Computer Supported Cooperative Work-Applikationen in der Bauwerksplanung auf Basis einer integrierten Bauwerksmodellverwaltung

Dissertation

zur Erlangung des akademischen Grades

Doktor-Ingenieur

an der Fakultät Bauingenieurwesen
der Bauhaus-Universität Weimar

vorgelegt von
Dipl.-Inf. Thomas Hauschild
geb. am 18. April 1967
in Weimar

Tag der Disputation: 24. September 2003

Gutachter:
1. Prof. Dr.-Ing. habil. Reinhard Hübler
2. Prof. Dr.-Ing. habil. Udo F. Meißner
3. Prof. Dr. rer. nat. Bernd Fröhlich

Danksagung

Die vorliegende Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter an der Professur Informations- und Wissensverarbeitung der Bauhaus-Universität Weimar entstanden. Daher danke ich zuerst Prof. Dr.-Ing. habil. Reinhard Hübler für die fachliche Betreuung, die Anregungen und seine Geduld während des langen Entstehungsprozesses der Arbeit sowie für die Schaffung organisatorischer Randbedingungen, die eine Beschäftigung mit diesem Gegenstand ermöglichten.

Weiterhin möchte ich besonders cand.ing. André Borrmann, Dipl.-Ing. Matthias Schleinitz, Dipl.-Ing. Katrin Wender und Dipl.-Ing. Toni Fröbel für ihr langfristiges Engagement und ihren wesentlichen Anteil bei der programmtechnischen Realisierung eines Großteils der vorgestellten Mechanismen danken. Die durch ihre Mitarbeit entstandenen Prototypen waren für die Verifikation und Optimierung der dargelegten Konzepte essentiell.

Schließlich bedanke ich mich ganz besonders bei meiner Frau Annett und meiner Familie für den andauernden Rückhalt, die Möglichkeit zum ungestörten Arbeiten, den Verzicht auf gemeinsame Unternehmungen sowie die Korrektur des Manuskripts; durch ihre Unterstützung wurde das Entstehen dieser Arbeit erst möglich.

Kurzfassung

Gegenstand der vorliegenden Arbeit ist die Konzeption und prototypische Umsetzung von Techniken des Computer Supported Cooperative Work (CSCW) im Rahmen einer integrierten objektorientierten und dynamischen Bauwerksmodellverwaltung zur Unterstützung der Bauwerksplanung.

Die Planung von Bauwerken ist durch einen hohen Grad an Arbeitsteiligkeit, aber auch durch eine schwache Strukturierung der ablaufenden Prozesse gekennzeichnet. Besonders durch den Unikatcharakter des Planungsgegenstands 'Bauwerk' ergeben sich signifikante Unterschiede zum Entwurf anderer, durch Serienfertigung produzierter Industriegüter. Zunehmend wird die Planung von Bauwerken in Virtual Enterprises ausgeführt, die sich durch eine dynamische Organisationsstruktur, geographische Verteilung der Partner, schwer normierbare Informationsflüsse und eine häufig stark heterogene informationstechnische Infrastruktur auszeichnen.

Zur rechnerinternen Repräsentation des Planungsgegenstands haben sich objektorientierte Bauwerksmodelle bewährt. Aufgrund der Veränderlichkeit der Bauwerke und deren rechnerinterner Repräsentation im Laufe des Bauwerkslebenszyklus ist eine dynamische Anpassung der Modelle unumgänglich. Derartige in Form von Taxonomien dargestellte dynamische Bauwerksmodellstrukturen können gemeinsam mit den in Instanzform vorliegenden konkreten Projektinformationen in entsprechenden Modellverwaltungssystemen (MVS) gehandhabt werden. Dabei wird aufgrund der Spezialisierung und Arbeitsteilung im Planungsprozess von einer inhaltlich verknüpften Partialmodellstruktur, die räumlich verteilt sein kann, ausgegangen.

Die vorgeschlagenen Methoden zur Koordinierung der Teamarbeit in der Bauwerksplanung beruhen auf der Nutzung von CSCW–Techniken für 'Gemeinsame Informationsräume' und 'Workgroup Computing', die im Kontext der als Integrationsbasis fungierenden Modellverwaltungssysteme umgesetzt werden. Dazu werden die zur dynamischen Bauwerksmodellierung erforderlichen Metaebenenfunktionalitäten sowie Ansätze zur Implementierung von Modellverwaltungskernen systematisiert. Ebenso werden notwendige Basistechniken für die Realisierung von MVS untersucht und eine Architektur zur rollenspezifischen Präsentation dynamischer Modellinhalte vorgestellt. Da klassische Schichtenmodelle nicht auf die Verhältnisse in Virtual Enterprises angewendet werden können, wird eine physische Systemarchitektur mit einem zentralen Projektserver, Domänenservern und Domänenclients vorgestellt. Ebenso werden Techniken zur Sicherung des autorisierten Zugriffs sowie des Dokumentencharakters beschrieben. Zur Unterstützung der asynchronen Phasen der Kooperation wird der gemeinsame Informationsraum durch Mappingtechniken zur Propagation und Notifikation von Änderungsdaten bezüglich relevanter Modellinformationen ergänzt. Zur Unterstützung synchroner Phasen werden Techniken zur Schaffung eines gemeinsamen Kontexts durch relaxierte WYSIWIS–Präsentationen auf Basis der Modellinformationen verbunden mit Telepresence–Techniken vorgestellt. Weiterhin werden Methoden zur Sicherung der Group–Awareness für alle Kooperationsphasen betrachtet.

Inhaltsverzeichnis

B: GroupPlan- ein Groupwaresystem für die Bauwerksplanung

0 Einleitung

Die Nutzung computergestützter Lösungen hat in der Bauwerksplanung weite Verbreitung gefunden; die dazu genutzten Einzelapplikationen haben einen hohen Reifegrad erreicht. Da die Planung von Bauwerken durch die Kooperation von einer Vielzahl von Beteiligten charakterisiert ist, kommt dem **Betrachtungsgegenstand** der vorliegenden Arbeit, der Integration der Einzelapplikationen und der Schaffung einer effektiven Kommunikationsbasis eine hohe Bedeutung zu. In der Praxis der Bauwerksplanung stellt der diesbezügliche Informationsaustausch noch immer ein Problem dar, leider haben sich in den vergangenen Jahren die Hoffnungen in standardisierte Austauschformate wie STEP nicht vollständig erfüllt. Es ist zu erwarten, dass die Dringlichkeit der Verbesserung der Technologie des Informationsaustauschs und der Kooperationsunterstützung in naher Zukunft noch höher wird, da:

- die Projektbearbeitung in Virtual Enterprises bzw. Arbeitsgemeinschaften (ARGE's) zunehmend an Bedeutung gewinnt und derartige Organisationsstrukturen auf einen möglichst reibungslosen Informationsaustausch auch in einem heterogenen Umfeld angewiesen sind,
- aufgrund der Globalisierung der Märkte der Grad der geographischen Verteilung der Auftraggeber und Partner untereinander im Planungsprozess zunimmt und damit synchron nutzbare Planungstools zur Überbrückung der Distanzen im Falle notwendiger Kommunikations- und Koordinierungsaktivitäten unverzichtbar werden.

Die in den bereits im Bereich der industriellen Bauwerksplanung bzw. im akademischen Umfeld existierenden global oder kontinental verteilten Planungsgruppen eingesetzten Tools offenbaren derzeit noch einen beträchtlichen weiteren Entwicklungsbedarf. Andererseits haben sich in den letzten Jahren neue Technologien und Arbeitsorganisationsformen entwickelt, die bislang kaum umfassend ausgeschöpft werden und deren Anwendung zur Lösung der genannten Problematik beitragen kann. Zu diesen gehören unter anderem:

- Fortschritte in der Technologie und gestiegene Akzeptanz der objektorientierten Bauwerksmodellierung,
- Erkenntniszuwächse im Computer Supported Cooperative Work sowie in der Telearbeit mit den damit verbundenen Änderungen im Arbeitsumfeld, die auch durch das Schlagwort 'AnyTime / AnyPlace' versinnbildlicht werden,
- die Entwicklung, Verbreitung und mit vertretbaren Kosten bestehende Verfügbarkeit breitbandiger Netzwerktechnologien im LAN– und WAN–Bereich,
- die Entwicklung des Leistungsspektrums und die komfortable Anwendbarkeit hochentwickelter Middleware-Systeme,

- die Entwicklungen in der Gesetzgebung bezüglich der Rechtsverbindlichkeit digital signierter Dokumente als Basis eines elektronischen Datenaustauschs unter Verzicht des Austauschs von Papierdokumenten.

Deskriptive objektorientierte Ansätze der Bauwerksmodellierung haben sich in den letzten Jahren als geeignete Basis zur Schaffung einer digitalen, rechnerinternen Repräsentation der relevanten Bauwerksinformationen einzelner Fachgebiete der Bauwerksplanung und auch in der Gebäudebewirtschaftung erwiesen. Ebenso zeichnen sich Fortschritte im Management der Kohärenz der gesamten Informationsbestände aller Disziplinen eines Bauplanungsprojekts ab. Die Motivation für den Einsatz deskriptiver objektorientierter Bauwerksmodelle liegt in:

- ihrer Fähigkeit, unter Nutzung der Abstraktionsmechanismen des objektorientierten Paradigmas die relevanten Informationen realer Bausubstanz oder mentaler Planungsmodelle mit ihrer hohen Komplexität in handhabbaren, rechnerinternen Modellen abzubilden,
- deren universeller Nutzbarkeit innerhalb verschiedener Fachdisziplinen und für diverse Aktivitäten am Planungsgegenstand Bauwerk,
- der Möglichkeit, Modelle zu schaffen, die das Bauwerk in den überwiegenden Anteilen des sehr langen Bauwerkslebenszyklus mit seinen langen Nutzungsphasen begleiten können,
- der Möglichkeit, Bauwerksmodellinformationen und Applikationsinfrastruktur zu trennen,
- die Möglichkeit der Schaffung von selbstinterpretierenden Modellen, die über große Teile der Bauwerkslebensdauer lesbar, interpretierbar und damit nutzbar bleiben.

Für Planungsaufgaben wie den architektonischen Entwurf, die Revitalisierung von Bauwerken und das Facility Management haben sich laufzeitdynamische Bauwerksmodelle in aktuellen Forschungsprojekten sowie partiell im Praxiseinsatz bewährt. Die Motivation für die Dynamisierung der Modelle liegt unter anderem in der inhärenten Unvollständigkeit vorgefertigter Modelle, im Wunsch, persönliches Know How abzubilden oder in der Notwendigkeit der Anpassung der Bauwerksrepräsentation an das vorliegende Projekt.

Zielsetzung der vorliegenden Arbeit ist, den Grad der Unterstützbarkeit kooperativer Arbeit im Planungsprozess auf Basis deskriptiver, dynamischer, objektorientierter Bauwerksmodelle zu untersuchen sowie geeignete derartige Techniken zu konzipieren und prototypisch umzusetzen. Dazu sind Modellverwaltungstechniken zu untersuchen, die den Ansprüchen verteilter, kooperativer Entwurfsumgebungen genügen. Die in den zu beschreibenden Modellverwaltungskernen vorliegenden Bauwerksinformationen müssen für einen entfernten Zugriff zur Verfügung gestellt werden, ebenso sind Techniken der Modellpartitionierung zu untersuchen. Auf der Grundlage dieser Techniken können Mechanismen zur Unterstützung synchroner und asynchroner Kooperation in der Bauwerksplanung geschaffen werden. Die Eignung von CSCW–Basistechniken sind in diesem Zusammenhang zu untersuchen. Ferner müssen Gesamtsystem-

architekturen entworfen werden, die für die Gegebenheiten der Planungstätigkeiten in Virtual Enterprises geeignet sind.

Ausgehend von diesen Zielstellungen umfasst die vorliegende Arbeit nachfolgenden **Inhalt**: Im **Teil A** der Arbeit wird die Ausgangssituation der rechnergestützten Teamarbeit im Bauwerksentwurf betrachtet. Dazu wird im Kapitel 1 der Gegenstand der Bauwerksplanung behandelt und besonders die Aspekte der Kooperation mit ihrer Charakteristik und typischen Szenarien beleuchtet. Ebenso wird die Planung in Virtual Enterprises als typische Organisationsform und die Technologie des Concurrent Engineering behandelt. Ferner wird auf Anforderungen und Vorteile des Einsatzes von Techniken des Computer Supported Cooperative Work im Planungsprozess sowie auf bestehende Ansätze der Einführung derartiger Techniken eingegangen.

Einen Überblick über den aktuellen Entwicklungsstand im Bereich des Computer Supported Cooperative Work wird im Kapitel 2 gegeben. Dabei werden Eigenschaften von CSCW–Systemen näher betrachtet und Groupware–Architekturen behandelt. Darauf aufbauend werden Basistechniken des CSCW zur Unterstützung asynchroner und synchroner Kooperation eingeführt, bevor sich eine Übersicht der Klassen von Groupware–Systemen anschließt. Den Abschluss des Kapitels bildet eine Darstellung des aktuellen Standes der Technik bezüglich Verteilter Systeme und besonders Verteilter Objekte.

Gegenstand des Kapitels 3 sind Techniken der Bauwerksmodellierung als Integrationsbasis für Planungsprozesse. Nach den Grundlagen der Bauwerksmodellierung werden die Vorteile der Nutzung des objektorientierten Paradigmas für den vorliegenden Einsatzfall beleuchtet. Weiterhin werden Aspekte verteilter sowie dynamischer Bauwerksmodelle thematisiert.

Im **Teil B** der Arbeit wird die Konzeption und Realisierung von Groupware–Systemen für die Bauwerksplanung exemplarisch an GroupPlan und dem digitalen Bauwerksmodell des SFB 524[1] vorgestellt.

Die Komponenten zur Verwaltung eines einzelnen Domänenmodells werden im Kapitel 4 beschrieben. Nach der Vorstellung des Objektsystems und des genutzten Metamodells werden verschiedene Ansätze zur Implementierung von dynamischen Modellverwaltungstechniken systematisiert. Abschließend wird ein Konzept und dessen Realisierung zur rollenspezifischen Präsentation von laufzeitdynamisch definierten Bauwerksmodellinhalten vorgestellt.

Aufbauend auf der vorangehend dargestellten lokalen Sicht werden im Kapitel 5 Techniken zur Modellpartitionierung und zum entfernten Zugriff behandelt. Ebenso wird das Konzept einer zur Nutzung in Virtual Enterprises geeigneten Gesamtsystemarchitektur dargelegt. Abschließend

[1] Ein Teil der vorliegenden Arbeit ist im Rahmen der Bearbeitung der Teilprojekts D3 "Digitales Bauwerksmodell als Grundlage der Prozessintegration" des Sonderforschungsbereichs (SFB) 524 "Werkzeuge und Konstruktionen für die Revitalisierung von Bauwerken" entstanden.

werden nötige Basistechniken verteilter Systeme unter besonderer Beachtung des Einsatzes in einer räumlich getrennten, kooperativ genutzten Bauwerksplanungsumgebung thematisiert.

Im Kapitel 6 werden CSCW–Techniken zur Unterstützung der Teamarbeit in der Bauwerksplanung behandelt. Neben allgemeinen Groupware–Basistechniken werden Möglichkeiten, Konzepte und Realisierungsansätze von CSCW–Methoden zur Unterstützung sowohl synchroner als auch asynchroner Kooperation im Bauwerksentwurf betrachtet. Dabei werden sowohl allgemeine, im Anwendungskontext sinnvoll einsetzbare als auch rein anwendungsspezifische Techniken behandelt.

A Bauwerksplanung als Teamarbeit: Ausgangssituation

Übersicht:

1 Computergestützte arbeitsteilige Bauwerksplanung

1.1 Arbeitsteilige Bauwerksplanung: Aspekte der Teamarbeit

1.1.1 Planungsgegenstand

Im ersten Kapitel sollen Spezifika und Anforderungen an Groupware–Systemen für die Unterstützung von Prozessen im Bauwerksentwurf sowie ausgewählte Forschungsprojekte, die sich mit dieser Problematik befassen, behandelt werden. Zu Beginn des Kapitels sollen einige Besonderheiten von Bauwerken als Gegenstand der Planungs– und Entwurfsprozesse im Bauwesen betrachtet werden.

Eine Auswahl wesentlicher Eigenschaften, durch die sich Bauwerke bezüglich ihrer Planungsprozesse von den Gegenständen anderer typischer Ingenieurdomänen, wie z.B. dem Maschinenbau, der Elektroindustrie oder dem Automobilbau etc. unterscheiden, wird im folgenden dargestellt, da diese Eigenschaften beim Entwurf von computer– und netzwerkgestützten Entwurfsunterstützungssystemen berücksichtigt werden müssen:

- Unikatcharakter: Bauwerke sind Unikate und werden als solche in Einzelfertigung (bzw. in Ausnahmen in Kleinserien) gefertigt und geplant. Die Planungsprozesse besitzen einen hohen Anteil von nichtalgorithmierbaren Aktivitäten. Neben funktionellen und technischen Aspekten spielen bei der Planung auch ästhetische und künstlerische Anforderungen eine Rolle. Daher ist das Ergebnis der Planungsleistungen nicht in allen Aspekten formell zu verifizieren, ebenso ist die Verifikation auch nicht durchgängig automatisierbar. Erschwerend kommt hinzu, dass die Anfertigung von Prototypen aus wirtschaftlichen, terminlichen und technologischen Gründen nicht möglich ist. Um so wichtiger werden rechnerinterne Bauwerksmodelle, die bei der Prüfung der Entwurfsausgaben aus der Sicht verschiedener Disziplinen helfen können.
- Relative Entwurfskosten: Die Investitionssummen pro Einzelprodukt sind für Bauwerke sehr hoch, die Entwurfskosten sind absolut betrachtet – verglichen mit anderen Produkten wie z.B. Automobilen oder Mikroprozessoren – gering bis mittelhoch. Aufgrund des Unikatcharakters ist aber der relative Anteil der Entwurfskosten an den Gesamtproduktionskosten sehr hoch, da dieser Anteil häufig in der Größenordnung eines Zehntels der Gesamtkosten liegt. Angesichts der langen Lebensdauer und der hohen Gesamtkosten müssen die Planungsleistungen jedoch in einer guten Qualität ausgeführt werden. Dies unterstreicht die Notwendigkeit einer möglichst optimalen Unterstützung

sowohl der einzelnen Planungsaktivitäten als auch des gesamten Planungsprozesses, um in kurzer Planungszeit und bei niedrigen Planungskosten ein Ergebnis hoher Qualität erzielen zu können.

