll der Wert einer mathematischen Funktion von einem Zustand abhängen, so muss dieser Zustandsparameter explizit im Argument und Resultat der Funktion aufgeführt werden[5], etwa in folgender Form:

$$Ystate == \mathbb{Z}$$

$$\begin{array}{|l} foo : \mathbb{Z} \longrightarrow Ystate \longrightarrow \mathbb{Z} \times Ystate \\ y_0 : Ystate \\ \hline \forall\, x : \mathbb{Z} \bullet \\ \quad foo\, x = (\lambda\, y : Ystate \bullet (x + y, y + 1)) \\ y_0 = 0 \end{array}$$

Damit ist explizit gemacht, dass der Wert von $foo\ 1$ von einem weiteren Argument, einem Zustandswert vom Typ $Ystate$ abhängig ist, der auch bei der Definition von $main$ berücksichtigt werden muss:

$$\begin{array}{|l} main : \mathbb{B} \\ \hline main = (\mu\, d : \mathbb{Z};\ y : Ystate \mid (d, y) = foo\ 1\ y_0 \bullet \\ \quad \textbf{if}\ (d, y) = foo\ 1\ y\ \textbf{then}\ True\ \textbf{else}\ False) \end{array}$$

Hier ist es *offensichtlich*, dass $foo\ 1\ y$ im allgemeinen nur dann durch d ersetzen werden kann, wenn gilt $y = y_0$; auch ohne dass die Definition von foo an dieser Stelle bekannt ist.

Referentielle Transparenz ist selbstverständlich Voraussetzung für die Anwendbarkeit vieler Formen der Manipulation von Ausdrücken im Rahmen von mathematischen Beweisverfahren. Noch wichtiger für ein Projekt wie DynAMITE, das die implementierungsunabhängige Beschreibung von Systemen mit neuartigen Funktionen zum Ziel hat, ist aber die Tatsache, dass die referentielle Transparenz von mathematischen Spezifikationssprachen die explizite Analyse von Zustandsabhängigkeiten erzwingt. Damit wird ein genauere Untersuchung und ein besseres Verständnis der Problemdomäne erreicht. Dies ist eine wichtige Voraussetzung für die Entwicklung von Lösungskonzepten, die einen möglichst umfassenden Anwendungsbereich haben.

8.2.4 Die Sprache Z

Dieser Abschnitt enthält eine kondensierte Zusammenfassung des Z-Referenz-Handbuchs [14]. Für den hier gegebenen Überblick erheben wir weder den Anspruch auf vollständige, noch auf korrekte Wiedergabe der Z-Sprachdefinition. Im Zweifelsfall sind die Definitionen in [14, 11] maßgeblich.

Z erlaubt die mathematische, zustandsbasierte Beschreibung von Software- und Hardwaresystemen. Neben den Mechanismen der Prädikatenlogik und der typisierten Mengentheorie wird vor allem das Konzept des *Schemas* und des darüber definierten *Schema-Kalküls* angeboten, das die Definition von Systemzuständen und den möglichen Übergängen zwischen Systemzuständen erleichtert.

Ein Schema (siehe auch Abschnitt 8.2.4.4) definiert eine Menge von strukturierten Objekten, analog zu Tupeln, in denen die Komponentenobjekte über Namen angesprochen werden können[6]. Zusätzlich zu den Namen der Komponenten und der erlaubten Typen enthält ein Schema Beziehungen, die zwischen den Komponenten gelten sollen.

[5] Diese grundsätzlich unterschiedliche Behandlung von Zustandsabhängigkeiten stellt eine der wesentlichen Verständnishürden zwischen (prozeduralen) Programmiersprachen und (mathematischen) Spezifikationssprachen dar. Mathematiker sind eher ungeübt darin, ständig den potentiellen Einfluss eines komplexen aber impliziten Zustands im Kopf zu haben; IT-Ingenieuren dagegen fällt es tendenziell schwer, diesen Einfluss explizit und rigoros zu beschreiben.

[6] Als sehr grobe Vereinfachung kann man ein Schema als Z-Äquivalent zu einem `struct`-Typ in C bzw. einem `record`-Typ in Pascal betrachten.

8.2.4.1 Objekte, Typen und Deklarationen

Eine Z-Spezifikation führt eine Menge mathematischer *Objekte* ein und definiert Beziehungen, die zwischen diesen Objekten gelten müssen.

Jedes Objekt, das durch eine Z-Spezifikation beschrieben wird, besitzt einen *Typ*. Durch eine *Deklaration* werden *Namen* für Objekte mit bestimmten Typen eingeführt. Beispielsweise definiert die Deklaration

$$n : \mathbb{N}$$

ein Objekt mit dem Namen n, das vom Typ der natürlichen Zahlen ist.

8.2.4.2 Namen

Als Namen für Objekte, *Variablennamen*, können alphanumerische Zeichenketten verwendet werden. Innerhalb eines Namens ist der Unterstrich „_“ als Zeichen erlaubt. Ein Name kann mit Sonderzeichen als *Dekoration* versehen werden; einige dieser Sonderzeichen haben per Konvention eine bestimmte Bedeutung. Typische Dekorationen sind Super- und Subskripts, das Zeichen $'$ sowie ! und $?$.

Innerhalb eines Schemas werden Namen die auf $?$ enden auch als „Eingaben“, auf ! endende Namen als „Ausgaben“ bezeichnet.

8.2.4.3 Typdeklarationen

Das Typsystem von Z enthält die folgenden – im weiteren Verlauf näher erläuterten – Konstruktoren:

- Teilmengentyp $T = \mathbb{P}\ T'$,
- Produkttyp $T = T_1 \times T_2$,
- Schematyp $T = \langle n_1 : T_1, \ldots, n_k : T_k \rangle$.

Typen repräsentieren disjunkte Mengen von Objekten, in denen die Objekte jeweils eine bestimmte Struktur besitzen (unstrukturiert, Menge, Tupel oder Schema). In einer Deklaration legt der Typ die Menge von Objekten fest, aus der das vom Namen referenzierte Objekt stammt.

Zwei Typen sind *kompatibel*, wenn sie dieselbe Struktur besitzen.

In einer Deklaration wird der Typ des deklarierten Objektes in Form eines mengenwertigen Ausdrucks definiert, so daß eine Deklaration im allgemeinen die Form

$$name : set$$

besitzt. Hierbei ist set ein Ausdruck vom Typ $\mathbb{P}\ T$ für einen Typ T. Durch eine solche Deklaration werden zwei Aspekte festgelegt:

- Der Typ von $name$, T.
- Eine *Einschränkung* von $name$ auf die im Wert des Ausdrucks set enthaltenen Elemente.

Da 1 vom Typ $\mathbb{Z}$ und somit $\{1\}$ vom Typ $\mathbb{P}\,\mathbb{Z}$ ist, führt die Deklaration $n : \{1\}$ den Namen n mit dem Typ $\mathbb{Z}$ ein, zusammen mit der Einschränkung $n \in \{1\}$ (insgesamt gilt also $n = 1$). Ein weiteres Beispiel ist der Mengenausdruck $\{1\} \times \{2\}$. Er hat den Typ $\mathbb{Z} \times \mathbb{Z}$ und den Wert $\{(1, 2)\}$.

Z bietet eine Reihe von speziellen Operatoren für Mengen, mit denen verschiedene Funktions-, Relations- und Listen„typen“ definiert werden können (eine detaillierte Aufstellung erfolgt weiter unten). Festzuhalten ist dabei, daß z.B. für Mengen A, B die Ausdrücke $A \leftrightarrow B$, $A \longrightarrow B$, $A \rightarrowtail\!\!\!\!\!\rightarrow\ B$ sämtlich vom gleichen Typ $\mathbb{P}(A \times B)$ und damit typkompatibel sind. Die Typkompatibilität zweier Objekte $a : S_a, b : S_b$ wird *allein* anhand der Typkompatibilität der zur Typdefinition verwendeten Mengenausdrücke S_a und S_b entschieden. Die *Werte* der Mengenausdrücke sind hierbei ohne Bedeutung. Der Ausdruck $((\lambda\, x : \{1\} \bullet 2)\, 3)$ ist daher korrekt getypt, sein Wert dagegen ist undefiniert.

Z bietet weiterhin folgende spezielle Mechanismen für die Definition von Typen an:

Gegebene Mengen: Die Notation

$$[ABC]$$

führt die Menge ABC als *gegebene* Menge, als Basistyp, ein, ohne Aussagen über die Art ihrer Elemente zu machen. Der Ausdruck ABC hat als Wert die Menge aller Elemente in ABC und ist daher vom Typ $\mathbb{P}\ ABC$.

Abkürzungen: Die Notation

$$N == E$$

führt den Namen N als Abkürzung für den Ausdruck E ein. Zum Beispiel wird durch die Definition

$$A == \mathbb{N} \longrightarrow \mathbb{Z}$$

der Name A als Abkürzung für die Menge der Funktionen von $\mathbb{N}$ nach $\mathbb{Z}$ eingeführt.

Freie Typen: Es ist möglich, einen Typ durch eine Menge von *Konstanten* und *Konstruktoren* zu definieren. Beispielsweise kann der Typ der binären Bäume in der folgenden Form angegeben werden:

$$\begin{aligned} TREE ::= & \ Leaf \\ | & \ Node \langle\!\langle TREE \times TREE \rangle\!\rangle \end{aligned}$$

$Leaf$ ist eine Konstante, während $Node$ ein zweistelliger Konstruktor ist, der aus zwei $TREE$s einen neuen erzeugt. Dies ist vollständig äquivalent zu der Definition:

$$[TREE]$$

$$\begin{array}{|l} Leaf : TREE \\ Node : TREE \times TREE \rightarrowtail TREE \\ \hline disjoint \langle \{Leaf\}, \operatorname{ran} Node \rangle \\ \forall\, W : \mathbb{P}\ TREE \bullet \\ \quad \{Leaf\} \cup Node(\!| W \times W |\!) \subseteq W \Rightarrow TREE \subseteq W \end{array}$$

Für die Konstruktion von $TREE$ gilt das Erzeugungsprinzip (die kleinste Teilmenge von $TREE$, die alle Konstanten enthält und abgeschlossen unter den Konstruktoren ist, ist $TREE$ selbst). Weiterhin müssen die Typausdrücke in den Konstruktoren *finitär* sein, wenn sie den freien Typ selbst enthalten, damit die implizit definierte rekursive Typgleichung einen Fixpunkt besitzt.

8.2.4.4 Schemas

Eine *Signatur* ist eine Menge von Variablennamen, denen ein Typ zugeordnet ist. Signaturen werden durch Deklarationen erzeugt und bilden das Vokabular für die Notation mathematischer Aussagen, die in der Form von *Prädikaten* ausgedrückt werden. Die Deklaration $z, y : \mathbb{Z}$ erzeugt beispielsweise eine Signatur mit zwei Variablen, x und y, vom Typ $\mathbb{Z}$. In dieser Signatur drückt das Prädikat $x < y$ die Eigenschaft aus, daß der Wert von x kleiner als der von y ist.

Für eine gegebene Signatur ist eine *Situation* eine bestimmte Belegung der Variablen der Signatur mit Werten aus ihren Typen (im o.g. Beispiel ist $x = 3, y = 5$ eine Situation). Der Begriff „Situation“ entspricht dem Struktur- bzw. Interpretationsbegriff der mathematischen Logik. Eine *Eigenschaft* über einer Signatur wird durch die Menge der Situationen charakterisiert, in denen sie wahr ist.

