Integrationskonzept für den Entwurf multimedialer Umgebungen

Jörg Berdux

Integrationskonzept für den Entwurf multimedialer Umgebungen

Zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften
der Fakultät für Informatik der Universität Karlsruhe (TH)

Genehmigte Dissertation von Jörg Berdux aus Marburg

Erster Gutachter: Prof. Dr.-Ing. D. Schmid
Zweiter Gutachter: Prof. Dr. rer. nat. Abeck

Tag der mündlichen Prüfung: 17. April 2002

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der Begriff Multimedia wurde in den 90er Jahren zum Schlagwort der IT-Branche. Doch die Bedeutung dieses neuen Begriffs mit all seinen Facetten, wie neue Medien, digitale Medien, interaktive Medien etc., wird sich erst in den nächsten Jahren genauer fassen lassen, da der heutige Umbruch von der Industrie- zur Informationsgesellschaft erst der Beginn eines umfassenden Wandels ist, der in alle Bereiche des Lebens hineinspielen wird. Mit dieser Entwicklung bahnt sich auch eine neue Wertigkeit an, bei der nicht mehr materielle Güter, sondern immaterielle Daten bzw. Informationen im Vordergrund stehen [Schm99].

Betrachtet man die traditionellen Medien (Theater, Buch, Foto, Film, Fernsehen, ...), so bezeichnet der Begriff Medium den Träger bzw. den Vermittelnden der „Information". Durch das Zusammenwirken der einzelnen Medien, die durch den Computer nur noch als immaterielle Daten verarbeitet werden, ergibt sich ein neuer Medienbegriff, bei dem der Computer zum einen als Projektionsgerät, zum anderen als aktive Schnittstelle für den Rezipienten zur Verfügung steht.

Der Computer als Projektionsgerät ermöglicht den Übergang von dem materiellen, physischen zum immateriellen, abstrakten Ort als Informationsstätte. Durch das Zusammenwirken und Verschmelzen unterschiedlicher (immaterialisierter) traditioneller Medienarten entstehen neue Formen und Orte der Vermittlung [Schw95]. Auf diese Verschmelzung lässt sich zunächst auch der Begriff Multimedia zurückführen, der aus technischer Sicht als Verbindung zeitkonstanter (Text, Bild, ...) mit zeitkontinuierlichen (Film, Ton, ...) Medien interpretiert wird [Stei99] [WaHi00]. Diese Interpretation bezieht sich jedoch nur auf traditionelle Medien, so dass sie lediglich einen Teilaspekt multimedialer Präsentationen abdeckt.

Aus medientheoretischer Sicht ist die zeitliche Aufzeichnung bzw. Wiedergabe ein wesentlicher Faktor multimedialer Präsentationen, dessen Grundlagen auf den Film und das Kino zurückzuführen sind. Betrachtet man den Film als erstes Medium, das die Manipulation von Raum und Zeit durch eine lineare Aufzeichnung von Einzelbildern ermöglicht hat, so kann die digitale Repräsentation als Fortführung dieses Gedankens angesehen werden [Mano95]. Im Gegensatz zu dem linearen Abspielen eines Films kommt durch die digitale Speicherung, Bearbeitung und den digitalen Zugriff die Möglichkeit des Direktzugriffs hinzu. Der direkte Zugriff auf unterschiedliche Daten ermöglicht es, mehrschichtige Wahrnehmungsebenen zu kombinieren, so dass die Präsentation unterschiedliche Sinne des Rezipienten

ansprechen kann. Das Zusammenwirken verschiedener Sinneseindrücke wird dabei häufig mit dem Gedächtnis verglichen, das einzelne Erinnerungsstücke miteinander verbindet und in einen (zeitlichen) Bezug setzt [Scot95].

Neben dem zeitlichen Aspekt kann durch die digitale Bearbeitung die analoge Vorlage in der Realität durch computergenerierte Bilder, Töne etc. ergänzt bzw. ersetzt werden. Mithilfe des Computers wird somit eine künstliche bzw. virtuelle Realität erschaffen, die ausschließlich auf Algorithmen beruht. Die Virtualität basiert auf der Integration in eine imaginäre Welt, in der der Rezipient Sinneseindrücke erfährt, die ihm in seiner realen Umgebung nicht oder nur eingeschränkt zugänglich sind. Die grundlegenden Eigenschaften der virtuellen Realität beruhen dabei auf der Erzeugung einer symbolischen Welt mit eigenen Regeln und der Möglichkeit der Beobachtung und Teilnahme an dieser Welt [Kräm88]. Dies bedeutet für die Gestaltung multimedialer Präsentationen, dass neben der „Gestaltung" eines semantischen Raums durch eine medien- und inhaltsbezogene Darstellung, die Schnittstelle zwischen Realität und virtueller Realität von entscheidender Bedeutung ist. Diese Schnittstelle sollte die verschiedenen Sinne in gleicher Weise ansprechen, um den Rezipienten sowohl räumlich als auch sinnlich zu integrieren.

Gerade die Schaffung einer geeigneten Schnittstelle zwischen beiden Umgebungen berührt insbesondere den Aspekt der Interaktion multimedialer Präsentationen. Durch die Interaktion wird das traditionelle Sender-/Empfänger-Prinzip aufgehoben und somit der Rezipient über das gleiche Medium zum Sender bzw. Akteur. Diese interaktive Komponente eröffnet auch den Interpretationsraum, in welchem die Medien nicht nur gedeutet, sondern auch beeinflusst werden können.

Die Interaktion ermöglicht dem Rezipienten, unterschiedliche Beziehungen und zeitliche Zusammenhänge individuell zu bestimmen, d.h. im Gegensatz zu den traditionellen linearen Medien beeinflusst der Benutzer den Weg der „Erzählung" selbst. Greift man die Metapher des Gedächtnisses wieder auf, können die einzelnen Daten als Gedanken bzw. Erinnerungen interpretiert werden. Diese Metapher war bereits 1945 Grundlage des von Vannevar Bush entworfenen Systemkonzepts *Memex* [Bush45], das als Ursprung des World Wide Web angesehen werden kann. Basierte dieses Konzept noch auf analogen Medien, so wurde erst durch die Verschmelzung zwischen den traditionellen Medien und dem Computer die Gleichbehandlung von Daten und Zugriffsstrukturen möglich. Ebenso wie bei der Darstellung digitaler Medien ergeben sich damit grundlegend neue Eigenschaften für die Realisierung interaktiver multimedialer Anwendungen.

Die Zuordnung von Eigenschaften ermöglicht, die digitalen Medien in der virtuellen Umgebung als eigenständige Elemente zu interpretieren, die durch den Benutzer und seine Umgebung beeinflusst werden können. Auch hier gilt es, eine eigene „Gestaltung", nun bezogen auf das Verhalten der virtuellen Umgebung, zu erschaffen. Somit sollte die Schnittstelle zwischen realer und virtueller Umgebung auch für die Interaktion einen natürlichen und damit fließenden Übergang unterstützen.

Multimediale Anwendungen und Wahrnehmungsumgebungen

Durch den Computer werden die neuen Gestaltungsformen gegenüber „traditionellen" Medien durch die Verbindung von Daten, Interaktionsbeschreibung, Speicherung, Algorithmen etc. innerhalb eines Mediums unterstützt. Im Gegensatz zu den „traditionellen" Medien, kann somit das Projektionsgerät (Präsentationsmedium) als Softwareprogramm verändert werden. In den meisten Fällen dient das Programm zurzeit als Präsentationsmedium „traditioneller" multimedialer Anwendungen, die als geschlossene Präsentation auf einem Computer ablaufen. Durch die Bedeutung der Schnittstelle zwischen realer und vir-

tueller Umgebung gewinnen aber in zunehmenden Maße neue multimediale Präsentationsformen an Gewicht. Dabei wird der Computer, der letztendlich aus der Arbeitswelt stammt, nicht mehr als alleiniges Abspielgerät (Präsentationsmedium) multimedialer Daten verwendet, sondern als Teil einer zu gestaltenden Umgebung eingesetzt.

Gerade in den typischen multimedialen Genres wie Entertainment, Infotainment, Edutainment und der interaktiven Medienkunst finden immer häufiger derartige verteilte multimediale Anwendungen ihr Einsatzgebiet. Hier wird nicht nur die Wiedergabe virtueller Daten betrachtet, sondern ebenso die Einbeziehung der realen Umgebung des Rezipienten. Durch den fließenden Übergang zwischen virtueller und realer Umgebung werden dabei die Grenzen zwischen beiden „Welten" aufgehoben, so dass von einer so genannten Wahrnehmungsumgebung gesprochen wird [WiKD98]. Die Gestaltung von Wahrnehmungsumgebungen betrifft sowohl die Wiedergabe der Medien durch unterschiedliche Präsentationsformen als auch die Interaktion. Wie in Kapitel 7 anhand von Beispielen gezeigt wird, kann so z.B. die Fassade eines Gebäudes durch ansteuerbare Lichtelemente als virtuelle Projektionsfläche der realen Umgebung dienen, wobei die Farben der Fassade die Helligkeit der Umgebung widerspiegeln und sich der Geschwindigkeit und Richtung des Winds anpassend über die Fassade bewegen.

Technische und gestalterische Sicht

Grundlegend für die in dieser Arbeit betrachteten Ansätze sind die während des Entwurfs von Wahrnehmungsumgebungen gemachten Erfahrungen. Derartige verteilte multimediale Anwendungen wurden in interdisziplinär aufgebauten Teams erstellt, die aus Studenten und Universitätsmitarbeitern der Informatik und gestalterischer Fachrichtungen bestanden.

Die Projekte hatten einen experimentellen Charakter, um traditionelle Grenzen der interaktiven Mediengestaltung auszuwägen und neue Wege zu beschreiten. Hierbei wurde gerade die Gestaltung zwischen realer und virtueller Umgebung betont, die zu unterschiedlichen Ausprägungen von Wahrnehmungsumgebungen führten. So wurden neben eher „traditionellen" multimedialen Dokumentationen verteilte Anwendungen für museale Präsentationsformen und für die Gestaltung von öffentlichen Räumen erprobt. Wie bereits weiter oben erwähnt wird dabei ein fließender Übergang zwischen virtueller und realer Umgebung angestrebt, in der sich der Rezipient bzw. (Be-)Nutzer in einer eigenen Erfahrungswelt bewegt. Für die Schaffung derartiger Wahrnehmungsumgebungen bedeutet dies aber, dass nicht ausschließlich einzelne lokale multimediale Anwendungen zu erstellen sind, sondern vielmehr das Zusammenspiel unterschiedlicher Wiedergabe- und Interaktionsformen innerhalb einer realen Umgebung betrachtet werden muss.

Durch den experimentellen Charakter der Projekte waren dabei insbesondere unterschiedliche Denkmodelle der einzelnen Teammitglieder zu beobachten, die zu konträren Ansätzen führten. So neigen technisch orientierte Spezialisten zu einem analytischen und strukturierten Vorgehen, das häufig durch die objektorientierte Modellierung zu einem Bottom-Up-Entwurf führt. Dahingegen neigen gestalterisch orientierte Spezialisten eher zu einer Gesamtsicht auf die zu erstellende multimediale Anwendung, so dass hier der Charakter des Kommunikationsmediums betont wird.

Die sich daraus ergebenden charakteristischen Sichten und Diskurswelten flossen in die Darstellung der Arbeit ein, so dass in den einzelnen Kapiteln durchgehend sowohl die technische als auch die gestalterische Sicht auf multimediale Anwendungen betrachtet wird. Zum einen wird dadurch eine dem Themengebiet angemessene holistische Betrachtungsweise unterstützt, zum anderen werden einzelne Themengebiete durch diese Sichtweisen in neuen Zusammenhängen dargestellt. Um den Charakter der jeweiligen Diskurswelt zu

betonen, wird je nach Gewichtung/Blickwinkel das Vokabular der jeweiligen Sichtweise verwendet. Insbesondere werden hierdurch die entgegengesetzten Facetten hervorgehoben, die sich an der Schnittstelle, die durch den „Computer“ als Präsentationsmedium gegeben ist, treffen.

Entwurfsprozesse und Ablaufumgebungen

Bei der Erstellung (verteilter) multimedialer Anwendungen sind zwei grundlegende Aspekte zu betrachten. Der erste Aspekt ist die Gestaltung, durch die das Erscheinungsbild (Mediendaten, Layout) und die Interaktionsmöglichkeiten (Verhalten) einer multimedialen Anwendung definiert werden. Der zweite Aspekt ist die Software, die als Ablaufumgebung für multimediale Präsentationen die Realisierung einer Anwendung bzw. verteilten Umgebung in geeigneter Weise unterstützen muss.

Bereits während des Entwurfsprozesses sind diese beiden Aspekte zurzeit vorherrschend. Zum einen wird ein spezieller Inhalt multimedial präsentiert, so dass Entwurfsmethoden traditioneller Medien, wie z.B. der Filmproduktion, angewendet werden, die die gestalterischen Aspekte betonen. Zum anderen entsteht ein ablauffähiges Programm bzw. eine darstellbare Beschreibung mit interaktiven Elementen, so dass die Ansätze der Softwaretechnik verwendet werden, die auf technischen Modellierungsformen basieren.

Die auf gestalterischen Aspekten aufbauenden Entwurfsverfahren basieren häufig auf Skizzen oder verbalen Beschreibungen der zu entwerfenden multimedialen Anwendung. Zwar wird hierdurch bereits ein zusammenhängendes „Bild“ der Anwendung vorgegeben, doch dient diese Beschreibung ausschließlich als lose Vorlage der eigentlichen Definition bzw. Implementierung. Bei technisch basierten Entwurfsverfahren, wie etwa bei objektorientierten oder datenbankbasierten Entwurfsparadigmen, werden zwar unterschiedliche Ansätze der Analyse und der Entwurfsmodelle durch Entwurfswerkzeuge unterstützt, doch sind diese Ansätze durch die Klassifikation bzw. analytische Dekomposition charakterisiert, so dass sie nur schwer von Spezialisten aus anderen Fachdisziplinen nachzuvollziehen sind.

Auch wenn derartige Entwurfsverfahren bereits auf verfügbare „traditionelle“ Ablaufumgebungen, wie etwa HTML-Browser, abgestimmt sind, so spielt für die Schaffung von Wahrnehmungsumgebungen neben der Gestaltung der Oberfläche ebenso die „Gestaltung“ des Präsentationsmediums als Softwareprodukt eine entscheidende Rolle. Dabei müssen sowohl räumlich verteilte Präsentationsumgebungen als auch die Integration von unterschiedlichen Ein- und Ausgabemedien berücksichtigt werden. Dies setzt aber für eine Softwareumgebung als Präsentationsmedium voraus, dass diese eine offene Modellierung unterstützt, um eine dem Präsentationskontext angemessene Gestaltung zu ermöglichen.

Es existieren zwar gerade durch komponenten- bzw. agentenorientierte Modellierungstechniken hierfür notwendige Grundkonzepte, um eine verteilte offene Ablaufumgebung zu entwickeln. Doch durch die zurzeit bekannten Ablaufumgebungen sind die Architekturstruktur und die speziellen multimedialen Aspekte zu eng miteinander verwoben. So wird durch komponentenorientierte Ablaufumgebungen, vergleichbar mit den softwareorientierten Entwurfsverfahren, bereits eine fest vorgegebene (seitenbasierte) Architekturstruktur zur Verfügung gestellt, so dass diese Ablaufumgebungen für eine offene Modellierung verteilter Präsentationen nur eingeschränkt einzusetzen sind.

Agentenorientierte Softwareumgebungen, die häufig Basis für die Umsetzung filmorientierter Entwurfsverfahren sind, betonen dagegen gerade die Eigenständigkeit einzelner (Medien-) Elemente. Allerdings werden durch die aus der Literatur bekannten Agentensysteme ausschließlich Teilaspekte multimedialer Präsentationen unterstützt, da sie entweder nur eine zu allgemein gehaltene Implementierungsgrundlage zur Verfügung stellen oder

bereits als anwendungsbezogene Implementierung lediglich als Ablaufumgebung für spezielle Anwendungsszenarien geeignet sind.

Integrierendes Konzept

Die weiter oben erwähnten Entwurfsverfahren betonen nur die aus dem jeweiligen Fachgebiet stammenden Modellierungsansätze, die aber zu kurz greifen, da sie ursprünglich für andere Problembeschreibungen entwickelt wurden. Somit unterstützen sie nicht gleichberechtigt die unterschiedlichen Blickwinkel der interdisziplinär zusammengesetzten Entwicklergruppe, die typischerweise aus Medienspezialisten, Graphikdesignern, Experten des zu präsentierenden Inhalts, Informatikern etc. gebildet wird.

Der hier vorgestellte Ansatz basiert dagegen auf Erfahrungen, die in verschiedenen Multimediaprojekten gesammelt wurden und betont die rollenspezifischen Arbeitsweisen der während einer Produktion beteiligten Spezialisten. Um die Arbeitsweisen gleichberechtigt unterstützen zu können, werden durch den so genannten subjektorientierten Entwurf unterschiedliche, voneinander unabhängige Sichten, die den einzelnen, individuellen Blickwinkeln der an dem Entwurf beteiligten Spezialisten entsprechen, integriert. Dabei wird eine lose Verbindung zwischen den zu betrachtenden Rollen unterstützt, so dass der Entwurfsprozess unterschiedliche zeitliche Abläufe zwischen den Spezialisten ermöglicht.

Durch die Ablauforganisation wird somit lediglich ein Rahmen während des Entwurfs vorgegeben, so dass der subjektorientierte Entwurf als pragmatischer Ansatz anzusehen ist, der zwischen einer festen Organisationsstruktur und einem eher experimentellen Ansatz steht. Der dabei vorgegebene Rahmen ermöglicht sowohl die Realisierung „traditioneller" multimedialer Anwendungen als auch die Gestaltung verteilter Wahrnehmungsumgebungen, indem typische Arbeitsweisen gleichberechtigt unterstützt werden, die je nach Anwendungskontext von unterschiedlichen Spezialisten ausgefüllt werden können.

Die Trennung zwischen den eigentlichen multimedialen „Inhalten" (Medien, Layout, Verhalten), die bei dem subjektorientierten Entwurf durch Sichten unterstützt werden, und der Organisation des Entwurfs, die ausschließlich einen Rahmen vorgibt, wird bei der in dieser Arbeit vorgestellten Ablaufumgebung fortgeführt. Wie bereits weiter oben erwähnt, wird durch die zurzeit existierenden Ablaufumgebungen keine Trennung zwischen eigentlicher Softwarearchitektur und multimedialem „Inhalt" vorgenommen. Hierdurch sind sie allerdings bereits auf unterschiedliche Entwurfsmodelle abgestimmt bzw. stellen lediglich eine zu allgemeine Grundlage für die Realisierung verteilter multimedialer Anwendungen dar.

In der in dieser Arbeit vorgestellten Architektur wird durch mobile Software-Agenten die Organisationsstruktur einer multimedialen Anwendung modelliert. Die multimedialen „Inhalte" und Ein- und Ausgabegeräte werden durch Komponenten realisiert. Die Software-Agenten nehmen den eigentlichen „Inhalt" auf und fungieren somit als organisierende Elemente in einer verteilten Umgebung, indem sie die Kommunikation, die Wiedergabe der Mediendaten auf unterschiedlichen Ausgabegeräten und die Steuerung durch natürliche Interaktionsformen mithilfe von Eingabegeräten übernehmen.

Durch diese Trennung werden die Abbildung des Entwurfs, die Wiederverwendbarkeit und die Erweiterbarkeit der Ablaufumgebung als verteiltes Präsentationsmedium unterstützt. Somit überbrückt die hier vorgestellte Architektur die Lücke zwischen einer zu allgemeinen Softwareumgebung und einer auf spezielle Anforderungen eingeschränkten Ablaufumgebung. Hierdurch wird erreicht, dass die Architektur eine flexible Gestaltung unterschiedlicher Wahrnehmungsumgebungen unterstützt und somit für unterschiedliche Anwendungsszenarien einsetzbar ist.

Gliederung

Wie bereits weiter oben erwähnt, lassen sich multimediale Anwendungen durch unterschiedliche Aspekte charakterisieren. Die in dieser Arbeit betrachteten Aspekte lassen sich technischen bzw. gestalterischen Sichten zuordnen, die in Kapitel 2 vorgestellt werden und zu einer Charakterisierung multimedialer Anwendungen führen.

In Kapitel 3 findet die Fortführung der vorgestellten Sichtweisen statt, da die zurzeit vorherrschenden Entwurfsverfahren auf traditionellen Entwurfsparadigmen des Software- und Medienentwurfs aufbauen und somit als Vertreter der technischen bzw. gestalterischen Sicht angesehen werden können. Die Grundlagen für die Realisierung der im Rahmen dieser Arbeit entwickelten verteilten Ablaufumgebung, d.h. eines Präsentationsmediums für die Gestaltung von Wahrnehmungsumgebungen, werden in Kapitel 4 gelegt. Besonderes Augenmerk gilt hierbei komponenten- und agentenorientierten Softwareumgebungen, die bereits in unterschiedlichen multimedialen Anwendungsgebieten zum Einsatz kommen.

Aufbauend auf den vorherigen Kapiteln wird in Kapitel 5 und Kapitel 6 das im Rahmen dieser Arbeit entwickelte integrierende Konzept für die Gestaltung und Realisierung multimedialer Umgebungen vorgestellt. Zunächst wird in Kapitel 5 der subjektorientierte Entwurf eingeführt, der die technischen und gestalterischen Sichten in gleichberechtigter Weise miteinander verbindet. Neben der Produktion „traditioneller“ Anwendungen unterstützt der subjektorientierte Entwurf die Umsetzung in eine verteilte Ablaufumgebung, deren Softwarearchitektur in Kapitel 6 vorgestellt wird.

Sowohl durch den subjektorientierten Entwurf als auch durch die verteilte Ablaufumgebung wird, wie bereits weiter oben erwähnt, ausschließlich ein Rahmen vorgegeben. Um dabei typische Vorgehensweisen während des Entwurfs, die durch dieses neue Entwurfsparadigma unterstützt werden, und unterschiedliche Realisierungen von Wahrnehmungsumgebungen aufzuzeigen, werden in Kapitel 7 verschiedene Einsatzgebiete anhand von Beispielen vorgestellt. Abschließend werden in Kapitel 8 die wichtigsten Aspekte dieser Arbeit zusammengefasst und weiterführende Fragestellungen diskutiert.

Kapitel 2

Multimedia

Multimediale Aspekte lassen sich in zahlreichen Anwendungsfeldern finden, so dass je nach Anwendungsgebiet von neuen Medien, interaktiven Medien, virtuellen Realitäten, simulierten Welten etc. die Rede ist. In der Fachliteratur werden nur einzelne Aspekte multimedialer Anwendungen betrachtet [Hünn97] [Kräm98] [Stei99] [Boll98] [Wats98] [Feld96], wodurch keine einheitliche Begriffsdefinition möglich ist. Ein erster Zugang zu dem Themengebiet Multimedia bietet sich durch die Klärung seiner beiden Wortbestandteile [Broc98]:

multi- [lat. multus „viel"], Wortbestandteil mit der Bedeutung: viel, vielfach.

Medium [lat. „Mitte"], vermittelndes Element, im Singular Bezeichnung für jede Art eines Trägers oder Übermittlers von Bedeutung, Information und Botschaft (z.B. Tafel, Buch oder Folie als Unterrichts- bzw. Anschauungsmaterial), im Plural Bezeichnung für gesellschaftliche Träger bzw. Vermittlungssysteme für Informationen aller Art.

Diese sehr allgemeine Begriffsbestimmung macht schon deutlich, dass unterschiedliche Aspekte zu betrachten sind, um sich dem Themengebiet Multimedia zu nähern. Aufbauend auf einer näheren Bestimmung des Begriffs Medium werden spezifische Charakteristika des Mediums „Computer" eingeführt. Medien sind dabei zum einen Kommunikationsmittel im Sinne von Kanälen, die die Kommunikation physisch ermöglichen. Medien sind zum anderen aber auch Zeichensysteme, die die für jede Kommunikation notwendige Semantik tragen.

Betrachtet man multimediale Systeme als neues Kommunikationsmittel im Sinne eines Kanals, ergibt sich eine Klassifikation aus Sicht der verwendeten digitalisierten „traditionellen" Medien und der unterstützten Kommunikations- bzw. Interaktionsmöglichkeiten. Diese eher auf den „Methoden und der Technik" [Andr76] beruhende Sicht ermöglicht jedoch keine Charakterisierung multimedialer Anwendungen aus gestalterischer Sicht, da sowohl die verwendete Syntax und Semantik als auch der Rezipient nicht einbezogen werden.

Bei der Untersuchung multimedialer Anwendungen als Zeichensysteme wird nach dem „Zweck und Wert" [Andr76] gefragt. Hierbei wird die Syntax und Semantik eines Mediums als Sprachsystem aufgefasst, in dem die verwendeten Stilmittel analysiert werden können. Dieser Ansatz, der die Semiotik als Sprachwissenschaft auf die Medien anwendet, ermöglicht eine Charakterisierung einzelner Genres multimedialer Anwendungen.

2.1 Kommunikationsmedien

Obwohl die Definition des Begriffs Kommunikation durch unterschiedliche Theorien erfasst wird, lassen sich fünf grundsätzliche Aspekte der Kommunikationstheorien erkennen. Eine Kommunikation wird als *Übermittlung* einer *Nachricht* zwischen *Sender* und *Empfänger* beschrieben, die eine *Reaktion* des Empfängers bewirkt [WaHi00].

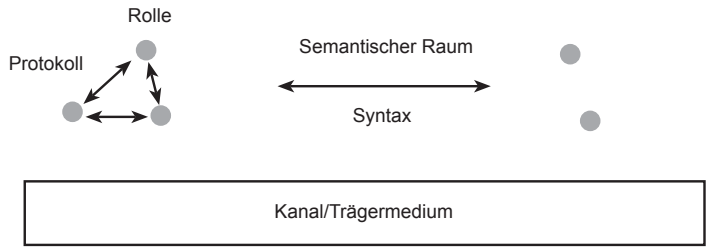


Abb. 2.1.1: Aspekte eines Mediums.

Betrachtet man das Zusammenspiel der einzelnen Aspekte eines Kommunikationsprozesses als ein Produzenten-Rezipienten-Verhältnis, so lässt sich ein Kommunikationsmedium, wie in Abb. 2.1.1 dargestellt, durch seinen Kanal, seine Syntax, seinen semantischen Raum, die Rollen der Akteure (Produzenten, Rezipienten, ...) und die existierenden Protokolle zwischen den Akteuren definieren [Schm98]:

Kanal: Der Kanal eines Mediums bildet die Grundlage für den Austausch von Informationen bzw. Daten und kann somit als Verbindung zwischen Sender und Empfänger gesehen werden. Ein solcher Kanal nimmt die zu übermittelnden Informationen auf und wird daher häufig auch als Trägermedium bezeichnet. Differenziert man die Übermittlung der Information genauer, unterscheidet man zwischen Repräsentations-, Speicher-, Übertragungs- und Präsentationsmedium [Stei99].

Als Repräsentationsmedien werden die Formate der Daten bezeichnet, wie z.B. die unterschiedlichen rechnerinternen Bild- und Textformate. Unter Speichermedien werden alle Datenträger wie Disketten, CD-ROMs etc. zusammengefasst. Übertragungsmedien unterstützen im Gegensatz zu den Speichermedien eine kontinuierliche Übertragung der Daten, wie etwa über Koaxialkabel oder Funkverbindungen. Als Präsentationsmedium werden alle Hilfsmittel und Geräte für die Ein- und Ausgabe von Informationen bezeichnet. Hier wird primär eine Unterscheidung nach Ausgabe- (Bildschirm, Lautsprecher, etc.) und Eingabemedien (Tastatur, Maus, etc.) getroffen. Berücksichtigt man die unterstützten Methoden und Techniken der Präsentation, lässt sich die Ablaufumgebung, d.h. die Software, als zentrales Präsentationsmedium multimedialer Anwendungen auffassen, die die einzelnen Aus- und Eingabemedien miteinander verbindet.

Syntax: Die durch den Kanal übermittelten Inhalte werden durch eine Syntax strukturiert, wobei die Strukturierung syntaktischen Regeln unterliegt. Man spricht dabei häufig auch von einer eigenen Sprache, die charakteristisch für ein jeweiliges Medium bzw. Genre eines Mediums ist [ToCa99]. Die Syntax eines Mediums ist dabei nicht als gegeben anzusehen. Vielmehr ist durch ein Medium eine Sprache erkennbar, die ihre eigenen Codes entwickelt. Die verwendeten Codes können sich dabei auf reale Erfahrungen bzw. auf andere Medien beziehen.

Semantischer Raum: Die Kommunikation, die durch das Medium geschieht, erfordert eine Interpretation der übermittelten Inhalte durch den Empfänger, die mit der Intention des Senders übereinstimmen sollte. Erst dadurch kann erreicht werden, dass eine erfolgreiche Kommunikation stattfindet. Man spricht in diesem Fall von der Semantik der Inhalte. Die Semantik ist weder allein im Kanal noch in der Syntax enthalten. Sie definiert eine übergeordnete Ebene, die die Referenz zur externen Welt bzw. die Interpretation der einzelnen Inhalte ermöglicht. Man spricht daher von dem semantischen Raum bzw. der semiotischen Bedeutung eines Mediums.

Ein semantischer Raum ist ein wesentlicher Bestandteil des Mediums, da durch ihn erst der Austauschmechanismus des Mediums für Informationen definiert wird. Die Semantik umfasst zum einen Begriffe und Symbole, die Grundlage für eine gemeinsame Kommunikation sind. Zum anderen bestimmt die Semantik Welten, über die unter Verwendung der Begriffe und Symbole Inhalte ausgetauscht werden können. Derartige Welten können als (abstrakte) Referenz zu Strukturen bzw. (realen) Umgebungen angesehen werden, die den Akteuren bekannt sind.

Rollen: Durch ein Medium werden die Akteure, die an diesem Medium partizipieren, definiert. Neben dem Sender und Empfänger eines Mediums sind dies im Falle der traditionellen Printmedien beispielsweise Autoren, Redakteure, Graphikdesigner, Händler, etc. Die Aufgabenprofile der Akteure werden dabei als Rollen bezeichnet. Die Rollen beschreiben somit den organisatorischen Aspekt eines Mediums.

Protokoll: Durch das Protokoll eines Mediums wird das Zusammenspiel der einzelnen Rollen beschrieben, so dass durch dieses eine Ablauforganisation festgelegt wird.

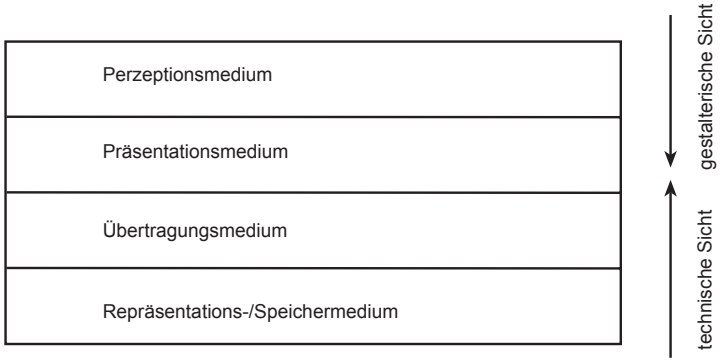


Abb. 2.1.2: Analysekategorien von Medien.

Die oben betrachteten Charakteristika eines Kommunikationsmediums umfassen unterschiedliche Aspekte, die verschiedene Ansätze einer Analyse ermöglichen. Die technischen Ansätze einer Analyse lassen sich nach [Andr76] in „Rohmaterial“ und „Methoden und Technik“ aufteilen, die gestalterischen Ansätze in „Form und Gestalt“ sowie „Zweck und Wert“. Jede dieser Kategorien charakterisiert einen anderen Aspekt des (Kommunikations-) Prozesses, der die Technik, die Produzenten und die Rezipienten verbindet. Diese Kategorien lassen sich ebenso auf das Medium „Computer“ anwenden.

Das Rohmaterial multimedialer Anwendungen sind die Repräsentations-, Speicher- sowie Übertragungsmedien. Bei der Analyse der „Methoden und Technik“ werden die Möglichkei-

ten der Präsentation betrachtet, so dass hier die Ablaufumgebungen im Vordergrund stehen. Beide Analyseansätze beschäftigen sich somit mit der Produktionsseite. Bei der Analyse des „Zwecks und Werts“ multimedialer Anwendungen steht die Beziehung zwischen Rezipient und Anwendung im Vordergrund. Die Analyse der „Form und Gestalt“ kann als entgegengesetzte Facette der Analyse der „Methoden und Technik“ angesehen werden, wobei die erste einen theoretischen und die zweite einen praktischen Ansatz darstellt. Somit treffen sich die Kategorien, wie in Abb. 2.1.2 dargestellt, an der Schnittstelle, die durch das Präsentationsmedium gegeben ist.

Um eine systematische Betrachtung multimedialer Anwendungen in dieser Arbeit zu ermöglichen, werden zunächst im Anschluss die beiden charakteristischen Blickwinkel der technischen und gestalterischen Sicht eingeführt. Hierzu werden neben der aus der Informatik bekannten technischen Sichtweise die aus der Medientheorie bekannte Analyse eines Kommunikationsmediums als Zeichensystem herangezogen. Um die Analyse von Wahrnehmungsumgebungen als Zeichensystem zu ermöglichen, wird hierzu in dieser Arbeit die Theorie der Semiotik im Hinblick auf (verteilte) multimediale Anwendungen erweitert.

Erst hierdurch wird es möglich, die beiden Sichtweisen in einer einheitlichen Weise durchgängig in dieser Arbeit zu berücksichtigen und somit die Analogien beider Diskurswelten gegenüberzustellen. Dabei werden die oben beschriebenen Aspekte eines Kommunikationsmediums betrachtet, so dass neben den technischen Aspekten insbesondere gestalterische und organisatorische Aspekte berücksichtigt werden. Gerade durch den in dieser Arbeit entwickelten Ansatz wird dabei eine einheitliche holistische Betrachtung des Kommunikationsmediums „Computer“ unterstützt.

2.2 Multimedia aus technischer Sicht

Bei der Analyse der „Rohmaterialien“ und „Methoden und Technik“ werden die zur Verfügung stehenden Elemente betrachtet, die für den Entwurf multimedialer Anwendungen verwendet werden. Somit handelt es sich um einen konstruktiven Ansatz, durch den multimediale Anwendungen aufgrund der technischen Eigenschaften eines Systems charakterisiert werden.

Zum einen spielen bei der Analyse die verwendeten unterschiedlichen Medientypen eine wichtige Rolle, die ja gerade den Begriff „Multimedia“ prägen. Zum anderen werden während des Entwurfs die funktionalen Beziehungen zwischen den Elementen einer multimedialen Anwendung definiert, die als Sender bzw. Initiator oder Empfänger bzw. Aktor einer Beziehung auftreten können. Durch dieses Sender-Empfänger-Modell lassen sich sowohl die Beziehungen zwischen den einzelnen Medien als auch die Interaktionsmöglichkeiten der Benutzer multimedialer Anwendungen beschreiben.

2.2.1 Medientypen

Wichtigster Bestandteil multimedialer Präsentationen ist die Integration unterschiedlicher Medientypen für die gezielte Darstellung der zu Grunde liegenden Informationen. Die einzelnen Mediendaten werden dabei durch Medienverwalter bzw. -elemente gekapselt, wobei man zwischen elementaren und komplexen Medienverwaltern unterscheidet.

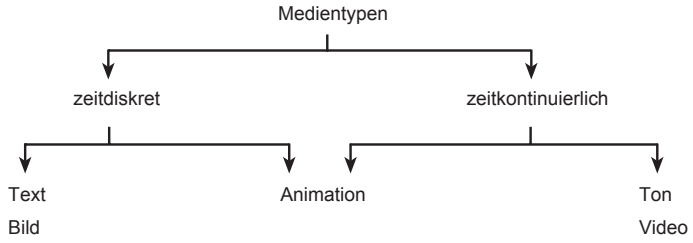


Abb. 2.2.1.1: Klassifikation von Medientypen.

Elementare Medienverwalter verwalten ausschließlich einzelne Mediendaten, wie Text, Bild, Video etc. Komplexe Medienverwalter setzen sich aus elementaren und/oder komplexen Medienverwaltern zusammen, so dass unterschiedlicher Mediendaten als zusammenhängende Einheit in eine multimediale Anwendung integriert werden können. Da komplexe Medienverwalter auf elementare Medienverwalter zurückzuführen sind, werden für die Klassifikation multimedialer Anwendungen lediglich die verwendeten Mediendaten durch ihren Medientyp „quantitativ“ charakterisiert. Dabei unterscheidet man zwischen zeitdiskreten und zeitkontinuierlichen Medientypen:

Zeitdiskrete Medientypen: Einige Mediendaten wie Text und Graphik sind zeitunabhängig. Informationen in diesen Mediendaten bestehen ausschließlich aus einer Folge einzelner Elemente oder aus einem Kontinuum ohne Zeitkomponente. Derartige Mediendaten werden als zeitunabhängige bzw. zeitdiskrete Medientypen charakterisiert.

Zeitkontinuierliche Medientypen: Die Werte anderer Mediendaten wie Ton und Bewegtbild verändern sich über die Zeit kontinuierlich. Die Information steckt somit nicht nur in einem einzelnen Wert, sondern auch im Zeitpunkt des Auftretens. Derartige Mediendaten werden als zeitabhängige bzw. zeitkontinuierliche Medientypen bezeichnet.

Wie in Abb. 2.2.1.1 dargestellt, lassen sich somit die in multimedialen Anwendungen verwendeten Mediendaten klassifizieren. Eine hybride Stellung nehmen dabei allerdings Animationen ein, die je nach verwendetem Medienformat bereits eine zeitliche Information tragen bzw. als einzelne Bilder einer beliebig ansteuerbaren Bilderfolge gleichen [Stei99] [GiBe98].

2.2.2 Beziehungen zwischen Medienverwaltern

Die Charakterisierung des Verhaltens einer multimedialen Anwendung lässt sich auf die funktionalen Beziehungen zwischen den einzelnen Medienverwaltern zurückführen. Die Kapselung der einzelnen Mediendaten ermöglicht es dabei, unabhängig von den Medientypen über Medienverwalter zu argumentieren, die als Sender und/oder Empfänger einer Beziehung auftreten können. Eine Beziehung wird dabei durch die beiden Aspekte *<Ereignistyp, Aktionstyp>* eines auslösenden Ereignisses des Senders und einer auszuführenden Aktion des Empfängers beschrieben.

In Abb. 2.2.2.1 sind die Ereignis- und Aktionstypen zusammengefasst, durch die beide Aspekte einer Beziehung charakterisiert werden. Mit Ausnahme der interaktiven Beziehungsaspekte können die unterschiedlichen Typen sowohl als Ereignis als auch als Aktion auftreten, wobei in den meisten Fällen Ereignis- und Aktionstyp unterschiedlich sind. Im folgenden werden die einzelnen Beziehungsaspekte näher betrachtet [Bole98].

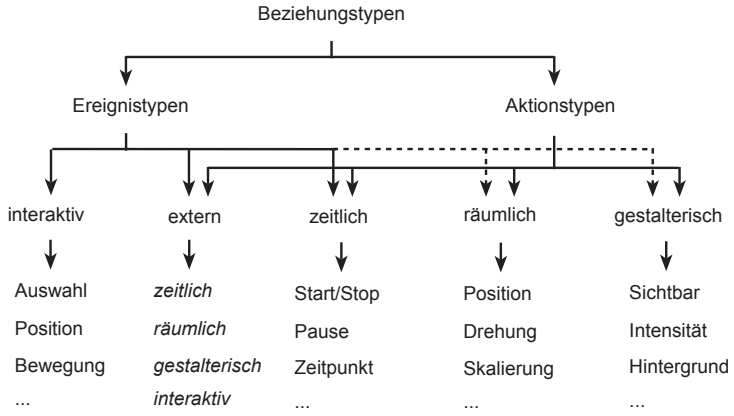


Abb. 2.2.2.1: Klassifikation durch Beziehungen zwischen Medienverwaltern.

Interaktive Beziehungsaspekte: Interaktive Aspekte treten ausschließlich als Ereignistypen auf. Über die Auslösung von Interaktionsereignissen kann der Benutzer eine multimediale Anwendung steuern. Hierzu stehen ihm unterschiedliche Eingabemedien wie Tastatur, Maus etc. zur Verfügung. In den meisten multimedialen Ablaufumgebungen, wie *Director* [BaCM98], *iShell* [iShe01], *RealPlayer* [Goet00] etc., kann jeder Medienverwalter Mausereignisse registrieren und kann somit als Interaktionselement verwendet werden. Andere Modelle, wie etwa die *Virtual Reality Modeling Language (VRML)* [CaBe97], unterscheiden hingegen zwischen Interaktions- und Medienelementen. Hier stehen lediglich so genannte Sensorobjekte als Interaktionselemente dem Benutzer zur Verfügung.

Externe Beziehungsaspekte: Im Gegensatz zu den anderen Beziehungsaspekten können durch externe Beziehungsaspekte eigenständige Applikationen, wie Datenbanken, Bild- oder Spracherkennung, in eine multimediale Anwendung funktional eingebunden werden. Dabei können externe Programme sowohl Ereignisse auslösen als auch auf Ereignisse durch entsprechende Aktionen reagieren. Die unterstützten Ereignis- bzw. Aktionstypen sind durch das eingebundene Programm definiert.

Zeitliche Beziehungsaspekte: Bei diesen Aspekten wird die zeitliche Komponente eines Medienverwalters beschrieben, so dass sie nur bei zeitkontinuierlichen Medientypen als Aktionstyp auftreten. Allerdings werden zeitliche Aspekte häufig als Ereignistypen für die Beeinflussung zeitdiskreter Medientypen verwendet, um z.B. einen Text zu einem festgelegten Zeitpunkt eines Films anzuzeigen oder eine Folge von Bildern nacheinander darzustellen.

Räumliche Beziehungsaspekte: Räumliche Aspekte beschreiben die geometrischen Eigenschaften eines Medienelements bzw. der verwalteten Mediendaten. Diese Aspekte treten hauptsächlich als Aktionstypen auf, um die Position, Ausrichtung oder Größe einzelner Medienelemente zu beeinflussen. In dreidimensionalen Umgebungen können räumliche Aspekte häufig auch für akustische Daten definiert werden, so dass eine räumliche Klangwirkung erzeugt werden kann.

Gestalterische Beziehungsaspekte: Gestalterische Aspekte beschreiben die Darstellung bzw. Wiedergabe der Mediendaten und lassen sich ebenso wie räumliche Aspekte fast nur als Aktionstypen in multimedialen Anwendungen finden. So lässt sich z.B. über

die Intensität die Transparenz visueller Mediendaten bzw. die Lautstärke akustischer Mediendaten beeinflussen.

Durch die unterschiedlichen Beziehungstypen lässt sich das Verhalten der einzelnen Medienverwalter beschreiben, die auf interne Änderungen anderer Verwalter bzw. auf externe Interaktionsereignisse, die von dem Benutzer ausgelöst werden, reagieren. Bewirkt ein Ereignis eine unmittelbare Reaktion, so spricht man von einer direkten Beziehung. Dahingegen sind die Auswirkungen auf den Empfänger bei indirekten Beziehungen erst zu einem späteren Zeitpunkt zu beobachten. So können z.B. während der Interaktion des Benutzers verschiedene Ereignisse auf eine spätere Aktion Einfluss haben.

2.2.3 Klassifikation multimedialer Anwendungen

Die technische Klassifikation multimedialer Anwendungen abstrahiert von den einzelnen Mediendaten und den unterschiedlichen Beziehungen zwischen den Elementen, so dass keine nähere Aussage über die Gestaltung der Anwendungen getroffen wird. Vielmehr beschränkt sich die Klassifikation ausschließlich auf „quantitative“ Aspekte.

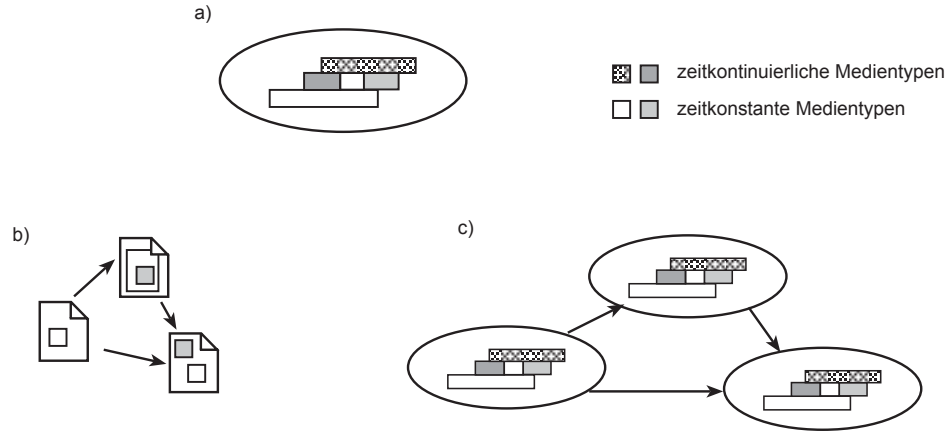


Abb. 2.2.3.1: Charakteristische Merkmale von multimedialen Präsentationen (a), Hypertexten (b) und Hypermedia (c).

Aus dieser „quantitativen“ Sicht werden multimediale Anwendungen nach ihren verwendeten Medientypen und den realisierten interaktiven Beziehungsaspekten klassifiziert. In Abb. 2.2.3.1 sind die sich daraus ergebenden Klassen multimedialer Anwendungen zusammengefasst [Stei99]:

Multimediale Präsentationen: In multimedialen Präsentationen werden die verschiedenen Medientypen (Text, Bild, Bewegtbild, Ton, 3D-Welten, ...) integriert. Mindestvoraussetzung ist dabei das Zusammenwirken eines zeitdiskreten und eines zeitkontinuierlichen Medientyps. Multimediale Präsentationen realisieren allerdings keine Interaktionsbeziehung, so dass keine Steuerung der Präsentation durch den (Be-)Nutzer möglich ist.

Hypertext-Anwendungen: In diesen Anwendungen kommen lediglich zeitdiskrete Mediendaten wie Text und Bild zum Einsatz, die in den meisten Fällen durch komplexe Medienverwalter als „Seiten“ bzw. Dokumente für den Benutzer repräsentiert werden. Die

Interaktionsbeziehungen beschreiben die Verbindungen, die so genannten „Hyperlinks“, zwischen den einzelnen Dokumenten. Durch eine derart verknüpfte (Dokumenten-) Struktur kann der Benutzer somit zwischen den Dokumenten navigieren.

Hypermedia-Anwendungen: Hypermedia-Anwendungen vereinen die Eigenschaften von Hypertext-Anwendungen und multimedialen Präsentationen, d.h. es werden sowohl zeitdiskrete als auch zeitkontinuierliche Mediendaten eingesetzt. Neben der multimedialen Präsentation realisieren Hypermedia-Anwendungen Interaktionsbeziehungen, so dass der Benutzer die Präsentation beeinflussen kann.

Häufig wird der Begriff *Multimedia* als Oberbegriff von Hypertext-, Hypermedia-Anwendungen und multimedialen Präsentationen benutzt, wobei je nach betontem Aspekt auch von multimedialen Dokumenten, interaktiven multimedialen Anwendungen bzw. multimedialen Inhalten gesprochen wird.

2.3 Multimedia aus gestalterischer Sicht

Die im vorherigen Abschnitt betrachtete „quantitative“ Charakterisierung multimedialer Anwendungen basiert nur auf der Typisierung „traditioneller“ Medien und der funktionalen Beziehung zwischen diesen Medientypen. Allerdings lässt sich durch diese Sicht der Charakter multimedialer Anwendungen nicht erfassen, da hier ein eher „qualitativer“ Zugang erforderlich wird, wo nach dem „Zweck und Wert“ und der „Form und Gestalt“ gefragt wird. Wie in Abschnitt 2.1 erwähnt, steht dabei die Beziehung zwischen Rezipient und multimedialer Anwendung im Vordergrund, d.h. wie findet die Kommunikation zwischen beiden statt.

Die Untersuchung eines Mediums als Kommunikationssystem baut auf der aus den Sprachwissenschaften bekannten Semiotik auf. Dabei wird die Kommunikation als Zeichensystem aufgefasst, unabhängig davon, ob sie zwischen Personen, zwischen Mensch und Maschinen, in einer Institution oder mittels eines Massenmediums abläuft. Damit greift sie wesentlich über die naturwissenschaftlichen Modelle von Sender und Empfänger hinaus. Um dabei die Theorie der Semiotik auf verteilte multimediale Anwendungen anwenden zu können, werden in dieser Arbeit neue Kategorien für die Charakterisierung räumlicher und zeitlicher Strukturen sowie unterschiedliche Interaktionskategorien eingeführt.

Im Gegensatz zu der Analyse der „Methoden und Technik“ basiert die Untersuchung multimedialer Anwendungen als Zeichensystem nicht auf konstruktiven Modellen, die eine multimediale Anwendung aufgrund der Möglichkeiten eines Systems beschreiben. Hier wird vielmehr der umgekehrte Weg beschritten, d.h. durch eine multimediale Anwendung lässt sich ein (Zeichen-)System erkennen, das von dem Rezipienten bewusst oder unbewusst interpretiert wird. Wie in Kapitel 5 noch näher gezeigt wird, werden jedoch durch den subjektorientierten Entwurf so genannte rollenspezifische Sichten eingeführt, die als konstruktive Methoden der nachfolgend vorgestellten semiotischen Kategorien dienen.

2.3.1 Mediendaten

Die Zeichen eines Kommunikationssystems werden in der Semiotik als Beziehungen zwischen Signifikant (Bezeichnendes) und dem Signifikat (Bezeichnetes) interpretiert, d.h. ein Zeichen wird von dem Rezipienten „gelesen“ und eine semiotische Struktur als konstruierte "Realität" projiziert.

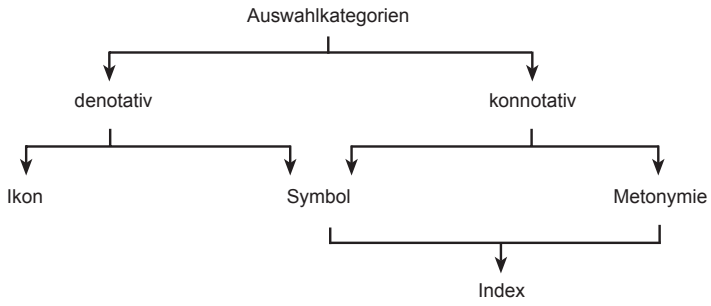


Abb. 2.3.1.1: Auswahlkategorien von Medien.

Bei der Interpretation der Zeichen findet ein dynamischer Prozess statt, bei dem den Zeichen eine direkte (denotative) bzw. assoziierte (konnotative) Bedeutung zugeordnet wird. Im Fall eines Passbildes besteht z.B. eine denotative Bedeutung, wohingegen eine schriftliche Beschreibung der abgebildeten Person als eher konnotativ zu bezeichnen ist. Die Bedeutung wird somit durch die Auswahl einer bestimmten Darstellung assoziiert, die von dem Rezipienten, bewusst oder unbewusst, mit anderen möglichen Darstellungen verglichen wird. Man spricht dabei von so genannten Auswahlkategorien eines Zeichensystems. In Abhängigkeit von der Beziehung zwischen Signifikant und Signifikat unterscheidet man, wie in Abb. 2.3.1.1 dargestellt, zwischen unterschiedlichen Zeichentypen:

Ikon: Zeichen, dessen Beziehung zum Signifikat auf einem Abbildungsverhältnis, d.h. auf Ähnlichkeiten, beruht. Dies betrifft z.B. optische Ähnlichkeiten bei Piktogrammen etc. und akustische Ähnlichkeiten bei Geräuschen.

Index: Zeichen, die in einem Folge-Verhältnis zum Signifikat oder Gemeinten stehen und somit Rückschlüsse auf etwas anderes, wie z.B. Ursache oder Grund, zulassen. So kann etwa eine Abbildung einer lachenden Person als indexikalisches Zeichen für Freude stehen oder ein Dialekt bzw. die Stimmqualität Rückschlüsse auf eine bestimmte Herkunft bzw. auf das Alter eines Sprechers ermöglichen.

Symbol: Zeichen, dessen Beziehung zum Signifikat weder auf einem Folge-Verhältnis (indexikalisches Zeichen) noch auf Ähnlichkeit (ikonisches Zeichen), sondern auf (kulturellen) Vereinbarungen beruht. So sind z.B. Laut- und Schriftzeichen der menschlichen Sprache symbolische Zeichen.

Metonymie: Zeichen, dessen Beziehung zum Signifikat sich durch ein assoziiertes Detail oder eine assoziierte Vorstellung aufbaut. So kann in der Literatur vom König (und der Idee des Königtums) als der Krone gesprochen werden.

Die einzelnen Medientypen einer multimedialen Anwendung sind durch unterschiedliche Zeichentypen charakterisiert. So hat z.B. ein Film eine denotative Wirkung, da er eine große Annäherung an die Realität vermitteln kann. Die Bilder sind ikonische Zeichen, die in direkter Beziehung zu realen Objekten stehen. Ein Film ist somit in der Lage, ein präziseres Wissen weiterzugeben, als dies die geschriebene Sprache im allgemeinen kann. Dahingegen ist die geschriebene Sprache besser für eine Auseinandersetzung mit der nichtkonkreten Welt der Ideen und Abstraktionen geeignet.

Auch wenn die denotative Wirkung protokollierender Medien, wie etwa bei Filmen oder bei Bildern, diesen inhärent ist, besitzen sie ebenso konnotative Wirkung durch indexikalische

Zeichen und Metonymien [Fein81] [Mona00]. Auch in multimedialen Anwendungen findet man derartige Zeichen. So spiegeln z.B. virtuelle Welten neben der sicherlich vorhandenen denotativen Wirkung eine symbolische Welt wider [Kräm88], die häufig literarische bzw. filmische Codes übernimmt [Wenz99].

Durch die Transformation der „traditionellen" Medien in digitale Medien ist ebenso ein Wandel festzustellen, der einen gleichberechtigten semiotischen Gebrauch der Medien hervorruft [Sand97]. Dieser Aspekt, der unmittelbar mit der Möglichkeit der Interaktion zusammenhängt, wird in Abschnitt 2.3.3 genauer betrachtet.

2.3.2 Form

Bei der Interpretation der einzelnen Medien kann die Bedeutung, wie im vorherigen Abschnitt gezeigt, durch die gewählte Darstellung repräsentiert werden. Allerdings hängt die Bedeutung nicht ausschließlich vom Vergleich einer bestimmten Darstellung mit möglichen anderen Darstellungen ab, sondern ebenso von dem Vergleich mit anderen tatsächlich vorkommenden Medien. Man spricht hierbei von Aufbaukategorien eines Zeichensystems.

Durch das Zusammenspiel unterschiedlicher Medien werden diese in Beziehung zueinander gesetzt, so dass nicht nur die einzelnen Medien einer multimedialen Anwendung eine semiotische Deutung hervorrufen, sondern vor allem die Anordnung eine (syntagmatische) Konnotation hervorruft. Es geht hierbei um die Frage, wie man die Medien präsentiert, um eine Wahrnehmungsumgebung in Abhängigkeit des Rezeptionskontexts zu gestalten. Während die Präsentationsformen traditioneller Medien bereits durch die materiellen Eigenschaften der Medien eingeschränkt sind, kann die Schnittstelle multimedialer Umgebungen frei gestaltet werden. Diese Offenheit der Gestaltung einer Wahrnehmungsumgebung ist somit ein wesentliches Charakteristikum multimedialer Anwendungen.

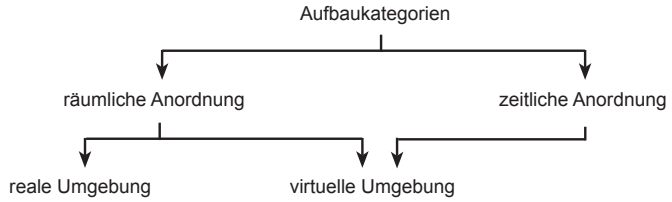


Abb. 2.3.2.1: Aufbaukategorien multimedialer Anwendungen.

Aufbaukategorien können, wie in Abb. 2.3.2.1 dargestellt, im Raum wie auch in der Zeit operieren. Die Anordnung ist sowohl durch die Gestaltung der realen Umgebung als auch durch die Gestaltung der virtuellen Umgebung charakterisiert. Die reale Umgebung, in der sich der Rezipient befindet, ist durch das Zusammenwirken einzelner Präsentationsmedien bestimmt. So kann z.B. durch eine räumliche multimediale Installation, die traditionelle Trägermedien integriert und durch die Erfahrungen des Rezipienten mit diesen Medien direkt eine semiotische Bedeutung imaginiert werden.

Die Aufbaukategorien der virtuellen Umgebung ermöglichen die räumliche und zeitliche Anordnung der einzelnen Medien. Die dadurch definierte Struktur greift häufig auf kulturell abgeleitete Codes zurück, die auch in anderen Medien auftreten [ToCa99]. Aber erst durch die Aufhebung der Linearität und festen Syntax der „traditionellen" Medien bildet das räumliche und zeitliche Zusammenspiel der Medien innerhalb einer Wahrnehmungsumgebung den

spezifischen Code multimedialer Anwendungen [Pazz99]. Dieser dadurch erzielte Wechsel von Aktivität und Passivität führt unmittelbar zu der semiotischen Bedeutung der Interaktion.

2.3.3 Interaktion

Der Wechsel zwischen Aktivität und Passivität ist insbesondere durch die Interaktivität gekennzeichnet, die den Rezipienten zum (Be)Nutzer multimedialer Anwendungen macht. Die semiotische Bedeutung entsteht nicht nur durch das „Lesen" einer vorgefertigten linearen Struktur, wie in den „traditionellen" Medien. Der Benutzer hat vielmehr verschiedene Auswahlmöglichkeiten, die zu einer Vielfalt an linearen Strukturen führt [Wenz99]. Die semiotische Struktur von multimedialen Anwendungen bezieht sich damit im Wesentlichen auf die Frage der Bedeutung der Interaktionsstrukturen.

Die Interaktionsstrukturen lassen sich präzisieren, wenn man sie ebenso wie die Medien als (Meta-) Zeichensystem betrachtet. Ein Zeichen wird dabei wiederum als Beziehung zwischen Signifikant und Signifikat interpretiert. Die Bedeutung eines Zeichens wird hier durch die tatsächliche „programmierte" Beziehung zu anderen Zeichen repräsentiert. Stehen Signifikant und Signifikat in einer direkten Beziehung, spricht man von einer beschreibenden (deskriptiven) Bedeutung. Im Fall einer indirekten Beziehung ist die Bedeutung erzählend (narrativ).

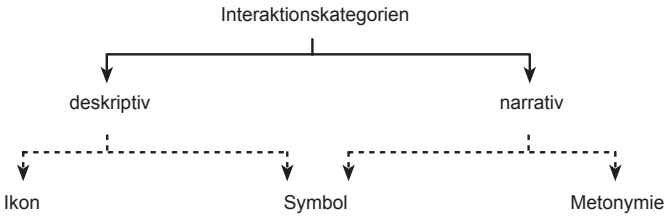


Abb. 2.3.3.1: Interaktionskategorien multimedialer Anwendungen.

Die deskriptive Interaktionskategorie, die Suchfunktionen, Menüs, Verweise etc. umfasst, ist eher als räumliche Verbindung zwischen Zeichen aufzufassen. Dahingegen repräsentiert die narrative Interaktion eher zeitliche Verbindungen. Dies führt unmittelbar zu der Dramaturgie multimedialer Anwendungen, die nicht auf dem Prinzip der Linearität „traditioneller" Medien beruht [Heid99]. In multimedialen Anwendungen wird ausschließlich ein (flexibler) Rahmen vorgegeben, innerhalb dessen durch die Interaktion des Benutzers unterschiedliche dramaturgische Formen „aufgerufen" werden können.

Durch die Interaktionskategorien wird deutlich, dass Zeichen multimedialer Anwendungen eine Doppelbedeutung besitzen können. Sie repräsentieren jedoch nicht nur wie bei den „traditionellen" Medien denotative bzw. konnotative Bedeutung. Vielmehr wird durch die Interaktionsfunktion eines Zeichens, wie in Abb. 2.3.3.1 dargestellt, eine zusätzliche Bedeutungsebene eingeführt. So werden z.B. in Hypertexten Zeichen unterschiedlicher Auswahlkategorien in ikonischer Weise verwendet. Sowohl bildliche als auch schriftliche Zeichen verweisen deskriptiv auf andere Zeichen. Umgekehrt werden bildliche Darstellungen von dem Benutzer „gelesen", deren symbolische bzw. indexikalische Bedeutung durch die Interaktionsfunktion repräsentiert wird [Sand97] [Wenz99].

2.3.4 Genres multimedialer Umgebungen

Da die Entwicklung des neuen Mediums „Computer“ noch am Anfang steht, ist die Gestaltung multimedialer Anwendungen einem stetigen Wandel unterworfen. Es ist sicherlich nicht zu verkennen, dass die einzelnen „traditionellen“ Medien, die durch die Digitalisierung in einheitlicher Weise durch den Computer abrufbar sind, dieses neue Medium beeinflussen. Allerdings zeichnen sich charakteristische Merkmale einiger multimedialer Genres bereits ab.

Aus Sicht der Semiotik lassen sich multimediale Anwendungen von anderen Medien durch ihr verwendetes Zeichensystem und die sich daraus ergebenden semantischen Codes unterscheiden. Zwar können multimediale Anwendungen die Codes der „traditionellen“ Medien übernehmen, aber erst durch das Zusammenwirken von Auswahl-, Aufbau- und Interaktionskategorien können für multimediale Anwendungen neue Ausdrucksformen entstehen. Multimediale Genres lassen sich somit durch das Trippel <*Auswahlkategorie, Aufbaukategorie, Interaktionskategorie*> charakterisieren.

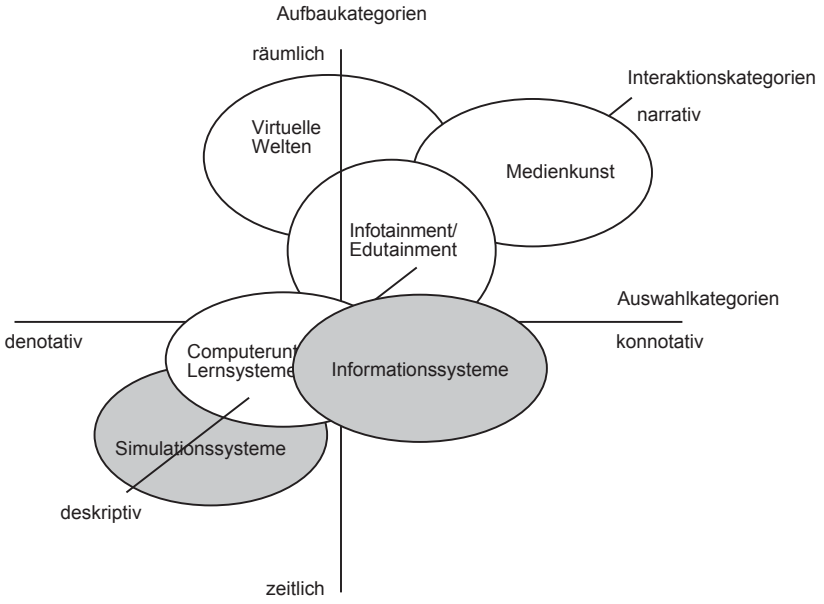


Abb. 2.3.4.1: Hauptkategorien multimedialer Genres.

Insbesondere bei der Betrachtung von Wahrnehmungsumgebungen spielen neben den bekannten Auswahl- und Aufbaukategorien der traditionellen Medien neue gestalterische Formen für die Präsentation und Interaktion eine wesentliche Rolle. Hieraus ergibt sich die Notwendigkeit, neue experimentelle Wege zu beschreiten, um das Spannungsfeld zwischen realer und virtueller Umgebung zu gestalten. Von besonderem Interesse scheinen in diesem Zusammenhang sowohl verteilte Präsentationsformen als auch narrative Interaktionsformen zu sein. Wie in Kapitel 7 anhand von Beispielen noch näher gezeigt, kann die Schaffung von Wahrnehmungsumgebungen dabei für unterschiedliche multimediale Genres erfolgen.

In Abb. 2.3.4.1 sind die zurzeit erkennbaren Hauptkategorien multimedialer Genres zusammengefasst. Da die Übergänge zwischen den einzelnen Genres zum Teil fließend sind,

werden in der Abbildung ausschließlich die vorherrschenden Merkmale der einzelnen Kategorien berücksichtigt.

Informationssysteme: Die Gestaltung von Informationssystemen basiert auf den entsprechenden traditionellen Genres wie Zeitungen, Magazinen, Nachschlagewerken etc. Die charakteristischen Codes von Medien und Form sind bestimmendes Merkmal der interaktiven Pendants. Sie ermöglichen somit die (kulturelle) Einordnung des Informationskontexts, indem sie bekannte Layout-Metaphern der traditionellen Medien übernehmen [ToCa99].

Auch die unterstützten Interaktionsformen folgen traditionellen deskriptiven Interaktionskategorien. So sind Inhaltsverzeichnisse als Menüs, Stichwortverzeichnisse als Suchfunktionen und Verweise als Verbindungen zu anderen „Seiten“ realisiert. Erst durch den Online-Zugriff auf Informationen bilden sich neue Charakteristika interaktiver Informationssysteme [ShWa98]. Dies ist zum einen auf die gleichwertige Bereitstellung von Informationszugang und Information zurückzuführen, wodurch ein zeit- und ortsunabhängiger Zugang unterstützt wird. Zum anderen kann der Nutzer mit den Informationslieferanten und anderen Nutzern direkt bzw. indirekt über das Informationssystem in Kontakt treten. Hierdurch wird der so genannte personalisierte Informationszugang unterstützt [amaz01] [yaho01]. Informationsmedien fungieren somit nicht mehr als Massenmedien, sondern als individuelle bidirektionale Informationsressourcen.

Simulationssysteme: Simulationssysteme werden in der Wissenschaft, Ausbildung, im Bereich Entertainment und in den kombinierten Genres Info- bzw. Edutainment eingesetzt. Ziel derartiger multimedialer Anwendungen ist es, durch die Schaffung einer Wahrnehmungsumgebung ein zu Grunde liegendes Modell bzw. Regelwerk dem Benutzer zugänglich zu machen. Hierdurch steht dem Benutzer eine virtuelle Experimentierumgebung zur Verfügung, in der er die modellierten Prozesse, die in der Realität zu klein, zu groß, zu langsam, zu schnell, zu kostspielig, zu gefährlich, etc. sind, beobachten und beeinflussen kann [Stei99].

Für die Darstellung der Modelle werden ikonische Zeichen verwendet, die in direktem Bezug zu den modellierten „Prozessen“ stehen [AlPr99] [Sied01]. Auch wenn für die Visualisierung abstrakter wissenschaftlicher Modelle, wie z.B. bei der Simulation von Petrinetzen, symbolische Zeichen verwendet werden, haben diese durch ihre standardisierte Darstellung eine denotative Wirkung. Diese wird durch die deskriptive Funktion der Zeichen verstärkt, indem sie einzelne Funktionen des Modells widerspiegeln bzw. die direkte Beeinflussung des Modells unterstützen. So kann der Benutzer über diese Ikons das Verhalten beeinflussen bzw. neue Modelle erzeugen.

Das ikonische Zeichensystem des Wahrnehmungsraums wird häufig auch durch die Fortsetzung in der realen Umgebung unterstützt. So werden z.B. für die Ausbildung von Piloten Flugsimulatoren in hydraulisch steuerbare reale Cockpits integriert, die durch reale Steuerinstrumente zu beeinflussen sind. Auch im Bereich Entertainment wird durch den Einsatz entsprechender Steuerinstrumente das ikonische Zeichensystem in der realen Umgebung des Benutzers weitergeführt.

Computerunterstützte Lernsysteme: Der Übergang von einem rein formal analytischen Vorgehen hin zu einem experimentellen Vorgehen bei der Erklärung von komplexen Prozessen findet sich auch in computerunterstützten Lernsystemen wieder. Das lange Zeit bei den Lerntheorien vorherrschende behavioristische Modell, in dem das Lernen als

eine Reiz-Reaktions-Kausalität aufgefasst wird, wird zunehmend durch ein kognitions- und handlungspsychologisches Modell ergänzt [Ott97] [Stei99]. Liegen beiden Ansätzen unterschiedliche Annahmen über das Lernen zu Grunde, so ist gerade die Verflechtung beider Grundkonzeptionen für das didaktische Design von computerunterstützten Lernsystemen von Bedeutung [Kerr98].

Das kognitionspsychologische Modell wird durch die deskriptive Verbindung einzelner Lerninhalte umgesetzt, in denen die Aufbaukategorien bereits eine kognitive Struktur der Lerninhalte widerspiegeln. Durch die Kombination unterschiedlicher semiotischer Zeichensysteme werden dabei die aus der Lernpsychologie bekannten Wahrnehmungsarten durch sprachliche und bildliche Informationen unterstützt [ObWe98] [Vest98]. Die Lerninhalte können dabei durch Assoziations-, Anmerkungs- und Kommentarnetze ergänzt sein, um einen gemeinsamen semantischen Raum zu erzeugen [Lévy90]. Durch die Verflechtung der kognitiven Struktur des Lerninhalts mit Animationen und Simulationen können Sachverhalte und dynamische Vorgänge veranschaulicht werden. Dieser direkte Zugang ermöglicht es dem Lernenden handlungsorientiert sein Wissen zu überprüfen bzw. zu vertiefen [Stei99] [ShWa98].

Neben der deskriptiven Interaktion werden in computerunterstützten Lernsystemen häufig „geführte" Touren durch den Lerninhalt angeboten [ABFP99] [Nykä97] [ObWe98], die einen narrativen Charakter aufweisen [Wats98]. Diese narrative Form der Wissensvermittlung wird in einigen Systemen durch künstliche personalisierte Tutoren unterstützt [RiJo97] [JRSM98] [LCKB97]. Steht der narrative Charakter einer Anwendung im Vordergrund fällt diese in den Bereich Edutainment.

Virtuelle Welten (Entertainment): Computersimulationen und Computerspiele sind durch ein zu Grunde liegendes Regelwerk vergleichbar, das das Verhalten eines Modells beschreibt. Allerdings spielt bei der Schaffung virtueller Welten die narrative Beeinflussung des Modell eine wesentlich größere Rolle. Das Regelwerk beschreibt die Strukturen, die Funktionen und die Ziele als Modell des Geschehens, welches entlang der narrativen Zeitlinie entwickelt wird [Wenz99]. Der Spieler interagiert mit der virtuellen (Spiel-)Umgebung, und konstruiert damit seine eigene „Erzählung".

Die Gestaltung des Raums ist die zentrale Kategorie in Computerspielen. In den virtuellen Welten sind Informationen ebenso wie Erfahrungen an Orte gebunden. Der Spieler „bewegt" sich in der virtuellen Welt und entschlüsselt sie. Die zu entschlüsselnde Bedeutung wird dabei sowohl durch die Auswahlkategorien als auch durch die Interaktionskategorien repräsentiert, die auf bekannten Codes aus der Literatur und dem Film basieren [Wenz99].