- Standardisierungsgrad: Der Standardisierungsgrad der Komponenten von Bauteilen ist relativ gering. Je nach Bauweise bestehen die Grundstrukturen von Gebäuden aus einer geringen (*in–situ* gefertigte Stahlbetonbauten) bis begrenzten (Stahlbau– (Misch–) konstruktionen) Anzahl standardisierter Bauelemente. Dies beeinflusst sowohl den Entwurf durch die Notwendigkeit der konstruktiven Durchbildung der Bauwerkskomponenten als auch die Fertigung, welche nur begrenzt auf nicht speziell vorgefertigte Elemente zurückgreifen kann.
- Langer Lebenszyklus: Bauwerke haben vergleichsweise lange Nutzungsphasen und einen möglicherweise mehrere Jahrhunderte langen Lebenszyklus. In der Planungsphase werden Entscheidungen getroffen, die über lange Zeiträume die Nutzung und die Betriebskosten beeinflussen. Wichtig sind daher Verwaltungs– und Speicherformen für Bauwerksmodelle, die möglichst unabhängig von aktueller Hardware, Betriebssystemen und Fachapplikationen sind, da die Wahrscheinlichkeit hoch ist, dass bei der nächsten Redesignphase die aktuell verwendete Entwurfsumgebung nicht mehr genutzt wird und möglicherweise auch nicht mehr verfügbar ist.
- Wiederholung von Entwurfs– / Redesign–Phasen durch Umnutzung: Durch Tradition, die hohen Neubaukosten und häufig wirtschaftlicheren Umbaukosten sowie die lange mögliche Lebensdauer kommt es bei Bauwerken zu Redesign–, Umbau– bzw. Revitalisierungsarbeiten, die wesentliche Änderungen oder Erweiterungen der Bauwerksstruktur mit sich bringen können. Dies ist vom Schiffsbau abgesehen eine Spezifik des Bauwesens, da die überwiegende Menge der Industrieprodukte ein festes Design haben und nicht für gravierende Änderungen vorgesehen sind. Auch dieser Aspekt unterstreicht die Notwendigkeit von digitalen Modellen für Bauwerke, die deren Lebenszyklus begleiten können. Ebenso sollte die Konsistenz zwischen realer Bausubstanz und Bauwerksmodell weitgehend gesichert werden.
- Strukturierung der Industrie: Die Strukturierung der Bauindustrie unterscheidet sich wesentlich von der anderer Branchen. Dieser Aspekt und dessen Auswirkungen auf die Planungsprozesse sowie deren Unterstützung wird im Abschnitt 3.3 betrachtet.
- Geringer Vorbestimmungsgrad der Fertigungstechnologie: Bauwerke besitzen eine vergleichsweise hohe Bandbreite von möglichen Bau–Technologien. Architekten, Tragwerksplaner und andere Fachplaner können aus einem großen Vorrat von Alternativen von Planungsdetails wählen und diese in sehr vielen Kombinationen anwenden. Daher ist es kaum möglich, a priori diejenigen Technologien zu bestimmen, die in einem bestimmten Projekt eingesetzt werden. Die Folge ist, dass Domänenmodelle zur Abbildung des Entwurfsfachwissens der jeweiligen Disziplin sehr groß werden, aber pro Projekt nur ein kleiner

Ausschnitt davon genutzt wird. Eine Möglichkeit, mit dieser Problematik umzugehen, sind dynamische Modelle, die zur Laufzeit um benötigte Informationen erweitert werden.

- Abstraktionsgrad der Dokumente: Die Ergebnisse der Planung besitzen einen beträchtlichen Abstraktions– und Unschärfegrad und eine vergleichsweise geringe Detaillierung. Die Domänenmodelle sind ebenfalls sehr abstrakt und enthalten viele implizite Annahmen über Entwurfsobjekte.

1.1.2 Szenarien und Charakteristik arbeitsteiliger, verteilter Bauwerksplanungsaktivitäten

Die Aktivitäten der Bauwerksplanung erfordern im allgemeinen eine arbeitsteilige Bearbeitung. Nur in Ausnahmefällen wird ein einzelner Bearbeiter alle Planungsleistungen in Vorbereitung des Neubaus oder der Revitalisierung eines Bauwerks erbringen; nur wenige Bauwerkstypen sind durch die Bauordnungen der Länder von einer Prüfung befreit und nur in einzelnen Bundesländern sind Planungsleistungen wie die Erstellung einer Baustatik nicht für alle Bauwerkstypen obligatorisch. Typischer für die Bauplanung sind Szenarien, in denen es zur Zusammenarbeit einer Vielzahl von Fachplanern, Gutachtern, Prüfern und Behörden kommt, z.B. sind häufig Architekten, Bodengutachter, Tragwerksplaner, HLS–Planer, Elektroprojektplaner, Prüfer und Baubehörden beteiligt. Nach Abschluss der Planungsphasen sind deren Ergebnisse für die bauausführenden Firmen, die Verantwortlichen für das Facility Management und den Eigentümer von Interesse.

Eine Auswahl von denkbaren Beispiel–Szenarien für Beziehungen zwischen Beteiligten im Bauwerksentwurf, die einen Austausch der Informationen über das Planungsobjekt erfordern, werden im folgenden dargestellt:

- Sequentielle Planung: bei kleineren Bauvorhaben, wie z.B. Einfamilienhäusern, werden die Unterlagen der Genehmigungsplanung durch ein Architekturbüro erstellt, diese einem Tragwerksplanungsbüro zur Erstellung der statischen Unterlagen übergeben und anschließend die genannten Dokumente der Bauaufsichtsbehörde übergeben. Diese übergibt die Dokumente an ein Prüfbüro zur Prüfung. Nach der Genehmigung werden die Dokumente der Ausführungsplanung, wie beispielsweise Bewehrungs– und Schalpläne, erstellt und die Planungsleistungen durch bestimmte Fachplaner (HLS, Elektro) erbracht, bevor die Ausführung der Bauleistungen beginnt. Bei diesem Szenario werden die vorhandenen Bauwerksinformationen sequentiell von einem Bearbeiter zum nächsten weitergegeben; wichtig ist die Interpretierbarkeit dieser Informationen und die Vermeidung von Informationsverlusten.
- Synchrone Bearbeitung mit Entwurfskoordinierung: Besonders in der Ausführungsplanung können mehrere Fachplaner parallel arbeiten. Beispielsweise könnten die Planungen für die Tragwerksdetaillierung, HLS, Elektroinstallation, Aufzüge, Detaillierung von Feuertreppen parallel ausgeführt werden. Dies kann zu Abstimmungs– bzw. Änderungsbedarf zwischen

den betroffenen Bearbeitern führen; kann der Konflikt nicht durch Absprachen zwischen diesen gelöst werden, muss der Projektverantwortliche darüber entscheiden. Daher ist es wichtig, effektive Werkzeuge zur Unterstützung der Absprachen als auch zur Bekanntgabe der Änderungen zur Verfügung zu stellen. Bei diesem Szenario werden die Informationen als gemeinsames Repository behandelt, auf das die Planer im Bereich ihrer Sicht zugreifen. Änderungen werden als inkrementelle Aktualisierungen sofort oder kurzfristig in das Repository übertragen und über ein zu definierendes Medium sind andere Bearbeiter zu benachrichtigen.

- Beratung durch Fachplaner/Spezialisten: Bei der Erstellung des Entwurfs wird ein Spezialist mit der Überprüfung spezieller, beispielsweise bauphysikalischer Aspekte des zu erstellenden Bauwerks betraut. Im Resultat der Analyse der aktuellen Situation werden (möglicherweise alternative) Vorschläge zur Gestaltung bestimmter Konstruktionselemente übermittelt, die in nächsten Entwurfsvarianten berücksichtigt werden. Diese neue Variante wird u.U. wiederum einer Analyse unterzogen, deren Resultate in den Entwurf einfließen. Dieser Prozess läuft, bis eine befriedigende Lösung erzielt wird. Bei diesem Szenarium arbeitet der Spezialist auf den Datenbeständen des Architekten, greift aber auf eine Vielzahl externer Datenbestände zu, um entsprechende Analysen auszuführen. Die Varianten zur Verbesserung des Entwurfs werden in die übergebenen Dokumente eingearbeitet, nach der Auswahl einer Variante wird das entsprechende Dokument übergeben. Somit wird auf den Informationsbeständen einer Domäne potentiell parallel gearbeitet, diese mit externen Ressourcen abgeglichen und neue Versionen oder Änderungsmengen ausgetauscht. Dieses Szenario unterstreicht die Notwendigkeit der Unterstützung (quasi–)paralleler Teamarbeit auch innerhalb der Informationsbestände einer Domäne.
- Geographisch verteilte Bearbeitung: Im letzten Jahrzehnt mehren sich Beispiele von Planungsprojekten mit über mehrere Länder oder Kontinente verteilten Partnern. Gründe dafür können in kostengünstigen Angeboten aber auch im Wusch zur Erhöhung der Parallelität der Bearbeitung durch Ausnutzung der Zeitzonen sein. Änderungswünsche eines Fachplaners können so häufig 'über Nacht' erfüllt werden, was eine Vermeidung von Zwangspausen bei Planungstätigkeiten bedeuten muss. Derartige Konstellationen erfordern einerseits gute Möglichkeiten zum Austausch der Bauwerksinformationen zwischen den Partnern, um die asynchronen Arbeitsphasen zu unterstützen, aber ebenso ausgereifte Techniken zur synchronen Kooperation, um Absprachen und Konfliktbehandlungen ausführen zu können, da face–to–face–Meetings nur in Ausnahmefällen durchgeführt werden können.
- Telearbeit: Zunehmend werden Bürotätigkeiten in Telearbeit ausgeführt. Denkbar wäre, bestimmte Entwurfstätigkeiten definierter Bauabschnitte oder Gebäudeteile durch Telearbeit ausführende Mitarbeiter realisieren zu lassen. Bei derartigen Szenarien sind wiederum Zugriffsmechanismen auf die gemeinsamen Informationsbestände und inkrementelle

Aktualisierungstechniken von Bedeutung. Telekonferenz– und Konfliktbehandlungs-Techniken sind ebenfalls hilfreich, aber nicht derart essentiell wie im vorigen Szenario, da es bei Telearbeit im allgemeinen auch zu Präsenzphasen kommt.

- Übergabe des Modells nach Abschluss der Planungsleistungen: Nach Fertigstellung der Bauausführung übergibt die Baufirma die Unterlagen an den Eigentümer des Gebäudes und dieser wiederum an den Facility Manager, nachdem die Informationen um evtl. im Bauprozess vorgenommene Änderungen aktualisiert wurden. Danach können die relevanten Informationsanteile in ein entsprechendes FM–System eingebracht werden und um weitere Informationen wie Ausstattungsdetails, Daten zur Betriebskostenabrechnung, Zugangs– und Schlüsselpläne etc. erweitert werden. In diesem Szenario findet ein unidirektionaler Informationsaustausch statt, anschließend erfolgt eine Extraktion relevanter Anteile des Bauwerksmodells. Dieses Szenario unterstreicht die Wichtigkeit von Import– und Exportfunktionalitäten und der Interpretation von Inhalt und Semantik der Bauwerksmodellinformationen.

Das es bei der Anzahl der Rollen im Bauplanungsprozess im allgemeinen zwangsläufig zu einer Vielzahl von Beziehungen zwischen verschiedenen Beteiligten kommen wird und abgesehen von einem Teil sehr kleiner Vorhaben nicht eine Person oder Firma eine größere Anzahl von Rollen ausüben wird, hat mehrere Ursachen:

- Struktur der Baubranche: In der deutschen Baubranche dominieren kleine bis mittlere Planungsbüros, die sich auf eine oder wenige Teilaufgaben des Planungsprozesses spezialisiert haben. Große Planungsfirmen, die in der Lage sind, nahezu alle Aufgaben abzudecken, existieren, haben aber zusammen nur einen Anteil von weniger als ein Fünftel des gesamten Bruttoumsatzes der Baubranche[1].
- Spezialisierung / Qualifikation: Ein weiter Grund für die Arbeitsteiligkeit ist die Ausbildung der Bearbeiter, deren Spezialisierung auf bestimmte Aufgaben oder andere besondere Qualifikationen. Diese Spezialisierung ist typisch für Domänen mit komplexen Aufgaben.
- Gesetzliche Vorgaben: Teilweise ist Arbeitsteiligkeit auch vorgeschrieben, so kann beispielsweise der Tragwerksplaner nicht gleichzeitig die statische Prüfung vornehmen, andererseits existieren Vorschriften zur Rolle der Bauaufsichtsbehörden im Planungsprozess.
- Parallele Bearbeitung: Komplexe Aufgaben werden häufig parallel bearbeitet, sofern die Teilaufgaben parallelisierbar sind, um die Bearbeitungszeit zu minimieren.

[1] Laut Statistischem Bundesamt lag 1999 der Anteil von Firmen mit mehr als 200 Beschäftigten bei ca. 18% des Gesamtumsatzes der Baubranche, andererseits arbeiteten zu diesem Zeitpunkt 35% der Beschäftigten in Firmen von 1–19 Beschäftigten, aber nur ca. 10% in Firmen mit mehr als 200 Mitarbeitern.

Trotz der Unterschiede der oben genannten Szenarien lassen sich einige Gemeinsamkeiten ableiten, die bei einer Unterstützung der verteilten kooperativen Prozesse im Bauwerksentwurf berücksichtigt werden sollten:

- Im Regelfall werden Bauplanungsprozesse in Zusammenarbeit mehrerer Planungsbüros ablaufen. Daher muss mit dem Einsatz heterogener Hard– und Software gerechnet werden.
- Der Integration der Bauwerksinformationen kommt in jedem Falle eine große Bedeutung zu, die relevanten Informationen müssen für die Entwurfs–Bearbeiter verfügbar gemacht werden, Mechanismen zur inkrementellen Aktualisierung der gemeinsamen Informationsbestände sind wesentlich.
- Im Bauwerksentwurf treten Phasen sowohl synchroner als auch asynchroner Kooperation auf, beide müssen adäquat im Sinne von Shared Design Repositories, Workgroup Computing und Computer Mediated Communication unterstützt werden.
- Bauwerksentwurfsprozesse laufen häufig auch geographisch verteilt ab, daher sind Mechanismen mit der nötigen Robustheit für einen Einsatz im Internet, in VPN's oder WAN's zu entwerfen. Ferner sind Techniken zur synchronen Kooperation dann äußerst bedeutend, wenn aus organisatorischen oder ökonomischen Gründen face–to–face–Meetings nicht oder nur selten zu realisieren sind.

1.2 Einsatzintentionen für Techniken des Computer Supported Cooperative Work in der Bauwerksplanung

1.2.1 Charakteristik der Planungsprozesse

Die Entwurfsprozesse im Bauwerksentwurf können unabhängig vom konkret vorliegenden Grad der IT–Unterstützung durch folgende Charakteristika gekennzeichnet werden [Maher 00] [Fenves 94] [Fruchter 98]:

- Multiple Disziplinen bzw. Fachgebiete: Die Prozesse im Bauwerksentwurf umfassen die Zusammenarbeit von Spezialisten einer Anzahl von Fachgebieten bzw. (Teil–)Disziplinen. Maher et al. gehen soweit, Entwurfsaufgaben allgemein als inhärent kooperativ zu charakterisieren. Jedes dieser Fachgebiete repräsentiert eine Spezialisierung, deren Aufgaben im allgemeinen aufgrund fehlender Expertise nicht vollständig durch Angehörige anderer Disziplinen erbracht werden können. Weiterhin bestehen häufig Unterschiede in der Terminologie der Disziplinen, in der Ausbildungsrichtung der Fachplaner, in Grundhaltungen im Prozess usw.
- Multiple Sichten auf das Bauwerk: Jede Fachdisziplin hat eine eigene Perspektive auf das Bauwerk, benötigt andere Teilmengen des Gesamtinformationsbestandes über den Entwurfsgegenstand und erzeugt Informationen unterschiedlicher Semantik. Ebenso

unterscheiden sich die direkten Kooperationsbeziehungen im Bauwerksentwurf zwischen den Disziplinen genauso wie die Menge der relevanten Eingangsinformationen.

- Variierende Grade der Automatisierbarkeit: Je nach Komplexität der Aufgaben, Grad der wissenschaftlichen Fundierung und der Algorithmierbarkeit der Aktivitäten sowie der Problemklasse der Entwurfsaufgaben[2] variiert der Grad der Überführbarkeit von Teilaktivitäten in rechnergestützte Lösungen. Ebenso kann von einem Entwurfsausführenden zum nächsten der Grad der Nutzung von Computerunterstützung variieren.
- Inkompatibilität von Hard– und Software: Jede Firma nutzt Hard– und Software für die Bewältigung der eigenen Aufgaben. Aufgrund der wechselnden Vertragsbeziehungen und der dynamischen Natur von Virtual Enterprises spielt die Kompatibilität von Hard– und Software zu derjenigen von Partnern nur eine untergeordnete Rolle bei deren Beschaffung. Durch die vergleichsweise große Zahl von Software–Insellösungen für verschiedene Entwurfsteilaufgaben und der oben behandelten Problematik neutraler Formate müssen diese Applikationen zumindest potentiell als inkompatibel angesehen werden. Daher kommen weiteren Integrationsbemühungen eine große Bedeutung zu.
- Interaktions–Modi: Die Ausprägung kollaborativer Entwurfsprozesse kann als eine Funktion von Zeit, Raum und gemeinsamen Informationsbestand aufgefasst werden; dabei kennzeichnet der Parameter Zeit die Ausführung synchroner oder asynchroner Kooperation, der Parameter Raum den Grad der räumlichen bzw. geographischen Verteilung und der Parameter gemeinsamer Informationsbestand beschreibt die Strukturierung und Inhalte der projektzentralen und domänenspezifischen Informationsbestände.