Ein *Schema* ist eine Signatur mit einer Eigenschaft über dieser Signatur. Das o.g. Beispiel kann so durch folgendes Schema beschrieben werden:

$Aleph$

$$\begin{array}{|l} x, y : \mathbb{Z} \\ \hline x < y \end{array}$$

x und y heißen die *Komponenten* von $Aleph$. Der Eigenschaftsteil eines Schemas kann leer sein. In diesem Fall wird $true$, das immer wahre Prädikat, angenommen.

Ein Schema kann auch in *horizontaler Form* angegeben werden:

$$Aleph \mathrel{\widehat=} [x, y : \mathbb{Z} \mid x < y]$$

Die allgemeinen Formen für normale und horizontale Schemadefinition sind:

$$N \;\widehat{=}\; [D \mid P]$$

Hier, wie auch folgend, steht D für eine Menge von Deklarationen, P für eine Menge von Prädikaten und N für einen Schema-Namen. Weiterhin bezeichnet E einen beliebigen Ausdruck.

Eine *Schema-Referenz* ist ein Schema-Name mit einer optionalen Dekoration, wobei diese systematisch auf alle im Schema definierten Variablen angewendet wird. (Beispielsweise besitzt das Schema $Aleph'$ die Variablen x' und y'.)

Schema-Referenzen können für die folgenden Zwecke verwendet werden:

Als Ausdruck. Wird eine Schema-Referenz als Ausdruck verwendet, so bezeichnet dies die Menge aller Objekte mit der Signatur des Schemas, die die Eigenschaft des Schemas erfüllen.

Sei beispielsweise die Deklaration $a : Aleph$, in der $Aleph$ als Ausdruck verwendet wird, gegeben. Dann besitzt die Variable a zwei Komponenten x und y, auf die mit $a.x$ bzw. $a.y$ zugegriffen werden kann und die der Signatur und Eigenschaft von $Aleph$ gehorchen, d.h.: $a.x \in \mathbb{Z}$, $a.y \in \mathbb{Z}$ und $a.x < a.y$.

In diesem Beispiel ist a eine Variable vom *Schema-Typ* $Aleph$. Variablen eines Schema Typs enthalten als Werte *Bindungen*. Eine Bindung ist eine Abbildung der Namen, die als Komponenten in der Deklaration des Schemas definiert wurden, auf Werte aus ihrem Typ. a enthält als Wert eine Bindung, die die Namen x und y auf entsprechende Werte abbildet. Hierbei wird allerdings statt der üblichen Syntax für Funktionsapplikation die „.“-Notation verwendet. Wie natürliche Zahlen oder Listen, so sind auch Bindungen normale Objekte in Z.

Bindungen können auch explizit mit Hilfe des θ-Operators erzeugt werden: für ein Schema S mit Komponenten $c_1, \ldots, c_n$ beschreibt θS ein Bindungsobjekt vom Schema-Typ S genau dann, wenn im aktuellen Gültigkeitsbereich Variablen $c_1, \ldots, c_n$ deklariert sind, deren Typen kompatibel zu den gleichnamigen Komponenten von S sind. Beispiel:

$$\begin{array}{|l} z : Aleph \\ x, y : \mathbb{N} \\ \hline x = 1 \\ y = 2 \\ z = \theta Aleph \end{array}$$

Aus diesen Deklarationen folgt: $(\theta Aleph).x = z.x = 1$ und $(\theta Aleph).y = z.y = 2$.

Insbesondere gilt damit für ein Schema S: $S = \{ S \bullet \theta S \}$.

Als Deklaration. Wird eine Schema-Referenz als Deklaration verwendet, so steht dies – aus Sicht des Gültigkeitsbereichs der Deklaration – für eine textuelle Kopie des entsprechenden Schemas. Beispiel:

$$\forall\, Aleph \bullet P \equiv \forall\, x, y : \mathbb{Z} \mid x > y \bullet P$$

In Bezug auf das *charakteristische Tupel* besteht jedoch ein wesentlicher Unterschied zwischen der Verwendung einer Schema-Referenz und einer textuellen Kopie ihres Deklarationsteils. Siehe hierzu Abschnitt 8.2.4.8.

Als Prädikat. Wird eine Schema-Referenz als Prädikat verwendet, müssen sich alle Komponenten im aktuellen Gültigkeitsbereich befinden. Das Prädikat ist dann der Eigenschaftsteil des referenzierten Schemas.

$$\forall\, x, y, z : \mathbb{Z} \bullet Aleph \equiv \forall\, x, y, z : \mathbb{Z} \bullet x > z$$

Es gibt ein globales, namenloses Schema, dessen Signatur und Eigenschaft sukzessive durch *axiomatische Beschreibungen* erweitert werden kann. Eine axiomatische Beschreibung hat die Form

$$\begin{array}{|l} D \\ \hline P \end{array}$$

8.2.4.5 Das Schema-Kalkül

Schemas können verknüpft werden, um neue Schemas zu bilden.

Schema-Inklusion Durch die Verwendung einer Schema-Referenz im Deklarationsteil einer Schema-Definition ist es möglich, Schemas hierarchisch aufzubauen. Zum Beispiel wird das Schema

$Beth$

$Aleph$
$z : \mathbb{N}$

$z = x + y$

expandiert zu:

$Beth$

$x, y : \mathbb{Z}$
$z : \mathbb{N}$

$x < y$

$z = x + y$

Logische Verknüpfung. Zwei Schemas A und B mit typkompatibler Signatur können mit Hilfe logischer Verknüpfungen kombiniert werden. Das Ergebnis der Verknüpfung ist ein neues Schema C, dessen Signatur die Vereinigung der Signaturen von A und B ist; der Eigenschaftsteil von C ist die logische Verknüpfung der beiden Eigenschaftsteile von A und B. Sei beispielsweise das Schema $Aleph$ wie zuvor definiert und das Schema $Gimel$ als:

$Gimel$

$y : \mathbb{Z}$
$z : 1 \mathinner{..} 10$

$y = z * z$

dann ist $Aleph \land Gimel$ das Schema

$x, y : \mathbb{Z}$
$z : 1 \mathinner{..} 10$

$x < y \land y = z * z$

Für die anderen logischen Konnektive wie etwa $\lor, \Rightarrow, \Leftrightarrow$, gilt ein analoges Vorgehen.

Schema-Quantifikation. Die Quantoren $\forall$ und $\exists$ können auf Schemas angewendet werden. Für ein Schema S und Deklarationen D beschreibt der Ausdruck

$$\forall D \mid P \bullet S$$

ebenfalls ein Schema. Die Signatur des Resultats ist die von S mit Ausnahme der von D eingeführten Variablen. Die Eigenschaft des Resultats bestimmt sich wie folgt: Für jede Situation des Resultats betrachte die möglichen Erweiterungen auf die Signatur von S. Falls *jede* dieser Erweiterungen D, P und die Eigenschaft von S erfüllt, dann erfüllt die ursprüngliche Situation das Schema $\forall D \mid P \bullet S$. Für den Existenzquantor gilt, daß es mindestens *eine* Erweiterung geben muß.

Eine Quantifikation enfernt die in D angegebenen Variablen aus der Signatur des Schemas und führt sie mit einer entsprechenden Quantifikation im Eigenschaftsteil ein. Beispielsweise ist

$$\forall z : \mathbb{Z} \mid \mathbb{Z} > 5 \bullet Gimel$$

das Schema

$$\begin{array}{|l}
y : \mathbb{Z} \\
\hline
\forall\, z : \mathbb{Z} \mid z > 5 \bullet \\
\quad z \in 1 \,..\, 10 \land y = z * z
\end{array}$$

Verstecken von Komponenten. Wenn ein Schema S Komponenten $n_1 : T_1, \ldots, n_k : T_k$ enthält, dann beschreibt der Ausdruck

$$S \setminus (n_1, \ldots, n_k)$$

das Schema $\exists\, n_1 : T_1, \ldots n_k : T_k \bullet S$. Beispielsweise ist $Gimel \setminus (z)$ das Schema

$$\begin{array}{|l}
y : \mathbb{Z} \\
\hline
\exists\, z : 1 \,..\, 10 \bullet y = z * z
\end{array}$$

Sequentielle Komposition. Existieren zwei Schemas S und T, dann ist der Ausdruck $S \fatsemi T$ die sequentielle Komposition von S und T. Damit die sequentielle Komposition definiert ist, müssen die gestrichenen Komponenten von S mit den ungestrichenen Komponenten von T übereinstimmen, d.h., falls S eine Komponente x' besitzt, muß in T eine Komponente x mit demselben Typ existieren. Außerdem müssen die Typen aller gemeinsamen Ein- und Ausgabekomponenten übereinstimmen.

Das Schema $S \fatsemi T$ enthält dann alle ungestrichenen Komponenten aus S, alle gestrichenen Komponenten aus T und alle Ein- und Ausgabekomponenten von S und T. Wenn $State$ ein Schema ist, das genau die ungestrichenen Komponenten von T enthält und damit $State'$ die gestrichenen Komponenten von T, dann ist das Schema $S \fatsemi T$ wie folgt definiert:

$$\begin{array}{l}
\exists\, State'' \bullet \\
\quad (\exists\, State' \bullet [S \mid \theta State' = \theta State'']) \land \\
\quad (\exists\, State \bullet [T \mid \theta State = \theta State''])
\end{array}$$

In dieser Definition ist $State''$ der versteckte Zustand, in dem S terminiert und T beginnt.

8.2.4.6 Generische Konstrukte

Z unterstützt parametrisierten Polymorphismus durch die Möglichkeit, *generische Ausdrücke* anzugeben. Ein generischer Ausdruck besitzt eine Reihe von Typvariablen, die in eckigen Klammern angegeben werden müssen. Wenn der Ausdruck verwendet wird, müssen für die Typ-Parameter konkrete Typen angegeben werden, außer wenn sich diese eindeutig aus der Umgebung ableiten lassen. Beispielsweise führt die *generische* Deklaration

$$\begin{array}{|l}
[X, Y] \\
\hline
\mathit{first} : X \times Y \longrightarrow X \\
\hline
\forall\, x : X; y : Y \bullet \\
\quad \mathit{first}(x, y) = x
\end{array}$$

die generische Funktion first ein, die für beliebige Paare die erste Komponente liefert. Ein vollständiger Aufruf von first lautet zum Beispiel:

$$\mathit{first}[\mathbb{Z}, \mathbb{Z}](3, 4) = 3$$

Da sich aber aus dem Tupel $(3, 4)$ die Typen inferieren lassen, kann kürzer $\mathit{first}(3, 4) = 3$ geschrieben werden.
Auch in Schemadefinitionen und Abkürzungen dürfen Typvariablen verwendet werden.

8.2.4.7 Präfix-, Infix- und Postfix-Symbole

Funktionsapplikationen (und Relationen) werden im allgemeinen in Präfixschreibweise als „$\mathit{func}(\mathit{arg})$“ bzw. „$\mathit{func}\ \mathit{arg}$“ geschrieben. Operationen wie „$+$“ sind ebenfalls Funktionen. Um hier die übliche Infix-Schreibweise zu unterstützen, d.h. „$a + b$“ anstatt „$+(a, b)$“, läßt sich ein Name als Infix(Postfix)-Funktions- bzw. Relationssymbol deklarieren, in dem man die Argumentpositionen in der Deklaration des Namens durch die Markierung „$\_$“ ersetzt. Es ergibt sich dadurch beispielsweise:

$$\begin{array}{|l} \_ + \_ : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{Z} \\ \hline \forall\, a, b : \mathbb{Z} \bullet \\ \quad a + b = \ldots \end{array}$$

Die Zuordnung von Argumenten zu Platzhaltern erfolgt von links nach rechts. Ein Infix(Postfix)-Symbol a kann als Präfix-Symbol verwendet werden, in dem man den Namen mit den Platzhaltern in Klammern angibt, d.h.:

$$\forall\, a, b : \mathbb{Z} \bullet \\ \quad a + b = (\_ + \_)(a, b)$$

8.2.4.8 Charakteristische Tupel

Eine Mengenkomprehension hat die Form

$$\{\, D \mid P \bullet E \,\}$$

und ihr Wert ist die Menge der Werte die E liefert, wenn die in D eingeführten Variablen Werte annehmen, die sowohl die Typbedingungen von D, als auch das Prädikat P erfüllen. Der Teil „$\bullet\ E$“ ist dabei optional, sein Default ist das *charakteristische Tupel* der Deklaration D.