Auch wenn die virtuellen Welten durch die verwendeten Codes eine konnotative Bedeutung repräsentieren, stellen sie gleichzeitig auch einen direkten Bezug zu realen Umgebungen her. Dies geschieht zum einen durch die sicherlich vorhandene ikonische Bedeutung der virtuellen Welt, zum anderen durch die Darstellungsperspektive, durch die der Spieler als handelnder „Charakter" in der virtuellen Welt agiert. Virtuelle Welten stellen somit auch eine mögliche Welt mit ihrer inhärenten Eigenschaft des „Als-Ob" dar.

Die verschwimmenden Grenzen zwischen realer und virtueller Welt werden besonders deutlich bei so genannten Online-Rollenspielen wie *Ultima Online* [Ulti01], in denen die (Mit-)Spieler durch einen selbstkreierten virtuellen „Charakter" dargestellt werden. Die Bedeutung der Charaktere wird zwar auch hier durch symbolische bzw. indexikali-

sche Zeichen repräsentiert, allerdings wird der narrative Charakter der virtuellen Umgebung durch das Verhalten der einzelnen Spieler bestimmt. Hierdurch entsteht eine (Ersatz-) Welt, die ihre eigene, nicht durch ein festes Regelwerk bestimmte Dynamik entwickelt [Plog00].

Infotainment/Edutainment: Multimediale Anwendungen im Bereich Info- bzw. Edutainment verbinden die narrative Form von Computerspielen mit den entsprechenden Inhalten, d.h. den zu vermittelnden Informationen bzw. Lehrinhalten. Entsprechend zu Computerspielen entwickelt der Benutzer durch die (narrative) Interaktion mit der virtuellen Umgebung eine eigene „Erzählung".

Der narrative Charakter derartiger Anwendungen spiegelt sich auch in den verwendeten Zeichensystemen wider, die häufig auf filmischen Zeichensystemen basieren. D.h. neben dem vorherrschenden bildlichen und akustischen Zugang werden bekannte (konnotative) Codes verwendet, um den Benutzer in die Wahrnehmungsumgebung einzubeziehen [Phys99] [Kreu99].

Die Fortführung der Wahrnehmungsumgebung in der realen Umgebung des Benutzers spielt bei der Gestaltung von Infotainment-Anwendungen für Ausstellungen eine wesentliche Rolle. Auch wenn zurzeit häufig zusätzliche Informationen lediglich durch Informationssysteme angeboten werden, die einen eher „traditionellen" Zugang über Monitor, Tastatur und Maus ermöglichen [Wirt95], kann erst durch die räumliche Gestaltung ein dem Präsentations- und Benutzungskontext angemessener Wahrnehmungsraum geschaffen werden [WiKD98] [BeSy00a].

Medienkunst: Im Bereich der Medienkunst steht die experimentelle Erforschung der Schnittstelle zwischen realem und virtuellem Raum im Vordergrund, so dass das Interface nicht nur eine zusätzliche Idee, sondern ein wesentlicher Bestandteil der Werke ist [Dugu95] [LeSc98].

Dabei wird, ähnlich wie bereits im Bereich der Videoinstallationen [KlBF97], die semiotische Bedeutung der virtuellen Umgebung dem realen Raum gegenübergestellt [Lond95]. Hier werden Aspekte der symbolischen Präsenz des Rezipienten in der virtuellen Umgebung, wie in der Arbeit *Psychic Space* [Dink97], betrachtet, das Spannungsfeld zwischen ikonischen und symbolischen Zeichen, wie in den Arbeiten *Who Are You?* und *Beyond Pages* [Schw97], untersucht bzw. die bereits in Abschnitt 2.2.3 angesprochene Transformation der semiotischen Bedeutung digitaler Medien thematisiert [Shaw95].

Durch die Möglichkeiten der Beeinflussung eines Kunstwerks steht aber nicht mehr allein das Vorstellen bzw. Präsentieren einer Idee im Vordergrund, vielmehr handelt es sich hier um einen Prozess, den der Rezipient selbst steuern kann und damit als aktiver Teil an dem Kunstwerk partizipiert. Der Rezipient entwickelt somit auch das Kunstwerk weiter, das ausschließlich, wie etwa in den Arbeiten *Ultimo Ratio* [DiBr99] und *Frontiers of Utopia* [Scot95], den Rahmen bereitstellt. Der Künstler stellt das zu Grunde liegende Konzept, den Kontext und die sich daraus ergebenden Vorgaben als Regelwerk zur Verfügung, innerhalb dessen der Rezipient das Kunstwerk „erforschen" kann.

Kapitel 3

Entwurf multimedialer Anwendungen

Einer der zentralen Unterschiede zwischen multimedialen Anwendungen und traditionellen Medien wie etwa dem Film ist, dass bei dem Entwurf multimedialer Anwendungen neben der gestalterischen auch die technische Sicht zu betrachten ist. Bei der Produktion eines Films, Buchs etc. existieren geringe Einflüsse durch das Informationsübermittlungs- und Präsentationsmedium, da die (eingeschränkten) Wahlmöglichkeiten nur einen geringen Einfluss auf das Produkt haben. Das Präsentationsmedium multimedialer Anwendungen kann allerdings als Softwareprogramm selbst entwickelt werden, so dass bei dem Entwurf neben den gestalterischen Aspekten technische Aspekte zu betrachten sind. Wie in Abschnitt 2.3 gezeigt, spielen beide Aspekte dabei eine wichtige Rolle für die Schaffung einer Wahrnehmungsumgebung.

Während des Entwurfsprozesses multimedialer Anwendungen sind somit unterschiedliche Rollen zu betrachten, die neben der Konzeption, den Mediendaten und dem Design auch die Implementierungsphase des Entwurfsprozesses unterstützen. Um die einzelnen Rollen auszufüllen, sind an der Entwicklung unterschiedliche Spezialisten beteiligt, die einen gestalterischen bzw. technischen Blickwinkel auf den Entwurfsprozess haben.

Die unterschiedlichen Blickwinkel lassen sich auch in bekannten Ablauforganisationen für den Entwurf multimedialer Anwendungen wiederfinden. Zum einen steht ein von der Softwareentwicklung bekanntes strukturelles Konzept im Vordergrund, zum anderen kann der Entwurfsprozess aber auch auf einem mentalen Konzept basieren, das die Sicht des Rezipienten betont. Verglichen mit der in Kapitel 2 eingeführten Charakterisierung multimedialer Anwendungen, kann somit entweder von einem eher technischen oder eher gestalterischen Ansatz gesprochen werden.

Um die Ablauforganisation zu unterstützen, kann der Entwurfsprozess auch aus Sicht der Dokumente betrachtet werden, die als Austauschmedien zwischen den Rollen fungieren. Bei der sich daraus ergebenden Dokumentenarchitektur, die aus der Buchproduktion bekannt ist, wird zwischen dem Entwurf, der Repräsentation und der Abbildung der Dokumente auf ein Präsentationsmedium bzw. der Produktion eines Präsentationsmediums unterschieden.

3.1 Rollen

Der Entwurf multimedialer Anwendungen unterscheidet sich von gängigen Produktionen anderer „traditioneller“ Medien (Buch, Film, Musik, ...) grundsätzlich durch die Rollen, die bei der Produktion zu berücksichtigen sind. Zum einen sind dies medientypische Tätigkeiten, wie die Aufbereitung der zu präsentierenden Inhalte, die Erstellung unterschiedlicher Mediendaten - wie Bilder, Text, Filme, Musik - und die Beschreibung des Layouts eines Mediums. Zum anderen müssen während des Entwurfs die dynamischen Beziehungen zwischen den einzelnen Mediendaten festgelegt werden, die die Programmierung der interaktiven Struktur beinhalten.

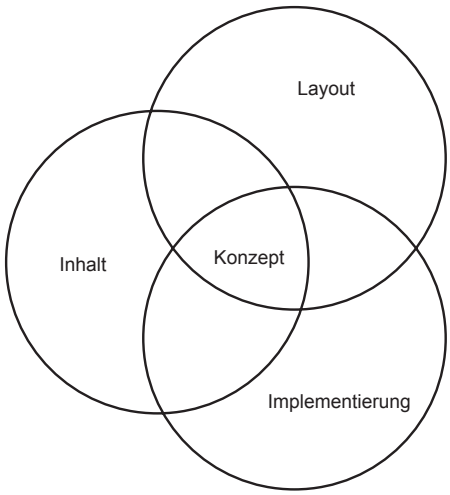


Abb. 3.1.1: Aspekte während des Entwurfsprozesses multimedialer Anwendungen.

Fasst man die gestalterischen und technischen Aspekte multimedialer Anwendungen zusammen, so können die zu betrachtenden Rollen während des Entwurfs, wie in Abb. 3.1.1 dargestellt, in die drei Hauptgebiete Inhalt, Layout und Implementierung eingeteilt werden [Fibi99]. Die durch dieses Modell erfassten Aspekte des Entwurfs, die für gewöhnlich durch ein Konzept miteinander verbunden sind, lassen sich durch folgende Aufgabenprofile näher beschreiben [ScGH98]:

Konzept: Das Konzept verbindet die einzelnen Aspekte des Entwurfs. Wie weiter unten noch genauer gezeigt wird, werden je nach Sichtweise auf den Entwurf unterschiedliche Modellierungsweisen angewendet. Man unterscheidet dabei zwischen strukturellem und mentalem Konzept, das zum einen die technischen, zum anderen die gestalterischen Aspekte des Entwurfs betont.

Recherche/Medienerfassung: Bei der Recherche für multimediale Anwendungen sind zwei Aspekte zu betrachten. Der erste Aspekt hierbei ist, ähnlich wie bei der Produktion anderer Medien, die Erschließung des zu präsentierenden Themengebiets bzw. Inhalts. Der zweite Aspekt ist die Erfassung verschiedener Informationsarten für die spätere Erstellung unterschiedlicher Medientypen.

Medienaufbereitung: Die während der Recherche bzw. Medienerfassung zusammengestellten Medien für die Präsentation müssen für die gewählte Ablaufumgebung aufbereitet werden. Neben der Digitalisierung einzelner Medien werden für gewöhnlich neue Medien wie Filme aus Bildmaterial oder 3D-Rekonstruktionen von Gebäuden erstellt.

Layout: Wie bei allen visuellen Medien gehört die Gestaltung zu einer Multimediaproduktion. Hierbei können unterschiedliche Techniken zum Einsatz kommen. So werden z.B. für Informationssysteme häufig so genannte Rastertechniken angewendet, um durch eine einheitliche Layoutstruktur die Inhalte zu präsentieren.

Verhalten: Durch die Beschreibung der Beziehungen zwischen den einzelnen Mediendaten wird das Verhalten einer multimedialen Anwendung definiert. Die dynamischen Beziehungen werden dabei durch eine Verhaltensbeschreibung spezifiziert, um z.B. durch interaktive bzw. zeitliche Aspekte das dynamische Verhalten einer Anwendung zu definieren.

Implementierung: Da multimediale Anwendungen in den meisten Fällen dynamische Beziehungstypen beinhalten, benötigt man ein eigenständiges Programm bzw. eine von speziellen Präsentationsprogrammen zu interpretierende Beschreibung der Präsentation. Erst durch die Realisierung während der Implementierungsphase werden dabei die aufbereiteten Mediendaten, das Layout und das definierte Verhalten der Präsentation zusammengeführt.

Installation: Die Implementierung multimedialer Anwendungen findet in den meisten Fällen auf einer speziellen Entwicklungsumgebung statt, so dass die erstellte Anwendung der Zielgruppe zur Verfügung gestellt werden muss. Gerade bei der Gestaltung von Wahrnehmungsumgebungen in realen Umgebungen, spielt die Installationsphase dabei eine wichtige Rolle.

Test: Da es sich bei multimedialen Anwendungen um eigenständige Programme bzw. zu interpretierende Beschreibungen handelt, ist wie bei anderen Softwareprojekten eine Testphase anzuschließen. Neben dem üblichen funktionalen Test müssen hierbei auch die Benutzerinteraktion, das Layout und die verwendeten Mediendaten aus der Sicht des Rezipienten betrachtet werden.

Um diese Aufgabenprofile während des Entwurfsprozesses auszufüllen, ist die Entwicklergruppe interdisziplinär zusammengesetzt. Die unterschiedlichen Spezialisten, wie Fachexperten, Graphik-Designer, Medientechniker und Informatiker betrachten die Produktion multimedialer Anwendungen somit aus unterschiedlichen Blickwinkeln.

Fachexperte: Der Fachexperte ist zuständig für den inhaltlichen Aufbau der multimedialen Anwendung. Hierzu gehören neben der Recherche und Medienerfassung ebenso die inhaltlich orientierten Strukturen der Anwendung. Während der Testphase ist der Fachexperte für die inhaltliche Qualitätskontrolle zuständig. Somit betrachtet er den Entwurf aus der Sicht des Inhalts.

Graphik-Designer: Aufgabe des Graphik-Designers ist es, die Form der Präsentation festzulegen. Dabei müssen neben dem zu präsentierenden Inhalt ebenso die technischen Möglichkeiten des Präsentationsmediums berücksichtigt werden. Erst hierdurch kann eine spezifische Form für multimediale Anwendungen entwickelt werden.

Neben dem häufig angewandten seitenbasierten Layout multimedialer Anwendungen, muss während des Entwurfs verteilter multimedialer Anwendungen ebenso die Gestal-

tung der realen Umgebung beachtet werden. Hierzu sind sowohl dreidimensionale Beschreibungen als auch die Bestimmung der Präsentations- und Interaktionsmedien notwendig.

Medientechniker: Der Medientechniker ist für die Aufbereitung der zu präsentierenden Inhalte zuständig. Dabei werden von ihm sowohl bereits vorliegende Inhalte in ein digitales Format gebracht als auch neue Mediendaten erstellt. So ist es z.B. Aufgabe eines Medientechnikers vorliegende Aufnahmen zu bearbeiten und in eine Panoramadarstellung zu konvertieren.

Der Medientechniker betrachtet somit, vergleichbar mit den Fachexperten, den Entwurf einer multimedialen Anwendung aus Sicht des Inhalts. Dabei ist diese Sicht gegenüber der Sicht des Fachexperten jedoch technischer ausgerichtet.

Informatiker: Um eine multimediale Anwendung zu realisieren, muss eine durch den Computer interpretierbare Beschreibung bzw. ein Programm erstellt werden. Dabei werden funktionale Beziehungen zwischen den einzelnen Elementen und die Interaktionsmöglichkeiten einer Anwendung von einem Informatiker oder Programmierer definiert. Des Weiteren übernimmt der Informatiker/Programmierer häufig die Aufgabe der Implementierung einer vorliegenden Beschreibung. Dabei wird die Programmierung häufig durch Werkzeuge, die bereits auf ein spezielles Präsentationsmedium zugeschnitten sind, unterstützt.

Neben diesen typischen Programmieraufgaben übernimmt der Informatiker die Installation und den funktionalen Test einer multimedialen Anwendung. Wie bereits weiter oben erwähnt spielen diese Aufgaben gerade bei der Realisierung verteilter multimedialer Anwendungen eine wesentliche Rolle, da hierbei häufig technisch komplexe Installationen aufzubauen sind.

Projektleiter: Durch die unterschiedlichen Blickwinkel der einzelnen Spezialisten besteht einer der wesentlichen Aufgaben des Projektleiters darin, die Gesamtsicht auf das Projekt zu wahren. Um dabei die Koordination zwischen den Experten zu steuern, sollte der Projektleiter die unterschiedlichen Diskurswelten der Experten verstehen. Erst hierdurch kann er vermittelnd und steuernd innerhalb der Projektgruppe auftreten.

Die unterschiedlichen Blickwinkel der einzelnen Experten auf die zu erstellende multimediale Anwendung drücken sich während des Entwurfs somit sowohl in den verschiedenen Diskurswelten als auch in der Art der Modellierung aus. Gerade hierdurch treten allerdings Schwierigkeiten auf, die den Entwurfsprozess behindern. Erst durch eine rollenspezifische Unterstützung der einzelnen Blickwinkel kann, wie in Kapitel 5 gezeigt wird, ein flexibler Rahmen für den Entwurf multimedialer Anwendungen bereitgestellt werden. Dieser Rahmen bildet das formale Gerüst, in dem sich die vermittelnde Gesamtsicht des Projektleiters widerspiegelt.

Die technisch orientierten Experten betrachten eine multimediale Anwendung als ein ablauffähiges Programm bzw. eine darstellbare Beschreibung mit interaktiven Elementen, so dass die Ansätze der Softwaretechnik anwendbar sind. Gestalterisch orientierte Experten legen den Schwerpunkt auf die Art der multimedialen Präsentation eines speziellen Inhalts, so dass die Sichten auf traditionelle Medien, wie der Film- bzw. Buchproduktion, angewendet werden können.

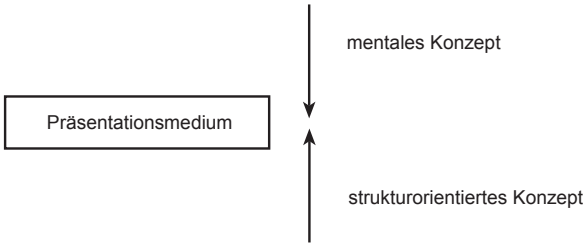


Abb. 3.1.2: Technische und gestalterische Sicht während der Konzepterstellung.

Diese beiden grundsätzlichen Blickwinkel spiegeln sich auch in der Modellierung des Konzepts wieder. Wie bereits weiter oben erwähnt, unterscheidet man zwischen strukturorientierten und mentalen Konzepten. Da bereits jedes Konzept ein Denkmodell der zu entwerfenden Anwendung festlegt, hat dies jedoch einen weitreichenden Einfluss auf die semiotische Bedeutung einer multimedialen Umgebung.

Bei dem strukturorientierten Konzept wird eine Anwendung durch ihre zu Grunde liegende logische Struktur beschrieben, d.h. die Modellierung findet anhand wahrgenommener Beziehungen zwischen einzelnen Elementen statt, die bereits indirekt das Erscheinungsbild der Anwendung festlegen.

Bei der Erstellung eines mentalen Konzepts führen hingegen assoziierte Modelle zu der Konzeptbeschreibung. Im Gegensatz zu dem strukturorientierten Konzept wird durch einen derartigen Ansatz somit das Gesamterscheinungsbild der Anwendung durch ein zu Grunde liegendes Modell bestimmt.

3.2 Entwurfsparadigmen

Wie im vorherigen Abschnitt gezeigt, lässt sich der Entwurf multimedialer Anwendungen aus unterschiedlichen Blickwinkeln betrachten. Dadurch ist der Entwicklungsprozess nicht mit den traditionellen Methoden aus den einzelnen Fachgebieten zufriedenstellend zu lösen. Je nach Sichtweise wird man daher für die Entwicklung unterschiedliche Methoden bevorzugen, die den Entwurf multimedialer Anwendungen unterstützen.

Durch das strukturorientierte Konzept werden die technischen Aspekte multimedialer Anwendungen betont, da während der Konzeptphase die logische Strukturierung einzelner Medienverwalter im Vordergrund steht und somit indirekt bereits das Erscheinungsbild der Anwendung festgelegt wird. Somit handelt es sich eher um einen Bottom-Up Entwurf, der mit der technischen Sicht auf multimediale Anwendungen vergleichbar ist. Die strukturorientierten Ansätze, die auf bekannten Methoden des Softwareentwurfs aufbauen, unterstützen den Entwurf deskriptiver multimedialer Anwendungen.

Im Gegensatz dazu wird durch den mentalen Ansatz zunächst das Gesamterscheinungsbild betont, das auf die einzelnen Mediendaten und deren Beziehungen abgebildet wird, so dass es sich hier eher um einen Top-Down Entwurf handelt. Dieser Ansatz, der von der agentenorientierten Modellierung bekannt ist, ist mit der Filmproduktion vergleichbar und unterstützt somit eher den Entwurf narrativer multimedialer Anwendungen.

3.2.1 Strukturorientierter Entwurf

3.2.1.1 Software-Entwurfsphasen

Die Entwicklung multimedialer Anwendungen wird in vielen Fällen als „normale“ Softwareproduktion angesehen. Um die Komplexität eines Programms beherrschen zu können, wird dabei die zu modellierende Aufgabenstellung in einzelne funktionale Einheiten aufgeteilt, die von unterschiedlichen Personen (Rollen) gelöst werden. Hierfür kann mithilfe unterschiedlicher Modelle, wie Prototyp-Entwurf, Spiralmodelle und Phasen- bzw. Wasserfallmodelle, eine Ablauforganisation (Protokolle) für die Beteiligten einer Entwicklergruppe vorgegeben werden, die die Kontrolle des Entwurfs ermöglicht [PaSi94].

Der prototypische Ansatz [HoKT93] eignet sich allerdings nur sehr eingeschränkt für die Entwicklung multimedialer Anwendungen, da die einzelnen Phasen während des Entwurfs zum Teil mit erheblichem Kostenaufwand verbunden sind. Somit wird dieser Ansatz hauptsächlich für erste experimentelle Entwürfe verwendet, wobei die Gestaltung der Oberfläche, wie in Abschnitt 3.2.2 noch näher gezeigt wird, im Vordergrund steht. Ebenso werden Spiralmodelle [MoFi96], die die einzelnen Schritte einer Multimediaproduktion integrieren, wegen ihrer schlechten finanziellen Kalkulierbarkeit abgelehnt.

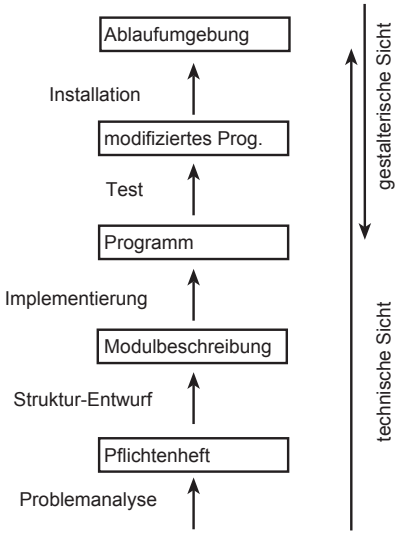


Abb. 3.2.1.1.1: Phasen der Softwareentwicklung nach [KKST79].

Die meisten Entwurfsmethoden multimedialer Anwendungen [Kinn94] [BuMo93] [Sawh95] [GaPS93] [GaMP96] [IsSB95] [ScRo98] basieren auf dem Wasserfallmodell, in dem der Entwurfsprozess eines Programms einzelne (abgeschlossene) Phasen durchläuft. Grundsätzlich lassen sich, wie in Abb. 3.2.1.1.1 dargestellt, fünf getrennte Phasen während des Softwareentwurfs unterscheiden [KKST79]:

Problemanalyse: Während der Problemanalyse werden die spezifischen Produktanforderungen festgelegt, d.h. es wird ermittelt, was das Produkt leisten soll, ohne auf das Wie einzugehen. Die möglichst detaillierte Spezifikation dessen, was das fertige Produkt leisten soll, wird im Anforderungskatalog bzw. Pflichtenheft niedergelegt.

Struktur-Entwurf: Der Entwurf beschäftigt sich auf der Grundlage der Analyse mit der internen Struktur des Softwaresystems. Es wird eine Zerlegung in einzelne Module vorgenommen und deren Zusammenwirken spezifiziert, so dass als Ergebnis des Entwurfs eine Softwarearchitektur feststeht.

Implementierung: Ziel der Implementierung ist es, ein in der gewünschten Systemumgebung lauffähiges Programm herzustellen. Dabei stehen auf dieser Stufe der Softwareentwicklung nicht mehr das System als Ganzes, sondern die einzelnen Module im Vordergrund. Hier wird die Datenrepräsentation und der Steuerfluss jedes Moduls und seiner Funktionen definiert und in einer konkreten Programmiersprache implementiert.

Test: In der Regel liegt das Gesamtsystem erst mit dem Abschluss der Implementierung vor, so dass erst in dieser Phase ein Integrations- und Leistungstest durchgeführt werden kann.

Installation: Da die Entwicklung des Programms in den meisten Fällen auf eigenen Rechnersystemen stattfindet, schließt sich eine Installationsphase an, in der das erstellte Programm auf die reale Umgebung übertragen wird.

Aufbauend auf diesem Modell existieren entsprechende Entwurfsmethoden, die einen für multimediale Anwendungen abgestimmten strukturierten Entwurf unterstützen. Ziel dieser strukturorientierten Methoden ist es, eine systematische Ablauforganisation bereitzustellen, die auf eine Beschreibung der Implementierung hinausläuft. Dabei werden die Mediendaten als eigenständige Elemente bzw. Objekte betrachtet, die zunächst von den eigentlichen Medientypen abstrahieren.

Grundsätzlich lassen sich zwei unterschiedliche Ansätze der Strukturbeschreibung unterscheiden. Der eine basiert auf dem von der Datenbankmodellierung bekannten Entity-Relationship (ER) Modell [LaLo95], der andere unterstützt die Strukturbeschreibung durch eine objektorientierte Modellierung.

Das *Hypermedia Design Model* (HDM) [GaPS93] bzw. die Weiterentwicklung HDM2 [GaMP96] befassen sich mit der Modellierung eines ER-Modells und den darauf aufbauenden Beziehungen zwischen den einzelnen Elementen bzw. Gegenständen, die als abstrakte Mediendaten aufgefasst werden. Die Definition unterschiedlicher medienspezifischer Beziehungstypen und die Beschreibung des Layouts der Anwendung wird durch diese Modelle allerdings nicht unterstützt. Einen weiter führenden Ansatz, der neben der Modellierung einer Navigationsstruktur auch die Beschreibung der Benutzerschnittstelle beinhaltet, stellt die *Relationship Management Methodology* [IsSB95] dar.

Bei der objektorientierten Modellierung werden die Mediendaten durch Objekte repräsentiert, so dass die Beziehungen zwischen den einzelnen Mediendaten durch Verfahren aus dem objektorientierten Entwurf beschrieben werden können. Bekanntester Vertreter dieser Klasse ist die *Object-Oriented Hypermedia Design Method* [ScRo98]. Werden durch diese Methode die einzelnen Phasen zunächst durch unterschiedliche Beschreibungsformen unterstützt, so sind aus der Literatur ebenso Ansätze bekannt, die durch Erweiterungen der Beschreibungssprache *Unified Modeling Language (UML)* [Burk99] einen durchgehenden Entwurfsprozess unterstützen [HeKo00] [BaKM99] [SaEn99].

Unabhängig von der Beschreibungsform wird bei diesen Methoden der Entwurf durch die Mediendaten, die als abstrakte Datenelemente bzw. Objekte modelliert werden, vorangetrie-

ben. Allerdings werden die Interaktionsmöglichkeiten einer multimedialen Anwendung und damit die Sicht des Benutzers erst in späteren Phasen betrachtet.

Durch die Beschreibung der Schnittstelle bei der Modellierung einer multimedialen Anwendung wird bei einigen Methoden der umgekehrte Weg unterstützt, d.h. es findet zunächst die Definition der Interaktionsmöglichkeiten statt, die dann in darauffolgenden Phasen durch ein zumeist objektorientiertes Modell umgesetzt werden. Auch hier sind aus der Literatur unterschiedliche Ansätze bekannt [Carr00] [BeHo99] [LKKC99], die wie die hier näher betrachtete *Scenario-Based Hypermedia Methodology* [LeLY99] in einzelnen Phasen zu einer multimedialen Anwendung führen.

3.2.1.2 Relationship Management Methodology

Bei der unter Federführung von T. Isakowitz, A. Kamis und M. Koufaris entwickelten *Relationship Management Methodology (RMM)* wird der Entwurfsprozess multimedialer Anwendungen in sieben Phasen aufgeteilt. Die Ablauforganisation und die einzelnen Rollen werden dabei durch die Entwicklungsumgebung *RMCase* [DIMG96] unterstützt, die eine durchgehende graphische Beschreibungsmöglichkeit für die einzelnen Phasen zur Verfügung stellt. Die graphische Repräsentation lässt sich mithilfe einer automatischen Transformation in das *Relationship Management Data Model (RMDM)* überführen.

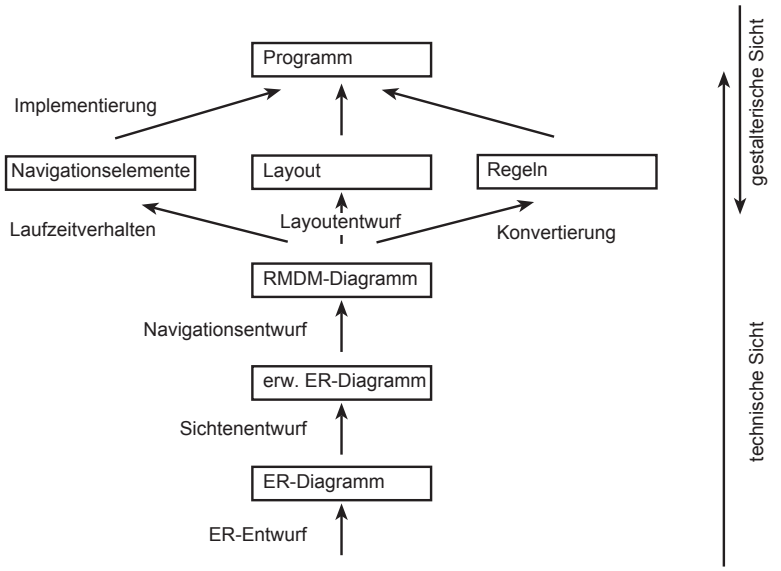


Abb. 3.2.1.2.1: Ablauforganisation von RMM.

Die einzelnen Phasen unterstützen eine sequenzielle Ablauforganisation, die die in Abschnitt 3.1 vorgestellten Rollen weiter unterteilt. So sind die ersten drei Phasen, wie in Abb. 3.2.1.2.1 dargestellt, eher dem Konzeptentwurf zuzurechnen. Die darauffolgenden Phasen befassen sich mit der Umsetzung in ein Präsentationsmedium, so dass sie dem Struktur-Entwurf bzw. der Implementierung zuzuordnen sind [IsSB95] [IsKK96] [IsKK97]:

ER-Entwurf: Ziel des ER-Entwurfs ist die Modellierung der Struktur des Themengebiets, das der multimedialen Anwendung zu Grunde liegt. Dazu werden die Elemente mit ihren Attributen sowie die Relationen zwischen den Elementen bestimmt. Die Attribute eines Elements dienen als Platzhalter der später einzufügenden Mediendaten einer Anwendung, so dass die Elemente bereits einen inhaltlichen Zusammenhang zwischen den Mediendaten repräsentieren. Teile der Relationen werden in der dritten Phase als Navigationsstruktur verwendet.

Der Entwurf wird durch die gängigen graphischen und formalen Methoden, die aus dem Datenbankentwurf [LaLo95] bekannt sind, unterstützt, so dass man nach Abschluss dieser Phase ein vollständig definiertes ER-Diagramm erhält.

Sichtenentwurf: Durch die Definition von so genannten Sichten wird festgelegt, welche Attribute (Mediendaten) eines Elements präsentiert werden. Den Sichten liegt eine Seitenmetapher zu Grunde, so dass diese ebenso als Seiten der Präsentation angesehen werden können. Da die Sichten jedoch nur auf einzelnen Elementen definiert werden können, werden somit bereits durch den ER-Entwurf die Informationseinheiten einer Seite bestimmt.

Die Definition von Sichten ist ebenfalls aus dem Datenbankentwurf bekannt. Im Gegensatz zu der dort verwendeten Modellierung ist in RMM die Beschreibung mithilfe von graphischen Symbolen vorgesehen [IsKK96]. Dadurch wird dem Entwickler eine intuitive graphische Benutzerschnittstelle zur Verfügung gestellt, die ein erweitertes ER-Diagramm repräsentiert, in dem jedes Element mit den entsprechenden Sichtendiagrammen gefüllt wird.

Navigationsentwurf: Aufbauend auf dem erweiterten ER-Diagramm werden in der dritten Phase die Navigationspfade definiert, die dem Benutzer zur Verfügung gestellt werden sollen. Dabei werden alle Relationen des erweiterten ER-Diagramms betrachtet und eventuelle Pfade auf entsprechende Navigationspfade abgebildet. RMM unterstützt drei Typen von Pfaden, einen Index-Pfad für den gezielten Zugriff auf einzelne Sichten, einen Tour-Pfad für eine geleitete „Führung“ durch eine Reihe von Sichten und eine Kombination aus beiden Pfadtypen.

Die Navigationsstruktur lässt sich wieder durch eine graphische Beschreibung bestimmen. Durch eine Transformation dieser Struktur in ein RMDM-Diagramm stehen die Zugriffsstrukturen für eine formale Bearbeitung in späteren Schritten zur Verfügung.

Während die ersten drei Phasen noch unabhängig von dem einzusetzenden Präsentationsmedium durchzuführen sind, werden in den folgenden Phasen von der Implementierung abhängige Aspekte berücksichtigt.

Konvertierungsentwurf: Die Definition von Konvertierungsregeln beschreibt die Abbildung der Navigationstypen des RMDM-Diagramms auf Elemente des Präsentationsmediums, mit denen sich die Navigationsinteraktionen implementieren lassen. So kann z.B. eine Listen-Box bzw. ein HTML-Menü für die Implementierung verwendet werden.

Die Definition der Konvertierungsregeln kann zum einen durch einfache Textbeschreibung erfolgen, die für die Implementierung als Spezifikation dient und somit von dem Entwickler in der entsprechenden Entwicklungsumgebung implementiert werden muss. Zum anderen können bereitgestellte Konvertierungsregeln verwendet werden, die eine au-

tomatische Transformation der Navigationstypen in eine HTML-Beschreibung ermöglichen.

Layoutentwurf: Um die visuelle Gestaltung der multimedialen Anwendung festzulegen, werden in dieser Phase die durch die Sichten zusammengefassten Attribute durch ein Seitenlayout beschrieben. Neben der statischen Beschreibung des Layouts können visuelle Übergänge durch die vom Präsentationsmedium angebotenen Überblendtechniken ergänzt werden.

Die Beschreibung des Layouts findet in der für das jeweilige Präsentationsmedium spezifischen Entwicklungsumgebung statt und stellt somit bereits einen Teil der Implementierung dar. Handelt es sich um eine HTML-Beschreibung, lassen sich die einzelnen Layoutbeschreibungen als Musterseiten für die dynamische Erzeugung von HTML-Seiten speichern.

Entwurf des Laufzeitverhaltens: Durch das Laufzeitverhalten werden die Elemente und Attribute des ER-Entwurfs als dynamische bzw. statische Inhalte charakterisiert, um festzulegen welche Inhalte erst während der Laufzeit bzw. bereits während der Implementierung in das Präsentationsmedium eingebunden werden können.

Neben der Betrachtung der Mediendaten wird in dieser Phase die funktionale Umsetzung der einzelnen Navigationsstrukturen für das Präsentationsmedium beschrieben.

Implementierung: Die durch den Konvertierungsentwurf, Layoutentwurf und durch die Definition des Laufzeitverhaltens getroffenen Festlegungen dienen als Vorlage für die Implementierung. Hier werden somit die Beschreibungen innerhalb einer speziellen Entwicklungsumgebung für das ausgewählte Präsentationsmedium umgesetzt.

Durch die feste Ablauforganisation von RMM werden die unterschiedlichen Rollen, die in Abschnitt 3.1 vorgestellt wurden, nicht gleichberechtigt behandelt. Vielmehr stehen durch die von der Softwaretechnik bekannte Strukturierung formal-analytische Tätigkeiten im Vordergrund, wodurch bereits wesentliche Einschränkungen für die visuelle Gestaltung der multimedialen Anwendung getroffen werden. Dadurch wird der erst während des Übergangs zur Implementierung durchgeführte Layoutentwurf ausschließlich auf den Sichten des erweiterten ER-Diagramms unterstützt.

Durch die Einschränkung der Sichtdefinitionen (Seiten) auf ein Element können hierbei unterschiedliche Elemente nicht berücksichtigt werden, so dass mehrere Informationseinheiten eines Typs nicht auf einer Seite dargestellt werden können.

Da die Definition der Interaktionsbeziehungen nur einfache Navigationsformen zwischen den einzelnen Sichten vorsieht, können innerhalb einer Seite keine Aktionen zwischen einzelnen Mediendaten angestoßen werden.

Aufgrund dieser Nachteile eignet sich RMM daher nur als Entwurfsmethode für Informationssysteme realisiert als Hypertext-Anwendungen im World Wide Web, die auf einem Datenbanksystem aufbauen. Für diese Art von Anwendungen ergeben sich allerdings gegenüber anderen Methoden wesentliche Vorteile, da die HTML-Präsentation der Informationen dynamisch aus der Datenbank erzeugt werden kann [IsSB95].

3.2.1.3 Object-Oriented Hypermedia Design Method

Wie bereits weiter oben erwähnt baut die von D. Schwabe, G. Rossi, C. J. P. Lucena und D. D. Cowan entwickelte *Object-Oriented Hypermedia Design Method (OOHDM)* auf

bekannten Konzepten des objektorientierten Softwareentwurfs auf. Ziel dieser Ablauforganisation ist die möglichst unabhängige Behandlung einzelner Phasen, so dass einzelne definierte Objektbeschreibungen für weiter führende Entwicklungen wieder verwendet werden können. Die einzelnen Phasen werden dabei durch bekannte Beschreibungswerkzeuge aus der Softwareentwicklung unterstützt, die einen von dem Präsentationsmedium unabhängigen Strukturentwurf ermöglichen.

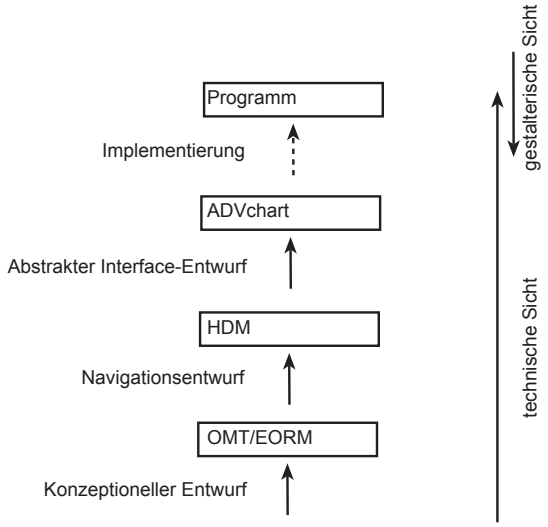


Abb. 3.2.1.3.1: Ablauforganisation von OOHDM.

Das OOHDM-Konzept unterscheidet, wie in Abb. 3.2.1.3.1 dargestellt, vier Phasen. Dabei liegen die Hauptunterschiede gegenüber RMM neben der Strukturierung durch objektorientierte Verfahren in der Trennung zwischen der Navigationsstruktur und der Definition einer abstrakten Interfacebeschreibung [RoSG97] [ScRo98]:

Konzeptioneller Entwurf: In der ersten Phase wird das Themengebiet der multimedialen Anwendung in eine objektorientierte Beschreibung überführt. Zentrale Elemente sind hierbei so genannte Konzeptklassen, die mit den Elementen der RMM-Modellierung zu vergleichen sind. Infolge der objektorientierten Beschreibung kann zwischen den Konzeptklassen durch Generalisierung, Spezialisierung und Vererbung eine Hierarchie definiert werden, so dass im Gegensatz zu der RMM-Modellierung eine feiner granulierte Beschreibung der Informationselemente und deren Beziehungen möglich ist.

Die Modellierung des Themengebiets wird in dieser Phase durch bekannte objektorientierte Modellierungstechniken (OMT) [RBPE91] bzw. durch das *Enhanced Object Relationship Model (EORM)* [Lang96] unterstützt, die als Grundlage der nachfolgenden Phasen verwendet werden.

Navigationsentwurf: Aufbauend auf dem konzeptionellen Entwurf werden in dieser Phase die Navigationsstrukturen definiert. Dabei bilden so genannte Knoten die Informationseinheiten und können mit den Sichten von RMM verglichen werden. Zwischen den Knoten lassen sich 1:n-Beziehungen definieren, die einfache Navigationsverweise auf an-

dere Knoten repräsentieren. Neben diesen einfachen Beziehungen lassen sich ebenso wie bei RMM Index- und Tour-Pfade definieren.

Die Beschreibung der Navigationsstruktur erfolgt mithilfe von HDM-Diagrammen und ist somit mit der Modellierung von RMM als Relationsbeziehungen vergleichbar.

Abstrakter Interface-Entwurf: In der dritten Phase werden die Interaktionsaspekte und das interne Verhalten der in der vorherigen Phase definierten Knoten festgelegt. Jedem Knoten muss dabei mindestens eine abstrakte Interfacebeschreibung zugeordnet werden. Die Beschreibung bleibt dabei unabhängig von späteren Implementierungswerkzeugen und Präsentationsmedien, so dass das Layout in dieser Phase nicht näher spezifiziert wird.

Die Definition der abstrakten Interfacebeschreibungen findet durch so genannte *Abstract Data View Charts (ADVchart)* [CCCL94] statt. Mithilfe von ADVcharts, die eine abstrakte und von der Implementierung unabhängige Definition unterstützen, können externen und internen Ereignissen Aktionen durch eine graphische Beschreibung zugeordnet werden.

Implementierung: In dieser Phase findet die eigentliche Umsetzung in eine ausgewählte Ablaufumgebung, wie *Director* [BaCM98] oder *Toolbook* [Eber00], bzw. in eine Beschreibungssprache wie HTML statt. Die Definitionen der ersten drei Phasen dienen dabei ausschließlich als formale Spezifikation für die manuelle Implementierung.

Neben der Programmierung des Verhaltens findet sowohl die Zuordnung der Mediendaten als auch ihre Anordnung innerhalb der ausgewählten Entwicklungsumgebung für das entsprechende Präsentationsmedium statt.

Die durch OOHDM festgelegte Ablauforganisation ist sehr stark an die Softwareentwicklung angelehnt. So werden die konzeptionellen Phasen nur durch Modellierungswerkzeuge aus der Softwaretechnik unterstützt. Damit steht, ebenso wie bei RMM, die technische Sicht auf die Produktion im Vordergrund.

Durch die objektorientierte Modellierung stehen den Entwicklern zwar weitreichende Definitionsmöglichkeiten zur Verfügung, die allerdings durch die meisten Entwicklungswerkzeuge eines Präsentationsmediums nicht in vollem Umfang unterstützt werden. Somit ist der Übergang zwischen den einzelnen Phasen nicht eindeutig definiert.

Der konzeptionelle Entwurf dient bei OOHDM jedoch nur als Spezifikation der Implementierung, die mittels eines Entwicklungswerkzeugs manuell durchgeführt wird. Erst hier findet die Integration der Mediendaten und die Gestaltung des Präsentationsmediums statt. Durch derartige Entwicklungswerkzeuge wird allerdings die Ablaufstruktur zwischen der interdisziplinär zusammengesetzten Entwicklergruppe nicht ausreichend unterstützt.

Insgesamt lässt sich festhalten, dass sich die durch OOHDM festgelegte Ablauforganisation sehr stark an der softwareorientierten Modellierung anlehnt, so dass die gestalterischen Aspekte einer multimedialen Anwendung auch hier vernachlässigt werden. Somit ist OOHDM ebenso wie RMM nur für den Entwurf reiner Hypertext-Anwendungen oder für Teilbereiche einer Anwendung, die hauptsächlich aus Hypertext bestehen, geeignet. Dabei sollten vorwiegend Graphen verwendet werden, um eine Gesamtsicht über die Anwendung und die Navigation zu bekommen. Durch die Trennung zwischen dem Navigationsentwurf und dem abstrakten Interface-Entwurf liegt dabei die Stärke von OOHDM gegenüber von RMM in der Wiederverwendbarkeit von Entwurfsmustern [LyRS99].

3.2.1.4 Scenario-Based Hypermedia Design Methodology

Die unter Federführung von C. Lee entwickelte *Scenario-Based Hypermedia Design Methodology (SHDM)* basiert ebenso wie OOHDM auf einer objektorientierten Modellierung. Jedoch wird durch SHDM der Entwurf nicht ausschließlich durch die Modellierung der Daten vorangetrieben. Vielmehr findet zunächst eine Betrachtung und Beschreibung unterschiedlicher Benutzerszenarien statt, wobei zunächst die Interaktion im Vordergrund steht.

Zur Unterstützung der Modellierung und der Rollen steht dabei die Entwicklungsumgebung *SHDMTool* [LeLY99] zur Verfügung, die eine durchgehende graphische Beschreibung ermöglicht. Dabei liegt das besondere Augenmerk der Entwicklungsumgebung auf der Unterstützung unterschiedlicher Präsentationsumgebungen, da durch ein spezielles Implementierungsdesign die Abbildung der Modellierung auf ein Zielsystem beschrieben wird.

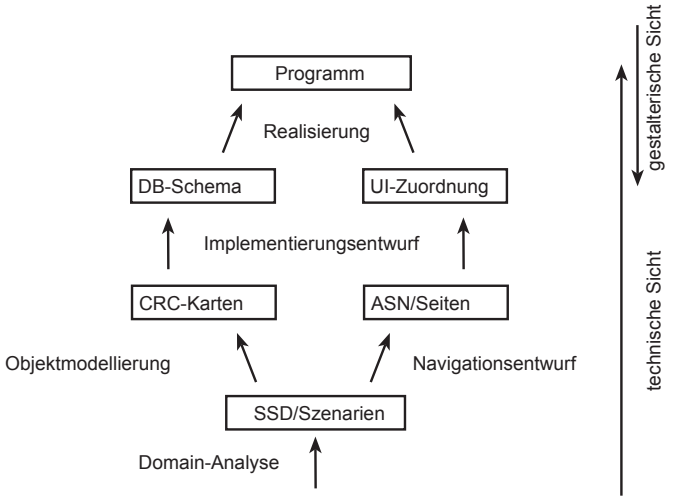


Abb. 3.2.1.4.1: Ablauforganisation von SHDM.

Durch das SHDM-Konzept werden, wie in Abb. 3.2.1.4.1 dargestellt, fünf Phasen unterschieden. Wie bereits weiter oben erwähnt wird in einem ersten Schritt die Schnittstelle zwischen multimedialer Anwendung und dem Benutzer analysiert und definiert. Die nachfolgende Definition der Objektmodellierung und des Navigationsentwurfs findet in Anlehnung an das OOHDM-Konzept statt [LeLY99] [LKKC99]:

Domain-Analyse: Um die Anforderungen an eine zu entwerfende multimediale Anwendung zu beschreiben, werden in einem ersten Schritt der zu modellierende Bereich und die Sichten der Benutzer definiert. Dabei steht zunächst die Eingrenzung der zu modellierenden Domäne im Vordergrund. Darauf aufbauend findet während der Analyse die Beschreibung unterschiedlicher Szenarien statt, die einzelne Use-Cases zusammenfasst.

Für die Beschreibung der zu modellierenden Domäne werden *System Scope Diagramme (SSD)* verwendet, um die Systemgrenzen und die Benutzersichten festzulegen. Die aus Benutzersicht definierten Szenarien werden in Tabellenform zusammengefasst, wobei die möglichen Aktivitäten/Interaktionsmöglichkeiten, die Reaktionen in Abhängigkeit des Zustands und die Informationsanzeigen aufgeführt werden.

Objektmodellierung: Um ein durch die Domain-Analyse beschriebenes Szenario umzusetzen, dient die Beschreibung als Basis der Objektmodellierung und des Navigationsdesigns.

Zunächst werden die Objekte als zusammenhängende logische Einheiten innerhalb der Szenariendefinition extrahiert. Wie von der objektorientierten Modellierung bekannt, werden die Objekte durch ihre Attribute und Methoden beschrieben. Durch unterschiedliche Beziehungstypen zwischen den Objekten können Vererbungen, spezifizierende Objekte und assoziierte Objekte beschrieben werden.

Die Objektmodellierung findet mithilfe so genannter *CRC-Karten (Class Responsibilities Collaborations)* statt. Um dabei die Beziehungen zwischen den Objekten zu beschreiben und graphisch darstellen zu können, werden Klassen-Struktur Diagramme (CSD) verwendet.

Navigationsentwurf: Abgeleitet von dem Objektmodell und der Beschreibung der Szenarien wird die Navigation festgelegt. Hierzu werden zunächst Sichten definiert, die die anzuzeigenden Daten der Objekte beschreiben. Dabei wird zwischen so genannten Basissichten, spezifizierenden und assoziierten Sichten entsprechend der bestehenden Objektbeziehungen unterschieden. Je nach Ausprägung spiegeln diese ausschließlich die Daten eines Objekttyps, die Daten eines spezifizierenden Objekttyps bzw. die Daten mehrerer assoziierter Objekttypen wider.

Um die Sichten miteinander zu verbinden, werden durch so genannte Zugriffsnavigationsknoten (ASN) Navigationseinheiten und Darstellungsbeziehungen festgelegt. Die einzelnen Einheiten werden letztendlich zu Seiten zusammengefasst, die durch Navigationsbeziehungen verbunden sind. Vergleichbar mit OOHDM können dabei unterschiedliche benutzerspezifische Interfacebeschreibungen definiert werden, wobei die Beschreibungen der einzelnen Szenarien als Vorlage dienen.

Implementierungsentwurf: Während des Implementierungsentwurfs wird die Umsetzung der Objektmodellierung und des Navigationsentwurfs festgelegt, so dass hier das zu verwendende Zielsystem zu berücksichtigen ist.

In einem ersten Schritt wird das Objektmodell auf ein Datenbankmodell transferiert. Hierbei werden zunächst die einzelnen Objekte als Elemente mit den entsprechenden Attributen umgesetzt. In einem zweiten Schritt findet die Abbildung der unterschiedlichen Beziehungstypen statt, die durch Fremdschlüssel bzw. durch eigene Elemente umgesetzt werden.

Die Umsetzung des Navigationsentwurfs wird durch die Beschreibung einzelner Interface-Komponenten definiert. Insbesondere findet dabei eine Zuordnung der unterschiedlichen Navigationsbeziehungen zwischen den Darstellungseinheiten und Seiten zu Links, zu Checkboxen, Menüs, etc. statt.

Realisierung: Während der Realisierungsphase findet die eigentliche Abbildung des Modells auf ein Zielsystem statt. Dabei werden die durch die Objektmodellierung und durch den Navigationsentwurf beschriebenen Elemente auf der entsprechenden Umgebung implementiert. Durch die Art der Modellierung sind datenbankbasierte und objektorientierte Systeme die vorherrschenden Architekturen einer Implementierung.

Im Gegensatz zu RMM und OOHDM wird zwar durch SHDM zunächst die Modellierung des Anwendungsbereichs und des Interfaces unterstützt, doch findet auch hier die Modellie-

rung nur mithilfe formaler Beschreibungsformen statt. Insbesondere findet kein eigentlicher Design-Entwurf statt, der das Erscheinungsbild der Anwendung beschreibt.

Durch die objektorientierte Modellierung ist der Entwurf durch die Objektmodellierung und den Navigationsentwurf vergleichbar mit OOHDM. Dies bedeutet aber auch, dass durch SHDM weiterhin die strukturierte Modellierung den wesentlichen Teil des Entwurfs ausmacht. Durch die dabei verwendeten Modellierungsarten werden hier jedoch ebenso ausschließlich technische Arbeitsweisen unterstützt, so dass die Aufgabenverteilung während des Entwurfs hauptsächlich von Informatikern übernommen wird.

Auch wenn die von der Softwaretechnik entlehnten Ansätze ein systematisches Vorgehen bei dem Entwurf unterstützen, lässt sich insgesamt festhalten, dass derartige Entwurfsmethoden zu restriktiv sind [BaLa01] [LaBa01] [Hira99]. Dies liegt zum einen an den formalen Beschreibungsformen, die der Denkweise technischer Experten entsprechen jedoch für gestalterisch ausgebildete Spezialisten eine Barriere darstellen. Zum anderen sind durch die Methoden bereits feste Phasen vorgesehen, so dass ein spontanes experimentelles Vorgehen, das aber typisch für den Entwurf multimedialer Anwendungen ist, nicht möglich ist. Daher werden durch diese Methoden die technischen Aspekte überbetont, so dass das Erscheinungsbild nur als Anhängsel betrachtet wird.

3.2.2 Mentaler Entwurf

3.2.2.1 Filmproduktionen

Die im vorherigen Abschnitt beschriebenen strukturorientierten Entwurfsmethoden basieren auf einer statischen Beschreibung, in die die Mediendaten integriert werden. Durch diesen Ansatz werden allerdings ausschließlich deskriptive Beziehungen zwischen den Mediendaten unterstützt, so dass sie nicht weiter für die in Abschnitt 2.3 vorgestellten narrativen multimedialen Genres geeignet sind.

Aspekte des narrativen Entwurfs multimedialer Anwendungen lassen sich mit der Filmproduktion vergleichen. Um eine Geschichte zu „erzählen“, stehen die Figuren der Geschichte sowie der zeitliche Ablauf einzelner Ereignisse im Vordergrund. Bei der Produktion eines Films stehen somit die gestalterischen Sichten des Drehbuchautors und Regisseurs im Mittelpunkt. Erst in späteren Phasen kommen technische Aspekte wie das Filmmaterial zum Tragen.

Da die Filmproduktion stark von dem Genre abhängig ist und z.B. bei den Dreharbeiten eines Spielfilms neben Aufnahmeaspekten vor allem schauspielerische Aspekte zu betrachten sind, sollen hier nur die für die Entwicklung multimedialer Anwendungen zentralen Aspekte der Filmproduktion, die in Abb. 3.2.2.1.1 zusammengefasst sind, anhand des Trickfilms betrachtet werden:

Handlungsbeschreibung: Um eine zu „erzählende“ Geschichte filmisch umzusetzen, muss der Autor eines Drehbuchs eine Auswahl an Ereignissen der Geschichte treffen, die den Ablauf der Erzählung vorantreiben. Ein Ereignis findet innerhalb einer Szene statt und erzeugt damit Bild, Handlung und Dialog zwischen Figuren [McKe00]. Durch die Dialoge und das Verhalten der Figuren werden Beziehungen zwischen den einzelnen Figuren aufgebaut, in deren Rahmen die eigentliche Geschichte „erzählt“ wird.

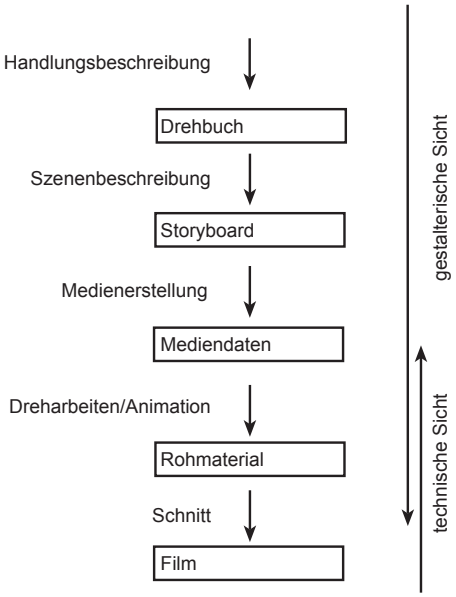


Abb. 3.2.2.1.1: Ablauforganisation der Filmproduktion.

Szenenbeschreibung: Die Szenenbeschreibung baut direkt auf dem Drehbuch auf. In dieser Phase steht der visuelle Grobentwurf der einzelnen Szenen im Vordergrund. Neben der Gestaltung der einzelnen Szenen wird die „Kameraführung“ beschrieben, die die Erzählperspektive einer Szene charakterisiert, die wiederum durch die in Abschnitt 2.3 aufgeführten Aufbau- und Auswahlkategorien bestimmt wird [Mona00].

Medienerstellung: Das Storyboard dient als Vorlage für die Erstellung der Medien. Die Gestaltung der Figuren betont die einzelnen Charaktere, um das Verhalten der Figuren durch das visuelle Erscheinungsbild zu unterstützen. Neben der Gestaltung der Charaktere findet die Medienerzeugung von Objekten und Hintergründen der Umgebung statt.

Dreharbeiten/Animation: Während der Dreharbeiten werden die einzelnen Szenen aus den Mediendaten zusammengesetzt und entsprechend des Drehbuchs und Storyboards animiert und aufgenommen. Nach den Dreharbeiten liegen die einzelnen Szenen als Rohmaterial vor.

Schnitt: Das Zusammenführen der einzelnen animierten bzw. aufgenommenen Szenen findet während der Schnittphase statt. Hierzu wird das Rohmaterial in einem Schnittprogramm zusammengeführt und als zusammenhängender Film produziert. Neben der Auswahl der Szenen wird die Filmmusik unterlegt, die eine nicht zu vernachlässigende Verdichtung der einzelnen Szenen und der Gesamtwirkung des Films hervorruft.

Die Erstellung eines Drehbuchs als eine „formale“ Beschreibung einer Geschichte kann ebenso als Metapher für die Modellierung einer Idee angesehen werden [Schö87], die als Basis für die Kommunikation zwischen unterschiedlichen Experten dient [Eric95] [Eric96]. Bei der Modellierung multimedialer Anwendungen findet sich dieser Gedanke bei den agentenorientierten Entwurfsmethoden wieder, in denen die Mediendaten als Akteure (Agenten) einer Szene interpretiert werden [Laur90]. Die Akteure reagieren dabei auf Änderungen ihrer Umgebung und sind somit häufig, wie in Abschnitt 4.3.3 gezeigt wird, als reaktive bzw. abwägende autonome Agenten realisiert.

Allerdings führt die Definition des Verhaltens einzelner Akteure auf eine explizite Beschreibung der Agenten zurück, so dass bei agentenorientierten Verfahren die Programmierung im Vordergrund steht, die bereits auf ein spezielles Agentensystem als Präsentationsmedium eingeschränkt ist. Die aus der Literatur bekannten Entwurfsverfahren unterstützen somit nur Teilaspekte des Entwurfs bzw. ermöglichen keine homogene Einbettung der Ablauforganisation.

Bei dem „Entwurf" multimedialer Anwendungen im Bereich Entertainment und für die Erzeugung computerunterstützter Animationen wird das Verhalten der Agenten durch Regeln definiert, die sowohl die Interaktion einzelner Akteure innerhalb einer dreidimensionalen Szene als auch die Interaktion mit dem Benutzer beschreiben. Um dabei eine realitätsnahe Modellierung zu ermöglichen, wird die Beobachtung der virtuellen Szene durch die Agenten mithilfe von virtuellen Sensoren implementiert [BeTh96] [ThNH97].

Zum Teil sind agentenorientierte „Verfahren" aber auch bereits in spezielle Anwendungen aus dem Bereich Edutainment bzw. Entertainment eingebettet. Der „Entwickler" bzw. Benutzer kann dabei jedoch nur das Verhalten der einzelnen vorgegebenen Agenten manipulieren [SmCS97] [HRBG97] [StLe97] [GrCl98], so dass diese Anwendungen erst in Kapitel 4 als Präsentationsumgebung näher betrachtet werden.

Sind die erwähnten „Entwurfsverfahren" bereits auf spezielle Anwendungsszenarien spezialisiert, so wird durch die *Dynamic Design Method* ein allgemeinerer Ansatz verfolgt. Hier werden sowohl das Erscheinungsbild einzelner Mediendaten als auch die Interaktionsbeziehungen durch das Verhalten der Agenten definiert, das zuvor in einem filmähnlichen Drehbuch beschrieben wird.

Betonen die agentenorientierten Entwurfsmethoden die einzelnen Medienelemente und deren Verhalten untereinander, steht bei drehbuchorientierten Entwurfsverfahren das Gesamtbild einer multimedialen Anwendung stärker im Vordergrund. Dabei wird die gestalterische Sicht des Entwurfs betont, da bei derartigen Methoden der Entwurf zumeist mithilfe der Anordnung der einzelnen Mediendaten innerhalb einer Szene bzw. Darstellungsfläche erfolgt [BaCM98] [Land96] [BaKC01].

Durch die Betonung der Gestaltung einer multimedialen Anwendung werden diese Methoden häufig für erste Entwurfsentscheidungen und für die Entwicklung von Prototypen eingesetzt. Durch den gestalterischen Schwerpunkt vermitteln die Prototypen dabei einen ersten Eindruck der Benutzeroberfläche, so dass bereits während des Entwurfs benutzerbezogene Test durchgeführt werden können. Um dabei einen schnellen Entwurf zu unterstützen, bieten die meisten Entwurfsumgebungen eigene Skriptsprachen an, wodurch der Entwurf allerdings auf proprietäre Präsentationsumgebungen eingeschränkt bleibt [BaCM98] [iShe01] [Bail99].

Auch wenn drehbuchorientierte Entwurfsverfahren zunächst vergleichbar mit den in Abschnitt 3.2.1 erwähnten szenarienorientierten Entwurfsverfahren erscheinen, liegt der wesentliche Unterschied in der Art der Beschreibung. Da die szenarienorientierten Verfahren Beschreibungsformen der Softwaretechnik unterstützen, steht bei diesen der formale Ansatz im Vordergrund. Dagegen werden durch die drehbuchorientierten Ansätze die Arbeitsformen gestalterischer Spezialisten betont, wie etwa durch den Ansatz des *Designing Multimedia Applications with Interactive Storyboards (DEMAIS)* [BaKC01a].

3.2.2.2 Dynamic Design Method

Die von S. Ishizaki entwickelte *Dynamic Design Method (DDM)* modelliert den Entwurf multimedialer Anwendungen als ein System einzelner Layoutelemente. Jedes Element, das

als Layout-Agent betrachtet wird, repräsentiert dabei eine einzelne Informationseinheit der Präsentation. Durch die Beschreibung bzw. Implementierung der einzelnen Layout-Agenten, entsteht somit ein System, in dem die Agenten auf Änderungen des Kontexts reagieren und die verwalteten Mediendaten der Situation angepasst präsentieren.

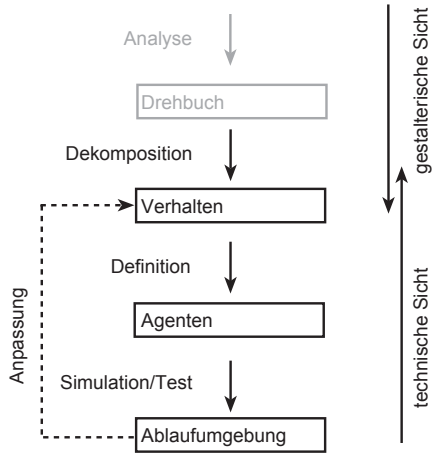


Abb. 3.2.2.2.1: Ablauforganisation von DDM.

Durch die verwendete Metapher einzelner separat agierender „Schauspieler“ lehnt sich die Ablauforganisation von DDM, wie in Abb. 3.2.2.2.1 dargestellt, an die Phasen der Filmproduktion an. Allerdings lässt sich durch die implizite Beschreibung der Beziehungen zwischen einzelnen Agenten das Erscheinungsbild einer multimedialen Anwendung erst während der Simulation bzw. Testphase vollständig überprüfen. Somit unterstützt der Entwurfsprozess ausschließlich ein experimentelles Vorgehen, bei dem die Rückkopplung mit der Regie bzw. Choreographie einer Theaterperformance vergleichbar ist [LoIs95] [Ishi97]:

Analyse: Während der Analyse werden der zu präsentierende Inhalt und der Kontext der Präsentation betrachtet. Hierbei werden die Charakteristika der Informationen, das Ziel der Präsentation und die Benutzeranforderungen untersucht.

Dekomposition: In dieser Phase wird die Kommunikationsanalyse auf einzelne Informationseinheiten aufgeteilt. Die Informationseinheiten werden dabei als eigenständige Agenten betrachtet, deren Regeln und Organisationsstrukturen bestimmt werden. Je nach Typ der Information werden Charakteristika des Agenten festgelegt, die die Präsentationsform der Information einer umgangssprachlichen Form beschreiben. Neben der separaten Sicht auf einzelne Agenten wird ebenso das Zusammenspiel mehrerer Agenten innerhalb einer Szene betrachtet, das das Gesamterscheinungsbild der Präsentation festlegt.

Definition: Die Umsetzung der Verhaltensbeschreibung in ein auszuführendes Programm wird durch eine spezielle Sprache unterstützt [LoIs95]. Die Beschreibung des Verhaltens umfasst die internen Zustände, Sensoren, Strategien und auszuführenden Aktionen eines Agenten.

Durch die Zustände wird das „Wissen“ eines Agenten repräsentiert, das das Erscheinungsbild eines Agenten widerspiegelt und Informationen über andere Agenten und Be-

nutzeranforderungen beinhaltet. Durch die Definition von Sensoren wird die Schnittstelle eines Agenten beschrieben, so dass dieser sowohl auf Änderungen anderer Agenten als auch auf Benutzeragenten reagieren kann.

Die Definition der Aktionen eines Agenten beschreibt die Möglichkeiten des Agenten auf Änderungen seines internen Zustands zu reagieren. Neben gestalterischen Aktionen, wie z.B. die Änderung seiner Position oder Größe, stehen weitere Aktionen für die Kommunikation mit anderen Agenten zur Verfügung.

Die eigentliche Definition des Verhaltens eines Agenten wird durch so genannte Strategien festgelegt, die durch ein Regelwerk beschrieben werden. Durch das Regelwerk wird dabei festgelegt, auf welche Änderungen des internen Zustands der Agent reagieren soll. Die Regeln setzen sich als Bedingungsabfrage und Ausführungsblock, in dem die anzustoßenden Aktionen definiert werden, zusammen.

Simulation/Test: Wie bereits weiter oben erwähnt spielt die Simulation des Gesamtsystems durch die implizite Beschreibung der Präsentation eine wesentliche Rolle bei der durch DDM unterstützten Ablauforganisation. Erst in dieser Phase wird das Zusammenspiel der einzelnen Agenten überprüft. Hierzu werden den Agenten die Mediendaten zugeordnet, und unterschiedliche Szenarien betrachtet. Erst durch diesen Test kann sichergestellt werden, dass die Präsentation der Mediendaten in der gewünschten Form stattfindet.

Auch wenn durch die von DDM gegebene Ablauforganisation filmische Aspekte berücksichtigt werden, so steht doch die Implementierung der einzelnen Agenten im Vordergrund. Zwar können vergleichbar mit der Filmproduktion auch hier durch die Analyse und die Dekomposition einzelne Szenen beschrieben werden, doch der Übergang von diesen konzeptionellen Phasen zu der Implementierung der Agenten stellt einen Bruch der Ablauforganisation dar.

Durch die ausschließliche Modellierung einzelner Mediendaten als separate Agenten können keine komplexen Strukturen aufgebaut werden. So ist es etwa nicht möglich, mehrere Agenten, die in einem funktionalen Zusammenhang stehen, zu gruppieren. Dies gilt auch für die implizite Definition des Layouts einer Präsentation. Es lassen sich zwar zeitliche Beziehungen zwischen den einzelnen Agenten beschreiben, allerdings wird durch die fehlende Möglichkeit einer Strukturierung die Beschreibung komplexer Anwendungen nicht hinreichend unterstützt.

Zusammenfassend kann somit festgehalten werden, dass sich DMM eher als Metapher für die Modellierung eignet, da es keine durchgehende Ablauforganisation zur Verfügung stellt und somit nur für kleine experimentelle Anwendungen geeignet ist.

3.2.2.3 Designing Multimedia Applications with Interactive Storyboards

Der unter Federführung von B. P. Bailey und J. A. Konstan entwickelte Ansatz des *Designing Multimedia Applications with Interactive Storyboards (DEMAIS)* stützt sich unmittelbar auf eine spezielle Entwicklungsumgebung, durch die der filmorientierte Entwurfsansatz betont wird. Der Schwerpunkt von DEMAIS liegt auf den frühen Entwurfsphasen, wobei insbesondere auf die schnelle, experimentelle Umsetzung erster Entwurfsideen geachtet wird.

Um dabei die rollenspezifischen Arbeitsformen gestalterischer Spezialisten zu unterstützen, bietet die Entwicklungsumgebung natürliche Definitionsformen für die Beschreibung des Ablaufs einer multimedialen Anwendung. Die einzelnen Beschreibungsformen sind dabei

von anderen Entwicklungsumgebungen bereits bekannt. So lassen sich, vergleichbar mit SILK [Land96], einzelne Szenen und Beziehungen durch Skizzen beschreiben, das Verhalten durch visuelle Programmierung definieren [iShe01], wobei die einzelnen Definitionen und Medien, wie dies etwa auch bei dem Entwurfswerkzeug Director [BaCM98] der Fall ist, gemeinsam verwaltet werden.

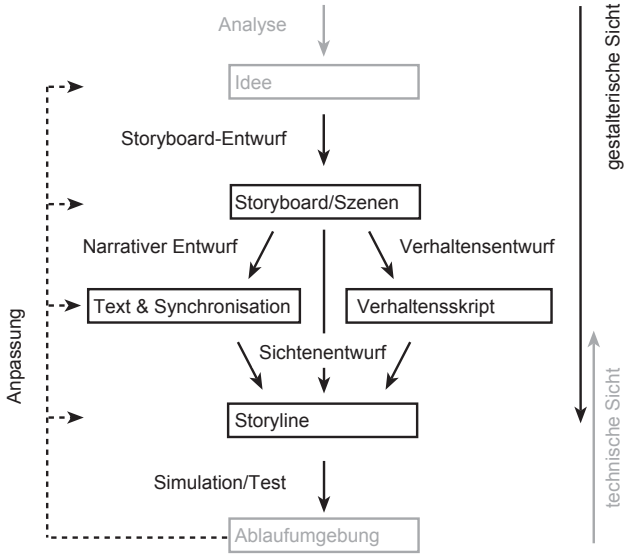


Abb. 3.2.2.3.1: Ablauforganisation von DEMAIS.

Wie bereits weiter oben erwähnt stehen bei DEMAIS einzelne Szenen im Vordergrund, die als filmische Plots aufzufassen sind. Im Gegensatz zu DDM wird somit der Entwurf durch einzelne Szenen geleitet, in denen die Mediendaten platziert werden. Hierbei lassen sich vier einzelne Beschreibungsformen unterscheiden, die innerhalb eines allgemeinen Entwurfsprozesses, wie in Abb. 3.2.2.3.1 dargestellt, eingebunden werden [BaKC01a] [BaKC01b] [BaKo00]:

Storyboard-Entwurf: Innerhalb eines Storyboards werden einzelne Szenen beschrieben, indem Elemente in der Szene bzw. auf der Darstellungsoberfläche platziert werden. Die Elemente dienen zunächst als Rahmen, denen einzelne Mediendaten zugewiesen werden können.

Die Beschreibung einer Szene findet dabei durch Skizzen statt, die über ein Graphik-Tablett eingegeben werden können. Durch diesen zeichnerischen Zugang kann die typische Arbeitsweise während der experimentellen Entwurfsphase unterstützt werden.

Aufbauend auf den einzelnen Szenen innerhalb des Storyboards findet die Definition des Verhaltens statt, die sich in drei unterschiedliche Beschreibungsformen aufspaltet. Der narrative Entwurf sowie der Verhaltensentwurf beschreiben das Verhalten einzelner Elemente innerhalb der Szenen. Durch den Sichtentwurf werden die einzelnen Szenen miteinander verknüpft.

Narrativer Entwurf: Um das zeitliche Verhalten innerhalb einer Szene zu beschreiben, können einer Szene beschreibende Texte zugewiesen werden, die als so genannte Off-

Stimme die jeweilige Szene kommentiert. Die Off-Stimme kann dabei als Text oder als Audio-Datei vorliegen.