Die traditionelle Form der kooperativen Arbeit im Bauwerksentwurf nutzt keine Möglichkeiten von Rechnernetzwerken als Medium der Kommunikation im Bauwerksentwurf. Computer werden hier, wenn überhaupt, nur als Basis für die Ausführung von Insellösungen genutzt, die vorhandene Telekommunikationsinfrastruktur wird nur im POTS–Bereich genutzt. Derartige Entwurfsprozesse haben folgende charakteristischen Eigenschaften [Fenves 94] [Fruchter 98]:

- Eingeschränkte Kommunikation: Die Kommunikation erfolgt ausschließlich auf verbalem Wege und durch Austausch von Zeichnungen und Dokumentationen. Es gibt aber keine Mechanismen zur Kontrolle, ob nach Änderungen alle betroffenen Partner mit allen

[2] Diverse Forschungsarbeiten aus den achtziger und neunziger Jahren beschäftigten sich mit der Klassifikation von Designproblemen. Gängige Schemata definieren Well–Definied–Problems (Routine-Design) als Probleme, deren Lösungsziel klar definierbar ist, Ill–Defined–Problems (Innovative–Design) als Aufgaben, bei denen initial sowohl das Verfahren als auch die Elemente der Lösung als auch die exakten Entwurfsziele unbekannt sind und Wicked Problems (Creative Design) als Probleme, die derartig unzureichend definiert sind, dass weder eine endgültige Aufgabenstellung, noch ein Endpunkt des Entwurfsprozesses noch Mittel zur Verifikation einer Lösung definierbar sind [Steinmann 97a].

aktualisierten Informationen versorgt worden sind. Die Ursache hinter dieser Problematik ist, dass die Weitergabe von Änderungen hauptsächlich vom Gedächtnis der Fachplaner anhängig ist, die gemeinsame Interessen an bestimmten Bauwerksinformationen vor allem im mentalen Bauwerksmodell speichern.

- Unklare Lösungsevolution: Die Teammitglieder entwickeln eine Anzahl Lösungsalternativen und greifen teilweise nach Sackgassen im Entwurfsprozess auch auf frühere Varianten zurück, die Evolution der Lösungen und die Interaktionen im Team, die zu Änderungen führen, sind im Nachhinein schwer nachzuvollziehen.
- Späte Erkennung von Inkonsistenzen: Fehler und Inkonsistenzen werden häufig während Treffen von Bearbeitern entdeckt und müssen aufgelöst werden, bevor das Team weiterarbeiten kann. Durch die unzureichende Kommunikation im Prozess und das Fehlen automatisierter Ansätze zur Konsistenzprüfung kommt es zu dieser Problematik.
- Begrenzter Zugriff, erschwerte Aktualisierung und aufwendiges Retrieval von Bauwerksinformationen: Aufzeichnungen zum Projekt sind häufig private und daher nicht gemeinsam nutzbare Dokumente. Aktennotizen bzw. Memos werden an ausgewählte Teammitglieder übermittelt und dann abgelegt. Derartige Dokumente lassen sich nur schwer aktualisieren und durch die Entwurfsumgebung auffinden. Zeichnungen und Sachdaten, die durch eine Zeichnungsnummer indiziert werden, sind ebenfalls nur schwer auffindbar und in Papierform nur aufwendig zu aktualisieren oder zu erweitern. Dasselbe trifft auf Dokumentationen in Form sequentiell genehmigter Versionen zu, die als unterzeichnete Papierdokumente abgelegt werden.

1.2.2 Anforderungen und erwartete Vorteile des CSCW–Einsatzes

Die Intention des Einsatzes von Techniken des Computer Supported Cooperative Work (CSCW) und der Computer Mediated Communication (CMC) ist die Lösung eines Teils oder aller oben genannter Probleme traditioneller Entwurfsprozesse im Bauwerksentwurf. Darüber hinaus werden vom Einsatz derartiger Techniken weitere Vorteile erwartet [Schmitt 94]:

- Synergien und Beschleunigung: Durch verbesserte Kommunikation, Team–Awareness, assistierende Unterstützung durch die netzwerkbasierte Entwurfsumgebung und die Kombinierbarkeit der Informationsbestände der Projektpartner werden Synergien erwartet, ebenso resultieren diese Techniken in einer deutlich schnelleren Fertigstellung der Projektarbeit als bei traditionellen Verfahren, da ein höherer Grad der Parallelisierbarkeit erreicht, Kommunikation schneller und teilweise effektiver ausgeführt werden kann und Wartezeiten minimiert werden können.
- Komplexität: Die Komplexität von Großprojekten kann nur durch Einsatz von integrierten CAx–Lösungen und Automatisierung adäquat bewältigt werden, aktuelle Großprojekte werden generell unter Einsatz aktueller IT–Systeme ausgeführt.

- Designqualität: Die Ergebnisse der einzelnen Phasen des Entwurfs sollten auf den Resultaten vorangehender Phasen anderer Teams beruhen, eine bloße Kenntnisnahme der Dokumente ist nicht ausreichend. Durch die Integration der Systeme und die Kommunikation der Partner kann eine bessere Abstimmung der Teilaktivitäten im Entwurf erreicht werden.
- Fokussierung: Das Zentrum der Aufmerksamkeit der Bearbeiter wird auf das gemeinsame Projekt gelenkt und nicht auf die Gruppe der zusammenarbeitenden Personen.
- Autorenschaft: Die Autorenschaft an Teilergebnissen ist häufig eine Konfliktquelle im Team, die individuellen Ergebnisse werden in einem gemeinsamen Design gebündelt. Insofern es nötig wird, sollte das Repository der integrierten Entwurfsumgebung die Autorenschaft einzelner Teile der Entwurfsresultate ermitteln können.

Um die genannten Effekte beim Einsatz von CSCW– und CMC–Techniken zu erreichen, muss das eingesetzte System einige Anforderungen erfüllen und bestimmte Funktionalitäten aufweisen. Zu diesen gehören unter anderem:

- Der Informationsaustausch zwischen den Projektpartnern ist fundamental, anstelle von Techniken wie FTP müssen direkte, nutzerfreundlichere Varianten treten.
- Eine Informationsrepräsentation mit multiplen Views ist für multidisziplinäre Aufgaben essentiell, dafür muss eine geeignete Modellarchitektur bereitgestellt werden. Rosenman und Gero argumentieren dazu, dass für den Bauwerksentwurf ein einzelnes Modell nicht fähig ist, mehrere Views darzustellen, da es nicht möglich sei, durch Sichtenbildung disziplinen-spezifische Repräsentationen abzuleiten. Der Grund dafür ist, dass die Modellierungs-primitive schon ihrerseits sichtenspezifisch modelliert werden sollten. Zum Beispiel würden für die Architektursicht Wände für jeden umschlossen Raum auf jeder Etage separat modelliert werden, während Tragwerksplaner unter Umständen von einer durchgehenden Wand ausgehen würden. Bei multiplen (Teil–)Modellen muss jedoch die Konsistenz zwischen diesen gesichert werden.
- Unique ID's: Alle Objekte aller beteiligten Domänen müssen eine eindeutige, unique Identifikation besitzen, so dass nach Änderungen von Attributen in Applikationen, die möglicherweise mit Repliken arbeiten, eine eindeutige Zuordnung und dadurch ein Abgleich mit dem logischen Gesamtmodell nach deren Freigabe erfolgen kann.
- Partialupdates: Funktionalitäten zum Partialupdate müssen zum Erreichen eines effektiven Informationsaustauschs integriert werden. Dabei ist es wichtig, Mechanismen umzusetzen, welche die nichttriviale Problematik der Hinzufügung oder Löschung von Objekten ebenso wie Modifikationen behandeln.
- Kommunikationskanäle und gemeinsame Informationen: Zur effektiven Projektarbeit sowie zur Erfüllung der oben genannten Erwartungen an CSCW–basierte Entwurfsumgebungen

müssen mehrere Kommunikationskanäle und geeignete Typen gemeinsamer Informationsbestände bereitgestellt werden. Diese Kanäle werden zur direkten Kooperation, aber auch zur dafür notwendigen Koordinierung und der wiederum grundlegenden Kommunikation verwendet. Eine Auswahl von Kandidaten geeigneter Technologien ist:

- Nutzung des WWW, gemeinsamer WWW–Projektraum: Webbasierte Präsentationsformen besitzen den Vorteil, dass clientseitig nur geringe Anforderungen an Hard– und Software bestehen. Sowohl textuelle Informationen, die beispielsweise allgemeine Angaben zum Projekt oder einzelne Aufgabenbereiche, zum Bearbeitungsfortschritt u.ä. enthalten, können gut auf diese Weise dargestellt werden. Nachteilig ist die schwierige Integrierbarkeit derartig verwalteter Daten in Fachapplikationen zu bewerten, was auch durch neuere Entwicklungen wie XML nur marginal verbessert wird. XML und verwandte Technologien stellen für einige Aufgabenbereiche eine geeignete Basis für den filebasierten Informationsaustausch dar, es muss aber bezweifelt werden, ob Browserdarstellungen eine adäquate Präsentationsform für komplexe Modelle mit fünf– oder mehrstelligen Objektanzahlen sind.
- Interdisziplinäres Kommunikationsmedium Produktmodell: Die Kommunikation zwischen den Projektpartnern erfolgt hauptsächlich durch implizite Kommunikation durch die gemeinsamen Bauwerksmodellinformationen. Daher sind effektive Zugriffsmöglichkeiten auf diese Modellinformationen essentiell.
- Techniken zur Unterstützung asynchroner Kooperation: Während der Projektarbeit werden asynchrone Groupwaretechniken benötigt, um beispielsweise über Änderungen zu informieren oder Kommunikations– und Koordinierungsaufgaben wahrzunehmen. Asynchrone Techniken wie E–Mail und Bulletin Boards wirken im allgemeinen Arbeitsablauf weniger störend und bieten trotzdem eine hohe Übermittlungsgeschwindigkeit.
- Techniken zur Unterstützung synchroner Kooperation: Für Absprachen, Konfliktlösungen und andere Sitzungen sind eine Auswahl synchroner Techniken wie Videokonferenzen, Audiokanäle, Desktop–Sharing, Application Sharing, WhiteBoards zum gemeinsamen Zeichnen, textbasierte Talk–Lösungen usw. erforderlich.

Die interdisziplinäre Kommunikation durch das Bauwerksmodell sowie diejenige bei der Nutzung asynchroner und synchroner Groupwaretools kann als Computer Mediated Communication (CMC) aufgefasst werden. Dies ist eine Voraussetzung zu Bildung von Virtual Communities, weiterhin zeigen Untersuchungen, dass CMC in geographisch verteilten Teams stark dazu beiträgt, dass aus reiner Kooperation ('Gruppenarbeit' bei Zimbardo) Kollaboration ('Teamarbeit') erwächst, also eine gemeinsame Arbeit an einem gemeinsamen Gegenstand mit einem gemeinsamen (Team–)Ziel.

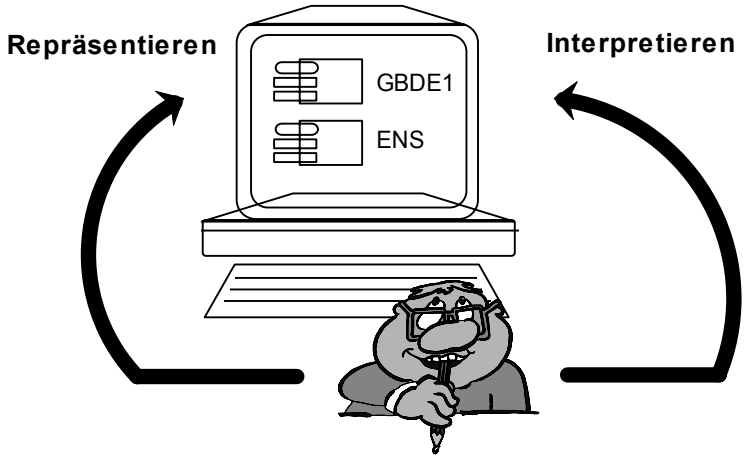

Abbildung 1: Reflexive Konversation mit gemeinsamem Informationsbestand [Maher00]

Maher, Simoff und Cicognani sehen in der dynamischen Interaktion zwischen dem Entwerfenden und der Repräsentation der gemeinsamen Informationsbestände die eigentliche Triebkraft in CMC–basierten Entwurfsumgebungen. Diese gemeinsame Informationsdarstellung sollte in der Lage sein, die Handhabung als auch die kognitive Reflektion des privaten und des 'gemeinsamen Verständnisses' des Planungsgegenstands zu ermöglichen. Die Entwicklung des 'gemeinsamen Verständnisses' in Verteilten, CMC–basierten Planungsabläufen basiert auf der zyklischen Repräsentation der individuellen Entwurfsintentionen im gemeinsamen Informationsbestand verbunden mit der Interpretation aller relevanter Daten im gemeinsamen Bestand [Maher 00].

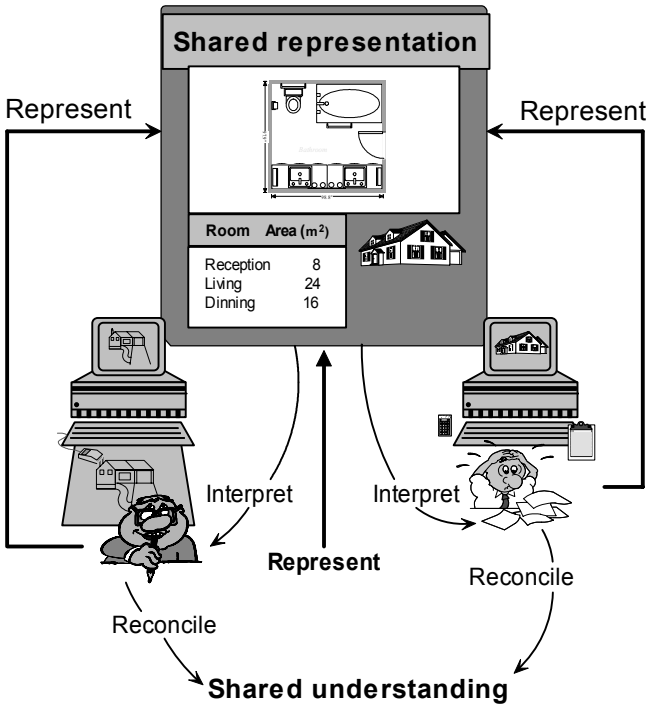

Abbildung 2: Gemeinsame Informationsbestände im Bauwerksentwurf [Maher00]

Die gemeinsamen Informationsbestände stellen eine notwendige, aber keine hinreichende Bedingung für die Entwicklung eines 'gemeinsamen Verständnisses' dar, da hierzu auch die Abstimmung der Arbeit und die Lösung von auftretenden Entwurfskonflikten notwendig ist.

Die Realisierung eines gemeinsamen Informationsbestandes erfordert die Definition einer Struktur der Informationsverwaltung und die Festlegung des Inhalts des gemeinsamen Bestandes. Dabei muss sowohl die Systemgesamtarchitektur als auch die Strukturierung der einzelnen Domänen betrachtet werden. Maher et al. schlagen für die Inhalte der gemeinsamen Informationsbestände Ontologien, also Beschreibungen der Semantik des Exkursbereichs der betreffenden Domänen[3] vor. Diese Ontologien dienen im Entwurfsprozess als Integrations–Framework für die verschiedenen Facetten des gemeinsamen Projektwissens und werden mit Mitteln der Bauwerksmodellierung erzeugt. Jedoch bestehen noch offene Fragen, wie die verschiedenen Entwurfsmetaphern verschiedener Domänen vereint werden können und wie die verschiedenen konzeptuellen Sichten und Modellierungsstrategien harmonisiert werden können.

1.3 Organisationsstrukturen in der Bauwerksplanung

1.3.1 Virtual Enterprises

Virtual Enterprises (Virtuelle Unternehmen) haben sich zu einer sehr häufig anzutreffenden Organisationsstruktur für Vorhaben der Bauplanung und –durchführung entwickelt. Die Gründe dafür liegen einerseits in der Notwendigkeit einer kooperativen Projektbearbeitung und andererseits in der Struktur vieler Unternehmen der Baubranche, die sich auf Aufträge für spezifische Leistungen und mit einem begrenzten Volumen konzentrieren. Im Gegensatz zu anderen Branchen gibt es relativ wenig Unternehmen, die sämtliche Aktivitäten der Planung und Produktion ihrer Produkte ohne weitere Partner durchführen können. Ebenso fehlt häufig der zentral agierende Produzent, der über ein Netz von Zulieferfirmen verfügt, wie dies beispielsweise im Maschinen–, Schiff–, Flugzeug– und Automobilbau häufig anzutreffen ist. Für diese Zulieferketten ist jedoch typisch, dass sie auf längerfristigen Geschäftsbeziehungen beruhen und der Informationsfluss zwischen Partnern in Übereinstimmung mit den Vertragsbeziehungen erfolgt. Dies ist im Bauwesen häufig nicht der Fall [Hannus 00].

Die Struktur eines Virtual Enterprise ist im allgemeinen dynamisch. Während der Projektdurchführung kommen neue Partner hinzu, deren Spezialisierung zu diesem Zeitpunkt benötigt werden, ebenfalls scheiden Partner aus, nachdem ihre vereinbarten Aktivitäten durchgeführt wurden. Im allgemeinen ist der Typ aller beteiligten Unternehmen noch nicht bei Projektbeginn definierbar, da beim Eintreten bestimmter Situationen weitere Firmen beteiligt

[3] Nach einer alternativen Definition sind Ontologien als explizite Beschreibung des gemeinsamen Verständnisses, also der allgemein anerkannten Kommunikationsbasis einer Domäne. Ontologien können beispielsweise durch Taxonomien oder auch Datenbanktabellen beschrieben werden und sollten dann durch einen Thesaurus mit Begriffen, Synonymen und Antonymen ergänzt werden.

werden müssen. Dies gilt besonders für Revitalisierungsvorhaben, aber auch für komplexe Neubaumaßnahmen. Ebenso können einzelne Unternehmen gleichzeitig mehreren Virtual Enterprises angehören, die die Durchführung verschiedener Projekte betreiben.