Das charakteristische Tupel einer Deklaration $x_1 : E_1;\ \ldots;\ x_n : E_n$ ist $(x_1, \ldots, x_n)$. Damit ist das charakteristische Tupel für die Deklaration $x, y : \mathbb{N}$ das Tupel (x, y). Tritt in einer Deklaration eine Schema-Referenz S auf, so wird sie im charakteristischen Tupel durch den Ausdruck θS repräsentiert. Damit ist für die Deklaration $x, y : \mathbb{N};\ Aleph$ das charakteristische Tupel $(x, y, \theta Aleph)$. Mit diesem Beispiel wird auch klar, daß eine Variable in mehr als einem Element des charakteristischen Tupels enthalten sein kann: Für alle charakteristischen Tupel $t = (x, y, a)$ zur Deklaration $x, y : \mathbb{N};\ Aleph$ gilt: $x = a.x$ und $y = a.y$, denn in einer Deklaration ist eine Schema-Referenz äquivalent zur Kopie ihres Deklarationsteils.

8.2.4.9 Ausdrücke

$\mathbb{P}\, S, \mathbb{P}_1\, S$ – Potenzmengen.

Für eine Menge S ist $\mathbb{P}\, S$ die Menge aller Teilmengen und $\mathbb{P}_1\, S$ die Menge aller nicht-leeren Teilmengen.

$\{\, D \mid P \bullet E \,\}$ – Mengenkomprehension.

Die Bedeutung einer Mengenkomprehension wurde bereits in Abschnitt 8.2.4.8 erläutert. Hier nur ein kurzes Beispiel:

$$\{\, x : \mathbb{N} \mid x < 4 \bullet x + x \,\} = \{0, 2, 4, 6\}$$

$\lambda\, D \mid P \bullet E$ – Lambda-Ausdruck.

Ein λ-Ausdruck beschreibt eine anonyme Funktion, die Argumente der durch $D \mid P$ beschriebenen Form auf den Wert von E abbildet.

Beispiel: $(\lambda\, x : \mathbb{Z} \bullet x + x)\, 7 = 14$

Die Argumente eines λ-Ausdrucks müssen dabei von der Form der charakteristischen Tupel der Deklaration des λ-Ausdrucks sein. Der λ-Ausdruck $(\lambda\, Aleph \bullet x)$ beschreibt beispielsweise eine einstellige Funktion, die Bindungen vom Schema-Typ $Aleph$ auf die x-Komponente projiziert.

$\mu\, D \mid P \bullet E$ – Explizite Beschreibung.

Der Wert eines μ-Ausdrucks ist nur definiert, wenn es genau eine Belegung der Variablen in D gibt, die P erfüllt. In diesem Fall liefert der Ausdruck den Wert von E, bzw. das charakteristische Tupel von D, falls „$\bullet\ E$“ weggelassen wird.

Beispiel: $(\mu\, x : \mathbb{N} \mid x = 5 \bullet x + x) = 10$

Insbesondere gilt: $(\mu\, x : \{5\} \bullet x + x) = 10$. D.h., für einelementige Mengen $\{e\} \subseteq S$ gilt die Äquivalenz

$$(\mu\, x : \{e\} \bullet E) = (\lambda\, x : S \bullet E)(e)$$

$\mathbf{let}\ x_1 == E_1;\ \ldots x_n == E_n \bullet E$ – Lokale Definition

Liefert den Wert des Ausdrucks E, wobei die Variablen x_i innerhalb von E durch die Werte der Ausrücke E_i ersetzt werden.

Beispiel: $(\mathbf{let}\ x == 5 \bullet x + x) = 10$

Anmerkung: $(\mathbf{let}\ x == V \bullet E) = (\mu\, x : \{\, V \,\} \bullet E)$

$E_f\ E_a$ – Funktionsapplikation.

Im Ausdruck $E_f\ E_a$ muß E_f vom Typ $\mathbb{P}(t_1 \times t_2)$ und E_a vom Typ t_1 sein. Dann ist der Wert des Ausdrucks vom Typ t_2. Falls E_f genau ein Paar enthält, dessen erste Komponente gleich E_a ist, wird die zweite Komponente zurückgegeben, ansonsten ist das Ergebnis undefiniert:

$$f\ x = (\mu\ y : Y \mid (x, y) \in f)$$

$E.i, E.n$ – Tupel- und Schemaselektion.

Für n-Tupel E und $i \leq n$ liefert $E.i$ die i-te Komponente des Tupels.

Beispiel: $(a, b, c, d).3 = c$

Anmerkung: Tupel-Selektion ist im Standard definiert, nicht jedoch Teil des in [14] definierten Sprachumfangs. Der Type-Checker fuzz unterstützt daher Tupel-Selektion nicht.

Für ein Objekt E von einem Schema Typ S, wobei S eine Komponente mit dem Namen n besitzt, liefert $E.n$ den Wert dieses Objektes für die angegebene Komponente.

Beispiel: $(\mu\ g : Gimel \mid g.z = 5 \bullet g.y) = 25$

θ – Schema-Bindung.

Sei S ein Schema und enthalte weiterhin die aktuelle Variablenumgebung alle Komponentennamen, die in der Deklaration von S auftreten. Dann liefert der Ausdruck θS ein Objekt von Schema-Typ S, dessen Komponenten die Werte der Variablen in der Umgebung besitzen, in welcher der θ-Ausdruck ausgewertet wurde.

Beispiel: Der Ausdruck

$$\mu\ y : \{1\}; z : \{2\} \bullet \theta Gimel$$

liefert ein Objekt vom Schema-Typ $Gimel$, auf das mit Komponentenselektion zugegriffen werden kann. Es gilt damit:

$$\begin{aligned} g = (\mu\ y : \{1\}; z : \{2\} \bullet \theta Gimel) \Rightarrow \\ g.y = 1 \land \\ g.z = 2 \end{aligned}$$

Zu beachten ist, daß die Werte für die Komponenten von g der aktuellen Umgebung entnommen werden, ohne zu beachten, ob diese Wertbelegung die Eigenschaft des Schemas erfüllt. (In der hier angegebenen Situation ist dies beispielsweise nicht der Fall.)

$\mathbf{if} - \mathbf{then} - \mathbf{else} -$ – Bedingter Ausdruck.

Der bedingte Ausdruck wird wie folgt als „syntaktischer Zucker“ definiert:

$$\begin{aligned} &\mathbf{if}\ B\ \mathbf{then}\ T\ \mathbf{else}\ F \equiv \\ &\quad (\mu\ x : \{T, F\} \mid B \land x = T \lor \neg\ B \land x = F) \end{aligned}$$

8.2.4.10 Prädikate

Neben Schema-Referenzen enthält Z die Relationen „=“, „∈“, die üblichen Verknüpfungen der Aussagenlogik ($\neg$, $\land$, $\lor$, $\Rightarrow$, $\Leftrightarrow$), die Quantoren der Prädikatenlogik ($\forall, \exists, \exists_1$) sowie das immer wahre Prädikat $true$ und das immer falsche Prädikat $false$. Weiterhin können Relationen in Prädikaten verwendet werden, wobei für ein zweistelliges Relationssymbol R und Ausdrücke E_1, E_2 gilt $(E_1\ R\ E_2) \Leftrightarrow (E_1, E_2) \in (\_R\_)$. Analoges gilt für einstellige Präfix-Relationssymbole.

Ein Prädikat der Form $E_1\ R_1\ E_2\ R_2 \ldots\ R_{n-1}\ E_n$ ist äquivalent zum Ausdruck $E_1\ R_1\ E_2 \land E_2\ R_2\ E_3 \land \ldots \land E_{n-1}\ R_{n-1}\ E_n$.

8.2.4.11 Das mathematische Toolkit

Das mathematische Toolkit von Z definiert über die Basismechanismen von Z hinaus eine Reihe nützlicher Typen und Funktionen.

Mengenoperationen Die Relationen und Operationen auf Mengen (Leere Menge, Teilmengenrelation, Vereinigung, Schnittmenge und Mengendifferenz) sind wie üblich definiert.

$\bigcup, \bigcap$ – Generalisierte Vereinigungs- und Schnittmengen.

$$
\begin{array}{|l}
[X] \\
\hline
\bigcup, \bigcap : \mathbb{P}(\mathbb{P}\, X) \longrightarrow \mathbb{P}\, X \\
\hline
\forall\, A : \mathbb{P}(\mathbb{P}\, X) \bullet \\
\quad \bigcup A = \{\, x : X \mid \exists\, S : A \bullet x \in S \,\} \land \\
\quad \bigcap A = \{\, x : X \mid \forall\, S : A \bullet x \in S \,\} \\
\hline
\end{array}
$$

$first$,$second$ – Zugriffsfunktionen für Paare.

$$
\begin{array}{|l}
[X, Y] \\
\hline
first : X \times Y \longrightarrow X \\
second : X \times Y \longrightarrow Y \\
\hline
\forall\, x : X; y : Y \bullet \\
\quad first(a, b) = a \land \\
\quad second(a, b) = b \\
\hline
\end{array}
$$

$\_ \leftrightarrow \_$ – Relationskonstruktor.

Relationen sind Mengen von geordneten Paaren.

$$X \leftrightarrow Y == \mathbb{P}(X \times Y),$$

d.h., eine Relation zwischen zwei Mengen X und Y ist eine Teilmenge des kartesischen Produktes dieser Mengen. Funktionen sind spezielle Arten von Relationen.

$\mapsto$ – Zuordnungsoperator.

$$
\begin{array}{|l}
[X, Y] \\
\hline
\_ \mapsto \_ : X \times Y \longrightarrow X \times Y \\
\hline
\forall\, x : X; y : Y \bullet \\
\quad x \mapsto y = (x, y) \\
\hline
\end{array}
$$

Der Zuordnungsoperator ist eine andere Schreibweise für ein geordnetes Paar, der dessen Bedeutung als Element einer Funktion hervorhebt.