Das zeitliche Verhalten einzelner Elemente innerhalb der Szene kann durch das Platzieren spezieller Synchronisationspunkte innerhalb des Textes definiert werden. Somit wird mithilfe des Textes eine Geschichte „erzählt". Während ihres Verlaufs werden unterschiedliche Aktionen innerhalb der Szene aufgerufen.

Verhaltensentwurf: Auch wenn durch den narrativen Entwurf bereits das zeitliche Verhalten einer Szene beschrieben werden kann, lassen sich durch die Definition des Verhaltens den Elementen spezielle Eigenschaften zuweisen. Zum einen wird dabei das Interaktionsverhalten der Elemente, zum anderen das Verhalten der Elemente untereinander definiert.

Gegenüber dem narrativen Entwurf wird das Verhalten durch eine deskriptive Beschreibung festgelegt. Dabei können wiederum durch zeichnerische Beschreibungsformen Navigationsbeziehungen und zeitliche Beziehungen zwischen Elementen definiert werden oder durch eine visuelle Programmierumgebung typische Ereignis-Aktionsbeziehungen festgelegt werden.

Sichtentwurf: Durch den Sichtentwurf lassen sich die einzelnen Szenen miteinander verknüpfen, um die so genannte Storyline zu definieren. Dabei stehen als Beziehungstypen die unterschiedlichen Definitionsformen der vorhergehenden Phasen zur Verfügung. So können Elemente innerhalb einer Szene Ausgangspunkt einer Verknüpfung sein bzw. narrative Synchronisationspunkte oder Ereignisse des Verhaltensentwurfs einen Szenenwechsel auslösen.

Wie bereits weiter oben erwähnt wird durch DEMAIS eine Entwicklungsumgebung zur Verfügung gestellt, die die schnelle Umsetzung erster Entwurfsideen unterstützt. Um dabei den experimentellen Charakter derartiger Entwürfe zu betonen, stehen skizzenartige Beschreibungsformen zur Verfügung, so dass die typischen rollenspezifischen Arbeitsweisen gestalterisch ausgebildeter Spezialisten unterstützt werden.

Zwar führen die Beschreibungen zu einem ersten (ausführbaren) Entwurf, doch kann durch eine derartige Beschreibung der Entwurf weder verfeinert werden noch auf unterschiedliche Präsentationsmedien abgebildet werden. Somit stellt DEMAIS, wie in Abb. 3.2.2.3.1 dargestellt, keine geeignete Implementierungsumgebung zur Verfügung. Zwar unterstützen andere drehbuchorientierte Entwurfswerkzeuge, wie etwa Director [BaCM98], eher die Implementierung, doch diese greifen während des Entwurfs zu kurz, da sie keine Abstraktionsstufen unterstützen.

Insgesamt lässt sich festhalten, dass die von der Filmproduktion entlehnten Ansätze im Gegensatz zu den softwareorientierten Ansätzen die gestalterische Sicht während des Entwurfs multimedialer Anwendungen betonen. Doch greifen sie oft zu kurz, da sie keinerlei Struktur vorgeben und somit nicht für größere Projekte mit einem interdisziplinären Team geeignet sind. Vielmehr wird hier der experimentelle Charakter für kleinere Entwürfe in den Vordergrund gestellt. Lassen die filmorientierten Entwurfsverfahren einen Organisationsrahmen gänzlich vermissen, so bieten die softwareorientierten Verfahren einen zu eng gefassten Rahmen während des Entwurfs.

Wie in Kapitel 5 noch gezeigt wird, wird der in dieser Arbeit entwickelte subjektorientierte Entwurf einen flexiblen Rahmen während des Entwurfs multimedialer Anwendungen zur

Verfügung, der das Zusammenspiel einer interdisziplinär zusammengesetzten Entwicklergruppe unterstützt. Dabei werden rollenspezifische Arbeitsformen während des Entwurfsprozesses in gleichberechtigter Weise zur Verfügung gestellt, so dass der Entwurfsprozess aus unterschiedlichen rollenspezifischen Sichten betrachtet werden kann. Somit kann der subjektorientierte Entwurf sowohl als experimentelle Entwurfsumgebung als auch als Implementierungsumgebung angesehen werden.

3.3 Beschreibungssprachen für den Entwurf

Neben den Rollen und Protokollen lässt sich der Entwurf multimedialer Anwendungen auch aus Sicht der Dokumente betrachten, die als Kommunikationsgrundlage der Ablauforganisation dienen. Aus dieser Sicht sind die Dokumente Grundlage des Entwurfs, der zu einer expliziten Beschreibung einer Anwendung führt.

Im vorherigen Abschnitt wurden bereits einige Beschreibungssprachen erwähnt. So basieren die strukturorientierten Entwurfsverfahren auf unterschiedlichen formalen Sprachen der Softwaretechnik, wohingegen die filmorientierten Verfahren lediglich auf einer nicht formalen Beschreibung einzelner Agenten basieren, die durch die Implementierung direkt in einem Präsentationsmedium umgesetzt wird.

Durch den rechnerunterstützten Entwurf bei der Medienproduktion liegt es nahe, die einzelnen Arbeitsschritte des Entwurfs durch eine einheitliche Beschreibungssprache zu unterstützen, die von Programmen interpretiert werden kann. Dieser aus der Buchproduktion bekannte Ansatz basiert auf so genannten Dokumentenarchitekturen. Die meisten Dokumentenarchitekturen betonen dabei die Strukturierung des Inhalts.

Vergleichbar mit den in Abschnitt 3.2.1 vorgestellten strukturorientierten Entwurfsverfahren werden somit durch die aus der Literatur bekannten Dokumentenarchitekturen bereits Strukturierungen vorgegeben, die einen einzelnen Blickwinkel betonen. Als formale Basis für den subjektorientierten Entwurf ergibt sich daher als Notwendigkeit die Einführung einer neuen Dokumentenarchitektur, durch die die unterschiedlichen rollenspezifischen Sichten gleichberechtigt unterstützt werden.

3.3.1 Dokumentenarchitekturen

Durch eine Dokumentenarchitektur wird die Syntax und Semantik von Beschreibungssprachen festgelegt, so dass der Übergang zwischen der Beschreibung einzelner Aspekte während der Erstellung eines Dokuments und der eigentlichen Realisierung durch eine Architektur definiert wird. Im Vordergrund der Erstellung stehen dabei die Inhalte, die durch die Beschreibungssprache zusammengeführt werden.

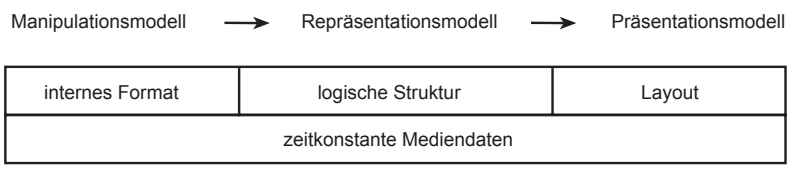


Abb. 3.3.1.1: Übersicht einer Dokumentenarchitektur für statische Dokumente.

Bei einer vollständig definierten Dokumentenarchitektur durchläuft ein Dokument von der Erzeugung der Inhalte über die Strukturierung bis zur Präsentation unterschiedliche Phasen. Die einzelnen Aspekte fließen während der Produktion in die Dokumentenbeschreibung ein bis eine vollständige Definition des Dokuments vorliegt. Durch das Zusammenbringen der einzelnen Aspekte wird im letzten Schritt das Dokument auf ein Präsentationsmedium abgebildet. Um die einzelnen Phasen des Entwurfs zu unterstützen, stellt eine Dokumentenarchitektur, wie in Abb.3.3.1.1 dargestellt, drei Modelle zur Verfügung:

Manipulationsmodell: Dieses Modell unterstützt die Erzeugung neuer und die Änderung bereits bestehender Dokumente. D.h. das Manipulationsmodell beschreibt den Umgang mit den Dokumenten durch entsprechende Programme, die während der Definitionsphase eingesetzt werden können.

Repräsentationsmodell: Repräsentationsmodelle definieren ein Austauschformat für Dokumente. Durch die Standardisierung von Austauschformaten werden die einzelnen Produktionsphasen zwischen Erstellung und Produktion/Implementierung miteinander verbunden.

Präsentationsmodell: Nach der Erstellung und Strukturierung der zu produzierenden Inhalte müssen diese letztendlich in eine zu präsentierende bzw. auszuführende Form gebracht werden. Hierfür wird durch das Präsentationsmodell die explizite (syntaktische) Struktur in eine durch das Präsentationsmedium bestimmte implizite (semantische) Wahrnehmung überführt.

Die Definition der Modelle einer Dokumentenarchitektur basiert auf einem Dokumentenformat, das auf der einen Seite die implizite semantische Sicht der Benutzer widerspiegelt, auf der anderen Seite die explizite Speicherung der Dokumente unterstützt. Um dies zu ermöglichen, haben sich Beschreibungssprachen durchgesetzt, die durch explizite Markierungen die Struktur eines Dokuments widerspiegeln. Durch eine fest vorgegebene Syntax der so genannten Auszeichnungssprachen wird festgelegt, wie diese Markierungen von unterschiedlichen Programmen zu verarbeiten sind.

Grundsätzlich lassen sich Auszeichnungssprachen in Repräsentations- und Präsentationssprachen unterteilen. Repräsentationssprachen legen nur eine Grammatik für die Beschreibung eines Dokuments fest, so dass sie auch als Metasprachen bezeichnet werden. Durch Repräsentationssprachen lassen sich für die zu repräsentierenden Inhalte selbst definierte Markierungen einführen, denen durch eine separate Beschreibung eine semantische Bedeutung zugewiesen werden kann. Gegenüber Repräsentationssprachen zeichnen sich Präsentationssprachen durch eine fest vorgegebene Semantik aus, die durch Programme interpretiert werden kann.

Als Grundlage der Auszeichnungssprachen kann die in den 60er Jahren entwickelte *Generalized Markup Language (GML)* angesehen werden. GML wurde als Grammatik zur Repräsentation von Dokumenten in der Buchproduktion entwickelt. Durch die Grammatik wird ein Rahmen festgelegt, in dem für unterschiedliche Dokumenttypen Auszeichnungssprachen entwickelt werden können. Um die einzelnen Aspekte einer Dokumentenverarbeitung zu trennen, wird dabei zwischen der logischen Struktur des Inhalts und der Formatierung unterschieden. Der Rahmen für die logische Struktur lässt sich durch eine so genannte *Dokument Typdefinition (DTD)* definieren, in der die formalen Regeln für die Struktur eines Dokuments angegeben werden. In ihr werden die Markierungen und Reihenfolge der Markierungen festgelegt, die in einem Dokument verwendet werden können.

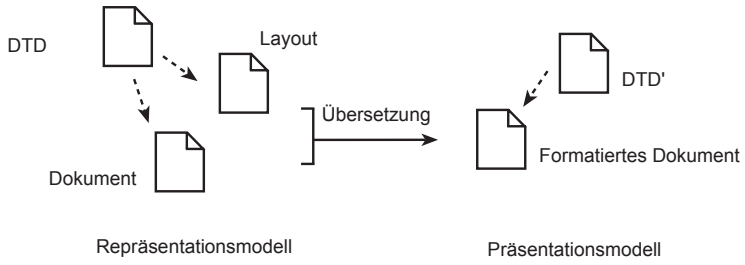


Abb. 3.3.1.2: Übergang zwischen Repräsentationsbeschreibung und Präsentationsbeschreibung.

Bei der Abbildung der einzelnen Aspekte einer Repräsentationssprache auf eine Präsentationssprache werden durch eine automatische Übersetzung, wie in Abb. 3.3.1.2 dargestellt, die Dokumentteile zusammengeführt. D.h. es findet eine durch das Präsentationsmodell gegebene (automatische) Transformation der Markierungen eines Dokuments statt, die auf der Überführung der DTD des Repräsentationsmodells in die DTD des Präsentationsmodells basiert.

Allerdings genügt die alleinige Definition einer logischen Struktur und der zu verwendenden Layoutregeln für multimediale Dokumente nicht. Vielmehr wird die logische Struktur durch die unterschiedlichen Beziehungsaspekte, die in Abschnitt 2.2.2 vorgestellt wurden, festgelegt. Neben der bei statischen Dokumenten üblichen linearen Gliederung des Inhalts durch eine Struktur, die sich letztendlich im Layout widerspiegelt, müssen die Strukturen der einzelnen Beziehungsaspekte festgelegt werden.

3.3.2 Repräsentationssprachen

Für die Modellierung multimedialer Anwendungen innerhalb eines Repräsentationsmodells sind unterschiedliche Auszeichnungssprachen aus der Literatur bekannt. Je nach Modellierungsansatz liegt der Schwerpunkt dieser Sprachen bei der flexiblen Definition einer logischen Struktur, bei der Beschreibung der Beziehungen zwischen einzelnen Mediendaten oder bei der Beschreibung eines einfach zu handhabenden (objektorientierten) Austauschformats.

Modell	**Struktur**	**Layout**	**Verhalten**	**Mediendaten**
SGML	Inhalt	DSSSL-Regeln	-	zeitkonstant
ODA	Inhalt / Layout	Struktur	-	zeitkonstant
HyTime	Inhalt	DSSSL-Regeln	Indirekte Beziehungen	zeitkonstant/-kontinuierlich
XML	Inhalt	XSL-Regeln	XLL-Links	zeitkonstant/-kontinuierlich
MHEG	Container	2D-Flächen 3D-Objekte	Skript/Aktionen	zeitkonstant/-kontinuierlich

Tab. 3.3.2.1: Dokumentbestandteile von Repräsentationssprachen.

In Tab. 3.3.2.1 sind die bekanntesten Repräsentationssprachen zusammengefasst. Durch die beiden aus der Buchproduktion stammenden Repräsentationssprachen *Standard Generalized*

Markup Language und die *Open Document Architecture* lassen sich ausschließlich zeitkonstante Mediendaten strukturieren, die durch eine separate Layoutbeschreibung in ein seitenorientiertes Präsentationsmodell transformiert werden können. Neben diesen beiden Sprachen werden die für multimediale Anwendungen entwickelten Repräsentationssprachen *Hypermedia/Time-based Structuring Language* und *Extensible Markup Language* sowie das von der *Multimedia and Hypermedia Expert Group* entwickelte Austauschformat vorgestellt:

Standard Generalized Markup Language: Die Beschreibungssprache *Standard Generalized Markup Language (SGML)* [Brya88] [Gold94] ist eine Metabeschreibungssprache für statische Dokumente. Wie bereits im vorherigen Abschnitt gezeigt, wird durch eine Metasprache zunächst nur die Syntax festgelegt, die durch eine DTD als Schema einer logischen Struktur weiter eingeschränkt werden kann. Neben der direkten Integration von Texten als Mediendaten können Bilder als externe zeitkonstante Mediendaten referenziert werden, so dass sie in die logische Struktur eingebunden werden können.

Um ein durch SGML repräsentiertes Dokument in ein Präsentationsmodell zu überführen, lassen sich für die verwendeten Marken und Attribute Regeln definieren, durch die die Transformation der logischen Struktur in eine Layoutbeschreibung festgelegt wird. Die Formatierungsregeln werden durch die auf *LISP* basierende *Document Style Semantics and Specification Language (DSSSL)* beschrieben. Eine DSSSL-Regel setzt sich aus einer Bedingung und einem Ausführungsblock zusammen. Durch die Bedingung wird festgelegt, für welche Marken und Attributwerte der logischen Struktur eines Dokuments die Regel Gültigkeit besitzt. Durch den Ausführungsblock wird die zu verwendende Beschreibung des Präsentationsmodells festgelegt.

Open Document Architecture: Die *Open Document Architecture (ODA)* [Stei99] ist ebenso wie SGML als Repräsentationssprache für statische Dokumente entwickelt worden. Ein Dokument wird in ODA durch zeitkonstante Mediendaten, durch deren logische Struktur sowie deren Layoutstruktur beschrieben. Die Mediendaten werden durch eine Referenz, durch die unterstützten Zugriffsmethoden und durch die Beschreibung des Datentyps innerhalb der Dokumentenbeschreibung repräsentiert. Basierend auf der Medienrepräsentation ist sowohl die logische Struktur als auch die Layoutstruktur definiert.

Die logische Struktur ist mit den Strukturierungsmöglichkeiten von SGML zu vergleichen. Ähnlich wie bei SGML wird demnach durch die logische Struktur der Aufbau eines Dokuments beschrieben. Im Gegensatz zu SGML wird jedoch die Layoutbeschreibung explizit als Baumstruktur definiert. Die Struktur spiegelt dabei ineinander geschachtelte Rahmen wider, in denen die Mediendaten platziert werden, so dass die Position (in Abhängigkeit von dem übergeordneten Rahmen) und die Größe der Darstellung beschrieben werden. Ebenso lässt sich für die einzelnen Mediendaten das Erscheinungsbild, wie z.B. der zu verwendende Schrifttyp und die zu verwendende Schriftfarbe, bestimmen. Durch diese Trennung unterstützt ODA eine flexiblere Layoutbeschreibung als SGML, da das Layout nicht (indirekt) von der logischen Struktur eines Dokuments abgeleitet wird.

Hypermedia/Time-based Structuring Language: Die *Hypermedia/Time-based Structuring Language (HyTime)* [ChGo91] ist eine Erweiterung von SGML, um die Strukturen multimedialer Dokumente zu beschreiben. Der Schwerpunkt dieser Repräsentationssprache liegt auf der Einführung eines standardisierten Mechanismus, der die Beschreibung der Beziehungen innerhalb eines Dokuments und zwischen unterschiedlichen

Dokumenten in einer einheitlichen und flexiblen Weise unterstützt. Die Referenzen basieren auf einer indirekten Adressierung, durch die sich sowohl räumliche und zeitliche Beziehungen als auch Navigationsbeziehungen zwischen den Dokumenten aufbauen lassen.

Die Strukturierung einzelner Dokumente ist durch HyTime nicht vorgegeben. Ebenso wie durch SGML lässt sich die logische Struktur eines Dokuments mithilfe einer DTD bestimmen, so dass HyTime ebenso eine Metasprache für unterschiedliche Anwendungsszenarien ist. Die von SGML geerbte Trennung der logischen Struktur und der Layoutbeschreibung unterstützt dabei die flexible Einbindung unterschiedlicher Dokumentenmodelle. Die eigentliche Layoutbeschreibung der Dokumente wird wie bei SGML durch die separate Definition von DSSSL-Regeln bestimmt.

Extensible Markup Language: Die *Extensible Markup Language (XML)* [xml99] [BeMi98][Brad98] basiert auf dem SGML-Standard und wurde für die Repräsentation von Dokumenten und strukturierten Informationen in verteilten Programmsystemen entwickelt. Hierfür wird durch XML die Syntax von SGML eingeschränkt, um eine einfachere rechnerunterstützte Weiterverarbeitung zu ermöglichen.

Neben der Überarbeitung der SGML-Syntax wurde XML gegenüber SGML um die Sprache *Extensible Link Language* (für die Definition von Navigationsbeziehungen) und die Sprache *Extensible Pointer Language* (für die Referenzierung von separaten Dokumenten) erweitert. Die für SGML entwickelte funktionale Transformationssprache DSSSL wurde durch die prozedurale Sprache *Extensible Style Language (XSL)* [xsl99] ersetzt.

Da durch XML ebenso wie durch SGML eine Grammatik für die Definition neuer Sprachen zur Verfügung steht, gewinnt XML für die Definition von Datenaustauschformaten und Präsentationssprachen in jüngster Zeit zunehmend an Bedeutung. Um eine eindeutige semantische Bedeutung derartiger Sprachen zu beschreiben, existieren bereits mehrere Standards als DTD für die Beschreibung von mathematischen, chemischen Formeln, bibliographischen Informationen etc.

Multimedia and Hypermedia Expert Group: Der durch die *Multimedia and Hypermedia Expert Group (MHEG)* [Stei99] eingeführte Standard definiert die Repräsentation vollständig beschriebener multimedialer Objekte. Im Gegensatz zu den bis jetzt betrachteten Metasprachen, die eine flexible Unterstützung unterschiedlicher Dokumentenmodelle ermöglichen, wird durch MHEG eine feste DTD eingeführt, die eine objektorientierte Beschreibungsstruktur wesentlicher Aspekte multimedialer Anwendungen bereits vorgibt.

Die einzelnen Aspekte umfassen die Beschreibung von Mediendaten, der Positionierung, der Aktionen bzw. Beziehungen und eine allgemeine Informationsbeschreibung. Die Beschreibungen werden in MHEG durch Container zusammengefasst, wobei die Schachtelung von Containern den Aufbau einer multimedialen Anwendung bereits widerspiegelt. Im Gegensatz zu den anderen in diesem Abschnitt beschriebenen Repräsentationssprachen ergibt sich durch die fest vorgegebene Strukturierung von MHEG bereits ein (objektorientierter) Modellierungsansatz.

3.3.3 Präsentationssprachen

Damit eine Ablaufumgebung einer multimedialen Anwendung die Dokumente einer Präsentationssprache interpretieren kann, besitzen diese Sprachen eine fest vorgegebene Semantik

der einzelnen Markierungen. In der folgenden Darstellung werden Präsentationssprachen betrachtet, die durch eine fest vorgegebene Dokument Typdefinition als Ausprägungen der Sprache XML definiert sind.

Modell	Struktur	Layout	Verhalten	Mediendaten
XHTML / HTML	Seite	Implizit	Links	zeitkonstant/-kontinuierlich
SMIL	Film	2D-Flächen	Links Zeitachse	zeitkonstant/-kontinuierlich
X3D / VRML	Szene	3D-Objekte	Skript	zeitkonstant/-kontinuierlich

Tab. 3.3.3.1: Dokumentbestandteile von Präsentationssprachen.

Zeichnen sich Repräsentationssprachen wie *SGML* oder *HyTime* durch ihre zum Teil flexiblen Modellierungsformen aus, so existieren zurzeit jedoch nur für spezielle Anwendungsszenarien entwickelte Präsentationssprachen. Im Folgenden werden zur Illustration die in Tab. 3.3.3.1 zusammengefassten Präsentationssprachen *XHTML* als seitenorientierte Sprache, *SMIL* als zeitorientierte und *X3D* als Sprache zur Beschreibung dreidimensionaler Szenen näher betrachtet:

Extensible Hypertext Markup Language: Durch die *Extensible Hypertext Markup Language (XHMTL)* [HaKK00] wurde die Präsentationssprache HTML der XML-Syntax angeglichen. In dieser wohl durch das World Wide Web bekanntesten Präsentationssprache lassen sich Texte und externe Mediendaten innerhalb einer Seite integrieren. Die Marken der Sprache besitzen dabei sowohl eine logische als auch eine (implizite) gestalterische Semantik.

Da XHTML-Dokumente unabhängig von der verwendeten Plattform präsentiert werden sollen, wird durch die Markierungen allerdings nur ein implizites Layout vorgegeben. Die Interpretation der vorgesehenen Marken findet erst während der Aufbereitung innerhalb der Ablaufumgebung statt. Diese Flexibilität der Darstellung wird durch die Einschränkung der Gestaltungsmöglichkeiten erkauft, die sich größtenteils auf die Formatierung einzelner Textteile beschränkt.

Die Interaktionsbeziehungen zwischen einzelnen Dokumenten lässt sich jedoch nur durch unidirektionale 1:1-Beziehungen beschreiben, so dass der Benutzer zwischen einzelnen Seiten navigieren kann. Neben der Einbindung unterschiedlicher externer Mediendaten unterstützt XHTML jedoch auch die Beschreibung einzelner Elemente der Benutzeroberfläche, wie etwa Menüs und Auswahlpunkte, so dass so genannte Formulare innerhalb einer XHTML-Seite integriert werden können. Die dadurch unterstützte programmähnliche Bedienung ermöglicht allerdings auch hier nur eine deskriptive Interaktion.

Synchronized Multimedia Integration Language: Die *Synchronized Multimedia Integration Language (SMIL)* [Goet00] basiert auf der XML-Syntax und wurde speziell für die Beschreibung zeitlicher Beziehungen innerhalb einer Präsentation eingeführt. Um die Reihenfolge der zu präsentierenden Mediendaten festzulegen, können durch SMIL einzelne Mediendaten auf eine virtuelle Zeitachse platziert werden sowie sequenzielle bzw. parallele Abläufe zwischen einzelnen Mediendaten definiert werden. Neben den zeit-

lichen Beziehungsaspekten zwischen einzelnen Mediendaten lassen sich nur die durch HTML unterstützten Interaktionsbeziehungen definieren, so dass der Benutzer auch hier ausschließlich zwischen einzelnen Präsentationen „navigieren“ kann.

Die Darstellung der Mediendaten wird in SMIL durch separate Layoutelemente definiert. Diese dienen als zweidimensionale rechteckige Projektionsfläche, auf der Ausschnitte von Mediendaten bzw. skalierte Mediendaten angezeigt werden können. Der Bezug zwischen Mediendaten und Layoutelement wird dabei durch eine Referenz festgelegt. Im Gegensatz zu HTML wird somit das Layout einer Präsentation explizit beschrieben.

Um durch ein SMIL-Dokument unterschiedliche Präsentationsmedien bzw. Benutzerprofile unterstützen zu können, lassen sich alternative Medienreferenzen definieren. Somit können z.B. für unterschiedliche Übertragungsgeschwindigkeiten geeignete komprimierte Mediendaten eingebunden werden bzw. kann durch die Auswahl einer bevorzugten Sprache die Präsentation durch die Einbindung entsprechender Mediendaten den Benutzerwünschen angepasst werden. Die alternativen Medienreferenzen lassen sich mit den im vorherigen Abschnitt vorgestellten Layout-Regeln von XML vergleichen, wobei hier die Auswahl erst während der Präsentation durch die Ablaufumgebung stattfindet.

Extensible 3D Language: Durch die *Extensible 3D Language (X3D)* wurde die *Virtual Reality Modeling Language (VRML)* [vrml97] als Präsentationssprache dreidimensionaler interaktiver Szenen der XML-Syntax angeglichen. Die Definition einer Szene wird durch eine Baumstruktur repräsentiert, die die räumlichen Beziehungen mithilfe von Gruppierungsknoten zwischen virtuellen Objekten einer Szene beschreibt. Durch die virtuellen Objekte lassen sich geometrische „materielle“ und „immaterielle“ Objekte definieren.

Materielle Objekte werden durch ihre Form, ihren visuellen Charakter und ihre Textur beschrieben. Die Form eines Objekts kann dabei sowohl durch geometrische Grundobjekte (Kegel, Kugel, Box, Zylinder) als auch durch beliebige Polygonflächen festgelegt werden. Der visuelle Charakter eines Objekts wird durch die Materialeigenschaften definiert, die die Lichtdurchlässigkeit, die Strahlkraft und die Grundfarbe eines Objekts bestimmen. Die Textur wird durch externe Mediendaten festgelegt, die auf die Oberfläche des Objekts projiziert werden. Somit können materielle Objekte als Layout-Objekte der Mediendaten interpretiert werden.

Für die Strukturierung einzelner virtueller Objekte stehen in X3D Gruppierungsknoten zur Verfügung, die als „Hülle“ mehrerer virtueller bzw. weiterer Gruppierungsknoten dienen. Dadurch wird es möglich, zusammenhängende Objekte als eigenständiges Objekt aufzufassen, die wie virtuelle Objekte angesprochen werden können und somit gemäß Abschnitt 2.2.1 komplexe Medienverwalter darstellen. So kann z.B. durch die Gruppierung einzelner Wände ein Raum und durch die Gruppierung mehrerer Räume ein Haus als eigenständiges Objekt definiert werden.

Durch immaterielle Objekte einer X3D-Szene lassen sich neben Licht- und Tonquellen so genannte Sensorobjekte definieren, die die zeitlichen Beziehungen und die Interaktion eines Benutzers mit der Szene unterstützen. Hierbei stehen neben einem Zeitsensor unterschiedliche Interaktionssensoren zur Verfügung, die z.B. „Bewegungen“ des Benutzers in der virtuellen Umgebung und die Sichtbarkeit eines durch den Sensor beschriebenen Bereichs registrieren.

Um die ausgelösten Ereignisse eines Sensorobjekts an andere Objekte weiterzuleiten, lassen sich durch X3D 1:1-Beziehungen zwischen Objekten definieren, so dass Ereignisse eines Objekts Attribute eines anderen Objekts ändern können. Diese einfache Ereignis-Aktions-Beziehung lässt sich durch so genannte Skriptobjekte erweitern, die zunächst das Ereignis verarbeiten und in Abhängigkeit von dem definierten Skript Aktionen des Zielobjekts auslösen. Als Beschreibungs- bzw. Programmiersprachen können in Skriptobjekten unterschiedliche Sprachen verwendet werden, die von der gewählten Ablaufumgebung unterstützt werden müssen.

Kapitel 4

Softwarebasierte Präsentationsmedien

Durch die in Kapitel 3 vorgestellten Entwurfsmethoden, werden die Rollen und Ablauforganisationen während der Produktion multimedialer Anwendungen festgelegt. Die einzelnen Phasen führen letztendlich zu einer Abbildung des Entwurfs auf ein Präsentationsmedium. Es ist sicherlich mittlerweile unstrittig, dass die Unterstützung der zu betrachtenden Aspekte des Entwurfs von entscheidender Bedeutung für den Erfolg oder Misserfolg eines Multimediaprojekts ist. Allerdings führt der Entwurf zu der Erstellung eines Programms, so dass die vorgestellten Entwurfsverfahren bereits die unterstützten Implementierungstechniken und deren funktionale Möglichkeiten berücksichtigen. Somit ist insgesamt eine erhebliche Rückwirkung der Modellierungskonzepte des Präsentationsmediums auf die Methoden des Entwurfs festzustellen.

Diese Problematik wird noch dadurch verstärkt, dass neben der Gestaltung der Oberfläche ebenso die „Gestaltung“ des Präsentationsmediums als Softwareprodukt eine entscheidende Rolle für die Schaffung einer Wahrnehmungsumgebung spielt. Wie in Abschnitt 2.3 gezeigt, müssen dabei sowohl die räumlich verteilte Abbildung als auch die Integration von unterschiedlichen Ein- und Ausgabemedien berücksichtigt werden. Dies setzt aber für eine Softwareumgebung voraus, dass sie eine offene Modellierung unterstützt, um eine dem Präsentationskontext angemessene Gestaltung zu ermöglichen.

Bei der Modellierung einer derartigen Softwareumgebung sind unterschiedliche Aspekte zu betrachten. Zum einen besteht eine verteilte Softwareumgebung aus separaten Software-Elementen, die einzelne Funktionen einer Umgebung realisieren, zum anderen muss durch die Softwareumgebung das Zusammenwirken bzw. die Kopplung der Software-Elemente unterstützt werden, wodurch die Gesamtfunktionalität einer Anwendung beschrieben wird. Je nach Abstraktionsgrad unterscheidet man zwischen objektorientierter, komponentenorientierter bzw. agentenorientierter Modellierung, die sich sowohl in der Beschreibung der Software-Elemente als auch in der Art der Kopplung zwischen einzelnen Elementen widerspiegelt.

Da die in dieser Arbeit vorgestellte verteilte multimediale Ablaufumgebung die Konzepte der komponenten- und agentenorientierten Modellierung miteinander verbindet, werden hier zunächst die grundlegenden Ansätze miteinander verglichen und typische Ausprägungen im Bereich multimedialer Anwendungen näher betrachtet.

4.1 Charakterisierung von Softwareumgebungen

Bei der Modellierung multimedialer Softwareumgebungen lassen sich unterschiedliche Aspekte unterscheiden. Zum einen kann die Softwareumgebung durch gängige Modellierungsansätze auf unterschiedlichen Ebenen betrachtet werden, zum anderen können anwendungsbezogene Ausprägungen multimedialer Aspekte berücksichtigt werden.

Da bei beiden Ansätzen grundsätzlich ein weites Spektrum an Charakteristika zu betrachten ist, wird hier der Schwerpunkt auf Aspekte gelegt, die für die Modellierung verteilter multimedialer Anwendungen von besonderer Bedeutung sind.

4.1.1 Verteilte Softwareumgebungen

Unter verteilten Systemen wird eine Art der Verarbeitung verstanden, bei der verschiedene Software-Elemente, die eine Anwendung bilden, auf unterschiedlichen Rechnersystemen eines Netzwerks ausgeführt werden. Die Grundlage für solche Systeme bildet die so genannte Dienstnehmer-Dienstleister-Architektur, bei der ein oder mehrere Dienstnehmer (Clients) mit einem oder mehreren Dienstleister/n (Server) über Netzprotokolle miteinander kommunizieren. Dabei interagiert lediglich der Client mit dem Benutzer, der eine Anfrage über den Client an den Server stellen kann. Der Client leitet diese Anfrage an den Server weiter und stellt das Ergebnis der Anfrage dem Benutzer zur Verfügung.

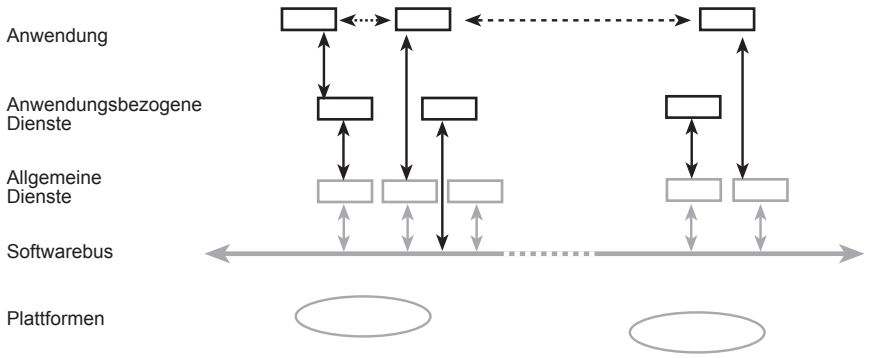


Abb. 4.1.1.1: Abstraktionsebenen verteilter Softwareumgebungen.

Diese traditionelle Sicht der Dienstanforderung und Dienstlieferung lässt sich auf komponentenbasierte und agentenbasierte Systeme nicht ohne weiteres abbilden, da hier Komposition und Kooperation zwischen verteilten Software-Elementen im Vordergrund stehen. Eine Charakterisierung verteilter Systeme, die von dem Client/Server-Konzept abstrahiert, ergibt sich nach [Grif98]:

Struktur: Ein verteiltes System ist aus einer Menge autonomer lokaler Systeme aufgebaut, deren Kommunikation ausschließlich auf dem Austausch von Nachrichten über ein Netzwerk beruht. Diese Forderung schließt somit die Kommunikation über gemeinsame Speicherbereiche ausdrücklich aus.

Zustand: Ein verteiltes System hat keinen festzustellenden globalen Zustand, da die Kommunikation zwischen den lokalen Systemen Vermittlungszeiten beinhaltet und durch die Struktur kein gemeinsamer Speicherbereich zur Verfügung steht.

Kooperation: Die einzelnen autonomen Knoten eines verteilten Systems definieren die Gesamtfunktionalität durch zielgerichtete Kooperation.

Durch diese Charakterisierung eines verteilten Systems wird allerdings noch keine Aussage über die technischen Grundlagen getroffen. Gerade die Einbindung unterschiedlicher Plattformen während der Realisierung verteilter Systeme, stellt jedoch hohe Anforderungen an die Softwareentwicklung.

Zentrale Probleme ergeben sich bei der Integration unterschiedlicher Plattformen, Programmiersprachen, Beschreibungsformen und dynamischer Strukturen in ein solches heterogenes verteiltes System. Um von einzelnen Aspekten abstrahieren zu können, sind in den letzten Jahren Softwarelösungen entwickelt worden, durch die tiefere Schichten einer Architektur verdeckt werden. Man unterscheidet dabei grundsätzlich, wie in Abb. 4.1.1.1 dargestellt, zwischen Plattformen, dem so genannten Softwarebus, allgemeinen Diensten, anwendungsbezogenen Diensten und den eigentlichen Anwendungen:

Plattformen: Als Plattformen werden die einzelnen Rechnersysteme verstanden, durch die ein verteiltes System konfiguriert wird. Neben der Hardware spielen dabei insbesondere die unterschiedlichen Betriebssysteme eine wichtige Rolle, da durch diese die Repräsentation unterschiedlicher Datentypen und die Grundfunktionalität eines lokalen Systems bestimmt wird.

Softwarebus: Unter einem Softwarebus versteht man eine standardisierte Infrastruktur, die den kommunizierenden Software-Elementen sowohl Formate als auch Mechanismen für den Austausch und die Bindung bereitstellt. Die Elemente melden sich bei einem Bus an und erhalten so einen Zugriff auf andere Elemente. Der Softwarebus stellt dabei eine direkte Sichtbarkeit der angemeldeten Elemente bereit, wobei von den unterschiedlichen Kommunikationsmechanismen und Datentypen der Betriebssysteme abstrahiert werden kann.

Allgemeine Dienste: Auch wenn durch einen Softwarebus betriebssystemnahe Funktionen verdeckt werden, können weitere Funktionen für die Modellierung verteilter Anwendungen durch allgemeine Dienste angeboten werden, die die Verwaltung, Kopplung und Verteilung einzelner Software-Elemente unterstützen. Durch diese Dienste wird allerdings weiterhin von der eigentlichen Anwendung abstrahiert, so dass sie in unterschiedlichen Anwendungsszenarien eingesetzt werden können.

Die durch einen Softwarebus und die allgemeinen Dienste bereitgestellten Softwarelösungen sind somit zwischen dem Betriebssystem und der eigentlichen Anwendung angesiedelt und werden daher auch als „Middleware“ bezeichnet. Die Middleware kann dabei als Schnittstelle, die bereits eine komplexe Funktionalität für Anwendungsprogramme bereitstellt, angesehen werden.

Anwendungsbezogene Dienste: Durch anwendungsbezogene Dienste kann für die Entwicklung von Anwendungen bereits eine vorgegebene Menge an spezialisierten Funktionen bzw. Diensten zur Verfügung gestellt werden. Je nach Spezialisierungsgrad können diese Dienste auf spezielle Anwendungen zugeschnitten sein bzw. ein weiter gefasstes Anwendungsgebiet abdecken.

Im Bereich multimedialer Anwendungen sind dabei insbesondere Dienste für unterschiedliche Medientypen, für unterschiedliche Beziehungen zwischen den Medienverwaltern und

für verschiedene Präsentationsmedien innerhalb einer verteilten Anwendung zu betrachten.

Anwendung: Als Anwendungen werden letztendlich die ablauffähigen Programme verstanden, die die definierte Funktionalität erfüllen. Multimediale Anwendungen können dabei durch Ablaufumgebungen dargestellt werden, die ausschließlich Beschreibungssprachen, wie etwa die in Abschnitt 3.3 vorgestellten Präsentationssprachen, interpretieren oder als übersetztes Programm eine fest definierte Anwendung realisieren.

Vergleicht man beide Ansätze mit „traditionellen" Präsentationsmedien (Diaprojektor, Videorekorder, etc.), kann eine interpretierende Ablaufumgebung, wie bereits in Abschnitt 2.1 erwähnt, ebenso als Präsentationsmedium angesehen werden, auf dem unterschiedliche Präsentationen „abgespielt" werden können. Im Gegensatz hierzu sind durch übersetzte Programme Präsentationsmedium und Daten miteinander verwoben.

Die Abstraktionsebenen einer verteilten Softwareumgebung lassen sich auch in verschiedenen Modellierungsansätzen wiederfinden. Wie bereits in Kapitel 2 und Kapitel 3 dargestellt, ist die objektorientierte Beschreibung der Medien, durch die die Kapselung der Medientypen ermöglicht wird, bei der technischen Sicht auf multimediale Präsentationen die am häufigsten zum Einsatz kommende Modellierungsform. Auch bei den Entwurfsverfahren, wie etwa bei OOHDM, findet eine Fortführung der objektorientierten Sichtweise statt.

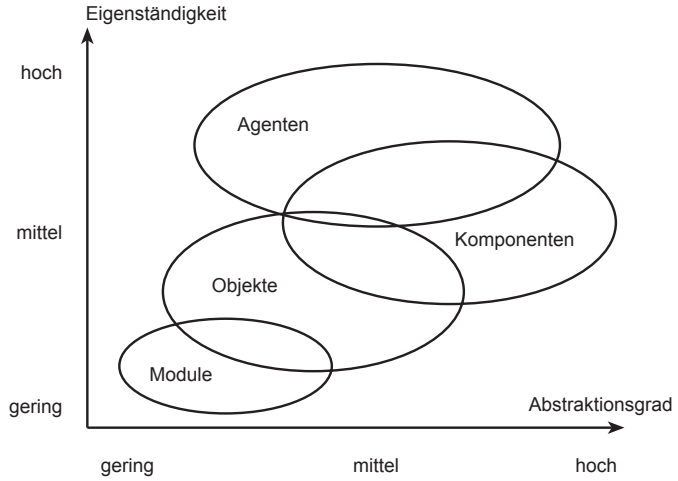


Abb. 4.1.1.2: Abstraktionsgrad und Eigenständigkeit von Software-Elementen.

Ausgehend von dem objektorientierten Ansatz, der sich bei der Modellierung und der Implementierung von Software mehr und mehr durchgesetzt hat, gewinnen bei verteilten Anwendungen komponenten- und agentenbasierte Ansätze immer mehr an Bedeutung. Wie in Abb. 4.1.1.2 dargestellt, zeichnen sich die verschiedenen Modellierungsansätze dabei durch unterschiedliche Eigenschaften aus. Ist der modul- und objektorientierte Ansatz noch sehr stark abhängig von der Implementierung, können durch die komponenten- und agentenorientierte Modellierung bereits höhere Abstraktionsgrade unterstützt werden.

Durch die komponentenorientierte Modellierung wird der Abstraktionsgrad betont, so dass Komponenten häufig als „Baukastensystem" anwendungsbezogener Dienste für einzelne Anwendungsgebiete eingesetzt werden. Im Bereich multimedialer Anwendungen sind dabei

Dokumentenmodelle, die in Abschnitt 4.2.3 näher beschrieben werden, Grundlage seitenbasierter Präsentationen. Vergleichbar mit den aus der objektorientierten Programmierung bekannten Medienverwaltern, können hierbei Mediendaten mithilfe von Komponenten gekapselt und zu komplexen Medienverwaltern zusammengeführt werden.

Gerade agentenorientierte Ansätze versprechen eine geeignete Modellierung für Software-Elemente innerhalb einer verteilten Anwendung, da durch die Eigenschaften einer losen Kopplung und der so genannten Migration einzelner Agenten die Eigenständigkeit von Agenten betont wird. Allerdings werden durch die aus der Literatur bekannten Agentensysteme ausschließlich Teilaspekte multimedialer Präsentationen unterstützt, da sie entweder nur als Implementierungsgrundlage allgemeine Dienste zur Verfügung stellen oder bereits als anwendungsbezogene Implementierungsgrundlage, wie in Abschnitt 4.3.3 noch gezeigt wird, lediglich für spezielle Anwendungsszenarien geeignet sind.

Vergleichbar mit den in Kapitel 3 vorgestellten strukturorientierten bzw. mentalen Entwurfsverfahren lassen sich die komponenten- bzw. agentenorientierten Ansätze wiederum eher der analytischen/konstruktiven Sicht bzw. eher einer Gesamtsicht auf eine multimediale Anwendung zuordnen. Somit spiegelt sich auch hier die technische und gestalterische Sicht auf die Software als Präsentationsmedium wider.

Dies wird gerade bei den in Abschnitt 4.2.3 bzw. 4.3.3 vorgestellten anwendungsbezogenen Komponenten- bzw. Agentensystemen durch die Art der Modellierung deutlich. So werden die dort vorgestellten komponentenbasierten Anwendungen durch eine vorgegebene Strukturierungsmöglichkeit, die so genannten Container, aufgebaut. Bei der Modellierung agentenbasierter Systeme steht dagegen die Interaktion bzw. Kommunikation zwischen autonomen Agenten im Vordergrund, wobei die Strukturierung meist ausschließlich implizit durch die Kommunikation zwischen den Agenten beschrieben wird.

4.1.2 Offene multimediale Ablaufumgebungen

Charakterisiert man multimediale Ablaufumgebungen sowohl durch die Möglichkeiten der Integration unterschiedlicher Medienformate bzw. verteilter Daten als auch durch die Möglichkeiten flexibler bzw. verschiedener Verhaltensbeschreibungen und verschiedener Präsentationssprachen, kann je nach den unterstützten Möglichkeiten zwischen offenen und geschlossenen Ablaufumgebungen gesprochen werden [Dale98].

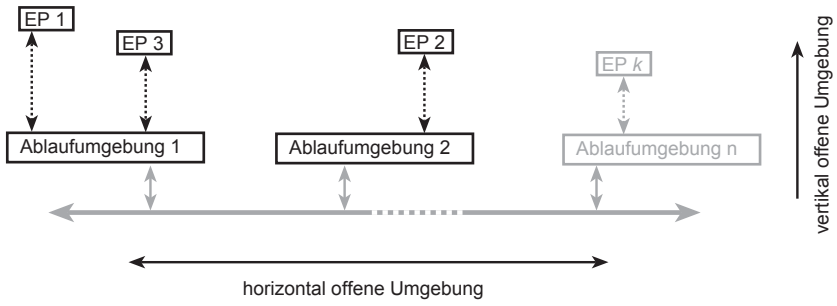


Abb. 4.1.2.1: Aspekte offener multimedialer Ablaufumgebungen.

Allerdings beziehen sich diese Aspekte ausschließlich auf eine lokale Ablaufumgebung als Präsentationsmedium bzw. auf tiefer liegende Aspekte, wie die in Abschnitt 2.1 beschriebenen Repräsentations- und Übermittlungsmedien, so dass sie für die Charakterisierung verteilter Anwendungen als Wahrnehmungsumgebungen zu kurz greifen.

Um die Charakteristika verteilter multimedialer Ablaufumgebungen für die Gestaltung von Wahrnehmungsumgebungen geeignet berücksichtigen zu können, werden in dieser Arbeit Aspekte der Integrationsmöglichkeiten verschiedener Präsentationsmedien betrachtet. Zum einen können die lokalen Ablaufumgebungen selbst, wie im vorherigen Abschnitt bereits erwähnt, als Präsentationsmedium interpretiert werden. Zum anderen muss, wie bereits in Abschnitt 2.3 gezeigt, die Integration externer Präsentationsmedien unterstützt werden, so dass sowohl externe Ausgabegeräte als auch „natürliche“ Interaktionsformen durch externe Eingabegeräte für die Gestaltung einer Wahrnehmungsumgebung berücksichtigt werden können. Hierdurch können multimediale Ablaufumgebungen, wie in Abb. 4.1.2.1 dargestellt, als horizontal und/oder vertikal offen bzw. geschlossen charakterisiert werden:

Horizontal offene Umgebungen: Durch die Interpretation der lokalen Ablaufumgebungen als Präsentationsmedium, kann durch diese in einem verteilten System eine zusammenhängende Präsentation gestaltet werden. Als horizontal offene Ablaufumgebungen werden somit Präsentationsmedien bezeichnet, die die Integration mehrerer lokaler Umgebungen unterstützen. Wie im vorherigen Abschnitt erwähnt, wird dabei die Gesamtfunktionalität eines verteilten Systems, hier bezogen auf die Interaktion und die Wiedergabe der Mediendaten, durch gezielte Kooperation der lokalen Knoten definiert.

Vertikal offene Umgebungen: Betrachtet man eine einzelne lokale Ablaufumgebung, kann man zwischen der gewöhnlichen internen Wiedergabe der Mediendaten und externen Präsentationsmedien (EP) unterscheiden. Als vertikal offene Ablaufumgebungen werden dabei lokale Umgebungen bezeichnet, die die Integration unterschiedlicher Präsentationsmedien unterstützen, so dass neben der internen Wiedergabe externe Ein- und Ausgabegeräte eingebunden werden können.

Auch wenn Komponenten bzw. Agenten, wie im vorherigen Abschnitt bereits gezeigt, durch ihren Abstraktionsgrad bzw. ihre Eigenständigkeit Grundmechanismen für die Modellierung offener multimedialer Ablaufumgebungen zur Verfügung stellen, werden durch bestehende Ansätze in anwendungsorientierten Ebenen ausschließlich Teilaspekte realisiert.

Die im Rahmen dieser Arbeit entwickelte Architektur verbindet die Vorzüge der komponenten- mit denen der agentenorientierten Architekturen. Dabei werden die multimedialen Inhalte, die durch separate Beschreibungen der eigentlichen Mediendaten, des Erscheinungsbilds und des Verhaltens charakterisiert werden, durch unterschiedliche Komponenten repräsentiert. Durch die separate Modellierung der Aspekte wird dabei die vertikal offene Modellierung unterstützt. Um die Koordination der multimedialen Inhalte innerhalb einer verteilten multimedialen Anwendung zu unterstützen, wird durch eine entwickelte Agentenarchitektur die organisierende Struktur gebildet. Da die dort eingesetzten Agenten als mobile autonome Agenten modelliert sind, wird durch diese somit die horizontal offene Modellierung verteilter multimedialer Anwendungen unterstützt.

Um die Grundlagen für die im Rahmen dieser Arbeit entwickelte offene Ablaufumgebung darstellen zu können, werden komponenten- und agentenorientierte Softwareumgebungen in den nachfolgenden Abschnitten auf unterschiedlichen Abstraktionsebenen betrachtet und miteinander verglichen.

4.2 Komponentenorientierte Softwareumgebungen

Auch wenn der objektorientierte Entwurf bereits viele Vorteile gegenüber der prozeduralen Beschreibung aufweist, man denke hier etwa an die domänennahe Modellierung durch Klassen bzw. Objekte, gewinnt der komponentenorientierte Entwurf immer mehr an Bedeutung. Hierbei ist, wie bereits im vorherigen Abschnitt erwähnt, das Abstraktionsniveau entscheidendes Charakteristikum der komponentenorientierten Modellierung. Je nach Betrachtungswinkel wird dabei die komponentenorientierte gegenüber der objektorientierten Modellierung abgegrenzt [HoCh91] [NiDa95] [Jell98] [Grif98].

Um den Komponentenbegriff besser fassen zu können, werden hier unterschiedliche Aspekte von Komponenten betrachtet, wobei das Hauptgewicht auf den für diese Arbeit relevanten Aspekten der Kopplung zwischen Komponenten sowie der Verwaltung und Anpassungsfähigkeit von Komponenten liegt. Anhand spezieller Ausprägungen werden abschließend anwendungsbezogene Komponentenmodelle aus dem Bereich multimedialer Anwendungen vorgestellt.

4.2.1 Komponentenparadigma

Objekte werden eher als Repräsentanten von Daten angesehen, die eventuell eine Schnittstelle zur Manipulation der Daten anbieten, so dass Objekte im allgemeinen passive Software-Elemente sind. Durch die abstrakte Sichtweise auf Komponenten wird dagegen eher das Angebot eines Dienstes betont, den andere Software-Elemente in Anspruch nehmen können. Durch das Zusammenwirken einzelner Komponenten können neue Dienste erstellt werden, ohne die Komponenten neu entwerfen zu müssen. Um diese Sichtweise zu ermöglichen, werden bei der Modellierung und Realisierung von Komponenten die Aspekte der Abgrenzbarkeit, Eigenständigkeit, Integrationsfähigkeit und Anpassungsfähigkeit betont [Grif98].

Die erste Forderung ist von dem objektorientierten Entwurf bekannt und wird dort durch die Kapselung interner Zustände und Eigenschaften realisiert. Im Gegensatz zu diesem Ansatz, bei dem die Schnittstelle eines Objekts sowohl implizit als auch explizit beschrieben werden kann, wird bei dem komponentenorientierten Entwurf eine klare Trennung zwischen Schnittstellenbeschreibung und Implementierung getroffen.

Die Eigenständigkeit betont den bereits oben angesprochenen Dienstcharakter einer Komponente, d.h. der Objektbegriff wird in diesem Fall um eine Prozesssicht erweitert, so dass nicht nur Zustände, sondern auch das dynamische Verhalten charakterisiert werden kann.

Die beiden bisher betrachteten Forderungen beziehen sich auf die Modellierung einzelner Komponenten. Allerdings soll durch den komponentenorientierten Entwurf das problemlose Zusammenfügen mehrerer Komponenten unterstützt werden, um Anwendungen mithilfe vorgefertigter Elemente erweitern bzw. neu erstellen zu können. Komponenten sollen dabei dynamisch während der Laufzeit in eine Ablaufumgebung zu integrieren sein, so dass sie direkt der Anwendung zur Verfügung stehen. Erstens erfordert dies die Verwaltung von Komponenten, die der Ablaufumgebung obliegt. Zweitens muss die Kommunikation zwischen Komponenten entkoppelt werden, im Gegensatz zur Kommunikation zwischen Objekten. Drittens müssen Komponenten dynamisch konfigurierbar und mit anderen Komponenten kombinierbar sein.

Die Verwaltung wird in Softwareumgebungen häufig durch so genannte Namens- bzw. Verzeichnisdienste, die in Abschnitt 4.2.2.1 näher beschrieben werden, unterstützt. Die Komponenten können dabei durch logische Namen referenziert werden, wodurch bereits eine lose Bindung zwischen diesen unterstützt wird.

Die Entkopplung wird durch eine Standardisierung der Komponenten-Schnittstellen innerhalb eines Systems ermöglicht. Dabei kann die Schnittstelle durch eine Beschreibung der zur Verfügung stehenden Methoden spezifiziert werden, wie etwa bei den *JavaBeans* [Neil98], so dass Komponenten zur Laufzeit über die Möglichkeiten bzw. Methoden anderer Komponenten informiert werden können. Die eigentliche Kommunikation findet allerdings hierbei weiterhin durch Methodenaufrufe statt. Um eine weitergehende Entkopplung zu ermöglichen, basieren einige Kommunikationsmodelle auf so genannten semantischen Ereignissen, die durch die Komponenten als Nachrichten interpretiert werden. Dabei wird die Übermittlung der Nachrichten, wie in Abschnitt 4.2.2.2 noch näher gezeigt, durch Ereigniskanäle ermöglicht.

Die Anpassungsfähigkeit wird häufig mithilfe von Skriptsprachen unterstützt, durch die das Verhalten einer Komponente erweitert bzw. angepasst werden kann. Hierdurch können Komponenten in unterschiedlichen Anwendungsszenarien eingesetzt werden, wobei die Funktionalität durch die Erweiterung des internen Zustands einer Komponente bzw. durch das Zusammenfassen mehrerer Komponenten, die als Komposition die Funktionalität erfüllen, bestimmt werden kann.

Durch die oben beschriebenen Aspekte der Abgrenzbarkeit, Eigenständigkeit, Integrationsfähigkeit und Anpassungsfähigkeit, können Komponenten zu so genannten Frameworks zusammengestellt werden, aus denen eine Anwendung, vergleichbar mit einem Baukastensystem, zusammengesetzt bzw. durch weitere Komponenten ergänzt werden kann. Allerdings ist der Entwurf eines umfassenden „Baukastensystems“ für die Realisierung komplexer Anwendungen sehr zeitaufwendig.

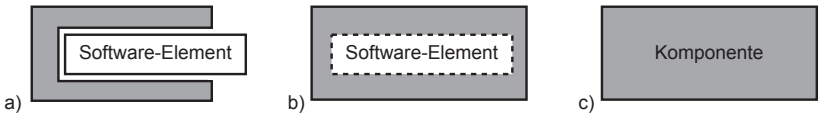


Abb. 4.2.1.2: Modellierungstechniken für die Integration externer Software-Elemente durch übersetzende (a), verpackende (b) Komponenten bzw. durch Neuimplementierung (c).

Da jedoch durch die Abgrenzbarkeit und Eigenständigkeit von Komponenten eine klare Trennung zwischen Implementierung und Modellierung getroffen wird, kann der komponentenorientierte Entwurf als Abstraktion der objektorientierten Beschreibung begriffen werden [NiMe94]. Somit können auch Modelle ohne explizite Objektsicht durch das Komponentenmodell abgedeckt werden, so dass bereits existierende Software-Elemente, wie in Abb. 4.2.1.2 schematisch dargestellt, in ein Komponentensystem mithilfe unterschiedlicher Modellierungstechniken integriert werden können:

Übersetzer: Eine Übersetzer-Komponente vermittelt zwischen bereits existierenden Software-Elementen und anderen Komponenten eines Systems. Die Kopplung und Kommunikation mit anderen Komponenten findet dabei über die Schnittstelle des Übersetzers

statt, so dass dieser die Interpretation und Übersetzung zwischen Komponentensystem und Software-Element übernimmt.

Verpackung: Bei der Verpackung eines Software-Elements wird der Code des Software-Elements direkt erweitert. Im Gegensatz zu Übersetzer-Komponenten hat somit die verpackende Komponente direkten Zugriff auf die internen Datenstrukturen des Software-Elements.

Komponente: Selbstverständlich kann ein Software-Element auch neu implementiert werden, um innerhalb eines Komponentensystems integriert werden zu können. Auch wenn hierdurch gegenüber den beiden anderen Ansätzen die Performanz eines Systems gesteigert werden kann, erfordert dieser Ansatz jedoch einen hohen Implementierungsaufwand.

4.2.2 Dienste

4.2.2.1 Namens- und Verzeichnisdienste

Die Integration neuer Komponenten während der Laufzeit einer Ablaufumgebung erfordert, wie im vorherigen Abschnitt bereits erwähnt, die Verwaltung der innerhalb des Systems zur Verfügung stehenden Komponenten. Wobei im Gegensatz zu den aus der objektorientierten Implementierung bekannten direkten Verweisen auf Objekte die indirekte Referenzierung einzelner Komponenten zu unterstützen ist.

Die Verwaltung innerhalb eines Systems wird durch Namens- bzw. Verzeichnisdienste unterstützt, die die einzelnen Komponenten, vergleichbar mit einem Telefonbuch, mithilfe logischer Namen bzw. attributierter Namen referenzieren [Grif98]. Die Zuordnung des Namens zu einer konkreten Komponente erfolgt dabei während der Anmeldung einer Komponente bei einem Namens- bzw. Verzeichnisdienst. Hierdurch wird eine gewisse Entkopplung ermöglicht, da z.B. die konkret zugeordnete Implementierung einer Komponente ausgetauscht werden kann, obwohl der logische Name beibehalten wird:

Namensdienst: Namensdienste treten als reine Informationslieferanten auf, da die Auswertung der erhaltenen Information bzw. der Dienstleistungen der zugewiesenen Komponente vollständig von der anfordernden Komponente zu erbringen ist.

Verzeichnisdienst: Um einer Komponente nicht nur aufgrund eines Namens, sondern auch z.B. aufgrund bestimmter Eigenschaften einen Kommunikationspartner zuordnen zu können, stellen Verzeichnisdienste eine Erweiterung der Namensdienste dar. Ein Verzeichnisdienst unterstützt dabei die Speicherung weitergehender Informationen über eine angemeldete Komponente, indem Attribute mit zugehörigen Werten, so genannte attributierte Namen, verwaltet werden können.

4.2.2.2 Ereigniskanäle

Durch die im vorigen Abschnitt vorgestellten Namens- und Verzeichnisdienste kann eine Entkopplung von Komponenten, im Gegensatz zu der direkten Referenzierung eines Software-Elements, bereits erreicht werden. Allerdings beruht die eigentliche Kommunikation, die durch einen Softwarebus unterstützt wird, weiterhin auf einer engen 1:1-Beziehung zwischen Sender und Empfänger.

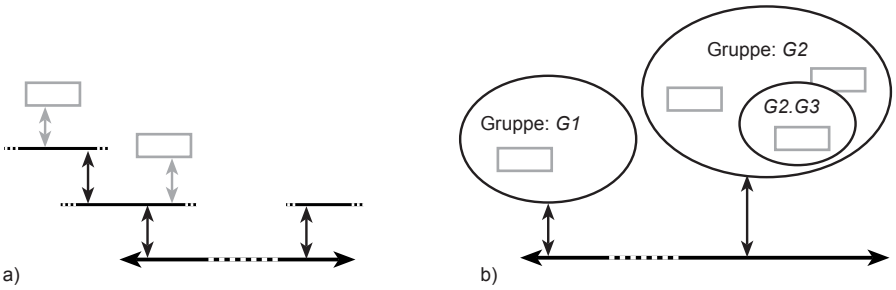


Abb. 4.2.2.2.1: Schematische Darstellung eines hierarchischen (a) und eines topologischen (b) Ereigniskanals.

Der wesentliche Unterschied zwischen Softwarebus und Ereigniskanal ist die Aufhebung einer 1:1-Beziehung zwischen Sender und Empfänger, da durch den Ereignisdienst mehrere Komponenten über einen Kanal kommunizieren können, so dass grundsätzlich eine 1:n-Verbindung zwischen Sender und Empfängern besteht. Neben diesen einfachen Ereigniskanälen lassen sich, wie in Abb. 4.2.2.2.1 schematisch dargestellt, zwei Typen von strukturierenden Ereigniskanälen unterscheiden [Grif98]:

Hierarchische Ereigniskanäle: Unterstützen einfache Ereigniskanäle ausschließlich 1:n-Verbindungen, an der alle angemeldeten Komponenten teilnehmen, kann durch eine hierarchische Strukturierung der Kanäle eine gezielte Kommunikation stattfinden. Durch einen hierarchischen Ereigniskanal wird eine Baumstruktur aufgebaut, so dass eine Sender-Komponente gezielt alle Empfänger eines Astes ansprechen kann. Die Ereignisse werden von einer Hierarchiestufe an die nächst tiefer liegende weitergeleitet, so dass alle Komponenten dieses Kanals das Ereignis erhalten.

Topologische Ereigniskanäle: Eine weitere Möglichkeit der Strukturierung bietet die Gruppenkommunikation. Hier melden sich Komponenten bei einem Ereigniskanal mit einem Bezeichner, der aus einzelnen Namenskomponenten zusammengesetzt sein kann, an. Die Ereignisse, die ebenfalls als Kennung einen zusammengesetzten Namen besitzen, werden durch einen topologischen Ereigniskanal ausschließlich an diejenigen Komponenten weitergeleitet, die sich mit dem gleichen Bezeichner bei dem Kanal angemeldet haben. Im Gegensatz zu den hierarchischen Ereigniskanälen können somit unterschiedliche und überlappende Kanäle aufgebaut werden.

Durch Ereigniskanäle wird gegenüber Namensdiensten eine stärkere Entkopplung von Komponenten unterstützt, da Ereignisse aus konzeptioneller Sicht als Auslöser einer Aktion bzw. Methode bei den Empfängern agieren. Somit können Komponenten getrennt voneinander entworfen werden, d.h. die bei dem objektorientierten Entwurf übliche Vererbung von Methoden wird durch das koordinierte Verhalten von Komponenten über festgelegte Ereignisse ersetzt.

4.2.3 Anwendungsbezogene Komponentensysteme

Seit Anfang der 70er Jahre werden für die Bedienung von Personalcomputern Oberflächen zur direkten Beeinflussung von digitalen Mediendaten eingesetzt. Mittlerweile hat sich die Schreibtischmetapher als Oberfläche durchgesetzt. Durch die visuelle Darstellung von Software- und Medien-Elementen, wie etwa die Darstellung von Mediendaten unterschiedlichen Typs, Ordnern zur Sammlung von Mediendaten, einem Papierkorb zum Löschen von Me-

diendaten usw., wird dem Benutzer ein direkter, aus der realen Welt vertrauter Zugang ermöglicht.

Um den Aufbau derartiger Oberflächen zu unterstützen, wurden spezielle Komponentenmodelle entwickelt, die die Kapselung des eigentlichen Inhalts unterstützen, indem sie eine von außen zu beeinflussende standardisierte Schnittstelle bereitstellen. Dienen diese so genannten Dokumentenmodelle dabei als „Baukastensystem“ für unterschiedliche Anwendungsszenarien, so werden durch spezialisierte Modelle, wie etwa durch *iShell* [iShe01], auf multimediale Aspekte eingeschränkte Systeme angeboten.

Durch die layoutbasierte Strukturierung derartiger Systeme werden diese jedoch in den meisten Fällen für die Umsetzung von Informationssystemen eingesetzt. Der Benutzer kann dabei über deskriptive Interaktionsformen innerhalb einer dokumentenähnlichen Struktur navigieren. Somit sind die zurzeit vorherrschenden Dokumentenmodelle nur sehr eingeschränkt für die Gestaltung von Wahrnehmungsumgebungen einsetzbar.

Dokumentenmodelle: Dokumentenmodelle sind eine spezielle Ausprägung des Komponentenparadigmas. Seit Mitte der 90er Jahre wurden für unterschiedliche Plattformen Dokument-Komponenten, wie *OpenDoc*, *OLE/DCOM* und *NeXT*, entwickelt [OrHE96]. Grundlage dieser Modelle ist die dokumentenzentrierte Sicht, bei der die Komponenten einzelne Teile eines Dokuments widerspiegeln, so dass sie als Umsetzung der in Abschnitt 3.3 beschriebenen layoutbasierten Präsentationsmodelle angesehen werden können.

Zentrale Komponente der Dokumentenmodelle ist ein so genannter Container, der mehrere Aufgaben erfüllt. Zunächst dient er als Behälter für die Komponenten, die die eigentlichen Daten enthalten. Derartige Komponenten ermöglichen die Darstellung und Bearbeitung des Inhalts. Weiterhin stellen die Container für die Einbettung der Komponenten eine gemeinsame Kommunikationsstruktur zur Verfügung, die die gegenseitige Interaktion der Komponenten unterstützt. Um die visuelle Struktur eines Dokuments nachzubilden, lassen sich Container, ebenso wie dies bei Dokumenten der Fall ist, rekursiv schachteln. Ein Container repräsentiert damit eine Schnittstelle eines Dokuments bzw. Teildokuments und stellt Funktionen bereit, die eine Hierarchiebildung unterstützen.

Neben diesen Grundmechanismen wird die Bindung bzw. Kommunikation, d.h. die Forderung nach der Integrationsfähigkeit und Anpassungsfähigkeit von Containern durch die Dokumentenmodelle unterstützt. Anhand von *OpenDoc* wird hier ein Ansatz vorgestellt, der die Definition von Komponenten-Schnittstellen, von semantischen Ereignissen und von auszuführenden Aktionen in einer benutzerverständlichen Sprache ermöglicht. Eine weitergehende Einführung findet man in [MaSu96].

Die Komposition verschiedener Komponenten kann, wie in Abschnitt 4.2.1 betrachtet, eine Erweiterung des Verhaltens erfordern. Bei OpenDoc spricht man von so genannten Gruppierungen, die mehrere Komponenten als Komposition zusammenfassen. Durch die Zuordnung vorgegebener Verhaltensbeschreibungen können dabei einer Gruppierung neue Eigenschaften zugeordnet werden. Die Kopplung anderer Komponenten mit der Gruppierung wird durch eine implizite selbstbeschreibende Schnittstelle der Verhaltensbeschreibungen unterstützt, die von anderen Komponenten abgefragt werden kann.

Die Kommunikation zwischen den Komponenten findet in OpenDoc über semantische Ereignisse statt, die abstrakte Aktionen repräsentieren, mit denen durch die Verhaltensbe-

schreibung spezifizierte Aktionen auf der Programm-, Dokument- und Datenebene angestoßen werden können. Um die gegenseitige Verständigung der Komponenten zu unterstützen, steht dabei ein festgelegtes Vokabular an Ereignissen zur Verfügung, durch das, neben der Bezeichnung des Ereignistyps und der Kennung des Empfängers, Attribute übergeben werden können, die durch die Aktionen des Empfängers interpretiert werden. Durch die Wahl gleicher semantischer Ereignisse lässt sich der aus der Objektorientierung bekannte Subtyp-Polymorphismus nachbilden, da durch die Ereignisse unterschiedliche Aktionen, z.B. in Abhängigkeit des Dokumenttyps oder des Anwendungsprogramms, in den Empfänger-Komponenten angestoßen werden können.

Um die Kommunikation zwischen Anwender und Komponenten zu unterstützen, bietet OpenDoc neben der Definition von semantischen Ereignissen einen Skriptinterpreter an, wie etwa für die Skriptsprache *AppleScript* [Gump96]. Mithilfe einer derartigen Skriptsprache kann der Anwender über eine „natürliche“ Beschreibung Aktionen einzelner Komponenten anstoßen. Der Interpreter setzt die Beschreibung in semantische Ereignisse um, so dass die eigentliche Kommunikation entsprechend der Kommunikation zwischen Komponenten verläuft.

iShell: Zwar bieten die Dokumentenmodelle wie OpenDoc bereits eine Infrastruktur für seitenbasierte Anwendungen, doch wird durch das hohe Abstraktionsniveau die Entwicklung multimedialer Anwendungen nicht hinreichend unterstützt. Als spezialisiertes Dokumentenmodell für multimediale Anwendungen kann die Entwicklungs- und Ablaufumgebung *iShell* [iShe91] angesehen werden, durch die bereits die Einbindung häufig verwendeter Medienformate und die Definition der wichtigsten Ereignis- und Aktionstypen in multimedialen Anwendungen unterstützt werden.

Entsprechend der abstrakten Dokumentenmodelle basiert iShell auf einer Containerstruktur, durch die sowohl einzelne Mediendaten als auch weitere Container gekapselt werden können, so dass die Container mit den in Abschnitt 2.2.1 eingeführten elementaren und komplexen Medienverwaltern vergleichbar sind. Durch einen Container bzw. Medienverwalter wird dabei sowohl das interne Verhalten und die Platzierung des verwalteten Mediums als auch eine Schnittstelle, die standardisierte auszuführende Aktionen des Verwalters bereitstellt, beschrieben.

Die Schnittstelle eines Medienverwalters setzt sich aus zwei Teilen zusammen. So werden durch die Schnittstelle bereits fest vorgegebene Aktionen des Medienverwalters bereitgestellt, die eine direkte Beeinflussung des verwalteten Mediums bewirken. Die angebotenen Aktionen sind dabei abhängig von dem Medientyp, so dass z.B. zeitkontinuierliche Mediendaten gestartet bzw. beendet werden können, wohingegen bei zeitdiskreten Mediendaten diese Aktionen nicht angeboten werden. Die bereitstehenden Aktionen werden, vergleichbar mit den abstrakten Dokumentenmodellen, durch eine implizite Schnittstellenbeschreibung anderen Medienverwaltern bekannt gegeben.

Um die Schnittstelle erweitern zu können, unterstützt diese zusätzlich das Empfangen semantischer Ereignisse, die während des Entwurfs einer multimedialen Anwendung durch die Verhaltensbeschreibung eines Medienverwalters frei definiert werden können. Dabei basiert die Kommunikation zwischen den Medienverwaltern auf den in Abschnitt 4.2.2.2 eingeführten hierarchischen Ereigniskanälen, die durch die Containerstruktur aufgebaut werden.

Das interne Verhalten eines Medienverwalters kann durch eine festgelegte einfache Beschreibungssprache definiert werden, durch die externe semantische Ereignisse und interne Interaktionsereignisse auf die Beeinflussung des verwalteten Mediums und das Versenden von semantischen Ereignissen abgebildet werden können. Hierbei stehen die in Abschnitt 2.2.2 beschriebenen internen Ereignis- und Aktionstypen zur Verfügung, so dass z.B. durch unterschiedliche Benutzeraktionen interne Aktionen angestoßen werden können, die zeitliche, räumliche und gestalterische Auswirkungen definieren.