Ferner nennen Hannus und Kazi folgende weitere Spezifika dynamischer Virtueller Unternehmen [Hannus 2000]:

- häufig fehlt ein dominanter Handlungsträger, der in der Lage ist, Festlegungen für das gesamte Virtual Enterprise zu treffen
- Informationsflüsse und –formate sind schwer normierbar, Einflüsse auf die Informations– und Kommunikationsinfrastruktur der beteiligten Unternehmen ergeben sich kaum
- dasselbe trifft auf die verwendeten Applikationen zu
- die Beziehungen zwischen den Projektpartnern sind im allgemeinen temporärer Natur
- die Vertragsbeziehungen weisen eine andere Struktur als die Informationsflüsse auf
- komplementäre Kompetenzen werden i.a. durch unterschiedliche Unternehmen erbracht.

1.3.2 Concurrent Engineering

Unter Concurrent Engineering wird die kooperative Arbeit physisch verteilter Teams an ein und demselben Arbeitsgegenstand verstanden, bei denen mit Hilfe von Informations– und Kommunikationstechniken eine analoge Arbeitsweise zu lokaler Kollaboration erreicht wird [Scherer 98].

Grabowski definiert in [Grabowski 94] angelehnt an Penell und Winner Concurrent Engineering als systematischen Ansatz zum integrierten, nebenläufigen Entwurf von Produkten und ihren zugeordneten Prozessen einschließlich Herstellung und Wartung. Dieser Ansatz beabsichtigt, den Entwerfenden von Anfang an alle Phasen des Produktlebenszyklus von der Konzeption bis zur Entsorgung sowie die Qualität, Kosten und Nutzeranforderungen berücksichtigen zu lassen.

Concurrent Engineering verfolgt zwei hauptsächliche Ziele:

- die parallele und integrierte Realisierung von Entwicklung, Entwurf und Herstellung des Produkts um die Entwicklungszeit zu kürzen
- die Verbesserung der Produktqualität.

Um diese Anforderungen zu erfüllen, sind nach Anderl drei Hauptaspekte der Produkt-entwicklung hinreichend zu lösen [Anderl 92]:

- die Dekomposition der Entwurfsaufgaben in einzelne Aktivitäten
- die Kommunikation zwischen den einzelnen Entwurfsaktivitäten
- die Koordinierung und Synchronisation der Entwurfsaktivitäten.

Dabei ist zu beachten, dass die oben genannten Aspekte zur Einführung von Concurrent Engineering besonders die Verhältnisse im Maschinenbau, der Elektrotechnik und ähnlichen Branchen berücksichtigen. Durch die Spezifik der Entwurfsprozesse im Bauwesen erscheint eine Betrachtung jedes einzelnen Projekts nach diesen Kriterien im Sinne der exakten Beschreibung der Teilaktivitäten und der Definition des Workflows als zu aufwendig. Eine implizite, assistierende Unterstützung des Entwerfenden durch die integrierte Applikationsumgebung erscheint hier als ein realistischerer Weg, die genannten Aspekte abzudecken.

Concurrent Engineering im Bauwesen bedingt im allgemeinen die Existenz eines Virtual Enterprise und ist die natürliche Arbeitsweise in solchen. Systeme, die Entwurfsaktivitäten in Virtual Enterprises bzw. bei Concurrent Engineering unterstützten sollen, sollten aus Prozesssicht u.a. folgende Anforderungen erfüllen [Hannus 00] [Scherer 98] [Khemlani 97] [Wasserfuhr 98]:

- Eignung für die o.g. Spezifika der Planungsprozesse in Virtual Enterprises
- Bereitstellung einer systeminternen Bauwerksrepräsentation, die eine Grundlage der Arbeit aller Fachdisziplinen darstellen und somit als Basis eines gemeinsamen Verständnisses des Planungsgegenstands und dessen Fortschrittes dienen kann
- Unterstützung von Legacy–Software
- Verträglichkeit der Client–Lösung mit Infrastruktur des VE als auch mit lokaler Software
- kurze Einarbeitungszeit in die Infrastruktur des VE
- kurze Zeit zur Anbindung an Umgebung des VE‘s
- Kommunikation im VE möglichst konform zu Standard
- Schutz proprietären Know How’s
- Periodische Releases der Planungsergebnisse an Partner im VE
- Möglichkeiten zur Harmonisierung und Integration der verschiedenen Modelle
- Aktive Unterstützung des Flusses, der Verteilung und der Archivierung der Informationen
- Bereitstellung von Überblicksinformationen an Partner im VE zur Sicherung der Auswahl der richtigen Prioritäten bei Entwurfsentscheidungen
- Zentrale Archivierung, Dokumentenhandling, Versions–Management
- Dokumentation der Aktivitäten der Partner im Virtual Enterprise

Einige dieser Anforderungen ergeben sich ebenso aus Datensicht für die effziente Unterstützung von Concurrent Engineering. Weitere Anforderungen dafür sind unter anderem [Eastman 99]:

- Gewährleistung einer eineindeutigen (uniquen) Identifizierung
- Sicherung der Möglichkeit partieller Updates

- Bereitstellung von effektiven Möglichkeiten zur Navigation durch Modelle
- Bildung von Objektteilmengen sowohl bei interaktiver Modellbearbeitung als auch für automatisierte Bearbeitungsschritte in Applikationen
- Möglichkeiten zur Erkennung von Inkonsistenzen zwischen Partialmodellen sind wünschenswert, diese können den Nutzer benachrichtigen bzw. bei der Behebung der Entwurfskonflikte unterstützen
- Private Workspaces sind wünschenswert, in dem eine Exploration von Entwurfsalternativen möglich ist, ohne dass diese Änderungen propagiert werden, bevor ein Stadium erreicht ist, bei dem eine Freigabe erfolgen kann

Ein denkbarer Lösungsansatz zur Erfüllung der genannten Anforderungen liegt in der Schaffung eines offenen, verteilten, integrierten Bauwerksmodells mit Applikationen, die geeignete Groupware–Module aufweisen.

1.4 Ausgewählte teamunterstützende Bauplanungssysteme

Im folgenden sollen einige relevante Projekte bzw. Planungssysteme betrachtet werden, die entweder spezifische Ansätze zur Unterstützung kooperativer Arbeit in der Bauwerksplanung aufweisen oder durch den gewählten Integrationsansatz entsprechende Eigenschaften aufweisen. Dabei soll die getroffene Auswahl allerdings keinen Anspruch auf Vollständigkeit erheben, sondern eher das Spektrum der Ansätze illustrieren und dabei den derzeitigen Entwicklungsstand sowie die noch existierenden Defizite aufzeigen.

VEGA

VEGA (Virtual Enterprises using Groupware tools and distributed Architectures) ist ein ESPRIT–Projekt, welches im Zusammenwirken von Partnern aus der Industrie und akademischen Einrichtungen bearbeitet wurde. Zur Unterstützung der Teamarbeit stellt VEGA neben der Integration über gemeinsam genutzte Bauwerksmodelle einen 'Distributed Workflow Service', einen 'Interoperability Service' und einen 'Distrubuted Information Service' bereit. Der 'Distributed Workflow Service' ist ein Verteiltes Workflow–Managementsystem, das einschlägigen Spezifikationen der Workflow Management Coalition (WfMC) folgt, aber um ein Workflow–Repository und Komponenten zur Migration der Prozesse über Partnergrenzen hinweg erweitert wurde. Der 'Interoperability Service' stellt hauptsächlich Mittel zum Zugriff auf die im STEP-Format gespeicherten gemeinsamen Bauwerksinformationen bereit, dazu existieren Import– und Export–Module, ein SDAI–Dienst und Funktionalitäten zur Modellüberprüfung. Der 'Distributed Information Service' nutzt die COAST–Module zur Speicherung und Verfügbarmachung von strukturierten Dokumenten, die in Formaten wie SGML vorliegen, um eine Grundlage für Funktionalitäten eines verteilten standardisierten elektronischen Dokumentenmanagements zu schaffen. Die beschriebenen Funktionen von VEGA basieren technisch auf der Nutzung von Web–Diensten und der Anwendung von CORBA–Techniken. Die

relevanten Aspekte der Bauwerksmodellierung in VEGA werden im Kapitel 3 besprochen [Stephens 98].

ToCEE

Das ESPRIT–Pojekt ToCEE (Towards a Concurrent Engineering Environment in the Building and Engineering Structures Industry) zielt darauf ab, ein Framework für Concurrent Engineering und Simultanous Engineering in verteilten Entwurfsprozessen bereitzustellen. ToCEE ist das Nachfolgeprojekt von COMBI (Computer–Intergrated Object-Oriented Product Modelling Framework for the Building Industry), einem ESPRIT–Projekt, welches auf der Grundidee von "Integration durch Kommunikation" beruht und einen zu ToCEE ähnlichen Ansatz der Bauwerksmodellierung nutzte. ToCEE nutzt in der Datensicht eine spezifische Kommunikationsarchitektur für die Präsentation der Projektinformationen für die Teammitglieder, außerdem werden in der Prozess-Sicht Ansätze des Workflow–Managements verfolgt. Als Werkzeug zum Prozessmanagement wird ein sogenanntes 'Management Process and Co-ordination Board' genutzt, welches für die Planung, Detaillierung und Aktualisierung der Angaben über Aufgaben durch den Projektmanager sowie durch diesen beauftragte Mitarbeiter genutzt wird und entsprechende Informationen für die Teammitglieder bereitstellt. Weiterhin stellt ToCEE Server für gemeinsame Dokumente, Bauwerksmodellinformationen und Prozessmodelle als gemeinsame Informationsräume bereit. Als technische Infrastruktur werden erweiterte Webdienste verwendet, dazu wurde das Konzept der URL auf die Uniform Project Resource Locator (UPRL) erweitert [Scherer 98]. Weitere Aspekte von ToCEE und COMBI werden im Kapitel 3 behandelt.

CaribCAD

Das CaribCAD–Projekt des Georgia Institute of Technology untersucht Möglichkeiten des Outsourcings von Konstruktionsleistungen und betrachtet daher auch Werkzeuge zur Unterstützung entfernter Kooperation. CaribCAD nutzt dabei einen Ansatz mit drei Ebenen aus Kommunikationsunterstützung, CAD–File– und Dokumentenmanagement sowie Kollaborationsunterstützung und Workflow Management. Auf der Ebene der Kommunikationsunterstützung wird auf Intranet– und VPN–Techniken zurückgegriffen. Die Ebene zum CAD–File– und Dokumentenmanagement besitzt ein Web–Interface und beruht technisch auf einer spezifischen Konfiguration von einzelnen Serverkomponenten. Zum Workflow Management wird auf WfMC-konforme Lösungen zurückgegriffen, die Implementierung beruht auf Microsofts proprietären Exchange Collaborative Data Objects(CDO)– Environment.

INTESOL

INTESOL ist ein Projekt des Instituts für Industrielle Bauplanung (IFIB) der Universität Karlsruhe, welches sich vorrangig mit der integralen Planung solaroptimierter Gebäude befasste. Zur Verbesserung der Unterstützung des Planungsprozesses wurden auch CSCW–Techniken betrachtet. Im Rahmen des Projekts wurde der Prototyp einer webbasierten Planungsplattform

entwickelt, die Bearbeiter im Kontext ihrer Arbeit und zur Entscheidungsunterstützung mit adäquaten Informationen versorgen sollen und als Basis der Integration dienen soll. Weiterhin wurden im Rahmen des Projekts dediziert Techniken zur Gestaltung von Interfaces von Webportalen untersucht [Forgber 97] [Forgber 98] [Dinger 99].

InteGrA

Das InteGrA–Projekt wurde am IFIB der Universität Karlsruhe bearbeitet und entwickelte mit einem Praxispartner einen Prototyp einer integrierenden Groupwareanwendung für ein Architekturbüro. Ziele waren dabei eine für das Qualitätsmanagement geeignete Projektdokumentation, eine aktive Projektmanagementunterstützung, der Aufbau von Wissenspools sowie die Möglichkeit eines standortunabhängigen Arbeitens. Dazu wurde eine Lotus Notes basierte Groupwareplattform realisiert, die den Anwendern hauptsächlich zum Projektmanagement dient, Auskunft über den Projektfortschritt gibt und einfache Workflows unterstützen kann. Eine Integration der Bauwerksmodellinformationen ist nicht dokumentiert [Müller 98].

InduSys

InduSys ist ein kooperationsunterstützendes System für die Tragwerksplanung im industriellen Stahlhochbau, welches an der Ruhr–Universität Bochum entwickelt wurde. Es beruht auf dem stahlbauspezifischen Produktmodell 'PlaKon' und dem anwendungsneutralen Prozessmodell 'Cooperate'. Das System basiert auf einer vollständig replizierten Systemarchitektur und der Nutzung von CORBA [Bretschneider 98].

Collaborative 4D–CAD

Collaborative 4D–CAD ist im Rahmen der Arbeiten an 4D–CAD–Systemen, also Umgebungen, die die temporale Dimension der Planung und Bauausführung berücksichtigen, am Center for Integrated Facility Engineering (CIFE) der Stanford University entstanden. Collaborative 4D–CAD soll die Interaktion zwischen Projektkomponenten und Ressourcen im Laufe der Zeit erfassen, Interaktionen visualisieren und Real–Time–Interaktionen der Nutzer mit dem 4D–Modell erlauben. Als Basis der kooperativen Arbeit wird ein Austausch von zeichnungsbasierten AutoCAD–Modellen und die Bereitstellung von Primavera–Projektinformationen realisiert, die als gemeinsamer Informationsraum genutzt werden [McKinney 98] [McKinney 00].

IdeAs–Architektur Berkeley University

Khemlani und Kalay beschreiben eine Architektur zur interdisziplinären Zusammenarbeit im Bauwerksentwurf. Auf der Grundlage mehrerer für alle Beteiligten zugreifbaren Objekt– und Projekt–Datenbanken sollen diverse 'Intelligent Design Assistants' (IdeAs) arbeiten. Zur Abbildung der Geometrie wird eine spezifische Datenstruktur entworfen, aus der raum– und bauteilorientierte Sichten leicht abgeleitet werden können. Die Objekt-Datenbanken sind spezifische Wissensbasen, die Informationen über diverse Bauteile enthalten. In den Projekt-

Datenbanken sollen alle Elemente des Bauwerks mit ihren geometrischen Eigenschaften und ihrer Lage abgebildet werden. Der Ansatz ist jedoch als äußerst geometrie- und topologiezentriert anzusehen, übliche in Produktmodellen gespeicherte Informationen bleiben unberücksichtigt. Ebenso ist zweifelhaft, ob mit einem einzelnen Geometriemodell der Bedarf aller beteiligten Domänen berücksichtigt werden kann (siehe auch Kap. 3) [Khemlani 97].

Stanford PBL

Am Project Based Learning (PBL) Laboratory der Stanford University werden regelmäßig verteilte A/E/C-Entwurfsprojekte für Studierende mehrerer Universitäten durchgeführt, wobei häufig die Teammitglieder über mehrere Kontinente räumlich verteilt sind. Zur verteilten Arbeit kommt in diesem Falle noch teils synchrone, teils asynchrone Kooperation aufgrund der Zeitverschiebung hinzu. In dieser Lehrumgebung lösen die Studenten eine Entwurfsaufgabe, die Aktivitäten mehrerer Domänen beinhaltet. Zur technischen Umsetzung werden gemeinsame Arbeitsräume im Web genutzt, in die beispielsweise VRML–Modelle, AutoCAD–Zeichnungen, Spreadsheets, textuelle Notizen u.ä. eingespeichert werden. Zur synchronen Kooperation werden einfache Konferenztechniken wie NetMeeting genutzt, um einen gemeinsamen Kontext für die Projektbearbeiter bereitzustellen; parallel wird dabei eine Videokonferenz geschaltet. Veröffentlichungen des PBL konstatieren die Notwendigkeit einer besseren Interoperabilität zwischen den Planungstools und die Vorteile modellbasierten Informationsaustauschs, die bislang in der Lehrumgebung nicht verfügbar sind [Fruchter 98]. Ebenso wirken sich die fehlenden Awareness–Eigenschaften erschwerend bei der Arbeit in synchronen Phasen aus.

Phase(X)

Eine weitere kollaborative Lehrumgebung ist Phase(X) der Eidgenössischen Technischen Hochschule Zürich. Auch hier wird auf eine räumliche Verteilung Wert gelegt, allerdings liegt der Schwerpunkt vorrangig auf Aktivitäten des architektonischen Entwurfs. Die Ergebnisse der einzelnen Entwurfsschritte werden in einer relationalen Datenbank abgelegt und durch diese anderen Teammitgliedern zur Verfügung gestellt [Schmitt 98].

WINDS

Eine EU–geförderte Lehrumgebung ist WINDS, die Studenten der Architektur und des Bauingenieurwesens Kenntnisse in der Durchführung von Projekten in einem verteilten Umfeld vermitteln soll. Dabei existieren Partner in zehn europäischen Ländern. Zur Unterstützung der Kooperation werden von partiell vom Fraunhofer–FIT entwickelte Kooperationstools genutzt. Als technische Infrastruktur werden Techniken des WWW's und XML genutzt [WINDS 00] [Specht 01].

2 Computer Supported Cooperative Work und Groupware

2.1 Gegenstandsbestimmung: CSCW und Groupware

2.1.1 Begriffsdefinitionen

Der Terminus *‚Computer Supported Cooperative Work' (CSCW)* wurde erstmals 1984 von I. Greif (MIT) und P. Cashman (Digital Equipment) als allgemeiner Sammelbegriff für Belange der Unterstützung der Zusammenarbeit mehrerer Personen durch Computersysteme verwendet [Bannon91]. Eine endgültige Definition existiert für diesen Begriff bislang nicht, im allgemeinen wird jedoch unter CSCW das vergleichsweise junge interdisziplinäre *Forschungsgebiet* verstanden, dass sich mit Gruppenarbeit und deren Unterstützung durch Computertechnologie befasst [Wilson91] [Rüdebusch 93] [Teufel 95].