Insbesondere gilt zum Beispiel: $\{1 \mapsto 2\}\, 1 = 2$

dom,ran – Urbild und Bildbereich von Relationen.

$$
\begin{array}{|l}
[X, Y] \\
\hline
\mathrm{dom} : (X \leftrightarrow Y) \longrightarrow \mathbb{P}\, X \\
\mathrm{ran} : (X \leftrightarrow Y) \longrightarrow \mathbb{P}\, Y \\
\hline
\forall\, R : X \leftrightarrow Y \bullet \\
\quad \mathrm{dom}\, R = \{\, x : X \mid \exists\, y : Y \bullet (x, y) \in R \,\} \land \\
\quad \mathrm{ran}\, R = \{\, y : Y \mid \exists\, x : X \bullet (x, y) \in R \,\} \\
\hline
\end{array}
$$

$\mathrm{id}, \_ \fatsemi \_, \_ \circ \_$ – Identische Relation und Komposition von Relationen.

$$\mathrm{id}\, X == \{ x : X \bullet x \mapsto x \}$$

$$\begin{array}{|l} [X, Y, Z] \\ \hline \_ \fatsemi \_ : (X \leftrightarrow Y) \times (Y \leftrightarrow Z) \longrightarrow (X \leftrightarrow Z) \\ \_ \circ \_ : (Y \leftrightarrow Z) \times (X \leftrightarrow Y) \longrightarrow (X \leftrightarrow Z) \\ \hline \forall R : X \leftrightarrow Y; S : Y \leftrightarrow Z \bullet \\ \quad R \fatsemi S = S \circ R = \\ \quad\quad \{ x : X; y : Y; z : Z \mid \\ \quad\quad\quad (x \mapsto y) \in R \land (y \mapsto z) \in S \bullet x \mapsto z \} \\ \hline \end{array}$$

Für Funktionen $f : X \longrightarrow Y$ und $g : Y \longrightarrow Z$ gilt insbesondere:

$$(f \fatsemi g)\, x = (g \circ f)\, x = g(f(x))$$

$\_ \lhd \_, \_ \rhd \_, \_ \lhd\!\!\!\cdot \_, \_ \rhd\!\!\!\cdot \_$ – Beschränkung von Bild- und Urbildbereich einer Relation.

$$\begin{array}{|l} [X, Y] \\ \hline \_ \lhd \_ : \mathbb{P}\, X \times (X \leftrightarrow Y) \longrightarrow (X \leftrightarrow Y) \\ \_ \rhd \_ : (X \leftrightarrow Y) \times \mathbb{P}\, X \longrightarrow (X \leftrightarrow Y) \\ \_ \lhd\!\!\!\cdot \_ : \mathbb{P}\, X \times (X \leftrightarrow Y) \longrightarrow (X \leftrightarrow Y) \\ \_ \rhd\!\!\!\cdot \_ : (X \leftrightarrow Y) \times \mathbb{P}\, X \longrightarrow (X \leftrightarrow Y) \\ \hline \forall S : \mathbb{P}\, X; T : \mathbb{P}\, Y; R : X \leftrightarrow Y \bullet \\ \quad S \lhd R = \{ r : R \mid \mathit{first}\, r \in S \} \land \\ \quad R \rhd T = \{ r : R \mid \mathit{second}\, r \in T \} \land \\ \quad S \lhd\!\!\!\cdot R = \{ r : R \mid \mathit{first}\, r \notin S \} \land \\ \quad S \lhd\!\!\!\cdot T = \{ r : R \mid \mathit{second}\, r \notin T \} \\ \hline \end{array}$$

Für eine Funktion $f : X \longrightarrow Y$ und Mengen $S : \mathbb{P}\, X$; $T : \mathbb{P}\, Y$ ist:

$(S \lhd f)$ die Funktion f, eingeschränkt auf den Argumentbereich S,

$(f \rhd T)$ die Funktion f, eingeschränkt auf den Bildbereich T,

$(S \lhd\!\!\!\cdot f)$ die Funktion f, eingeschränkt auf die Elemente des Argumentbereichs, die nicht in S liegen,

$(f \rhd\!\!\!\cdot T)$ die Funktion f, eingeschränkt auf die Elemente des Bildbereichs, die nicht in T liegen.

$\_^{\sim}$ – Invertierung einer Relation.

$$\begin{array}{|l} [X] \\ \hline \_^{\sim} : (X \leftrightarrow Y) \longrightarrow (Y \leftrightarrow X) \\ \hline \forall R : X \leftrightarrow Y \bullet \\ \quad R^{\sim} = \{ x : X; y : Y \mid (x \mapsto y) \in R \bullet (y \mapsto x) \} \\ \hline \end{array}$$

$\_(\!|\_|\!)$ – Relationales Bild.

$$\begin{array}{|l} [X, Y] \\ \hline \_(\!|\_|\!) : (X \leftrightarrow Y) \times \mathbb{P}\, X \longrightarrow \mathbb{P}\, Y \\ \hline \forall R : X \leftrightarrow Y; S : \mathbb{P}\, X \bullet \\ \quad R(\!|S|\!) = \{ y : Y \mid \exists x : S \bullet (x \mapsto y) \in R \} \\ \hline \end{array}$$

Für eine Funktion $f : X \longrightarrow Y$ und eine Menge $S : \mathbb{P}\, X$ ist $f(\!|S|\!)$ die Menge aller Elemente aus Y, denen f mindestens ein Element aus S zuordnet.

$_^+, _^*$ – Transitive und reflexiv-transitive Hülle einer Relation.

$$
\begin{array}{|l}
[X] \\
\hline
\_^+, \_^* : (X \leftrightarrow X) \longrightarrow (X \leftrightarrow X) \\
\hline
\forall R : X \leftrightarrow X \bullet \\
\quad R^+ = \bigcap \{ Q : X \leftrightarrow X \mid R \subseteq Q \land Q \fatsemi Q \subseteq Q \} \land \\
\quad R^* = \bigcap \{ Q : X \leftrightarrow X \mid \mathrm{id}\, X \subseteq Q \land R \subseteq Q \land Q \fatsemi Q \subseteq Q \} \\
\hline
\end{array}
$$

Für eine Relation R ist R^+ die kleinste Relation, die R enthält und transitiv ist. R^* ist die kleinste Relation die R enthält und sowohl transitiv als auch reflexiv ist.

$\_ \pfun \_, \_ \fun \_$ – Partielle und totale Funktionen.

Partielle Funktionen sind linkseindeutige Relationen. Formal:

$$
\begin{array}{l}
X \pfun Y == \{ f : X \leftrightarrow Y \mid \\
\quad (\forall x : X; y_1, y_2 : Y \bullet \\
\qquad (x \mapsto y_1) \in f \land (x \mapsto y_2) \in f \Rightarrow y_1 = y_2) \}
\end{array}
$$

Totale Funktionen sind linkstotale partielle Funktionen. Formal:

$$X \fun Y == \{ f : X \pfun Y \mid \mathrm{dom}\, f = X \}$$

$\oplus$ – Funktionsüberlagerung.

$$
\begin{array}{|l}
[X, Y] \\
\hline
\_ \oplus \_ : (X \pfun Y) \times (X \pfun Y) \longrightarrow (X \pfun Y) \\
\hline
\forall f, g : X \pfun Y; x : X \bullet \\
\quad (f \oplus g) x = \mathbf{if}\ x \in \mathrm{dom}\, g\ \mathbf{then}\ g\, x\ \mathbf{else}\ f\, x \\
\hline
\end{array}
$$

Relationale Operationen auf Funktionen: Funktionen sind lediglich spezielle Relationen, so daß alle für Relationen verfügbare Operationen wie $\_ \fatsemi \_$ und $\_ \lhd \_$ auch auf Funktionen anwendbar sind.

$\mathbb{Z}, \mathbb{N}, \mathbb{N}_1$ – Zahlen.

$\mathbb{Z}$ ist die (gegebene) Menge der ganzen Zahlen, für die die üblichen Operationen zugelassen werden:

$$
\begin{array}{|l}
[\mathbb{Z}] \\
\\
\hline
\_ + \_, \_ - \_, \_ * \_ : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{Z} \\
\_ \,\mathsf{div}\, \_, \_ \,\mathsf{mod}\, \_ : \mathbb{Z} \times (\mathbb{Z} \setminus \{0\}) \longrightarrow \mathbb{Z} \\
- : \mathbb{Z} \longrightarrow \mathbb{Z} \\
\_ < \_, \_ \leq \_, \_ \geq \_, \_ > \_ : \mathbb{Z} \leftrightarrow \mathbb{Z} \\
\mathbb{N} : \mathbb{P}\, \mathbb{Z} \\
\mathbb{N}_1 : \mathbb{P}\, \mathbb{Z} \\
succ : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\ldots \\
\mathbb{N} = \{ n : \mathbb{Z} \mid n \geq 0 \} \\
\mathbb{N}_1 = \mathbb{N} \setminus \{0\} \\
\forall n : \mathbb{N} \bullet succ\, n = n + 1 \\
\hline
\end{array}
$$

R^k – Iteration.

$$[X]$$
$$iter : \mathbb{Z} \longrightarrow (X \leftrightarrow X) \longrightarrow (X \leftrightarrow X)$$
$$\begin{aligned}&\forall R : X \leftrightarrow X; n : \mathbb{N}_1 \bullet \\ &\quad iter\, 0\, R = \mathrm{id}\, X \land \\ &\quad iter\, n\, R = (iter\, (n-1)\, R) \fatsemi R \land \\ &\quad iter\, (-n)\, R = iter\, n\, R^{\sim}\end{aligned}$$

Statt $iter\, k\, R$ schreibt man üblicherweise R^k.

$\_ \mathinner{..} \_$ – Zahlenintervall.

$$\_ \mathinner{..} \_ : \mathbb{Z} \times \mathbb{Z} \longrightarrow \mathbb{Z}$$
$$\begin{aligned}&\forall, a, b : \mathbb{Z} \bullet \\ &\quad a \mathinner{..} b = \{\, n : \mathbb{Z} \mid a \leq n \leq b \,\}\end{aligned}$$

Für $a, b : \mathbb{Z}$ beschreibt der Ausdruck $a \mathinner{..} b$ die Menge der ganzen Zahlen zwischen a und b einschließlich.

$\mathbb{F}, \mathbb{F}_1, \#$ – Endliche Mengen, Mächtigkeit von Mengen.

$$\begin{aligned}&\mathbb{F}\, X == \{\, S : \mathbb{P}\, X \mid \exists\, m : \mathbb{N} \bullet \exists f : 1 \mathinner{..} n \longrightarrow S \bullet \mathrm{ran}\, f = S \,\} \\ &\mathbb{F}_1\, X == \mathbb{F}\, X \setminus \{\varnothing\}\end{aligned}$$

$$[X]$$
$$\#\, : \mathbb{F}\, X \longrightarrow \mathbb{N}$$
$$\begin{aligned}&\forall S : \mathbb{F}\, X \bullet \\ &\quad \#S = (\mu\, n : \mathbb{N} \mid (\exists f : 1 \mathinner{..} n \longrightarrow S \bullet \mathrm{ran}\, f = S))\end{aligned}$$

Für eine Menge X ist $\mathbb{F}\, X$ die Menge aller endlichen Teilmengen von X, $\mathbb{F}_1\, X$ die Menge aller nicht-leeren endlichen Teilmengen und $\#X$ die Mächtigkeit von X.