4.3 Agentenorientierte Softwareumgebungen

Ebenso wie bei der Beschreibung von Komponenten, sind unterschiedliche Definitionen über den Agentenbegriff in der Literatur zu finden [NwNd96][FrGr96][JeSW98] [ABJL98]. Dabei lassen sich Übereinstimmungen mit dem Komponentenbegriff feststellen, wie etwa in [Laur90], wo ein Agent als eigenständiges Software-Element angesehen wird, das Aktionen ausführt. Diese Übereinstimmung wird auch deutlich, wenn der aus der englischen Sprache stammende Begriff *agent*, nicht wie üblich als Agent, sondern präziser als Dienstleister bzw. Agentur übersetzt wird.

Um den Agentenbegriff verständlich zu machen, werden hier unterschiedliche Aspekte von Agenten betrachtet, wobei der Schwerpunkt der Darstellung auf den für diese Arbeit relevanten Aspekten so genannter mobiler Agenten und der Agenten-Kommunikation liegt. Abschließend findet eine Einordnung anwendungsorientierter Agentensysteme in unterschiedliche multimediale Bereiche statt.

4.3.1 Agentenparadigma

Das Agentenparadigma basiert, ebenso wie dies bei dem Komponentenparadigma der Fall ist, auf der Modellierung anpassungsfähiger, eigenständiger und abgrenzbarer Software-Elemente. Jedoch wird bei der Entwicklung von Agenten dieses Paradigma spezialisiert, indem sowohl die Interaktion mit bzw. zwischen Agenten als auch das Verhalten charakterisiert werden. Grundlage des Paradigmas ist die Zuordnung von „menschlichen“ Eigenschaften, die das interne und externe Verhalten von Agenten beschreiben [Bate94] [Shoh97a].

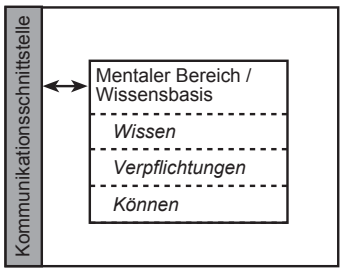


Abb. 4.3.1.1: Modellierung von Agenten.

Der Aufbau eines Agenten lässt sich, wie in Abb. 4.3.1.1 dargestellt, somit in zwei Bereiche gliedern. Der so genannte mentale Bereich beschreibt den Zustand [Shoh97a] [GrCl98] [WoJe95a] [WoJe95b] und das Verhalten [Broo90] [Broo91] [GrCl98] eines Agenten. Durch die Schnittstelle, die als Abstraktion einer Komponentenschnittstelle zu betrachten ist,

werden die Kommunikationseigenschaften eines Agenten beschrieben [FiLM97] [Shoh97b] [FIPA97].

Da der mentale Bereich den Zustand und das Verhalten eines Agenten widerspiegelt, ist er zunächst mit den Parametern und Methoden von Objekten vergleichbar. Jedoch werden, wie der Name schon andeutet, die Zustände eines Agenten durch „menschliche“ Eigenschaften beschrieben. So wird in [CoLe90] zwischen Glauben/Vorstellungen und Entscheidungen unterschieden. In anderen Arbeiten werden feinere Untergliederungen vorgestellt, wie etwa in [Shoh97b] eine Aufteilung in Glauben/Vorstellungen, Verpflichtungen, Können und Entscheidungen.

Der Bereich des Glaubens bzw. der Vorstellungen repräsentiert das Wissen eines Agenten über seine Umwelt, d.h. ein Agent besitzt ein explizites bzw. implizites Modell seiner Umwelt, das seine subjektive Sicht widerspiegelt. Unterteilt man dieses Modell, können die Verpflichtungen und das Können eines Agenten als separate Eigenschaften betrachtet werden. Durch diese Metapher lassen sich die Sicht auf die Umwelt eines Agenten, die sich durch Sinneseindrücke ergibt, Verpflichtungen, die durch Kooperation mit der Umwelt eingegangen werden, und das Können, das die Möglichkeiten eines Agenten mit der Umwelt zu interagieren beschreibt, unterscheiden.

Um eine lose Kopplung auch innerhalb heterogener Systeme zwischen einzelnen Agenten zu ermöglichen, basiert die Kommunikation häufig auf speziellen Koordinations- bzw. Kommunikationssprachen, die von der Schnittstelle unterstützt werden. Dient eine derartige Sprache bei den Komponenten, wie in Abschnitt 4.2.1 gezeigt, zur Erweiterung der Schnittstelle durch semantische Ereignisse, wird bei dem Agentenparadigma häufig eine spezielle Sicht der Kommunikation eingeführt. Auch hier werden wieder „menschliche“ Eigenschaften, diesmal bezogen auf die Kommunikation, durch so genannte Sprechakte, die in Abschnitt 4.3.3.2 noch näher betrachtet werden, betont.

Je nach Ausprägung der einzelnen Bereiche lassen sich Agenten durch ihre „menschlichen“ Eigenschaften, die sich aus der Autonomie, Kooperation und der Lernfähigkeit von Agenten zusammensetzen, charakterisieren [Nwan96]:

Autonomie: Autonome Agenten können selbstständig Handlungen anstoßen, ohne von einem Benutzer spezielle Anweisungen zu bekommen [Maes91], so dass autonome Agenten die Fähigkeit besitzen, selbst aktiv zu werden. Zudem kann gefordert werden, dass die Handlungen die Umgebung direkt bzw. indirekt beeinflussen, so dass der autonome Agent hervorgerufene Änderungen selbstständig erfassen kann [FrGr96].

Kooperation: Durch die Kooperation von Agenten kann ihr „soziales Verhalten“ charakterisiert werden, wobei die Kooperation auf verschiedenen Ebenen zwischen einem Agenten und einem Benutzer, zwischen mehreren Agenten oder zwischen einem Agenten und der Ablaufumgebung stattfinden kann. Die verschiedenen Ebenen der Kooperation bzw. Kommunikation können dabei durch die zur Verfügung stehenden Sprechakte unterstützt werden, die durch eine Kommunikationssprache vorgegeben werden [MaLF95].

Lernfähigkeit: Lernfähige Agenten erweitern ihr Wissen bzw. verändern ihr Verhalten durch das „Erkennen“ ihrer Umwelt und können somit ihre Reaktionen bzw. ihr Kooperationsverhalten aktuellen Randbedingungen anpassen.

Neben den „menschlichen“ Eigenschaften werden häufig weitere Charakteristika von Agenten betrachtet, die sich auf das Verhalten von Agenten beziehen [WoJe95b] [Brad97b]:

Abwägend/Reaktiv: Abwägende Agenten besitzen ein internes symbolisches Modell ihrer Umgebung, das zur Planung und zur Kooperation herangezogen wird, so dass sich mehrere Agenten koordinieren können. Dagegen basiert das Verhalten reaktiver Agenten lediglich auf einem Reiz/Reaktionsmechanismus, der eventuell durch den internen Zustand des Agenten beeinflusst wird.

Mobil/Stationär: Man unterscheidet zwischen mobilen Agenten, die sich in einer verteilten Umgebung bewegen können, und stationären Agenten, die an eine lokale Umgebung gebunden sind. Wie in Abschnitt 4.3.2.1 noch näher gezeigt wird, können mobile Agenten des weiteren durch die Art der Übermittlung charakterisiert werden, bei der der gesamte Agent oder ausschließlich Teile des mentalen Bereichs transferiert werden [GiAA95].

Sicherlich treten die einzelnen Charakteristika bei Agenten nicht ausschließlich getrennt auf, so dass durch unterschiedliche Ausprägungen der Eigenschaften bzw. des Verhaltens vielmehr ein mehrdimensionaler Raum aufgespannt wird [ABJL98]. Wie in Abschnitt 4.3.3 gezeigt wird, lassen sich dabei einzelne Agentenmodelle unterschiedlichen multimedialen Bereichen zuordnen.

Durch die Charakteristika wird aber nicht nur ein Rahmen für die Zuordnung von Agenten in typische Anwendungsbereiche vorgegeben, sondern sie lassen sich auch aus Sicht der Implementierung verstehen. So können während der Implementierung zum Teil unterschiedliche Programmiersprachen zum Einsatz kommen, die spezielle Vorzüge für die Beschreibung aufweisen. Dabei unterscheidet man, vergleichbar mit der komponentenorientierten Realisierung, unterschiedliche Sprachebenen. Erstens sind dies Sprachen für die eigentliche Implementierung von Agenten. Zweitens können eventuell eigene Sprachen für die Beschreibung des mentalen Bereichs eingesetzt werden. Um dabei unabhängig von der eigentlichen Realisierung den mentalen Bereich aus funktionaler Sicht nach außen zu beschreiben, spricht man von einer virtuellen Wissensbasis, auf der Operationen zur Manipulation definiert sind. Drittens unterstützt häufig die Schnittstelle eines Agenten, wie bereits weiter oben erwähnt, spezielle Koordinations- bzw. Kommunikationssprachen. Durch diese Trennung zwischen Kommunikation und interner Repräsentation können wiederum auch externe Software-Elemente, entsprechend den in Abschnitt 4.2.1 bereits beschriebenen Modellierungstechniken, innerhalb eines Agentensystems mithilfe „übersetzender“ oder „verpackender“ Agenten integriert werden [Dale98].

4.3.2 Dienste

4.3.2.1 Migration

Wie im vorherigen Abschnitt gezeigt, ist die Mobilität ein Charakteristikum von Agenten, die das selbstständige Aufsuchen verschiedener lokaler Ablaufumgebungen beschreibt. Im Gegensatz zu anderen Software-Elementen, können mobile Agenten hierbei auch während ihrer Ausführung unterschiedliche Ablaufumgebungen aufsuchen. Auch wenn die Mobilität bei Agentensystemen durch unterschiedliche Funktionalitäten unterstützt wird, basiert diese auf einer einheitlichen Sichtweise, die als so genannte Agenten-Migration den Transfer zwischen lokalen Anwendungen, wie in Abb. 4.3.2.1.1 dargestellt, beschreibt.

Zu unterscheiden ist die Migration bei Prozessen und die Migration bei Agenten durch die Art der Initiierung, d.h. wer entscheidet wann und wohin der Transfer durchgeführt wird [Dale98]. Bei der Prozess-Migration wird der Transfer durch die Ablaufumgebung angestoßen, um z.B. die Lastverteilung in einem verteilten System zu steuern. Dagegen wird die

Agenten-Migration von einem mobilen Agenten selbst initiiert, der somit selbst entscheidet wann und zu welcher Ablaufumgebung der Transfer stattfindet, wobei die Ablaufumgebung ausschließlich die Migration durch einen entsprechenden Dienst anbietet und ausführt.

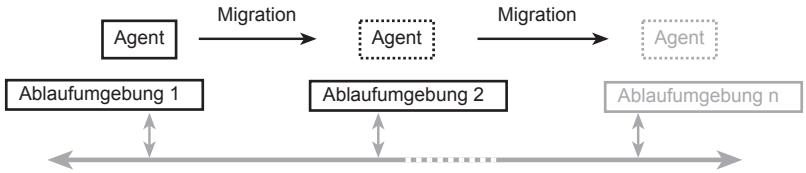


Abb. 4.3.2.1.1: Migration eines mobilen Software-Agenten.

Durch diese Sichtweise der Agenten-Migration wird, wie bereits in Abschnitt 4.1.1 erwähnt, die typische Client/Server-Kommunikation aufgehoben, bei der einzelne Software-Elemente entweder nur einen Dienst anfordern bzw. nur einen Dienst erbringen können. Mobile Agenten können dagegen als gleichberechtigte Elemente einer verteilten Umgebung angesehen werden, die sowohl Dienste innerhalb einer lokalen Umgebung anfordern als auch Dienste erbringen. So kann ein Agent unterschiedliche Ablaufumgebungen aufsuchen, und selbst als aktives Software-Element in der lokalen Umgebung fungieren [HaCK95].

Ein mobiler Agent tritt dabei als eigenständiger Prozess in einer verteilten Umgebung auf, so dass man bei der Migration unterschiedliche zu transferierende Aspekte eines Agenten zu betrachten hat. Erstens kann der Code eines Agenten (das Programm selbst) übertragen werden, zweitens seine Daten und drittens sein aktueller Zustand. Je nach Ausprägung des Dienstes eines Agentensystems unterscheidet man grundsätzlich zwischen orthogonaler Migration, bei der ein automatischer Transfer aller Aspekte des Agenten stattfindet, und nicht-orthogonaler Migration, bei der die zu transferierenden Aspekte durch den Agenten selbst beschrieben werden [Dale98].

4.3.2.2 Kommunikationssprachen

Durch die Mobilität von Agenten ist die lose Kopplung zwischen einzelnen Agenten von entscheidender Bedeutung. Zwar führt die Kommunikation zwischen Komponenten mithilfe semantischer Ereignisse, wie in Abschnitt 4.2.2.2 gezeigt, bereits zu einer weitgehenden Entkopplung einzelner Software-Elemente, allerdings wird durch die Ereignisse keine Trennung zwischen eigentlichem Inhalt der Kommunikation und dem Kommunikationsakt unterstützt. Somit besitzen semantische Ereignisse eine Doppelbedeutung, da durch sie zum einen Parameter übergeben werden, zum anderen der Empfänger zu einer auszuführenden Aktion angestoßen wird.

Um eine Trennung zwischen Inhalt und Kommunikationsakt zu unterstützen, werden bei der Agenten-Kommunikation häufig spezielle Sprachen eingesetzt, durch die die grundsätzlichen Kommunikationseigenschaften festgelegt werden können. Eine Kommunikationssprache sollte dabei folgende Punkte berücksichtigen [FiLM97]:

Form: Die Sprache sollte eine einfache Syntax haben, die mit geringem Aufwand von den Agenten bearbeitet werden kann. Sie sollte für den Programmierer verständlich sein und kurze Mitteilungen erzeugen, um den Transportaufwand gering zu halten. Neben dieser grundsätzlichen Form ist die Erweiterbarkeit der Sprache zu berücksichtigen, damit sie in unterschiedlichen Systemen bzw. Anwendungen zum Einsatz kommen kann.

Inhalt: Man unterscheidet zwischen der Kommunikationssprache, welche die Grammatik sowie das Vokabular des Kommunikationsakts definiert und der Sprache, in welcher der Inhalt einer Nachricht vorliegt. Die beiden können, müssen aber nicht identisch sein. Durch die unabhängige Sprachdefinition für die Inhaltsbeschreibung kann die Kommunikationssprache in unterschiedlichen Anwendungsszenarien eingesetzt werden.

Implementierung: Im Hinblick auf die Bearbeitung und die Netzauslastung muss die Implementierung effizient sein. Dabei ist zu berücksichtigen, dass die Schnittstelle einfach zu benutzen ist und Details der darunter liegenden Netzwerkschichten vor dem Programmierer versteckt werden. Einfache Agenten sollten nicht gezwungen sein alle Nachrichtentypen zu verarbeiten, sondern sich auf eine Untermenge beschränken können.

Netzwerk: Die Sprache sollte unterschiedliche Verbindungstypen wie 1:1- und 1:n-Verbindungen für die synchrone und asynchrone Kommunikation unterstützen. Die Verbindungstypen sind dabei als höherschichtige Protokolle aufzufassen, so dass von tieferen Protokollschichten abstrahiert werden kann.

Zuverlässigkeit: Zuverlässigkeit und Sicherheit sind zwei wesentliche Kriterien für eine Kommunikationssprache. Sie sollte robust gegen falsch gebildete und unbrauchbare Nachrichten sein sowie Mechanismen zur Erkennung und Signalisierung von Fehlern und Warnungen bieten.

Grundsätzlich können Kommunikationssprachen in prozedurale Sprachen [Whit97] [McCl95], und in deklarative Sprachen [FiLM97] [Shoh97b] unterteilt werden [GeKe94]. Durch prozedurale Sprachen findet die Kommunikation mithilfe von eigenständigen Programmen statt, d.h. die Software-Elemente verschicken untereinander Programme. Dieser sehr effektive und flexible Ansatz unterstützt zwar die Definition komplexer Aktionen, die durch den Empfänger auszuführen sind, dies wird jedoch durch die Verletzung einer klaren Trennung zwischen externer Schnittstelle und internem Zustand der Agenten erkauft. Gerade hierdurch ergibt sich für prozedurale Sprachen eine enge Kopplung, so dass sie nicht näher betrachtet werden.

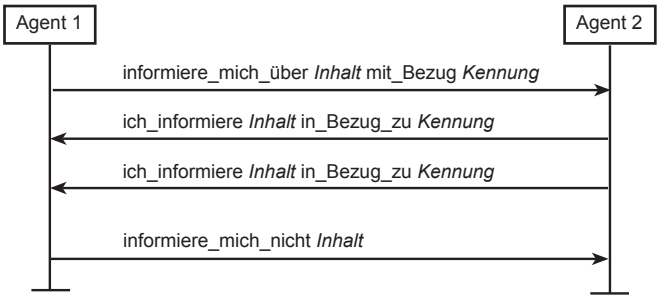


Abb. 4.3.2.2.1: Beispiel eines Sprechakts zwischen zwei Agenten.

Im Gegensatz zu prozeduralen Sprachen werden durch deklarative Sprachen Informationen und deren Austausch in Form von Aussagen beschrieben. Durch Aussagen lassen sich Definitionen, Annahmen, Anforderungen in einer formalen Logik [Shoh97b] bzw. in einer höheren Sprache [FiLM97] beschreiben. Unter Berücksichtigung der oben angesprochenen Anforderungen an eine Kommunikationssprache haben höhere Sprachen den Vorteil der

Verständlichkeit für den Programmierer und der klaren Trennung zwischen Kommunikationsakt und Inhalt.

Höhere Kommunikationssprachen basieren dabei auf so genannten Sprechakten, die als Abstraktion typischer Gesprächssituationen, wie in Abb. 4.3.2.2.1 anhand eines Beispiels dargestellt, aufzufassen sind. Um eine Abstraktion von Gesprächssituationen zu ermöglichen, werden diese durch die Auswirkungen auf den „Gesprächspartner" bzw. durch die Reaktion des „Gesprächspartners" charakterisiert und entsprechend der Auswirkungen bzw. Reaktionen durch Aktionstypen, wie z.B. Anweisungen, Anfragen und Antworten, klassifiziert.

Schlüsselwort	**Bedeutung**
content	Inhalt der Nachricht
language	Name derjenigen Sprache, in der der Inhalt beschrieben ist
ontology	Name der Ontologie, die von der Sprache benutzt wird
sender	Absender der Nachricht
receiver	Empfänger der Nachricht
reply-with	Kennung, mit der auf die Nachricht reagiert werden soll
in-reply-to	Kennung der Nachricht, auf die sich diese Nachricht bezieht

Tab. 4.3.2.2.1: Zusammenstellung der reservierten Parameter-Schlüsselwörter in KQML.

Als höhere Kommunikationssprache hat sich in den letzten Jahren die Sprache *Knowledge Query and Manipulation Language (KQML)* [MaLF96] [FiLM97] für die Agenten-Kommunikation als Quasi-Standard etabliert. KQML basiert auf einem einfachen semantischen Modell, bei dem Nachrichten zum Austausch virtueller Wissensbasen verwendet werden. So können durch spezielle Sprechakte Teile der virtuellen Wissensbasen abgefragt, ergänzt und gelöscht werden. Durch die Form von KQML lassen sich, wie in Tab. 4.3.2.21 zusammengefasst, der Kommunikationsbezug, der Inhalt sowie der Typ des Sprechakts getrennt beschreiben.

Um den Bezug der Kommunikation zu beschreiben, werden die Parameter, wie Absender (sender) und Empfänger (receiver), einer Nachricht definiert. Neben diesen grundlegenden Informationen, die für die Identifikation und Weiterleitung der Nachricht verwendet werden, können Sprechakte durch einen Bezugskontext (reply-with, in-reply-to) spezifiziert werden, so dass Sender und Empfänger auf die vorangegangene Kommunikation Bezug nehmen können.

KQML bietet für die Definition des Inhalts ausschließlich einen Rahmen, so dass der Inhalt (content) in einer für den Anwendungskontext geeigneten Sprache (language) definiert werden kann. Diese Trennung beinhaltet nicht nur die Syntax und Semantik des Inhalts, sondern ebenso die zu verwendenden Datentypen. Um dabei die Kommunikation mit unterschiedlichen Sprachen unterstützen zu können, kann eine Ontologie angegeben werden, durch die die Semantik des Vokabulars der Inhaltsbeschreibung festgelegt wird.

Kategorie	Typ	Schlüsselwörter
Benachrichtigung	Information	tell, deny, untell
	Fähigkeiten	advertise
Beeinflussung	Wirkung	achieve, unachieve
	Datenbank	insert, delete, delete-one, delete-all
Frage-Antwort	Einfach	ask-if, ask-about, ask-one, ask-all, evaluate, reply
	Mehrfach	stream-about, stream-all, eos
	Protokoll	standby, ready, next, rest, discard, generator
	Beobachtung	subscribe, monitor
	Rückmeldung	error, sorry
Netzwerk	Verbindung	forward, broadcast, pipe, break
	Verwaltung	register, unregister, transport-address

Tab. 4.3.2.2.2: Zusammenstellung der reservierten Nachrichten-Schlüsselwörter in KQML.

Durch die Nachrichtenart wird der Aktionstyp, der in KQML durch Schlüsselwörter definiert wird, im Sinne der Sprechakt-Theorie festgelegt. Die durch KQML definierten Sprechakte lassen sich, wie in Tab. 4.3.2.2.2 dargestellt, in unterschiedliche Kategorien aufteilen, die von der einfachen Informationsvermittlung über Anfragen und Antworten bis hin zu Verwaltungsaktionen reichen. Da die Beschreibung der Semantik aller Sprechakte den Rahmen dieser Arbeit sprengen würde, werden hier nur die verschiedenen Kommunikationskategorien vorgestellt [FiLM97]:

Benachrichtigung: Durch diese Sprechakte wird die Übermittlung von Nachrichten unterstützt, auf die der Sender keine Antwort von dem Empfänger erwartet. So können z.B. aktuelle Zustände der virtuellen Wissensbasis des Senders übermittelt (tell) werden. Um die Kommunikationseigenschaften in Bezug auf mögliche KQML-Sprechakte einem Kommunikationspartner zu übermitteln, steht einem Sender die Bekanntgabe (advertise) der unterstützten Sprechakte zur Verfügung. Wie bereits weiter oben gefordert, kann somit sichergestellt werden, dass Software-Agenten, die ausschließlich eine Teilmenge von KQML implementieren, eine erfolgreiche Kommunikation aufbauen können.

Beeinflussung: Beeinflussende Sprechakte beziehen sich direkt auf die virtuelle Wissensbasis des Empfängers. Im Gegensatz zu den Benachrichtigungen, durch die der Sender seinen eigenen Zustand bzw. seine Kommunikationsfähigkeiten übermittelt, werden durch diese Sprechakte interne Zustände des Empfängers beeinflusst. Zum einen kann der Empfänger aufgefordert werden, den Inhalt der Nachricht zu erfüllen (achieve) bzw. nicht zu erfüllen (unachieve). Zum anderen können durch datenbankverwandte Sprechakte (insert, delete, etc.) direkte Änderungen in der Wissensbasis des Empfängers verursacht werden.

Frage-Antwort: In dieser Kategorie sind die Sprechakte zusammengefasst, die verschiedene Ausprägungen typischer Frage-Antwort-Szenarien definieren. Damit die Antwort auf

die Frage von dem Fragenden identifiziert werden kann, wird die Anfrage durch eine Kennung (reply-with *Kennung*) gekennzeichnet. Der Empfänger der Anfrage fügt der Antwort diese Kennung (in-reply-to *Kennung*) an. Durch einfache Fragen kann der Sender die virtuelle Wissensbasis des Empfängers abfragen, woraufhin der Empfänger antwortet (reply) bzw. eine Fehlermeldung (error, sorry) zurückliefert. Auch wenn dieser einfache Sprechakt ein Frage-Antwort-Szenarium vollständig beschreibt, stehen in KQML komplexere Frage-Antwort-Szenarien zur Verfügung, um die weiter oben angesprochenen Übertragungskosten der Kommunikation zu verringern. So können z.B. mehrere Antworten (stream-about) auf eine Frage angefordert werden oder Veränderungen der Wissensbasis des Empfängers beobachtet (monitor) werden.

Netzwerk: Die Sprechakte dieser Kategorie bieten die bereits weiter oben geforderten unterschiedlichen Verbindungstypen für die Kommunikation an. So kann der Sender Nachrichten weiterleiten (forward), die Verteilung einer Nachricht an mehrere Empfänger hervorrufen (broadcast) und den Empfänger zum Weiterleiten (pipe) bzw. zum Beenden des Weiterleitens (break) von Nachrichten veranlassen. Neben diesen Verbindungstypen dienen die weiteren Sprechakte zur Anmeldung und Abmeldung bzw. Abfrage der bereits weiter oben beschriebenen Namensdienste.

4.3.3 Anwendungsbezogene Agentensysteme

Auch wenn durch die Mobilität und die Agenten-Kommunikationssprachen, wie bereits in Abschnitt 4.1.1 erwähnt, die Eigenständigkeit von Agenten in verteilten Umgebungen unterstützt wird, beschränken sich die aus der Literatur bekannten anwendungsorientierten Agentensysteme ausschließlich auf einzelne Aspekte multimedialer Anwendungen. Somit können sie nicht als abstraktes flexibles „Baukastensystem“ einer offenen multimedialen Ablaufumgebung eingesetzt werden. Vielmehr stellen sie bereits für einzelne Anwendungsbereiche spezialisierte Umgebungen dar und sind daher nicht als flexible Architektur für die Gestaltung von Wahrnehmungsumgebungen geeignet.

Um trotzdem einen Überblick über einzelne Aspekte multimedialer Anwendungen geben zu können, werden hier Agentensysteme aus Sicht ihres Einsatzgebiets betrachtet. Im Gegensatz zu den in Abschnitt 4.2.3 vorgestellten anwendungsbezogenen Komponentensystemen werden durch Agentensysteme häufig narrative multimediale Anwendungen realisiert. Dies ist zum einen auf die Kommunikations- bzw. Interaktionsformen, und zum anderen auf die visuelle (figurative) Repräsentation zurückzuführen. Somit wird durch so genannte Interface-Agenten die Kommunikation zwischen Benutzer und System betont, bei Informationsagenten der Umgang mit verteilten Informationen innerhalb eines heterogenen Systems betrachtet, durch Simulationsagenten das Verhalten komplexer Systeme innerhalb einer geschlossenen Anwendung modelliert und durch Entertainment-Agenten die Modellierung virtueller „Schauspieler“ unterstützt:

Interface-Agenten: Wie in Abschnitt 4.2.3 bereits gezeigt, dienen Dokumentenmodelle als Grundlage gängiger Benutzeroberflächen, die auf der Schreibtischmetapher basieren. Auch wenn dadurch die direkte Beeinflussung von Elementen ermöglicht wird und dem Benutzer eine aus der realen Welt vertraute Oberfläche zur Verfügung steht, so werden immer komplexer werdende Aufgaben durch diese einfache Form der Interaktion nur eingeschränkt unterstützt. Dabei sind das Bearbeiten großer Datenbestände, das verzögerte Ausführen von Aktionen, die Kombination mehrerer Aktionen und die ausschließlich funktionale Sicht auf Elemente ungelöste Probleme der Schreibtischmetapher [Brad97b].

Der Einsatz von Interface-Agenten ermöglicht den Übergang von der direkten zur indirekten Beeinflussung digitaler Daten. Dadurch wird erreicht, dass der Anwender von den einzelnen Schritten einer Bearbeitung abstrahieren kann, indem er diese Aufgabe an einen Interface-Agenten delegiert [Kay90] [Negr90] oder indem Interface-Agenten als so genannte persönliche Assistenten fungieren, die neben der reinen Bearbeitung von Aufträgen die Angewohnheiten, Vorlieben und Interessen des Anwenders lernen und somit eigenständig Aufgaben verrichten [Maes97]. In den meisten Arbeiten werden die Suche nach bzw. das Sortieren von Informationen durch Interface-Agenten unterstützt [Maes97] [DaWR95] [Whit97], so dass ein fließender Übergang zwischen Interface-Agenten und Informationsagenten, die im Anschluss betrachtet werden, besteht.

Neben der Suche bzw. dem Sortieren von Informationen werden lernende Interface-Agenten als künstliche Tutoren in Lernumgebungen bzw. Anwendungsprogrammen eingesetzt [RiJo97] [JRSM98] [LCKB97] [StLe97]. Der Agent führt den Benutzer je nach Wissensstand durch den zu vermittelnden Inhalt bzw. hilft bei der Bedienung von Programmen, wobei die Wissensbasis des Agenten, aus der Antworten und Hinweise für den Anwender generiert werden, das jeweilige Themengebiet als symbolisches Modell repräsentiert.

Die Kommunikation zwischen Anwender und Interface-Agent beruht auf der in Abschnitt 3.2.2.2 bereits vorgestellten Theatermetapher [Laur90]. Neben der dort vorgestellten Modellierung von Interface-Agenten als steuernde Software-Elemente einzelner Mediendaten [Ishi97], werden häufig Charaktere als Darstellungsform eines Agenten betrachtet, um den aktuellen Zustand des Agenten als Emotion zu repräsentieren [BaLK97] [Maes97] [JRSM98] [LCKB97].

Ein Interface-Agent tritt somit als künstliches (abstraktes) Individuum auf, das zum einen seinen internen Zustand durch sein Erscheinungsbild widerspiegelt und zum anderen die Kommunikation mit dem Benutzer auf „natürliche“ Weise unterstützt. Wie bereits gesagt, besteht dabei ein fließender Übergang zwischen Interface-Agenten und Informationsagenten. Betrachtet man jedoch alle Ausprägungen von Interface-Agenten, können sie nach Abschnitt 4.3.1 als stationäre Agenten angesehen werden, im Gegensatz zu den Informationsagenten, die sich gerade durch ihre Kooperation mit anderen Agenten und durch ihre Mobilität auszeichnen.

Informations-/Internetagenten: Informationsagenten werden zur Suche bzw. Bearbeitung von Daten in großen Datenbeständen eingesetzt, die in den meisten Fällen auf unterschiedlichen Plattformen verteilt sind. Dabei stellt das Internet das wichtigste Übermittlungsmedium für Informationsagenten dar, so dass Informationsagenten häufig auch als Internetagenten bezeichnet werden. Informationsagenten besitzen die Fähigkeit, mit unterschiedlichen Dokumenttypen umgehen zu können, um eine zielgerechte Zusammenstellung der vom Anwender gewünschten Informationen bereitzustellen. Somit hat ein Informationsagent typische Charakteristika eines Interface-Agenten, da auch hier die Beauftragung durch den Benutzer und die Darstellung der Ergebnisse eine entscheidende Rolle spielen.

Hauptcharakteristikum von Informationsagenten ist jedoch die Mobilität. Die Agenten bewegen sich von einer Plattform zur nächsten, um die entsprechende Aufgabe zu lösen. Durch den Transport des Agenten über mehrere Plattformen kann zum einen erreicht werden, dass die zeitaufwendige Netzkommunikation reduziert wird, da der Agent mit den Plattformen, die die Informationen bereitstellen, direkt kommuniziert und erst nach der

Beendigung seiner Aufgabe zu dem Anwender zurückkehrt. Zum anderen wird durch die Mobilität der Agenten erreicht, dass der Anwender während der Bearbeitung seines Auftrags keine Verbindung zu den entfernten Plattformen aufrecht erhalten muss, sondern zu einem beliebigen Zeitpunkt die Ergebnisse abfragen kann.

Anfang der 90er Jahre entwickelte die Firma *GeneralMagic* einen tragbaren Informationsassistenten mit einer Agentenplattform. Der Anwender kann mithilfe der Beschreibungssprache *Telescript* [Whit97] Agenten als Händler und Informationsvermittler programmieren, die das Internet als Übermittlungsmedium nutzen. Telescript unterstützt dabei die wesentlichen Sprachkonstrukte für den elektronischen Handel und die Informationssuche, so dass der Anwender in einer einfach gehaltenen englischen Sprache die Aufgaben der Agenten beschreiben kann. Bezieht sich hierbei die Suche auf den Informationsinhalt, können Informationsagenten auch für die Suche spezifischer Datenformate eingesetzt werden, die aufgrund von Übertragungsleistungen oder unterstützten Präsentationsformen bestimmt werden. Hierbei dienen z.B. HTML-Dokumente als Schablone, in die Informationsagenten die entsprechenden Mediendaten einfügen [FaKa97a] [FaKa97b].

Neben der alleinigen Suche nach benutzerspezifischen Informationen werden Informationsagenten auch für die Verwaltung innerhalb einer heterogenen verteilten Umgebung eingesetzt [DaDe97] [Dale98]. Die Ablaufumgebung stellt dabei ausschließlich einen Rahmen zur Verfügung, in den unterschiedliche Programme für die Ansicht und Bearbeitung der Mediendaten eingebunden werden können. Allerdings wird durch diesen Ansatz keine geeignete Darstellung multimedialer Inhalte unterstützt, da die Mediendaten lediglich als separate Elemente innerhalb der ausgewählten Programme angezeigt werden.

Simulationsagenten: Typischer Ansatz bei der Simulation ist die objektorientierte Umsetzung der Modellierung. Dabei beschreiben die Modelle häufig technische Prozesse, deren einzelne Elemente und deren Zusammenwirken durch mathematische Formeln definiert sind. Betrachtet man komplexe Systeme, in denen das Zusammenwirken unterschiedlicher autonomer Objekte im Vordergrund steht, wie dies etwa bei der Modellierung soziologischer Systeme der Fall ist, werden häufig kooperierende Agenten als Nachbildung der Objekte für die ereignisorientierte Simulation eingesetzt [Uhrm97] [UhSc98]. Durch die agentenorientierte Umsetzung können Simulationsumgebungen geschaffen werden, die ein breites Spektrum an geologischen [GMQG96] [Marc98], biologischen [SoMa99], soziologischen [Uhrm96] und ökonomischen [Well96] [WaWe99] [WhPa98] [Tesf97] Modellen abdecken.

Abgeleitet von den beiden in der Soziologie vorherrschenden Modellen, in denen zum einen das Verhalten einer Gruppe auf einfache Beziehungen zwischen den Individuen nachgebildet wird bzw. Individuen als eigenständige Lebewesen betrachtet werden, werden Simulationsagenten als reaktive Agenten bzw. als abwägende Agenten (mit einem internen Modell ihrer Umgebung) modelliert [Uhrm96].

Der erste Ansatz wird häufig bei der Modellierung komplexer Systeme eingesetzt, deren Dynamik auf das Zusammenwirken einzelner Teile zurückzuführen ist. So werden z.B. Simulationsagenten zur Modellierung geophysikalischer Prozesse, wie etwa zur Modellierung der Lavabewegung eines Vulkans, eingesetzt. Dabei wird der Prozess als selbstorganisierendes kritisches System betrachtet, das durch zwei Arten von Agententypen beschrieben wird, um zum einen das Mikroverhalten der Elemente und zum anderen globale physikalische Gesetze als makroskopische Sicht zu beschreiben [GMQG96] [Marc98]. Weitere Ansätze reaktiver Agenten werden bei der Simulation von Schwärmen

[BoDT99] eingesetzt bzw. für die Modellierung mobiler Agenten, deren Kommunikationsmodell auf dem von Ameisen bekannten chemischen Reiz-Reaktionsmuster basiert [WhPa98].

Abwägende Agenten werden z.B. für die so genannte marktorientierte Programmierung eingesetzt [Well96] [WaWe99], die als Programmierparadigma verwendet wird, um eine Ressourcenverteilung auf der Grundlage des Käufer/Verkäufer-Modells zu simulieren. Die Akteure dieses Modells werden dabei als eigenständige Agenten umgesetzt, die durch Kooperation mit den Ressourcen handeln.

Auch wenn die vorgestellten Simulationsmodelle auf unterschiedlichen Agententypen aufbauen, besteht aus Modellierungssicht der größte Unterschied in der Beschreibung der Kooperation. Der Vorteil von Simulationsagenten ist hierbei, dass die Kooperation in einer für das jeweilige Fachgebiet „natürlichen" semantischen Form definiert werden kann. Allerdings ist die Umsetzung, ähnlich wie dies bei der in Abschnitt 3.2.2.2 vorgestellten *Dynamic Design Method* [Ishi97] der Fall ist, auch hier speziell auf das verwendete Agentensystem zugeschnitten.

Entertainment-Agenten: Da Agenten, wie in Abschnitt 4.3.1 beschrieben, häufig durch „menschliches" Verhalten charakterisiert werden, bietet sich der Einsatz so genannter Entertainment-Agenten in Computerspielen und interaktiven Filmen für Agenten an. Ebenso wie bei Interface-Agenten spielt die Kommunikation zwischen den Agenten und dem Anwender eine besondere Rolle, so dass auch hier die Theatermetapher als grundlegendes Modell anzusehen ist. Allerdings wird bei der Modellierung von Entertainment-Agenten das Verhalten in einer virtuellen Umgebung betont, in der die Agenten durch virtuelle Sensoren auf Änderungen ihrer Umgebung reagieren.

Zum einen wird Entertainment-Agenten durch die Verhaltensbeschreibung ein emotionales und/oder motorisches Verhalten zugeordnet, das durch die graphische Darstellung dem Benutzer vermittelt wird. Zum anderen agiert der Benutzer mit den Entertainment-Agenten direkt bzw. indirekt über eine Schnittstelle, wobei die direkte Kommunikation [HRBG97] durch Sprechakte und die indirekte Kommunikation durch räumliche, zeitliche und emotionale Beziehungen zwischen den Agenten und dem Benutzer, der in der virtuellen Welt als so genannter Avatar repräsentiert wird [HRGe96] [WaGr96], stattfindet.

Die direkte Programmierung von Entertainment-Agenten wird für die Nachbildung realer Schauspieler angewendet [BeTh96] [ThNH97]. Dabei können durch die Beschreibung der Agenten virtuelle visuelle, taktile und akustische Ereignisse auf motorische Fähigkeiten, wie etwa das Greifen eines Gegenstands, abgebildet werden. Allerdings sind bei diesem Ansatz die Agenten eher als Marionetten anzusehen, bei denen die aufwendige Definition einzelner Bewegungsabläufe durch die Programmierung der virtuellen Schauspieler auf unterschiedlichen Abstraktionsniveaus [WaCo97] ersetzt werden kann.

Neben der direkten Programmierung sind, wie bereits in Abschnitt 3.2.2.2 erwähnt, unterschiedliche Ansätze aus der Literatur bekannt, in denen Entertainment-Agenten als virtuelle Puppen anzusehen sind, deren Verhalten von dem Benutzer beeinflusst werden kann. Die Interaktion zwischen Benutzer und Agenten findet dabei durch die direkte Auswahl möglicher Aktionen statt [HRBG97] bzw. durch indirekte Beeinflussung der Agenten, wie etwa durch die Manipulation der Umgebung [SmCS97] oder durch das Trainieren des Verhaltens eines Agenten [GrCl98].

Kapitel 5

Subjektorientierter Entwurf

Die Betonung einzelner Bereiche während der Produktion multimedialer Anwendungen führt, wie in Kapitel 3 und Kapitel 4 gezeigt, zu unterschiedlichen Modellen, die lediglich auf einem speziellen Blickwinkel der an dem Medium partizipierenden Personen beruhen. Doch gerade während der Produktion multimedialer Anwendungen ist es von entscheidender Bedeutung, die unterschiedlichen Blickwinkel miteinander gleichberechtigt zu verbinden, so dass neben den softwaretechnischen Fragestellungen ebenso die gestalterischen Aspekte und die Sicht der Rezipienten berücksichtigt werden können.

Die unterschiedlichen Ansätze werden in dieser Arbeit durch so genannte *multimediale Mittlerelemente* miteinander verbunden. Dieser Begriff ist dem Begriff der *Mittlerobjekte*, die als Metapher die Verbindung zwischen der künstlerischen und wissenschaftlichen Modellierung beschreiben und somit die anwendungsorientierte Realisierung betonen [Bec98], entlehnt. Im Gegensatz hierzu stellen multimediale Mittlerelemente als integrierendes Modell ein durchgängiges Konzept für den Entwurfsprozess, die Modellierung eines Präsentationsmediums und die Umsetzung von verteilten Wahrnehmungsumgebungen bereit. Dabei unterstützen multimediale Mittlerelemente eine lose Verbindung zwischen den zu betrachtenden Rollen, so dass der Entwurfsprozess unterschiedliche zeitliche Abläufe zwischen den Spezialisten ermöglicht. Im Gegensatz zu den in Kapitel 3 vorgestellten Ablauforganisationen, die ausschließlich eine feste Abfolge vorgeben, können somit dem Anwendungskontext angemessene Vorgehensweisen unterstützt werden.

In diesem pragmatischen Ansatz [Brow00], der zwischen einer festen Ablauforganisation und einem eher experimentellen Ansatz steht, gibt die Ablauforganisation lediglich einen Rahmen während des Entwurfs vor. Der dabei durch den subjektorientierten Entwurf vorgegebene Rahmen unterstützt sowohl die Realisierung „traditioneller“ multimedialer Anwendungen als auch die Gestaltung verteilter Wahrnehmungsumgebungen, indem typische Arbeitsweisen gleichberechtigt unterstützt werden, die je nach Anwendungskontext von unterschiedlichen Spezialisten ausgefüllt werden können. Wie in Kapitel 7 anhand von Beispielen noch näher betrachtet, können durch den subjektorientierten Entwurf somit anwendungsspezifische Zeitbereiche vorgegeben werden, die sich erst während des Entwurfsprozesses näher herausbilden.

5.1 Ablauforganisation

Die in Kapitel 3 vorgestellten Ablauforganisationen betonen die aus dem jeweiligen Fachgebiet stammenden Modellierungsansätze, die aber zu kurz greifen, da sie ursprünglich für andere Problembeschreibungen entwickelt wurden. Dagegen basiert der hier vorgestellte Ansatz auf Erfahrungen, die in verschiedenen Multimediaprojekten gesammelt wurden und betont somit rollenspezifische Arbeitsweisen während einer Produktion.

Um eine gemeinsame Basis zur Verfügung stellen zu können, stehen die multimedialen Mittlerelemente im Zentrum der Produktion. Multimediale Mittlerelemente unterstützen dabei ein Organisationsprinzip, indem sie Protokolle für typische Rollen während der Produktion multimedialer Anwendungen bereitstellen. Sie treten als vermittelnde Elemente einzelner Blickwinkel von Spezialisten einer Entwicklergruppe auf und dienen somit als Kommunikationsgrundlage und Denkmodell.

5.1.1 Integrationsmodell

Die an der Entwicklung von multimedialen Anwendungen beteiligten Spezialisten und die Nutzer dieser Anwendungen leben in zum Teil völlig „fremden“ Welten. Dies wird am stärksten dort deutlich, wo Programmierer nicht nur für den Programmcode, der die eigentliche Funktionalität der Anwendung beschreibt, verantwortlich sind, sondern auch die Schnittstelle gestalten. Diese ist dann häufig geprägt durch die dem Programm inhärente Logik und Sprache, nicht jedoch von der semantischen Bedeutung des Inhalts durch die eine erfolgreiche Kommunikation erst möglich wird.

Aber auch innerhalb der aus Spezialisten zusammengesetzten Entwicklergruppe ist eine gemeinsame semantische Diskurswelt der entscheidende Faktor einer erfolgreichen Kommunikation. Bereits die Erstellung des Konzepts einer multimedialen Anwendung spiegelt, wie in Abschnitt 3.1 gezeigt, diese unterschiedlichen Diskurswelten wider, die sich in der Gestaltung bzw. Modellierung eines Präsentationsmediums wiederfinden. Das Hauptproblem bei Multimediaprojekten ist dabei die Lücke zwischen technischen und gestalterischen Rollen während des Entwurfs. Der Unterschied zwischen strukturorientierten und mentalen Modellen, die zum einen eher die technische Sicht, zum anderen eher die gestalterische Sicht betonen, spiegelt sich auch in der Denk- und Arbeitsweise der Spezialisten wider.

Die gestalterische Definition findet häufig durch Skizzen oder verbale Beschreibungen des Modells statt, durch die ein zusammenhängendes Bild der Anwendung vorgegeben wird. Durch den verstärkten Einsatz von Computern in den gestalterischen Arbeitsfeldern setzt sich aber immer mehr der digitale Entwurf durch, der mithilfe entsprechender Layout-Programme bzw. Medienbearbeitungsprogramme stattfindet.

Aus technischer Sicht, wie bei dem objektorientierten oder datenbankorientierten Entwurfsparadigmen, werden unterschiedliche Ansätze der Analyse und Entwurfsmodelle zwar durch Entwurfswerkzeuge unterstützt, doch sind diese Ansätze durch die analytische Dekomposition bzw. Klassifikation charakterisiert. Im Gegensatz zu gestalterischen Beschreibungen ist dabei eine strenge Formalisierung der Beschreibungsformen vorherrschend, um eine automatische Transformation zu unterstützen, so dass derartige Formen nur schwer von Spezialisten aus anderen Fachdisziplinen nachzuvollziehen sind.

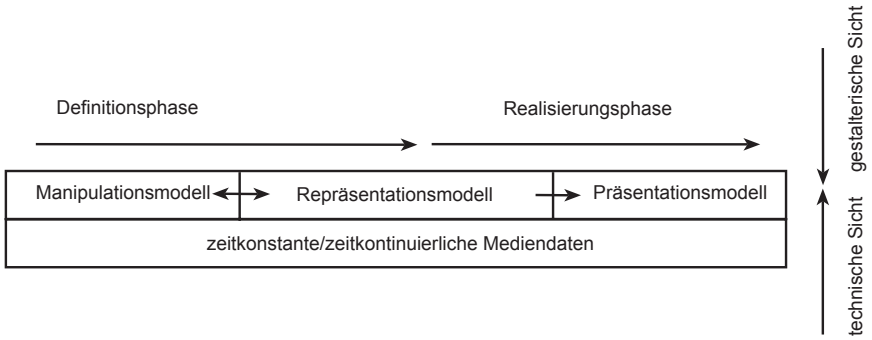


Abb. 5.1.1.1: Dokumentenarchitektur des subjektorientierten Entwurfs.

Die hier vorgestellte Ablaufumgebung integriert beide Sichten, indem für den gesamten Entwurfsprozess ein einheitliches Modell angeboten wird. Das Modell unterstützt dabei sowohl die Integration arbeitsspezifischer Beschreibungsformen, so dass die unterschiedlichen Spezialisten ihren eigenen Blickwinkel auf die Produktion haben können, als auch eine einheitliche Beschreibungsform, die die automatische Transformation auf unterschiedliche Präsentationsmedien unterstützt. Um beide Sichten miteinander zu verbinden, basiert die Ablauforganisation auf einer Dokumentenarchitektur, so dass der Übergang zwischen Definitionsphase und der eigentlichen Realisierung, wie in Abb. 5.1.1.1 dargestellt, durch ein entsprechend gewähltes Repräsentationsmodell beschrieben wird.

Die Definitionsphase unterstützt die in Abschnitt 3.1 näher beschriebenen Aufgabenprofile der Medienerfassung, der Medienaufbereitung, der Layoutbeschreibung und der Verhaltensdefinition. Im Gegensatz zu den dort vorgestellten Entwurfsverfahren kann dabei das Konzept einer multimedialen Anwendung aus unterschiedlichen Blickwinkeln entwickelt werden. Während der Realisierungsphase findet die eigentliche Abbildung auf ein Präsentationsmedium statt, so dass hier die Aufgabenprofile der Implementierung, der Installation und des Tests zu betrachten sind. Somit spielen hier die technischen Rollen während des Entwurfsprozesses eine dominierende Rolle.

Die einzelnen Arbeitsprofile werden durch multimediale Mittlerelemente miteinander verbunden, die zum einen als Metapher des mentalen Modells dienen, zum anderen formal durch die in dieser Arbeit entwickelte *Multimedia Definition Language (MDL)* repräsentiert werden. Multimediale Mittlerelemente können somit im Sinne der Semiotik als Metazeichensystem des subjektorientierten Entwurfs angesehen werden.

Damit wird durch die multimedialen Mittlerelemente das formale Gerüst für die vermittelnde Gesamtsicht des Projektleiters zur Verfügung gestellt. Wie bereits in Abschnitt 3.1 gezeigt, besteht dabei die wesentliche Aufgabe in der Koordination zwischen den einzelnen Sichten der Spezialisten. Um dies in einer geeigneten Weise unterstützen zu können, sollte der Projektleiter die unterschiedlichen Diskurswelten verstehen. Für die Bereitstellung einer formalen Grundlage bedeutet dies aber für die multimedialen Mittlerelemente, dass sie die unterschiedlichen Sichten der Spezialisten in gleichberechtigter Weise unterstützen müssen. Erst hierdurch wird erreicht, dass die Spezialisten ihre rollenspezifischen Arbeitsweisen beibehalten können und die multimedialen Mittlerelemente die Kommunikationsgrundlage zwischen den Spezialisten bilden.

Nachfolgend werden zunächst multimediale Mittlerelemente innerhalb der drei Modelle der Dokumentenarchitektur betrachtet. Im Anschluss findet eine nähere Beschreibung der Definitionsphase in Abschnitt 5.2 und der Realisierungsphase in Abschnitt 5.3 statt.

5.1.2 Manipulationsmodell

Im Gegensatz zu den in Kapitel 3 vorgestellten Ablauforganisationen, bei denen feste Phasen bereits ein enges Korsett für die Organisation vorgeben, liegt die Betonung bei dem subjektorientierten Entwurf auf der Dynamik zwischen den einzelnen Rollen und den rollenspezifischen Arbeitsweisen. Hierdurch wird erreicht, dass durch die Ablauforganisation die Arbeitsweisen der an dem Entwurf beteiligten Spezialisten selbst im Vordergrund stehen und nicht durch eine (auf das Präsentationsmedium) eingeschränkte Struktur vorgegeben werden.

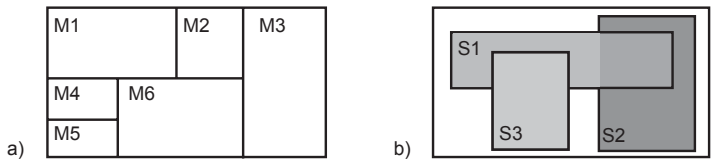


Abb. 5.1.2.1: Modularisierung multimedialer Anwendungen in unabhängige Teilbereiche (a) bzw. Sichten (b).

Um die Arbeitsweisen gleichberechtigt unterstützen zu können, basiert die hier vorgestellte Ablauforganisation auf unterschiedlichen, voneinander unabhängigen Sichten, die den einzelnen individuellen Blickwinkeln der an dem Entwurf beteiligten Spezialisten entsprechen. Die jeweiligen Sichten sind im Gegensatz zu den bestehenden Verfahren dabei nicht durch eine vorherrschende Struktur auf einzelne Bereiche einer Anwendungsbeschreibung beschränkt, sondern definieren vielmehr verschiedene Problembeschreibungen mit überlappenden Elementen.

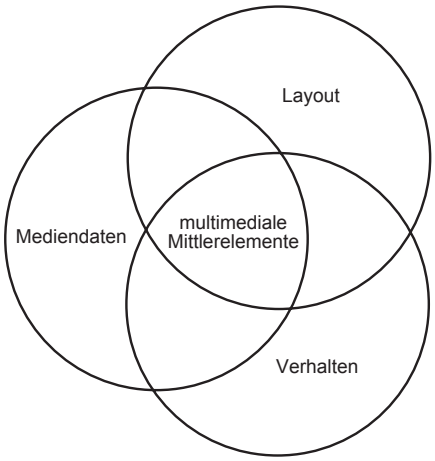


Abb. 5.1.2.2: Unterstützte Sichten auf multimediale Mittlerelemente während der Definitionsphase.

Somit wird durch eine Sicht ein einzelner rollenspezifischer Aspekt einer gesamten multimedialen Anwendung beschrieben, im Gegensatz zu der Aufteilung in Module, die ausschließlich Teilkomponenten einer dominanten Struktur darstellen. Diese beiden in Abb. 5.1.2.1 dargestellten Ansätze müssen sich allerdings nicht gegenseitig ausschließen, so dass sich die Sichten sowohl für eine gesamte Anwendung als auch auf einzelne Module definieren lassen. Die Sichten unterstützen somit unterschiedliche Granularitäten und können für den Entwurf einzelner Akteure, komplexer Objekte, von Seiten bzw. Szenen und verteilten Wahrnehmungsumgebungen eingesetzt werden.

Unabhängig von der Granularität wird der subjektorientierte Entwurf durch drei Sichten unterstützt, die die konstruktiven Methoden der semiotischen Kategorien (Auswahl-, Aufbau-, Interaktionskategorien) einer multimedialen Anwendung widerspiegeln, so dass multimediale Anwendungen aus Sicht der Mediendaten, des Layouts und des Verhaltens beschrieben werden. Die Sichten der Mediendaten und des Layouts bzw. des 3D-Entwurfs spiegeln dabei typischerweise die gestalterischen Rollen von Medienspezialisten und Graphikdesignern bzw. 3D-Modellierern wider, wohingegen die Beschreibung des Verhaltens der Rolle der Informatiker bzw. Programmierer entspricht. Um die Sichten der Spezialisten miteinander in Beziehung zu setzen, stehen die multimedialen Mittlerelemente, wie in Abb. 5.1.2.2 dargestellt, als Vermittler zwischen den Sichten. Somit dienen sie als Metazeichen des Entwurfs, die die einzelnen Elemente einer multimedialen Anwendung repräsentieren. Sie können als leere Elemente mit einer eindeutigen Bezeichnung angesehen werden, denen während des Entwurfsprozesses, wie in Abb. 5.1.2.3 dargestellt, die (subjektive) Sicht der Spezialisten zugewiesen wird.

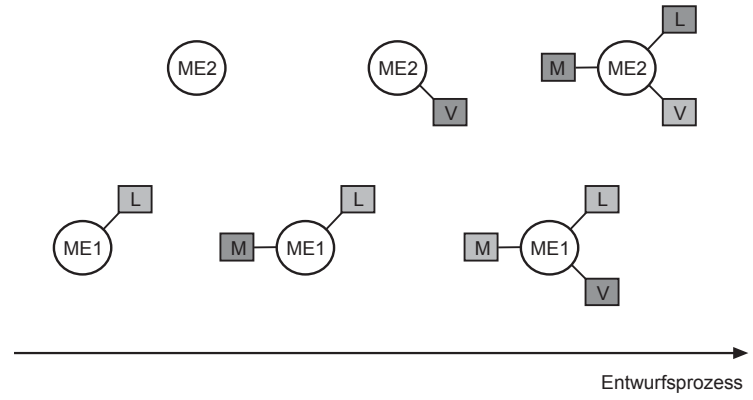


Abb. 5.1.2.3: Zuweisung der Layoutsicht *L*, Mediensicht *M* und Verhaltenssicht *V* unterschiedlicher multimedialer Mittlerelemente *ME* während des Entwurfsprozesses.

Im Gegensatz zu einem zentralen Kommunikationsmodell unterstützen die einzelnen Sichten unterschiedliche Abstraktionsniveaus, so dass auch während der Erstellung eines Konzepts den Spezialisten aus ihrer Arbeitswelt vertraute Beschreibungsformen, wie in Abschnitt 5.2 noch näher gezeigt wird, zur Verfügung stehen. Aus Sicht des Drehbuchs können multimediale Mittlerelemente als Akteure interpretiert werden, deren Erscheinungsbild (Mediensicht), Form (Layoutsicht) sowie Verhalten separat definiert werden. Um diesen filmorientierten Ansatz, bei dem die multimedialen Mittlerelemente als voneinander unabhängige Akteure

fungieren, mit dem strukturorientierten Ansatz zu verbinden, lassen sich innerhalb einer Sicht Hierarchien bilden. Somit unterstützt die subjektorientierte Ablauforganisation den Entwurf einzelner Akteure, das Zusammenfassen mehrerer Mittlerelemente zu komplexeren Darstellern und den Aufbau ganzer Seiten bzw. Szenen innerhalb einer Hierarchie.

Da die Hierarchien der Sichten unabhängig voneinander entwickelt werden, besitzen diese im Gegensatz zu den strukturorientierten Entwurfsverfahren eine sichtspezifische Semantik. Wie in Abschnitt 5.2 noch näher betrachtet, können somit multimedialen Mittlerelementen aus Mediensicht unterschiedliche kontextabhängige Mediendaten zugewiesen werden, aus Layoutsicht Darstellungshierarchien gebildet werden und aus der Verhaltenssicht multimediale Mittlerelemente in Kommunikationshierarchien eingebunden werden. Somit kann ein multimediales Mittlerelement in unabhängig voneinander definierten Hierarchien eingebunden sein.

5.1.3 Repräsentationsmodell

Um auf der einen Seite die unterschiedlichen Sichten rollenspezifisch unterstützen zu können, und auf der anderen Seite eine einheitliche formale Beschreibung zu ermöglichen, die durch Transformationen in ein Präsentationsmodell überführt werden kann, wird innerhalb der in dieser Arbeit eingeführten Repräsentationssprache zwischen den multimedialen Mittlerelementen, der Repräsentation elementarer und komplexer Sichten und den so genannten Aspekten einer Sicht unterschieden.

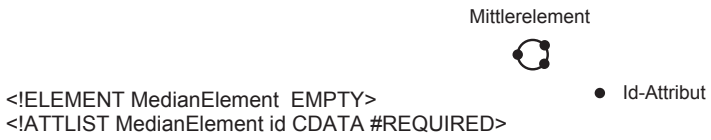
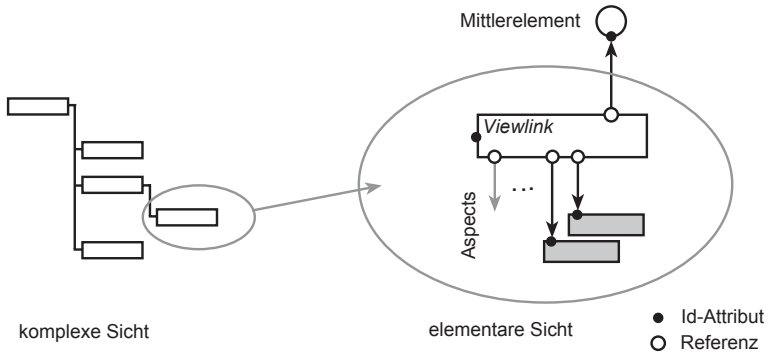


Abb. 5.1.3.1: Dokument Typdefinition multimedialer Mittlerelemente.

Wie bereits im vorherigen Abschnitt beschrieben, sind die multimedialen Mittlerelemente die Vermittler zwischen den einzelnen Sichten. Die Mittlerelemente werden dabei als leere Elemente angesehen, die durch die Zuweisung der drei Sichten der Mediendaten, des Layouts und des Verhaltens als Metazeichen eines Zeichens einer multimedialen Anwendung auftreten. Die Zuweisung zwischen einem Mittlerelement und einer Sicht erfolgt innerhalb der *Multimedia Definition Language (MDL)* über das Identifikationsattribut (id) eines Mittlerelements (MedianElement), das dieses, wie in Abb. 5.1.3.1 dargestellt, in eindeutiger Weise auszeichnet.

Die Sichten eines Mittlerelements werden, wie in Abb. 5.1.3.2 dargestellt, durch Verbindungselemente repräsentiert, so dass sie in einheitlicher Weise behandelt werden können. Ein Verbindungselement verwaltet sowohl den Verweis auf das Mittlerelement als auch die Definition der zuzuweisenden Aspekte einer elementaren Sicht und kann somit als Stellvertreter eines Mittlerelements innerhalb einer Sicht angesehen werden.




```
<!ELEMENT Viewlink (MedianElementId, Children?, AspectID_1, ..., AspectID_n)>
<!ELEMENT MedianElementId EMPTY>
<!ELEMENT Children (ViewLink | ViewLinkId)*>
<!ELEMENT ViewLinkId EMPTY>

<!ATTLIST Viewlink id CDATA #REQUIRED>
<!ATTLIST MedianElementId name CDATA #REQUIRED>
<!ATTLIST ViewLinkId name CDATA #REQUIRED>
<!ATTLIST AspectID_i name CDATA>
```

Abb. 5.1.3.2: DTD-Struktur von Sichten.

Komplexe Sichten werden durch die Schachtelung einzelner Verbindungselemente repräsentiert, so dass durch eine komplexe Sicht eine Kapselung sichtspezifischer Aspekte auf mehrere Mittlerelemente definiert wird. Komplexe Sichten werden somit, ähnlich wie in XML-Dokumenten üblich, durch eine Hierarchie repräsentiert. Im Gegensatz zu XML-Dokumenten, die eine logische Struktur des Inhalts repräsentieren, werden durch die Hierarchie einer komplexen Sicht allerdings ausschließlich rollenspezifische Beziehungen zwischen elementaren Sichten beschrieben. Komplexe Sichten können somit, wie in Abschnitt 5.1.2 gefordert, überlappen und unterstützen die unterschiedliche sichtspezifische Strukturierung einer multimedialen Anwendung.

```
<!ELEMENT Aspect ( Parameter_1, …, Parameter_n) >

<!ATTLIST Aspect id CDATA #REQUIRED>
<!ATTLIST Parameter_i
        value_1 CDDATA,
        …
        value_m_i CDDATA>
```

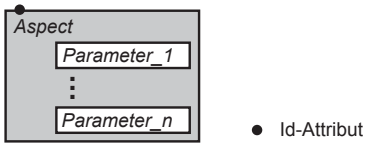


Abb. 5.1.3.3: DTD-Struktur von Aspekten einer Sicht.

Die Aspekte einer Sicht sind die kleinsten Einheiten von MDL. Zum einen werden durch die Aspekte die Parameter einer Sicht repräsentiert, so dass sie entsprechend wie in XML während der Abbildung auf ein Präsentationsmodell als Regeln interpretiert werden können. Im Gegensatz zu den in XML verwendeten Regeln, die auf der logischen Struktur definiert werden, werden die Aspekte hier jedoch als Regeln auf elementaren und komplexen Sichten interpretiert. Zum anderen spiegeln Aspekte die Arbeitsweisen der Spezialisten wider, so dass sie im Gegensatz zu anderen Repräsentationssprachen als wieder verwendbare Einheiten

in den Arbeitsprozess eingebunden werden können. Um trotz der sicht- bzw. rollenspezifischen Aspekte diese in einer einheitlichen Art und Weise handhaben zu können, werden die Aspekte, wie in Abb. 5.1.3.3 dargestellt, durch einen eindeutigen Namen und den dazugehörigen Parametern in MDL repräsentiert.

5.1.4 Präsentationsmodell

Durch die präsentationsneutrale Beschreibung innerhalb des Repräsentationsmodells können unterschiedliche Präsentationsmodelle unterstützt werden. Wie in Abschnitt 5.3 näher gezeigt, wird die Abbildung durch einen für das jeweilige Präsentationsmodell erstellten Regelsatz definiert, so dass neben der Gestaltung realer Wahrnehmungsumgebungen auch die Abbildung auf „traditionelle“ geschlossene Präsentationsmedien unterstützt wird.

Vorherrschendes Modell „traditioneller“ Präsentationsmedien ist die Strukturierung durch die Layout-Beschreibung, durch die, vergleichbar mit den in Abschnitt 4.2.3 beschriebenen Dokumentenmodellen, elementare und komplexe Medienverwalter zusammengesetzt werden. Somit müssen, wie in Abschnitt 5.3 näher betrachtet, die einzelnen Aspekte und Sichten durch Regeln zusammengefügt und in die jeweilige Präsentationssprache übersetzt werden.

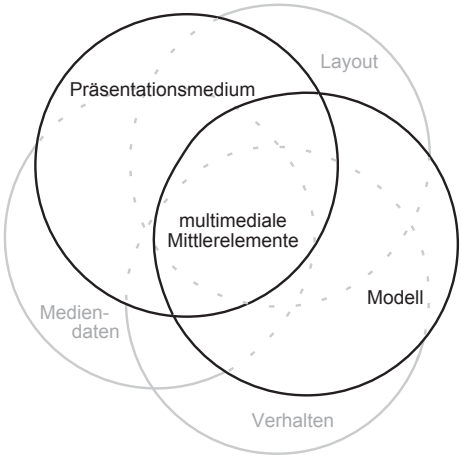


Abb. 5.1.4.1: Aspekte multimedialer Mittlerelemente während der Abbildung in eine Wahrnehmungsumgebung.

Um die Repräsentationsbeschreibung auf eine verteilte Anwendung für die Gestaltung einer Wahrnehmungsumgebung abzubilden, können während der Transformation die multimedialen Mittlerelemente sowohl auf unterschiedliche Rechnerumgebungen verteilt als auch externen Ein- und Ausgabegeräten zugewiesen werden. Somit dienen die Mittlerelemente als Vermittler zwischen der multimedialen Umgebung und dem realen Wahrnehmungsraum. Zur Unterstützung dieser Funktionalität wurde eine Architektur für Präsentationsmedien multimedialer Mittlerelemente entwickelt, die in Kapitel 6 behandelt wird.

Durch die Architektur modellieren multimediale Mittlerelemente die Organisationsstruktur einer multimedialen Anwendung, d.h. sie verwalten die Sichten und fungieren somit als organisierende Elemente in einer verteilten Ablaufumgebung, indem sie die Kommunikation, die Wiedergabe der Mediendaten auf unterschiedlichen Ausgabegeräten und die Integration

natürlicher Interaktionsformen mithilfe von Eingabegeräten übernehmen. Multimediale Mittlerelemente verbinden somit, wie in Abb. 5.1.4.1 dargestellt, die Mediendaten und die Layoutbeschreibung mit den durch die lokalen Ablaufumgebungen zur Verfügung gestellten realen bzw. virtuellen Präsentationsmedien. Um dabei die Steuerung durch externe Daten bzw. innerhalb der verteilten Umgebung zu unterstützen, können multimediale Mittlerelemente innerhalb spezieller Modellbeschreibungen eingebunden werden.

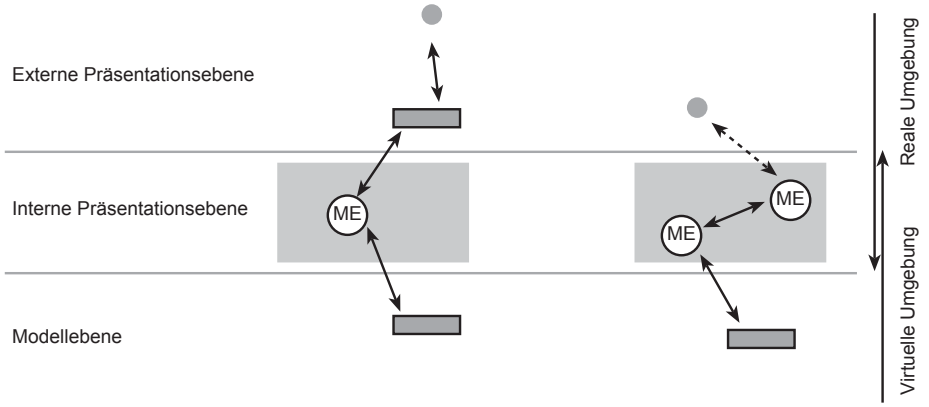


Abb. 5.1.4.2: Logisches Modell multimedialer Mittlerelemente innerhalb einer Wahrnehmungsumgebung.

Wie in Abschnitt 2.3 gezeigt, zeichnet sich die semiotische Bedeutung von Wahrnehmungsumgebungen sowohl durch die interne Gestaltung der virtuellen Darstellung als auch durch die Gestaltung bzw. Abbildung auf unterschiedliche Präsentationsmedien aus. Um die Abbildung der bereits durch das Repräsentationsmodell beschriebenen Sichten auf eine verteilte Anwendung zu unterstützen, wird eine Wahrnehmungsumgebung als abstraktes Modell, wie in Abb. 5.1.4.2 dargestellt, durch drei logische Ebenen beschrieben:

Modellebene: Die Modellebene spiegelt die globale virtuelle Sicht auf die Wahrnehmungsumgebung wider, in der die Mittlerelemente innerhalb eines gemeinsamen n-dimensionalen Datenraums repräsentiert werden. Je nach Anwendungskontext kann der Datenraum dabei z.B. als virtueller zwei- oder dreidimensionaler Raum, als Simulationsumgebung mit eigenen Regeln bzw. als abstraktes Datenmodell modelliert werden. Die Mittlerelemente interpretieren die Daten und führen entsprechende Aktionen aus, die die Interpretation der Daten innerhalb der Präsentationsebenen steuern.

Interne Präsentationsebene: Die internen Präsentationsmedien werden durch lokale Ablaufumgebungen, die mit „traditionellen" geschlossenen Präsentationsmedien vergleichbar sind, modelliert. Wie in Abschnitt 2.3 gezeigt, wird die Bedeutung einer geschlossenen Präsentation sowohl durch die Mediendaten selbst als auch durch die Aufbaukategorien in Form des Layouts und durch die Interaktionskategorien hervorgerufen. Die Aufteilung der internen Präsentation auf mehrere Ablaufumgebungen ermöglicht die Erweiterung der Aufbaukategorien, indem die Raumgestaltung der realen Umgebung mit berücksichtigt wird. Die hierdurch unterstützten Aufbaukategorien sind eher als horizontale Verteilung zu interpretieren, in der die Wiedergabe der Mediendaten weiterhin nur durch eine interne (virtuelle) Präsentation der Daten stattfindet. Allerdings stellen die lokalen Ablaufumgebungen nicht einzelne, unabhängige Präsentationsmedien

dar, sondern bilden durch das Zusammenwirken bereits einen fließenden Übergang zwischen virtueller und realer Umgebung.

Externe Präsentationsebene: Die externe Präsentationsebene wird durch Ein- und Ausgabemedien gebildet, die die Welt der „traditionellen" analogen Medien widerspiegelt. Externe Ausgabemedien ermöglichen somit die Präsentation durch unterschiedliche „traditionelle" Präsentationsmedien, wie etwa Diaprojektoren, Fernseher, Strahler etc., so dass durch diese der verwendete semantische Raum in den Erfahrungsraum des Rezipienten transformiert werden kann. Durch die Integration externer Eingabemedien, wie z.B. haptischer oder optischer Sensoren, stehen dem Rezipienten natürliche, direkte bzw. indirekte Interaktionsformen zur Verfügung, so dass auch hier aus dem täglichen Leben bekannte bzw. spielerische Beeinflussungsmöglichkeiten angeboten werden können.

Im Gegensatz zu der internen Präsentationsebene wird somit durch die externe Präsentationsebene eine eher als vertikale Verteilung zu bezeichnende Gestaltung der realen Umgebung unterstützt, die neben den Aufbaukategorien auch durch die Auswahl der externen Ein- und Ausgabemedien selbst beeinflusst wird.

Eine durch die drei Ebenen gestaltete Wahrnehmungsumgebung ist mit der aus der Medientheorie bekannten Spiegelmetapher [Kräm88] [Huht98] vergleichbar, bei der die virtuelle Umgebung als Spiegel aufgefasst wird, der eine imaginäre Weiterführung der realen Umgebung darstellt. Die „Spiegelwelt" wird durch eine symbolische Welt (Mediendaten und Layout) mit eigenen Regeln (Verhalten) und der Möglichkeit der Beobachtung (Präsentationsmedien) und Teilnahme (Interaktion) an dieser Welt beschrieben.