Unter *Groupware* werden CSCW–Applikationen, also auf Informationstechnologie basierende Systeme zur Unterstützung von Teamarbeit verstanden, wie es in der Definition von Johansen zum Ausdruck kommt: ‚Groupware is a generic term for specialized computer aids that are designed for the use of collaborative work groups.' [Johansen 88]. Die Bedeutung der gemeinsamen Arbeitsumgebung der Teammitglieder wird in der Groupware–Definition von Ellis et al. betont: ‚... we define groupware as computer–based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment' [Ellis 91].

Kooperative Arbeit findet als Zusammenwirken von mindestens zwei Personen an mindestens einer gemeinsamen Aufgabe statt. Dieser Personenkreis bildet zusammen ein *Team*[1]. Typische Eigenschaften eines Teams werden von Johansen im Kontext der Groupware–Definition genannt: ‚typically ... collaborative work groups ... are small project–oriented teams that have important tasks and tight deadlines' [Johansen 88].

---

[1] Die Begriffe ‚Team' und ‚Gruppe' bzw. ‚Arbeitsgruppe' können nicht als synonym betrachtet werden. Außerdem werden in der (deutschsprachigen) Literatur diese Termini nicht einheitlich verwendet. Deutungen des Gruppenbegriffs reichen beispielsweise von mehreren interagierenden Personen mit gegenseitiger Beeinflussung [Teufel 95] bis hin zu Teams mit speziellen Eigenschaften wie sozialer Zusammmengehörigkeit und Geborgenheit [Burger 97]. Für die folgenden Ausführungen soll die Unterscheidung nach Zimbardo angewandt werden: von einer Gruppe wird dann gesprochen, wenn eine Interaktion zwischen einer Anzahl Personen stattfindet, bei der eine gegenseitige Beeinflussung vorliegt. Eine Gruppe wird zum Team, sobald die Beiträge einzelner Individuen zugunsten des Erreichens eines gemeinsamen Ziels oder eines Auftrags koordiniert werden [Zimbardo 92].

2.1.2 Historische Entwicklung des Computer Supported Cooperative Work

Betrachtet man die Entwicklung der Informationstechnologie von der Einführung der ersten kommerziell verfügbaren Computer bis heute, ist festzustellen, dass sich der Schwerpunkt der Aktivitäten bei Entwicklungen in der Computertechnik schrittweise von der Hardware zur Software verschoben hat und diese nun Auswirkungen auf die sozialen Prozesse im Arbeitsumfeld haben. Bis heute haben sich in dieser Entwicklung fünf Interface–Ebenen abgelöst [Grudin 90]:

Hardware–Interface: In den fünfziger Jahren waren Ingenieure und Programmierer die typischen Computernutzer. Interaktionen mit dem Rechner liefen äußerst maschinennah im Binär–, Oktal– oder Hexadezimalsystem unter direkter Manipulation von Speicherstellen und Hardwareregistern ab. Die Hauptanstrengungen zu dieser Zeit liefen auf die Maximierung der Hardware–Performance hinaus. Für Verbesserungen der Mensch–Maschine–Schnittstelle gab es vorerst drei Wege: geeignetere und zuverlässigere Hardware wurde entwickelt, die Ergonomie der Präsentation und Manipulation der Hardware wurde beispielsweise durch verbessertes Arrangieren und Beschriften von Schaltern verbessert und vor allem wurden erste Programmiersprachen zur Hardware–Abstraktion entwickelt.

Software–Interface: Bis Mitte der siebziger Jahre blieben Programmierer die typischen Computernutzer. Forschungsarbeiten zur Verbesserung der Programmierschnittstelle wurden durchgeführt, um die gesamte Computernutzung effizienter zu gestalten. In dieser Phase wurden Konzepte wie Betriebssysteme, Multitasking und erste höhere Programmiersprachen entwickelt.

Terminal–Interface: Seit den siebziger Jahren öffnete die breite Verfügbarkeit von Bildschirmen völlig neue Entwicklungs– und Forschungsrichtungen. Der übliche Computeranwender wurde der Nutzer von Software. Der Begriff ‚User–Interface' ist in dieser Phase geprägt wurden, ebenso entstand die Forschungdisziplin der ‚Human–Computer–Interaction' (HCI). Themen wie perzeptionelle und kognitive Prozesse sowie Ergonomie bei der Computerinteraktion bekamen eine dominante Rolle.

Dialog–Interface: In den achtziger Jahren verschob sich der Fokus der Forschungsaktivitäten hin zur tieferen Untersuchung der kognitiven Prozesse. Ergebnisse aus Arbeiten auf den Gebieten der kognitiven Psychologie und der Psycholinguistik flossen in die Weiterentwicklung der Mensch–Maschine–Schnittstelle ein. Die Majorität der Anwender wurden ‚Endnutzer' im heutigen Sinne, forciert wurde diese Entwicklung durch die Verbreitung von PC's und Workstations.

Arbeitsumfeld–Interface: Die Erfüllung von Arbeitsaufgaben geschieht im allgemeinen in einem sozialen Kontext. Seit Beginn der neunziger Jahren häufen sich Bemühungen, Teamarbeit zu untersuchen und durch Informationstechnogie zu unterstützen. Es entstand das Forschungsgebiet des Computer Supported Cooperative Work (CSCW). Die notwendige

technologische Basis für Groupware entstand durch Fortschritte der Netzwerktechnik und der Verteilten Systeme.

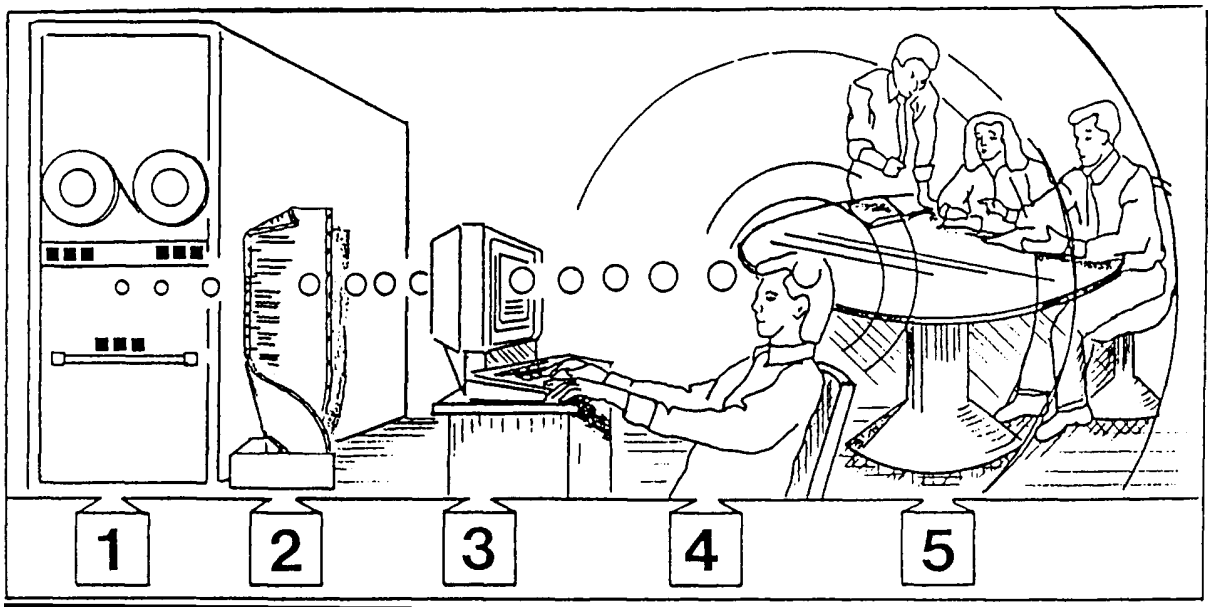

Abbildung 1 : Evolution der forschungsrelevanten Interface–Ebenen nach Grudin

Die Entwicklung der ersten CSCW–Applikation wird Douq Engelbart mit der Entwicklung von NLS (oN Line System) in den sechziger Jahren am Stanford Research Institute zugeschrieben, über zwei Dekanden bevor derartige Applikationen ein breiteres Forschungsinteresse fanden.

Mehrbenutzer–Editoren

NLS ist ein System zur ‚Gruppenverstärkung' (augmentation) und ermöglichte im wesentlichen eine kooperative Bearbeitung von strukturierten Texten unterstützt durch On–Screen–Teleconferencing, Shared Screen–Kooperation und Telepointing. NLS hatte über den CSCW–Bereich hinaus großen Einfluß auf die Entwicklung des Arbeitsplatzrechners: zweidimensionales bildschirmorientiertes Edieren, graphische Nutzerschnittstellen, Bildschirmfenster, hypermediale Informationsstrukturen sowie die Nutzung der Maus haben Ursprünge im NLS–System. NLS wurde 1968 erstmals öffentlich präsentiert und hatte einen kommerziellen Nachfolger *‚Augment'*.

GROVE (Group Outline Viewing Editor) wurde Ende der achtziger Jahre prototypisch von Ellis, Gibbs und Rein bei der Firma MCC entwickelt, um Erfahrungen bei der Implementierung und Anwendung von ‚Echtzeit–Multi–User–Werkzeugen' zu sammeln [Ellis 91]. GROVE ist ein kooperativ nutzbarer Texteditor. Das verwendete Gruppenfenster zur Dokumentenbearbeitung folgt dem WYSIWIS (What You See Is What I See)–Prinzip, d.h. allen Teilnehmern an der Rechnerkonferenz werden identische Informationen im Fenster angezeigt. Das System propagiert alle Änderungen am Text sofort, d.h. Einfügungen, Löschungen von Zeichen, Öffnen und Schließen von Teildokumenten und Scrollbar–Operationen werden sofort übertragen. In Bezug auf die Position und Anzahl der Fenster, die Darstellung mehrerer Cursors und die Anzeige der

Zugriffsrechte sind Abweichungen vom strikten WYSIWIS möglich. Weiterhin realisiert GROVE ein Konzept der Zugriffsbeschränkung auf Teildokumente, indem Teilbäume als ‚private' (nur durch die Erzeuger zugreifbar), ‚public' (öffentlich) oder ‚shared', d.h. einer selektierten Menge von Teilnehmern zugänglich deklariert werden können. Spätere Versionen von GROVE unterstützen temporäre Unterbrechungen der Mitarbeit einzelner Teilnehmer, indem nach der Wiederanmeldung neue Textbausteine hervorgehoben werden.

Quilt ist ein weiteres historisch relevantes Werkzeug zur kooperativen Bearbeitung hypermedialer Textstrukturen und wurde gleichzeitig circa 1988, also gleichzeitig zu GROVE, bei Bellcore (Bell Communication Research) auf Grundlage sozialwissenschaftlicher Untersuchungen entwickelt. Daher werden soziale Rollen beim Edieren sehr weit unterstützt, indem eine Rollenhierarchie existiert, nach der die Rechte in Bezug auf Operationen am Dokument sowie Anmerkungen festgelegt werden können. Rechte, die einer Rolle zugewiesen sind, werden implizit auf die höher in der Hierarchie aufgeführten Rollen übertragen. Das arbeitsteilige Edieren von Text wird von Quilt durch Nutzung einer zentralen Datenbank realisiert, welche alle Textbestandteile und Anmerkungen enthält. Die Teilnehmer werden über neue Textversionen erst dann benachrichtigt, wenn deren Urheber diese festschreibt bzw. Anmerkungen sendet. Sollten zwei Bearbeiter gleichzeitig dasselbe Dokument modifizieren, wird von Quilt eine Warnung an diese gesandt. Außerdem kann Quilt Dokumente sperren, die sich in Bearbeitung befinden. Wurde ein Dokument gleichzeitig kooperativ bearbeitet, generiert Quilt einen Vorschlag, der die Modifikationen aller Bearbeiter enthält.

Konferenzsysteme

EMISARI (Emergency Management Information System and Reference Index) wurde Anfang der siebziger Jahre von Murray Turroff am New Jersey Institute of Technology entwickelt und dessen Diskussionsmodul gilt als Prototyp für elektronische Konferenzsysteme. Das System ermöglicht kooperatives Schreiben und Lesen von Texten in verteilt nutzbare Notizbücher nach Themenbereichen (Konferenzen) strukturiert und den Versand von Nachrichten unter Bezugnahme auf Einträge. Die Nachfolger *EIES* und *EIES2* (1976) erweitern das Systemkonzept um Notizbücher für einzelne Benutzer und Teilteams.

Confer ist eine Entwicklung der University of Michigan aus den späten siebziger Jahren. Erste Versionen wurden zur Übermittlung von Vorschlägen an Teilnehmer von elektronischen Konferenzen und deren Voten genutzt. Später wurde das System um Features für Real–Time–Rechnerkonferenzen erweitert. Confer ermöglichte eine Unterteilung der Informationen nach Themenbereichen.

Usenet[2] ist das vermutlich verbreitetste Bulletin–Board–System und existiert seit Anfang der achtziger Jahre. Es basiert auf der Netzwerkinfrastruktur des Internets und ermöglicht

[2] andere gängige (wenn auch nicht exakte) Bezeichnungen für usenet sind ‚Internet News' oder nur kurz ‚News'

asynchrone Diskussionen bzw. Konferenzen zu weltweit einigen tausend hierarchisch strukturierten Themengebieten. Die Anzahl der Beiträge kann je nach Forum Dimensionen bis zu einigen hundert 'Posts' pro Tag annehmen.[3]

Application Sharing / Information Sharing

MBlink ist einer der ersten Systeme zum ‚Application Sharing' und wurde von Greif und Sarin Anfang der achtziger Jahre am MIT entwickelt. Es bestand aus einer Programmbibliothek, die zu existierenden Programmen für Xerox–Workstations nach einer geringfügigen Modifikation dieser gebunden wurde. MBlink ermöglichte dann eine Darstellung der graphischen Ausgaben des Programms auf mehreren Workstations und kann somit als Vorgänger von Shared X angesehen werden.

RTCAL wurde von Greif und Sarin Anfang der achtziger Jahre am Massachusetts Institute of Technology (MIT) entwickelt [Greif 88]. RTCAL ist eine Erweiterung des zu dieser Zeit am MIT in Nutzung befindlichen individuellen Kalendersystems PCAL um Features für Real–Time–Konferenzen und Information Sharing. Die PCAL–Kalender der Teilnehmer können diesen jeweils auf ihrem Bildschirm angeordnet werden, um sie bei der Suche nach einem möglichen Termin zu unterstützen und mit einem Abstimmungsmechanismus kann dieser festgelegt werden. Eine Funktion zur Suche einer gemeinsamen freien Zeit in allen Kalendern ist in RTCAL noch nicht integriert. Im Schwesterprojekt *MPCAL* wurden Kalender mit überwachter Aufteilung und Delegierung von Autorität auf Basis von Rollen im Team wie beispielsweise Kalendereigentümer, Sekretär, Team und Öffentlichkeit untersucht.

Nachrichtensysteme

Information Lens wurde Ende der achtziger Jahre am Massachusetts Institute of Technology (MIT) entwickelt und stellt ein System zur Erstellung, Versand und Verwaltung semistrukturierter Nachrichten zur Aktivitätskoordination dar. Die semistrukturierten Nachrichten enthalten ein Textfeld mit freiem Format sowie Felder mit vordefinierter Syntax und Semantik zum Beispiel für Terminangaben, Betreffzeile und Angaben zum Sender oder Empfänger. Dadurch wird ein intelligentes Filtern und Weiterverarbeiten der eingegangenen Nachrichten möglich und einer Überflutung des Empfängers vorgebeugt. *Object Lens* ist eine Weiterentwicklung des Systems und wird von den Entwicklern als ‚wissensbasierte Entwicklungsumgebung für kooperative Systeme' sowie als ‚Nutzerschnittstelle, welche Hypertext, objektorientierte Datenbanken, elektronische Nachrichten und regelbasierte intelligente Agenten integriert' bezeichnet. Object Lens verwendet semistrukturierte Objekte zur Darstellung von Nachrichten, Aktivitäten, Personen, Sitzungen etc., wobei das System zwischen Objekttypen und konkreten Instanzen unterscheidet. Sogenannte ‚Felder', sie entsprechen

[3] Die Diskussionsgruppen werden auch Foren oder Newsgroups genannt. Die Anzahl der Beiträge pro Tag und Forum wird häufig als ‚traffic' bezeichnet.

Attributen, können mit Feldern anderer Objekte verbunden werden oder auf andere Objekte verweisen.

E–Mail–Systeme gelten als bislang erfolgreichste CSCW–Applikation und haben ihren Ursprung in der Entwicklung des Arpanets, dem Vorläufer des Internets in den siebziger Jahren. Es existieren verschiedene Weiterentwicklungen, die häufig auch weitere Medientypen als Text zulassen. Das *Andrew*–Projekt der Carnegie Mellon University und IBM untersucht verteilte Infrastrukturen für kooperative Prozesse und die gemeinsame Nutzung bestimmter Informationen. Das Andrew–System besteht im wesentlichen aus dem Andrew–Nachrichtensystem, dem transparenten und verteilten Andrew–Filesystem und dem Andrew–Toolkit für die portable Implementierung von Benutzerschnittstellen. Das Nachrichtensystem ist ein portables Mail– und Bulletin–Board–System, das Text, Graphik, Tabellen und spezielle Nachrichtentypen wie Voten verarbeiten kann.