$\_ \pfun \_, \_ \pinj \_$ – Endliche partielle Funktionen und Injektionen.

$$\begin{aligned}&X \pfun Y == \{\, f : X \longrightarrow Y \mid \mathrm{dom}\, f \in \mathbb{F}\, X \,\} \\ &X \pinj Y == (X \pfun Y) \cap (X \rightarrowtail Y)\end{aligned}$$

$X \pfun Y$ ist die Menge der endlichen partiellen Funktionen von X nach Y (d.h., der Urbildbereich ist eine endliche Menge). $X \pinj Y$ ist die Menge der endlichen partiellen Funktionen von X nach Y, die auch Injektionen sind.

min, max – Minimum und Maximum von Zahlenmengen.

$$min, max : \mathbb{P}_1\, \mathbb{Z} \pfun \mathbb{Z}$$
$$\begin{aligned}&\forall S : \mathbb{P}_1\, \mathbb{Z} \bullet \\ &\quad min\, S = (\mu\, n : S \mid \forall\, n' : S \bullet n \leq n') \land \\ &\quad max\, S = (\mu\, n : S \mid \forall\, n' : S \bullet n \geq n')\end{aligned}$$

$\mathrm{seq}, \mathrm{seq}_1, \mathrm{iseq}$ – Listen.

$$\begin{aligned}&\mathrm{seq}\, X == \{\, f : \mathbb{N} \pfun X \mid \mathrm{dom}\, f = 1 \mathinner{..} \#f \,\} \\ &\mathrm{seq}_1\, X == \{\, f : \mathrm{seq}\, X \mid \#f > 0 \,\} \\ &\mathrm{iseq}\, X == \mathrm{seq}\, X \cap (\mathbb{N} \pinj X)\end{aligned}$$

Listen sind Funktionen, die ein Intervall der natürlichen Zahlen auf eine Elementmenge abbilden. seq_1 sind nicht-leere Listen, iseq beschreibt Listen, deren Elemente paarweise verschieden sind.

Für Listen gibt es eine wie folgt definierte Kurzschreibweise:

$$\langle x_1, x_2, \ldots, x_n \rangle = \{1 \mapsto x_1, 2 \mapsto x_2, \ldots, n \mapsto x_n\}$$

$\_ \frown \_$ – Konkatenation von Listen.

$$[X]$$
$$\_ \frown \_ : \operatorname{seq} X \times \operatorname{seq} X \longrightarrow \operatorname{seq} X$$
$$\forall s, t : \operatorname{seq} X \bullet$$
$$s \frown t = s \cup \{i : \operatorname{dom} t \bullet (i + \#s) \mapsto t\, i\}$$

Für Listen s und t ist $s \frown t$ die Liste bestehend aus den Elementen von s, gefolgt von den Elementen von t.

$head, tail, last, front$ – Zugriffsoperationen für Listen.

Es gilt:

$$[X]$$
$$head, last : \operatorname{seq}_1 X \longrightarrow X$$
$$tail, front : \operatorname{seq}_1 X \longrightarrow \operatorname{seq} X$$
$$\forall x : X; s : \operatorname{seq} X \bullet$$
$$head(\langle x \rangle \frown s) = x \land$$
$$tail(\langle x \rangle \frown s) = s \land$$
$$last(s \frown \langle x \rangle) = x \land$$
$$front(s \frown \langle x \rangle) = s$$

rev – Listenumkehrung.

Die Funktion rev ist definiert als:

$$[X]$$
$$rev : \operatorname{seq} X \longrightarrow \operatorname{seq} X$$
$$\forall x : X, s : \operatorname{seq} X \bullet$$
$$rev\, \langle\rangle = \langle\rangle \land$$
$$rev\, (\langle x \rangle \frown s) = (rev\, s) \frown \langle x \rangle$$

$\_ \upharpoonright \_$ – Listenfilterung.

$$[X]$$
$$\_ \upharpoonright \_ : \operatorname{seq} X \times \mathbb{P}\, X \longrightarrow \operatorname{seq} X$$
$$\forall V : \mathbb{P}\, X; s : \operatorname{seq} X; x, y : X \mid x \in V \land y \notin V \bullet$$
$$\langle\rangle \upharpoonright V = \langle\rangle \land$$
$$(\langle x \rangle \frown s) \upharpoonright V = \langle x \rangle \frown (s \upharpoonright V) \land$$
$$(\langle y \rangle \frown s) \upharpoonright V = s \upharpoonright V$$

Relationale Operationen auf Listen. Da Listen spezielle Funktionen (und diese wiederum spezielle Relationen) sind, können die auf Relationen definierten Operationen auch auf Listen angewendet werden. Zum Beispiel ist für $s : \operatorname{seq} X; f : X \longrightarrow Y$ die Komposition $f \circ s$ vom Typ $\operatorname{seq} Y$. Der Effekt dieser Komposition läßt sich wie folgt beschreiben:

$$f \circ \langle\rangle = \langle\rangle$$
$$f \circ (\langle x \rangle \frown s) = \langle f\, x \rangle \frown (f \circ s)$$

Weiterhin liefert $\operatorname{ran} s$ für $s : \operatorname{seq} X$ die Menge der Listenelemente von s.

disjoint $\_, \_$ partition $\_$ – Elementfremdheit von Mengen, Partitionierung von Mengen.

$[I, X]$

$\mathsf{disjoint}\ \_ : \mathbb{P}(I \pfun \mathbb{P}\, X)$
$\_\ \mathsf{partition}\ \_ : (I \pfun \mathbb{P}\, X) \rel \mathbb{P}\, X$

$\forall\, S : I \pfun \mathbb{P}\, X;\ T : \mathbb{P}\, X \bullet$
$\quad (\mathsf{disjoint}\ S \Leftrightarrow$
$\quad\quad \forall\, i, j : \mathrm{dom}\, S \mid (S\, i) \cap (S\, j) \neq \varnothing \bullet i = j) \land$
$\quad (S\ \mathsf{partition}\ T \Leftrightarrow$
$\quad\quad \mathsf{disjoint}\ S \land \bigcup(\mathrm{ran}\, S) = T)$

Eine indizierte Familie von Mengen S (z.B. eine Liste von Mengen) ist elementfremd, wenn die Mitglieder von S paarweise elementfremd sind. Eine indizierte Mengenfamilie S ist eine Partitionierung einer Menge T, falls S elementfremd ist und die Vereinigung aller Mitglieder T ist.

$\mathrm{bag}, \mathit{count}, \_ \sqsubseteq \_$ – Multimengen.

Multimengen sind Mengen, in denen Elemente mehrfach auftreten können. In Z werden Multimengen als Funktionen beschreiben, die Elemente auf Häufigkeiten abbilden.

$$\mathrm{bag}\, X == X \pfun \mathbb{N}_1$$

$[X]$

$\mathit{count} : \mathrm{bag}\, X \surj (X \fun \mathbb{N})$
$\_ \sqsubseteq \_ : X \rel \mathrm{bag}\, X$

$\forall\, x : X;\ b : \mathrm{bag}\, X \bullet$
$\quad \mathit{count}\, B = (\lambda\, x : X \bullet 0) \oplus B \land$
$\quad (x \in B \Leftrightarrow x \in \mathrm{dom}\, B)$

count liefert für ein Element x und eine Multimenge B die Häufigkeitszahl, bzw. 0, falls x nicht in B ist.

Auch für Multimengenextensionen gibt es eine Kurzschreibweise:

$$[\![b_1, \ldots, b_n]\!] = \{x_1 \mapsto k_1, \ldots, x_m \mapsto k_m\}$$

Wobei für jedes $i \in 1 \mathinner{..} m$ das Element x_i genau k_i-mal in der Liste $b_1, \ldots, b_n$ auftritt.

$\_ \uplus \_$ – Multimengenvereinigung.

$[X]$

$\_ \uplus \_ : \mathrm{bag}\, X \times \mathrm{bag}\, X \fun \mathrm{bag}\, X$

$\forall\, A, B : \mathrm{bag}\, X;\ x : X \bullet$
$\quad \mathit{count}\, (A \uplus B)\, x = \mathit{count}\, A\, x + \mathit{count}\, B\, x$

items – Listenelemente als Multimenge.

$[X]$

$\mathit{items} : \mathrm{seq}\, X \fun \mathrm{bag}\, X$

$\forall\, s : \mathrm{seq}\, X;\ x : X \bullet$
$\quad \mathit{count}\, (\mathit{items}\, s)\, x = \#\{\, i : \mathrm{dom}\, s \mid s\, i = x \,\}$

8.2.4.12 Sequentielle Systeme

Sequentielle Systeme können auf der Basis von Schemas mit Hilfe der Schema-Komposition beschrieben werden. Als Beispiel soll ein Zähler definiert werden:

Counter

$\mathit{value} : \mathbb{N}$

Das Schema $Counter$ beschreibt alle möglichen Zustände eines Zählers.

$$
\begin{array}{|l}
\hline
\textit{InitCounter} \\
Counter \\
\hline
value = 0 \\
\hline
\end{array}
$$

$InitCounter$ definiert den Anfangszustand des Zählers. Weiterhin werden Schemas eingeführt, die verschiedene mögliche Zählerzustände miteinander in Beziehung setzen, d.h., die erlaubten Operationen eines Zählers beschreiben:

$$
\begin{array}{|l}
\hline
Inc \\
Counter \\
Counter' \\
\hline
value' = value + 1 \\
\hline
\end{array}
$$

Inc definiert das Erhöhen des Zählers um eins. Die ungestrichenen Komponenten repräsentieren den Zustand vor, die gestrichenen Komponenten den Zustand nach der Operation.

$$
\begin{array}{|l}
\hline
Add \\
Counter \\
Counter' \\
jump? : \mathbb{N} \\
val! : \mathbb{N} \\
\hline
value' = value + jump? \\
val! = value' \\
\hline
\end{array}
$$

Add beschreibt das Erhöhen des Zählers um einen vorgegebenen Wert $jump?$. In diesem Fall besitzt die Operation eine Eingabe $jump?$ und eine Ausgabe $val!$, der neue Zählerstand. Wie bereits erwähnt, werden Eingaben für Operationen per Konvention mit $?$, Ausgaben mit ! gekennzeichnet.

Es können dann Funktionen definiert werden, die Operationen auf Zählern ausführen:

$$
\begin{array}{|l}
new : Counter \\
inc : Counter \longrightarrow Counter \\
add : Counter \times \mathbb{N} \longrightarrow Counter \times \mathbb{N} \\
\hline
new = \theta InitCounter \\
inc = \lambda\, Counter \bullet (\mu\, Counter' \mid Inc) \\
add = \lambda\, Counter; jump? : \mathbb{N} \bullet (\mu\, Counter'; val! : \mathbb{N} \mid Add)
\end{array}
$$

Weiterhin lassen sich Zustandsübergänge mit $\fatsemi$ zusammensetzen. So bedeutet $Inc \fatsemi Add$:

$$
\begin{array}{|l}
\hline
Counter \\
Counter' \\
\hline
value' = value + 1 + jump? \\
val! = value' \\
\hline
\end{array}
$$

$Add \fatsemi Inc$ hat dagegen die folgende Bedeutung:

$$
\begin{array}{|l}
\hline
Counter \\
Counter' \\
\hline
value' = value + jump? + 1 \\
val! = value + jump? \\
\hline
\end{array}
$$

Operationen auf Objekten, die durch Schemas beschrieben werden, zeichnen sich dadurch aus, daß sie im Deklarationsteil jeweils eine ungestrichene und eine gestrichene Kopie des Schemas enthalten, das den Zustandsraum der Objekte beschreibt. Um die Definition sequentieller Systeme zu erleichtern, wird daher implizit mit jedem Schema $State$ ein Schema $\Delta State$ und ein Schema $\Xi State$ wie folgt deklariert:

$\Delta State$

$State$
$State'$

$\Xi State$

$\Delta State$

$\theta State = \theta State'$

Um eine Operation auf $State$ zu beschreiben, muß dann lediglich $\Delta State$ (bzw. $\Xi State$) im Deklarationsteil eingetragen werden. $\Xi State$ ist dabei für die Beschreibung von Operationen nützlich, die nur lesend auf $State$ zugreifen.

Literaturreferenzen zu [Kapitel 8.2]

[1] Adelard LLP. Free Version of SpecBox [online, cited Dec. 2003]. Available from: http://www.adelard.co.uk/software/specbox/index.htm.