Wie in Abb. 5.1.4.2 dargestellt, bilden die multimedialen Mittlerelemente die Spiegeloberfläche, so dass sie als Vermittler zwischen der virtuellen und realen Umgebung auftreten und somit einen fließenden Übergang zwischen beiden Umgebungen unterstützen. Der Rezipient kann sich in natürlicher Weise in der multimedialen Umgebung bewegen und mit dieser interagieren. Wie bereits weiter oben näher beschrieben, wird dies sowohl durch die räumlich verteilten Ablaufumgebungen der internen Präsentationsebene erreicht, die jeweils einen Teilbereich der virtuellen Umgebung widerspiegeln, als auch durch die externe Präsentationsebene, in der „traditionelle" Präsentations- und natürliche Interaktionsformen unterstützt werden. So kann die virtuelle Umgebung z.B. direkt über reale Gegenstände mithilfe haptischer Interaktionsformen beeinflusst werden bzw. Umgebungseinflüsse wie etwa Temperatur oder Helligkeit der realen Umgebung widerspiegeln.

5.2 Definitionsphase

Um während der Definitionsphase die einzelnen Sichten gleichberechtigt unterstützen zu können, bietet der subjektorientierte Entwurf die Integration unterschiedlicher Entwurfswerkzeuge an, die die typischen Arbeitsschritte der Spezialisten ermöglichen. Hierbei wird der Entwurf während der Definitionsphase verfeinert, so dass die einzelnen Sichten in einzelne Aspekte aufgegliedert werden können. Hierdurch wird des Weiteren erreicht, dass die Sichten weitgehend unabhängig voneinander entwickelt werden können, da die Definition spezifischer Parameter, die sichtübergreifend definiert werden müssen, auf späte Entwurfsphasen verschoben werden können.

Zur Verdeutlichung der Dynamik des Entwurfsprozesses wird hier ein typisches Vorgehen innerhalb der Sichten vorgestellt. Wie bereits in Abschnitt 5.1.3 gesagt, werden die einzelnen Aspekte durch das Repräsentationsmodell dargestellt, so dass anhand der in dieser Arbeit entwickelten *Multimedia Definition Language (MDL)* die Definitionsmöglichkeiten innerhalb der Arbeitsschritte charakterisiert werden.

5.2.1 Mediensicht

5.2.1.1 Manipulationsmodell

Durch die Mediensicht wird der Umgang mit den zu präsentierenden Mediendaten unterstützt, so dass diese Sicht den Medienspezialisten und Experten des zu präsentierenden Inhalts entspricht. Um dabei einen rollenspezifischen Umgang mit den Mediendaten zu ermöglichen, können die Mediendaten in einer Mediendatenbank verwaltet werden, auf die Standardprogramme für die Bearbeitung der elementaren Daten aufsetzen.

Für die Erstellung der Mediendaten einer multimedialen Anwendung können die Rohdaten für die Bearbeitung bereits in die Mediendatenbank eingefügt werden. Hierbei können während der Recherche erfasste Medien digitalisiert bzw. Teilkomponenten für zusammengesetzte Medien, wie etwa digitale Filme, erzeugt werden. Wie in Abb. 5.2.1.1.1 dargestellt, basiert die eigentliche Erstellung auf diesen Daten und ist von den jeweiligen Medientypen abhängig. Durch die Integration unterschiedlicher Programme werden dabei gängige Arbeitsformen der Mediengestalter bzw. Medientechniker unterstützt, so dass die Erstellung und Bearbeitung der Mediendaten z.B. durch Bildbearbeitungsprogramme wie *Adobe Photoshop* [DaDa96], die Erzeugung von Filmen durch Schnittprogramme wie *Adobe Premiere* [EiWe99] bzw. *Adobe AfterEffects* [MeMe00] etc. stattfindet.

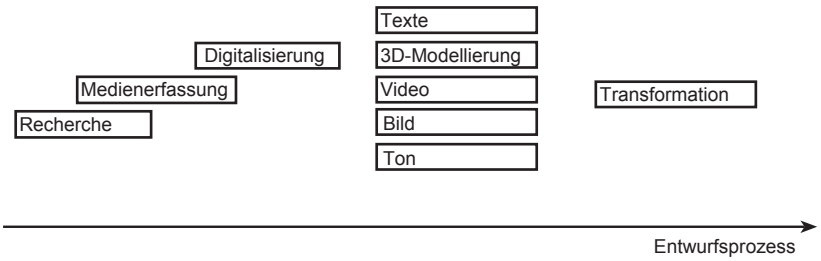


Abb. 5.2.1.1.1: Ablauf des Entwurfsprozesses aus Sicht der Medienspezialisten.

Erst in einem letzten Arbeitsschritt werden die Mediendaten in die von dem Präsentationsmedium unterstützten Datenformate transformiert, so dass sie von dem Präsentationsmedium wiedergegeben werden können. Hierbei spielt neben den eigentlichen Datenformaten insbesondere die Kompression der Mediendaten eine zentrale Rolle, die an die Übertragungszeiten des Informationsaustauschmediums angepasst werden muss.

5.2.1.2 Repräsentationsmodell

Damit die Mediendaten in einheitlicher Weise in der Mediendatenbank verwaltet werden können, werden die Daten unabhängig von ihrem Typ mithilfe eines so genannten Medienelements, das durch einen eindeutigen Namen repräsentiert wird, gekapselt. Die Verbindung

findet, wie in Abb. 5.2.1.2.1 dargestellt, über die in Abschnitt 5.1.3 eingeführte Sichtdefinition statt, die hier einer 1:1-Beziehung zwischen multimedialem Mittlerelement und Medienelement entspricht. Dabei wird die Zuordnung durch erzeugte Schlüsselwörter der Datenbank unterstützt.

```
<!ELEMENT MediaView  (MediaElementId)>
<!ELEMENT MediaElementId EMPTY>

<!ATTLIST MediaView
        id CDATA #REQUIRED>
<!ATTLIST MedianElementId
        name CDATA #REQUIRED >
<!ATTLIST MediaElementId
        name CDATA>
```

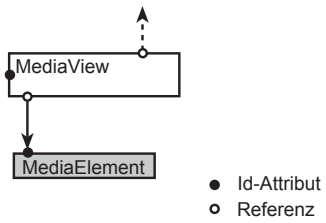


Abb. 5.2.1.2.1: DTD der Mediensicht.

Durch die Kapselung der Mediendaten lassen sich den Mittlerelementen sowohl elementare als auch komplexe Medienelemente zuordnen. Elementare Medienelemente werden, wie in Abb. 5.2.1.2.2 dargestellt, durch ihren logischen Namen, ihren Typ und die Speicherposition des Mediums in MDL repräsentiert.

```
<!ELEMENT MediaElement ((Type, Ref) | Switch)>
<!ELEMENT Switch (Case*)>
<!ELEMENT Case (MediaElement)>

<!ELEMENT Type EMPTY>
<!ELEMENT Ref EMPTY>

<!ATTLIST MediaElement
        id CDATA  #REQUIRED>
<!ATTLIST Case
        value CDATA  #REQUIRED>
<!ATTLIST Type
        name (”text”  | ”image” | ”movie” | ”audio”  |  ”animation”)  #REQUIRED >
<!ATTLIST Ref
        href CDATA  #REQUIRED>
```

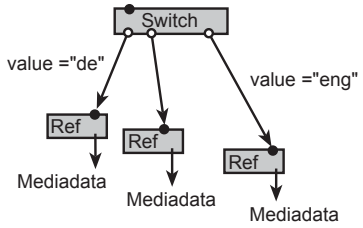


Abb. 5.2.1.2.2: DTD elementarer und komplexer Medienelemente.

Komplexe Medienelemente setzen sich, ähnlich wie die in Abschnitt 2.2 eingeführten komplexen Medienverwalter, aus elementaren oder komplexen Medienelementen zusammen. Im Gegensatz zu den Medienverwaltern werden durch komplexe Medienelemente allerdings jedoch nur alternative Daten zusammengefasst, so dass einem multimedialen Mittlerelement eine dem Präsentationskontext bzw. dem Zustand des Mittlerelements angepasste Darstellung zugeordnet werden kann. So können z.B. unterschiedliche Sprachen für eine Präsentation zur Verfügung gestellt werden oder unterschiedliche Zustände eines Buttons durch ein komplexes Medienelement repräsentiert werden. Die Kontextabhängigkeit eines komplexen Medienelements wird in MDL, wie in Abb. 5.2.1.2.2 dargestellt, mithilfe einer Kontextbeschreibung (switch name) und der jeweiligen Bedingung (case value) repräsentiert.

5.2.2 Layoutsicht

5.2.2.1 Manipulationsmodell

Durch die Layoutbeschreibung wird die Form einer multimedialen Anwendung definiert. Hierbei werden unterschiedliche Aspekte wie die geometrische Form, die Positionierung als auch die eigentliche Wiedergabe der Mediendaten innerhalb des geometrischen Rahmens festgelegt. Somit können die Layoutbeschreibungen als Projektionsflächen der Mediendaten interpretiert werden, so dass sie sowohl für die Definition der virtuellen als auch der realen Wahrnehmungsumgebung verwendet werden können. Durch die Strukturierung elementarer Layoutsichten können komplexe Layoutsichten beschrieben werden.

Die Aspekte der Layoutsicht spiegeln die Arbeitsweise der Graphikdesigner bzw. Szenengestalter wider, die durch gängige 2D-Layoutprogramme wie *Quark Xpress* [Dout95] bzw. *Adobe Illustrator* [ScSa00] bzw. durch 3D-Programme wie *3D Studio Max* [Vels00] mit unterschiedlicher Granularität unterstützt werden und den in Abb. 5.2.2.1.1 dargestellten Abstraktionsniveaus während des Entwurfs entsprechen.

Während des 2D-Entwurfs kann durch Rastervorlagen die Positionierung bzw. durch Formatvorlagen das Erscheinungsbild und die Darstellungsart einzelner Layoutelemente bestimmt werden. Durch so genannte Bibliotheksvorlagen lassen sich elementare und komplexe Layoutelemente in eine Seite einfügen, so dass lediglich die Positionierung vorzunehmen ist. Die Wiederverwendbarkeit bereits vollständig definierter elementarer und komplexer Layoutsichten als Vorlage für einzelne Seiten wird durch so genannte Musterseiten unterstützt [Khaz95].

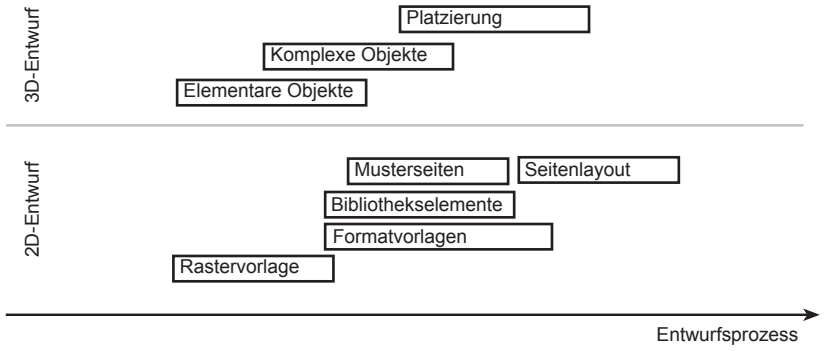


Abb. 5.2.2.1.1: Ablauf des Entwurfsprozesses aus Sicht der Graphikdesigner und 3D-Modellierer.

Der Entwurf dreidimensionaler Szenen kann sowohl für die Gestaltung virtueller Umgebungen als auch für die Gestaltung realer Wahrnehmungsumgebungen, die zunächst ebenfalls als virtuelle (Test-)Umgebungen modelliert werden, eingesetzt werden. Wie in Abschnitt 5.3.3 noch näher betrachtet, findet erst während der Abbildung die eigentliche Zuordnung auf eine verteilte Präsentationsumgebung statt. Typischerweise basiert der 3D-Entwurf zunächst auf der Erstellung einzelner elementarer und komplexer Objekte, die anschließend in der Szene platziert werden. 3D-Programme unterstützen dabei zum Teil, wie etwa *3D Studio Max*, die Erstellung von Skripten, die mit den Formatvorlagen von Layoutprogrammen vergleichbar sind. Auch wenn die Skripte für die unterschiedlichen Aspekte separat zu definieren sind, so

stellen sie zurzeit allerdings noch keine zentrale Veränderung der Parameter zur Verfügung, da diese durch die Skripte in den Layoutobjekten direkt gesetzt werden.

5.2.2.2 Repräsentationsmodell

```
<!ELEMENT LayoutView  (MedianElementId, Children?
                       GeometryId?, AppearanceId?, TransformId?, StyleId?)>
<!ELEMENT MedianElementId EMPTY>
<!ELEMENT Children (LayoutView*)>
<!ELEMENT GeometryId EMPTY>
<!ELEMENT TransformId EMPTY>
<!ELEMENT StyleId EMPTY>

<!ATTLIST LayoutView
        id CDATA #REQUIRED
<!ATTLIST MedianElementId
        name CDATA #REQUIRED >
<!ATTLIST GeometryId
        name CDATA >
<!ATTLIST AppearanceId
        name CDATA >
<!ATTLIST TransformId
        name CDATA>
<!ATTLIST StyleId
        name CDATA #REQUIRED >
```

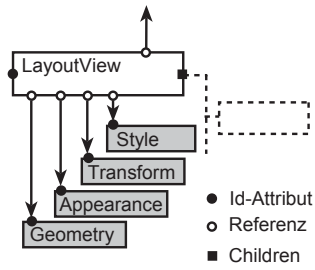


Abb. 5.2.2.2.1: DTD der Layoutsicht.

Da die oben beschriebenen Aspekte der Layoutsicht unabhängig voneinander sind und während der Definition unterschiedlich verwendet werden, werden die Aspekte innerhalb des Repräsentationsmodells einzeln abgespeichert. Somit können einzelne Aspekte von mehreren Layoutsichten referenziert werden, wodurch die Wiederverwendbarkeit und Erweiterbarkeit unterstützt wird. In MDL verweist eine Layoutsicht somit, wie in Abb. 5.2.2.2.1 dargestellt, auf eine Geometriebeschreibung (Geometry), auf die Beschreibung des Erscheinungsbilds des Layoutelements (Appearance), die Position (Transform) und auf die Definition der Darstellungsart (Style) des zu projizierenden Mediums. Die Layoutstruktur wird durch die Schachtelung der Sichten repräsentiert.

```
<!ELEMENT Geometry ( (X2D | X3D), size) >
<!ATTLIST Geometry id CDATA #REQUIRED>

<!ELEMENT X2D ( Plane | Oval | Polygon) >
<!ELEMENT X3D (Box | Cone | Cylinder | ElevationGrid | …)>
<!ELEMENT Plane EMPTY>
<!ELEMENT Oval EMPTY>

<!ATTLIST Plane
            height      N #REQUIRED
            width       N #REQUIRED>
<!ATTLIST Oval
            height      N #REQUIRED
            width       N #REQUIRED>
```

Abb. 5.2.2.2.2: DTD der Geometriebeschreibung zwei- und dreidimensionaler Layoutelemente.

Die Form eines Layoutelements kann durch unterschiedliche Geometrien beschrieben werden. Dabei stehen sowohl zweidimensionale Formen, die durch Layoutprogramme unterstützt werden, als auch die durch die *Extensible 3D Language (X3D)* bzw. *Virtual Reality*

Modeling Language (VRML) standardisierten dreidimensionalen Objekte zur Verfügung. Wie bereits weiter oben beschrieben, lassen sich durch dreidimensionale Objekte räumliche Wahrnehmungsumgebungen gestalten, denen die zweidimensionalen Wiedergabebereiche zugeordnet werden. Die einzelnen Bereiche werden dabei durch ihre Form und Größe definiert, was der Definition elementarer Bibliothekselemente entspricht, die innerhalb einer Fläche frei platziert werden können.

```
<!ELEMENT Appearance (X2DMaterial | X3Dmaterial | GUI)>
<!ATTLIST Appearance id CDATA #REQUIRED>
<!ELEMENT X2DMaterial (Color, Intensity)>
<!ELEMENT X3DMaterial (AmbientIntensity, … , EmissiveColor, Intensity)>
<!ELEMENT GUI (EMPTY)>


<!ELEMENT Color EMPTY>
<!ELEMENT Intensity EMPTY>

<!ATTLIST Color
        red [0,256] #REQUIRED
        green [0,256] #REQUIRED
        blue [0,256] #REQUIRED >
<!ATTLIST Intensity
        value [0, 100] #REQUIRED>
<!ATTLIST GUI
        name (“button” | “menu” | “radiobutton” |
                “checkbox” | “hSlider” | “vSlider”) #REQUIRED>
```

Abb. 5.2.2.2.3: DTD der Materialeigenschaft von Layoutelementen.

Durch die Interpretation der Layoutelemente als Projektionsflächen der wiederzugebenden Mediendaten wird die Gestaltung der Layoutelemente selbst unterstützt, d.h. jedes Layoutelement kann eine so genannte Materialeigenschaft besitzen. Bei zweidimensionalen Layoutelementen wird die Materialeigenschaft eines Layoutelements, wie in Abb. 5.2.2.2.3 dargestellt, durch die Hintergrundfarbe und die Transparenz (Intensity) dieser Farbe beschrieben. Um das Erscheinungsbild von dreidimensionalen Layoutobjekten zu bestimmen, stehen die durch X3D definierten Aspekte zur Verfügung. So können Objekten z.B. eine Reflexionseigenschaft (AmbientIntensity), Farbe (EmissiveColor) und Transparenz (Intensity) zugewiesen werden.

```
<!ELEMENT Transform (Rotation, Translation)>
<!ATTLIST Transform id CDATA #REQUIRED>

<!ELEMENT Rotation EMPTY>
<!ELEMENT Translation EMPTY>

<!ATTLIST Translation
        x   CDATA #REQUIRED
        y   CDATA #REQUIRED
        z   CDATA>
<!ATTLIST Rotation
        x   CDATA #REQUIRED
        y   CDATA #REQUIRED
        z   CDATA
        angle [0,360] #REQUIRED>
```

Abb. 5.2.2.2.4: DTD der Platzierungsbeschreibung von Layoutelementen.

Neben der Beschreibung des Erscheinungsbilds als Projektionsfläche können Layoutelemente als standardisierte Interaktionselemente (button, menu, radiobutton, ...) für die Gestaltung von Benutzeroberflächen (GUI) fungieren. Hierbei handelt es sich im Gegensatz zu der vorhe-

rigen expliziten Beschreibung um eine implizite Beschreibung, da die eigentliche graphische Umsetzung dem Präsentationsmedium obliegt.

Die Positionierung der Layoutelemente wird, wie in Abb. 5.2.2.2.4 dargestellt, durch die Position und Rotation relativ zu dem übergeordneten Layoutelement repräsentiert. Hierdurch können komplexe Layoutsichten als zusammengefasste Elemente in einer zwei- bzw. dreidimensionalen Beschreibung gemeinsam positioniert werden, so dass insbesondere zweidimensionale Seitenbeschreibungen als komplexe Layoutsichten innerhalb einer dreidimensionalen Szene eingefügt werden können. Die Position eines Layoutelements wird, wie in 3D-Programmen üblich, durch die Verschiebung (Translation) relativ zu dem Bezugspunkt des übergeordneten Elements ausgegeben. Die Rotation wird durch einen Bezugspunkt bzw. Vektor und den Rotationswinkel definiert.

```
<!ELEMENT Style (Picture | Text | Sound)>
<!ATTLIST Style id CDATA #REQUIRED>

<!ELEMENT Picture (Translation, (Scale | Automatic), Intensity)>
<!ELEMENT Translation EMPTY>
<!ELEMENT Scale EMPTY>
<!ELEMENT Automatic EMPTY>
<!ELEMENT Intensity EMPTY>

<!ATTLIST Offset
        x N #REQUIRED
        y N #REQUIRED>
<!ATTLIST Scale
        height N #REQUIRED
        width N #REQUIRED>
<!ATTLIST Automatic
        kind (“height” | “width”) #REQUIRED >
<!ATTLIST Intensity
        value [0,100] #REQUIRED >


<!ELEMENT Text (Font, Style, Size, Color)>
<!ELEMENT Font EMPTY>
<!ELEMENT Style EMPTY>
<!ELEMENT Size EMPTY>
<!ELEMENT Color EMPTY>

<!ATTLIST Font
        name CDATA #REQUIRED >
<!ATTLIST Style
        name (“normal” | “bold” | “italics”) #REQUIRED >
<!ATTLIST Size
        value CDATA #REQUIRED >
<!ATTLIST Color
        red [0,256] #REQUIRED
        green [0,256] #REQUIRED
        blue [0,256] #REQUIRED >


<!ELEMENT Sound (Intensity)>
<!ELEMENT Intensity EMPTY>

<!ATTLIST Intensity
        value [0,100] #REQUIRED >
```

Abb. 5.2.2.2.5: DTD für die Beschreibung der Wiedergabe von bildhaften, textuellen und akustischen Mediendaten.

Die bisher betrachteten Aspekte sind unabhängig von den Mediendaten zu bestimmen, da durch diese ausschließlich die Projektionsfläche für die Mediendaten festgelegt wird. Die Wiedergabe der Daten innerhalb eines Layoutelements ist hingegen von dem Medientyp abhängig, so dass in MDL, wie in Abb. 5.2.2.2.5 dargestellt, zwischen bildhaften Mediendaten (Image, Movie, Animation), Texten (Text) und akustischen Mediendaten (Sound), bei denen nur die Lautstärke (Intensity) definiert werden kann, unterschieden wird.

Bei der Darstellung von Texten wird ein Text als geschlossenes Medium betrachtet, so dass dieser als Ganzes formatiert wird. Hierbei lässt sich, wie in Abb. 5.2.2.2.5 dargestellt, der Schrifttyp (Font), der Schriftschnitt (Style), die Größe der Schrift (Size) und die Farbe (Color) bestimmen.

Bei der Definition der Darstellung bildhafter Mediendaten innerhalb eines Layoutelements kann der so genannte Versatz, die Darstellungsgröße und die Transparenz des Mediums festgelegt werden. Die standardmäßige Darstellung gibt ein Medium in Originalgröße ohne Transparenz beginnend in der linken oberen Ecke des Elements wieder. Durch den Versatz (Offset) kann der Bezugspunkt der Darstellung in x- und/oder y-Richtung verschoben werden. Um die Darstellungsgröße des Mediums zu bestimmen, kann dieses in seiner Breite (width) und/oder Höhe (height) skaliert werden bzw. der Größe des Layoutrahmens angeglichen werden, so dass durch eine automatische Skalierung die Darstellung der Breite oder der Höhe des Layoutrahmens entspricht. Mithilfe der Transparenz (Intensity) kann festgelegt werden, inwieweit durch das Medium tiefer liegende Ebenen verdeckt werden sollen.

5.2.3 Verhaltenssicht

5.2.3.1 Manipulationsmodell

Die Verhaltenssicht spiegelt die Sicht der Programmierer wider, so dass diese Sicht durch Entwurfswerkzeuge für die Programmierung unterstützt wird. Wie bereits in Kapitel 4 gezeigt, können dabei unterschiedliche Programmierparadigmen für die Definition des Verhaltens verwendet werden. Allerdings unterstützen die zurzeit gängigen Präsentationsmedien wie CD-ROM Programme und 3D-Präsentationsprogramme lediglich eine imperative Definition des Verhaltens, so dass die Beschreibung des Verhaltens für eine automatische Transformation auf einer standardisierten Befehlsmenge basieren muss.

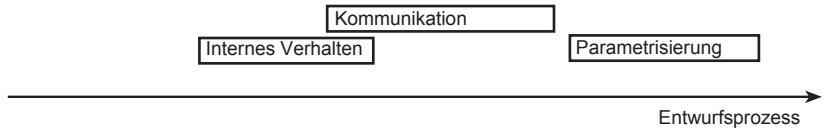


Abb. 5.2.3.1.1: Ablauf des Entwurfsprozesses aus Sicht der Programmierer.

Zur Unterstützung einer standardisierten Beschreibung des Verhaltens kann das für die Erstellung von CD-ROMs bekannte Programm *iShell* [iShe01] verwendet werden, da es eine für multimediale Anwendungen eingeschränkte Befehlsmenge zur Verfügung stellt. Die Definition erfolgt hierbei, wie in Abschnitt 4.2.3 bereits erwähnt, durch die Beschreibung einfacher Reaktionsmuster einzelner (Medien-)Verwalter, die durch Nachrichten anderer Verwalter und durch Benutzerinteraktionen ausgelöst werden. Als Reaktionsmuster stehen die in Abschnitt 2.2.2 beschriebenen Ereignisse und Aktionen, wie z.B. die Änderung der Position

eines Verwalters und der Austausch einzelner Mediendaten zur Verfügung. Die Verhaltensbeschreibung setzt sich somit aus unterschiedlichen Aspekten zusammen, die die (interne) funktionale Verbindung zu der Medien- und Layoutsicht eines Mittlerelements, die Interaktionsfähigkeiten und die Kommunikationsbeziehungen zu anderen Mittlerelementen beschreiben.

Um neben dieser standardisierten Beschreibung eine flexible Beschreibungsmöglichkeit zu unterstützen, wird durch die Verhaltenssicht, vergleichbar mit der komponentenbasierten Modellierung, zwischen der Modellierung und der eigentlichen Implementierung unterschieden. Wie in Abschnitt 4.2.1 gezeigt, können somit neben einer standardisierten Beschreibung unterschiedliche Programmiersprachen für spezielle Präsentationsmedien eingesetzt werden.

Auch wenn die Verhaltenssicht nicht wie bei der Layoutsicht eine fest vorgegebene Aufteilung in Teilaspekte vorschreibt, so bietet sich die in Abb. 5.2.3.1.1 dargestellte Aufteilung des internen Verhaltens, der Kommunikation zwischen einzelnen Mittlerelementen und der eigentlichen Parametrisierung in Teilaspekte an. Hierdurch ist es z.B. möglich, innerhalb eines Aspekts das interne Grundverhalten eines Bedienungselements zu definieren, das das Austauschen der entsprechenden Mediendaten beschreibt, und durch einen zweiten Aspekt die bei der Aktivierung eines Bedienungselements auszuführenden Aktionen zu bestimmen.

Durch die getrennte Repräsentation einzelner Verhaltensaspekte lässt sich ebenso das Verhalten innerhalb eines vorgegebenen Modells bereits während der Definitionsphase beschreiben. Durch ein gegebenes Modell wird die virtuelle Umgebung einer Wahrnehmungsumgebung definiert, so dass derartige Verhaltensaspekte, wie in Abschnitt 5.3.3.1 noch näher gezeigt, die Verbindung zwischen der Interpretation der Modelldaten und dem gewählten Modell beschreiben.

5.2.3.2 Repräsentationsmodell

Wie in Abb. 5.2.3.2.1 dargestellt, kann eine elementare Verhaltenssicht unterschiedliche Verhaltensbeschreibungen referenzieren. Diese aus der subjektorientierten Programmierung bekannte Beschreibungsform ermöglicht es das Verhalten in unterschiedliche Aspekte aufzuteilen. Im Gegensatz zu der objektorientierten Programmierung können dadurch die einzelnen Aspekte unabhängig voneinander definiert werden, da das Zusammenfügen der Aspekte erst während der Abbildung auf ein Präsentationsmodell stattfindet.

```
<!ELEMENT BehaviorView  (MedianElementId, Children?, CodeId *)>
<!ELEMENT MedianElementId EMPTY>
<!ELEMENT Children (BehaviorView*)>
<!ELEMENT CodeId EMPTY>

<!ATTLIST BehaviorView
          id CDATA #REQUIRED
<!ATTLIST MedianElementId
          name CDATA #REQUIRED >
<!ATTLIST CodeId
          name CDATA >
```

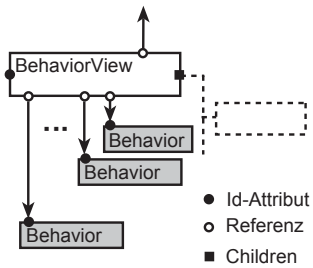


Abb. 5.2.3.2.1: DTD der Verhaltenssicht.

Ebenso wie Layoutsichten lassen sich elementare Verhaltenssichten zu komplexen Sichten verbinden. Die Hierarchie spiegelt dabei die Grundstruktur der Kommunikation wider, so dass sie als Kommunikationskanal zu interpretieren ist. Wie weiter unten noch näher betrachtet, wird durch diese Trennung zwischen Verhaltensaspekt und Kommunikationsreferenz die Wiederverwendbarkeit von Verhaltensaspekten unterstützt.

```
<!ELEMENT Behavior (Interface , (Script | Inline))>
<!ELEMENT Interface … >
<!ELEMENT Script (EMPTY)>
<!ELEMENT Inline … >

<!ATTLIST Behavior
                id CDATA #REQUIRED>
<!ATTLIST Script
                language CDATA #REQUIRED
                href CDATA #REQUIRED>
```

Abb. 5.2.3.2.2: DTD eines Verhaltensaspekts.

Damit externe und interne Verhaltensbeschreibungen gleich behandelt werden können, setzt sich ein Verhaltensaspekt aus einer Schnittstellenbeschreibung und der eigentlichen Definition des Verhaltens zusammen. Die funktionale Definition kann dabei durch MDL, wie in Abb. 5.2.3.2.2 dargestellt, selbst repräsentiert werden (Inline) oder durch eine Referenz (Script) auf eine Verhaltensbeschreibung in einer anderen Programmiersprache vorliegen.

```
<!ELEMENT Interface (Communication ?, Interaction ?)>
<!ELEMENT Communication (MessageIn, MessageOut)>
<!ELEMENT MessageIn (EventDef *)>
<!ELEMENT MessageOut (EventDef *)>
<!ELEMENT Interaction (   MouseDown | MouseUp | MouseMove |
                          MouseDrag | MouseEnter | MouseLeave | Text | Value)*>

<!ATTLIST EventDef
                valueName_1 CDATA
                …
                valueName_n CDATA>
<!ATTLIST Mouse
                x CDATA #REQUIRED
                y CDATA #REQUIRED
                z CDATA >
<!ATTLIST Text
                kind CDATA #REQUIRED>
                text CDATA #REQUIRED>
<!ATTLIST Value
                kind CDATA #REQUIRED>
                value R #REQUIRED>
```

Abb. 5.2.3.2.3: DTD-Struktur der Interface-Beschreibung.

Die Aufteilung in externe Kommunikation zwischen den Mittlerelementen und der internen Definition des Verhaltens findet sich in der Schnittstellenbeschreibung wieder. So werden im ersten Teil die durch die Verhaltenbeschreibung zu empfangenden (MessageIn) und die zu sendenden (MessageOut) Kommunikationsereignisse (*EventDef*) definiert. Im zweiten Teil (Interaction) werden die Interaktionsereignisse aufgeführt, auf die der Verhaltensaspekt reagieren kann. Hierfür stehen die in Abb. 5.2.3.2.3 aufgeführten standardisierten Interaktionsereignisse zur Verfügung, die durch Mausaktionen charakterisiert sind bzw. ein Ereignis durch eine Text- oder Wertübergabe beschreiben.

```
<!ELEMENT Inline (Properties?, Events?)>

<!ELEMENT Properties (Property)>
<!ELEMENT Property ((Property*)?)>

<!ELEMENT Events (Event *)>
<!ELEMENT Event ((EventDef | MouseDown | MouseUp | …), Commands)>
<!ELEMENT Commands (Message | Action | Set | If)*>
<!ELEMENT Message … >
<!ELEMENT Action … >
<!ELEMENT Set (PropId | Properties | Property)>
<!ELEMENT If ((PropId | Property), (PropId | Property), Commands, (Else, Commands)?)>
<!ELEMENT PropId (EMPTY)>

<!ATTLIST Set
                id CDATA #REQUIRED>
<!ATTLIST If
                relation (“=” | “<” | “<=” | “>” | “>=”) #REQUIRED>
<!ATTLIST Property
                id        CDATA #REQUIRED
                value_1 CDATA
                …
                value_n CDATA>
<!ATTLIST PropId
                nameId CDATA #REQUIRED
                valueId CDATA>
```

Abb. 5.2.3.2.4: DTD für die interne Beschreibung von Ereignis-Aktions-Beziehungen.

Um die Repräsentation der Verhaltensbeschreibung innerhalb von MDL zu unterstützen und damit eine automatische Transformation zu ermöglichen, lässt sich das Verhalten durch die in Abschnitt 2.2.2 eingeführten Ereignis-Aktions-Beziehungen beschreiben. Wie in Abb. 5.2.3.2.4 dargestellt, wird ein Verhaltensaspekt durch eine Zustandsbeschreibung (Properties) und die Auflistung der funktionalen Beschreibung (Events) repräsentiert.

Eine funktionale Beschreibung setzt sich aus einem Funktionskopf bzw. einer Ereignisbeschreibung und den auszuführenden Kommandos als Funktionsrumpf zusammen. Die Ereignisbeschreibung entspricht dabei den in der Schnittstellenbeschreibung aufgeführten Kommunikations- (*EventDef*) oder Interaktionsereignissen (MouseDown, ...). Der Funktionsrumpf (Commands) setzt sich aus einer Folge von zu versendenden Nachrichten und auszuführenden Aktionen innerhalb eines Mittlerelements zusammen, deren Ausführung durch Bedingungsabfragen (If ... Else) kontextabhängig gesteuert werden kann. Somit kann z.B. das Verhalten in Abhängigkeit des internen Zustands oder in Abhängigkeit der Benutzereingabe variieren, wobei der interne Zustand verändert (Set) bzw. gelesen (PropId) werden kann.

Die Beschreibung der Kommunikation zwischen Mittlerelementen findet mithilfe von Nachrichten statt. Eine Kommunikation wird somit, wie in Abb. 5.2.3.2.5 dargestellt, durch einen Empfänger (Receiver) und eine Inhaltskomponente (Property) definiert. Hierdurch wird es möglich, erst während der Transformation in ein Präsentationsmedium festzulegen, ob die Kommunikation als Funktionsaufruf wie bei der objektorientierten Programmierung, als Ereignis, wie bei der komponentenbasierten Modellierung oder als Inhaltsdefinition eines Sprechakts, wie bei der agentenorientierten Modellierung, umgesetzt wird.

```
<!ELEMENT Message (Receiver, Property)>
<!ELEMENT Parameter (EMPTY)>

<!ELEMENT Receiver (Root | Children | Ancestor | Sibling | Id)>
<!ELEMENT Root (EMPTY)>
<!ELEMENT Children (EMPTY)>
<!ELEMENT Ancestor (EMPTY)>
<!ELEMENT Sibling (EMPTY)>
<!ELEMENT Id (EMPTY)>

<!ATTLIST Id  name CDATA #REQUIRED>
```

Abb. 5.2.3.2.5: DTD der Kommunikationsbeschreibung durch Nachrichtenaustausch.

Mittlerelemente können direkt (Id) über ihren Namen bzw. indirekt referenziert werden, um als Empfänger einer Nachricht zu fungieren. Wie bereits weiter oben erwähnt, spiegelt die Hierarchie komplexer Verhaltenssichten die Grundstruktur der Kommunikation wider, so dass sie mit den in Abschnitt 4.2.2.2 beschriebenen hierarchischen Ereigniskanälen zu vergleichen ist. Ein Empfänger einer Nachricht kann somit über die Definition des zu verwendenden Kommunikationskanals indirekt bestimmt werden.

In MDL stehen für die Definition des Kommunikationskanals das Wurzelelement (Root) einer komplexen Sicht, die Kinder einer Sicht (Children), die übergeordnete Sicht (Ancestor) und die Sichten der gleichen Hierarchieebene (Sibling) zur Verfügung. Neben der dadurch vereinfachten Beschreibung mehrerer Empfänger wird durch die indirekte Referenzierung die Wiederverwendbarkeit einzelner Verhaltensaspekte unterstützt. So kann z.B. das Verhalten von Interaktionselementen zentral definiert werden, indem diese Elemente als Kinder die zu steuernden Verhaltenssichten zugewiesen bekommen.

```
<!ELEMENT Action (Translation | Rotation | … | Enable | Replace | MediaSpeed | MediaTime)>
<!ELEMENT Translation (EMPTY)>
…
<!ELEMENT Enable (EMPTY)>
<!ELEMENT Replace (EMPTY)>
<!ELEMENT MediaSpeed (EMPTY)>
<!ELEMENT MediaTime (EMPTY)>

<!ATTLIST Translation
                x N #REQUIRED
                y N #REQUIRED
                z N>
…
<!ATTLIST Enable
                value (“true” | “false” | “toggle”) #REQUIRED>
<!ATTLIST Replace
                name CDATA #REQUIRED>
<!ATTLIST MediaSpeed
                value N #REQUIRED>
<!ATTLIST MediaSpeed
                value N+ #REQUIRED>
```

Abb. 5.2.3.2.6: DTD der Aktionen zwischen den einzelnen Sichten.

Mithilfe von so genannten Aktionen lassen sich die Aspekte der einzelnen Sichten innerhalb eines multimedialen Mittlerelements anstoßen. Hierzu stehen, wie in Abb. 5.2.3.2.6 dargestellt, die den Parametern der Aspekte entsprechenden Aktionen (Translation, Rotation, …) zur Verfügung, so dass sich alle Parameter durch Aktionen während der Laufzeit verändern lassen.

Neben diesen durch die Medien- und Layoutsicht bereits beschriebenen Parametern stehen weitere Aktionen zur Verfügung. So kann ein Mittlerelement als aktives bzw. passives (Enable) Interaktionselement auf Benutzerinteraktionen reagieren bzw. Benutzerinteraktionen ignorieren. Einzelne Mittlerelemente lassen sich austauschen (Replace), so dass z.B. eine neue Präsentationsseite aufgerufen werden kann. Um die Steuerung zeitkontinuierlicher Medien zu unterstützen, können sowohl deren Abspielgeschwindigkeit (MediaSpeed) als auch deren Abspielzeitpunkt (MediaTime) bestimmt werden.

5.3 Realisierungsphase

Wie bereits in Abschnitt 5.1.1 erwähnt, wird in einem zweiten Schritt die eigentliche Realisierung einer multimedialen Anwendung durch die Ablauforganisation ermöglicht, die durch die Abbildung der Repräsentationsbeschreibung in eine durch das Präsentationsmodell unterstützte Sprache erfolgt. Die Abbildung auf „traditionelle“ Präsentationsmodelle erfolgt nach den Prinzipien der subjektorientierten Programmierung, bei der einzelne Sichten innerhalb eines Software-Elements integriert werden, weshalb hier nur die Grundmechanismen vorgestellt werden. Wie in Abschnitt 5.1.4 gezeigt, erfordert dahingegen die Realisierung einer verteilten Wahrnehmungsumgebung die Zuordnung interner bzw. externer Präsentationsmedien und der virtuellen Modellumgebung, wodurch eine Ergänzung des Repräsentationsmodells nötig wird.

Neben der vollständigen Abbildung der Repräsentationsbeschreibung können zum Zweck des Tests aber auch lediglich Teilaspekte der zu entwerfenden Anwendung in das jeweilige Präsentationsmodell abgebildet werden. So können z.B. nur die Layout- und Mediensichten zusammengeführt werden, um das Erscheinungsbild der Anwendung zu überprüfen bzw. bei der Realisierung von Wahrnehmungsumgebungen kann diese zunächst als virtuelle dreidimensionale Umgebung getestet werden, bevor die eigentliche Umsetzung in eine verteilte Ablaufumgebung mit externen Ein- und Ausgabegeräten stattfindet.

5.3.1 Kompositions- und Transformationsregeln

Um unterschiedliche Präsentationsmedien unterstützen zu können, lässt sich die Abbildung auf ein Präsentationsmodell durch Regeln definieren, die zum einen die Beziehungen zwischen den Mittlerelementen und Sichten bzw. zwischen den Sichten und Aspekten, zum anderen die eigentliche Übersetzung in die Präsentationssprache beschreiben.

Die Regeln werden während der Abbildung durch ein Programm interpretiert, das die interne Manipulation der MDL-Repräsentation durch das *Document Object Model (DOM)* [Brad98] unterstützt. Durch DOM wird eine von dem *World Wide Web Konsortium* standardisierte Schnittstelle zur Verfügung gestellt, die den Zugriff auf XML-Konstrukte ermöglicht.

Wie in Abb. 5.3.1.1 dargestellt, werden durch das Programm die MDL-Repräsentationen eingelesen, entsprechend der Abbildungsregeln über die DOM-Schnittstelle manipuliert und letztendlich in der Zielsprache abgespeichert. Um die Beziehungen zwischen den Repräsentations-Elementen und die eigentliche Sprachtransformation in einheitlicher Weise definieren zu können, wird die Abbildung durch einen Regelsatz aus einzelnen so genannten Kompositions- bzw. Transformationsregeln definiert.

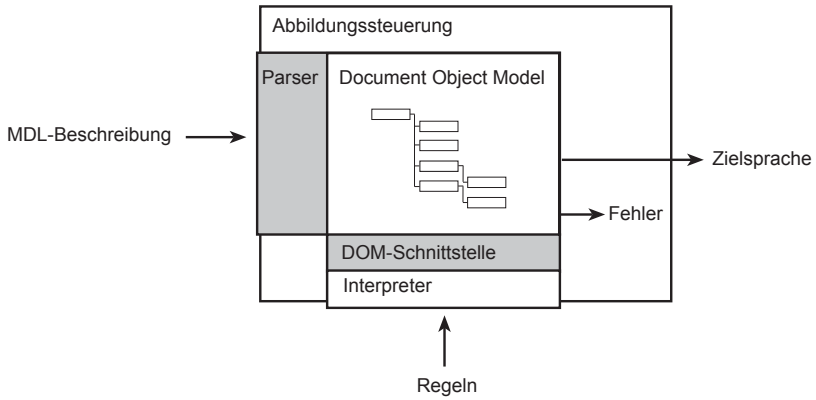


Abb. 5.3.1.1: Schematische Darstellung der Abbildung des Repräsentationsmodells auf ein Präsentationsmodell.

Durch Kompositionsregeln lassen sich die Beziehungen zwischen den einzelnen Sichten und Aspekten der MDL-Repräsentation näher spezifizieren. Eine Kompositionsregel wird dabei, wie in Abb. 5.3.1.2 dargestellt, durch eine gerichtete Beziehung zwischen einem Zielelement (TargetElement) und einem zu integrierenden Element (ResourceElement) beschrieben. Beide Bezugselemente können durch ihre Attribute und übergeordneten Elemente näher spezifiziert werden, so dass die Regeln als Schablone für eine Menge von Beziehungen, aber auch für spezielle Beziehungen definiert werden können. Wie in Abschnitt 5.3.3 noch näher gezeigt, können somit einzelne Aspekte der MDL-Repräsentation für spezielle Aus- bzw. Eingabegeräte näher bestimmt werden. Die Auflösung möglicher Konflikte zwischen mehreren Regeln findet nach dem XML-Standard statt, d.h. spezielle Regeln besitzen gegenüber allgemeinen Regeln eine höhere Priorität.

```
<!ELEMENT CompositionRule ((Element | Any | TargetElement), ResourceElement)>
<!ELEMENT Element (Attribute*, (Element* | Any | TargetElement))>
<!ELEMENT Any (TargetElement)>
<!ELEMENT TargetElement (Attribute*)>
<!ELEMENT ResourceElement (Attribute*)>
<!ELEMENT Resource (Attribute*)>

<!ATTLIST CompositionRule
        priority N
        type (“merge” | “overwrite” | “select”)>
<!ATTLIST Element
        type CDATA #REQUIRED>
<!ATTLIST TargetElement
        type CDATA #REQUIRED>
<!ATTLIST ResourceElement
        type CDATA #REQUIRED>
<!ATTLIST Resource
        type CDATA #REQUIRED>
<!ATTLIST Attribute
        name CDATA #REQUIRED
        value (CDATA | “equal”) #REQUIRED>
```

Abb. 5.3.1.2: DTD von Kompositionsregeln.

Neben der Beschreibung der Bezugselemente einer Kompositionsregel wird durch die Semantik der Beziehung die Art der Komposition definiert. Wie in Abschnitt 5.2 gezeigt, werden z.B. durch komplexe Medienelemente alternative Darstellungen beschrieben, wohingegen

durch Layoutaspekte orthogonale Definitionen, die sich gegenseitig ergänzen, repräsentiert werden. Um die Art der Komposition zwischen den einzelnen Aspekten bzw. Sichten zu berücksichtigen, wird zwischen zusammenführenden (merge), überschreibenden (overwrite) und auswählenden (select) Kompositionsregeln unterschieden:

Zusammenführen: Die Komposition durch „Zusammenführen“ ist die am häufigsten verwendete Kompositionsregel. Wenn einzelne Sichten unterschiedliche orthogonale Aspekte einer Anwendung beschreiben, wird durch die Kompositionsregel eine Vereinigung der einzelnen Teile der Komposition beschrieben. Bei überlappenden Teilen einer Komposition müssen die Teile zusammengefügt werden, was z.B. im Fall des Verhaltens der Ausführung mehrerer Funktionen entspricht.

Überschreiben: Eine überschreibende Komposition entspricht dem aus der objektorientierten Programmierung bekannten Mechanismus der Vererbung, bei der geerbte Methoden durch neue Definitionen überschrieben werden können. Das Ersetzen findet bei Namensgleichheit statt, die sich auf die unterschiedlichen Granularitäten des Repräsentationsmodells beziehen kann. Somit können Sichten, Aspekte und Parameter von Aspekten überschrieben bzw. ersetzt werden.

Auswählen: Durch eine Auswahl-Regel lassen sich kontextabhängige Aspekte der Repräsentation extrahieren, d.h. in Abhängigkeit der zu präsentierenden Daten bzw. der Realisierung der Wahrnehmungsumgebung können einzelne Aspekte ausgewählt werden. Die Auswahl kann bereits statisch während der Transformation in ein Präsentationsmodell bzw. dynamisch während der Laufzeit der Anwendung stattfinden. Eine dynamische Auswahl kann z.B. durch Parameter der Ablaufumgebung selbst bestimmt sein, so dass spezielle benutzerspezifische Aspekte, wie die Auswahl einer Sprache, berücksichtigt werden können.

Können die Kompositionsregeln unabhängig von der Präsentationssprache definiert werden, so erfordert die syntaktische Transformation eine flexible Beschreibung der Regelspezifikationen, die auf der durch die *Extensible Style Language (XSL)* [xsl99] unterstützten Skriptsprache *JavaScript* [Koch99] basiert.

```
<!ELEMENT TransformationRule ((Element | Any | TargetElement), Specification)>
<!ELEMENT TargetElement (Attribute*)>
<!ELEMENT Specification (EMPTY)>

<!ATTLIST Target-Element
        type CDATA #REQUIRED>
<!ATTLIST Attribute
        name CDATA #REQUIRED
        value (CDATA | “equal”) #REQUIRED>
```

Abb. 5.3.1.3: DTD von Transformationsregeln.

Die Transformationen lassen sich entsprechend den Kompositionsregeln auf einzelne Elemente definieren. Wie in Abb. 5.3.1.3 dargestellt, werden somit die Transformationsregeln durch ein Bezugselement (TargetElement) und einen in JavaScript definierten Ausführungsblock (Specification) beschrieben.

Funktion	Bedeutung
ancestor(elementType, element)	Nächstliegender Vorgänger mit dem in der Funktion übergebenen Namen
child(elementType, element)	Nächstliegender Nachfolger mit dem in der Funktion übergebenen Namen
Children()	Aufzählung aller direkten Nachfolger
GetTag(tagName)	Element mit dem gesuchten Namen
getAttributes()	Attributbezeichner des aktuellen Elements
getAttribute(attributeName)	Attributwert des Attributbezeichners

Tab. 5.3.1.1: Zusammenstellung der JavaScript-Erweiterungen.

Um den Zugriff auf die interne DOM-Repräsentation durch JavaScript zu unterstützen, stehen durch XSL Funktionen zur Verfügung, die die Kommunikation zwischen JavaScript und der DOM-Schnittstelle definieren. Dabei sind die Funktionen durch die aus der objektorientierten Programmierung bekannte Punktnotation innerhalb eines JavaScript-Ausdrucks aufzurufen. Wie in Tab. 5.3.1.1 zusammengefasst, kann mithilfe der Funktionen auf Vorgängerelemente, Kinderelemente, einzelne Elemente innerhalb der gleichen Hierarchieebene und Attribute eines Elements zugegriffen werden [BeMi98].

5.3.2 Abbildung auf geschlossene Präsentationsmedien

Bei der Abbildung auf geschlossene Präsentationsmedien werden die Sichten als Seiten- bzw. Szenenbeschreibungen interpretiert, die als Ganzes durch das Präsentationsprogramm wiedergegeben werden sollen. Da durch derartige Präsentationsmedien keine Trennung zwischen multimedialem Inhalt und Präsentationsumgebung unterstützt wird, können somit multimediale Mittlerelemente innerhalb derartiger Modelle nicht beschrieben werden. Vielmehr ist die Layoutsicht die vorherrschende Struktur der Präsentationsbeschreibung und dient daher als verbindende Sicht der Abbildung, in die die anderen Sichten integriert werden.

Bei der Abbildung auf geschlossene Präsentationsmedien lassen sich, wie in Abb. 5.3.2.1 schematisch dargestellt, grundsätzlich drei Schritte unterscheiden. Zunächst werden in einem ersten Schritt die einzelnen Aspekte den Sichten zugewiesen. In einem zweiten Schritt werden die Sichten ineinander integriert, so dass vergleichbar mit den Dokumentenmodellen ein Element durch seine Layout-, Mediendaten- und Verhaltensbeschreibung definiert wird. Im dritten Schritt findet die eigentliche Transformation in die jeweilige Präsentationssprache statt:

Aspekt-Komposition: Wie in Abschnitt 5.2 bereits gezeigt, werden die Beziehungen innerhalb der MDL-Repräsentation zwischen Sichtbeschreibung und Aspektrepräsentation mithilfe der Identifikationsattribute beschrieben. Somit können die Kompositionsregeln als Schablonen eines Aspekttyps definiert werden, die bei Namensgleichheit der Attributwerte einer elementaren Sicht den entsprechenden Aspekt zuweisen.

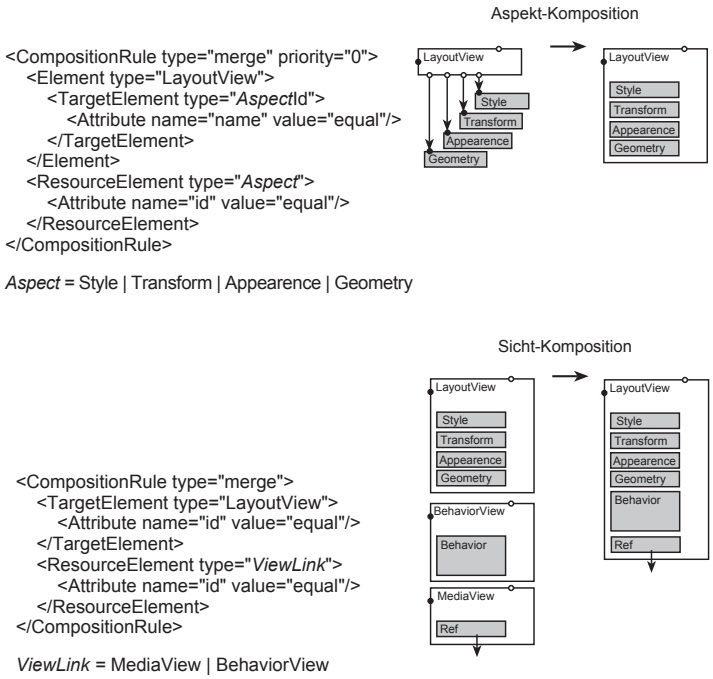


Abb. 5.3.2.1: Komposition der Aspekte.

Da die Layoutaspekte orthogonal zueinander definiert sind, werden diese, wie in Abb. 5.3.2.1 dargestellt, durch zusammenführende Kompositionsregeln integriert. Nach der Komposition der Layoutaspekte ist eine Layoutsicht somit durch die Parameter der Aspekte in eindeutiger Weise definiert.

Die Komposition der Mediendaten erfordert eine Selektion der zu präsentierenden Daten, da die zurzeit gängigen Präsentationsprogramme keine kontextabhängige Präsentation unterstützen. Durch die Kompositionsregeln der Mediensicht wird bereits eine 1:1-Beziehung zwischen Mediensicht und Mediendaten beschrieben, d.h. jede Mediensicht verweist nach der Komposition ausschließlich auf ein Medium.

Wie bereits in Abschnitt 5.2.4 beschrieben, kann das Verhalten mithilfe einer internen Repräsentation durch einfache Reaktionsmuster oder durch eine externe Skript- bzw.

Programmiersprache, die durch das Präsentationsmedium unterstützt wird, definiert werden. Die Komposition der Verhaltensaspekte findet durch überschreibende Regeln statt, so dass die Komposition, wie bereits im vorherigen Abschnitt erwähnt, der Vererbung aus der objektorientierten Implementierung entspricht.

Sicht-Komposition: Im zweiten Schritt werden die Sichten ineinander integriert. Wie bereits weiter oben erwähnt, wird die Layoutsicht als seiten- bzw. szenenorientierte Beschreibung innerhalb eines geschlossenen Präsentationsmediums angesehen, so dass sie die verbindende Struktur einer Präsentationsbeschreibung ist. Da die Sichten zueinander orthogonal sind, wird die Komposition durch „zusammenführende" Regeln beschrieben. Die Medien- und Verhaltenssicht wird somit durch Kompositionsregeln als ergänzende Beschreibung eines elementaren Medienverwalters angesehen und innerhalb der Layoutsicht, wie in Abb. 5.3.2.1 dargestellt, integriert.

Sprach-Transformation: Bei der eigentlichen Sprachtransformation einer Seiten- bzw. Szenenbeschreibung, die durch die Kompositionsregeln vollständig mithilfe der Layoutstruktur angegeben wird, werden die Parameter der Aspekte entsprechend der Zielsprache transformiert, so dass die Transformationsbeschreibung von dem Präsentationsmodell abhängt. Wie in Abb. 5.3.2.1 anhand einer Transformationsregel angedeutet, werden die Parameter der Beschreibung eines Medienverwalters eingelesen, innerhalb des Skripts interpretiert, entsprechend dem Präsentationsmodell übersetzt und abschließend in eine Datei geschrieben.

Bei der Transformation der internen Verhaltensbeschreibung muss neben den standardisierten Ereignissen und Aktionen, durch die das interne Verhalten der Medienverwalter beschrieben wird, ebenso die externe Kommunikation zwischen den Medienverwaltern, die durch Nachrichten repräsentiert wird, transformiert werden. Hierzu werden zunächst die indirekten Beziehungen der Kommunikationsbeschreibung in direkte Beziehungen umgeformt. Darauf aufbauend findet die Umsetzung der Nachrichtenkommunikation durch Funktionsaufrufe bei objektorientierten Präsentationsmodellen bzw. durch semantische Ereignisse bei komponentenorientierten Modellen statt.

5.3.3 Abbildung auf offene Präsentationsmedien

5.3.3.1 Beschreibung von externen Präsentationsmedien und Modellen

Wie bereits in Abschnitt 5.1.4 gezeigt, fungieren multimediale Mittlerelemente innerhalb einer Wahrnehmungsumgebung als Vermittler zwischen der virtuellen und realen Umgebung. Dabei treten die Mittlerelemente als verbindende Elemente auf, indem sie neben den drei Sichten der Mediendaten, des Layouts und des Verhaltens, die bereits durch den Entwurf definiert sind, externe Präsentationsmedien der realen Umgebung und Modelldaten der virtuellen Umgebung integrieren. Die technische Umsetzung einer Wahrnehmungsumgebung wird durch die im nachfolgenden Kapitel beschriebene verteilte Ablaufumgebung ermöglicht, in der die Mittlerelemente durch mobile Agenten repräsentiert werden.

Um die Integration externer Präsentationsmedien und Modelle während der Abbildung auf die Ablaufumgebung zu unterstützen, werden diese durch eine interne funktionale Beschreibung repräsentiert. Die funktionale Beschreibung basiert dabei auf spezialisierten Schnittstellenbeschreibungen und zugewiesenen Skriptdefinitionen, die mithilfe der in Abschnitt 6.3.1.3 vorgestellten imperativen Skriptsprache erfolgen.

```
<!ELEMENT ExternalBehavior  (Information, Interface, Scripts)>
<!ELEMENT Information (...)>
<!ELEMENT Interface (...)>
<!ELEMENT Scripts (ScriptName*)>

<!ATTLIST ExternalBehavior
        id CDATA #REQUIRED>
<!ATTLIST ScriptName
        language "JMTS"
        href CDATA>
```

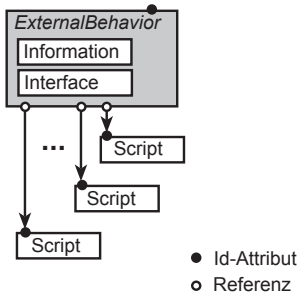


Abb. 5.3.3.1.1: DTD-Struktur von externen Präsentationsmedien und Modellbeschreibungen.

Die Definitionen sind somit mit den Verhaltensaspekten vergleichbar und können entsprechend wie diese entwickelt bzw. wieder verwendet werden. Ein derartiger externer Verhaltensaspekt setzt sich, wie in Abb. 5.3.3.1.1 dargestellt, aus einer Informationsbeschreibung, einer Interfacedefinition und den Referenzen auf die funktionalen Beschreibungen zusammen. Dabei kann eine Interfacedefinition als virtuelles Präsentationsmedium bzw. virtuelle Modellbeschreibung angesehen werden, die erst durch die Zuweisung der Verhaltensbeschreibungen instanziiert wird.

Die Ausprägungen einzelner externer Präsentationsmedien und Modellbeschreibungen ist sicherlich stark von dem Anwendungsszenarium abhängig, so dass hier ausschließlich der grundsätzliche Aufbau der Beschreibungen betrachtet wird. Spezielle Ausprägungen, die bereits von der verteilten Ablaufumgebung unterstützt werden, werden in Abschnitt 6.3 näher betrachtet.

```
<!ELEMENT Device  (Information, Interface, Scripts)>
<!ELEMENT Information (MediaType?, InteractionType?, AliasNames?)>
<!ELEMENT Interface (MediaControl, ProjectionControl, InteractionControl)>
<!ELEMENT Scripts (MediaScript, ProjectionScript, InteractionScript)>
<!ELEMENT AliasNames (Alias*)>

<!ATTLIST Device
        id CDATA #REQUIRED
<!ATTLIST MediaType
        type ("text","image", "image_sequence", ...)>
<!ATTLIST InteractionType
        type ("touch","move", "environment", ...)>
<!ATTLIST Alias
        name CDATA>
<!ATTLIST Script
        language "JMTS"
        href CDATA>
```

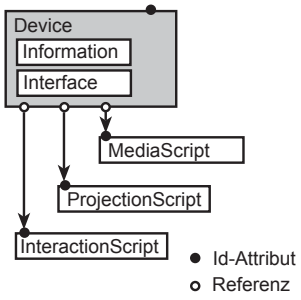


Abb. 5.3.3.1.2: DTD der Gerätebeschreibungen.

Ein externes Präsentationsmedium wird, wie in Abb. 5.3.3.1.2 dargestellt, durch die funktionale Beschreibung der Ansteuerung der Mediendaten (MediaScript) und der Projektionsfläche (ProjectionScript) sowie durch die funktionale Interpretation von externen Ereignissen

(InteractionScript) definiert. Durch die Trennung der Bereiche lässt sich eine Gerätebeschreibung aus bereits erstellten Definitionen, vergleichbar mit der Verhaltensbeschreibung, zusammensetzen.

Neben der funktionalen Beschreibung kann ein Präsentationsmedium durch weitere Informationen näher charakterisiert werden. Es können z.B. der von dem externen Präsentationsmedium unterstützte Medientyp (text, image, image_sequence, ..) und die Art der Interaktion (touch, move, environment, ...) beschrieben werden. Somit können Präsentationsmedien nicht nur durch ihre unterstützte Funktionalität, die durch die Schnittstellenbeschreibung gegeben ist, sondern auch durch ihre charakteristischen Eigenschaften bestimmt werden.

```
<!ELEMENT MediaControl (MediaSpeed?, MediaTime?, Switch?)>
<!ELEMENT MediaSpeed EMPTY>
<!ELEMENT MediaTime EMPTY>
<!ELEMENT Switch EMPTY>

<!ELEMENT ProjectionControl (Appearance?,Transform?, Style?)>
<!ELEMENT Appearance (X2DMaterial | X3DMaterial)>
…

<!ELEMENT InteractionControl (Enable?, MouseDown?, …Text?, Value?)>
<!ELEMENT Enable EMPTY>
<!ELEMENT MouseDown EMPTY>
…
```

Abb. 5.3.3.1.3: DTD der Schnittstellenbeschreibung eines externen Geräts.

Die Schnittstelle eines externen Ein- und Ausgabegeräts spiegelt dabei, wie bereits in Abschnitt 5.1.4 erwähnt, die Aspekte bzw. Parameter der Mediensicht und der Layoutsicht wider, so dass die Steuerung externer Geräte bzw. die Reaktion auf Ereignisse in gleicher Weise wie die interne Wiedergabe von Mediendaten bzw. wie die Interpretation von internen Ereignissen stattfinden kann.

```
<!ELEMENT Interface (MessageIn, MessageOut)>
<!ELEMENT MessageIn (EventDef *)>
<!ELEMENT MessageOut (EventDef *)>
<!ELEMENT EventDef (EventDef*)>

<!ATTLIST EventDef
                valueName_1 CDATA
                …
                valueName_n CDATA>
```

Abb. 5.3.3.1.4: DTD-Struktur der Schnittstellendefinition einer Modellbeschreibung.

Um das Verhalten innerhalb der Modellumgebung zu definieren bzw. die Interpretation von Modelldaten zu unterstützen, kann den Mittlerelementen eine Sicht auf das zugrundeliegende Modell zugewiesen werden. Eine Sicht ist wiederum durch eine Schnittstellenbeschreibung und die entsprechenden Definitionen des „Verhaltens“ aufgebaut. Im Gegensatz zu den Gerätebeschreibungen, die ein fest vorgegebenes Vokabular für die Schnittstellenbeschreibungen besitzen, werden die Schnittstellen von Modellsichten durch semantische Ereignisschnittstellen definiert. Wie in Abb. 5.3.3.1.4 dargestellt, ist die Schnittstellenbeschreibung mit der externen Kommunikationsschnittstelle der Verhaltensaspekte identisch, so dass die Kommunikation zwischen Modell und Mittlerelement frei zu definieren ist.

Durch die Ablaufumgebung werden, wie in Abschnitt 6.3.2.3 noch näher beschrieben, für die Definition von Modellsichten bereits vorgefertigte Schablonen zur Verfügung gestellt. Neben der Anbindung externer Programme und Datenbanken werden dabei bereits so genannte euklidische Modellsichten angeboten. Durch diese lassen sich die Mittlerelemente innerhalb eines zwei- bzw. dreidimensionalen Raums repräsentieren, wobei neben der Position ebenso Interaktionsereignisse übertragen werden können.

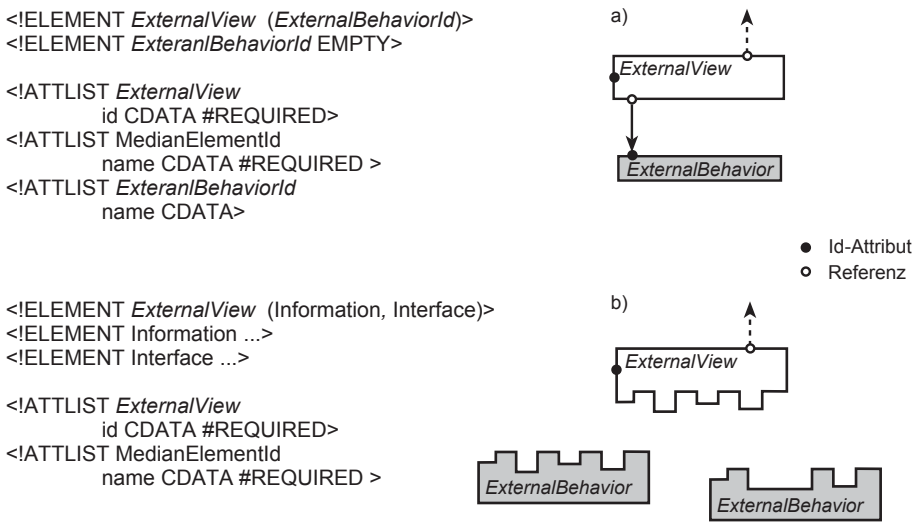


Abb. 5.3.3.1.5: Direkte (a) und indirekte (b) Zuordnung von Geräte- und Modellbeschreibungen.

Durch die Repräsentation der externen Präsentationsmedien und Modellsichten als spezialisierte Verhaltenaspekte lassen sich diese den Mittlerelementen zuweisen. So können bereits während des Entwurfs, wie in Abschnitt 5.2.3 erwähnt, vorliegende Modelle bzw. externe Präsentationsmedien innerhalb der Verhaltenssicht als spezielle Verhaltensaspekte integriert werden oder während der Abbildung auf eine verteilte Ablaufumgebung den Mittlerelementen zugewiesen werden.

Zum einen können die externen Verhaltensaspekte, genau wie während der Definitionsphase, durch Sichten mit den Mittlerelementen verbunden werden, zum anderen kann eine Gerätebeschreibung bzw. Modellbeschreibung als virtuelle Darstellung interpretiert werden. Hierbei werden, wie in Abb. 5.3.3.1.5 dargestellt, ausschließlich die Informationen und die Schnittstellenbeschreibung spezifiziert, die von einem Präsentationsmedium bzw. einer Modellbeschreibung erfüllt werden müssen. Die eigentliche Zuweisung einer speziellen Ausprägung findet dabei erst während der Laufzeit der Anwendung statt.

5.3.3.2 Bindung der Sichten

Im Gegensatz zu den geschlossenen Präsentationsmedien werden die Sichten sowie die externen Präsentationsmedien und Modelle in der im Rahmen dieser Arbeit entwickelten verteilten Ablaufumgebung erst während der Laufzeit gebunden. Dabei übernehmen die Mittlerelemente als aktive organisierende Elemente die Bindung des multimedialen „Inhalts“, der ihr Erscheinungsbild, Verhalten und die Wiedergabemöglichkeiten der Mediendaten bestimmt. Somit können die Mittlerelemente auf unterschiedlichen Ablaufumgebungen ver-

schiedene Ausprägungen des multimedialen Inhalts verwalten, so dass sie ihr Erscheinungsbild, Verhalten etc. verändern können.

Um die Mittlerelemente innerhalb des Präsentationsmodells einer verteilten Anwendung zu beschreiben, werden ihnen zum einen die für die Bindung der Sichten notwendigen Informationen zugewiesen, zum anderen müssen die Mittlerelemente den unterschiedlichen lokalen Ablaufumgebungen zugeordnet werden.

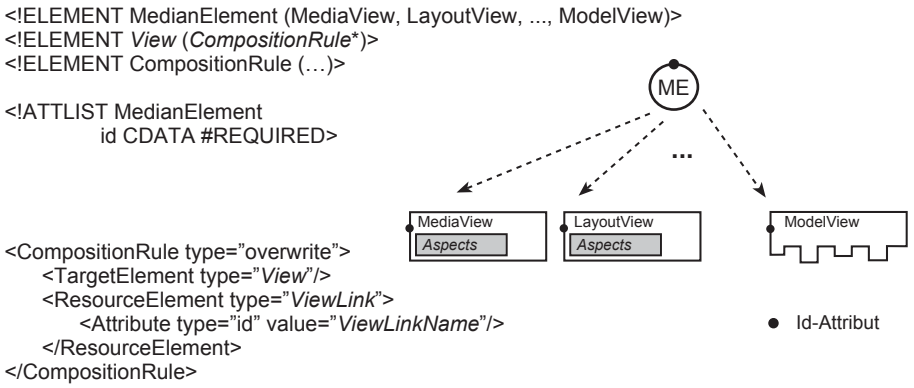


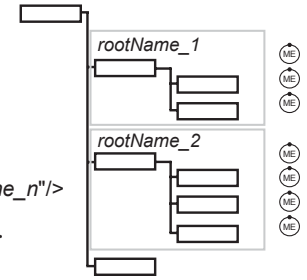
Abb. 5.3.3.2.1: DTD-Struktur eines Mittlerelements innerhalb des Präsentationsmodells.

Die Bindung der Sichten wird entsprechend der Abbildung auf geschlossene Präsentationsmedien durch Kompositionsregeln beschrieben. Allerdings werden die Regeln nicht direkt während der Abbildung auf eine verteilte Ablaufumgebung angewendet, sondern als ergänzende Definition den Mittlerelementen zugewiesen. Die Integration der Kompositionsregeln findet dabei durch vorgegebene Regeln statt, so dass die Transformation automatisch erfolgen kann.

Wie in Abb. 5.3.3.2.1 dargestellt, wird ein Mittlerelement nach der Transformation durch entsprechende Kompositionsregeln für die einzelnen Sichten (MediaView, LayoutView, …, ModelView) definiert. Alternative Bindungen einer Sicht werden durch mehrere Kompositionsregeln beschrieben, wobei jede Regel durch eine Priorität spezifiziert werden kann. Beispielsweise wird jedem Mittlerelement eine Kompositionsregel für die Bindung eines so genannten internen Präsentationsmediums mit der geringsten Priorität zugewiesen, so dass die von dem Mittlerelement verwalteten digitalen Mediendaten innerhalb einer verteilten Umgebung wiedergegeben werden können. Durch die einzelnen Kompositionsregeln werden die Sichten bzw. Sichtaspekte mithilfe ihrer Identifikationsattribute (id) referenziert bzw. die für die Geräte- und Modellbeschreibungen spezifischen virtuellen Schnittstellenbeschreibungen verwendet. Um dabei innerhalb einer verteilten Ablaufumgebung unterschiedliche Sichten dynamisch während der Laufzeit integrieren zu können, sind die Regeln als überschreibende Komposition (overwrite) definiert.

In einem zweiten Transformationsschritt werden die Mittlerelemente den lokalen Ablaufumgebungen zugewiesen. Hierbei dienen die komplexen Layoutsichten des Entwurfs als Vorlage. Wie bereits in Abschnitt 5.2.2 beschrieben, lassen sich die Layoutbeschreibungen als Projektionsflächen der Mediendaten interpretieren, die innerhalb einer verteilten Ablaufumgebung separate Projektionen in der realen Umgebung beschreiben können. So kann z.B. eine dreidimensionale virtuelle Umgebung, die durch einzelne elementare und komplexe

Layoutdefinitionen eine verteilte Projektion beschreibt, während der Abbildung den realen Projektionsstellen bzw. lokalen Ablaufumgebungen zugeordnet werden.




```
<TranslationRule priority="0">
    <Element type="LayoutView">
        <Attribute name="id" value="rootName_n"/>
        <Any/>
        <TargetElement type="LayoutView"/>
    </Element>
    <Specification>
        name := getTag("MedianElementId").getAttribute("name")
        write("<MedianElementId=", name,"/>")
    </Specification>
</TranslationRule>
```

Abb. 5.3.3.2.3: Zuordnung der Mittlerelemente zu lokalen Ablaufumgebungen.