2.1.3 Gruppenbewusstsein (Awareness)

Eine grundlegende Eigenschaft von CSCW– Systemen ist die des Gruppenbewusstseins (Group Awareness oder auch Collaboration Awareness), die leicht vereinfacht dargestellt aussagt, ob für Anwender von Mehrbenutzer–Applikationen die Präsenz und Aktivität einer Anzahl anderer Teammitglieder augenscheinlich wird oder transparent bleibt. Dourish und Bellotti definieren Gruppenbewußtsein folgendermaßen: *‚awareness is an understanding of the activities of others, which provides a context for your own activity‘* [Dourish 92]. Anwendungen, die als echte Groupware–Systeme konzipiert worden sind, besitzen die Eigenschaft der *Collaboration Awareness*. Im Gegensatz arbeiten Systeme, die eine gemeinsame Benutzung existierender ehemaliger Einzelbenutzer–Applikationen wie beispielsweise Spreadsheets erlauben, im allgemeinen *collaboration transparent* [Lauwers 90a].

Die Awareness–Eigenschaften sind äußerst wesentliche Attribute einer CSCW–Applikation, da ein angemessenes Maß an Maßnahmen zur Erzielung von Gruppenbewusstsein essentiell für den Gruppenprozess ist. Durch eine Verbesserung der Awareness wird einerseits die Aufnahme einer spontanen, informellen Kommunikation zwischen Teammitgliedern über Audio– oder Videokanäle befördert, da Teammitglieder besser über die aktuellen Arbeitsaufgaben und Belastung der Personen im Team informiert sind. Andererseits werden die Teammitglieder besser über den Grad der Zielerreichung der Teamarbeit, aber auch über eventuell aufgetretene Schwierigkeiten besser in Kenntnis gesetzt. Dies führt wiederum zu bewussteren Entscheidungen über die eigenen, nächsten Aktivitäten im Sinne der Erreichung des Teamzieles [Borghoff 98]. Dourish und Bellotti konstatieren zur Bedeutung von Gruppenbewusstsein: ‚Awareness information is always required to coordinate group activities, whatever the task domain‘ [Dourish 92]. Allerdings ist ein wohlbalancierter Einsatz von Maßnahmen zur Erzielung von Awareness nötig, da ein Übermaß zu Störungen und damit zu negativen Auswirkungen auf die Teamarbeit führt. Hudson und Smith sprechen von zwei fundamentalen, dualen

Kompromissen zwischen Awareness und Privatsphäre und zwischen Awareness und Störung der Arbeit [Hudson 96].

Nach Greenberg können folgende vier Typen von Awareness unterschieden werden:

- Informelles Gruppenbewusstsein (informal awareness): bezeichnet das allgemeine Wissen über den aktuellen Gruppenzustand. Dazu gehören beispielsweise Informationen über den Aufenthaltsort anderer Teammitglieder sowie über deren aktuellen Arbeitsgegenstand und die Möglichkeit der Kommunikationsaufnahme mit ihnen. Informelles Gruppenbewusstsein schafft Gelegenheiten zu synchroner Kooperation und hilft, Kooperationspartner zu finden [Gutwin 96a]. Räumliche Awareness–Modelle und deren Erweiterung an nichträumliche Domänen [Rodden 96] betreffen vor allem informelles Gruppenbewusstsein.
- Arbeitsgegenstands–Gruppenbewusstsein (workspace awareness): umfasst Wissen über den aktuellen Zustand von Gegenständen der Teamarbeit sowie über aktuelle Zugriffe und Änderungen an den gemeinsamen Informationsbeständen. Workspace Awareness hilft bei der Koordinierung von Aufgaben und Ressourcen im Team und beim Übergang von individuellen zu gemeinsamen Aktivitäten
- Gruppenstrukturbewusstsein (group–structural awareness): ist Wissen über die Gruppenzugehörigkeit und die Rollen und Verantwortlichkeiten anderer Teammitglieder.
- Soziales Gruppenbewusstsein (social awareness): bedeutet Kenntnis über den sozialen Kontext der Gruppe, also beispielsweise über Motivation, emotionalen Zustand aber auch Fähigkeiten und Kenntnisse anderer Teammitglieder.

Vor allem die beiden ersten Typen von Awareness sind ein aktueller Forschungsgegenstand. Informelles, Arbeitsgegenstands– und bedingt Gruppenstruktur–Gruppenbewusstsein lassen sich gut durch Systemfeatures von CSCW–Applikationen unterstützen. Viele der unten näher beschriebenen klassischen CSCW–Techniken zur Kooperationsunterstützung bewirken implizit eine Förderung der Awareness. Hier sollen kurz einige Techniken genannt werden, deren primäres Ziel die Awareness–Steigerung ist. Diese Techniken sind meist Gestaltungsmaßnahmen der graphischen Nutzeroberfläche, also spezielle Widgets und spezifische Awareness–Windows:

- *Präsenz–Indikatoren* sind ikonische Darstellungen, die Auskunft über die aktuelle Aktivität der Teammitglieder geben. In Peepholes [Greenberg 96a] existieren beispielsweise Icons für eingeloggte, aktive Teammitglieder, für seit wenigen Minuten inaktive Mitglieder, für nicht ausgeloggte, aber seit einiger Zeit inaktive Personen und für ausgeloggte Teammitglieder. Diese Icons können auch durch aufwendigere Techniken wie periodisch aktualisierte Bilder wie in Portholes [Dourish 92a] oder durch ständige Video–Ubertagung ersetzt werden.
- *Radar–Views* sind Darstellungen des gesamten gemeinsamen Arbeitsbereichs, bei denen die für die einzelnen Teammitglieder sichtbaren Ausschnitte hervorgehoben werden. Diese Darstellungen können auch mit Portraitdarstellungen der Teammitglieder hinterlegt werden [Gutwin 96c]. Radar–Views sind vor allem für Groupware–Systeme geeignet, bei denen sich

leicht Übersichtsdarstellungen des gemeinsamen Arbeitsgegenstandes erzeugen lassen, wie dies beispielsweise beim kollaborativen Edieren von Dokumenten oder bei 2D–Zeichentools der Fall ist.

- *Fisheye–Views* stellen das gesamte Dokument in einem Fenster dar, wobei die aktuellen Arbeitspositionen der Teammitglieder in einer graphisch hervorgehobenen Weise in normaler Größe dargestellt werden. Von diesen Bereichen ausgehend werden die anderen Bereiche (evtl. sukzessive) verkleinert, so dass ein ständiger Überblick über das Gesamtdokument und die Aktivitäten des Teams gewährleistet wird [Greenberg 96b]. Es ist augenscheinlich, dass diese Technik im wesentlichen für kollaboratives Dokumentenedieren in kleinen Teams geeignet ist.
- *Multi–User–Scrollbars* stellen die Position der Bildlaufleisten (Scrollbars) der anderen Teammitglieder neben der eigenen aktiven Scrollbar in einer graphisch abgesetzten Weise dar.
- *What–You–See–Is–What–I–Do* bezeichnet vereinfachte Darstellungen des Arbeitskontexts eines Teammitglieds. Dabei wird die Darstellung bei Mausbewegungen immer so verschoben, dass der Mauszeiger des Teammitglieds sich im Zentrum der Darstellung befindet.
- *Tickertape* ist eine einzeilige Darstellung eines Lauftextes, mit der Nachrichten an Gruppenmitglieder weitergegeben werden können. Dabei sind unidirektionale Anwendungen möglich, bei denen nur eine verantwortliche Person Nachrichten einspeist, oder bidirektionale Anwendungen, bei denen alle Teammitglieder Informationen an das Team über dieses Medium weitergeben können [Fitzpatrick 98].

Nach Fuchs et al. existieren zwei orthogonale Kriterien zur Einordnung von Ereignissen, nach denen vier mögliche Awareness–Modi identifiziert werden können. Synchrone Ereignisse ereignen sich gerade, während asynchrone Ereignisse in der Vergangenheit stattfinden. Unter dem Kopplungsgrad wird die Bedeutung der Tätigkeit der anderen Teammitglieder für die aktuellen Arbeitsschritte ausgedrückt. Gekoppelte Awareness bedeutet, dass eine enge Verbindung zwischen den Aktivitäten von Teammitgliedern vorliegt, wie dies beispielsweise bei gemeinsamen Edieren eines Dokuments der Fall ist. Bei ungekoppelter Awareness möchte der Nutzer über Aktivitäten und Ereignisse des Teams unterrichtet sein, diese haben aber keinen direkten Einfluss auf seine aktuelle Arbeit. So kann das Erfüllung einer Teilaufgabe durch ein anderes Teammitglied von grundlegender Bedeutung für die eigene Arbeit sein, sie muss deswegen aber nicht zur zwangsläufigen und sofortigen Unterbrechung der aktuellen Aktivität führen.

Nach diesen Orientierungsmodi kann der Einfluss von Ereignissen und Vorgängen auf das Gruppenbewusstsein unterschieden werden. Daher ist es wichtig, dass Groupware–Systeme diese Modi unterschiedlich behandeln, um eine Informations–Überfrachtung von Teammitgliedern zu verhindern.

	synchron	asynchron
gekoppelt	Was passiert gerade im aktuellen Fokus der Tätigkeit?	Was hat sich am aktuellen Arbeitsgegenstand seit dem letzten Zugriff geändert?
ungekoppelt	Welche interessanten Ereignisse geschehen gerade irgendwo im Team?	Gab es in letzter Zeit irgendwelche interessanten Ereignisse irgendwo im Team?

Tabelle 1: Orientierungsmodi des Gruppenbewusstseins nach Fuchs et al.

2.1.4 Klassifikationsansätze

Groupware–Systeme können nach verschiedenen Kriterien klassifiziert werden. Einige häufiger angewandte Kriterien sind [Borghoff 98] [Teufel 95]:

- *Geographische Verteilung:* es wird zwischen lokalen, d.h. räumlich benachbarten und verteilten, d.h. räumlich verteilten Systemen unterschieden.
- *Zeitliche Verteilung:* es wird zwischen Systemen unterschieden, die entweder synchrone oder asynchrone Teamkommunikation unterstützen.
- *Teamgröße:* die Anzahl der Personen im Team hat Einfluß auf die Auswahl sinnvoller bzw. möglicher Kommunikations– und Kooperationstechnologien. Bei zwei Personen wird von bilateraler, bei mehreren Personen von multilateraler Kooperation gesprochen.
- *Genutzte Interaktionsmedientypen:* es wird zwischen der Realisierung der Übertragung bestimmter, verschiedene Sinne ansprechenden Informationstypen oder direkter face–to–face– Kommunikation unterschieden. Einige Beispiele sind Audio–, Video–, Grafik– oder auch taktile Informationen.
- *Organisatorische Klassifikation:* es wird zwischen der Unterstützung von face–to–face–Sitzungen und räumlich verteilten elektronischen Rechnerkonferenzen unterschieden.
- *Soziale Klassifikation*: es wird unterschieden, ob die Kommunikation im Team bzw. die Vergabe und Unterscheidung der Rollen in der Gruppe formell oder eher informell abläuft.
- *Zugrundeliegendes Gruppenprozessmodell*[4]*:* es können drei Kategorien von Gruppenprozessmodellen unterschieden werden: zentrale, verteilte nichtreplizierende sowie verteilte replizierende Gruppenprozessmodelle.
- *Kommunikation in der Gruppe:* es kann sowohl nach der Richtung des Kommunikationsflusses, d.h. unidirektional bzw. bidirektional und der Kardinalität der Kommunikationsverbindungen mit den Möglichkeiten 1:1, 1:m, n:1 und m:n unterteilt werden.

[4] Die Begriffe des Gruppenprozesses und des Gruppenprozessmodells werden im Kapitel 1.3.1. näher betrachtet.

- *Grad der Handlungsinterdependenzen:* besteht eine größere Anzahl von wechselseitigen Abhängigkeiten zwischen den Aktivitäten des Teams, liegt eine konjunktive Kooperation vor, ist dieser Abhängigkeitsgrad gering, spricht man von disjunktiver Kooperation.

- *Grad der Beschränkung der Nutzeraktivitäten:* es kann zwischen restriktiven und permissiven Groupware–Systemen unterschieden werden. Erstere beschränken oder lenken die Aktionen der Benutzer mit dem System, da sie häufig ein Modell möglicher oder wünschenswerter Arbeitsabläufe enthalten, das die in der Nutzung ausgeführten Aktionen vorschreibt oder beschränkt. Workflow–Management–Systeme gehören oft zu dieser Gruppe. Permissive Systeme gestatten jedem Nutzer jede Aktion zu jeder Zeit. Sie übertragen den Nutzern die Koordination und erlauben die Ausprägung sozialer Protokolle. Beispiele für permissive Systeme sind Konferenzsysteme oder Shared Whiteboards. Die Termini ‚restriktiv' und ‚permissiv' müssen als Extremwerte einer kontinuierlichen Skala der Beschränkung von Nutzeraktivitäten angesehen werden. Reale Groupware–Systeme können sowohl restriktive als auch permissive Aspekte enthalten.

- *Strukturiertheit der Problemlösungsstrategie:* Bei der Unterstützung von hochstrukturierten Problemlösungsprozessen spricht man von rigider Kooperation, liegen schwach strukturierte Prozesse vor, so wird die Kooperation als lose bezeichnet. Sind die organisatorischen Rahmenbedingungen der Kooperation a priori vorgegeben, führt dies zu rigider Kooperation. In Systemen mit rigider Kooperation existiert meistens eine zentralisierte Koordinierung durch das System oder ein definiertes Teammitglied. Lose Kooperation erfordert Koordinationsbeziehungen und damit Kommunikation zwischen allen an der Teamarbeit beteiligten Personen. Systeme zur Unterstützung rigider Kooperationsprozesse tendieren dazu, restriktiv umgesetzt zu werden, während schwach strukturierte Prozesse eher zum Design permissiver Systeme führen.

- *Verwendetes Koordinationsmodell:* Es kann zwischen Systemen mit formular–, vorgangs–, konversations[5]– oder kommunikationsorientierten Koordinationsmodellen unterschieden werden.

---

[5] Konversationsorientierte Koordinationsmodelle sind eine Umsetzung der Sprechakttheorie nach Austin. Diese nimmt an, dass Menschen durch die Sprechakte handeln [Winograd 88]. Unter einem lokutionären Akt wird die Artikulation eines Satzes verstanden, die im Satz enthaltene Absicht ist ein illokutionärer Akt. Das resultierende Verhalten des Empfängers nennt man perlokutionären Akt. Für konversationsorientierte Koordinationsmodelle sind im allgemeinen illokutionäre Akte von Interesse, bei denen es die Typen *Assertive*, *Directive*, *Commissive*, *Declaration* und *Expressive* gibt. Im Action Workflow–Ansatz [Medina–Mora 92] werden die einzelnen Sprechakte als Arbeitsschritte aufgefasst, aus denen in einem mehrstufigen Prozess ein ‚Business Process Map' genanntes Workflow–Schema erstellt wird. Der Action Workflow–Ansatz ist vor allem zur Modellierung von Soll–Zuständen für betriebliche Abläufe gut geeignet, demgegenüber lassen sich nicht in jedem Falle existierende Workflows abbilden [Teufel95].

- *Art der Team–Kommunikation bzw. –Interaktion (explizit vs. implizit):* Es wird zwischen Systemen mit Unterstützung expliziter und impliziter Kommunikation unterschieden. Explizite Kommunikation liegt beim aktiven und direkten Informationsaustausch zwischen Teammitgliedern über einen geeigneten Kanal vor. Findet die Informationsübertragung über das Medium formalisierter und evtl. systemspezifischer Objekte bzw. eines gemeinsamen automatisiert verwalteten Informationsbestandes statt, spricht man von impliziter Kommunikation. Groupware–Systeme, welche explizite Kommunikation unterstützen, erfüllen das Kriterium der ‚Cooperation Awareness'. Diese Klassifikation ist mit der Unterscheidung von Information Sharing und Information Exchange verwandt [Rüdebusch 93].

- *Information Sharing vs. Information Exchange:* es kann zwischen Systemen unterschieden werden, bei denen Teammitglieder auf einer gemeinsamen Informationsmenge operieren (Information sharing) und solchen, die auf dem Austausch definierter Informations(teil–) mengen basieren. Bei Systemen, die auf Information Sharing beruhen, werden Interaktionen der Teammitglieder durch das System interpretiert. Die Resultate dieses Prozesses werden im Informationsbestand des Systems abgelegt und anderen Teammitgliedern über die Nutzerschnittstelle präsentiert. Im Gegenteil dazu werden bei Systemen, die auf Information Exchange beruhen, Aktionen eines Teammitglieds wie Sprechen, Gestikulieren oder Interaktionen mit der Nutzerschnittstelle transparent an andere Nutzer übertragen. Dabei finden keine Interpretationen der Informationen durch das System statt bzw. nur solche, die zum reinem technischen Übertragungsprozess nötig sind. Diese Klassifikation ist jedoch problematisch, da eine Dualität zwischen Information Sharing und Informationsaustausch besteht: der Versand einer Nachricht kann dem Gestatten des Zugriffs auf gemeinschaftliche Informationen bzw. eine Kopie derer angesehen werden. So werden beispielsweise Computer–Konferenzsysteme häufig der einen oder anderen Kategorie zugeordnet. Ein anderes Klassikationsproblem tritt auf, wenn es Anwendern möglich ist, eine größere Anzahl von Manipulationen am gemeinsamen Datenbestand vorzunehmen, bevor dieser evtl. auf Nutzeranforderung re–synchronisiert wird. Diese Re–Synchronisation wird den Anwendern häufig als ‚Update'– oder ‚Sende' – bzw. ‚Austausch'– Funktionalität präsentiert [Hofte 98].

- *3–K Modell:* CSCW–Systeme können nach dem Grad der Unterstützung von Kommunikation, Koordination und Kooperation klassifiziert werden. Die Art und Intensität der zu unterstützenden Tätigkeit im Team hat Einfluss auf die Betonung bestimmter Aspekte. Kommunikationsunterstützung dient primär zur Verständigung von Personen durch Informationsaustausch in der Gruppe. Die Unterstützung von Koordination dient dazu, die Aktivitäten und Ressourcennutzung von Gruppenmitgliedern in Einklang zu bringen. Kooperation in der Gruppe impliziert die Verfolgung gemeinsamer Gruppenziele. Die funktionale Klassifikation nach Sauter, Teufel, et al. baut auf diesem Modell auf [Teufel 95], [Sauter95].