[2] Jonathan Bowen. The World Wide Web Virtual Library: Formal Methods [online, cited Dec. 2003]. Available from: http://www.afm.lsbu.ac.uk/.

[3] Jonathan Bowen. Virtual Library formal methods: CSP [online, cited Dec. 2003]. Available from: http://www.afm.sbu.ac.uk/csp/.

[4] Roger Duke and Graeme Smith. Temporal Logic and Z Specifications. *Australian Computer Journal*, 21(2):62–66, May 1989.

[5] Andy S. Evans. An improved recipe for specifying reactive systems in Z. In Jonathan P. Bowen, Michael G. Hinchey, and David Till, editors, *ZUM'97: The ZFormal Specification Notation*, volume 1212 of *Lecture Notes in Computer Science*, pages 275–294. Springer-Verlag, 1997. Available from: citeseer.nj.nec.com/evans97improved.html.

[6] John Fitzgerald. VDM Information [online, cited Dec. 2003]. Available from: http://www.csr.ncl.ac.uk/vdm/.

[7] Formal Methods Europe. Homepage [online, cited Dec. 2003]. Available from: http://www.fmeurope.org/.

[8] Fraunhofer IESE (ed.). Virtuelles Software Engineering Kompetenzzentrum (ViSEK) [online, cited Dec. 2003]. Available from: http://www.visek.de/.

[9] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International, 1985. Available from: http://www.usingcsp.com/cspbook.pdf.

[10] ISO/IEC. Information technology – Programming languages, their environments and system software interfaces – Vienna Development Method – Specification Language – Part 1: Base language. International Standard ISO/IEC 13817-1, International Organization for Standardization (ISO), 1996.

[11] ISO/IEC/JTC1/SC22. Information technology – Z formal specification notation – Syntax, type system and semantics. International Standard ISO/IEC 13568:2002, International Organization for Standardization (ISO), 2002.

[12] Cliff B. Jones. *Systematic Software Development using VDM*. Prentice-Hall, Upper Saddle River, NJ 07458, USA, 1990. Available from: citeseer.nj.nec.com/jones95systematic.html.

[13] Thomas McGibbon. An Analysis of Two Formal Methods: VDM and Z. Technical Report DACS-CRTA-97-1, ITT Industries, System Division, 1997.

[14] J. Michael Spivey. *The Z Notation: a reference manual*. Prentice Hall, 2nd edition, 1992. Available from: http://spivey.oriel.ox.ac.uk/~mike/zrm/ [cited Dec. 2003].

[15] J. Michael Spivey. The fuzz type-checker for Z, 2000. Available from: http://spivey.oriel.ox.ac.uk/mike/fuzz/ [cited Dec. 2003].

[16] Axel van Lamsweerde. Formal Specification: a Roadmap. In *Future of Software Egnineering*, Limerick, England, 2000. ACM. ISBM 1-58113-253-0.

[17] Xioaping Jia. Z Type Checker [online]. 1998 [cited Dec. 2003]. Available from: http://se.cs.depaul.edu/fm/ztc.html.
[18] Z Users Group. The World Wide Web Virtual Library: The Z notation [online, cited Dec. 2003]. Available from: http://www.zuser.org/z/.
[19] Andreas Zeller. *Softwaretechnik II (Vorlesung)*. Universität des Saarlandes, 2002. Available from: http://www.st.cs.uni-sb.de/edu/se2/ [cited Dec. 2003].

9. Relevante Standardisierungsgremien

- ANSI (American National Standards Institute)
  (www.ansi.org/)
- BSIG (Bluetooth Special Interest Group)
  (https://www.bluetooth.org/)
- CELF (CE Linux Forum)
  (www.celinuxforum.org)
- CABA (Continental Automated Building Association)
  (www.caba.org)
- (CECED) European Committee of Manufacturers of Domestic Equipment
  (www.ceced.org)
- DHWG (Digital Home Working Group)
  (www.dhwg.org)
- DVB (Digital Video Broadcasting Project
  (www.dvb.org)
- ECMA (Standardizing Information and Communication Systems)
  (www.ecma-international.org/)
- EIBA (European Installation Bus Association)
  (www.eiba.org)
- ETSI (European Telecommunication Standards Institute)
  (www.etsi.org)
- FIPA (Foundation for Intelligent Physical Agents)
  (www.fipa.org)
- HAVi (Home Audio Video Interoperability, Inc.)
  (www.havi.org)
- HomePlug (Home Plug Powerline Alliance)
  (www.homeplug.com)
- HomePNA (Home Phoneline Networking Alliance)
  (www.homepna.org)
- IEC (International Electrotechnical Commission)
  (www.iec.ch/)
- IEEE (Institute of Electrical and Electronics Engineers)
  (www.ieee.org/)
- IETF (Internet Engineering Task Force)
  (www.ietf.cnri.reston.va.us/)
- INCITS (InterNational Committee for Informations Technology Standards)
  (www.incits.org/
- ISO (International Organization for Standardization)
  (www.iso.ch/)
- ISOC (Internet Society)
  (www.isoc.org/)
- ITU (International Telecommunication Union)
  (www.itu.ch/)
- Konnex Association
  (www.konnex.org/)
- MPI (Message Passing Interface Forum)
  (http://www.mpi-forum.org/)
- MPEG (Moving Picture Experts Group, ISO/IEC JTC1/SC29 WG11)
  (www.cselt.it/mpeg/)
- NIST (National Institute of Standards and Technology)
  (csrc.ncsl.nist.gov)
- OMG (Object Management Group)
  (www.omg.org/)
- The Open Group
  (www.opengroup.org)
- OSGi Alliance (Open Service Gateway Initiative)
  (www.osgi.org)

- TCG (Trusted Computing Group) (https://www.trustedcomputinggroup.org)
- UPnP-Forum (Universal Plug and Play Forum) (www.upnp.org)
- W3C (World Wide Web Consortium) (www.w3.org)
- WiMedia Alliance (www.wimedia.org)
- ZigBee Alliance (www.zigbee.org)

10. Glossar

Der hier vorgestellte Glossar enthält eine Sammlung an Begriffen, die für das Projekt DynAMITE relevant sind. Es dient dem gemeinsamen Verständnis der in dem Projekt häufig verwendeten Begrifflichkeiten und soll auch als ständiges Nachschlagewerk dienen. Die Sammlung wird während der Projektlaufzeit ständig um neue Begriffe erweitert.

10.1. Ad-hoc Networking

Ad-Hoc Netzwerke unterscheiden sich von klassischen Netzwerken dadurch, dass sie keinerlei feste Infrastruktur und Topologie besitzen. Oft wird auch angenommen, dass alle Teilnehmer in einem solchen Netz mobil sind und daher die Zusammensetzung zeitvariant ist.
Die Teilnehmer kommunizieren häufig drahtlos und die gesamte Netzstruktur entsteht dynamisch durch Selbstorganisation und Selbstverwaltung. Es gibt also keine zentrale Stelle, die das Routing oder die Netzstruktur festlegt, das Management in Ad-hoc-Netzwerken ist somit verteilt.
Typische Anwendungsgebiete sind Koordinierung von Rettungsaktionen in Krisengebieten oder Einsatz in militärischen Kampfgebieten. Ein weiterer Bereich ist die Verkehrstelematik.

10.2. Agent, Transducer und Channel

Jacques Ferber [1], [2] liefert eine generelle Definition eines Agenten (soll aus Gründen der Authentizität in der Originalsprache bleiben):
An agent is a physical or virtual entity
1. which is capable of acting in an environment.
2. which can communicate directly with other agents.
3. which is driven by a set of tendencies (in the form of individual objectives or of a satisfaction/survival function which it tries to optimize).
4. which possesses resources of its own.
5. which is capable of perceiving its environment (but to a limited extent).
6. which has only a partial representation of its environment (and perhaps none at all).
7. which possesses skills and can offer services.
8. which may be able to reproduce itself.
9. whose behaviour tends towards satisfying its objectives, taking account of the resources and skills available to it and depending on its perception, its representation and the communications it receives.

Bei Wooldridge [4] wird der Begriff Agent folgendermaßen definiert:

An agent is a computer system that is situated in some environment,
and that is capable of autonomous action in this environment in order to meet its design objectives.
Autonomous action means that agents are able to act without the intervention of
humans and other systems: they have control both over their own internal state, and over their behaviour.

Intelligent agents are capable of flexible autonomous actions, where flexibility means:
1) Reactivity: intelligent agents are able to perceive their environment, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.
2) Pro-activeness: intelligent agents are able to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives.
3) Social ability: intelligent agents are capable of interacting with other agents (and possibly humans) in order to satisfy their design objectives.

Generell können noch Untergruppen von Agenten (z.B. Mobile Agenten, Situative Agenten, etc.) gebildet werden, die dann die obigen Eigenschaften um zusätzliche Eigenschaften ergänzt.

Wenn man die Definitionen von Wooldrige und Ferber zusammen betrachtet und den Dynamite-Kontext (wir interessieren uns nur für intelligente Agenten) im Hinterkopf behält,
scheinen folgende Stichwörter für einen Agenten essentiell zu sein:
1) Social ability - communicative (Punkte 2, 9 bei Ferber)
2) Reactivity - perceive and respond (Punkte 1, 5, 6 bei Ferber)
3) Pro-activeness -- goal-orientation (Punkt 3, 7, 9 ber Ferber)

4) Autonomy

Folgende Definition eines Agenten kann sich daraus ergeben:
An agent is a computer system that is
(1) situated in some environment, capable of perceiving the environment and responding in a timely fashion to changes that occur in it,
(2) that is able to interact with other agents and possibly humans, and
(3) that is able to exhibit goal-directed behaviour by taking the initiative in order to satisfy their design objectives.

Channels und Transducer:
Gemäß der Soda-Pop-Definition [3] teilt sich eine Topologie von Komponenten in Channels und Transducern auf. Dabei organisieren sich Transducer (die in erster Näherung als Agenten angesehen werden können) gemäß der von ihnen unterstützten Ontologien in eine Topologie, deren einzelne Ontologiegrenzen durch Channels separiert werden. Channels und Transducer sind wie folgt definiert:

Channels:
- arbeiten auf derselben Ontologie
- sind verteilt
- besitzen keinen Speicher
- Nachrichten werden ohne jede Zeitverzögerung weitergesendet

Transducer:
- Lesen eine (oder mehrere) Nachrichten und generieren Ausgangsnachrichten
- mappen Ontologien, wandeln somit eine Nachricht der einen Ontologie in eine Nachricht einer anderen Ontologie um (siehe z.B. Kanalflusstopologie von EMBASSI)
- besitzen Speicher
- sind lokalisiert und nicht verteilt

[1] Jacques Ferber: Multiagentensysteme, Addison-Wesley, 2001
[2] Ferber: Definition of Agent: http://www.ryerson.ca/~dgrimsha/courses/cps720/agentDef.html
[3] Heider Th., Kirste Th.: Architecture considerations for interoperable multi-modal assistant systems, PreProceedings of the 9th International Workshop on Design, Specification, and Verification of Interactive Systems; DSV-IS 2002, 6,2002, Rostock, S.403 – 417, zu finden unter: http://www.embassi.de/publi/veroeffent/dsvis.pdf
[4] M. Wooldridge. Intelligent agents. In Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence, G. Weiss, editor, pages 21--35. The MIT Press, 1999.