Um eine Abbildung zu definieren, werden somit einzelne Layoutsichten eines zwei- bzw. dreidimensionalen Entwurfs als zentrale Knoten angesehen, die als separate lokale Ablaufumgebungen interpretiert werden. Wie in Abb. 5.3.3.2.3 dargestellt, wird die Abbildung durch eine Transformationsregel beschrieben, die auf alle elementaren Layoutsichten (TargetElement), die als Wurzelelement (Element) den zentralen Knoten (*rootName_n*) besitzen, angewendet wird. Die Präsentationsdefinition einer verteilten Anwendung setzt sich somit aus einzelnen lokalen Beschreibungen zusammen, wobei die Mittlerelemente die Sichten innerhalb einer lokalen Ablaufumgebung, wie im nachfolgenden Kapitel beschrieben, mithilfe ihrer Kompositionsregeln anfordern.

Kapitel 6

Offenes Präsentationsmedium

Wie in Kapitel 2 beschrieben, wird ein Medium durch unterschiedliche Aspekte definiert. Neben den in Kapitel 5 eingeführten Rollen und Protokollen der interdisziplinär zusammengesetzten Entwicklergruppe wird in diesem Kapitel ein neues Präsentationsmedium für multimediale Mittlerelemente vorgestellt. Mittlerelemente können zwar mithilfe des subjektorientierten Entwurfs auf unterschiedliche Präsentationsmedien abgebildet werden, jedoch stellen die zurzeit gängigen Ablaufumgebungen, wie in Kapitel 4 gezeigt, keine Grundlage für eine offene Modellierung verteilter multimedialer Anwendungen zur Verfügung, so dass sie nicht als Präsentationsmedium für die Gestaltung von Wahrnehmungsumgebungen geeignet sind.

Die hier vorgestellte Softwarearchitektur führt die Trennung zwischen den Rollen bzw. Sichten und der Organisation mithilfe der multimedialen Mittlerelemente fort. Im Gegensatz zu dem subjektorientierten Entwurf, bei dem die Mittlerelemente als „passive“ Elemente des mentalen Modells die einzelnen Sichten miteinander verbinden, treten sie in der Ablaufumgebung als aktive eigenständige Software-Elemente auf. Die Modellierung der multimedialen Mittlerelemente erfolgt dabei durch mobile Agenten, die den eigentlichen multimedialen „Inhalt“ der Sichten mit unterschiedlichen Präsentationsmedien verbinden. Die Sichten und Präsentationsmedien werden in der Softwareumgebung durch Komponenten repräsentiert, die die Einbettung bereits bestehender interner Präsentations- und Interaktionsformen ermöglichen. Um die Charakteristika multimedialer Umgebungen in einer geeigneten Form zu unterstützen, sind die Komponenten als übersetzende Verbindungen realisiert, so dass externe Präsentationsmedien für die Wiedergabe der Mediendaten und für die Integration in die verteilte Ablaufumgebung eingebunden werden können.

Somit wird zum einen die homogene Einbindung externer Geräte unterstützt, zum anderen werden geeignete Präsentationsformen in virtuellen und realen Umgebungen durch die Realisierung als verteilte Umgebung ermöglicht. Die Mittlerelemente bzw. mobilen Agenten können zwischen den einzelnen lokalen Ablaufumgebungen migrieren, wobei sie die Wiedergabe und Interaktion der lokalen Ablaufumgebung anpassen. Die Softwarearchitektur für multimediale Mittlerelemente erfüllt damit die in Abschnitt 4.1.2 geforderte vertikal und horizontal offene Modellierung multimedialer Umgebungen und dient somit als technische Basis für den Entwurf von Wahrnehmungsumgebungen.

6.1 Verteilte Softwareumgebung

Wie bereits in Kapitel 4 gezeigt, kann eine Softwareumgebung als Präsentationsmedium unter verschiedenen Aspekten betrachtet werden. Zunächst ist hier sicherlich die Softwarearchitektur zu berücksichtigen, durch die das Entwurfskonzept der Ablaufumgebung beschrieben wird.

Aber gerade durch die Verbindung der komponenten- und agentenorientierten Modellierung spielt neben dieser statischen Sicht die dynamische Kopplung zwischen den einzelnen Software-Elementen eine zentrale Rolle, wobei hier die offene Modellierung der Ablaufumgebung als Präsentationsmedium für verteilte multimediale Umgebungen betont wird.

6.1.1 Softwarearchitektur

Für den Aufbau einer Softwarearchitektur sind, wie bereits in Abschnitt 4.1.1 gezeigt, unterschiedliche Abstraktionsgrade zu betrachten. Die hier vorgestellte Architektur abstrahiert von den tiefer liegenden Schichten der verschiedenen Plattformen, Softwarebusse und Agentensysteme, indem die einzelnen Dienste durch eine Zwischenschicht gekapselt werden. Somit kann die Entwicklung des Softwaresystems in den einzelnen Schichten der Architektur erfolgen, so dass unterschiedliche Realisierungen der Schichten möglich sind und insbesondere von unteren Schichten abstrahiert werden kann.

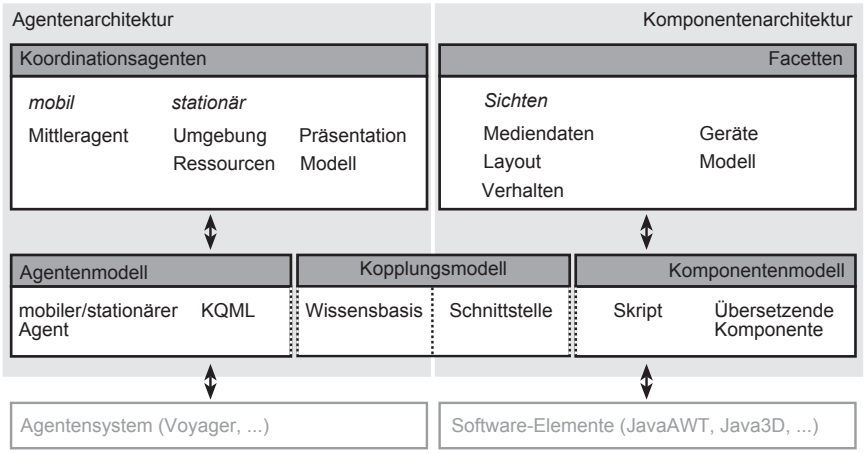


Abb. 6.1.1.1: Bausteine der Softwarearchitektur.

Wie in Abb. 6.1.1.1 dargestellt, setzt sich die Gesamtarchitektur aus einer Komponenten- und Agentenarchitektur zusammen, die jeweils unterschiedliche Abstraktionsgrade unterstützen:

Komponentenmodell: Das Komponentenmodell baut auf übersetzenden Komponenten auf, so dass bereits bestehende Software-Elemente in die Architektur eingebunden werden können. Gerade bei der Integration interner Präsentationsmedien kann somit auf bekannte Klassen-Bibliotheken zurückgegriffen werden. Um dabei die Entwicklung von neuen Facetten in einheitlicher Weise unterstützen zu können, bietet das Komponentenmodell eine Skriptsprache an, die die Einbindung von Software-Elementen unterstützt.

Die Realisierung der Sichten multimedialer Mittlerelemente durch Komponenten erfordert es, insbesondere die Facetten während der Laufzeit miteinander kombinieren zu können, so dass das Komponentenmodell eine Schnittstellenbeschreibung zur Verfügung stellt.

Kopplungsmodell: Durch das Kopplungsmodell werden die Komponenten- und Agentenarchitektur miteinander verbunden. Das Kopplungsmodell stellt dafür die einheitliche Verwaltung der Schnittstellen des Komponentenmodells und der Wissensbasis zur Verfügung, indem zum einen eine gemeinsame Sprache angeboten wird, und zum anderen gleiche Zugriffsmechanismen und Verwaltungsmechanismen durch das Modell vorgegeben werden. Hierdurch können die Schnittstellen als Inhalt der Wissensbasis integriert werden.

Agentenmodell: Das Agentenmodell abstrahiert von den grundlegenden Mechanismen eines gegebenen Agentensystems. Damit wird erreicht, dass die Architektur unabhängig von dem zugrundeliegenden Agentensystem entwickelt werden kann. Wie bereits in Kapitel 4 gezeigt, spielen die Kommunikation und Kopplung zwischen einzelnen Software-Elementen eine wesentliche Rolle für den Aufbau eines offenen Systems. Das Agentenmodell baut auf allgemeinen Diensten auf, so dass für eine nähere Beschreibung auf Kapitel 4 verwiesen wird.

Die Agentenverwaltung bietet die grundlegenden Mechanismen der in Abschnitt 4.3.2 beschriebenen Dienste für die Kommunikation und Migration an. Durch die orthogonale Migration eines Agenten wird der interne Zustand festgehalten und der gesamte Programmcode einschließlich des Zustands zu einer anderen Ablaufumgebung transferiert.

Für die ortsunabhängige Verwaltung der Elemente wird durch diese Schicht ein Namens- und Verzeichnisdienst angeboten. Um dabei die Verwaltung unabhängig von der eigentlichen Realisierung zu gestalten, werden die Dienste durch Agenten gekapselt, so dass die Kommunikation mithilfe einer standardisierten Sprache erfolgen kann. Die Wissensbasis, die durch die Kommunikation abgefragt und beeinflusst werden kann, stellt die Agentenseite der Kopplung zwischen beiden Architekturen dar.

Die Kommunikation zwischen den Agenten wird durch die Kommunikationssprache KQML in einer einheitlichen syntaktischen und semantischen Form unterstützt. Wie bereits in Abschnitt 4.3.2.2 gezeigt, ist somit auch hier die Trennung zwischen Kommunikationsstruktur und eigentlichem Inhalt der Kommunikation gewährleistet.

Dienen die bis hierher beschriebenen Schichten als Grundlage der eigentlichen verteilten Ablaufumgebung, wird durch die Facetten und durch die spezialisierten Koordinationsagenten die Funktionalität der Ablaufumgebung bestimmt:

Facetten: Die unter der Bezeichnung Facetten zusammengefassten Elemente der Architektur repräsentieren die Sichten (Mediendaten, Layout, Verhalten), Präsentationsmedien und Modellbeschreibungen, die auf dem Komponentenmodell aufbauen. Somit wird durch die Facetten die eigentliche multimediale Funktionalität beschrieben.

Dabei werden die internen Daten der Facetten, wie etwa die einzelnen Medientypen oder externen Präsentationsmedien, durch spezialisierte Facetten mithilfe der durch das Kopplungsmodell vorgegebenen Schnittstellenbeschreibung gekapselt.

Koordinationsagenten: Die Organisationsstruktur wird durch die Koordinationsagenten realisiert. Wie bereits einführend erwähnt, werden dabei durch mobile Agenten die mul-

timedialen Mittlerelemente repräsentiert. Die mobilen Agenten treten dabei als aktive Elemente auf, die vergleichbar mit den Mittlerelementen die einzelnen Sichten bzw. Facetten miteinander verbinden, so dass sie im Folgenden vereinfachend als Mittleragenten bezeichnet werden.

Um die Funktionalität für die Facetten zur Verfügung zu stellen, wird die verteilte Umgebung durch stationäre Agenten aufgebaut, die spezialisierte Dienste für die unterschiedlichen Facettentypen realisieren. Da die Facetten als übersetzende Komponenten modelliert werden, können somit bereits bestehende Dienste durch so genannte Service-Agenten zur Verfügung gestellt werden.

Das hier vorgestellte Architekturkonzept wurde in dieser Arbeit durch das *Java Media Tool (JMT)* realisiert. Aufbauend auf dem Agentensystem *Voyager* [Voya01], das bereits grundlegende Mechanismen zur Realisierung eines verteilten Multiagenten-Systems zur Verfügung stellt, werden durch das Java Media Tool die höher liegenden Schichten der Architektur angeboten. Für eine nähere Betrachtung der Realisierung sei auf [BeSy00b] [Delf99] [SBSZ99] [Otto98] verwiesen.

6.1.2 Offene Modellierung der Ablaufumgebung

Neben der statischen Sicht auf die Architektur kann die Ablaufumgebung als Koordinationsmodell aufgefasst werden, durch das die unterstützten Möglichkeiten der Modellierung mithilfe der Facetten und Agenten veranschaulicht werden. Die Modellierung bzw. Erstellung einer multimedialen Umgebung basiert dabei auf dem in Abschnitt 4.2.1 bereits beschriebenen „Baukastensystem“, aus dem unterschiedliche verteilte Ablaufumgebungen zusammengestellt werden können.

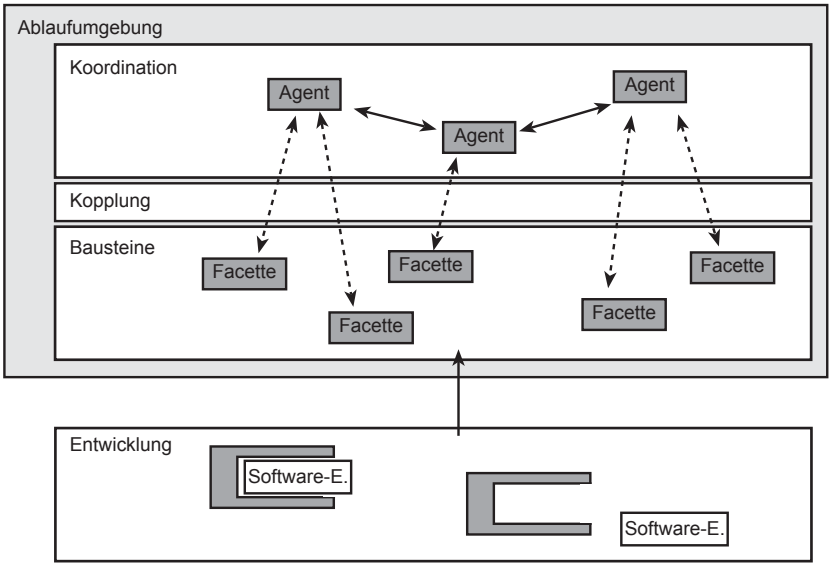


Abb. 6.1.2.1: Modellierung einer verteilten Ablaufumgebung.

Wie in Abb. 6.1.2.1 dargestellt, lassen sich in der hier vorgestellten Softwareumgebung drei Ebenen unterscheiden, die die logische Struktur für die Erstellung bzw. Erweiterung der Softwareumgebung als Präsentationsmedium charakterisieren.

Facetten-Entwicklung: Auch wenn durch die hier vorgestellte Softwareumgebung bereits die wichtigsten multimedialen Aspekte durch einzelne Facetten realisiert sind, erfordert gerade die Einbindung externer Präsentationsmedien die (vertikal) offene Modellierung. Um die Modellierung zu unterstützen, wird dabei durch das Komponentenmodell ein Rahmen für die Implementierung vorgegeben.

Im Gegensatz zu dem „traditionellen" Entwurf einzelner Software-Elemente wird hierdurch bereits eine Basis für die Implementierung angeboten, so dass die Ergänzung keine Veränderungen der oberen Ebenen nach sich zieht. Dies wird insbesondere durch die separate Realisierung multimedialer Aspekte erreicht, so dass die einzelnen Aspekte, vergleichbar mit dem subjektorientierten Entwurf, getrennt entwickelt werden können.

Handelt es sich bei der Facetten-Entwicklung um eine Implementierung bzw. Erweiterung des Systems, wird durch die beiden oberen Ebenen die eigentliche Ablaufumgebung für die Gestaltung von Wahrnehmungsumgebungen zur Verfügung gestellt.

Facetten-Bausteine: Die Facetten dieser Ebene bieten bereits vorgefertigte Bausteine, die typische Funktionalitäten multimedialer Anwendungen realisieren. Auch hier spielt die Trennung der Sichten eine wesentliche Rolle für die Wiederverwendbarkeit einzelner Facetten, da sie den Sichten entsprechend frei zu kombinieren sind.

Somit kann im Gegensatz zu den in Abschnitt 4.3.3 vorgestellten anwendungsorientierten Agentensystemen z.B. durch spezielle Modellfacetten eine Simulation unabhängig entwickelt werden, wobei die Interpretation der Simulationsdaten durch „gewöhnliche" Mediendaten-, Layout-, und Verhaltensfacetten umgesetzt wird.

Koordination: Wie bereits in Abschnitt 5.3.3 gezeigt, wird die Definition einer Anwendung als Präsentationssprache repräsentiert, durch die die Verbindung der Sichten mit (internen, externen) Präsentationsmedien und eventuellen Modellen mithilfe von Regeln beschrieben wird. Die Mittleragenten übernehmen die Koordination innerhalb der verteilten Umgebung und „präsentieren" den multimedialen Inhalt. Die horizontale Modellierung einer Ablaufumgebung findet dabei durch die stationären Agenten statt, die die Struktur der multimedialen Umgebung widerspiegeln.

Somit wird die Komposition einzelner „Bausteine" als zusammenhängende Anwendung, die in Komponentensystemen durch Skriptsprachen unterstützt wird, während der Laufzeit durch die Agenten übernommen. Durch die lose Kopplung von Agenten, die durch eine abstrakte Kommunikationsform unterstützt wird, können die Facetten als Inhaltsbeschreibungen unabhängig von der Koordinations- bzw. Kommunikationsstruktur integriert werden.

Durch die Kopplung des komponentenorientierten mit dem agentenorientierten Ansatz wird die in Abschnitt 4.1.2 geforderte vertikal und horizontal offene Modellierung einer Softwareumgebung für die Gestaltung von Wahrnehmungsumgebungen erreicht. Dabei steht durch die Kombination beider Modellierungsformen eine Architektur zur Verfügung, die bereits ein Baukastensystem als technische Basis für die Umsetzung verteilter multimedialer Anwendungen zur Verfügung stellt, ohne sich dabei auf eng umrissene Anwendungsgebiete zu beschränken.

Im Gegensatz zu den in Abschnitt 4.2.3 vorgestellten anwendungsbezogenen Komponentensystemen, werden bei dem hier eingeführten Ansatz mithilfe der Facetten ausschließlich separate multimediale Teilfunktionen realisiert, die noch keine weitere Struktur vorgeben. Erst durch das Agentensystem werden die Facetten, d.h. die Teilfunktionen, miteinander verbunden und somit das Gesamterscheinungsbild und die Gesamtfunktionalität der multimedialen Anwendung erzeugt. Verglichen mit den in Abschnitt 4.3.3 vorgestellten Agentensystemen liegt somit eine klare Trennung zwischen multimedialer Funktionalität, die durch die Facetten realisiert wird, und organisierender Struktur, die durch die Agenten aufgebaut wird, vor.

Um die zurzeit von der Architektur unterstützten Modellierungsmöglichkeiten aufzuzeigen, wird in Abschnitt 6.2 zunächst die Verbindung zwischen Komponenten- und Agentenmodell vorgestellt. In Abschnitt 6.3 bzw. Abschnitt 6.4 wird die Komponenten- bzw. Agentenarchitektur näher betrachtet, wobei zur Veranschaulichung auf die Realisierung des Java Media Tools zurückgegriffen wird. In Kapitel 7 werden anhand von Beispielen mögliche Konfigurationen verteilter multimedialer Anwendungen dargestellt.

6.2 Kopplungsmodell

Wie im vorherigen Abschnitt gezeigt, werden die Organisationsstruktur und die multimediale Funktionalität durch eine einheitliche Beschreibung der Wissensbasis des Agentenmodells und der Schnittstelle des Komponentenmodells miteinander gekoppelt. Somit dient das Kopplungsmodell, vergleichbar mit den Sichten des subjektorientierten Entwurfs, als Verbindungsglied zwischen beiden Architekturen.

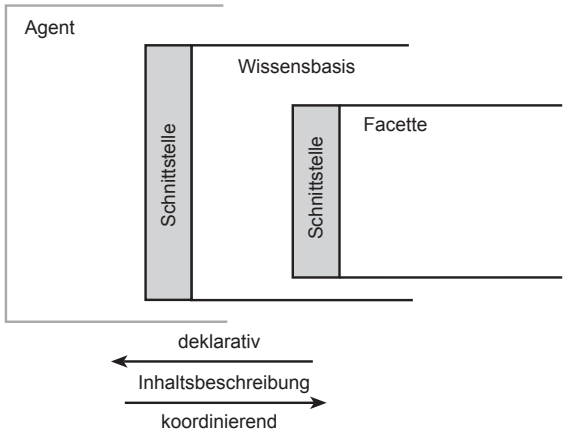


Abb. 6.2.1: Funktionale Verbindung zwischen Organisationsstruktur und multimedialer Funktionalität.

Im Gegensatz zu dem Entwurf werden hier die Sichten allerdings durch ihre zugrundeliegende Funktionalität beschrieben, so dass, abweichend von den rein deklarativen Schnittstellenbeschreibungen von Komponentenmodellen, zum einen der Zugriff als deklarative, zum anderen als koordinierende Beschreibung ermöglicht wird. Wie in Abb. 6.2.1 dargestellt, können die Agenten somit die Schnittstelle über die Wissensbasis abfragen und mithilfe der Schnittstellen die Koordination zwischen den einzelnen Facetten aufbauen. Dabei bleibt den Agen-

ten die eigentliche Implementierung der Facetten und der eventuell entfernte Zugriff verborgen. Die Funktionalität der Facetten wird durch Skripte und Software-Elemente beschrieben, wobei die Facettenschnittstelle die realisierte Funktionalität kapselt.

```
<!ELEMENT ContentName (ContentName* | Any | ValueName)>
<!ELEMENT Any EMPTY>
<!ELEMENT ValueName (VALUE)>

<!ATTLIST ContentName
                    type    (“set” | “get”)
                    id      CDATA>
<!ATTLIST ValueName
                    type    (“set” | “get” | “=” | “<” | “<=” | “>” | “>=”)>
```

Abb. 6.2.2: DTD-Struktur der Inhaltsbeschreibung.

Um den Inhalt der Wissensbasis und die Beschreibung der Schnittstelle in gleicher Weise behandeln zu können, basiert die Darstellung auf einer einheitlichen Sprache, so dass hier des Weiteren vereinfachend von dem Inhalt gesprochen wird.

Die Syntax der Inhaltsbeschreibung ist durch die in Abb. 6.2.2 dargestellte XML-Struktur vorgegeben, die die für XML typische Bildung von Namen/Wert-Paaren erlaubt. Die Inhaltsbeschreibung unterstützt somit die Darstellung der einzelnen Parameterbeschreibungen der Sichten, deren Funktionalität durch die Facetten beschrieben wird. Für die Repräsentation mehrerer Facetten innerhalb der Wissensbasis wird durch die Bildung von hierarchischen Beschreibungen ebenso die strukturierte Repräsentation unterschiedlicher Facetten ermöglicht. Eine hierarchische Inhaltsbeschreibung wird dabei durch eine Baumstruktur aufgebaut, in der jedes Namen/Wert-Paar wiederum eine hierarchische Inhaltsbeschreibung repräsentieren kann. Da die Namen hierdurch kontextabhängig verwaltet werden, kann der Name eines Namen/Wert-Paars innerhalb einer Inhaltsbeschreibung mehrmals verwendet werden.

Anfrage

```
<Content_1 id="name">
    <Any>
        <Value_1 type="<">
            6
        </Value_1>
    <Any>
</Content_1>
```

Inhaltsbeschreibung

```
<Content_1 id="name">
    <Content_2>
        <Value_1>
            5
        </Value_1>
        <Value_2>
            ...
    </Content_2>
    <Content_3>
        ...
        <Content_2>
            <Value_1>
                4
            <Value_1>
        </Content_2>
        ...
    </Content_3>
    <Content_5>
        <Value_1>
            7
        </Value_1>
    </Content_5>
</Content_1>
```

Abb. 6.2.3: Unvollständig spezifizierter Pfad einer Inhaltsbeschreibung.

Neben der Repräsentation des Inhalts sind durch die Wissensbasis und die Facettenschnittstelle unterschiedliche Zugriffsmechanismen realisiert. Der Grundmechanismus beruht dabei auf der Beschreibung des Pfads eines Namen/Wert-Paars. Dabei wird ein Wert durch die Angabe des vollständigen Pfads oder lediglich durch die Angabe eines Teilpfads spezifiziert. Teilpfade setzen sich aus Namen und Platzhaltern (Any) zusammen, wobei letztere als beliebige Bezeichner interpretiert werden, so dass Teilbäume der Inhaltsbeschreibung, wie in Abb. 6.2.3 dargestellt, bestimmt werden können. Durch die Sprache lassen sich sowohl schreibende (set) Zugriffe als auch lesende Zugriffe mithilfe unterschiedlicher Bedingungen (get, =, <, ...) definieren:

Schreiben: Das Erzeugen einer Inhaltsbeschreibung baut auf der vollständigen Pfadbeschreibung eines Namen/Wert-Paars auf, d.h. ein Wert, der wiederum eine Inhaltsbeschreibung darstellen kann, wird an einer eindeutig spezifizierten Stelle in die hierarchische Struktur eingefügt. Dieser Mechanismus kann ebenso bei der Erweiterung und Manipulation einer Inhaltsbeschreibung angewendet werden, indem neue Namen/Wert-Paare in die Struktur eingefügt werden bzw. bereits bestehende Daten überschrieben werden.

Um Daten auslesen zu können, stehen zwei Methoden zur Verfügung. Zum einen kann durch die Angabe des jeweiligen Pfads direkt auf die Daten zugegriffen werden, zum anderen können Daten „beobachtet“ werden.

Anfragen: Durch die Angabe des Pfads bzw. eines nicht vollständig spezifizierten Pfads kann ein Agent auf eine spezielle Inhaltskomponente zugreifen bzw. alle durch den Teilpfad spezifizierten Daten in Abhängigkeit der angegebenen Bedingung (get, =, <, ...) erhalten. Wie in Abb. 6.2.3 dargestellt, kann dabei durch die vergleichenden Bedingungen der gesuchte Inhalt näher spezifiziert werden. Um die eindeutige Zuordnung einer Anfrage zu ermöglichen, wird der Inhalt bzw. werden die Inhalte mit ihrem zugehörigen vollständigen Pfad zurückgeliefert. Durch diesen Mechanismus wird neben dem flexiblen Zugriff eine Abstraktion von der eigentlichen Struktur des Inhalts unterstützt.

Beobachten: Das Beobachten von Daten ermöglicht die Reaktion auf Änderungen einzelner Inhaltskomponenten und wird somit für die Kopplung der Facetten verwendet, die über die Schnittstelle erfolgt. Hierzu lassen sich durch die Angabe des vollständigen Pfads die zu beobachtenden Daten eines Namen/Wert-Paars bestimmen. Bei Änderungen der Daten werden die entsprechenden Ereignisse ausgelöst, wobei der alte und neue Wert übergeben werden.

Wie in Abschnitt 6.3.1.2 bzw. Abschnitt 6.4.3.3 noch näher gezeigt, wird die beschriebene Funktionalität der Komponentenschnittstellen bzw. der Wissensbasen durch das *Document Object Model (DOM)* [Brad98] realisiert, das bereits eine standardisierte Schnittstelle für die Manipulation einer hierarchischen XML-Struktur anbietet.

6.3 Komponentenarchitektur

Durch die Komponentenarchitektur wird ein Komponentenmodell vorgegeben, das einen Rahmen für die Entwicklung vorgibt, auf dem die eigentliche Realisierung der Facetten aufbaut. Das Komponentenmodell stellt für die Entwicklung einzelne Software-Elemente zur Verfügung, die je nach zu realisierender Funktionalität miteinander kombiniert werden können.

Die „multimediale Funktionalität“ wird durch bereits realisierte Komponenten, den so genannten Facetten, zur Verfügung gestellt, die sowohl die funktionale Umsetzung der Sichten (Mediendaten, Layout, Verhalten) als auch die Anbindung externer Präsentationsmedien und die Anbindung verschiedener Modelle unterstützen. Durch die Eigenständigkeit und Abgrenzbarkeit der verschiedenen Facettentypen wird somit durch die Komponentenarchitektur bereits eine Vielfalt möglicher Modellierungen angeboten.

6.3.1 Komponentenmodell

6.3.1.1 Übersetzende Komponenten

Durch das Komponentenmodell werden übersetzende Komponenten als Schablonen für die Entwicklung neuer Facetten angeboten. Hierdurch wird erreicht, dass aufbauend auf dem Komponentenmodell einerseits die Abgrenzbarkeit der Facetten, andererseits die Integrations- und Anpassungsfähigkeit neu entwickelter Facetten unterstützt wird.

Die Abgrenzbarkeit und Integrationsfähigkeit von Komponenten wird durch die in Abschnitt 6.2 bereits beschriebene Schnittstellen- bzw. Inhaltsbeschreibung erreicht, die zum einen unabhängig von der Programmiersprache ist, zum anderen Implementierungsdetails verdeckt. Aus dieser Sicht lassen sich die Komponenten, wie in Abb. 6.3.1.1.1 dargestellt, als Behälter für implementierungsabhängige Software-Elemente, die die eigentliche Funktionalität zur Verfügung stellen, interpretieren. Aus Sicht der Agenten stellt eine Facette somit eine durch die Inhaltsbeschreibung definierte Schnittstelle zur Verfügung, über die sie getrennt von anderen Facetten angesprochen werden kann. Aus Sicht der externen Systeme stellt eine Facette die spezifischen Funktionen zur Verfügung.

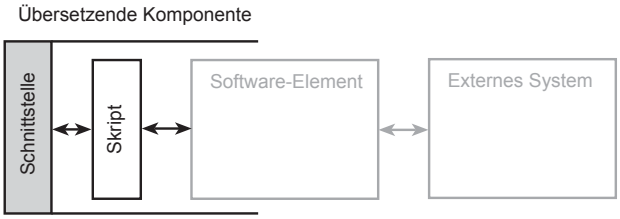


Abb. 6.3.1.1.1: Aufbau übersetzender Komponenten.

Die Integrationsfähigkeit der Facetten erfordert zusätzlich die Entkopplung, so dass die Kommunikation auf semantischen Ereignissen beruht. Die Interpretation der Ereignisse findet in Abhängigkeit der eigentlichen Funktionalität innerhalb der Facette mithilfe der Skriptsprache statt.

Die Skriptsprache unterstützt dabei die Anpassung bzw. funktionale Erweiterung einzelner Facetten, wie dies etwa bei der angesprochenen Integration neuer externer Präsentationsmedien und der Anbindung eines Modells als virtuelle Umgebung der Fall ist. Wie bereits in Abschnitt 5.3.3 gezeigt, kann dabei die funktionale Erweiterung der Facetten während der Abbildung auf die verteilte Umgebung stattfinden oder erst während des Aufbaus der Anwendung durch die Entwicklung neuer Facetten erfolgen.

6.3.1.2 Schnittstelle

Durch die Schnittstelle des Komponentenmodells wird, wie bereits in Abschnitt 6.2 beschrieben, die Kopplung zwischen den einzelnen Facetten und den Agenten der Ablaufumgebung realisiert. Um dabei innerhalb der Wissensbasis eines Agenten integriert werden zu können, bietet die Schnittstelle die Methoden eines *Element-Objekts* des *Document Object Models (DOM)* [Brad98] an.

Ein Element-Objekt von DOM ist mit den Element-Sprachkonstrukten von XML vergleichbar, so dass die Schnittstelle einer Komponente die Struktur einer XML-Beschreibung widerspiegelt. Wie in Abb. 6.3.1.2.1 anhand einer Sicht-Facette dargestellt, bilden die einzelnen Aspekte einer Sicht die funktionale Schnittstellenbeschreibung einer Komponente. Die Parameter werden dabei als Eingangs- (set) und/oder Ausgangskanäle (get) gekennzeichnet, die mit den von den *JavaBeans* bekannten *setter-* bzw. *getter-Methoden* zu vergleichen sind [Mons00]. Somit können die einzelnen Parameterwerte über eingehende Ereignisse beeinflusst bzw. durch beobachtende Komponenten „abgefragt“ werden.

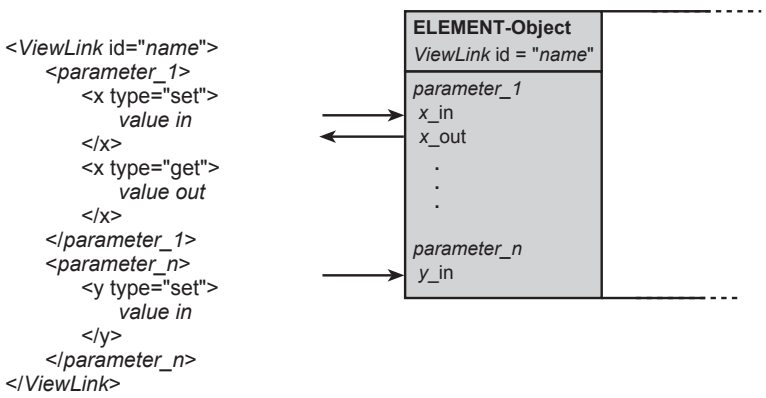


Abb. 6.3.1.2.1: Aufbau der Komponentenschnittstellen.

Neben dieser funktionalen Beschreibung der Schnittstelle, kann diese als Element-Objekt wiederum als einzelnes Element-Sprachkonstrukt interpretiert werden, so dass die Schnittstelle, wie in Abschnitt 6.4.1.3 gezeigt, innerhalb einer Hierarchie der Wissensbasis integriert werden kann.

6.3.1.3 Skriptsprache

Werden die Skriptsprachen von Komponentenmodellen für die Kopplung bereits vorliegender Komponenten eingesetzt, so dient hier die Skriptsprache zur Einbindung bereits bestehender Software-Elemente in das Komponentenmodell. Um dabei eine flexible Definition zu ermöglichen, unterstützt die Skriptsprache alle gängigen Datentypen und Anweisungen einer imperativen Sprache.

Neben dieser Grundfunktionalität lassen sich Klassen in Skripte integrieren, wodurch ein zu integrierendes Software-Element direkt angesprochen bzw. die Funktionalität durch vorgefertigte Klassenbibliotheken erweitert werden kann. Wie bereits in Abschnitt 6.1.2 erwähnt, wird durch die hier vorgestellte Skriptsprache die Übersetzung in ein Programm unterstützt, so dass der Übergang von einer neu erstellten Komponente zu einem Facetten-Baustein der

Ablaufumgebung keine Neuimplementierung erfordert. Die Skriptsprache, deren Sprachelemente in Abb. 6.3.1.2.1 zusammengefasst sind, ist der Syntax von Java [FlFC99] bzw. JavaScript [Koch99] entlehnt.

```
Program     =  {Import} {VarDecls} {FuncDecls}
Import      =  import ident {´.´ ident}´;´´
VarDecls    =  var VarDecl {´,´´ VarDecl} ´;´´
VarDecl     =  ident [ ´´=´´ Expr]
FuncDecls   =  [public | protected] function ident ´´(´´ [FormalPars] ´´)´´ Block
FormalPars  =  ident {´,´´ ident}
Block       =  Stat | ´´{´´ {Stat} ´´}´´
Stat        =  VarDecls
               | Expr ´´;´´
               | if ´´(´´ ExprList ´´;´´ Expr ´´;´´ ExprList ´´)´´ Block
               | while ´´(´´ Expr ´´)´´ Block
               | do Block while ´´(´´ Expr ´´)´´
               | for ´´(´´ ExprList ´´;´´ Expr ´´;´´ExprList ´´)´´ Block
               | foreach ´´(´´ ident in Expr ´´)´´ Block
               | break ´´;´´
               | continue ´´;´´
               | return [Expr] ´´;´´
               | ´´;´´
ExprList    =  Expr { ´´,´´ Expr}
Expr        =  CondExpr {AssignOp CondExpr}
CondExpr    =  OrExpr [´´?´´ CondExpr ´´:´´ CondExpr]
OrExpr      =  AndExpr { ´´||´´ AndExpr}
AndExpr     =  RelExpr { ´´&&´´ RelExpr}
RelExpr     =  SimpleExpr [Relation SimpleExpr]
SimpleExpr  =  [´´+´´ | ´´-´´] Term {AddOp Term}
Term        =  Factor {MulOp Factor}
Factor      =  [IncDecOp] Designator [IncDecOp]
               | new ident {´.´´ ident} ´´(´´ [ExprList] ´´)´´
               | new ´´[´´ Expr ´´]´´ {´´[´´ Expr ´´]´´}
               | null | true | false
               | number | string | char | ´´(´´ Expr ´´)´´| ´´!´´ Factor
Designator  =  ident {´´[´´Expr´´]´´ | ´´.´´ident | ´´(´´ [ExprList] ´´)´´ }
AssignOp    =  ´´=´´ | ´´*=´´ | ´´/=´´ | ´´%=´´ | ´´+=´´ | ´´-=´´
Relation    =  ´´==´´ | ´´!=´´ | ´´<´´ | ´´<=´´ | ´´>´´ | ´´=>´´
MulOp       =  ´´*´´ | ´´/´´ | ´´%´´
AddOp       =  ´´+´´ | ´´-´´
IncDecOp    =  ´´++´´ | ´´--´´
ident       =  letter {letter | digit | ´´_´´ } | ´´‘´´ {escape | character} ´´‘´´
string      =  ´´“´´ {escape | character} ´´“´´
char        =  ´´‘´´(escape | character) ´´‘´´
escape      =  ´´\´´(´´b´´ | ´´t´´ | ´´n´´ | ´´f´´ | ´´r´´ | ´´“´´ | ´´‘´´ | ´´\´´ )
number      =  digit {digit} [´´.´´{digit} [scale] [´´f´´ | ´´F´´ ]]
scale       =  (´´ e ´´ | ´´ E ´´) [´´+´´| ´´-´´ ] digit {digit}
```

Abb. 6.3.1.2.1: Syntaxbeschreibung der Skriptsprache in erweiterter Backus-Naur Notation.

Namensraumdefinition: Um die Skriptsprache durch bereits existierende Funktionen zu erweitern, können am Anfang eines Skripts Klassen in den Namensraum des Skripts eingebunden werden.

Variablendefinition: Die deklarierten Variablen sind global für ein Skript gültig, wobei eine direkte Initialisierung der Variablen unterstützt wird. Somit lassen sich durch globale Variablen von anderen Facetten zu beobachtende Zustände einführen.

Funktionsdefinition: Funktionen werden global definiert. Funktionen können eine beliebige Anzahl von Übergabe-Parametern spezifizieren und einen Ergebniswert zurückliefern. Der Körper einer Funktion besteht aus einer Anweisung oder einer Folge von

Anweisungen. Durch die Definition der Funktionen wird das Verhalten der Facetten bestimmt. So können Funktionen die nach außen sichtbare Schnittstelle erweitern (public) bzw. die interne Funktionalität (protected) einer Facette ergänzen.

Anweisungen: Die Anweisungstypen der Skriptsprache umfassen die in imperativen Sprachen üblichen Anweisungen:

Variablendefinition: Die innerhalb einer Funktion deklarierten Variablen werden als lokale Variablen der umgebenden Funktion interpretiert.

Ausdruck: Ein Ausdruck liefert einen Wert, dessen Berechnung Seiteneffekte hervorrufen kann. So kann etwa durch eine Zuweisung der Wert einer Variablen berechnet werden. Die verschiedenen Varianten von Ausdrücken werden im Anschluss noch näher betrachtet.

if (*cond*) *if-block* else *else-block*: Die Anweisungsfolge *if-block* wird genau dann ausgeführt, wenn der Ausdruck *cond* den boolschen Wert true ergibt, ansonsten wird der *else-block* abgearbeitet.

while (*cond*) *block*: Die Anweisungsfolge *block* wird solange ausgeführt, solange der Ausdruck *cond* den boolschen Wert true ergibt.

do *block* while (*cond*): Diese Anweisung verhält sich analog zu while. Die Schleifenbedingung *cond* wird jedoch am Ende jedes Schleifendurchlaufs geprüft, so dass die Anweisungsfolge *block* mindestens einmal ausgeführt wird.

for (*init-exprList*; *cond*; *loop-exprList*) *block*: Zunächst wird die Anweisungsfolge *init-exprList* ausgeführt. Der eigentliche Schleifendurchlauf beginnt mit dem Testen der Bedingung *cond*. Falls das Ergebnis den boolschen Wert false annimmt, wird die Schleife beendet. Ansonsten wird der Schleifenrumpf *block* ausgeführt und im Anschluss der Ausdruck *loop-exprList* ausgewertet.

foreach (*indent* in *expr*) *block*: Diese Anweisung beschreibt eine Schleife, in der über Skriptobjekte iteriert werden kann. Dabei werden der Variablen *ident* nacheinander die im Skriptobjekt *expr* vorhandenen Schlüssel zugewiesen. Die Schleife wird beendet, wenn alle Schlüssel zugewiesen wurden.

break: Diese Anweisung führt zu einem Schleifenabbruch. Es wird unmittelbar mit der Ausführung der ersten Anweisung nach dem umgebenden Schleifenkonstrukt fortgefahren.

continue: Mit Hilfe dieser Anweisung kann die Ausführung eines Schleifenrumpfs beendet werden. Die Schleife wird mit ihrem nächsten Durchlauf fortgesetzt, d.h. bei der while- und do-Schleife wird die Schleifenbedingung *cond* überprüft, bei der for-Schleife mit der Ausführung der *loop-exprList* und bei der foreach-Schleife mit dem nächsten Schlüssel fortgefahren.

return *expr*: Diese Anweisung beendet die Ausführung einer Funktion, wobei optional ein Wert *expr* an die aufrufende Anweisung zurückgegeben werden kann.

Ausdrücke: Die Ausdrücke der Skriptsprache setzen sich aus mathematischen bzw. logischen Verknüpfungsoperatoren zusammen. Dabei wird der Typ eines Ausdrucks durch die Konstanten, Variablen und Funktionsaufrufe einer Verknüpfung bestimmt. Wie bereits weiter oben erwähnt, werden Variablen untypisiert deklariert, so dass der Typ einer Vari-

ablen mit dem Typ desjenigen Ausdrucks übereinstimmt, der ihr zuletzt zugewiesen wurde. Die Operatoren der Skriptsprache entsprechen in ihrer Syntax und Semantik den jeweiligen Java-Pendants [FlFC99][Pütt00].

Die mit function definierten Funktionen eines Skripts dienen zur Definition des Verhaltens von Facetten. Wie bereits weiter oben beschrieben, kann durch die Definition von Funktionen somit die externe Schnittstelle der Facetten ergänzt werden bzw. das interne Verhalten der gekapselten Software-Elemente angesprochen werden. Funktionen werden innerhalb eines Ausdrucks in der üblichen Form *name*(*param1*, *param2*, …) aufgerufen, wobei *param1* etc. an die formalen Parameter der Funktion *name* übergeben werden. Die Anzahl der Parameter muss mit der Anzahl der formalen Parameter der Funktion übereinstimmen. Der Typ einer Funktion ergibt sich aus dem Argument *expr* der ersten ausgeführten return-Anweisung dieser Funktion.

Um die Skriptsprache durch neue Datentypen erweitern zu können und bereits vorhandene Klassenbibliotheken benutzen zu können, lassen sich durch das Schlüsselwort new neue Objekte ebenso wie Reihungen in ein Skript integrieren. Ein Objekt wird durch Angabe seiner Klasse mit den entsprechenden Parametern des Konstruktors bzw. eine Reihung durch Angabe seiner Elementanzahl erzeugt. Mit den Konstrukten *object.field* und *array*[*index*] lassen sich die entsprechenden Felder eines Objekts bzw. Elemente der Reihung adressieren.

Neben der Integration von Klassen und Reihungen können Skripte bzw. Skriptfunktionen als sogenannte Skriptobjekte in ein Skript eingebunden werden. Hierbei dient die einzubindende Funktion quasi selbst als Konstruktor und wird dementsprechend durch new *name*(*param1*, *param2*, ...) als Skriptobjekt erzeugt. Ein Skriptobjekt wird als Schlüssel/Wert-Paar interpretiert, so dass der Ausdruck *object.key* den Wert zum Schlüssel *key* im Objekt *object* zurückliefert. Ebenfalls wird der Zugriff auf den entsprechenden Wert eines Skriptobjekts mittels *object*[*key*] unterstützt, wobei *key* als Zeichenkette vorliegen muss. Der Wert zu einem Schlüssel kann von einem beliebigen Typ sein.

Um die aus der objektorientierten Implementierung bekannte Vererbung innerhalb der Skriptsprache zu integrieren, können Skriptobjekte mit anderen Objekten verbunden werden, die selbst wiederum Skriptobjekte sein können. Hierzu dient das Sprachelement proto, das auf ein mit dem entsprechenden Skriptobjekt zu verbindendes Objekt verweist. Durch diese spezielle Form des Verweises stehen dem Skriptobjekt alle Schlüssel/Wert-Paare des referenzierten Objekts zur Verfügung, so dass das Skriptobjekt alle Felder des Objekts erbt und somit direkten Zugriff auf diese Felder besitzt.

6.3.2 Facetten

6.3.2.1 Sicht-Facetten

Für die Integration der Sichten des subjektorientierten Entwurfs in die Ablaufumgebung bieten die so genannten Sicht-Facetten die jeweilige Funktionalität für die in Abschnitt 5.2 beschriebenen Parameter der Aspekte an. Die einzelnen Aspekte sind über die Schnittstellen der Facetten sowohl zu beobachten als auch zu beeinflussen, so dass die in Abschnitt 2.2.2 beschriebenen Beziehungsaspekte als auslösende Ereignisse und als auszuführende Aktionen auftreten können. Entsprechend den unterschiedlichen Ausprägungen der Sichten werden durch die Architektur folgende Facettentypen angeboten:

Medienfacetten: Wie bereits in Abschnitt 5.3.3 bei der Abbildung in eine verteilte multimediale Umgebung gezeigt, werden die Mediendaten getrennt von der Wiedergabe verwaltet, um die homogene Einbindung externer Geräte zu unterstützen. Die Medienfacetten verwalten somit ausschließlich die internen bzw. externen Mediendaten und stellen eine einheitliche Schnittstelle für den Umgang mit den Daten zur Verfügung. Diese werden von den entsprechenden Gerätefacetten, die im nächsten Abschnitt beschrieben werden, abgespielt. Durch diese Trennung können Mediendaten in Abhängigkeit von der Darstellungsumgebung unterschiedlich wiedergegeben werden. So können Textdaten z.B. intern als formatierter Text dargestellt oder extern als Laufschrift bzw. als synthetisierte Sprache abgespielt werden.

Für die Integration digitaler Medien werden die zurzeit für multimediale Anwendungen typischen Datenformate zeitkonstanter Medien, wie *GIF*, *JPEG*, *TIFF* als Bildformate sowie *HTML* und *RTF* als Textformate unterstützt. Als Datenformate für zeitkontinuierliche Medien können *QuickTime* und *MPEG1* als Video- und Soundformate sowie *QuickTime VR* als 360°-Panoramaformat verwendet werden.

Layoutfacetten: Die Layoutfacetten übernehmen die Verwaltung der „Projektionsfläche" für die Mediendaten. Wie in Abschnitt 5.2.2 beschrieben, unterstützen die Layoutfacetten somit die Darstellung als zweidimensionale Flächen und dreidimensionale Objekte, so dass die Facetten Objekte der *JavaAWT* [FlFC99] bzw. der *Java3D* [Barr00] Klassenbibliothek kapseln, die die einzelnen Aspekte der Layoutbeschreibung funktional unterstützen. Um dabei ebenso zweidimensionale Flächen innerhalb einer dreidimensionalen Umgebung darstellen zu können, werden durch die 3D-Facetten spezielle 3D-Objekte realisiert, die nur in x- und y-Richtung eine Ausdehnung besitzen.

Verhaltensfacette: Wie bereits in Abschnitt 5.3.3 gezeigt, wird die Verhaltensbeschreibung direkt in die von der Architektur unterstützte Skriptsprache transformiert, so dass die Verhaltensfacette lediglich den Interpreter für die Skriptsprache bereitstellt. Die Verhaltensfacette wird erst während der Erzeugung durch die entsprechenden Skripte initialisiert.

6.3.2.2 Gerätefacetten

Die Gerätefacetten repräsentieren die internen und externen Präsentationsmedien innerhalb einer Ablaufumgebung, die im folgenden vereinfachend als interne bzw. externe Gerätefacetten bezeichnet werden. Um ein Präsentationsmedium dabei als Abspielgerät und/oder Eingabegerät verwalten zu können, setzt sich eine Gerätefacette aus drei Bestandteilen zusammen. Erstens ist dies die Steuerung der Wiedergabe, so dass die Mediendaten in Abhängigkeit ihres Typs und der vorhandenen technischen Möglichkeiten abgespielt werden können. Zweitens verwalten die Gerätefacetten eine „Projektionsfläche", die durch die im vorherigen Abschnitt beschriebenen Layoutfacetten realisiert wird und drittens wird das Registrieren von Interaktionsereignissen unterstützt.

Betrachtet man die Gerätefacetten als internes bzw. externes Abspielgerät, lassen sich die Gerätefacetten durch die Zeitmetrik der wiederzugebenden Mediendaten unterscheiden, wobei zwischen zeitunabhängigen und zeitabhängigen Mediendaten unterschieden wird. Innerhalb dieser Klassifikation wird weiterhin, wie in Abb. 6.3.2.2.1 dargestellt, zwischen internen und externen Gerätefacetten unterschieden.

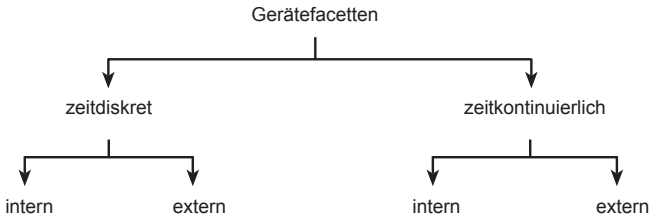


Abb. 6.3.2.2.1: Übersicht der Gerätefacetten.

Zeitunabhängige Mediendaten, wie etwa Texte oder Bilder, werden zu jedem Zeitpunkt der Wiedergabe durch die gleichen Daten repräsentiert. Zeitabhängige Mediendaten lassen sich als eine fest vorgegebene Folge von einzelnen zeitkonstanten Daten interpretieren, die durch eine Gerätefacette abgespielt werden kann. Dabei können zeitabhängige Mediendaten durch eine diskrete bzw. kontinuierliche Zeitmetrik definiert sein. Zeitdiskrete Mediendaten legen ausschließlich eine Reihenfolge der abzuspielenden Daten fest. Diese können durch die Definition einer absoluten bzw. relativen Position direkt von der Gerätefacette angesprochen werden. Im Gegensatz zu zeitdiskreten besitzen zeitkontinuierliche Mediendaten eine Zeitbasis, durch die eine vorgegebene Abspielgeschwindigkeit definiert ist.

Durch die Interpretation zeitabhängiger Mediendaten als Folge zeitkonstanter Daten können letztere als Spezialfall zeitdiskreter Mediendaten aufgefasst werden, die ausschließlich aus einer einelementigen Folge bestehen. Um zeitdiskreten Mediendaten eine Abspielgeschwindigkeit zuordnen zu können, realisieren die Gerätefacetten eine interne Uhr, so dass die Facetten eine einheitliche Zeitbasis zur Verfügung stellen, die vom Medientyp unabhängig ist.

```
<Device id="name">
    <MediaControl>
        <MediaSpeed>
            ...
        </MediaSpeed>
        <MediaTime>
            ...
        </MediaTime>
    </MediaControl>
    <ProjectionControl>
        ...
    </ProjectionControl>
    <InteractionControl>
        <MouseDown type="get">
            ...
        </MouseDown>
        ...
    </InteractionControl>
</Device>
```

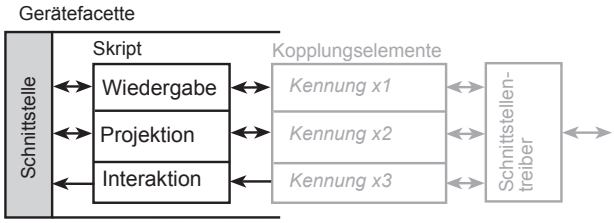


Abb. 6.3.2.2.2: Schnittstellenbeschreibung und Aufbau externer Gerätefacetten.

Zur Steuerung der Mediendaten wird durch die Gerätefacetten die Beeinflussung der internen Uhr angeboten. So kann, wie in Abb. 6.3.2.2.2 anhand der Schnittstellenbeschreibung dargestellt, die Abspielgeschwindigkeit (MediaSpeed) und der Abspielzeitpunkt (MediaTime) beeinflusst werden. Neben der Steuerung wird durch die Schnittstelle, wie bereits weiter

oben erwähnt, die Verwaltung der Projektionsfläche (Projection) und Beobachtung von Interaktionsereignissen (MouseDown, MouseUp, ...) angeboten.

Um die angebotene Funktionalität zu realisieren, basieren die internen Gerätefacetten auf dem *Java QuickTime Player* [MaSt99], der bereits die Steuerung unterschiedlicher Medienformate unterstützt. Dabei werden durch den Java QuickTime Player die Layoutbeschreibungen direkt verwaltet. Die externen Gerätefacetten sind als beobachtende Facetten realisiert, die auf Änderungen der Medien- und Layoutfacette reagieren. Die Parameter der Facetten werden, wie bereits in Abschnitt 5.3.3 beschrieben, durch separate Skripte entsprechend des externen Präsentationsmediums umgesetzt, so dass auch hier bereits vorgefertigte Skripte miteinander kombiniert werden können.

Die Ansteuerung externer Ausgabegeräte bzw. der Empfang extern ausgelöster Ereignisse wird durch einen Schnittstellentreiber unterstützt, mit dem die Gerätefacetten über Kopplungselemente verbunden sind. Wie in Abb. 6.3.2.2.2 dargestellt, wird das verwendete Protokoll der externen Schnittstelle durch den Treiber gekapselt, so dass verschiedene Schnittstellen integriert werden können, ohne die Gerätefacetten ändern zu müssen. Für die Identifikation externer Präsentationsmedien findet die Kommunikation zwischen Treiber und externem Gerät mithilfe einer Kennung statt, so dass über die gleiche Schnittstelle unterschiedliche Präsentationsmedien angesprochen werden können. Zurzeit wird dabei durch das Java Media Tool die Kommunikation über den seriellen und den parallelen Port unterstützt, an die Mikrokontroller für die Identifizierung und Ansteuerung externer Geräte und Sensoren angeschlossen werden können [Delf99].

6.3.2.3 Modellfacetten

Die Modellfacetten repräsentieren die Mittleragenten in der durch einen Datenraum definierten virtuellen Umgebung. Mithilfe der Modellfacetten können die Mittleragenten das zugrundeliegende Modell der virtuellen Umgebung beeinflussen und interpretieren. Somit stellen sie in Analogie zu den Gerätefacetten, die die Schnittstellen zur realen Umgebung repräsentieren, die Ein- und Ausgabeschnittstellen zur virtuellen Umgebung zur Verfügung.

Steuert der Rezipient in der realen Umgebung die Anwendung, können die Modellfacetten als steuernde Elemente der virtuellen Umgebung angesehen werden. Somit wird durch die Modellfacetten, im Gegensatz zu den in Kapitel 4 vorgestellten anwendungsorientierten Komponenten- und Agentensystemen, eine Trennung zwischen der „multimedialen Interpretation“ und den steuernden Modellen unterstützt. Wie in Abschnitt 4.3.3 gezeigt, wird gerade bei den dort vorgestellten Agentensystemen durch das zugrundeliegende Modell bereits das Anwendungsgebiet stark eingeschränkt.

Um die Funktionalität verschiedener Ausprägungen der Modellfacetten einheitlich modellieren zu können, werden die Zugriffe auf die Daten eines zugrundeliegenden Modells als Datenbankzugriffe interpretiert, so dass die Modellfacetten als Sichten auf eine virtuelle Datenbank fungieren. Die Übertragung der Modelldaten an die Mittleragenten wird dabei durch einen stationären Service-Agenten verwaltet, der die Facetten zyklisch aktiviert, um die Operationen der Modellfacetten anzustoßen. Damit in einer verteilten Umgebung Übertragungsengpässe zwischen den Modellfacetten und den Mittleragenten vermieden werden, können somit zum einen ausschließlich abstrakte Aktionen durch die Dateninterpretation innerhalb der Facetten weitergeleitet werden, zum anderen kann durch die Definition des Zeitintervalls der Abstand aufeinanderfolgender Aktualisierungen festgelegt werden.

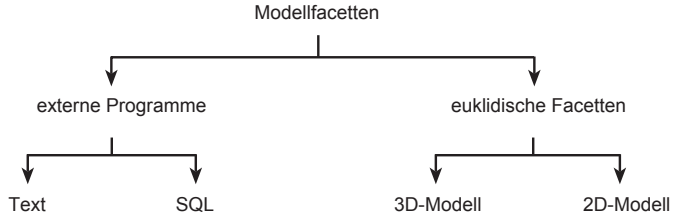


Abb. 6.3.2.3.1: Übersicht der Modellfacetten.

Durch das Java Media Tool werden bereits unterschiedliche Modellfacetten angeboten, um häufig verwendete Modelle zu unterstützen. Die Filterfacette bietet die Grundfunktionalität für das Lesen und Schreiben von textbasierten Schnittstellen an. Derartige Schnittstellen werden häufig durch Simulationsprogramme zur Verfügung gestellt, um Aktionen des Simulationsprogramms anzustoßen bzw. Daten der Simulation auszulesen. Die Datenbankfacette ist durch die *JDBC-API* [FlFC99] realisiert, über die externe Datenbanken mithilfe der Datenbanksprache *SQL2* angesprochen werden können. Beide Modellfacetten bieten somit eine Schnittstelle für extern vorliegende Modelle an, die ausschließlich die Kommunikation zwischen den Mittleragenten und den Modellen unterstützen. Die Anpassung an die Modelldaten erfolgt wie bei den externen Gerätefacetten durch ergänzende Skriptdefinitionen.

```
<Model id="name">
    <Position>
        ...
    </Position>
    <Scale>
        …
    </Scale>
    <Offset type="set">
        <Position>
            ...
        </Position>
        <Scale>
            …
        </Scale>
    </Offset>
    <Event>
        Migrate, MouseUp MouseDown, …
    </Event>
</Model>
```

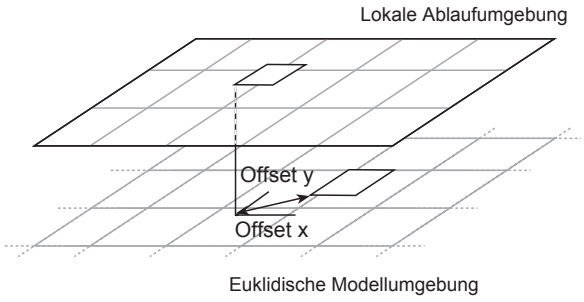


Abb. 6.3.2.3.2: Beziehung zwischen lokaler Ablaufumgebung und euklidischem Modell.

Der dritten Modellfacette liegt bereits ein Datenmodell zugrunde, das die Repräsentation der Mittleragenten in einem zwei- bzw. dreidimensionalen euklidischen Raum unterstützt. Durch dieses Modell wird den Mittleragenten eine einheitliche Sicht auf die gesamte verteilte multimediale Umgebung zur Verfügung gestellt. Die lokalen Ablaufumgebungen stellen dabei Teilbereiche des euklidischen Raums dar.

Durch das euklidische Modell werden die Mittleragenten, wie in Abb. 6.3.2.3.2 dargestellt, durch ihre Geometrie und Position in der virtuellen Umgebung repräsentiert. Dabei werden die unterschiedlichen Koordinatensysteme der lokalen Ablaufumgebung und der Modellumgebung durch die Verschiebung und Skalierung der Modellfacetten angeglichen.

Jede Modellfacette in der virtuellen Umgebung ist durch einen eindeutigen Namen bestimmt, über den die Mittleragenten in der gesamten verteilten Umgebung die virtuellen Objekte eindeutig identifizieren können. Hierdurch kann z.B. die Realisierung von Mehrbenutzersystemen unterstützt werden, in denen jeder Benutzer durch einen Avatar repräsentiert ist. Dienen diese Parameter als von dem Mittleragenten zu steuernde Parameter, kann die Modellfacette auch als steuerndes Element auftreten, indem sie Ereignisse der virtuellen Umgebung an den Mittleragenten weiterleitet. Somit kann die virtuelle Umgebung als zusammenhängendes Interaktionsmodell fungieren, das die entsprechenden Interaktionsereignisse an die lokalen Ablaufumgebungen weitergibt.

6.4 Agentenarchitektur

Die Agentenarchitektur ist durch das Kopplungsmodell mit der Komponentenarchitektur verbunden. Die Agenten haben durch die Modellierung der Wissensbasis einen einheitlichen Zugriff auf die Facetten, die den Zustand eines Agenten widerspiegeln.

Die Komponentenarchitektur stellt die einzelnen Sichten auf ein Mittlerelement zur Verfügung und realisiert somit die multimediale Funktionalität einer Anwendung. Die Mittleragenten verbinden die unterschiedlichen Facettentypen (Medien-, Layout-, Geräte-, Verhaltens-, Modellfacette) und nutzen die Dienste der so genannten Service-Agenten. Jeder Service-Agent ist dabei spezialisiert auf die charakteristischen Eigenschaften eines Facettentyps.

6.4.1 Agentenmodell

6.4.1.1 Stationäre und mobile Agenten

Die Agenten der Architektur sind modular aufgebaut, so dass die stationären und mobilen Agenten die gleiche Struktur aufweisen. Vergleichbar mit dem Komponentenmodell dient somit das Agentenmodell als Basis für die Entwicklung der unterschiedlichen Ausprägungen der koordinierenden Agenten.

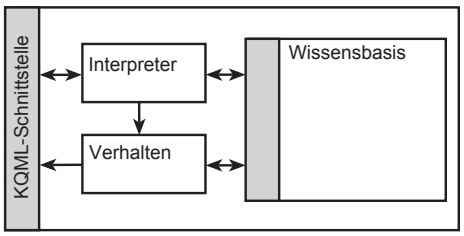


Abb. 6.4.1.1.1: Aufbau stationärer und mobiler Agenten.

In Abbildung 6.4.1.1.1 ist der grundsätzliche Aufbau der Agenten dargestellt. Jeder Agent setzt sich aus einer KQML-Schnittstelle, einem Interpreter, einer Verhaltensdefinition und einer Wissensbasis zusammen.

Wie bereits in Abschnitt 4.3.2.2 näher beschrieben, werden durch die Kommunikationssprache *Knowledge Query and Manipulation Language (KQML)* [MaLF96] [FiLM97] standardisierte Schlüsselwörter vorgegeben, durch die so genannte Sprechakte für einfache Informationsvermittlungen, Anfragen und Antworten etc. gebildet werden können. KQML

gibt dabei ausschließlich einen Rahmen vor, so dass der Inhalt in einer für den Anwendungskontext geeigneten Sprache definiert werden kann.

Um die Sprechakte entsprechend weiterverarbeiten zu können, werden durch den Interpreter die Bestandteile eines KQML-Ausdrucks analysiert und festgelegte Funktionen ausgelöst, die direkt auf die Wissensbasis zugreifen können oder das Verhalten eines Agenten beeinflussen. Durch den Interpreter werden somit die unterstützten KQML-Schlüsselworte festgelegt, wobei die Agenten, wie bereits in Abschnitt 4.3.2.2 erwähnt, unterschiedliche „Wortschätze“ anbieten können.

Durch die Verhaltensbeschreibung eines Agenten werden die individuellen Eigenschaften der stationären und mobilen Agenten definiert, wobei durch das Verhalten mobiler Agenten bereits die Grundfunktionalität für den Transfer vorgegeben wird. Der Aufbau der Wissensbasis richtet sich nach dem Kopplungsmodell der Architektur und repräsentiert, wie in Abschnitt 6.4.1.3 noch gezeigt wird, den individuellen Zustand eines Agenten.

6.4.1.2 KQML-Schnittstelle

Um das Empfangen und Senden von KQML-Sprechakten zu unterstützen, setzt sich die KQML-Schnittstelle der Agenten, wie in Abb. 6.4.1.2.1 dargestellt, aus einer Kommunikationsschnittstelle, einem Nachrichtenverwalter und einem Nachrichtenkonstruktor zusammen.

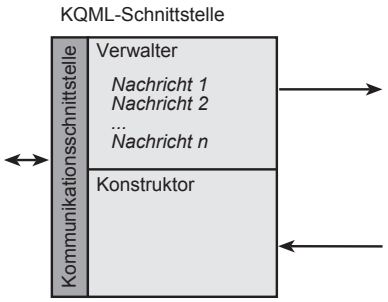


Abb. 6.4.1.2.1: Aufbau der KQML-Schnittstellen.

Durch die Kommunikationsschnittstelle wird die Verbindung mit der Kommunikationsinfrastruktur unterstützt. Hierzu hört die Schnittstelle den Kommunikationskanal ab und leitet Nachrichten mit der entsprechenden Empfängerkennung (receiver) an den Nachrichtenverwalter weiter.

Der Nachrichtenverwalter speichert ankommende Nachrichten, so dass ein Agent asynchrone Nachrichten zu unterschiedlichen Zeitpunkten bearbeiten kann. Die Bearbeitung findet dabei durch den bereits im vorherigen Abschnitt beschriebenen Interpreter statt. Das Versenden von KQML-Nachrichten wird durch den Konstruktor unterstützt, der die einzelnen Teile einer Nachricht (KQML-Schlüsselwort, Empfänger, Inhalt, ...) in die vorgegebene Syntax überführt.

Da sowohl die Kommunikationsschnittstelle als auch der Verwalter und Konstruktor ausschließlich die Interpretation der KQML-Parameter übernehmen, ist die KQML-Schnittstelle unabhängig von dem eigentlichen Inhalt der Nachricht realisiert.

6.4.1.3 Wissensbasis

Die Realisierung der Wissensbasis eines Agenten basiert auf dem *Document Object Model (DOM)*, so dass die Komponenten- bzw. Facettenschnittstellen innerhalb der Wissensbasis verwaltet werden können. Die Strukturdefinition einer Wissensbasis findet dabei entsprechend durch die Inhaltsbeschreibung des Kopplungsmodells statt. Wie bereits in Abschnitt 5.3.1 gezeigt, stehen für DOM standardisierte Methoden für die Manipulation des Modells zur Verfügung.

Die Wissensbasis eines Agenten repräsentiert unterschiedliche Bereiche, die mit den in Abschnitt 4.3.1 eingeführten mentalen Aspekten von Agenten vergleichbar sind. Wie in Abb. 6.4.1.3.1 dargestellt, werden Informationen über die Struktur der Wissensbasis und die angebotenen Dienste eines Agenten bereitgestellt. Diese Informationen, die das „Können" eines Agenten beschreiben, können von anderen Agenten abgefragt werden, so dass diese z.B. Informationen über die unterstützten Facettentypen eines stationären Agenten erhalten.

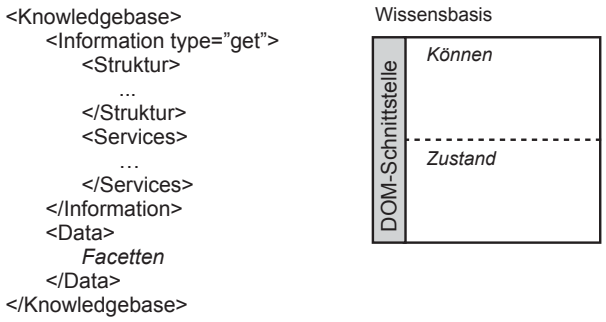


Abb. 6.4.1.3.1: Strukturbeschreibung der Wissensbasen.

Der Datenbereich einer Wissensbasis repräsentiert den Zustand bzw. das „Wissen" eines Agenten. Hier werden die durch den Agenten verwalteten Facetten gespeichert, wobei die Struktur des Datenbereichs die logischen Beziehungen zwischen den Facetten widerspiegelt.

Die Strukturbeschreibung einer Wissensbasis wird während der Initialisierung der Agenten geladen, so dass die Struktur der einzelnen Agenten unterschiedlichen Realisierungen der Architektur leicht angepasst werden kann. Die Strukturbeschreibungen liegen dabei als DTD in XML vor.

6.4.2 Koordinierende Agenten

6.4.2.1 Service-Agenten

Die Struktur einer verteilten Ablaufumgebung wird durch die so genannten Service-Agenten aufgebaut, die für die mobilen Mittleragenten bzw. unterschiedlichen Facettentypen spezialisierte Dienste anbieten.

Wie in Abb. 6.4.2.1.1 dargestellt, bieten die Service-Agenten die Schnittstellen zwischen externer Präsentationsebene und interner Präsentationsebene, zwischen den einzelnen lokalen Ablaufumgebungen bzw. zwischen interner Präsentationsebene und Modellebene an. Um zunächst einen Überblick über die angebotenen Dienste der einzelnen Service-Agenten zu vermitteln, werden hier kurz die unterschiedlichen Ausprägungen umrissen.

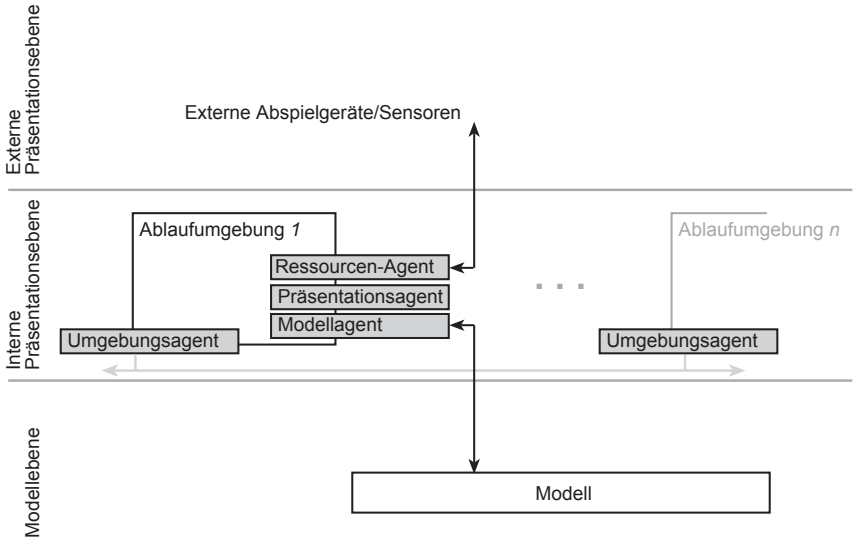


Abb. 6.4.2.1.1: Aufbau einer Ablaufumgebung.

Allerdings wird erst durch die Koordination der einzelnen Dienste die eigentliche Dynamik der Ablaufumgebung beschrieben. Dabei beruht die Koordination auf KQML-Sprechakten zwischen den einzelnen Service-Agenten und Mittleragenten, so dass sie im Anschluss an die funktionale Beschreibung von Mittleragenten in Abschnitt 6.4.2.3 betrachtet wird.

Umgebungsagent: Die Umgebungsagenten sind die zentralen Anlaufstellen für die mobilen Mittleragenten innerhalb einer verteilten Anwendung. Ein Umgebungsagent bietet dabei die Dienste für die Verwaltung und Kontrolle einer lokalen Ablaufumgebung an, die aus den unterschiedlichen Service-Agenten und den mobilen Mittleragenten gebildet wird.

Um als zentrale Anlaufstelle zu fungieren, repräsentiert der Umgebungsagent die von seiner Umgebung angebotenen Dienste, die von den mobilen Mittleragenten abgefragt werden können. Hierzu melden sich alle Service-Agenten bei dem Umgebungsagenten an, der daraufhin die Initialisierung übernimmt.

Neben dieser lokalen Sicht stellt ein Umgebungsagent die Verbindung zu anderen Ablaufumgebungen zur Verfügung, die ihrerseits ebenfalls durch Umgebungsagenten repräsentiert werden. Zur Unterstützung der Mobilität der Mittleragenten bieten die Umgebungsagenten einen orthogonalen Migrationsdienst an, so dass die Mittleragenten innerhalb der verteilten Umgebung verschiedene Ablaufumgebungen aufsuchen können.