Eine sehr breite Anwendung findet die Klassifikation der Groupware–Systeme nach Raum– und Zeit–Varianz. Diese basiert auf der ‚Geographic and Time Dispersion Matrix' nach Johansen, der allerdings erkannte, dass künftig interessante Bereiche in den Bereichen der Übergänge der Quadranten liegen. Grudin erweiterte diese Matrix in der Weise, dass die Fälle der geographisch verteilten sowie der asynchronen Kooperation nach dem Kriterium der Vorhersehbarkeit des Ortes bzw. der Zeit weiter klassifiziert werden [Johansen88] [Grudin 94].

Zeit / Raum		Lokal	Verteilt	
			Ort vorhersehbar	Ort nicht vorhersehbar
Synchron		‚face–to–face' Sitzungsunterstützung	Videokonferenz	Mobilfunkkonferenz
Asynchron	Zeit vorhersehbar	Schichtarbeit	E–Mail	Bulletin–Board
	Zeit nicht vorhersehbar	Schwarzes Brett	Kollaborative Dokumentenbearbeitung	Workflow–Unterstützung

Tabelle 2 : Groupware Time–Space Matrix nach Grudin

Existierende Groupware–Systeme bauen jedoch sehr häufig auf verschiedenen Basistechnologien, wie beispielsweise elektronischer Post und kollaborativer Dokumentenbearbeitung auf, so dass eine Einordnung in eine der Kategorien der oben genannten Matrix nicht in jeden Falle möglich ist. Ausgehend von dieser Problematik wurde von Sauter et al. eine Klassifizierung nach dem Grad der Unterstützung der Funktionalitäten Kommunikation, Kooperation und Koordination eingeführt [Sauter 95].

In der oben genannten Klassifikation lassen sich CSCW–Applikationen in vier grundlegende Systemklassen einteilen, wobei diese in bezug auf konkrete Groupware–Systeme keineswegs als disjunkt anzusehen sind:

- *Kommunikationssysteme:* dienen vor allem dem expliziten Informationsaustausch zwischen Teammitgliedern, in dem Raum– und/oder Zeitdifferenzen überwunden werden. Repräsentanten dieser Systemklasse sind Mail– und Konferenzsysteme, die Medientypen wie Text, Audio oder Video verarbeiten können.
- *Gemeinsame Informationsräume:* unterstützen die implizite Kommunikation zwischen Teammitgliedern und dienen zur eher längerfristigen Informationsspeicherung. Zu dieser Klasse werden Bulletin–Board–Systeme, verteilte Hypertextsysteme, spezielle gemeinsam genutzte multiuserfähige Datenbanken wie Kalendersysteme gezählt.

- *Workflow Management:* diese Klasse enthält Systeme, welche zur Ausführung und Koordination von Workflows dienen. Workflows sind häufig wiederkehrende und wohlstrukturierte Arbeitsabläufe, die aus einer endlichen Zahl von Aktivitäten bestehen (siehe Kap.2.2.3). Koordinationsfunktionalitäten werden auf Grundlage fester Organisationsregeln mit Hilfe von Prozeßdefinitionswerkzeugen spezifiziert. Mit Workflow–Überwachungs–Werkzeugen (Workflow Monitoring Tools) können die Teilnehmer Informationen zu verschiedenen Aspekten des Arbeitsablaufs abrufen. Zur Implementierung von Applikationen dieser Systemklasse werden im allgemeinen spezielle Mail– und Datenbank–Managementsysteme genutzt.
- *Workgroup Computing:* diese Applikationen unterstützen komplexe Arbeitsaufgaben mit mittlerem bis geringem Strukturierungs– und Wiederholungsgrad, die kooperativ erfüllt werden müssen. Der Fokus von Workgroup–Computing liegt bezüglich der unterstützten Funktionen im Bereich der kollaborativen Prozesse, also bei zielorientierter Kooperation. Wichtige Vertreter dieser Klasse sind elektronische Sitzungs– und Entscheidungsunterstützungsysteme, Planungssysteme und kooperative Dokumenten– und Zeichnungseditoren.

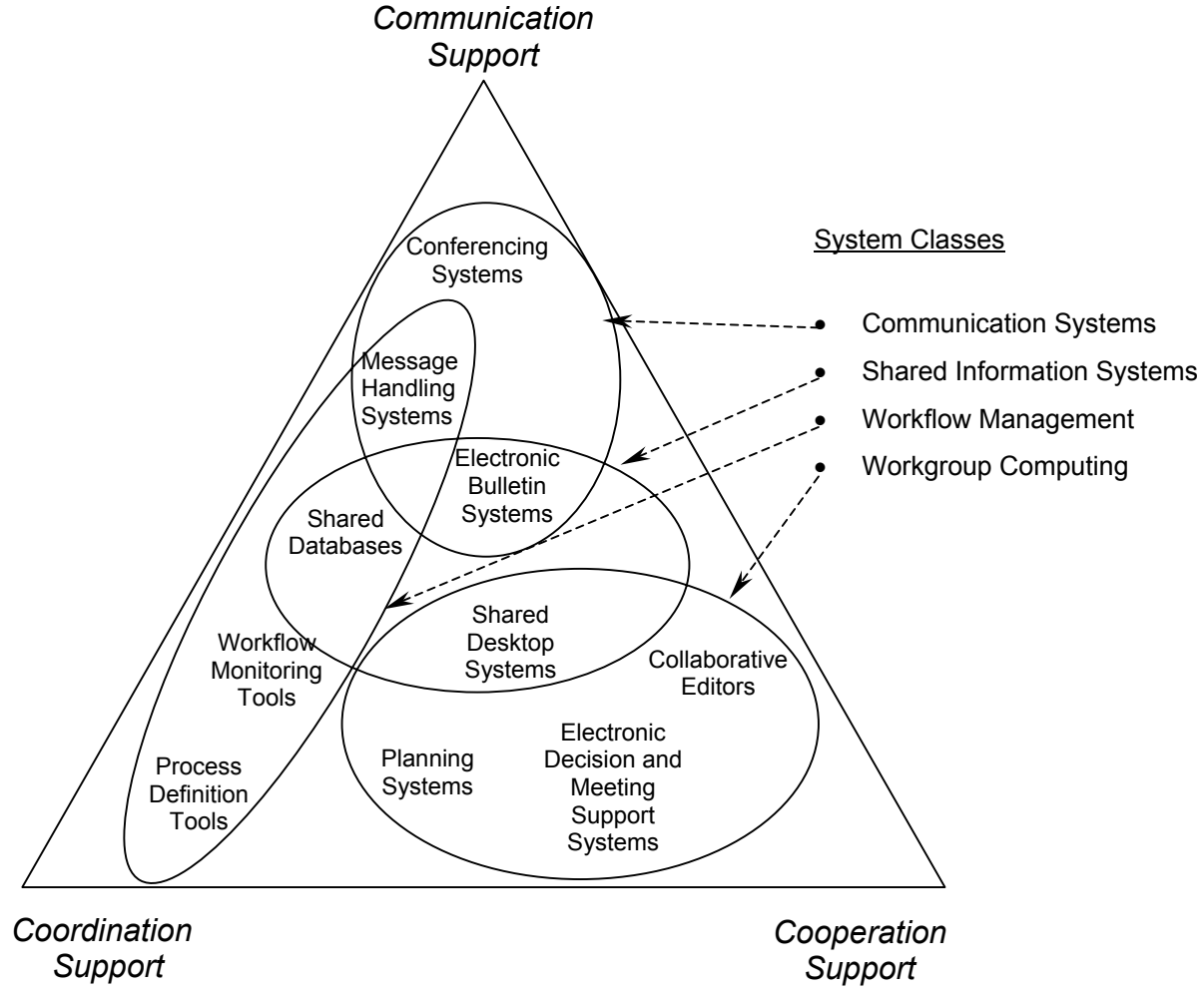

Abbildung 2 : Funktionale Klassifikation von Groupwaresystemen [Sauter 95]

2.2 Groupware- Systemtypen und Applikationen

Im folgenden Abschnitt sollen die im Abschnitt 2.1.4 vorgestellten Typen von Groupware–Systemen sowie einige bekannte konkrete CSCW–Applikationen näher vorgestellt werden. Die Systemtypen sollen im Kontext ihrer Zugehörigkeit zu den oben eingeführten CSCW–Systemklassen nach Sauter, Teufel et al. Kommunikation, Gemeinsame Informationsräume, Workflow Management sowie Workgroup Computing behandelt werden. Die möglicherweise fünfte Systemklasse der Agentensysteme soll im Rahmen dieser Arbeit nicht näher behandelt werden.

2.2.1 Kommunikationssysteme

Kommunikationssysteme zählen zu den ältesten und derzeit erfolgreichsten CSCW–Applikationen überhaupt. Es existieren Kommunikationsunterstützungssysteme sowohl für synchrone und asynchrone Teamarbeit als auch für lokale und geographisch verteilte Kooperation. Asynchrone Kommunikationssysteme gehören meist zu einer Form der elektronischen Postsysteme, während die synchronen Systeme häufig eine Unterstützung von Treffen und Konferenzen bieten. In der Klassifikation nach Sauter, Teufel et al. gehören die Bulletin–Board–Systeme sowohl zu den Kommunikationssystemen als auch zu den gemeinsamen Informationsräumen. Da diese auf der für gemeinsame Informationsräume typischen impliziten Kommunikation beruhen, sollen sie dort behandelt werden.

Elektronische Post–Systeme (EMail–Systeme)

E–Mail ist die derzeitig erfolgreichste CSCW–Applikation überhaupt. Auf der Mehrzahl der Personalcomputer mit direktem Anschluss an ein Rechnernetz (LAN, MAN, WAN) oder mit Zugangsmöglichkeiten über andere Kommunikationsmedien wie beispielsweise Telefonleitungen existiert eine E–Mail–Software.

Zu den grundlegenden Diensten, die alle Systeme für elektronische Post realisieren, gehört die Komposition, also das Erstellen einer neuen Nachricht. Diese kann aus Text bestehen und als Anhang Dateien beliebiger Formate sowie unter Umständen eine Prioritätsangabe enthalten. Diese Anhänge müssen aus Übertragungsgründen kodiert werden, wozu häufig der Standard MIME (Multipurpose Internet Mail Extension) verwendet wird. Daher muss ein E–Mail–System die entsprechenden Konvertierungen und möglichst auch alte Formate beherrschen. Weiterhin sind häufig Funktionalitäten zur Übertragung, also zum Senden und Empfangen von Nachrichten sowie zur Benachrichtigung über neue Nachrichten vorhanden. Weitere häufige Features sind eine Formatierung der Nachricht entsprechend der Window– Größe des Empfängers und die Disposition der Nachrichten durch Einordnen in bestimmte Nachrichtenordner. Zur Adressierung werden zwei Schemata häufig eingesetzt, das vom Internet bekannte *Domain Addressing* sowie die Adressierung nach dem OSI–Modell.

Konferenzsysteme

Konferenzsysteme unterstützen eine Menge von genau adressierbaren Partnern bei der synchronen, expliziten Kommunikation. Typischerweise sind die Teilnehmer geographisch verteilt und werden eingesetzt, wenn der persönliche Kontakt der Teammitglieder zum Erreichen des Ziels der Gruppenarbeit beiträgt.

Zu den Konferenzsystemen zählen textbasierte Konferenzsysteme sowie Audio– und Videokonferenzsysteme. Textbasierte Konferenzsysteme können auf eine vergleichsweise lange Geschichte zurückblicken, da schon in frühen Versionen des Betriebssystems UNIX® das Systemprogramm *talk* existierte. Ein aktueller Vertreter ist das über das TCP/IP– Protokoll arbeitende Internet Relay Chat (IRC), dass auch derzeit dank der Verbreitung des Internets stark genutzt wird. Dabei werden entweder die Bildschirme der einzelnen Teilnehmer horizontal in einen Bereich pro Teilnehmer geteilt und alle Eingaben buchstabenweise in die entsprechenden Bereiche aller Teilnehmer repliziert oder die Kommunikation erfolgt durch chronologische Darstellung der Äußerungen der Teilnehmer in einem gemeinsamen Fenster. Dabei wird zu jeder Äußerung der Teilnehmername dargestellt, teilweise werden die Texte der Konferenzteilnehmer auch farblich differenziert dargestellt.

Audio– bzw. Telefonkonferenzen sind mit der normalen Infrastruktur von Büroumgebungen zusammen mit spezifischen Diensten der Telefongesellschaften relativ einfach zu realisieren. Ebenso ist es möglich, diese Konferenzen mittels spezieller Telefoniesoftware über Weitverkehrs–Rechnernetze auszuführen. Trotzdem werden diese Techniken selten bei mehr als zwei Teilnehmern eingesetzt. Gründe dafür liegen im schwer steuerbaren Gesprächsablauf, in der nicht erkennbaren Mimik der Teilnehmer und bei größeren Gruppen in der Schwierigkeit herauszufinden, wer aktuell teilnimmt und spricht [Teufel 95].

Videokonferenzen können in drei grundsätzlichen Varianten abgehalten werden. Es können speziell ausgestattete Videokonferenzräume genutzt werden, die im allgemeinen über Großdisplays (Video– Whiteboards) und synchrone Audioübertragung verfügen. Außerdem wird häufig der Bildschirminhalt eines für Präsentationen genutzten Rechners mit übermittelt. Videokonferenzen sind ebenso über Bildtelefone möglich, wobei hier ebenso wie bei Audiokonferenzen nur Konferenzen mit kleinen Gruppen praktikabel sind. Obwohl die ersten Bildtelefone schon 1964 von AT&T vorgestellt wurden, erfolgte bis heute keine flächendeckende Verbreitung. Das kann einerseits an den vergleichsweise hohen Kosten, aber auch an der geringen Nutzerakzeptanz liegen. In einer Studie von 1995 wurde beispielsweise festgestellt, dass das Management von nur circa 10% Firmen in der Schweiz über Möglichkeiten für Videokonferenzen verfügt, von denen die Hälfte diese Techniken als wenig bedeutend einstuft [Sauter 95].

Die dritte Variante von Videokonferenz– Systemen stellen die Desktop– Konferenzsysteme dar. Sie erlauben die Durchführung der Konferenz vom Arbeitsplatz aus, indem die Konferenzteilnehmer in Fenstern der Nutzerschnittstelle dargestellt werden. Auch bei dieser

Technik werden häufig Fensterinhalte parallel mitübertragen. Somit sind Desktop–Konferenzen ein gut geeignetes Mittel zur Kommunikation und Koordination von Teams, besonders zum gemeinsamen Lösen von bei der Arbeit aufgetauchten Problemen, da sie keine gravierenden Unterbrechungen der Arbeit erfordern und ein geeignetes Mittel für Phasen synchroner Kooperation sind.

2.2.2 Gemeinsame Informationsräume

Gemeinsame Informationen (shared information) bilden immer einen Kontext der Teamarbeit, da sie einerseits als Mittel der impliziten Kommunikation dienen und andererseits Zwischen– und Endergebnisse der Kooperation beschreiben. Inhalte gemeinsamer Informationsräume sind typischerweise Texte, mediale Dokumente und Hypertext– Dokumente.

Zu den gemeinsamen Informationsräumen zählen Bulletin–Board–Systeme, Verteilte Hypertext–Systeme und bestimmte Datenbanken.

Bulletin–Board– Systeme

Bulletin–Board–Systeme gehören sowohl zu den Kommunikationssystemen als auch zu den gemeinsamen Informationsräumen. Diese Systeme unterstützen eine natürlichsprachliche Kommunikation zwischen Gruppenmitgliedern, bei der die Informationseinheiten in einem gemeinsamen Informationspool verwaltet werden. Technisch gesehen sind sie spezielle Datenbanken, die eine asynchrone Gruppeninteraktion nach dem Kammmodell unterstützen. Server mit Usenet– Informationen sind das weitverbreiteste Beispiel für öffentliche Bulletin–Board– Systeme; es gibt ebenso auch Systeme mit einer eingeschränkten Teilnehmermenge. Die Informationen in Bulletin–Board–Systemen sind nach Themenschwerpunkten geordnet, die einzelnen Nachrichten weisen ein semistrukturiertes Format ähnlich dem von E–Mail–Nachrichten auf. Derartige Systeme bieten eine schnelle Verfügbarkeit von Informationen für verteilte Teams und können für Ankündigungen, Hilfegesuche, Inserate aber auch zu asynchronen Diskussionen über bestimmte Teilaspekte des Themenschwerpunkts genutzt werden.

Die technische Realisierung als auch die Benutzung von Bulletin– Board– Systemen ähnelt stark Systemen für elektronische Post. Das hat dazu geführt, dass einige aktuelle Softwarepakete die Funktionalität für E–Mail und Usenet in einer Komponente verschmolzen haben. Beiträge für Bulletin–Boards werden im allgemeinen mit einem E–Mail–Client erstellt und dann an einen entsprechenden Server übermittelt. Im Falle des Usenets werden die Nachrichten sukzessize durch einen Flood–Fill–Mechanismus an benachbarte Server weitergegeben. Der Empfang der Nachrichten erfolgt durch News–Reader wieder ähnlich zur E–Mail mit dem Unterschied, dass vom Server die neuen Nachrichten des Themenschwerpunkts und nicht die des einzelnen Nutzers abgerufen werden.

Neben dem Usenet sind Bulletin–Boards häufig bei Herstellern von Hard– und Softwaresystemen zu finden, um den Anwendern ein durch Fachleute betreutes Diskussionsforum zur Verfügung zu stellen. Ebenso nutzen Großfirmen mit mehreren Standorten derartige Systeme für bestimmte interne Kommunikationszwecke. Im allgemeinen wird auch in diesen Fällen normale Usenet– Software genutzt, aber durch den Server unter Umständen eine Zugriffsberechtigung geprüft.

Bei Bulletin–Board– Systemen findet grundsätzlich implizite Kommunikation statt, da der Autor nicht die Menge der konkreten Leser bestimmen kann, daher liegt hier eine 1:m Kommunikationsbeziehung vor.