10.3. Conflict Resolution, Conflict Solving

In →Systemen, bei denen sich mehrere Bewerber auf ein Angebot (eine Ressource, ein Auftrag, …) melden können, muss zwischen Bewerbern ausgewählt werden. Die Verfügbarkeit mehrerer Bewerber ist der *Konflikt*. Die Auswahl ist die *Konfliktlösung*, und das zur Anwendung kommende Auswahlverfahren ist die *Konfliktlösungsstrategie*. Einfache Strategien wählen den ersten Bewerber („first match“, vgl. Prolog) oder treffen eine Zufallsauswahl. Komplexe Strategien versuchen die Eignung der Bewerber zu bewerten um so die beste Wahl zu treffen („best match“, vgl. CLIPS). „Best match“-Strategien erfordern ein elaboriertes System für die Bewertung der Qualität einer Bewerbung.

Es ist auch denkbar, dass *kein* Bewerber das *komplette* Angebot wahrnehmen möchte. In diesem Fall kann die Konfliktlösung ein →Problemdekompositionsverfahren einsetzen, um das Angebot sinnvoll auf die Bewerber aufzuteilen.

10.4. Dezentraler Ansatz

Der Begriff dezentraler Ansatz ist in der Agentenliteratur nicht etabliert. Es handelt sich vielmehr das Ziel des Projektes DynAMITE eine Topologie für Komponenten zu entwickeln, die aufgrund ihrer Fähigkeiten zur Selbstorganisation keinerlei zentralen Komponenten mehr benötigen. Wenn man diesen Begriff anwendet auf die FIPA- und KQML-Welt bedeutet dies den Verzicht auf einen zentralen Router, die Yellow-Pages und den White-Page-Dienst.

In der Open Agent Architecture würde dies zusätzlich noch den Verzicht auf spezielle Agenten, die Regeln mit sich bringen, bedeuten.
Nur durch den dezentralen Ansatz kann gewährleistet werden, dass der Ausfall einer Komponente nicht den Totalausfall des Gesamtsystems bedeuten muss. Ebenso ist nur so die spontane Zusammenfindung von Agenten zu Ensembles und damit zu einer selbstorganisierten Topologie möglich.

10.5. Ensemble

Als ein Ensemble wird ein Gruppe von Agenten angesehen, die sich aufgrund einer gegebenen Aufgabe zusammenfinden, um diese Aufgabe gemeinsam zu bewältigen. Bei einem Ensemble muss es sich somit nicht um die Gesamtheit aller Komponenten innerhalb einer Plattform handeln, sondern die Zugehörigkeit zu einem bestimmten Ensemble kann von Aufgabe zu Aufgabe wechseln.
[1] Weiss, G., (Ed.), 1999. Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence, The MIT Press, Cambridge, Massachusetts

10.6. Framework

Der Begriff des Frameworks bezeichnet eine vorgefertigte und domänenbezogene Softwarearchitektur die explizit dafür entworfen wurde, als wiederverwendbare Lösung für eine Familie von Systemen innerhalb einer Anwendungdomäne zu dienen, nicht als Gerüst einer konkreten Einzelanwendung[1].
Ein Framework zeichnet sich durch einen hohen Grad von Adaptier-und Erweiterbarkeit aus. Wie eine allgemeine Softwarearchitektur besteht ein Framework aus Komponenten.
Ein Framework sollte hinsichtlich seiner Komponenten flexibel, d.h. durch neue Komponenten erweiter- und veränderbar sein.
Ein Framework grenzt sich dadurch von einer reinen Klassenbibliothek ab, dass es sich bei Ersterem um eine vordefinierte Architektur handelt, die zusätzlich das Zusammenwirken zwischen verschiedenen Komponenten modelliert. Da die Architektur selbst wiederverwendet werden kann, erreicht man durch ein Framework einen Zugewinn an Wiederverwendbarkeit [2,3].
[1] Krysztof Czarnecki und Ulrich W. Eisenecker: "Generative Programming: Methods, Tools, and Applications", Addison-Wesley, 2000.
[2] Ilka Philoppow und Matthias Riebisch: "Systematic Definition of Reusable Architectures",
8 th International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 01), 2001.
[3] Marcus Fontoura, Wolfgang Pree und Bernhard Rumpe, "The UML Profile for Framework Architectures", Addison-Wesley, 2002.

10.7. Interoperabilität

Folgende Definition findet sich im Siemens Online Lexikon [1]:
Die Fähigkeit eines Gerätes, bei vergleichbarer Systemumgebung in einem Netz mit anderen Geräten desselben Standards sinnvoll kommunizieren zu können. Dabei sollte es keine Rolle spielen, dass die Geräte von verschiedenen Herstellern stammen.
Der Begriff repräsentiert das Ziel des Internetworking besonders dadurch, dass eine abstrakte, Hardware-unabhängige Netzwerkumgebung geschaffen wird. Diese ermöglicht es, Computer über die Netzwerkschicht miteinander zu verbinden, so dass die Zusammenarbeit möglich ist, ohne dass die Beteiligten wissen, welche Technik den benutzten Geräten zugrunde liegt.
[1] Siemens Online Lexikon
http://search.siemens.com/cgi-bin/keyword_redirect.pl?dest=http://w3.siemens.de/solutionprovider/lexikon/index.htm

10.8. Middleware

Middleware is a software that mediates between two separate and often already existing application programs. It manages the interaction between them across the heterogeneous computing platforms. The Object Request Broker (ORB) defined by CORBA, software that manages communication between objects, is an example of a middleware program.
Definition angelehnt an http://www.hyperdictionary.com/

10.9. Modalität

In [1] wird eine Modalität wie folgt für den Bereich der Mensch-Maschine-Kommunikation definiert: "Mit Modalität bezeichnen wir die menschlichen Sinne, die Informationen verarbeiten, d.h. sehen, hören, riechen und fühlen" [1].
Eine weitere Definition nach Bernsen bezeichnet eine Modalität als einen "Modus oder Weg, einem Menschen oder einer Maschine Informationen zu präsentieren" [2].
Es wird im Bereich der Mensch-Maschine-Kommunikation gewöhlich unterschieden zwischen Eingabemodalitäten (z.B. gesprochene Sprache oder Mauseingaben) und Ausgabemodalitäten (z.B. Grafik oder Sprachausgabe).
[1] Wahlster, W. (1998). Intelligent User Interfaces - An Introduction. In Maybury, M. T. and Wahlster, W., editors, Readings in Intelligent User Interfaces.
[2] Bernsen, N. O. (2001). Multimodality in Speech and Language – From Theory to Design Support Tool. In Granström, ed., Multimodality in Language and Speech Systems. Kluwer Academic Publishers.

10.10. Multi-Modalität

Ein multimodales System ist ein System, das mehrere Eingabemodalitäten verarbeiten kann (z.B. Sprache und Gestik) und/oder mehrere Ausgabemodalitäten generieren kann (z.B. Grafik oder Sprachausgabe). Vgl. Definition von 'Modalität'.

10.11. Ontologie

Eine Ontologie beschreibt einen Wissensbereich (knowledge domain) der realen Welt mit Hilfe einer standardisierenden Terminologie, d.h. Begriffen und Benennungen. Sie definiert die Objekte, Konzepte, Relationen oder andere Entitäten des Wissensbereichs sowie die Relationen unter diesen Begriffen [1].

Anmerkung 1: In vielen Fällen handelt es sich bei den heute als Ontologien bezeichneten Strukturen lediglich um kontrollierte Vokabularien wie Klassifikationen oder Thesauri. Von der Möglichkeit von Relationen über Relationen und Regeln wird unter anderem aufgrund ihrer Komplexität relativ selten Gebrauch gemacht, obwohl gerade diese Merkmale Ontologien von anderen Begriffssystemen unterscheiden [2].

Anmerkung 2: In einem System von Softwarekomponenten/agenten bilden Ontologien die Grundlage für konsistentes und kohärentes Agieren der Einzelkomponenten [3].

[1] Wolfgang Hesse,Informatik Spektrum Volume 25, Number 6, (2002)
[2] WIKIPEDIA, die freie Enzyklopädie, http://de.wikipedia.org/wiki/Ontologie_(Informatik)
[3] Tom Gruber, "What is an ontology?", http://ksl-web.stanford.edu/kst/what-is-an-ontology.html

10.12. OSI 7-Schichtenmodell

Das ISO/OSI-Referenzmodell, auch Schichtenmodell genannt, wurde 1979 von der ISO als Vorschlag entwickelt.
Das Referenzmodell dient als Verständigungsbasis zur Einteilung der Kommunikation von Rechnersystemen in einzelne Schichten (Layer). Das Modell sieht sieben Schichten vor, wobei jede Schicht der nächsthöheren Schicht ihre Dienste (Services) anbietet. Jede Schicht kommuniziert nur mit der nächsthöheren und nächstniedrigeren Schicht.
Jede höhere Schicht bedeutet einen weiteren Abstraktionsgrad der tiefer liegenden Schicht.
TCP/IP lässt sich nicht an allen Stellen eindeutig in das theoretische OSI-Modell einordnen. Es wird angemerkt, dass in Bezug auf die Internet-Protokolle häufig die oberen drei Schichten als eine "Anwendungsschicht" zusammengefasst werden (s. [5],[3]).
Im Folgenden werden die einzelnen Schichten sehr knapp beschrieben sowie einige Beispiele ([3]) genannt:

- Schicht 7 (Anwendungsschicht / Application Layer): bildet die Schnittstelle zum Anwendungsprogramm. Beispiele: HTTP, SMTP, SNMP, FTP, Telnet, NFS
- Schicht 6 (Darstellungsschicht / Presentation Layer): enthält Dienste wie Datenkompression und Verschlüsselung, transformiert die Anwenderdatenstrukturen in ein Standardformat. Beispiele: ASN.1, SMB
- Schicht 5 (Sitzungsschicht / Session Layer): steuert den Kommunikationsablauf zwischen beteiligten Endeinrichtungen. Beispiele: RPC, NetBIOS

- Schicht 4 (Transportschicht / Transport Layer): steuert Beginn und Ende einer Datenübertragung, nimmt Segmentierung und Zusammenfassung von Nachrichten vor, übernimmt die Fehlerbehandlung und Datensicherung. Beispiele: TCP, UDP
- Schicht 3 (Netzwerkschicht / Network Layer): Aufgaben dieser Schicht sind u.a. das Routing und Interpretieren von Adressen. Beispiele: IP, ICMP, ARP, X.25
- Schicht 2 (Leitungsschicht / Data Link Layer): interpretiert den Bitstrom aus Schicht 1 als Folge von Datenblöcken. Beispiele: Ethernet, Token ring, PPP, ISDN, ATM
- Schicht 1 (physikalische Schicht bzw. Bitübertragunsschicht / Physical Layer): ermöglicht die übertragung von Bits über ein Kommunikationsmedium. Beispiele: Elektrizität, Radio, Laser

[1] Informatik-Duden, Meyers Lexikonverlag, 1993
[2] B. Walke, Mobilfunknetze und ihre Protokolle (Band 1), B.G. Teubner, 2000
[3] Wikipedia, http://en2.wikipedia.org/wiki/TCP/IP
[4] C. Hunt, TCP/IP Network Administration, O'Reilly, 1994
[5] W.R. Stevens, Programmieren von UNIX-Netzwerken, Hanser, 2000

10.13. peer-to-peer Systeme

Peer-to-Peer (P2P) systems are distributed systems based on the concept of sharing by direct exchange between peer nodes. In contrast to a client-server model, each peer may function as a client and/or a server. Moreover, peer-to-peer systems generate their own organisation for their nodes, which is related to self-organization, the important difference from the centralized organization of client-server model:
1) With respect to a given P2P community, any peer granted permission to become a node in the community is, at the same time, granted equality to every other node in the community. Adding peers to a network requires no re-organization, central or otherwise.
2) Peer node organization is independent of whether an individual node is connected or not. This is the feature that handles variable connectivity. This is the same feature that facilitates scalability.
Definition angelehnt an http://www.ida.liu.se/conferences/p2p/p2p2001/p2pwhatis.html