Vor der Ausführung der Migration eines Mittleragenten wird dieser aus der Verwaltungsstruktur ausgetragen und es werden Informationen von der Zielumgebung durch den Umgebungsagenten angefordert. Der Mittleragent wird mit den verwalteten Facetten serialisiert und an die Zielumgebung übertragen. Schlägt die Übertragung des Agenten fehl, ist es Aufgabe des Umgebungsagenten, den Agenten in der eigenen Umgebung erneut zu starten. Nach einer erfolgreichen Migration wird der Mittleragent durch den Umgebungsagenten der Zielumgebung deserialisiert. Nach der Migration meldet sich der Mittleragent bei der Zielumgebung an und fragt die durch die Umgebung angebotenen Dienste über den Umgebungsagenten ab.

Ressourcen-Agent: Der Ressourcen-Agent einer lokalen Ablaufumgebung leistet eine Abstraktion zwischen Dateisystem, externen Ressourcen und den Mittleragenten. Die Mittleragenten nehmen die Dienste des Ressourcen-Agenten in Anspruch, um die in der Definition beschriebenen Facetten, wie z.B. die der internen bzw. externen Mediendaten und Interaktionselemente, zu laden. Dabei bleiben die Speicherposition und die Datenformate für die Mittleragenten verdeckt, so dass sie eine homogene (komponentenbasierte) Sicht auf die Ressourcen besitzen. Der Ressourcen-Agent stellt dabei den Zugriff auf ein verteiltes Dateisystem zur Verfügung, das auf dem in Abschnitt 4.2.2.1 näher betrachteten Verzeichnisdienst aufbaut. Somit können die Mittleragenten in einer Ablaufumgebung auf lokal und entfernt liegende Daten in gleicher Weise zugreifen.

Neben dem Zugriff auf digitale Daten unterstützt der Ressourcen-Agent die Einbindung externer Ressourcen, wie beispielsweise das Einbinden von analogen Medien, Sensoren oder externen Programmen. Die Einbindung findet dabei über die Gerätetreiber der externen Ressourcen statt, die von dem Ressourcen-Agenten beobachtet werden, so dass externe Ressourcen auch während der Laufzeit eingebunden werden können.

Die Bindung der Ressourcen und Sichten bzw. Facetten findet dynamisch während der Laufzeit statt und basiert auf den während der Realisierungsphase erzeugten Regeln, die von den Mittleragenten verwaltet werden. Der Ressourcen-Agent interpretiert die Regeln und stellt die entsprechenden Facetten zur Verfügung. Um dabei externe Ressourcen in gleicher Weise interpretieren zu können, werden diese durch ihre funktionale Schnittstellenbeschreibung und ihren logischen Namen repräsentiert.

Präsentationsagent: Der Präsentationsagent verwaltet eine interne graphische Oberfläche, in der die Layoutfacetten als graphische zweidimensionale bzw. dreidimensionale Komponenten platziert werden. Da die Layoutfacetten als Projektionsflächen der Mediendaten dienen, wird somit durch den Präsentationsagenten die Platzierung und Wiedergabe der Mediendaten realisiert.

Die Verwaltungsstruktur des Präsentationsagenten spiegelt dabei die Darstellungsbeziehung zwischen den Layoutfacetten wider. Durch die Beziehungen werden, wie in Abschnitt 5.2.2 bereits näher gezeigt, die Darstellungsreihenfolge der Facetten und das zugrundeliegende Bezugssystem, das durch die übergeordnete Layoutfacette vorgegeben wird, für die Platzierung und Ausrichtung beschrieben.

Modellagent: Der Modellagent bietet Dienste für die Verwaltung der Modellfacetten an, um die Verbindung der Mittleragenten mit der virtuellen Umgebung zu unterstützen. Wie bereits in Abschnitt 6.3.2.3 gezeigt, kann die virtuelle Umgebung durch unterschiedliche Modelle definiert sein. Mithilfe der Modellfacetten beobachten bzw. verändern die Mittleragenten einen Ausschnitt des abstrakten Datenraums, so dass dieser als virtuelle Datenbank interpretiert wird.

Der Modellagent realisiert somit die Schnittstelle zu einer virtuellen Datenbank und stellt entsprechende datenbankspezifische Operationen für das Einfügen, Löschen und Verändern von Werten des Datenraums zur Verfügung. Um dabei die Operationen auf der virtuellen Datenbank zu synchronisieren, aktiviert der Modellagent die verwalteten Facetten zyklisch. Treten innerhalb eines Zyklus Änderungen in der Datenbank auf, werden diese durch die angemeldeten Modellfacetten interpretiert und an die zugehörigen Mittleragenten weitergeleitet.

Um die Sicht auf den Datenraum einzuschränken, kann durch einen Modellagenten ausschließlich ein Teilbereich der virtuellen Umgebung sichtbar gemacht werden, so dass die lokalen Ablaufumgebungen einer verteilten multimedialen Anwendung auf unterschiedlichen Bereichen der Daten definiert sind. Die Definition eines (Teil-)Bereichs legt dabei die Beziehung zwischen der Präsentation und dem Datenraum des Modells fest. So lassen sich der darzustellende (Teil-)Bereich des Modells, die weiterzuleitenden Interaktionsereignisse und die Sichtbarkeit der lokalen Ablaufumgebungen auf die in Abschnitt 6.3.2.2 bereits erwähnten euklidischen Datenmodelle beschreiben.

Durch die Definition von Sichtbereichen auf einen euklidischen Raum können die mobilen Agenten als frei bewegliche Elemente der verteilten Präsentationsumgebung fungieren, wobei die Zuordnung durch die Sichten festgelegt wird. Die Migration wird, wie im nachfolgenden Abschnitt noch näher beschrieben, während des Verlassens des sichtbaren Bereichs der lokalen Ablaufumgebung angestoßen, wobei die Zielumgebung durch den Eintritt in einen neuen Sichtbereich ermittelt wird.

Die Interaktionsereignisse einer Ablaufumgebung können an andere lokale Umgebungen weitergeleitet werden, die dort in gleicher Weise wie lokale Interaktionsereignisse interpretiert werden. Hierbei dienen die Sichtbereiche der Modellagenten als „Sender“ bzw. „Empfänger“ der Ereignisse, wobei die Ereignisse innerhalb überlappender Sichtbereiche weitergeleitet werden. Hierdurch wird z.B. die zentrale Steuerung einer verteilten multimedialen Umgebung unterstützt. Wie in Abschnitt 7.2 anhand eines Beispiels noch näher beschrieben, kann somit eine durch mehrere Ablaufumgebungen verteilte Präsentation über eine homogene Interaktionsschnittstelle gesteuert werden.

Um sowohl die Migration als auch das Weiterleiten von Interaktionsereignissen einschränken zu können, lässt sich die Sichtbarkeit einer lokalen Ablaufumgebung gegenüber anderen Ablaufumgebungen definieren. So kann durch den Modellagenten festgelegt werden, ob die Ablaufumgebung von anderen Mittleragenten aufgesucht werden kann, bzw. ob Interaktionsereignisse an andere Umgebungen weitergeleitet bzw. von anderen Umgebungen empfangen werden sollen.

6.4.2.2 Mittleragent

Die multimedialen Mittlerelemente werden in der hier vorgestellten Architektur, wie bereits in Abschnitt 6.1 gezeigt, durch mobile Agenten realisiert. Diese so genannten Mittleragenten übernehmen als autonome, koordinierende Elemente der Ablaufumgebung die funktionale Kopplung der verschiedenen Facettentypen, die das Erscheinungsbild, die Interaktionsschnittstelle, die wiederzugebenden Mediendaten und das Verhalten beschreiben. Sie dienen somit als Mittler zwischen virtueller und realer Umgebung.

Der „multimediale Inhalt“ der Mittleragenten wird dabei durch einen Regelsatz bestimmt, der von den Mittleragenten als „Information“ innerhalb einer verteilten Ablaufumgebung mitgeführt wird. Der Regelsatz wird durch den Ressourcen-Agenten einer lokalen Ablaufumgebung interpretiert, so dass die Mittleragenten innerhalb der verteilten Umgebung unterschiedliche, der lokalen Ablaufumgebung angepasste Erscheinungsformen besitzen können. Der Datenbereich der Wissensbasis eines Mittleragenten setzt sich dementsprechend aus den Regeldefinitionen für die unterschiedlichen Facetten und den durch die Anwendung der Regeln bestimmten Facetten zusammen, wobei die funktionale Kopplung der einzelnen Facetten durch den Mittleragenten stattfindet.

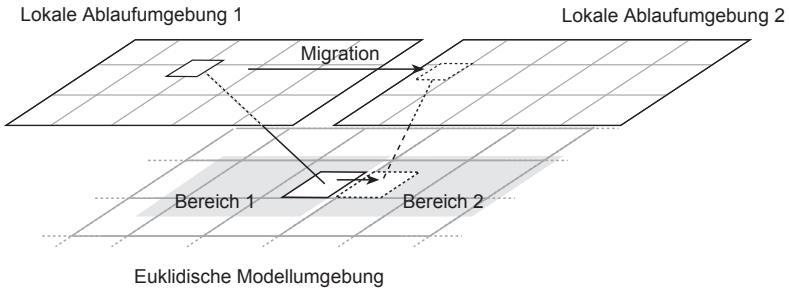


Abb. 6.4.2.2.1: Automatische Migration eines Mittleragenten.

Die Zuordnung eines Mittleragenten zu einer lokalen Ablaufumgebung kann, wie bereits in Abschnitt 6.4.2.1 erwähnt, mithilfe euklidischer Modellfacetten erfolgen. Die Modellfacetten repräsentieren einen Mittleragenten in der virtuellen Umgebung durch seine Position und Geometrie, wobei die lokalen Ablaufumgebungen, wie in Abb. 6.4.2.2.1 dargestellt, einen Teilbereich des virtuellen Raums abbilden. Verlässt ein Mittleragent bzw. seine zugewiesene Modellfacette den Teilbereich seiner Ablaufumgebung, wird der Agent durch die Modellfacette zur Migration angestoßen. Daraufhin meldet sich der Agent in seiner Ablaufumgebung ab und fordert den Umgebungsagenten auf, ihn zu der Zielumgebung zu transferieren. Nach der Migration wird der Mittleragent „gestartet", woraufhin er sich in der neuen Ablaufumgebung anmeldet. Durch die Interpretation der mitgeführten Regeln wird dort sein neues Erscheinungsbild ermittelt.

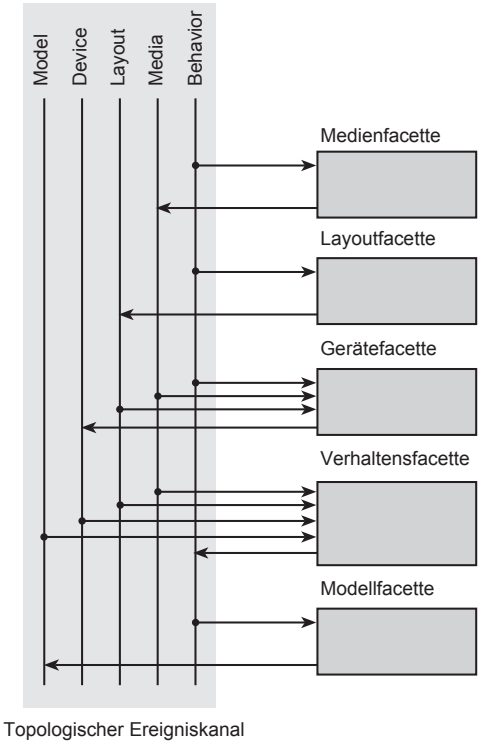


Abb. 6.4.2.2.2: Interne Kopplung der von einem Mittleragenten verwalteten Facetten.

Wie bereits weiter oben erwähnt, findet die funktionale Kopplung der von einem Mittleragenten verwalteten Facetten während der Initialisierung statt. Durch die Kopplung werden dabei die funktionalen Beziehungen zwischen den Facetten bestimmt, durch die letztendlich die multimediale Funktionalität realisiert wird. Bei der Kopplung der Facetten lassen sich zwei Kopplungsarten unterscheiden. Zum einen verbindet ein Mittleragent die von ihm verwalteten Facetten, zum anderen wird durch die Koordination zwischen den einzelnen Mittleragenten die Kopplung der Verhaltensfacetten innerhalb einer Ablaufumgebung gewährleistet.

Die interne Kommunikationsstruktur eines Mittleragenten baut auf der Schnittstellenbeschreibung der Facetten auf, die durch Eingangs- und/oder Ausgangskanäle, wie in Abschnitt 6.3.1.2 gezeigt, definiert ist. Die Kanäle werden entsprechend ihrer Definition als empfangende bzw. sendende Knoten an einen internen topologischen Ereigniskanal gekoppelt. Wie in Abschnitt 4.2.2.2 beschrieben, können durch einen topologischen Ereigniskanal Gruppen definiert werden, die hier aus den Kennungen der Facettentypen gebildet werden. Wie in Abb. 6.4.2.2.2 dargestellt, wird durch den topologischen Kanal somit für jeden Facettentyp ein logischer Kanal zur Verfügung gestellt, der von den anderen Facetten abgehört werden kann. Hierdurch wird eine lose Kopplung zwischen den einzelnen Facetten unterstützt, in der die Verhaltensfacette als interpretierende Komponente zwischen den Facetten der Medienwiedergabe (Medien-, Layout, Gerätefacette) und der Modellfacette fungiert.

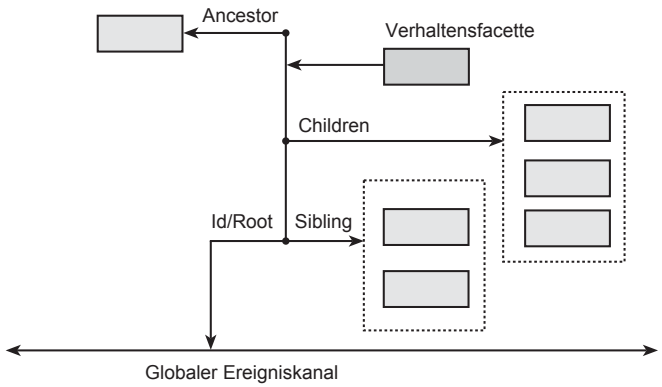


Abb. 6.4.2.2.3: Externe Kopplung der Verhaltensfacetten.

Die Kommunikation zwischen den einzelnen Verhaltensfacetten wird, wie bereits in Abschnitt 5.2.3 gezeigt, zum einen durch hierarchische Ereigniskanäle, zum anderen durch die direkte Referenzierung einzelner Verhaltensfacetten unterstützt. Um diese indirekte und direkte Referenzierung in gleicher Weise handhaben zu können, baut der externe Ereigniskanal ebenso wie der interne auf einem topologischen Kanal auf. Wie in Abb. 6.4.2.2.3 dargestellt, wird durch den externen Kanal sowohl die hierarchische Struktur als auch die direkte Referenzierung mithilfe von entsprechenden Kennungen (Root, Ancestor, Sibling, Children, Id) nachgebildet. Die Kennungen fungieren somit als Filter der Ereignisse, durch die eine lose Kopplung zwischen den einzelnen Verhaltensfacetten unterstützt wird.

6.4.2.3 Koordination

Durch die von KQML unterstützte Trennung zwischen Inhalt und Sprechakt, lässt sich die Koordination zwischen den Agenten unabhängig von dem eigentlichen „multimedialen

Inhalt“ beschreiben. Somit sind die Sprechakte mit Schablonen zu vergleichen, die für unterschiedliche Szenarien eingesetzt werden können.

KQML-Konstrukt	Agententyp	Inhaltstyp
advertise	Service-, Mittleragenten	unterstütze Sprechakte
tell, ask-about, reply	Service-, Mittleragenten	Informationen
error, sorry	Service-, Mittleragenten	-

Tab. 6.4.2.3.1: Übersicht informierender KQML-Konstrukte.

In Tab. 6.4.2.3.1 sind die von allen Service- und Mittleragenten zu verarbeitenden informierenden KQML-Konstrukte zusammengefasst. Um die in der Wissensbasis abgelegten Informationen anderen Agenten bekannt zu machen, können die Informationen den Agenten übermittelt werden (tell) bzw. von den einzelnen Agenten abgefragt (ask-about, reply) werden. Somit können die Agenten über die Struktur einer Wissensbasis und über die angebotenen Dienste informiert werden. Des Weiteren unterstützen die Agenten die Bekanntgabe der von ihnen zu verstehenden KQML-Konstrukte (advertise) und die Meldung von Fehlern (error) bzw. nicht zu erfüllender Dienste (sorry).

KQML-Konstrukt	Agententyp	Inhaltstyp
register, unregister	Umgebungsagent	Mittleragent, Service-Agent
broadcast	Umgebungsagent	KQML-Konstrukt
insert, delete	Ressourcen-Agent	Gerätefacette, Medienfacette
	Präsentationsagent	Layoutfacette
	Modellagent	Modellfacette
	Mittleragent	alle Facettentypen
ask-about, reply	Ressourcen-Agent	alle Facettentypen, Regeln
	Präsentationsagent	Layoutfacette
	Modellagent	Modellfacette
	Mittleragent	alle Facettentypen

Tab. 6.4.2.3.2: Übersicht der koordinierenden KQML-Konstrukte und Inhaltstypen.

Neben den allgemeinen informationsübermittelnden Sprechakten bieten die einzelnen Agententypen die Interpretation der in Tab. 6.4.2.3.2 zusammengefassten KQML-Konstrukte an, wobei der Inhaltstyp der Sprechakte von den angebotenen Diensten der einzelnen Agenten abhängt.

Um die Verwaltung einer lokalen Ablaufumgebung zu unterstützen, melden sich die einzelnen Agenten bei dem Umgebungsagenten an (register) bzw. ab (unregister), so dass der Umgebungsagent zu jedem Zeitpunkt die zur Verfügung stehenden Service-Agenten und die in seiner Umgebung platzierten Mittleragenten referenziert. Der Umgebungsagent ist damit die zentrale Anlaufstelle einer lokalen Ablaufumgebung, der die Gruppenkommunikation (broadcast) zwischen den Agenten ermöglicht.

Die Verwaltung der unterschiedlichen Facettentypen wird durch die spezialisierten Service-Agenten unterstützt, wobei neue Facetten eingefügt (insert) bzw. bereits verwaltete Facetten aus der Wissensbasis eines Agenten entfernt (delete) werden können. Entsprechend können die einzelnen Facettentypen von den Agenten mithilfe der Schnittstellenbeschreibung abgefragt (ask-about) werden, woraufhin die angeforderte Facette zurückgeliefert (reply) wird. Neben der einfachen Abfrage der Facetten kann durch den Ressourcen-Agenten die Auswahl der Facetten mithilfe der Regeln erfolgen, die während der Initialisierung eines Mittleragenten interpretiert werden.

Kapitel 7

Wahrnehmungsumgebungen

Wie in Kapitel 5 und Kapitel 6 gezeigt, steht durch multimediale Mittlerelemente ein durchgängiges Konzept für den Entwurfsprozess, die Architektur eines Präsentationsmediums und die Realisierung von Wahrnehmungsumgebungen zur Verfügung.

Der subjektorientierte Entwurf multimedialer Anwendungen unterstützt dabei die wichtigsten Rollen während der Produktion durch die Integration arbeitsspezifischer Programme der Experten. Dabei stellen Mittlerelemente einen Rahmen für die Definition multimedialer Anwendungen bereit, so dass die subjektorientierte Ablauforganisation für unterschiedliche Anwendungsszenarien eingesetzt werden kann. Die durch dieses Konzept integrierten Rollen unterstützen die drei wichtigsten Sichten auf multimediale Anwendungen. Zum einen sind dies die eher gestalterischen Sichten der Mediendaten und des Layouts bzw. der 3D-Modellierung, die durch Experten wie Mediengestalter und Graphikdesigner bzw. Architekten begleitet werden. Zum anderen ist dies die eher technische Sicht der Verhaltensdefinition multimedialer Mittlerelemente, die beispielsweise durch Informatiker stattfindet.

Durch die Repräsentation multimedialer Mittlerelemente als Software-Agenten innerhalb einer Ablaufumgebung wird, wie in Kapitel 6 beschrieben, die Trennung zwischen den Sichten und der eigentlichen organisierenden Struktur fortgeführt. Durch die dort vorgestellte Architektur eines Präsentationsmediums wurde somit ein Modellierungskonzept für die Integration multimedialer Inhalte eingeführt, in dem multimediale Mittlerelemente als Trägermedium auftreten. Dieses Konzept wurde in dieser Arbeit anhand des Java Media Tools [BeSy00b] [Delf99] [SBSZ99] [Otto98] umgesetzt, das als ein plattformunabhängiges Präsentationsmedium für multimediale Anwendungen dient. Hierdurch steht eine Ablaufumgebung zur Verfügung, die sowohl die Realisierung „traditioneller“ geschlossener Präsentationen als eigenständige Programme im World Wide Web als auch die Realisierung verteilter Präsentationen in realen Umgebungen, wie z.B. in Museen, unterstützt.

Um einen Eindruck der unterschiedlichen Anwendungsszenarien multimedialer Mittlerelemente zu vermitteln, wird in den folgenden Abschnitten die Definition und Realisierung von Wahrnehmungsumgebungen aus den Bereichen Edutainment, Infotainment, der Medienkunst und der Simulation anhand von im Rahmen dieser Arbeit entwickelten Beispielen vorgestellt.

7.1 Edutainment

Als Beispiel aus dem Bereich Edutainment wurde eine virtuelle Tour durch die älteste und größte neolithische Stadt, in der Kunstgegenstände gefunden wurden, als CD-ROM [Cata97a] und Internetumsetzung [Cata97b] [LöPü98] realisiert. Die Dokumentation über die Ausgrabungsstätte Çatal Höyük ermöglicht sowohl einen tiefen Einblick in den aktuellen Stand der Ausgrabung als auch eine auf den Funden basierende „Reise“ in die neolithische Zeit. Die verschiedenen inhaltlichen Bereiche der Dokumentation sind über vier Hauptseiten der Anwendung zugänglich, die inhaltsbezogene Interaktionsformen zur Verfügung stellen.

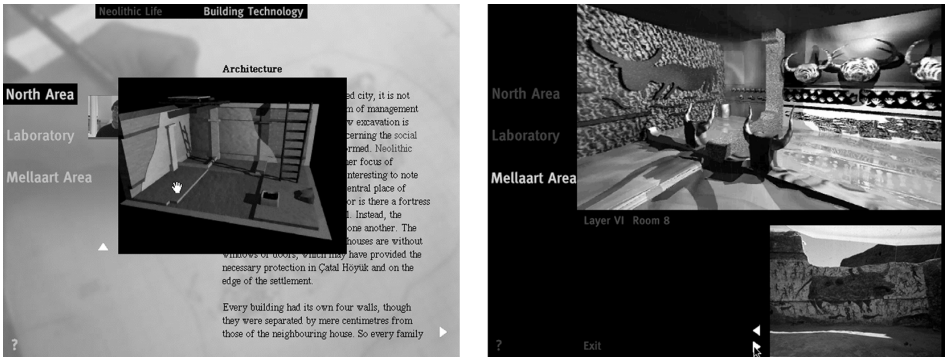


Abb. 7.1.1: Beispielseiten des „Nord-Areals“ und des „Mellaart-Areals“.

Auf der Startseite der multimedialen Anwendung erhält der Benutzer einen Überblick über das Ausgrabungsgebiet. Hier können einzelne Stellen des Ausgrabungshügels angewählt werden, von denen der Benutzer mithilfe von 360°-Panoramabildern einen Rundblick über das Gelände hat. Gleichzeitig dient der virtuelle Hügel als Orientierung und Navigationsstruktur zu den weiter führenden Hauptbereichen der Dokumentation, die die räumliche Struktur der realen Ausgrabungsstätte im Virtuellen widerspiegeln. Dabei handelt es sich um das so genannte „Mellaart-Areal“, das bereits in den 60er Jahren entdeckt wurde, das „Nord-Areal“, in dem die aktuelle Ausgrabung stattfindet, und das „Grabungshaus“, in dem die Fundstücke analysiert werden.

Im „Mellaart-Areal“ kann der Benutzer einzelne rekonstruierte Gebäude und Fundstücke betrachten, indem er unterschiedliche Bereiche und historische Ebenen des Stadtgrundrisses auswählt. Die Gebäude sind durch 360°-Panoramabilder (*QuickTime VR*) wiedergegeben, so dass der Benutzer innerhalb eines Gebäudes navigieren kann. Der Bezug zwischen virtueller Rekonstruktion und tatsächlichem Fundstück wird innerhalb der Gebäude durch die Möglichkeit der Aktivierung einzelner rekonstruierter Objekte und Wandmalereien hergestellt.

Im Gegensatz zu dem räumlichen Zugang zu der Ausgrabungsstätte und der rekonstruierten historischen Stadt, bieten die beiden anderen Hauptseiten einen themenbezogenen Zugang. Im „Nord-Areal“ sind Informationen über die aktuelle Ausgrabung zusammengestellt. Die Informationen über unterschiedlichen analytischen Betrachtungsweisen und Grabungsmethoden sind sowohl textuell als auch audiovisuell durch Fotos, Videos, Interviews und drehbare Objekte der gefundenen Skulpturen zugänglich. Zwischen den einzelnen Informationen, die die Disziplinen sowohl wissenschaftlich als auch populärwissenschaftlich wiedergeben, besteht eine themenbezogene deskriptive Navigationsstruktur. Im „Grabungshaus“ findet der Benutzer eine virtuelle Bibliothek, in der wissenschaftliche Texte über die Grabung zusam-

mengestellt sind. Um weiter führende aktuelle Informationen über Çatal Höyük zu erhalten, führen Internetadressen direkt zu den Informationsquellen im World Wide Web.

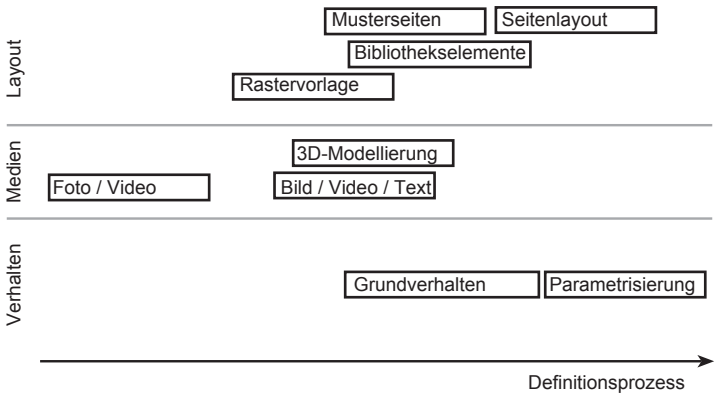


Abb. 7.1.2: Phasen des seitenorientierten Definitionsprozesses.

Für den Entwurf der multimedialen Anwendung standen bereits die meisten Mediendaten als Rohmaterial zur Verfügung, das während einer zweijährigen Dokumentation der Ausgrabung erstellt worden war. Somit waren die Daten, wie in Abb. 7.1.2 dargestellt, Ausgangspunkt des Entwurfs. Parallel zu der Aufbereitung der Daten und der Erstellung neuer Medien, wie der virtuellen Rekonstruktion der Gebäude, fand die Erstellung der Grundstruktur des Layouts statt. Der seitenbasierte Layoutentwurf basiert auf einer Rasterstruktur, die durch einzelne Musterseiten als Vorlage für die drei Hauptbereiche des „Mellaart-Areals“, des „Nord-Areals“ und des „Grabungshauses“ verfeinert wurde.

Die Definition des Verhaltens baut auf wiederverwendbaren Beschreibungen auf. Neben wiederkehrenden Elementen, z.B. dem auf allen Seiten vorhandenen Hauptmenü, wurde in einem ersten Schritt das Grundverhalten elementarer und komplexer Verhaltenssichten definiert. Die Verfeinerung dieser Sichten durch ergänzende subjektorientierte Zuordnung, wie etwa die Parametrisierung einzelner Bildfolgen, fand entsprechend den zugeordneten Mediendaten statt.

Auch wenn der seitenorientierte Aufbau der multimedialen Anwendung durch die in Abschnitt 3.2.1.2 vorgestellte *Relationship Management Methodology* unterstützt wird, zeigt sich gerade durch die Anordnung der einzelnen Entwurfsphasen die Stärke des subjektorientierten Entwurfs.

Im Gegensatz zu den in Abschnitt 3.2.1 vorgestellten strukturorientierten Verfahren, die auf formalen Beschreibungsformen der Informatik beruhen, wird in diesem Beispiel der Entwurf aus Layoutsicht vorangetrieben. Dabei werden die einzelnen rollenspezifischen Arbeitsschritte eines Graphikdesigners unterstützt. Hierdurch wird die Gesamtsicht auf die multimediale Präsentation mithilfe eines mentalen Modells während des Entwurfs ermöglicht. Erst das entwickelte Rasterlayout und die Zuordnung der Mediendaten gibt die Struktur der multimedialen Präsentation implizit vor. Gerade hierin unterscheidet sich der zeitliche Ablauf des Entwurfsbeispiels von den strukturorientierten Verfahren, bei denen das Layout ausschließlich in späten Phasen des Entwurfsprozesses definiert wird.

Auch wenn der layoutbasierte Ansatz von den filmorientierten Methoden unterstützt wird, eignen sie sich für den Entwurf eines Rasterlayouts nicht. So würde z.B. der Entwurf mithilfe des in Abschnitt 3.2.2.3 vorgestellten Ansatzes *Designing Multimedia Applications with Interactive Storyboards* zwar für die Erprobung erster Ideen einzusetzen sein, doch unterstützt dieser Ansatz weder die typischen Arbeitsschritte während der Definition eines Rasterlayouts noch die Abbildung auf unterschiedliche Ablaufumgebungen.

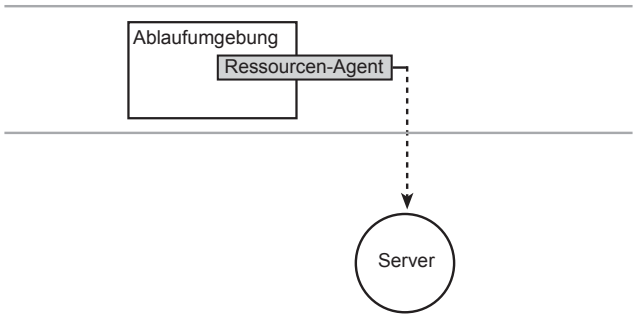


Abb. 7.1.3: Geschlossene Ablaufumgebung als Applet.

Auch die Realisierung der Ablaufumgebung folgt dem traditionellen seitenorientierten Entwurf als geschlossene Anwendung. Somit ist die interaktive Dokumentation zum einen durch das Java Media Tool als Applet im World Wide Web [Cata97b], zum anderen als CD-ROM Version [Cata97a] zugänglich. Innerhalb der WWW-Umgebung wird, wie in Abb. 7.1.3 dargestellt, auf die Seitenbeschreibungen und Mediendaten, die auf einem Server abgelegt sind, mithilfe des Ressourcen-Agenten zugegriffen.

7.2 Infotainment

Als Beispiel aus dem Bereich Infotainment dient der Entwurf einer interaktiven Dokumentation über die Nachkriegszeit in Saarbrücken, der sowohl als Vorlage für die Abbildung auf eine CD-ROM Präsentation als auch als Vorlage für die Realisierung einer verteilten multimedialen Anwendung verwendet wurde [BeSy00a].

Die Dokumentation ist in drei Hauptbereiche unterteilt, die unterschiedliche Zugangsformen zu dem Thema zur Verfügung stellen. Im ersten Bereich, der der zeitlichen Orientierung dient, können auf einer Zeitachse 19 Videos abgerufen werden. Die Videos, die durch einen Sprecher erläutert werden, zeigen den zeitlichen Ablauf der Entwicklung der Nachkriegszeit und setzen sich aus historischen Fotografien, Briefen und virtuell rekonstruierten Gebäuden zusammen. Werden die Videos in der durch die Zeitachse vorgegebenen Reihenfolge ausgewählt, entsteht eine kontinuierliche Geschichte, die einen Eindruck von der Trümmerzeit und der sich anschließenden Planungsphase bis hin zum Wiederaufbau vermittelt.

Der zweite Bereich stellt einen spielerischen Zugang zu den Videos dar. Hier kann der Benutzer den Stadtplan Saarbrückens als Puzzle zusammensetzen. Wird ein Puzzleteil an die korrekte Position platziert, kann über dieses Teil eines der Videos aufgerufen werden, so dass hier die Videos mithilfe des Stadtplans in eine räumliche Struktur eingebunden sind.

Abb. 7.2.1: Abbildung der dreidimensionalen Testumgebung.

Der dritte Bereich der Dokumentation ermöglicht den Zugang zu Darstellungen rekonstruierter Gebäude der Planungszeit und zu wichtigen historischen Plätzen und Gebäuden Saarbrückens. Die Gebäude und Plätze sind durch 360°-Panoramabilder realisiert. Der Betrachter kann innerhalb einer Darstellung die Blickrichtung bestimmen und sich durch die Gebäude virtuell bewegen.

Für die Gestaltung einer verteilten Wahrnehmungsumgebung wurden die einzelnen Bereiche auf in der realen Umgebung angeordnete Leinwände, wie in Abb. 7.2.1 dargestellt, projiziert. Somit dient die multimediale Anwendung zum einen als „passiver Film“, der innerhalb des Raums zu beobachten ist, zum anderen als interaktive Informations- bzw. Spielumgebung, die durch die Besucher über eine Navigationskugel gesteuert werden kann.

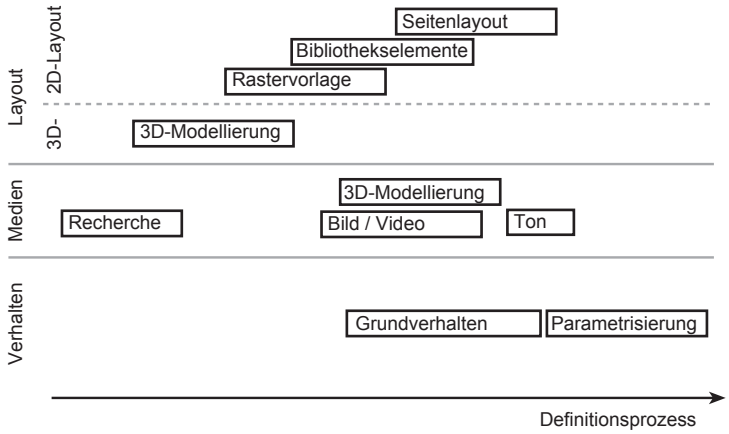


Abb. 7.2.2: Phasen des Definitionsprozesses.

Der Entwurfsprozess der interaktiven Installation setzt sich, wie in Abb. 7.2.2 dargestellt, aus einer zweidimensionalen Beschreibung der einzelnen Bereiche und einer dreidimensionalen Definition der realen Wahrnehmungsumgebung, die ausschließlich für die Konzeption und den Test der Installation verwendet wurde, zusammen. Die zweidimensionale Beschreibung basiert auf einer Rasterstruktur, die das Raster des Stadtplans widerspiegelt. Neben der Unterstützung der Platzierung der Layoutelemente, die zum Teil als Bibliothekselemente definiert wurden, wurde durch das Raster auch die Gestaltung der Mediendaten, wie etwa der Videos, vorgegeben, die in dieses Raster eingeordnet sind. Somit war der Layoutentwurf die

Grundlage des Konzepts, in den die Mediendaten und das Verhalten der Installation integriert wurden.

Bei der Definition des Verhaltens wurden einzelne wiederkehrende Verhaltensbeschreibungen entwickelt. Somit konnte etwa das Verhalten der Puzzleteile und der Indexeinträge zentral durch einzelne Verhaltensdefinitionen beschrieben werden, die durch ergänzende Beschreibungen parametrisiert wurden. Hierbei wurde z.B. die korrekte Position der Puzzleteile festgelegt und der Verweis auf das jeweils abzuspielende Video definiert.

Auch in diesem Anwendungsbeispiel wird der Entwurf durch gestalterische Entscheidungen geleitet. Hier kommt zum einen die Gestaltung der realen Umgebung, zum anderen die Gestaltung der virtuellen Umgebung zum Tragen. Dabei wird im Gegensatz zu den in Kapitel 3 vorgestellten Verfahren der Entwurf der realen und virtuellen Umgebung gleichberechtigt behandelt.

Die einzelnen rollenspezifischen Arbeitsschritte werden durch die entsprechenden Programme unterstützt, so dass die Spezialisten mit ihren gewohnten Werkzeugen die Ablaufumgebung definieren können. Erst durch die Transformation wird die virtuelle auf die reale Umgebung abgebildet. Dabei wird durch die Ablaufumgebung die horizontal und vertikal offene Modellierung unterstützt.

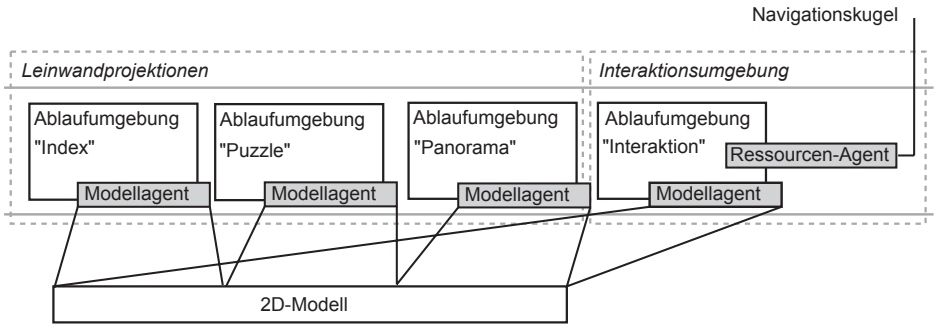


Abb. 7.2.3: Struktur der Ablaufumgebung.

Für den Test der Installation wurde die vollständige Beschreibung ausschließlich als geschlossene virtuelle Umgebung umgesetzt. Die eigentliche Realisierung als verteilte Anwendung fand durch die Abbildung auf das Java Media Tool statt. Die Struktur der Ablaufumgebung entspricht dabei der realen Wahrnehmungsumgebung. Wie in Abb. 7.2.3 dargestellt, werden die drei Leinwandprojektionen über lokale Umgebungen angesteuert. Innerhalb dieser lokalen Ablaufumgebungen wird der Aufbau der internen Präsentationsumgebung durch die jeweilige seitenbasierte Beschreibung bestimmt.

Um diese Umgebungen mit einer einzelnen Navigationskugel interaktiv beeinflussen zu können, sind die lokalen Mittlerelemente über zweidimensionale Modellfacetten mit einer virtuellen Benutzeroberfläche verbunden, so dass die Interaktionsereignisse an die entsprechende lokale Ablaufumgebung weitergeleitet werden.

Durch die Trennung in Modellebene, interne und externe Präsentationsebene unterstützt die Ablaufumgebung die unterschiedlichen Ansätze von komponenten- und agentenorientierten Softwarearchitekturen. So wird die horizontal verteilte Präsentation unterstützt, indem die Mittlerelemente als eigenständige Software-Elemente fungieren.

Damit sind sie mit den in Abschnitt 4.3.3 vorgestellten anwendungsbezogenen Agenten vergleichbar. Im Gegensatz zu den Agenten dienen die Mittlerelemente der Ablaufumgebung jedoch zunächst als abstrakte Software-Elemente, so dass die funktionale Sicht auf die verteilte Präsentationsumgebung durch den zweidimensionalen euklidischen Raum der Modellebene gebildet wird. Innerhalb der einzelnen Umgebungen sind die Mittlerelemente mit den in Abschnitt 4.2.3 vorgestellten anwendungsbezogenen Komponentenmodellen vergleichbar.

7.3 Medienkunst

Im Bereich der Medienkunst wurde der subjektorientierte Entwurf für die Planung, Konzeption und Demonstration einer ausstellungsunterstützenden multimedialen Präsentation verwendet [BeSy00c]. Die Ausstellung, die im Rahmen der EXPO2000 und der Feier des tausendjährigen Bestehens Saarbrückens stattfand, behandelte die Umwandlung des früheren Gebiets der Saarterrassen in ein Technologiegebiet, so dass neben den historischen Aspekten die Konzepte der Umwandlung und insbesondere die neuen Technologien des Informationszeitalters thematisiert wurden. Zentraler Ausstellungsort war das so genannte Expomedia-Gebäude, dessen Medienfassade interaktiv gestaltet werden sollte.

Abb. 7.3.1: Darstellung der virtuellen Testumgebung.

Das hier vorgestellte Konzept sieht die Einbeziehung der realen Umgebung des Gebäudes vor. So wurden für den zentralen Weg, der zu dem Ausstellungsgebäude führt, so genannte Hörstellen entworfen. An den dort stehenden Straßenlampen montiert, reagieren sie auf vorbeigehende Passanten und spielen daraufhin zu der Ausstellung erläuternde Texte, Interviews und in Bezug stehende Musikstücke ab. Die Ansteuerung erfolgt dabei über Näherungssensoren, die in der realen Umsetzung als Infrarotsensoren realisiert sind.

Die Ansteuerung der Medienfassade, die aus einer Rasterstruktur von LED-Röhren und einer LED-Leinwand besteht, kann durch ein frei definierbares Drehbuch bestimmt werden, das über das Internet zu bedienen ist. Somit dient die entwickelte Ablaufumgebung ausschließlich als Rahmen für die auf der Medienfassade wiederzugebenden Mediendaten.

In das Drehbuch können dabei Mediendaten für die LED-Leinwand und die LED-Röhren als Videos, Texte oder Bilder eingebaut werden, bzw. Sequenzen für die Projektion realer Umgebungsparameter integriert werden. So können z.B. einzelne Videos auf der LED-Leinwand

abgespielt werden, wobei die Medienfassade als Abstraktionsfläche der Videos ausschließlich die Konturen der Videobilder wiedergibt. Bei der Darstellung der vorbereiteten Mediendaten kann durch Umwelteinflüsse das Erscheinungsbild der Medienfassade auch direkt beeinflusst werden. Die Fassade wird dabei zur virtuellen Projektionsfläche der realen Umgebung, die durch unterschiedliche Farben die Helligkeit der Umgebung widerspiegelt, wobei die abgestuften Farben sich der Geschwindigkeit und Richtung des Winds anpassend über die Fassade bewegen.

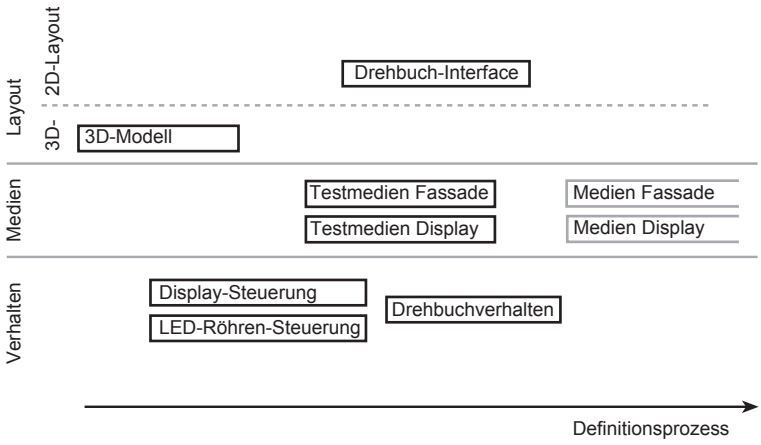


Abb. 7.3.2: Phasen des Definitionsprozesses.

Im Gegensatz zu den vorherigen Beispielen ist während des Entwurfs der Display- und Fassadensteuerung, wie bereits weiter oben beschrieben, ausschließlich ein Rahmen definiert worden, um unterschiedliche abzuspielende Mediendaten integrieren zu können. Somit beschränkt sich der Entwurf hauptsächlich auf die Schaffung einer virtuellen Umgebung, die aus dem virtuellen Nachbau der Ausstellungsumgebung und der funktionalen Beschreibung der Ansteuerung besteht. Erst in späteren Schritten wurden, wie in Abb. 7.3.2 dargestellt, entsprechende Mediendaten für die Wiedergabe auf der LED-Leinwand und der Medienfassade erstellt.

Für den Entwurf der Umgebung wurde zunächst das Ausstellungsgebäude als Drahtgittermodell konstruiert, das als Test und Demonstrationsmodell diente. Um die Steuerung bzw. die Mediendaten innerhalb des Modells integrieren zu können, wurden dabei die realen Präsentationsmedien durch virtuelle Lichtquellen als Nachbildung der LED-Röhren und eine virtuelle Projektionsfläche als Nachbildung der LED-Leinwand modelliert.

Parallel zu der Nachbildung der realen Umgebung als virtuelle Simulationsumgebung fand die Definition des Verhaltens der Mittlerelemente statt. Diese besteht dabei zum einen aus den grundlegenden Funktionen für die Interpretation der Mediendaten, zum anderen aus den Funktionen für die Steuerung der Präsentation, die die Interpretation der Drehbuchinhalte umfasst.

Für die Erstellung neuer Drehbücher wurde der Entwurf durch eine Benutzeroberfläche erweitert, durch die bereits vorhandene Mediendaten bzw. neu erstellte Daten innerhalb eines Zeitplans integriert werden können. Um dabei die realen Umgebungsparameter während des Tests und der Demonstration simulieren zu können, wurden diese zunächst als Standardele-

mente der Benutzeroberfläche modelliert, so dass sie über das Drehbuch interaktiv zu steuern sind. Erst während der Realisierungsphase der multimedialen Installation werden die virtuellen durch die realen Präsentationsmedien ersetzt, wobei die multimedialen Mittlerelemente die Ansteuerung der LED-Röhren und der LED-Leinwand übernehmen bzw. die Umgebungsparameter über Sensoren erfassen.

Somit führt der Entwurf, im Gegensatz zu den in Kapitel 3 vorgestellten Methoden, in diesem Beispiel zunächst nur zu einer speziellen Ausprägung einer Ablaufumgebung. Erst die Planung des Drehbuchs bestimmt das eigentliche Erscheinungsbild der Anwendung. Der dreidimensionale Entwurf dient dabei ausschließlich, vergleichbar mit dem vorherigen Beispiel, als Testumgebung. Erst durch die Transformation wird diese Testumgebung auf die reale Installation abgebildet.

Auch wenn durch die technische Sicht dieses Projekts der Entwurf zunächst mit der *Object-Oriented Hypermedia Design Method (OOHDM)* zu vergleichen ist bzw. der Entwurf der Drehbuchoberfläche mit der *Scenario-Based Hypermedia Design Methodology (SHDM)* hätte stattfinden können, wird gerade durch die Trennung zwischen realer und virtueller Umgebung und der damit verbundenen automatischen Transformation der Vorteil des subjektorientierten Ansatzes deutlich. Durch die getrennte Behandlung der Sichten wird dabei der Entwurf auf unterschiedlichen Abstraktionsstufen vorangetrieben. Zum einen wird durch die dreidimensionale Beschreibung die reale Umgebung nachgebildet, zum anderen wird durch die Definition der eigentlichen Ablaufumgebung ausschließlich der Rahmen für die spätere Präsentation vorgegeben.

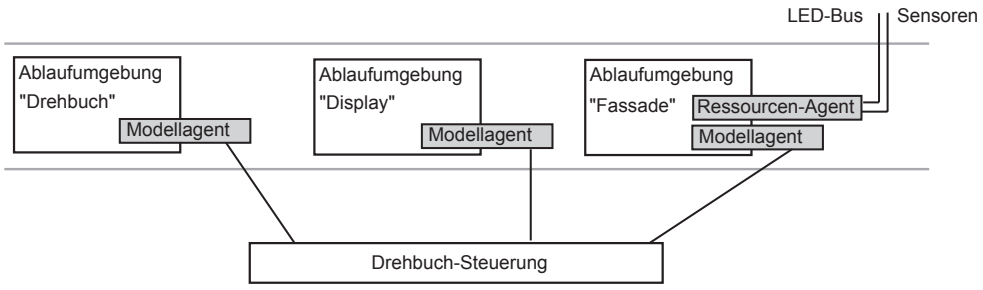


Abb. 7.3.3: Struktur der Ablaufumgebung für die Display- und Fassadensteuerung.

Die verteilte Ablaufumgebung für die Leinwand- und Fassadensteuerung gibt einen Rahmen vor, der die Wiedergabe unterschiedlicher Mediendaten unterstützt. Zentrales Element dieser Steuerung ist das Drehbuch, das durch eine speziell entwickelte Modellumgebung als aktive Datenbank interpretiert wird. Somit steht die Modellumgebung, wie in Abb. 7.3.3 dargestellt, zwischen der internen Präsentationsumgebung der LED-Leinwand, der externen Fassadensteuerung und der Definitionsumgebung des Drehbuchs.

Die Präsentationsumgebungen ermöglichen die Wiedergabe der Mediendaten, wobei die Ansteuerung und Synchronisation zwischen den Ablaufumgebungen über die Modellfacetten der Mittlerelemente stattfindet. Innerhalb der lokalen Ablaufumgebung der Displaysteuerung werden die abzuspielenden Mediendaten (Filme, Bilder, Texte) dargestellt. Die Ansteuerung der Fassade erfolgt durch die Mittlerelemente der zweiten Umgebung über ein Bussystem, mit dem die LED-Röhren durch Controller verbunden sind.

Im Gegensatz zu dem vorherigen Beispiel, bei dem die verteilte Ablaufumgebung bereits eine fest vorgegebene Präsentation umsetzt, dient hier die Ablaufumgebung ausschließlich als vorgegebener Rahmen für unterschiedliche Mediendaten. Hierdurch wird der hohe Abstraktionsgrad der in dieser Arbeit vorgestellten Softwarearchitektur deutlich. Einerseits wird auch hier die vertikal und horizontal offene Gestaltung als Präsentationsmedium unterstützt, andererseits dient hier die Ablaufumgebung als Rahmen, der die eigentliche Mediendaten integriert.

Vergleichbar mit dem Entwurf sind die multimedialen Mittlerelemente somit zunächst ausschließlich aus technischer und gestalterischer Sicht definiert. Die Zuordnung der Medien findet erst während der eigentlichen Präsentation mithilfe des Drehbuchs statt. Somit können, im Gegensatz zu den zu weit gefassten komponentenorientierten bzw. zu eng gefassten agentenorientierten Präsentationsmedien, die multimedialen Mittlerelemente auf unterschiedlichen Abstraktionsniveaus betrachtet werden.

7.4 Simulation

Im folgenden Beispiel aus dem Bereich der Simulation wurde eine Umgebung entwickelt [Pütt00], die die Beobachtung so genannten künstlichen Lebens [Lang97] [Adam98] ermöglicht. In den vorherigen Beispielen ging es vorwiegend um einen gestalterischen Ansatz, bei dem durch die Verhaltenssicht auf die Mittlerelemente ausschließlich die Interaktionsmöglichkeiten und die Steuerung der Mediendaten definiert wurden. Der Bereich des künstlichen Lebens verbindet dagegen bereits unterschiedliche interdisziplinäre Ansätze, um das Verhalten biologischer, ökologischer oder soziologischer Systeme innerhalb einer künstlichen „Computerwelt“ nachzubilden. Wie bereits in Abschnitt 4.3.4 gezeigt, werden dabei einzelne Elemente eines Systems durch künstliche Individuen modelliert, deren Verhalten durch Lernverfahren, evolutionäre Algorithmen bzw. zellulare Automaten beschrieben wird. Bei diesem eher von der Verhaltenssicht motivierten Ansatz gewinnt in jüngster Zeit zunehmend aber auch die gestalterische Sicht an Bedeutung.

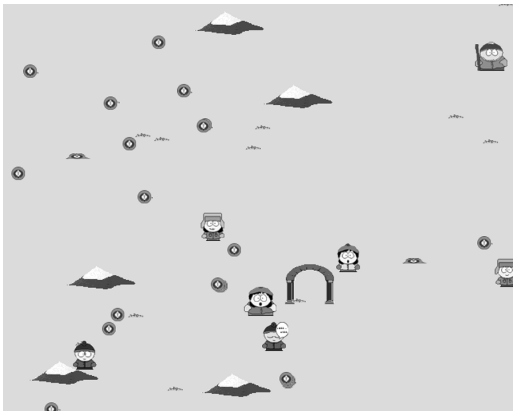


Abb. 7.4.1: Ausschnitt der figurativen Darstellung der künstlichen Welt.

Zum einen ist hier sicherlich die Interpretation bzw. Beobachtbarkeit komplexer Systeme zu nennen, die durch eine geeignete Darstellung unterstützt werden kann. Wie bereits in Abschnitt 2.3 gezeigt, sollte diese Darstellung neben deskriptiven Interaktionsformen für die

Steuerung der Simulation ebenso die Interpretation der Simulationsdaten als ikonische Darstellung veranschaulichen. Zum anderen findet der Bereich des künstlichen Lebens verstärktes Interesse in der Medienkunst, in der „lebende“ Strukturen als formbildende Systeme betrachtet werden [SoMi98].

Die hier entwickelte künstliche Welt simuliert ein verteiltes Ökosystem, in dem künstliche Individuen unterschiedliche Aktionen ausführen können, um in dieser Welt zu überleben. Für die Modellierung stehen dabei unterschiedliche Arten von Objekten und unterschiedlich ausgeprägte Individuen zur Verfügung. Die statischen Objekte der Welt sind Hindernisse, Nahrungselemente und Portale, die in der künstlichen Welt zufällig platziert werden. Als Individuen der Welt existieren so genannte Schwärmer, die sich frei bewegen können und sich durch die statischen Nahrungselemente ernähren. Die Schwärmer wiederum stellen die Nahrungsquelle für die höher entwickelten Individuen, die so genannten Bewohner, dar, die in Gruppen Jagd auf die Schwärmer machen [Pütt00].

Jeder Bewohner ist charakterisiert durch seine Physiologie, welche durch Geschlecht, Alter, Energie, Gesundheit, Geschwindigkeit und Libido festgelegt ist. Je nach den aktuellen Umgebungseigenschaften und der Ausprägung der Physiologie wird das Verhalten eines Bewohners bestimmt, so dass dieser z.B. die Suche bzw. Jagd nach neuer Beute, die Suche nach einer neuen künstlichen Welt über ein Portal oder die Suche nach einem Reproduktionspartner aufnehmen kann. Um die einzelnen Aufgaben zu erledigen, nehmen die Bewohner ihre unmittelbare Umgebung wahr und kommunizieren mit anderen Bewohnern, um Gruppen für die gemeinsame Jagd zu bilden. Hierdurch können während einer Simulation komplexe Gemeinschaften gebildet werden, um die unterschiedlichen physiologischen Eigenschaften mehrerer Bewohner gewinnbringend zu verbinden.

Zur Demonstration der gestalterischen Sichten auf die künstliche Welt wurden verschiedene Darstellungsformen entwickelt, die einen unterschiedlichen Zugang zu der Simulationsumgebung unterstützen. Wie bereits weiter oben angesprochen, wird hierbei die künstliche Welt als Simulationsumgebung durch einen deskriptiven Zugang ermöglicht bzw. durch eine abstrakte Sicht als sich veränderndes formbildendes System betrachtet.

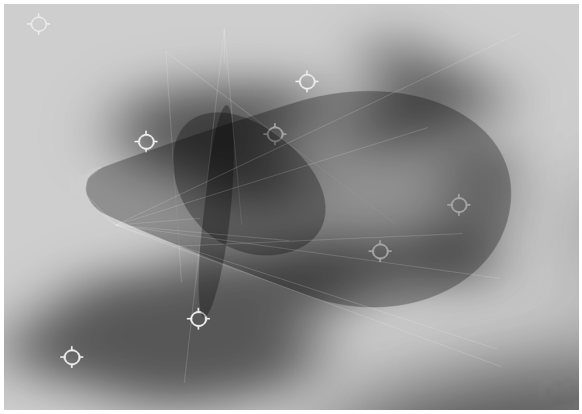


Abb. 7.4.2: Organische Darstellung der Gruppen.

Durch die Gestaltung der Benutzeroberfläche als Simulationsumgebung lassen sich die wichtigsten Parameter der künstlichen Welt beobachten, indem die einzelnen Objekte und Individuen, wie in Abb. 7.4.1 dargestellt, durch figurative Symbole repräsentiert werden. Die

unterschiedlichen Zustände der Individuen werden durch charakteristische Abbildungen symbolisiert, so dass eine einfache Interpretation der Parameter unterstützt wird. Hierdurch wird insbesondere ein schneller Überblick über sich ändernde Beziehungen und Verhaltensweisen der Individuen ermöglicht. Neben dieser symbolischen Repräsentation werden unterschiedliche deskriptive Beobachtungs- und Interaktionsmöglichkeiten angeboten, die eine analytische Interpretation der künstlichen Welt erlauben. So kann der Benutzer z.B. allgemeine Informationen über den aktuellen Zustand der Welt erhalten sowie Daten einzelner Individuen als Detailansicht und die Historie der Welt betrachten. Um die Simulation beeinflussen zu können, stehen dem Benutzer die Steuerung der Simulationsgeschwindigkeit sowie das Setzen einzelner Parameterwerte der Individuen und der künstlichen Welt zur Verfügung.

Die zweite entwickelte Darstellungsart abstrahiert von den Parametern der künstlichen Welt, indem die Gruppen der Individuen als organische Formen betrachtet werden. Wie in Abb. 7.4.2 dargestellt, werden die Gruppen durch halbtransparente Flächen repräsentiert, deren Form und Größe sich nach der Gruppenausbreitung richten. Je nach Zustand einer Gruppe werden mithilfe eines Sprachsynthese-Programms unterschiedliche Informationen über den Gruppenleiter verbalisiert, die der Rezipient als „Hintergrundgespräche“ innerhalb der Umgebung wahrnimmt.

Die Entwicklung der Simulationsumgebung basiert auf der Verhaltensdefinition der unterschiedlichen virtuellen Individuentypen und Objekttypen innerhalb der künstlichen Welt. Im Gegensatz zu den passiven Objekten der Welt, die ausschließlich durch eine Zustandsbeschreibung definiert wurden, sind die Individuen der Welt durch ein aktives Verhalten charakterisiert.

Wie weiter oben bereits gesagt, können die Individuen selbstständig Aktionen ausführen, um sich in der Welt zu bewegen, Nahrung aufzunehmen etc. Ein Individuum ist dabei durch sein internes Verhalten, seinen Zustand und durch virtuelle Sensoren definiert. Das interne Verhalten beschreibt die reaktiven Aktionen, die je nach internem Zustand auszuführen sind. Im Fall der Bewohner spiegelt der Zustand die aktuelle Physiologie, die Informationen über die erfasste Umgebung und die aktuellen Ziele eines Bewohners wider. Um die Wahrnehmung der Umgebung und die Kommunikation zwischen den Bewohnern zu modellieren, wurde die Schnittstelle der Bewohner mithilfe von KQML-Sprechakten definiert.

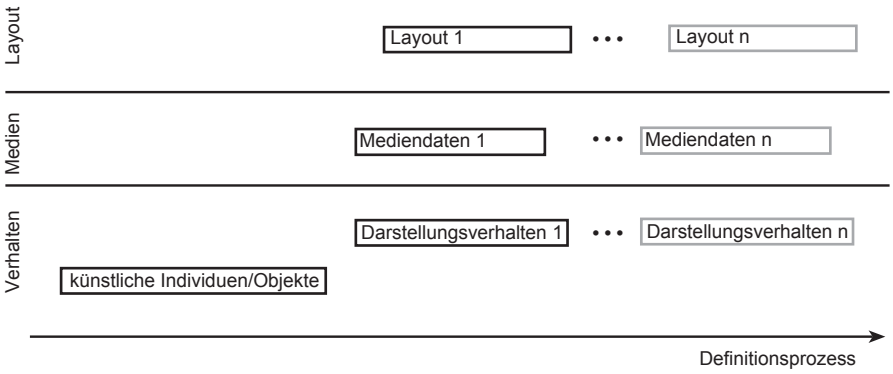


Abb. 7.4.3: Phasen des Definitionsprozesses.

Unabhängig von der eigentlichen funktionalen Beschreibung der Individuen fand, wie in Abb. 7.4.3 dargestellt, in nachfolgenden Schritten der Entwurf unterschiedlicher Darstel-

lungsarten der künstlichen Welt statt, wobei als Beispiel hier ausschließlich der deskriptive Entwurf betrachtet werden soll.

Bei dieser Darstellungsart werden die verschiedenen Zustände der Individuen durch einzelne Bilder charakterisiert, die die Interpretation des aktuellen Zustands eines Individuums ermöglichen. Um den Zustandswechsel bildlich umsetzen zu können, wurden parallel zu der Erstellung der Mediendaten die Verhaltenssichten ergänzt, so dass die Mittlerelemente neben der Beschreibung des Verhaltens in der virtuellen (künstlichen) Umgebung durch das Verhalten in der Präsentationsumgebung definiert sind. Durch dieses Verhalten wird die Interpretation der Parameter der Individuen beschrieben, um den Austausch der Mediendaten und die automatische Positionierung der Darstellung zu steuern. Erst während der Abbildung der Beschreibung auf eine verteilte Simulationsumgebung wurden die Verhaltensaspekte der Mittlerelemente, wie in Abschnitt 5.3.3 beschrieben, auf die Verhaltensfacetten und die Modellfacetten verteilt.

Da die Position der Mediendaten durch die Simulation direkt gesteuert wird, beschränkt sich die Beschreibung des Layouts auf die Definition der Darstellungsgröße der einzelnen Mediendaten und die relative Positionierung eines einzelnen Individuums.

Die Interpretation der Parameter der Individuen ist mit der in Abschnitt 3.2.2.2 eingeführten *Dynamic Design Method* vergleichbar. Durch diesen eher filmischen Ansatz wird die Gesamtsicht einer Präsentation unterstützt, bei der die einzelnen Mediendaten bzw. Inhalte als Charaktere einer Szene auftreten. Wie anhand dieses Beispiels gezeigt, werden dabei durch den subjektorientierten Entwurf unterschiedliche Interpretationsebenen unterstützt.

Neben der unterschiedlichen graphischen Interpretation der Daten, wird durch den subjektorientierten Entwurf allerdings ebenso die Bildung zusammenhängender Darstellungen unterstützt. Zum einen wird dies durch die Definition komplexer Sichten einzelner Individuen bzw. Mittlerelemente, zum anderen durch die Gruppierung einzelner Mittlerelemente zu komplexen Gebilden ermöglicht. Somit reicht die Definition der Sichten weiter als die ausschließlich auf einzelne Agenten bezogenen Beschreibungsmöglichkeiten der *Dynamic Design Method (DDM)*.

Wie in Abschnitt 3.2.2.2 gezeigt, eignet sich DDM daher eher zur Definition kleiner Anwendungen. Dabei ist die Beschreibung allerdings auf eine funktionale Definition beschränkt, durch die sowohl das Verhalten als auch das Erscheinungsbild beschrieben werden.

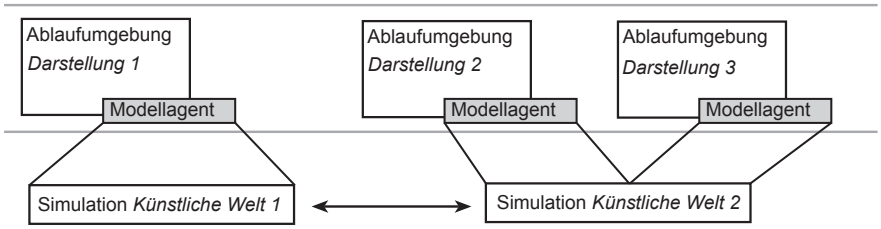


Abb. 7.4.4: Beispiel einer verteilten Simulationsumgebung mit zwei künstlichen Welten.

In Abb. 7.4.4 ist eine mögliche Struktur der Simulationsumgebung wiedergegeben. Hierbei ist zu beachten, dass durch die Trennung zwischen der Modellumgebung, in der die künstliche Welt simuliert wird, und den unterschiedlichen Präsentationsumgebungen verschiedene

Konfigurationen je nach Anwendungskontext unterstützt werden können. Die lokalen Präsentationsumgebungen sind dabei über die Modellagenten mit der eigentlichen Simulation verbunden, so dass die multimedialen Mittlerelemente durch ihre Modellfacetten die Objekte bzw. Individuen der Welt repräsentieren.

Im Gegensatz zu den bisher betrachteten Beispielen können mehrere Modellumgebungen innerhalb einer Anwendung eingebunden werden, um unterschiedliche Welten miteinander zu verbinden. Wie bereits weiter oben beschrieben, können die Mittlerelemente dabei die Modellumgebungen über die Portale aufsuchen, wobei sie je nach Präsentationsumgebung unterschiedliche Darstellungsformen annehmen.

Der Aufbau der Ablaufumgebung verdeutlicht die horizontal offene Gestaltung des Präsentationsmediums. Im Gegensatz zu den vorherigen Beispielen, bei denen bereits durch den Entwurf eine feste Struktur der Ablaufumgebung vorgegeben ist, werden hier ausschließlich die einzelnen Mittlerelemente vorgegeben, die sich innerhalb der Umgebung frei bewegen. Somit gleicht die Ablaufumgebung den Softwarearchitekturen für mobile Agenten, die in Abschnitt 4.3.3 als anwendungsbezogene Agentensysteme vorgestellt wurden.

Doch auch hier, vergleichbar mit den vorherigen Beispielen, ist die in dieser Arbeit vorgestellte Softwarearchitektur ausschließlich der abstrakte Rahmen für die spezielle Ausprägung der Anwendung. So können die Mittlerelemente innerhalb einer Ablaufumgebung unterschiedliche Erscheinungsformen annehmen bzw. ihr Verhalten ändern. Die Mittlerelemente dienen somit wiederum als Träger bzw. Vermittler des eigentlichen Inhalts der Anwendung.

Kapitel 8

Zusammenfassung

In dieser Arbeit wurde ein integrierendes Konzept für die Entwicklung multimedialer Anwendungen und multimedialer Umgebungen eingeführt, wobei der Produktionsprozess durch so genannte multimediale Mittlerelemente in einheitlicher Weise beschrieben wird. Die multimedialen Mittlerelemente dienen dabei als vermittelnde Elemente innerhalb des Entwurfsprozesses, innerhalb des Präsentationsmediums und innerhalb von Wahrnehmungsumgebungen.

Das auf dem integrierenden Konzept aufbauende Entwurfsparadigma unterstützt die unterschiedlichen Rollen der an der Produktion beteiligten Spezialisten und verbindet die einzelnen Phasen des Entwurfs mithilfe einer formalen Beschreibung. Eine Anwendung wird somit durch subjektive Sichten beschrieben, die sich sowohl bei der Definition aus Expertensicht wiederfinden als auch bei der Repräsentation innerhalb der formalen Beschreibung. Hierdurch werden, wie in Integrationskonzepten üblich, die rollenspezifischen Arbeitsweisen mit der technischen Umsetzung verbunden. Dabei stehen, wie in Kapitel 5 gezeigt, die multimedialen Mittlerelemente als integrierende Elemente während des subjektorientierten Entwurfs im Vordergrund, durch die die semiotischen Aspekte multimedialer Anwendungen (Mediendaten, Form, Verhalten) miteinander in Beziehung gesetzt werden.

Die drei Aspekte werden durch die vorgestellte Ablauforganisation als rollenspezifische Sichten berücksichtigt, so dass während des Entwurfs eine multimediale Anwendung aus unterschiedlichen, gleichberechtigten Blickwinkeln beschrieben werden kann, die den dominierenden Eigenschaften der Rollen der Spezialisten während des Entwurfsprozesses entsprechen. Die Sichten der Mediendaten und des Layouts bzw. des 3D-Entwurfs spiegeln dabei typischerweise die gestalterischen Rollen von Medienspezialisten und Graphikdesignern bzw. 3D-Modellierern wider, wohingegen die Beschreibung des Verhaltens die Rolle der Informatiker bzw. Programmierer unterstützt.

Bei den zurzeit gängigen Ablauforganisationen lassen sich zwei grundsätzliche Ansätze unterscheiden. Zum einen wird ein spezieller Inhalt multimedial präsentiert, so dass Entwurfsmethoden der Filmproduktion entlehnt werden. Zum anderen entsteht ein ablauffähiges Programm, so dass die Ansätze der Softwaretechnik verwendet werden, die aber ausschließlich auf technischen Modellierungsformen basieren. Entwurfsmethoden, die auf einem dieser beiden Ansätze beruhen haben den Nachteil, dass sie zu kurz greifen, da die Ansätze den

gestalterischen und technischen Aspekt multimedialer Anwendungen nicht gleichberechtigt unterstützen und somit eine Seite immer überwiegt bzw. zu kurz kommt.

Die Vorteile gegenüber Ablauforganisationen der Softwaretechnik, bei denen feste Phasen bereits ein enges Korsett für die Organisation vorgeben, liegen bei dem subjektorientierten Entwurf in der Betonung der Dynamik zwischen den einzelnen Rollen und den rollenspezifischen Arbeitsweisen. Hierdurch wird erreicht, dass durch die Ablauforganisation die Arbeitsweisen der an dem Entwurf beteiligten Spezialisten selbst im Vordergrund stehen, und somit eine dem jeweiligen Anwendungskontext angemessene Vorgehensweise unterstützt werden kann. Gegenüber den Ablauforganisationen der Filmproduktion, bietet die in dieser Arbeit eingeführte Ablauforganisation einen flexiblen Rahmen, innerhalb dessen das Zusammenspiel zwischen den unterschiedlichen Rollen ermöglicht wird, das gerade während der Entwicklung multimedialer Anwendungen charakteristisch ist.

Um dabei ein durchgehendes Konzept zwischen Entwurf, Präsentationsmedium und Wahrnehmungsumgebung zu realisieren, basiert die Ablauforganisation auf einer Dokumentenarchitektur. Die Mittlerelemente dienen innerhalb des Manipulationsmodells als Metazeichen des mentalen Modells und werden durch das Repräsentationsmodell in einer formalen Sprache beschrieben, so dass zum einen die Rollen der Spezialisten unterstützt werden, zum anderen die Abbildung der Repräsentationsbeschreibung auf unterschiedliche Präsentationsmodelle ermöglicht wird. Durch die Repräsentationsbeschreibung werden die einzelnen Sichten separat dargestellt, so dass die Trennung zwischen den rollenspezifischen Sichten beibehalten wird. Entgegen herkömmlichen Repräsentationsmodellen, die durch eine vorherrschende Struktur bestimmt sind, werden somit weiterhin die einzelnen Sichten gleichberechtigt behandelt. Erst während der Abbildung auf ein Präsentationsmodell werden die Sichten zusammengeführt, so dass die Repräsentationsbeschreibung sowohl auf „traditionelle“ multimediale Präsentationsmodelle als auch auf eine im Rahmen dieser Arbeit entwickelte verteilte Ablaufumgebung abgebildet werden kann.

Im Gegensatz zu den zurzeit verbreiteten softwarebasierten Präsentationsmedien, wird durch die in Kapitel 6 vorgestellte Ablaufumgebung die Trennung zwischen Rollen bzw. Sichten und der Organisation konsequent fortgeführt. Die Mittlerelemente bilden dabei die Organisationsstruktur einer multimedialen Anwendung, d.h. sie verwalten die multimedialen Sichten und fungieren somit als aktive vermittelnde Elemente in einer verteilten multimedialen Umgebung.