10.14. Pervasive Computing

Häufig wird der Begriff "Pervasive Computing" im gleichen Umfeld wie der Begriff "Ubiquitous Computing" verwendet. Oftmals werden sie sogar synonymisch verwendet.
Hierzu die Definition von searchNetworking.com im Originaltext:
Pervasive computing is the trend towards increasingly ubiquitous (another name for the movement is ubiquitous computing), connected computing devices in the environment, a trend being brought about by a convergence of advanced electronic - and particularly, wireless - technologies and the Internet. Pervasive computing devices are not personal computers as we tend to think of them, but very tiny - even invisible - devices, either mobile or embedded in almost any type of object imaginable, including cars, tools, appliances, clothing and various consumer goods - all communicating through increasingly interconnected networks. According to Dan Russell, director of the User Sciences and Experience Group at IBM's Almaden Research Center, by 2010 computing will have become so naturalized within the environment that people will not even realize that they are using computers. Russell and other researchers expect that in the future smart devices all around us will maintain current information about their locations, the contexts in which they are being used, and relevant data about the users.
The goal of researchers is to create a system that is pervasively and unobtrusively embedded in the environment, completely connected, intuitive, effortlessly portable, and constantly available. Among the emerging technologies expected to prevail in the pervasive computing environment of the future are wearable computers, smart homes and smart buildings. Among the myriad of tools expected to support these are: application-specific integrated circuitry (ASIC); speech recognition; gesture recognition; system on a chip (SoC); perceptive interfaces; smart matter; flexible transistors; reconfigurable processors; field programmable logic gates (FPLG); and microelectromechanical systems (MEMS).

A number of leading technological organizations are exploring pervasive computing. Xerox's Palo Alto Research Center (PARC), for example, has been working on pervasive computing applications since the 1980s. Although new technologies are emerging, the most crucial objective is not, necessarily, to develop new technologies. IBM's project Planet Blue, for example, is largely focused on finding ways to integrate existing technologies with a wireless infrastructure. Carnegie Mellon University's Human Computer Interaction Institute (HCII) is working on similar research in their Project Aura, whose stated goal is "to provide each user with an invisible halo of computing and information services that persists regardless of location." The Massachusetts

Institute of Technology (MIT) has a project called Oxygen. MIT named their project after that substance because they envision a future of ubiquitous computing devices as freely available and easily accessible as oxygen is today.

10.15. Plattform

Der allgemeine Begriff "Plattform" kann sich, je nach Kontext, auf eine Hardware- und/oder Software-Plattform beziehen.
Der Begriff der Hardware-Plattform bezeichnet den physischen Teil eines Produkts. Eine Software-Plattform beschreibt eine Softwareschnittstelle, je nach Kontext etwa die Betriebssystemebene oder die Ebene einer Middleware. Bei der Entwicklung eines Systems bezeichnet der Begriff Plattform häufig die Ausgangsbasis, z.B. es steht ein Intel Pentium-II PC mit dem Betriebssystem Linux SuSE 7.3 als Plattform zur Verfügung. Die gewählte Plattform bedeutet in der Regel eine Einschränkung des zu entwickelnden Systems, das die Vorraussetzungen dieser Plattform erfüllen muss.

10.16. Problem Decomposition

Problemdekomposition (PD) ist die Zerlegung einer komplexen Aufgabe in mehrere einfachere Teilaufgaben. PD kann verwendet werden, um Parallelverarbeitung zu realisieren, aber auch um mit einfachen Komponenten komplexe Probleme lösen zu können. Problemdekomposition erfordert eine *Zerlegungsstrategie*, mit deren Hilfe die komplexe Aufgabe in Teilaufgaben zerlegt werden kann, eine *Rekombinationsstrategie*, die die Erzeugung des Gesamtergebnisses aus den Teilergebnissen erlaubt und eine *Koordinationsstrategie*, die – wo notwendig – die Einhaltung von Synchronisationsbedingungen bei der Bearbeitung der Teilaufgaben sicherstellt.
Ein einfaches Beispiel für Problemdekomposition ist der Quicksort-Algorithmus *qsort*. Die Problemdekomposition besteht aus der der Wahl eines Pivot-Elements p aus der Eingabeliste lp und die Zerlegung von $l=lp-\{p\}$ in zwei neue Listen: a, bestehend aus den $x \in l$ mit $x \leq p$ und b, bestehend aus den $x \in l$ mit $x > p$. Die Rekombinationsstrategie ist dann einfach die Konkatenation von *qsort* a, $\{p\}$, und *qsort* b. Hier ist sehr deutlich sichtbar dass Problemdekomposition dann besonders einfach zu realisieren ist wenn die *Bearbeitungsstruktur* eines Auftrags mit der syntaktischen Struktur der Auftragsbeschreibung korresponiert.

10.17. Selbstorganisation

Der Aufbau von Strukturen und → Systemen aus ungeordneten Komponenten ohne äußere Einwirkung oder Steuerung. Die formgebenden beschränkenden Einflüsse gehen also von den Elementen des sich organisierenden Systems selbst aus, ohne dass die Elemente Kenntnis von der Gesamtstruktur des Systems haben müssen. (Ein System mit einer zentralen Steuerkomponente - selbst wenn diese dynamisch gewählt wird - ist daher nicht im strengen Sinn selbstorganisierend.) Differenzierungs- und Strukturbildungsprozesse in biologischen Geweben sind typische Beispiele für selbstorganisierende Systeme, weitere gute Beispiele finden sich im Verhalten physikalischer und chemischer Reaktionssysteme.
Einen überblick über das Thema bietet Self-Organizing Systems (SOS) FAQ:
http://www.calresco.org/sos/sosfaq.htm

Eine Grundfrage im Bereich selbstorganisierender Systeme ist wie operatives Strukturwissen verteilt repräsentiert werden kann. D.h., welche lokale Verhaltensmuster müssen die einzelnen Komponenten aufweisen, damit sie in einer Gruppe spontan Strukturen ausbilden können. Ein Beispiel ist die spontane Bildung eines Kreises aus zweiarmigen Elementen mit den lokalen Verhaltensmustern: "suche dir einen linken und einen rechten Partner" und "sorge dafür, dass der linke und der rechte Arm im gleichen Winkel zum Körper stehen".

10.18. Service Composition

Mit Service Composition wird die Technology beschrieben, existierende Dienste zu verbundenen Diensten zusammenzuführen.
Dadurch soll i.a. Mehrwert erreicht werden, da die Summe aller existierenden Dienste gebündelt höher ausfällt als die Einzeldienste zusammengezählt ergeben. Daher hängt dieser Begriff stark mit der Service Discovery zusammen.
Ein gutes Beispiel, welches sowohl Problem Decompostion als auch Service Compositon beinhaltet ist das Zusammenspiel von Planer und Scheduler in EMBASSI. Nachdem der Planer ein (höheres) Ziel in Einzelziele

(die mit der vorhandenen Infrastruktur erreichbar sind) heruntergebrochen hat,werden diese Einzelziele durch den Schedular ausgeführt, um den Mehrwert, nämlich das höhere Ziel zu erreichen.

Ein formales Beispiel:
Es gibt eine Funktion a: x → y und eine Funktion b: y → z
Einzeln betrachtet sind zwei (mathematische) Umformungen möglich.
Aber durch Service Composition ist das Ziel: x → z durch die Transformation in die beiden Teilziele x → y und y → z möglich.
Service Composition hat somit noch ein drittes zu erreichendes Ziel möglich gemacht.
Daher: 1 Dienst + 1 Dienst = 3 mögliche Dienste

Heute spielt Service Composition vor allem im Bezug auf die WebServices eine grosse Rolle (siehe Service Discovery).

10.19. Service Discovery

Die Metapher des Service Discovery ist ein Thema sowohl in Agentensystemen als auch auf Betriebssystemebene.
Bei Apples Rendezvous erlaubt die DNSServiceDiscovery die Registrierung von Netzwerkdienste, wie Drucker oder Fileserver, dass sie von anderen Applikationen gefunden und benutzt werden können. Ähnliche Verfahren finden sich auch bei Jini oder Microsoft Windows.
Agentensysteme verfolgen einen ganz ähnlichen Weg. Agenten melden ihre Dienste und Fähigkeiten am sogenannten Yellow-Page-Dienst an, so dass andere Komponenten auf der Suche nach Dienstleistern nach diesem Dienst fragen können und einen Adressaten erhalten. Diese Technik findet sich (m.E.) in jeder Agentenplattform. Auch im Internet hat die Metapher des Service Discovery Einzug gehalten. SOAP als WebService erlaubt die Auffindung von Diensten im Web und liefert für die Inanspruchnahme die nötigen Protokolle und Klassen mit.

10.20. System

Eine Menge von unabhängigen Komponenten, die miteinander in Wechselwirkung stehen (z.B. über Kommunikationsmechanismen) und die als Gesamtheit ein umfangreicheres Verhaltensrepertoire besitzt als die Summe ihrer Bestandteile.

10.21. Topologie

Zitat aus dem Brockhaus:
„Topologie die, Teilgebiet der Mathematik,
das ursprünglich als »Geometrie der Lage« Eigenschaften geometrischer Gebilde behandelte.
Die moderne Topologie umfasst die Theorie topologischer Räume.“

Topologie im Sinne einer Komponentenarchitektur.
In der Agententechnologie dient die Definition einer Topologie der Beschreibung der Art der Organisation von Komponenten innerhalb der Komponentengemeinschaft. Topologie hier meint mehr als die Architekturdefinition. Topologie definiert ebenso die Art der verwendeten Ontologie und das Mapping der Ontologien (u.a.), während herkömmlich unter Architektur die Anordnung (und bei Agentennetzen die Kommunikationswege) verstanden wird.
Die Begriffe Topologie, Architektur und Organisation werden jedoch in der Literatur häufig als Synonyme verstanden.

10.22. Ubiquitous Computing

Auszug aus der Free On-Line Dictionary of Computing (FOLDOC):
Computer überall: Viele Computer werden durch die physikalische Umgebung verfügbar gemacht, wobei diese effektiv für den Benutzer nicht sichtbar bzw. nicht direkt manipulierbar sind. Manchmal wird Ubiquitous Computing auch als "the Third Wave of Computing" bezeichnet. Die erste Welle bezeichnete die Tatsache, dass viele Menschen sich einen Computer geteilt haben, die zweite Welle war die Benutzung eines Computers pro Menschen. In der dritten Welle werden dementsprechend alle Menschen von einer Vielzahl an Computern umgeben sein.

Drei Schlüsseltechnologien werden für Ubiquitous Computing identifiziert:
Energieverbrauch, User Interfaces und drahtlose Verbindungen.

Mark Weiser wird oftmals als der "Vater" des Ubiquitious Computing" bezeichnet, hier findet sich eine Einführung in das Themengebiet und interessante Verweise:
http://www.ubiq.com/hypertext/weiser/UbiHome.html

Eine Definition interessanter Themengebiete und einen guten überblick liefert der Arbeitsbereich Verteilte Datenbanken und Systeme, TU Darmstadt:
http://www.dvs1.informatik.tu-darmstadt.de/research/percom/research_main.html

Häufig wird der Begriff Ubiquitous Computing synonymisch mit dem Begriff → Pervasive Computing verwendet.