Die multimedialen Mittlerelemente werden in der Ablaufumgebung durch autonome mobile Software-Agenten repräsentiert, die die Koordination der multimedialen Sichten (Mediendaten, Layout, Verhalten) innerhalb einer verteilten Anwendung übernehmen und diese mit steuernden Modellen und externen Präsentationsmedien verbinden. Die Struktur einer verteilten Ablaufumgebung lässt sich dabei durch stationäre Agenten aufbauen, die spezielle Dienste für die unterschiedlichen Sichten zur Verfügung stellen.

Die Sichten, externen Präsentationsmedien und steuernden Modelle werden durch Komponenten innerhalb der Architektur repräsentiert. Durch den hohen Abstraktionsgrad komponentenorientierter Modelle kann die Funktionalität der Ablaufumgebung sowohl durch neue externe Ein- und Ausgabegeräte als auch durch die Unterstützung neuer steuernder Modelle erweitert werden. Die Ablaufumgebung stellt somit einen Rahmen für die Konfiguration unterschiedlicher multimedialer Anwendungen bereit, so dass sie als offenes Präsentationsmedium für die Realisierung von Wahrnehmungsumgebungen geeignet ist.

Innerhalb einer Wahrnehmungsumgebung treten die multimedialen Mittlerelemente als Vermittler zwischen der virtuellen und realen Umgebung auf und unterstützen einen fließenden Übergang zwischen beiden Umgebungen. Dies wird sowohl durch die räumlich verteilten, lokalen Ablaufumgebungen erreicht, zwischen denen die Mittlerelemente migrieren können, als auch durch die externen Präsentationsmedien, durch die „traditionelle“ Präsentations- und natürliche Interaktionsformen unterstützt werden. Wie in Kapitel 7 anhand von Beispielen gezeigt wird, können derartige multimediale Umgebungen in den Bereichen Edutainment, Infotainment, der Medienkunst und der Simulation eingesetzt werden, die sich in jüngster Zeit gerade durch die Gestaltung multimedialer virtueller und realer Umgebungen auszeichnen.

Es bleibt sicherlich abzuwarten, welche charakteristischen Ausdrucksformen durch das neue Kommunikationsmedium Computer noch entstehen werden. Das in dieser Arbeit vorgestellte Konzept kann vor diesem Hintergrund als flexibler Ansatz für den experimentellen Einsatz angesehen werden, ohne dabei auf einen zusammenhaltenden Rahmen verzichten zu müssen, der durch die multimedialen Mittlerelemente gebildet wird.

Glossar

Ablaufumgebung: Spezielle Form eines Programms zur Realisierung multimedialer Anwendungen, auf dem unterschiedliche Präsentationen „abgespielt“ werden können. Im Gegensatz hierzu sind durch übersetzte Programme *Präsentationsmedium* und Daten miteinander verwoben. Vergleicht man beide Ansätze mit „traditionellen“ *Präsentationsmedien* (Diaprojektor, Videorekorder, etc.), kann eine interpretierende Ablaufumgebung ebenso als *Präsentationsmedium* angesehen werden.

Agent: siehe *Software-Agent*.

Anwendung: Als Anwendungen werden ablauffähige Programme verstanden, die die definierte Funktionalität erfüllen. *Multimediale Anwendungen* können dabei durch *Ablaufumgebungen* dargestellt werden, die ausschließlich Beschreibungssprachen interpretieren oder als übersetztes Programm eine fest definierte Anwendung realisieren.

Aufbaukategorie: Analysekriterium der Semiotik (siehe auch *Auswahlkategorie*, *Interaktionskategorie*), bei dem die Bedeutung eines Zeichens aufgrund der Beziehungen zu anderen vorkommenden Zeichen bestimmt wird. Die Zeichen stehen dabei durch ihre räumliche und zeitliche Anordnung in Beziehung. Aus Sicht des subjektorientierten Entwurfs wird die Aufbaukategorie durch die *Layoutsicht* widergespiegelt. Bei der Gestaltung von *Wahrnehmungsumgebungen* müssen hierbei neben der virtuellen Darstellung ebenso die räumlichen Beziehungen in der realen Umgebung des *Rezipienten* berücksichtigt werden.

Auswahlkategorie: Analysekriterium der Semiotik (siehe auch *Aufbaukategorie*, *Interaktionskategorie*), bei dem die Darstellungsart im Vordergrund steht. Die Bedeutung eines Zeichens wird dabei durch die Auswahl einer bestimmten Darstellung assoziiert, die von dem *Rezipienten*, bewusst oder unbewusst, mit anderen möglichen Darstellungen verglichen wird. Man spricht daher von so genannten Auswahlkategorien eines Zeichensystems. Aus Sicht des subjektorientierten Entwurfs wird die Auswahlkategorie durch die *Mediensicht* widergespiegelt.

Benutzer: Im Gegensatz zu dem Begriff *Rezipient* werden durch den Begriff Benutzer die Möglichkeiten der *Interaktion* und Beeinflussung einer *multimedialen Anwendung* durch einen Betrachter betont.

Bottom-Up-Entwurf: Ausgehend von speziellen Lösungen wird durch Generalisierung schrittweise der Abstraktionsgrad der Problemlösung erhöht (invers zu *Top-Down-Entwurf*) [Burk99].

Datenkapselung: Vereinbarung von Daten und den Methoden in einem *Software-Element*, das diese Daten verwaltet. Die Zugriffe auf die Daten sind einschränkbar und nur über definierte Zugriffsmethoden möglich [Burk99].

Definitionsphase: Phase des subjektorientierten Entwurfs. Während dieser Phase wird eine multimediale Anwendung aus unterschiedlichen Sichten (*Mediensicht*, *Layoutsicht*, *Verhaltenssicht*) in einer präsentationsneutralen Sprache beschrieben. Die Beschreibung baut dabei auf dem *Manipulations-* und *Repräsentationsmodell* einer *Dokumentenarchitektur* auf.

Denotative Bedeutung: Bei der Interpretation der Zeichen findet ein dynamischer Prozess statt, bei dem den Zeichen eine direkte Bedeutung zugeordnet werden kann. Im Fall eines Passbildes besteht z.B. eine denotative Bedeutung, wohingegen eine schriftliche Beschreibung der abgebildeten Person als eher *konnotativ* zu bezeichnen ist.

Dokumentenarchitektur: Durch eine Dokumentenarchitektur wird die Syntax und Semantik von Beschreibungssprachen festgelegt, so dass der Übergang zwischen der Beschreibung einzelner Aspekte während der Erstellung eines Dokuments und der eigentlichen Realisierung durch eine Architektur definiert wird.

Bei einer vollständig definierten Dokumentenarchitektur durchläuft ein Dokument von der Erzeugung der Inhalte über die Strukturierung bis zur Präsentation unterschiedliche Phasen. Die Phasen sind dabei unterschiedlichen Beschreibungsformen (*Manipulations-*, *Repräsentations-* und *Präsentationsmodell*) zuzuordnen.

Entwurf: Während des Entwurfs einer *multimedialen Anwendung* wird die Anwendung durch eine präsentationsneutrale Beschreibung definiert. Da durch den Entwurf zunächst die eigentliche technische Realisierung nicht betrachtet wird, wird somit die *gestalterische Sicht* auf eine *multimediale Anwendung* betont.

Entwicklung: Bei der Entwicklung einer *multimedialen Anwendung* wird der Aspekt der Realisierung eines ablauffähigen Programms betont. Somit wird bei der Entwicklung die *technische Sicht* auf eine *multimediale Anwendung* betont. Je nach Art des zu realisierenden Programms handelt es sich um die Implementierung einer Software bzw. um die Umsetzung in eine präsentationsabhängige Sprache (*Realisierungsphase*).

Ereignis: Allgemein betrachtet stößt ein Ereignis die Zustandsänderung eines *Software-Elements* an. Dabei kann zwischen internen Ereignissen eines *Software-Elements* und externen Ereignissen zwischen *Software-Elementen* unterschieden werden. Durch externe Ereignisse werden zwischen *Software-Elementen* Nachrichten ausgetauscht (siehe *semantisches Ereignis*).

Gestalterische Sicht: Für die gestalterische Sicht auf eine *multimediale Anwendung* ist die Sicht des Benutzers charakteristisch. Im Gegensatz zu der *technischen Sicht* steht somit das Gesamterscheinungsbild einer *multimedialen Anwendung* im Vordergrund, durch die der *semantische Raum* bestimmt wird. Somit handelt es sich eher um einen Ansatz, der zu einem *mentalen Konzept* führt.

Horizontal offene Ablaufumgebung: Durch die Interpretation von *Ablaufumgebungen* als *Präsentationsmedium* kann mithilfe dieser in einer *verteilten Anwendung* eine zusammenhängende Präsentation gestaltet werden. Als horizontal offene Ablaufumgebungen werden somit *Präsentationsmedien* bezeichnet, die die Integration mehrerer lokaler Umgebungen unterstützen, die von dem Rezipienten als zusammenhängende *Anwendung* interpretiert werden.

Hypermedia-Anwendungen: Hypermedia-Anwendungen vereinen die Eigenschaften von *Hypertext-Anwendungen* und *multimedialen Präsentationen*, d.h. es werden sowohl zeitdiskrete als auch zeitkontinuierliche Mediendaten eingesetzt. Neben der *multimedialen Präsentation* realisieren Hypermedia-Anwendungen *Interaktionsbeziehungen*, so dass der *Benutzer* die Präsentation beeinflussen kann.

Hypertext-Anwendungen: In diesen Anwendungen kommen lediglich zeitdiskrete Mediendaten wie Text und Bild zum Einsatz, die in den meisten Fällen durch komplexe Medienverwalter als „Seiten" bzw. Dokumente für den Benutzer repräsentiert werden. Die *Interaktionsbeziehungen* beschreiben die Verbindungen, die so genannten „Hyperlinks", zwischen den einzelnen Dokumenten. Durch eine derart verknüpfte (Dokumenten-) Struktur kann der *Benutzer* somit zwischen den Dokumenten navigieren.

Implementierung: Die Implementierung ist die systematische Programmierung einzelner *Software-Elemente* auf Grundlage eines spezifizierten Entwurfs. Damit ist die Implementierung die Umsetzung des *Entwurfs*.

Integrationskonzept: Integrationskonzepte verbinden die *technische* und *gestalterische Sicht* auf *multimediale Anwendungen*, wobei sie alle Phasen und *Sichten* auf das Gebiet berücksichtigen. Wo die *technische Sicht* zu einem *strukturierten* (analytischen) und die *gestalterische Sicht* zu einem *mentalen Konzept* führen, wird durch Integrationskonzepte das Zusammenspiel beider *Sichten* definiert.

Interaktion: Interaktion beschreibt den Vorgang der Beeinflussung einer *multimedialen Anwendung* durch einen *Benutzer*. Zwischenmenschlich findet Interaktion zwischen Personen bzw. Gruppen durch verbale oder nichtverbale *Kommunikation* statt, so dass zunehmend auch durch *multimediale Anwendungen* unterschiedliche Interaktionsformen angeboten werden. Hierbei spielt die *Schnittstelle* zwischen realer Umgebung und *multimedialer Anwendung* eine zentrale Rolle.

Interaktionsbeziehung: Durch Interaktionsbeziehungen werden die Möglichkeiten der Beeinflussung einer *multimedialen Anwendung* funktional beschrieben. Durch die Beschreibung werden dabei *Ereignisse*, die durch einen *Benutzer* ausgelöst werden können, auf Aktionen innerhalb der *multimedialen Anwendung* abgebildet.

Interaktionskategorie: Analysekriterium der Semiotik (siehe auch *Aufbaukategorie*, *Auswahlkategorie*), bei dem die Interaktion als Metazeichen interpretiert wird. Die Bedeutung eines Zeichens wird hier durch die tatsächliche „programmierte" Beziehung zu anderen Zeichen repräsentiert. Aus Sicht des subjektorientierten Entwurfs wird die Interaktionskategorie durch die *Verhaltenssicht* widergespiegelt.

Ikon: Zeichen, dessen Beziehung zum Signifikat auf einem Abbildungsverhältnis, d.h. auf Ähnlichkeiten, beruht. Dies betrifft z.B. optische Ähnlichkeiten bei Piktogrammen etc. und akustische Ähnlichkeiten bei Geräuschen.

Index: Zeichen, die in einem Folge-Verhältnis zum Signifikat oder Gemeinten stehen und somit Rückschlüsse auf etwas anderes, wie z.B. Ursache oder Grund, zulassen. So kann etwa eine Abbildung einer lachenden Person als indexikalisches Zeichen für Freude stehen oder ein Dialekt bzw. die Stimmqualität Rückschlüsse auf eine bestimmte Herkunft bzw. auf das Alter eines Sprechers ermöglichen.

Kanal: Der Kanal eines Mediums bildet die Grundlage für den Austausch von Informationen bzw. Daten und kann somit als Verbindung zwischen Sender und Empfänger gesehen werden. Ein solcher Kanal nimmt die zu übermittelnden Informationen auf und wird daher häufig auch als Trägermedium bezeichnet [Schm98]. Differenziert man die Übermittlung der Information genauer, unterscheidet man zwischen *Repräsentations-*, *Speicher-*, *Übertragungs-* und *Präsentationsmedium* [Stei99].

Klasse: Eine Klasse ist die typisierte Beschreibung einer Instanziierungsvorschrift für *Objekte*. Durch die Klassenbeschreibung findet typischerweise eine *Datenkapselung* innerhalb eines *Software-Elements* statt [Burk99].

Kommunikation: Eine Kommunikation wird als Übermittlung einer Nachricht zwischen Sender und Empfänger beschrieben, die eine Reaktion des Empfängers bewirkt [Wa-Hi00].

Kommunikationsmedium: Als Kommunikationsmedium wird ein physikalischer oder technischer *Kanal* verstanden, der die Übermittlung einer Nachricht unterstützt. Eine weiterreichende Betrachtung eines Kommunikationsmediums umfasst neben dem physikalischen bzw. technischen *Kanal*, die *Rollen* und *Protokolle* zwischen den *Rollen*, sowie die *Syntax* und den *semantischen Raum* eines Kommunikationsmediums [Schm98].

Komponente: Der Begriff Komponente ist innerhalb der Softwaretechnik eingeführt und wird dort eher als abstrakter Begriff auf *Software-Elemente* verwendet. Dabei wird durch die Art der Komponentenmodellierung das Angebot eines Dienstes betont, den andere *Software-Elemente* in Anspruch nehmen können. Durch das Zusammenwirken einzelner Komponenten können neue Dienste erstellt werden, ohne die Komponenten neu entwerfen zu müssen. Um diese Sichtweise zu ermöglichen, werden bei der Modellierung und Realisierung von Komponenten die Aspekte der Abgrenzbarkeit, Eigenständigkeit, Integrationsfähigkeit und Anpassungsfähigkeit betont [Grif98].

Konnotative Bedeutung: Bei der Interpretation eines Zeichens findet ein dynamischer Prozess statt, bei dem dem Zeichen eine assoziierte Bedeutung zugeordnet wird. Im Gegensatz zu einer *denotativen* (direkten) Bedeutung beruht die konnotative (assoziierte) Bedeutung auf bekannten/vergleichbaren Modellen, die z.B. auf kulturellen Gegebenheiten beruhen.

Kopplung: Als Kopplung wird die Bindung zwischen *Software-Elementen* verstanden. Eine Bindung zwischen *Software-Elementen* wird durch die Referenzen auf die Elemente und die möglichen Zugriffsmöglichkeiten auf ein *Software-Element* beschrieben. Dabei kann man grundsätzlich zwischen enger und loser Kopplung zwischen *Software-Elementen* unterscheiden.

Layoutsicht: Die Layoutsicht spiegelt die rollenspezifische Sicht des Graphikdesigners bzw. 3D-Modellierers während des subjektorientierten Entwurfs wider. Durch die Beschreibung lassen sich sowohl die Anordnung und das Erscheinungsbild einzelner

Mediendaten in der virtuellen Umgebung definieren als auch das Erscheinungsbild der realen Umgebung modellieren. Somit werden durch die Layoutsicht die Methoden für die Umsetzung der *Aufbaukategorie* einer *multimedialen Anwendung* bzw. *Wahrnehmungsumgebung* zur Verfügung gestellt (siehe auch *Mediensicht*, *Verhaltenssicht*).

Manipulationsmodell: Das Manipulationsmodell ist ein Teilmodell einer *Dokumentenarchitektur*. Es unterstützt die Erzeugung neuer und die Änderung bereits bestehender Dokumente, d.h. das Manipulationsmodell beschreibt den Umgang mit den Dokumenten durch entsprechende Programme, die während der *Definitionsphase* eingesetzt werden können.

Mediensicht: Durch die Mediensicht können den *multimedialen Mittlerelementen* während des subjektorientierten Entwurfs einzelne Mediendaten zugeordnet werden. Dabei werden die gängigen Programme für die Bearbeitung von Mediendaten unterstützt, so dass die Mediensicht die rollenspezifischen Arbeitsweisen von Medientechnikern widerspiegelt. Durch die Zuordnung der Mediendaten zu den *Mittlerelementen* werden somit die Methoden zur Umsetzung der *Auswahlkategorien* zur Verfügung gestellt.

Mentales Konzept: Bei der Erstellung eines mentalen Konzepts führen assoziierte Modelle zu der Konzeptbeschreibung. Im Gegensatz zu dem *strukturorientierten Konzept* wird durch einen derartigen Ansatz somit das Gesamterscheinungsbild einer *Anwendung* durch ein zu Grunde liegendes Modell bestimmt.

Metonymie: Zeichen, dessen Beziehung zum Signifikat sich durch ein assoziiertes Detail oder eine assoziierte Vorstellung aufbaut. So kann in der Literatur vom König (und der Idee des Königtums) als der Krone gesprochen werden.

Mittlerelement: siehe *multimediales Mittlerelement*.

Modellierung: Als Modellierung wird der Prozess der Modellbildung bezeichnet. Aus *technischer Sicht* versteht man dabei die Anwendung eines Entwurfsparadigmas aus der Softwaretechnik. Somit werden in Abhängigkeit des Entwurfsparadigmas unterschiedliche Werkzeuge, Methoden und Beschreibungsformen verwendet [Burk99].

Aus *gestalterischer Sicht* wird während der Modellierung ein *mentales Konzept* entwickelt. Hierbei unterscheidet man zwischen bereits bestehenden assoziierten Modellen oder neu entwickelten Modellen auf Basis neuer Erkenntnisse.

Modul: Im Gegensatz zu *Klassen* stellen Module nicht instanziierbare Kapselungseinheiten dar, die bevorzugt während des Entwurfs eingesetzt werden. Charakteristisch ist die Bedeutung von Modulen für die Wiederverwendung verfügbarer Lösungen und die für die Benutzung der Lösung notwendigen Abhängigkeiten [Burk99].

Multimedia: Oberbegriff für neue Medien, interaktive Medien, virtuelle Realitäten, simulierte Welten etc. Je nach Sichtweise ergeben sich unterschiedliche Definitionen.

Aus *technischer Sicht* bezeichnet der Begriff Multimedia das Zusammenspiel zeitkonstanter und zeitkontinuierlicher Medien. Aus *gestalterischer Sicht* steht der Begriff Multimedia für neue Ausdrucksformen elektronischer Medien, die durch das Zusammenspiel von *Auswahl-*, *Aufbau-* und *Interaktionskategorie* gestaltet werden.

Multimediale Anwendung: Multimediale Anwendungen sind Programme, die Charakteristika von *multimedialen Präsentationen* bzw. von *Hypermedia-Anwendungen* aufweisen.

Multimediales Mittlerelement: Der Begriff multimediales Mittlerelement ist dem Begriff der *Mittlerobjekte*, die als Metapher die Verbindung zwischen der künstlerischen und wissenschaftlichen Modellierung beschreiben und somit die anwendungsorientierte Realisierung betonen [Bec98], entlehnt. Im Gegensatz zu Mittlerobjekten stellen multimediale Mittlerelemente als integrierendes Modell ein durchgängiges Konzept für den Entwurfsprozess, die Modellierung eines *Präsentationsmediums* und die Umsetzung von *Wahrnehmungsumgebungen* bereit.

Multimediale Präsentation: In multimedialen Präsentationen werden die verschiedenen Medientypen (Text, Bild, Bewegtbild, Ton, 3D-Welten, ...) integriert. Mindestvoraussetzung ist dabei das Zusammenwirken eines zeitdiskreten und eines zeitkontinuierlichen Medientyps. Multimediale Präsentationen realisieren allerdings keine *Interaktionsbeziehung*, so dass keine Steuerung der Präsentation durch den *(Be-)Nutzer* möglich ist.

Nutzer: siehe *Benutzer*.

Objekt: Ein Objekt ist eine Instanz einer *Klasse*, das zur Laufzeit existiert und Speicherplatz belegt. Die instanziierten Objekte einer *Klasse* unterscheiden sich durch ihre unterschiedlichen Speicherbereiche [Burk99].

Perzeptionsmedium: Als Perzeptionsmedium wird die Charakterisierung eines *Kommunikationsmediums* durch die Art der Einbeziehung menschlicher Sinne bezeichnet [Stei99]. Hierbei spielen neben den typischen auditiven und visuellen Medien bei *multimedialen Anwendungen* für die Gestaltung von *Wahrnehmungsumgebungen* zunehmend haptische und indirekte Interaktionsformen eine wesentliche Rolle.

Pragmatischer Projektansatz: Der pragmatische Projektansatz verbindet Aspekte der *unstrukturierten* und *strukturierten Projektansätze*. Es werden hierbei ausschließlich einzelne Zeitspannen vorgegeben, innerhalb derer einzelne Aufgaben gelöst werden. Durch die flexible Definition einzelner Zeitspannen können diese innerhalb des Projektablaufs einfacher als bei *strukturierten Projektansätzen* verschoben werden [Brow00].

Präsentationsmedium: Als Präsentationsmedium werden alle Hilfsmittel und Geräte für die Ein- und Ausgabe von Informationen bezeichnet. Hier wird primär eine Unterscheidung nach Ausgabe- (Bildschirm, Lautsprecher, etc.) und Eingabemedien (Tastatur, Maus, etc.) getroffen [Stei99].

Präsentationsmodell: Das Präsentationsmodell ist ein Teilmodell einer *Dokumentenarchitektur*. Es legt die Beschreibungsform des zu präsentierenden bzw. auszuführenden Inhalts und dessen Struktur fest. Hierfür wird durch das Präsentationsmodell die explizite (syntaktische) Struktur in eine durch das *Präsentationsmedium* bestimmte implizite (semantische) Wahrnehmung überführt.

Produktion: Die Produktion multimedialer Anwendungen schließt den Entwurf und die Entwicklung der Anwendung ein. Dabei werden häufig weitere Aspekte, die über die in dieser Arbeit betrachteten Aspekte hinausgehen, betrachtet. Insbesondere sind hierbei wirtschaftliche und logistische Aspekte zu beachten.

Protokoll: Im Sinne eines *Kommunikationsmediums* wird durch das Protokoll eines Mediums das Zusammenspiel der einzelnen *Rollen* beschrieben, so dass eine Ablauforganisation festgelegt wird [Schm98].

Realisierungsphase: Während der Realisierungsphase des subjektorientierten Entwurfs wird die präsentationsneutrale Beschreibung der *Definitionsphase* in eine präsentationsabhängige Beschreibung umgeformt. Die Umformung kann dabei durch einen Regelsatz automatisch erfolgen.

Rezipient: Als Rezipient wird der Betrachter einer *multimedialen Anwendung* bezeichnet, der die Anwendung ausschließlich betrachtet oder aktiv beeinflusst (vergleiche *Benutzer*).

Repräsentationsmedium: Ein Repräsentationsmedium ist durch die unterschiedlichen Formate der Daten gekennzeichnet. So werden durch ein Repräsentationsmedium z.B. die unterschiedlichen rechnerinternen Bild- und Textformate beschrieben, die für die Darstellungen der Daten verwendet werden [Stei99].

Repräsentationsmodell: Das Repräsentationsmodell ist ein Teilmodell einer *Dokumentenarchitektur*. Durch ein Repräsentationsmodell wird ein Austauschformat für Dokumente definiert. Durch die Standardisierung von Austauschformaten werden die einzelnen Produktionsphasen zwischen Entwurf und Realisierung miteinander verbunden.

Rolle: Durch ein Kommunikationsmedium werden die Akteure, die an diesem Medium partizipieren, definiert. Neben dem Sender und Empfänger eines Mediums sind dies im Falle der traditionellen Printmedien beispielsweise Autoren, Redakteure, Graphikdesigner, Händler, etc. Die Aufgabenprofile der Akteure werden dabei als Rollen bezeichnet. Die Rollen beschreiben somit den organisatorischen Aspekt eines Mediums [Schm98].

Schnittstelle: Aus *gestalterischer Sicht* beschreibt eine Schnittstelle die Möglichkeiten und Formen der *Interaktion* zwischen *Benutzer* und *Anwendung*. Hierbei spielen neben den funktionalen Möglichkeiten der Beeinflussung insbesondere die angebotenen Eingabemedien eine zentrale Rolle für die Gestaltung von *Wahrnehmungsumgebungen*.

In der Softwaretechnik wird (aus *technischer Sicht*) eine Schnittstelle als von außen sichtbare Beschreibung eines *Software-Elements* oder einer *Anwendung* bezeichnet. Sie stellt dabei die veröffentlichten Zugriffsmöglichkeiten eines *Software-Elements* oder einer *Anwendung* zur Verfügung.

Semantischer Raum: Die *Kommunikation*, die durch ein Medium geschieht, erfordert eine Interpretation der übermittelten Inhalte durch den Empfänger, die mit der Intention des Senders übereinstimmen sollte. Erst dadurch kann erreicht werden, dass eine erfolgreiche *Kommunikation* stattfindet. Man spricht in diesem Fall von der Semantik der Inhalte. Die Semantik ist weder allein im *Kanal* noch in der *Syntax* enthalten. Sie definiert eine übergeordnete Ebene, die die Referenz zur externen Welt bzw. die Interpretation der einzelnen Inhalte ermöglicht. Man spricht daher von dem semantischen Raum bzw. der semiotischen Bedeutung eines Mediums [Schm98].

Semantisches Ereignis: Semantische Ereignisse sind eine spezielle Art von *Ereignissen*, die als Nachrichten zwischen *Software-Elementen* ausgetauscht werden. Um die gegenseitige Verständigung der *Software-Elemente* zu unterstützen, steht dabei ein festgelegtes Vokabular an Ereignissen zur Verfügung, durch das, neben der Bezeichnung des Ereignistyps und der Kennung des Empfängers, Attribute übergeben werden können. Die Interpretation eines semantischen Ereignisses findet innerhalb des Empfängers statt.

Sicht: Ausschnitt bzw. spezieller Blickwinkel auf ein Element bzw. eine Anwendung. Im Gegensatz zu der Modellierung von *Objekten* werden durch Sichten ausschließlich einzelne Aspekte betrachtet. Dabei werden die hier eingeführten Sichten (*Mediensicht*,

Layoutsicht, *Verhaltenssicht*) auf *multimediale Mittlerelemente* durch rollenspezifische Arbeitsweisen charakterisiert.

Software-Agent: Als Software-Agenten werden, vergleichbar mit *Komponenten*, anpassungsfähige, eigenständige und abgrenzbare *Software-Elemente* bezeichnet. Jedoch wird bei der Entwicklung von Agenten dieses Paradigma spezialisiert, indem sowohl die Interaktion mit bzw. zwischen Agenten als auch das Verhalten charakterisiert werden. Grundlage des Paradigmas ist die Zuordnung von „menschlichen" Eigenschaften, die das interne und externe Verhalten von Agenten beschreiben [Bate94] [Shoh97a].

Der Aufbau eines Agenten lässt sich dabei in zwei Bereiche gliedern. Der so genannte mentale Bereich beschreibt den Zustand [Shoh97a] [GrCl98] [WoJe95a] [WoJe95b] und das Verhalten [Broo90] [Broo91] [GrCl98] eines Agenten. Durch die Schnittstelle, die als Abstraktion einer Komponentenschnittstelle zu betrachten ist, werden die Kommunikationseigenschaften eines Agenten beschrieben [FiLM97] [Shoh97b] [FIPA97].

Software-Architektur: Unter einer Software-Architektur werden die Aufteilung eines Softwaresystems in seine *Software-Elemente*, seine *Schnittstellen*, die Prozesse und Abhängigkeiten zwischen ihnen sowie die Organisationsformen der Software verstanden [RePo99].

Software-Element: Bezeichnung einer logischen Einheit eines Programms, die Daten und Methoden zusammenfasst. Je nach Ausprägung/Modellierung eines Software-Elements spricht man von *Modul*, *Klasse/Objekt*, *Komponente* bzw. *Software-Agent*.

Speichermedium: Als Speichermedien werden die unterschiedlichen Datenträger bezeichnet. Die Speicherung von Daten ist dabei nicht auf die in einem Computer verfügbaren Komponenten (Festplatte, Diskette, etc.) beschränkt, sondern kann auch analoge Datenträger (Papier, Mikrofilm, etc.) umfassen [Stei99].

Strukturierter Projektansatz: Durch einen strukturierten Ansatz ist der Projektverlauf bereits fest vorgegeben. Die einzelnen Arbeitsschritte werden dabei durch einen Projektleiter kontrolliert und bei Überschreiten der zeitlichen Planung entsprechende Gegenmaßnahmen getroffen (siehe auch *unstrukturierter* und *pragmatischer Projektansatz*) [Brow00].

Strukturorientiertes Konzept: Bei einem strukturorientierten Konzept wird eine *Anwendung* durch ihre zu Grunde liegende logische Struktur beschrieben, d.h. die Modellierung findet anhand wahrgenommener Beziehungen zwischen einzelnen Elementen statt, die bereits indirekt das Erscheinungsbild der *Anwendung* festlegen.

Symbol: Zeichen, dessen Beziehung zum Signifikat weder auf einem Folge-Verhältnis (*indexikalisches Zeichen*) noch auf Ähnlichkeit (*ikonisches Zeichen*), sondern auf (kulturellen) Vereinbarungen beruht. So sind z.B. Laut- und Schriftzeichen der menschlichen Sprache symbolische Zeichen.

Syntax: Die durch den *Kanal* übermittelten Inhalte werden durch eine Syntax strukturiert, wobei die Strukturierung syntaktischen Regeln unterliegt. Man spricht dabei häufig auch von einer eigenen Sprache, die charakteristisch für ein jeweiliges *Kommunikationsmedium* bzw. Genre eines Mediums ist [ToCa99].

Technische Sicht: Durch die technische Sicht werden die zur Verfügung stehenden Elemente betrachtet, die für den Entwurf *multimedialer Anwendungen* verwendet werden. Somit handelt es sich um einen konstruktiven Ansatz, durch den *multimediale Anwendungen* aufgrund der technischen Eigenschaften eines Systems aufgebaut werden.

Top-Down-Entwurf: Ausgehend von einer abstrakten Anwendungsbeschreibung wird eine schrittweise Verfeinerung in Teilanwendungen durchgeführt, bis ein für die Implementierung ausreichend niedriges Abstraktionsniveau erreicht ist (invers zu *Bottom-Up-Entwurf*) [Burk99].

Trägermedium: siehe *Kanal*.

Übertragungsmedium: Übertragungsmedien unterstützen im Gegensatz zu den *Speichermedien* eine kontinuierliche Übertragung der Daten, wie etwa über Koaxialkabel oder Funkverbindungen [Stei99].

Unstrukturierter Projektansatz: Als unstrukturierter Projektansatz wird ein nicht vorgeplanter Projektverlauf verstanden. Im Gegensatz zu einem *strukturierten Projektansatz* werden durch diesen Ansatz unterschiedliche Wege während eines Projekts unterstützt. Hierdurch steht eher das Erlernen neuer Arbeitszusammenhänge als das konsequente Erreichen der Ziele im Vordergrund (siehe auch *strukturierter* und *pragmatischer Projektansatz* [Brow00].

Verhaltenssicht: Durch die Verhaltenssicht lassen sich die *multimedialen Mittlerelemente* während des subjektorientierten Entwurfs funktional beschreiben, so dass diese Sicht der rollenspezifischen Arbeitsweise der Informatiker bzw. der Programmierer entspricht. Durch die Beschreibung werden dabei sowohl das interne Verhalten der *Mittlerelemente* als auch die Interaktionsmöglichkeiten definiert. Somit werden durch die Verhaltenssicht die Methoden zur Umsetzung der *Interaktionskategorie* zur Verfügung gestellt.

Verteilte Anwendung: Eine verteilte Anwendung ist aus einer Menge autonomer lokaler Anwendungen aufgebaut, deren Kommunikation ausschließlich auf dem Austausch von Nachrichten über ein Netzwerk beruht. Diese Forderung schließt somit die *Kommunikation* über gemeinsame Speicherbereiche ausdrücklich aus.

Eine verteilte Anwendung hat keinen festzustellenden globalen Zustand, da die *Kommunikation* zwischen den lokalen Systemen Vermittlungszeiten beinhaltet und durch die Struktur kein gemeinsamer Speicherbereich zur Verfügung steht. Die einzelnen autonomen Knoten einer verteilten Anwendung definieren dabei die Gesamtfunktionalität durch zielgerichtete Kooperation [Grif98].

Verteilte multimediale Anwendung: Eine verteilte multimediale Anwendung verbindet die Eigenschaften *verteilter Anwendungen* und *multimedialer Anwendungen*. Insbesondere wird hierbei die *horizontal* und *vertikal offene* Gestaltung eines *Präsentationsmediums* vorausgesetzt, so dass durch den Begriff die *technische Sicht* auf eine *Wahrnehmungsumgebung* betont wird.

Vertikal offene Ablaufumgebung: Betrachtet man eine *Ablaufumgebung*, kann man zwischen der gewöhnlichen internen Wiedergabe der Mediendaten und externen *Präsentationsmedien* unterscheiden. Als vertikal offene Ablaufumgebungen werden dabei lokale Umgebungen bezeichnet, die die Integration unterschiedlicher *Präsentationsmedien* unterstützen, so dass neben der internen Wiedergabe externe Ein- und Ausgabegeräte eingebunden werden können.

Wahrnehmungsumgebung: Durch den Begriff einer Wahrnehmungsumgebung wird neben der eher *technischen Sicht* auf *verteilte multimediale Anwendungen* die *gestalterische Sicht* betont. Hierbei werden unter einer Wahrnehmungsumgebung nicht ausschließlich die einzelnen (internen, externen) *Präsentationsmedien* berücksichtigt, sondern es wird vielmehr das Zusammenspiel zwischen *multimedialer Anwendung* und realer Umgebung, in der sich der *Rezipient* befindet, betrachtet [WiKD98].

Literatur

[ABFP99] S. Abeck, J. Batlogg, D. Feuerhelm, D. Pracht
"Companion" - eine multimediale, webbasierte Lehr- und Lernumgebung
In: *Systemarchitektur auf dem Weg ins 3. Jahrtausend: Neue Strukturen, Konzepte, Verfahren und Bewertungsmethoden*, VDE Verlag, 1999, S. 271 - 277

[ABJL98] R. Aylett, F. Brazier, N. Jennings, M. Luck, H. Nwana, C. Preist
Agent Systems and Applications
In: *The Knowledge Engineering Review*, Vol. 13(3) 1998, S. 303 - 308

[Adam98] C. Adami
Introduction to Artificial Life
Springer-Verlag, 1998

[AlPr99] A. Alan, B. Pritsker
Simulation with Visual SLAM and AweSim
O'Reilly & Associates Inc., 1999

[amaz01] *amazon*
http://www.amazon.com
2001

[Andr76] J. D. Andrew
The Major Film Theories
Oxford University Press, 1976

[BaCM98] J. Bacon, K. Cagle, D. Miller
Director 6 Bible
IDG Books Worldwide Inc., 1998

[Bail99] B. P. Bailey
Interactive Sketching of Multimedia Storyboards
ACM Multimedia Doctoral Symposium, 1999, S. 205 - 206

[BaKC01a] B. P. Bailey, J. A. Konstan, J. V. Carlis
DEAMIS: Designing Multimedia Applications with Interactive Storyboards
ACM Multimedia, 2001
http://www-users.cs.umn.edu/~bailey/publications/acm-mm-2001.pdf

[BaKC01b] B. P. Bailey, J.A. Konstan, J.V. Carlis
Supporting Multimedia Designers: Towards More Effective Design Tools
Proceedings Multimedia Modeling, 2001
http://www-users.cs.umn.edu/~bailey/publications/mmm2001.pdf

[BaKM99] H. Baumeister, N. Koch, L. Mandel
Towards a UML extension for hypermedia design
UML´99 The Unified Modeling Language - Beyond the Standard, LNCS 1723,
Springer Verlag, 1999, S. 614 - 629

[BaKo00] B. P. Bailey, J. A. Konstan
Authoring Interactive Media
Encyclopedia of Electrical and Electronics Engineering,
John Wiley & Sons, 2000

[BaLa01] C. Barry, M. Lang
A Survey of Multimedia and Web Development Techniques and Methodology Usage
In: *IEEE Multimedia*, Vol. 8(3), S. 52 - 60

[BaLK97] G. Ball, D. Ling, D. Kurlander, et al.
Lifelike Computer Characters: The Persona Project at Microsoft
In: [Brad97a], S. 191 - 222

[Barr00] J. Barrileaux
3D User Interfaces With Java 3D
Manning Publications, 2000

[Bate94] J. Bates
The Role of Emotion in Believable Agents
In: *Communications of the ACM*, Vol. 37(4) 1994, S. 122 - 125

[Bec98] Louis Bec
Artificial Life under Tension - A lession in Epistemological Fabulation
In: [SoMi98], S. 92 - 98

[BeHo99] H. Beyer, K. Holtzblatt
Contextual Design : A Customer-Centred Approach to Systems Designs
Morgan Kaufmann Publishers, 1999

[BeMi98] H. Behme, S. Mintert
XML in der Praxis
Addison-Wesley, 1998

[BeSy00a] J. Berdux, M. Syrjakow
Subject-Oriented Design of Real and Virtual Multimedia Environments
18th IASTED International Conference on Applied Informatics (AI´2000),
2000, S. 323 - 329

[BeSy00b] J. Berdux, M. Syrjakow
Java Media Tool – A Framework for Real and Virtual Multimedia Environments
18th IASTED International Conference on Applied Informatics (AI´2000), 2000, S. 526 - 531

[BeSy00c] J. Berdux, M. Syrjakow
The Design of Alice's World
World Multiconference on Systematics, Cybernetics and Informatics (SCI´2000), Vol. 6(2) 2000, S. 79 - 84

[BeTh96] P. Becheiraz, D. Thalmann
A Model of Nonverbal Communication and Interpersonal Relationship between Virtual Actors
Computer Animation '96, IEEE Computer Society Press, 1996, S. 58 - 67

[BoDT99] E. Bonabeau, M. Dorigo, G. Theraulaz
Swarm Intelligence: From Natural to Artificial Systems
Oxford University Press, 1999

[Bole98] D. Boles
Begleitbuch zur Vorlesung Multimedia-Systeme
http://www-is.informatik.uni-oldenburg.de/~dibo/teaching/mm/buch/1998

[Boll98] S. Bollmann
Kursbuch Neue Medien
Rowohlt Taschenbuch Verlag, 1998

[Brad97a] J. M. Bradshaw, et al.
Software Agents
AAAI Press / The MIT Press, 1997

[Brad97b] J. M. Bradshaw
An Introduction to Software Agents
In: [Brad97a], S. 3 - 46

[Brad98] N. Bradley
The XML companion
Addison-Wesley, 1998

[Broc98] *Brockhaus – Die Enzyklopädie*
F. A. Brockhaus, 20. Auflage 1998

[Broo90] R. A. Brooks
Elephants Don't Play Chess
In: *Robotics and Autonomous Systems*, Vol. 6 1990, S. 3 - 15

[Broo91] R. A. Brooks
Intelligence without Representation
In: *Artificial Intelligence Journal*, Vol. 47 1991, S. 139 - 159

[Brow00] L. Brown
Integration Models: Templates for Business Transformation
SAMS-Markt&Technik Verlag, 2000

[Brya88] M. Bryan
SGML: An author's guide to the Standard Generalized Markup Language
Addison-Wesley, 1988

[BuMo93] M. Bunzel, S. K. Morris
Multimedia Applications Development Using Indeo Video and DVI Technology
McGraw-Hill, 1993

[Burk99] R. Burkhardt
UML – Unified Modeling Language: Objektorientierte Modellierung für die Praxis
Addison-Wesley, 1999

[Bush45] V. Bush
As We May Think
In: *The Atlantic Monthly*, No. 7 1945
http://www.isg.sfu.ca/~duchier/misc/vbush/

[CaBe97] R. Carey, G. Bell
The Annotated VRML 2.0 Reference Manual
Addison-Wesley Developers Press, 1997

[Carr00] J. M. Carroll
Making Use: Scenario-Based Design of Human-Computer Interactions
MIT Press, 2000

[Cata97a] *Çatal Höyük, als die Menschen begannen in Städten zu leben*
CD-ROM (Win/Mac)
Zentrum für Kunst und Medientechnologie, Karlsruhe, 1997

[Cata97b] *Çatal Höyük*
http://goethe.ira.uka.de/catal
1997

[CCCL94] L.M.F. Carneiro, M.H. Coffin, D. D. Cowan, C.J.P. Lucena
ADVcharts: A Visual Formalism for Highly Interactive Systems
In: *SIGCHI Bulletin*, Vol. 26(2) 1994, S. 74 - 77

[ChGo91] I.Charles F. Goldfarb,
Hytime: A standard for structured hypermedia interchange
In: *IEEE Computer*, Vol. 24(8) 1991, S.81 - 84

[CoLe90] P. R. Cohen, H. J. Levesque
Intention is choice with commitment
In: *Artificial Intelligence*, Vol. 42(3) 1990, S. 213 - 261

[DaDa96] L. Dayton, J. Davis
Insiderbuch Photoshop – Tips, Tricks & Techniken für die professionelle Arbeit mit Photoshop
Addison-Wesley, 1996

[DaDe97] J. Dale, D. DeRoure
A Mobile Agent Architecture for Distributed Information Management
International Workshop on the Virtual Multicomputer, 1997
http://www.bib.ecs.soton.ac.uk/data/1479/pdf/vim97.pdf

[Dale98] J. Dale
A Mobile Agent Architecture for Distributed Information Management
http://www.mmrg.ecs.soton.ac.uk/publications/papers/thesis98/thesis.pdf
1998

[DaWR95] J. Davies, R. Weeks, M. Revett
Jasper: Communicating Information Agents for WWW
4th International World Wide Web Conference, 1995, S. 473 - 482

[Delf99] D. Delfs
Entwicklung eines Multiagentensystems für reale und virtuelle Informationsräume
Diplomarbeit, Universität Karlsruhe, Inst. für Rechnerentwurf und Fehlertoleranz (D. Schmid), 1999

[DeRo97] D. DeRoure
Distributed Multimedia Information Systems
In: *IEEE Multimedia*, Vol. 4(4) 1997, S. 68 - 73

[DHDD96] D. DeRoure, W. Hall, H. Davis, J. Dale
Agents for Distributed Multimedia Information Management
1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, 1996, S. 91 - 102

[DiBr99] S. Dinkla, C. Brockhaus
Connected Cities - Kunstprozesse im urbanen Netz
Cantz-Verlag, 1999

[DIMG96] A. Díaz, T. Isakowitz, V. Maiorana, G. Gilabert
RMCase: A Tool To Design WWW Applications
In: *World Wide Web Journal*, Vol. 1(1) 1996, S. 559 - 566

[Dink97] S. Dinkla
Pioniere Interaktiver Kunst
Cantz-Verlag, 1997

[Dout95] K. Douté
Quark Xpress – Lösungen für Anwender
Rowohlt Taschenbuchverlag, 1995

[Dugu95] A.-M. Duguet
Führt Interaktivität zu neuen Definitionen in der Kunst?
In: [ScSh95], S. 36 - 41

[Eber00] H.-G. Eberhard
Toolbook II Instructor 7 – Interaktive Anwendungen für CD-ROM und Internet entwickeln
Addison-Wesley, 2000

[EbJa98] M. Eberl, J. Jacobsen
Macromedia Director 6 für Insider
SAMS Markt&Technik Verlag, 1998

[EiWe99] K. Eisenkolb, H. Weickardt
Das Einsteigerseminar Adobe Premiere 5
Kaarst-Büttgen, 1999

[Eric95] T. Erickson
Notes on Design Practice: Stories and Prototypes as Catalysts for Communication
In: *Scenario-Based Design: Envisioning Work and Technology in System Development*, Wiley, 1995, S. 37 - 58

[Eric96] T. Erickson
Design as Storytelling
In : *Interactions*, Vol. 3(4) 1996, S. 30 - 35

[FaKa97a] B. Falchuk, A. Karmouch
A mobile agent prototype for autonomous multimedia information access, interaction, and retrieval
Conference on Multimedia Modeling, 1997, S. 33 - 48

[FaKa97b] B. Falchuk, A. Karmouch
AgentsSys: A mobile agent system for digital media access and interaction on an internet
Globecom´97 Conference, 1997, S. 1876 - 1880

[Fein81] A. Feininger
Die hohe Schule der Fotografie
Heyne-Verlag, 1981

[Feld96] Tony Feldman
An Introduction to Digital Media
Routledge, 1996

[Fibi99] B. Fibiger
From Idea to Product
http://imv.au.dk/semiotics/semmm_uk.htm
1999

[FiLM97] T. Finin, Y. Labrou, J. Mayfield
KQML as an Agent Communication Language
In: [Brad97a], S. 291 - 316

[FIPA97] *FIPA 97 Specification Part 2 - Agent Communication Language*
http://www.cselt.it/fipa/spec/f7a12pdf.zip
1997

[FlFC99] D. Flanagan, J. Farley, W. Crawford
Java Enterprise in a Nutshell: A Desktop Quick Reference
O'Reilly & Associates Inc., 1999

[FrGr96] S. Franklin, A. Graesser
Is it an Agent, or Just a Program? A Taxonomy for Autonomous Agents
3rd International Workshop on Agent Theories, Architecture and Languages, Springer-Verlag, 1996, S. 21 - 35

[GaMP96] F. Garzotto, L. Mainetti, P. Paolini
Navigation in Hypermedia Applications: Modeling and Semantics
In: *Journal of organizational Computing and Electronic Commerce*, Vol. 6 (3) 1996, S. 211 – 238

[GaPS93] F. Garzotto, P. Paolini, D. Schwabe
HDM - A Model-based Approach to Hypermedia Application Design
In: *ACM Transactions on Information Systems*, Vol. 11 (1) 1993, S. 1 - 26

[GeKe94] M. R. Genesereth, S. P. Ketchpel
Software Agents
In: *Communications of the ACM (CACM)*, Vol. 37(7) 1994, S. 48 - 53

[GiAA95] D. Gilbert, M. Aparicio, B. Atkinson, et al.
IBM Intelligent Agent Strategy
IBM Corporation, 1995

[GiBe98] J. D. Gibson, T. Berger
Digital Compression for Multimedia: Principles and Standards
In: *Morgan Kaufmann Series in Multimedia Information and Systems*, Academic Press, 1998

[GMQG96] S. Giroux, P. Marcenac, J. Quinqueton, J. R. Grass
Modelling and Simulating Self-Organized Critical Systems
10th Annual European Simulation Multiconference (ESM´96), 1996, S. 1072 - 1076

[Goet00] F. Goetz
SMIL – Multimedia im Internet mit Realsystem G2
Addison-Wesley, 2000

[Gold94] C.F. Goldfarb
The SGML Handbook
Oxford University Press, 1994

[GrCl98] S. Grand, D. Cliff
Creatures: Entertainment Software Agents with Artificial Life
In: *Autonomous Agents and Multi-Agent Systems*, Vol. 1 1998, S. 39 - 57

[Grif98] F. Griffel
Componentware: Konzepte und Techniken eines Softwareparadigmas
dpunkt-Verlag, 1998

[Gump96] M. Gumpinger
AppleScript
Addison-Wesley, 1996

[HaCK95] C. G. Harrison, D. M. Chess, A. Kershenbaum
Mobile Agents: Are they a Good Idea?
IBM Research Report 19887, IBM Research Devision, 1995

[HaKK00] F. Harms, D. Koch, O. Kürten
Das große Buch HTML / XHTML und XML
DATA Becker, 2000

[Heid99] P. Heidkamp
Der Besucher als interaktiver Flaneur? oder Das Interface als Repräsentation digitaler Strukturen
In: *S.Schade, G. C. Tholen; Konfigurationen – Zwischen Kunst und Medien*
Wilhelm Fink Verlag, 1999, S. 426 - 435

[HeKo00] R. Hennicker, N. Koch
A UML-based Methodology for Hypermedia Design
3rd International Conference on the Unified Modeling Language: UML'2000, 2000, S. 410 - 424

[Hira99] M. Hirakawa
Do Software Engineers Like Multimedia?
IEEE International Conference on Multimedia Computing and Systems, 1999, S. 85 - 90

[HoKT93] U. Hoppe, M. Kretschmer, T. Teuber
Vorgehensmodelle für die Entwicklung von Teachware
Arbeitsbericht, Göttinger Wirtschaftsinformatik, Göttingen, 1993

[HoCh91] J. W. Hooper, R. O. Chester
Software Reuse - Guidelines and Methods
Plenum Press, 1991

[HRBG97] B. Hayes-Roth, L. Brownston, R. van Gent
Multiagent Collaboration in Directed Improvisation
In: *Readings in Agents*, Morgan Kaufmann Publishers, 1997, S. 141 - 147

[HRGe96] B. Hayes-Roth, R. van Gent
Improvisational puppets, actors, and avatars
Computer Game Developers' Conference, 1996
ftp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-96-09.ps

[Huht98] E. Huhtamo
Seeing at Distance – Towards an Archaeology of the "Small Screen"
In: [SoMi98], S. 262 - 278

[Hünn97] A. Hünnekens
Der bewegte Betrachter. Theorien der interaktiven Medienkunst
Wienand Verlag, 1997

[iShe01] iShell
http://www.tribeworks.com
2001

[Ishi97] S. Ishizaki
Multiagent Model of Dynamic Design
In: *Readings in Agents*, Morgan Kaufmann Publishers, 1997, S. 172 - 179

[IsKK96] T. Isakowitz, A. Kamis, M. Koufaris
Extending the Capabilities of RMM: Russian Dolls and Hypertext
31th Annual Hawaii International Conference on System Sciences, 1996, S. 177 - 186

[IsKK97] T. Isakowitz, A. Kamis, M. Koufaris
Reconciling Top-Down and Bottom-Up Design Approaches in RMM
Workshop in Information Systems and Technology (WITS´97), 1997
http://rmm-java.stern.nyu.edu/rmm/papers/RWITS97.pdf

[IsSB95] T. Isakowitz, E. A. Stohr, P. Balasubramanian
RMM: A Methodology for Structured Hypermedia Design
In: *Communications of the ACM*, Vol. 38 (8) 1995, S. 34 - 44

[Jell98] T. Jell
Component-Based Software Engineering
Cambridge University Press, 1998

[JeSW98] N. R. Jennings, K. Sycara, M. Wooldridge
A Roadmap of Agent Research and Development
In: *Autonomous Agents and Multi-Agent Systems*, Vol. 1 1998, S. 7 - 38

[JRSM98] W.L. Johnson, J. Rickel, R. Stiles, A. Munro
Integrating Pedagogical Agents into Virtual Environments
In: *Teleoperators and Virtual Environments*, Vol. 7(6) 1998, S. 523 - 546

[Kay90] A. Kay
User Interface: A Personal View
In: *The Art of Human-Computer Interface Design*, Addison-Wesley, 1990, S. 191 - 207

[Kerr98] M. Kerres
Multimediale und telemediale Lernumgebungen
Oldenburg Verlag, 1998

[Khaz95] C. D. Khazaeli
Crashkurs Typo und Layout
Rowohlt Taschenbuchverlag, 1995

[Kinn94] W. Kinnebrock
Marketing mit Multimedia. Neue Wege zum Kunden
Landsberg/Lech, 1994

[KKST79] R. Kimm, W. Koch, W. Simonsmeier, F. Tontsch
Einführung in Software Engineering
Walter de Gruyter, 1979

[KlBF97] H. Klotz, H. Bredekamp, U. Frohne, et al.
Contemporary Art - The Collection of the ZKM Center for Art and Media Karlsruhe
Prestel-Verlag, 1997

[Koch99] S. Koch
JavaScript - Einführung, Programmierung und Referenz
dpunkt-Verlag, 1999

[Kräm88] S. Krämer
Symbolische Maschinen - Die Idee der Formalisierung im geschichtlichen Abriß
Wissensch. BG., 1988

[Kräm98] S. Krämer
Medien, Computer, Realität. Wirklichkeitsvorstellungen und Neue Medien
Suhrkamp-Verlag, 1998

[Kreu99] *Die Kreuzzüge*
CD-ROM (Win)
Cornelson, 1999

[LaBa01] M. Lang, C. Barry
Techniques and Methodologies for Multimedia Systems Development: A Survey of Industrial Practice
In: *Realigning Research and Practice in Information Systems Development: The Social and Organizational Perspective*,
Kluwer Academic Publishers, 2001, S. 77 - 86

[LaLo95] S. M. Lang, P. C. Lockemann
Datenbankeinsatz
Springer-Verlag, 1995

[Land96] J. A. Landay
Interactive Sketching for the Early Stages of User Interface Design
Ph.D. Dissertation, Computer Science Department,
Carnegie Mellon University, 1996

[Lang96] D. B. Lange
An Object-Oriented Design Approach for Developing Hypermedia Information Systems
In: *Journal of Organizational Computing*, Vol. 6(3) 1996, S.269 - 293

[Lang97] C. G. Langton
Artificial Life: An Overview
Bradford Books, 1997

[Laur90] B. Laurel
Interface Agents: Metaphers with Character
In: *The Art of Human-Computer Interface Design*, Addison-Wesley, 1990,
S. 355 - 365

[LCKB97] J. C. Lester, S. A. Converse, S. E. Kahler, S. T. Barlow, B. A. Stone, R. S. Bhogal
The Persona Effect: Affective Impact of Animated Pedagogical Agents
Computer Human Interaction´97 - Human Factors in Computing Systems ACM, 1997, S. 359 - 366

[LeLY99] H. Lee, C. Lee, C. Yoo
A Scenario-Based Object-Oriented Hypermedia Design Methodology
Information & Management, 1999, S. 121 - 138

[LeSc98] H. Leopodseder, C. Schöpf
Cyberarts 98
Springer-Verlag, 1998

[Lévy90] P. Lévy
Die Methapher des Hypertextes
In: *C. Pias, J. Vogel, L. Engell, O. Fahle, B. Neitzel, Hrsg.; Kursbuch Medienkultur: Die maßgeblichen Theorien von Brecht bis Baudrillard*,
Deutsche Verlags-Anstalt, 1999, S. 525 - 528

[LKKC99] H. Lee, J. Kim, Y. G. Kim, S. H. Cho
A View-Based Hypermedia Design Methodology
In: *Journal of Database Management*, Vol. 10(2) 1999, S. 3 - 13

[LoIs95] I. Lokuge, S. Ishizaki
Geospace: An interactive visualization system for exploring complex information spaces
Computer Human Interaction ´95 (SIGCHI ACM), 1995, S. 409 - 414

[Lond95] B. London
Videoinstallationen: gestern/heute/morgen
In: [ScSh95], S. 98 - 101

[LöPü98] M. Löseke, D. Püttmann
Modellierung interaktiver virtueller Welten mit VRML
Studienarbeit, Universität Karlsruhe, Inst. für Rechnerentwurf und Fehlertoleranz (D. Schmid) 1998

[LyRS99] Fernando Lyardet, Gustavo Rossi, Daniel Schwabe
Discovering and Using Design Patterns in the WWW
In: *Multimedia Tools and Applications*, Vol. 8(3) 1999, S. 293 - 308

[Maes91] P. Maes
Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back
MIT Press, 1991

[Maes97] P. Maes
Agents that Reduce Work and Information Overload
In: [Brad97a], S. 145 - 164

[MaLF95] J. Mayfield, Y. Labrou, T. Finin
Desiderata for Agent Communication Languages
AAAI Symposium on Information Gathering from Heterogeneous,
Stanford University, 1995, S. 122 - 127

[MaLF96] J. Mayfield, Y. Labrou, T. Finin
Evaluation of KQML as an Agent Communication Language
In: *Intelligent Agents, Lecture Notes in Artificial Intelligence,* 1037,
Springer-Verlag, 1996, S. 347 - 360

[Mano95] L. Manovich
Kino und digitale Medien
In: [ScSh95], S. 42 - 48

[Marc98] P. Marcenac
Modeling MultiAgent Systems as Self-Organized Critical Systems
31th Hawaii International Conference on System Sciences HICSS-31, Vol. 5
1998, S. 86 – 95

[MaSt99] T. Maremaa, W. Stewart
Quick Time for Java : A Developer Reference
(The Quicktime Developer Series)
Morgan Kaufmann, 1999

[MaSu96] A. MacBride, J. Susser
Byte Guide to Opendoc
Osborne Publishing, 1996

[McCl95] F. G. McCabe, K. L. Clark
APRIL - Agent Process Interaction Language
In: *Intelligent Agents: Theories, Architectures and Languages, Lecture Notes in Artificial Intelligence*, No. 890, Springer-Verlag, 1995, S. 324 - 340

[McKe00] R. McKee
Story – Die Prinzipien des Drehbuchschreibens
Alexander-Verlag Berlin, 2000

[MeMe00] T. Meyer, C. Meyer
Creating Motion Graphics with After Effects
CMP-Books, 2000

[MoFi96] S. J. Morris, A. C. W. Finkelstein
Integrating Design and Development in the Production of Multimedia Documents
Internal Workshop on Multimedia Software Development, IEEE Computer Society Press, 1996, S. 98 - 10

[Mona00] J. Monaco
Film Verstehen
Rowohlt Taschenbuch Verlag, 2000

[Mons00] R. Monson-Haefel
Enterprise Javabeans
O'Reilly & Associates Inc., 2000

[Negr90] N. Negroponte
Hospital Corners
In: *The Art of Human-Computer Interface Design*, Addison-Wesley, 1990, S. 347 - 353

[Neil98] J. O'Neil
Java Beans Programming from the Ground Up
Osborne Publishing, 1998

[NiDa95] O. Nierstrasz, L. Dami
Component-Oriented Software Technology
In: *Object-Oriented Software Composition*, Prentice-Hall, 1995, S. 3 - 28

[NiMe94] O. Nierstrasz, T. D. Meijler
Requirements for a Composition Language
In: *P. Ciancarini, O. Nierstrasz, A. Yonezawa, Eds.; Object-Based Models and Languages for Concurrent Systems*, No. 924 July 1994, Springer-Verlag, S. 147 - 161

[Norm97] D. A. Norman
How Might People Interact with Agents
In: [Brad97a], S. 49 - 55

[Nwan96] H. S. Nwana
Software Agents: An Overview
In: *Knowledge Engineering Review*, Vol. 11(3) 1996, S. 205 - 244

[NwNd96] H. S. Nwana, D. Ndumu
An Introduction to Agent Technology
In: *British Telecom Technology Journal*, Vol. 14(4) 1996, S. 55 - 67

[Nykä97] O. Nykänen
Work in Progress: User Modeling in WWW with prerequisite graph model in Adaptive Systems and User Modeling on the World Wide Web
6th International Conference on User Modeling, 1997, S. 55 - 63

[ObWe98] T. Oberle, M. Wessner
Der Nürnberger Trichter – Computer machen Lernen leicht!?
Leuchtturm-Verlag, 1998

[OrHE96] R. Orfali, D. Harkey, J. Edwards
The Essential Distributed Object Survival Guide
John Wiley & Sons Inc., 1996

[Ott97] B. Ott
Grundlagen des beruflichen Lernens und Lehrens
Cornelsen Verlag, 1997

[Otto98] A. Otto
Realisierung eines Java-basierten Werkzeugs zur flexiblen Erstellung computerunterstützter Lernumgebungen
Diplomarbeit, Universität Karlsruhe, Inst. für Rechnerentwurf und Fehlertoleranz (D. Schmid), 1998

[PaSi94] B.-U. Pagel, H.-W. Six
Software Engineering: Die Phasen der Softwareentwicklung
Addison-Wesley, 1994

[Pazz99] K.-J. Pazzini
Zeige-Stöcke und andere Medien. Zur Aggressivität von Medien in der Bildung
In: *S.Schade, G. C. Tholen, Eds.; Konfigurationen – Zwischen Kunst und Medien*, Wilhelm Fink Verlag, 1999, S. 321 - 337

[Phys99] *Physikus*
CD-ROM (Win/Mac)
Heureka Klett, 1999

[Plog00] U. Plog
Totenmesse im Internet
In: *brand eins,* Nr. 5, 2000, S. 138 - 144

[Pütt00] D. Püttmann
Künstliches Leben – Simulation und Visualisierung durch ein verteiltes Multiagenten-System
Diplomarbeit, Universität Karlsruhe, Inst. für Rechnerentwurf und Fehlertoleranz (D. Schmid), 2000

[RBPE91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen
Object-Oriented Modeling and Design
Prentice-Hall Inc., 1991

[RePo99] P. Rechenberg, G. Pomberger
Informatik-Handbuch, 2. aktualisierte und erweiterte Ausgabe
Carl Hanser Verlag, 1999

[Reyn87] C. Reynolds
Flocks, Herds, and Schools: A Distributed Behavioral Model
In: *Computer Graphics (Siggraph'87)*, Vol. 21(4) 1987, S. 25 - 34

[RiJo97] J. Rickel, W.L. Johnson
Intelligent Tutoring in Virtual Reality: A Preliminary Report
8th World Conference on AI in Education, 1997, S. 294 - 301

[RoSG97] Gustavo Rossi, Daniel Schwabe, Alejandra Garrido
Design Reuse in Hypermedia Applications Development
8th ACM Conference on Hypertext (Hypertext´97), 1997, S. 57- 66

[SaEn99] S. Sauer, G. Engels
Extending UML for Modeling of Multimedia Applications
IEEE Symposium on Visual Languages, 1999, S. 80 - 87

[Sand97] M. Sandbothe
Digitale Verflechtungen. Medienphilosophische Untersuchungen zur Zeichentheorie des Internet
In: *K. Beck, G. Vowe, Eds.; Computernetze - ein Medium öffentlicher Kommunikation*, Spiess-Verlag, 1997, S. 125 - 137

[Sand00] M. Sandbothe
Grundpositionen zeitgenössischer Medienphilosophie und die pragmatische Theatralisierung unseres Zeichengebrauchs im Internet
In: *M. Leeker, Eds.; Interfaces - Interaktion - Performance. Zum Umgang mit digitaler Technik im Theater*, Alexander-Verlag Berlin, 2000, S. 23 - 31

[Sawh95] M. Sawhney
Entwicklung eines Vorgehensmodells für die Multimedia-Anwendungsentwicklung am Beispiel eines Informations- und Orientierungssystems für die Universität
Diplomarbeit, Universität Osnabrück, Fachgebiet Wirtschaftsinformatik, 1995

[SBSZ99] M. Syrjakow, J. Berdux, H. Szczerbicka, B. Zimmermann, A. Otto
A Flexible Java-Based Authoring System for Building Multimedia-Enriched CBT Applications
13th European Simulation Multiconference, Vol. 1 1999, S. 324 - 328

[ScGH98] R. S. Schifman, H. Günther, Y. Heinrich
Multimedia- Projektmanagement. Von der Idee zum Produkt
Springer Verlag, 1998

[Schm98] B. F. Schmid
Das Konzept des Wissensmediums
Arbeitsbericht, Universität St. Gallen, Institut für Medien und Kommunikationsmanagement, 1998
http://www.netacademy.org

[Schm99] S. Schmid-Isler
Neue Medien: Zu Rollen und Risiken einer digitalen Kunst
In: *Journal des Instituts für Versicherungswirtschaft*, IVW-HSG, Vol. 10 1999, (RiskVox 001/Oktober 99)

[Schö87] D. A. Schön
Educating the Reflective Practitioner
Jossey-Bass, 1987

[Schw95] H.-P. Schwarz
Das digitale Museum - Multimedialer Bildterror oder Chance zur Neubestimmung der Museen?
In: [ScSh95], S. 67 - 75

[Schw97] H.-P. Schwarz
Media Art History - Media Museum ZKM Center for Art and Media Karlsruhe
Prestel-Verlag, 1997

[Scot95] J. Scott
Wenn Erinnerungen, dann Multimedia-Phantasien
In: [ScSh95], S. 102 - 109

[ScRo98] D. Schwabe, G. Rossi
Developing Hypermedia Applications using OOHDM
Workshop on Hypermedia Development Processes, Methods and Models (Hypertext'98), 1998
http://www.eng.uts.edu.au/~dbl/HypDev/ht98w/Schwabe/schwabe.pdf

[ScSa00] J. Schöntauf, V. Sartorius
Illustrator 9
Galileo Press, 2000

[ScSh95] H.-P. Schwarz, J. Shaw, et al.
Perspektiven der Medienkunst - Media Art Perspectives
Cantz Verlag, 1995

[Shaw95] J. Shaw
Der entkörperte und wiederverkörperte Leib
In: *Kunstforum International,* 132, 1995, S. 168 - 171

[Shne97] B. Shneiderman
Direct Manipulation Versus Agents: Paths to Predictable, Controllable, and Comprehensible Interfaces
In: [Brad97a], S. 97 - 106

[Shoh97a] Y. Shoham
An Overview of Agent-Oriented Programming
In: [Brad97a], S. 3 - 46

[Shoh97b] Y. Shoham
Agent-oriented programming
In: *Readings in Agents*, Morgan Kaufmann Publishers, 1997, S. 329 - 349

[ShWa98] M. Shepherd, C. Watters
The Evolution of Cybergenres
31st Annual Hawaii International Conference on System Sciences, 1998,
S. 97 - 109

[Sied01] *Die Siedler 4*
CD-ROM (Win)
Blue Byte, 2001

[SmCS97] D. C. Smith, A. Cypher, J. Spohrer
KidSim: Programming Agents without Programming Language
In: [Brad97a], S. 165 - 190

[SoMa99] J. C. Soulié, P. Marcenac
Environmental Simulations using Multiagent Systems
IEEE International Conference on Information, Intelligence and Systems, 1999
http://www.univ-reunion.fr/~mas2/publications/articles/conferences/iciis99.pdf.gz

[SoMi98] C. Sommerer, L. Mignonneau
Art@Science
Springer Verlag, 1998

[Stei99] R. Steinmetz
Multimedia-Technologie – Grundlagen, Komponenten und Systeme
Springer-Verlag, 1999

[StLe97] B. A. Stone, J. C. Lester
Dynamical Sequencing an Animated Pedagogical Agent
In: *Readings in Agents*, Morgan Kaufmann Publishers, 1997, S. 156 - 163

[Terz98] D. Terzopoulos
Artificial Life for Computer Animation
In: [SoMi98], S. 69 - 77

[Tesf97] L. Tesfatsion
How Economists Can Get Alife
In: *The Economy as an Evolving Complex System II*,
Addison-Wesley, 1997, S. 533 - 564

[ThNH97] D. Thalmann, H. Noser, Z. Huang
Autonomous Virtual Actors based on Virtual Sensors
In: *Creating Personalities, Lecture Notes in Computer Science*, Springer-Verlag, 1997, S. 25 - 42

[ToCa99] E. G. Tomas, D. G. Campbell
Genre as Interface Metaphor: Exploiting Form and Function in Digital Environments
32nd Hawaii International Conference on System Science, 1999
http://dlib.computer.org/conferen/hicss/0001/pdf/00012008.pdf

[Uhrm96] A. M. Uhrmacher
Object-Oriented, Agent-Oriented Simulation: Implications for Social Science Applications
In: *Social Science Micro Simulation - A Challenge for Computer Science*,
Springer Lecture Notes in Economics and Mathematical Systems, 1996,
S. 432 - 447

[Uhrm97] A. M. Uhrmacher
Concepts of Object- and Agent-Oriented Simulation
In: *Transactions on SCS*, Vol. 14(2) 1997, S. 59 - 67

[UhSc98] A. M. Uhrmacher, B. Schattenberg
Agents in Discrete Event Simulation
ESS'98, Nottingham, 1998, S. 129 - 136

[Ulti01] Ultima Online
http://www.uo.com
2001

[Vels00] I. Velsz
3D Studio MAX R3 - Grundlagen und Praxis der 3D-Visualisierung und - Animation
Addison-Wesley, 2000

[Vest98] F. Vester
Denken, Lernen, Vergessen
Deutscher Taschenbuch Verlag, 1998

[Voya01] *Voyager*
Objectspace
http://www.objectspace.com

[vrml97] *VRML97 Specification*
http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm
1997

[WaCo97] P. Wavish, D. Conah
Virtual Actors that can Perform Scripts and Improvise Roles
1st International Conference on Autonomous Agents, ACM Press, 1997, S. 317 - 322

[WaGr96] P. Wavish, M. Graham
A Situated Action Approach to Implementing Characters in Computer Games
In: *Applied AI Journal*, Vol. 10(1) 1996, S. 53 - 74

[WaHi00] J. Watson, A. Hill
Dictionary of Media and Communication Studies (Arnold Student Reference)
Oxford University Press, 2000

[WaWe99] W. E. Walsh, M. P. Wellman
Efficiency and equilibrium in task allocation economies with hierarchical dependencies
16th International Joint Conference on Artificial Intelligence, 1999, S. 520 - 526

[Wats98] J. Watson
Media Communication: An Introduction to Theory and Process
MacMillan Press Ltd., 1998

[Weib98] P. Weibel
The Unreasonable Effectiveness of the Methodological Convergence of Art and Science
In: [SoMi98], S. 167 - 180

[Well96] M. P. Wellman
Market-Oriented Programming: Some Early Lessons
In: S. *Clearwater, Eds.; Market-Based Control: A Paradigm for Distributed Resource Allocation*, World Scientific Pub. Co., 1996
ftp://ftp.eecs.umich.edu/people/wellman/mbc95.ps.Z

[Wenz99] K. Wenz
Narrativität in Computerspielen
In: *S.Schade, G. C. Tholen, Eds.; Konfigurationen – Zwischen Kunst und Medien*, Wilhelm Fink Verlag, 1999, S. 209 - 218

[Whit97] J. E. White
Mobile Agents
In: [Brad97a], S. 437 - 472

[WhPa98] T. White, B. Pagurek
Towards Multi-Swarm Problem Solving in Networks
3rd International Conference on Multi-Agent Systems (ICMAS '98), 1998, S. 333 - 340

[Wirt95] A. Wirths
Das digitale Museum - Ein Museum im Datenraum
In: [ScSh95], S. 84 - 89

[WiKD98] A.Wirths, P. Kruse, T. Donga et al.
Der elektronische Raum
Cantz Verlag, 1998

[WoJe95a] M. Wooldridge, N. Jennings
Intelligent Agents: Theory and Practice
In: *The Knowledge Engineering Review*, Vol. 10(2) 1995, S. 115 - 152

[WoJe95b] M. Wooldridge, N. Jennings
Agent theories, Architecture and Languages : A Survey
In : *Intelligent Agent*, Springer-Verlag, 1995, S. 1 - 39

[xml99] *Extensible Markup Language*
http://www.oasis-open.org/cover/xml.html
1999

[xsl99] *Extensible Style Language*
http://www.oasis-open.org/cover/xsl.html
1999

[yaho01] *yahoo*
http://www.yahoo.com
2001