Verteilte Hypertextsysteme

Auch Verteilte Hypertextsysteme wie das World Wide Web (WWW) und die hypertextbasierenden Inhalte von Intranets größerer Firmen können als gemeinsame Informationsräume angesehen werden. Der Ursprung dieser Systeme liegt im nie umgesetzten Memex–Konzept von Vannemar Bush aus dem Jahre 1945, das zum schnellen Zugriff auf Textinformationen über ein Indizierungsschema dienen sollte. Der Ursprung des World Wide Webs liegt bei einem Projekt des Europäischen Laboratoriums für Partikelphysik CERN in Genf vom Anfang der neunziger Jahre. Grundgedanke war die Schaffung eines weltweit zugänglichen Pools menschlichen Wissens. Technisch gesehen ist das WWW eine verteilte Client–Server – Applikation. Auf dem WWW– Server läuft ein Prozess, der Anfragen nach definierten Dokumenten über das Hypertext Tranfer Protokoll (HTTP) bearbeitet. Der WWW–Client (Browser) stellt diese Dokumente dar. Die Hypertext–Dokumente werden im WWW derzeit im Format Hypertext Markup Language (HTML) erstellt, neue Formate wie XML befinden sich derzeit in der Normung.

Vorteile der Nutzung von Hypertext für gemeinsame Informationsräume sind die relativ große Mächtigkeit des Konzepts, die intuitiven Nutzungsmöglichkeiten auch für Anwender mit geringen Computerkenntnissen und die weitverbreitete Infrastruktur zur Nutzung von HTML–basierten Hypertextdokumenten.

Spezielle Datenbanken für gemeinsame Informationsräume

Sollen gemeinsame Informationsräume realisiert werden, benötigt man Mechanismen zur effizienten Verwaltung und persistenten Speicherung großer Datenmengen. Die Erfüllung dieser Anforderungen ist die grundlegende Aufgabe von Data Base Management Systems (DBMS). Leider erfüllen viele DBMS nicht die spezifischen Anforderungen von CSCW–Applikationen in bezug auf die Verwaltung von Zugriffsrechten, auf die Abspeicherbarkeit medialer Informationen, auf die Behandlung nebenläufiger Zugriffe, auf die Verteilbarkeit und die Replizierbarkeit der Informationen etc. Teufel et al. nennen Non–Standard–Datenbanken, Verteilte Datenbanken und Förderative Datenbanken als Teilgebiete der DBMS–Technik, in denen sich Entwicklungen abzeichnen, die auch in bezug auf den möglichen Einsatz in Groupware–Systemenm vielversprechend sind [Teufel 95]. Unter Non–Standard–Datenbanken werden Systeme

verstanden, die anstelle der vergleichsweise kleinen Tupeln üblicher betriebswirtschaftlicher Anwendungen komplex strukturierte, möglicherweise sehr große Tupel verwalten können. Typische Anforderungen an Non–Standard–Datenbanken sind Objektorientierung, Mengenorientierung, die Erweiterbarkeit um nutzerdefinierte Datentypen und die Nutzbarkeit in verteilten Umgebungen.

Unter Verteilten Datenbanken werden DBMS–Systeme verstanden, die eine Aufteilung der Informationsbestände auf mehrere Knoten eines Rechnernetzwerks zulassen. Dabei müssen Transaktionen über mehrere Knoten ausführbar und der gesamte Datenbestand von einem Rechner aus erreichbar sein. Zur Verbesserung des Interaktionsverhaltens kann ein Replikationsmechanismus bereitgestellt werden.

Förderative Datenbanken entstehen aus der Vereinigung mehrerer lokaler, heterogener DBMS einschließlich ihrer Datenbestände. Den Nutzern wird eine vereinheitlichende Schnittstelle zur Verfügung gestellt, mit der eine uniforme Nutzung des Gesamtdatenbestands möglich und die Illusion eines einzelnen, homogenen Informationsbestandes vermittelt wird. Um auf den einbezogenen DBMS lokal arbeitende Legacy–Applikationen nicht zu gefährden, bleibt die Autonomie der einzelnen Systeme erhalten.

2.2.3 Workflow Management Systeme

Workflow Management Systeme überwachen und steuern die Ausführung der Einzelaktivitäten der Teamarbeit unter Berücksichtigung kausaler und temporaler Abhängigkeiten sowie des Ausführungskontexts. Unter einem Workflow wird eine endliche Folge von Aktivitäten verstanden, wobei die einzelnen Aktivitäten durch auftretende Ereignisse ausgelöst und beendet werden. Der Begriff Geschäftsprozess wird synonym gebraucht. Workflow Management bezeichnet die Menge der Aufgaben, die bei der Modellierung, Simulation, Steuerung und Ausführung von Workflows ausgeführt werden müssen. Workflow Management Systeme sind somit Softwaresysteme, welche die Aufgaben des Workflow Managements unterstützen.

Erste Prototypen für Workflow Management–Systeme existieren seit Ende der siebziger Jahre, frühe Systeme waren *Scoop* von Zisman und *OfficeTalk–D* von Ellis und Bernal. Aufgrund der zu dieser Zeit unflexiblen Definitionsmöglichkeiten für Workflows und der ungenügenden Infrastruktur infolge der geringen Verbreitung von Personalcomputern und Netzwerken führten diese nicht zu kommerziellen Systemen. Bei der Entwicklung von Workflow Management–Systemen wurden von Abbot und Sarin bislang vier Generationen festgestellt [Abbot 94]. Die Systeme der ersten Generation sind als anwendungsgebietsspezifisch zu bezeichnen und basierten auf statischen Prozessdefinitionen für diese bestimmte Applikation. Systeme der zweiten Generation waren eigenständige Anwendungen, bei denen die Workflow–Funktionalität von der Logik des Anwendungsbereichs getrennt wurde. Die Systeme konnten dadurch in verschiedenen Domänen eingesetzt werden, weiterhin kamen erste Skriptsprachen zur Geschäftsprozessdefinition auf. Die dritte Generation existiert seit Anfang der neunziger Jahre

und zeichnet sich durch gut anpassbare Dienste und deren Nutzung durch Programmschnittstellen (API's) aus. Die Architekturen sind im allgemeinen offen und basieren auf Standards, Produkte von Drittherstellern sind integrierbar und zur Definition der Workflows existieren graphisch arbeitende Tools. Allerdings besitzen die Systeme der dritten Generation proprietäre Schnittstellen und Austauschformate. Von der vierten Systemgeneration wird erwartet, dass eine volle Integrierbarkeit der Systeme in bestehende Arbeitsumgebungen möglich wird. Zusammen mit elektronischer Post, Arbeitsoberflächen und Verzeichnisdiensten entsteht ein Middleware–Dienst, auf den aus den unterschiedlichsten Anwedungen zugegriffen werden kann. Abbot und Sarin charakterisieren die Systeme der vierten Generation als ‚allgegenwärtig aber unsichtbar'.

Die Entwicklung von Standards für Workflow–Management–Systeme wird besonders von der *Workflow–Management Coalition (WfMC)* vorangetrieben. Diese wurde 1993 gegründet und hat das Ziel, die Anwendung von Workflow Management Systemen über eine vereinheitlichte Terminologie und eine Interoperabilität der Systeme zu fördern. Ein Kernvorhaben ist dabei die Entwicklung eines Referenzmodells (Workflow Management Coalition Reference Model) mit fünf Schnittstellen der Workflow–Laufzeitkomponente zu externen Applikationen, Workflow–Client–Applikationen, Workflow–Modellierungs– und Administrationswerkzeugen sowie zu weiteren Workflow–Laufzeitkomponenten [WfMC 95].

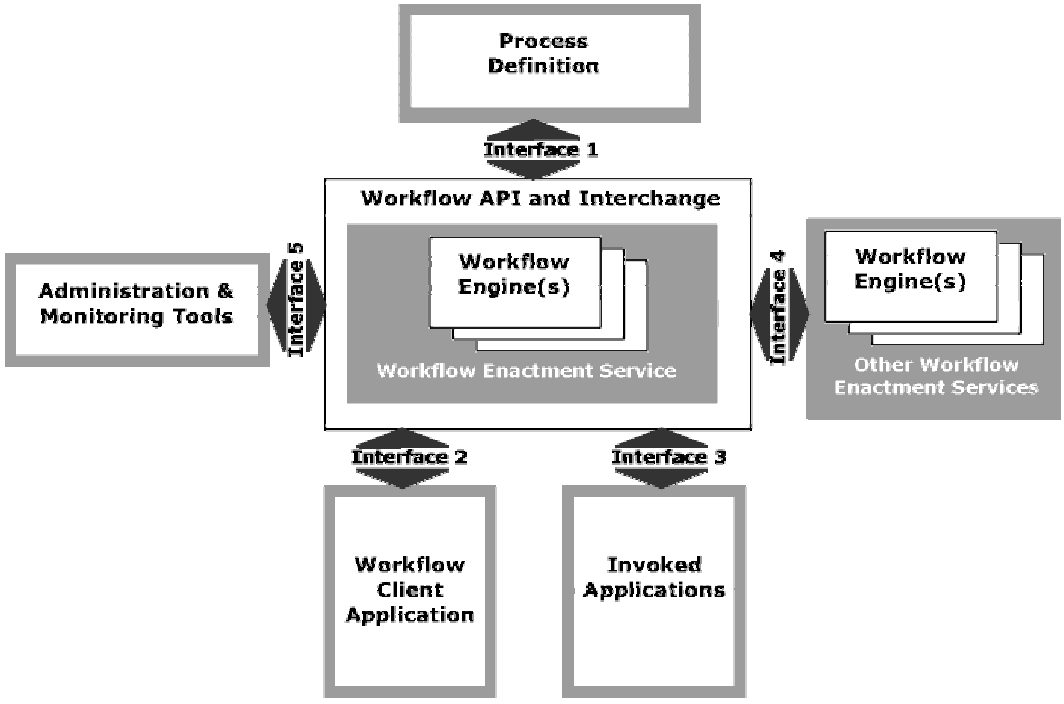

Abbildung 2: Workflow Management Coalition Reference Model nach WfMC

Workflow–Management–Systeme können dann gut eingesetzt werden, wenn Abläufe vorliegen, in die eine größere Anzahl Personen oder Applikationen einbezogen werden müssen. Weiterhin müssen die Abläufe einen hohen Strukturierungsgrad und eine geringe Komplexität haben. Ein letztes notwendiges Kriterium ist eine hohe Wiederholungsfrequenz und eine geringe Anzahl von

nötigen Mechanismen zur Behandlung von Ausnahmesituationen. Sind diese notwendigen Bedingungen erreicht, kann eine kürzere und optimierte Bearbeitung in den Geschäftsprozessen erreicht werden. Transport– und Liegezeiten von Dokumenten, die sonst einen Großteil der Bearbeitungszeit einnehmen, können drastisch gekürzt werden. Eine mehrfache Erfassung des Inhaltes von Dokumenten kann vermieden werden. Für die Bearbeiter erhöht sich weiterhin die Verfügbarkeit von zur Bearbeitung notwendigen Informationen. Die Effektivität der Prozesse kann ebenso erhöht werden, da neue Abläufe ermöglicht werden.

Beim Einsatz von Workflow–Management–Systemen müssen jedoch Akzeptanzprobleme überwunden werden. Von Seiten des Managements wird häufig eine statische Festschreibung der Geschäftsprozesse und ein großer Einführungsaufwand verbunden mit hohen Kosten befürchtet. Andererseits erzeugt die Funktionalität der Systeme zum exakten Monitoring der Prozesse Akzeptanzprobleme bei den Bearbeitern, die sich einer totalen Erfassbarkeit quantitativer Messgrößen ausgesetzt sehen. Weitere Probleme dieser Systeme erwachsen aus der Ignoranz sozialer Aspekte, die sich in der derzeitig oft technologiezentrierten Interaktionsgestaltung zwischen Anwender und System und der teilweise isolierten Sicht der Bearbeiter auf ihre auszuführenden Aktivitäten manifestieren. Teilweise wird vernachlässigt, dass Kooperation bei der Arbeit Koordination und damit Kommunikation zwischen Teammitgliedern erfordert. Die Bedeutung von informeller Kommunikation zwischen Teammitgliedern wird oft unterschätzt, was die isolierte Sicht der Bearbeiter weiter verschärft [Jablonski 95] [Borghoff 98] [Teufel 95].

2.2.4 Workgroup Computing

Workgroup Computing Systeme, auch Kooperationssysteme genannt, unterstützen Prozesse der Teamarbeit, die durch geringe bis mittlere Strukturierungsgrade und vergleichsweise niedrige Wiederholungsfrequenzen gekennzeichnet sind. Daher müssen Workgroup Computing Systeme sehr flexibel in bezug auf den aktuellen Arbeitsgegenstand sein. Häufig spielen Echtzeitanforderungen für eine adäquate Unterstützung synchroner Kooperation eine Rolle. Nach der Klassifikation nach Sauter, Teufel et al. gehören Planungssysteme, Gruppeneditoren, Entscheidungsunterstützungssysteme, Sitzungsunterstützungsysteme und verteilte Hypertextsysteme zur CSCW–Systemklasse ‚Workgroup Computing'. Letztere gehören ebenso der CSCW–Klasse der gemeinsamen Informationsräume an und sind dort schon behandelt worden.

Planungssysteme

Planungssysteme haben das Ziel, für bestimmte Domänen oder Aufgabengebiete einen optimierten Einsatz von Kapazitäten und Ressourcen zu gewährleisten.

Die häufig eingesetzten, traditionellen Produktionsplanungs– und –steuerungssysteme sind nur bedingt für Domänen geeignet, die eine schwache Strukturierung aufweisen, in denen gelegentlich unvorhersehbare Abläufe auftreten, in denen häufig unerwartete Ereignisse auftreten oder in denen eine längerfristige Planung nicht sinnvoll ist. Einen alternativen Ansatz

bieten die CSCW– Ressourcenplanungssysteme, die eine zentrale Planung für derartige Bereiche für wenig praktikabel halten und statt dessen auf eine Verlagerung der Koordination zu den Teams setzen. Ein Beispiel für derartige Systeme ist ein von Egger et al. vorgestelltes Krankenhausmanagement–System, das Funktionen wie E–Mail, Raumbelegungs–, Gerätenutzungs– und Personalkapazitätsplanung und Klinik– Informationssystem vereinigt [Egger 93]. Ein anderes Planungssystem ist COCOS vom FAW Ulm, dass eine Planung und Koordination von Teameinsätzen für durchzuführende Projekte unterstützt. Dabei werden u.a. Fähigkeiten und Kenntnisse von Personen, Zugehörigkeiten zu anderen Projektgruppen, Räume und technische Ressourcen betrachtet.

Die am häufigsten eingesetzte Gruppe von Planungssystemen sind Terminplanungs– und Kalendersysteme. Bei diesen werden die Zeitkapazitäten der Teammitglieder koordiniert, in dem sie zur Verwaltung der Termine der einzelnen Teammitglieder und zur Unterstützung von Terminvereinbarungen innerhalb der Gruppe genutzt werden können. Ebenso kann durch dasselbe System auch die Benutzung von Räumen oder Geräten verwaltet werden.

Grundvoraussetzung des Funktionierens dieser Systeme ist, dass alle Teammitglieder ihren persönlichen elektronischen Kalender ständig führen. Diese Voraussetzung wird durch die zunehmende Verbreitung von Personal Digital Assistants (PDA's) wesentlich erleichtert, da diese tragbaren Computer im Taschenrechnerformat im allgemeinen über eine mit Office–Paketen synchronisierbare Terminverwaltung verfügen. Die privaten Kalender der Teammitglieder sind begrenzt durch das gesamte Team einsehbar. Zur Vereinbarung eines Termins wird zuerst die Menge der potentiellen Teilnehmer des Treffens festgelegt, danach kann durch das System eine Anzahl von Terminvorschlägen ermittelt werden. Über E–Mail–Kommunikation wird dann ein endgültiger Termin bestimmt oder der Termin bestätigt.

Kooperative Dokumentensysteme

Mit kooperativen Dokumentensysteme soll die Bearbeitung von Texten, Zeichnungen, Bildern, etc. im Team unterstützt werden. Dabei lassen diese sich in die drei Gruppen Annotationssysteme, Koautorensysteme und kooperative Zeichentools unterteilen. Annotationssysteme erlauben es, an einem vorliegenden Dokument Randnotizen, Kommentare oder Korrekturen einzufügen. Dies kann durch typographisch hervorgehobene Einfügungen im Originaldokument, in Overlaytechnik auf einer darüber liegenden Ebene geschehen oder durch ‚Anbringen' von Notizen an bestimmten Textstellen geschehen.

Koautorensysteme dienen zum Verfassen von Dokumenten im Team. Diese Dokumente können beispielsweise Textdokumente, Poster, Webseiten oder auch Quelltexte sein. Je nach System wird den Teammitgliedern das gesamte Dokument oder ein Ausschnitt angezeigt, ebenso existieren Differenzen zwischen den Systemen in bezug auf mögliche Überschneidungen der aktuellen Arbeitsbereiche der Teammitglieder bzw. in bezug auf die Granularität gesperrter Bereiche. Da häufig inhaltliche Überschneidungen zwischen den Autoren auftreten und während der Bearbeitung oft Planungen und Absprachen zu Details notwendig werden, sind informelle

Kommunikationskanäle unbedingt notwendig und andere Maßnahmen zur Erhöhung des Gruppenbewusstseins von positivem Einfluss auf das Ergebnis. Gruppeneditoren sind ein häufiger Gegenstand der CSCW–Forschung, bekannte und häufig analysierte Vertreter sind Quilt [Fish 88], GROVE [Ellis 91], DistEdit [Knister 90] und Prep [Neuwirth 90].

Kooperative Zeichentools dienen zum Erstellen von Vektor– und Bitmapgraphiken durch mehrere Teammitglieder. Auch kooperativ nutzbare CAD–Systeme können im weitesten Sinne zu diesem Typ von CSCW–Applikationen gezählt werden. Je nach dem Typ der zu erstellenden Zeichnung müssen geeignete Sperrverfahren angewandt werden bzw. sind optimistische Verfahren möglich. Einige Beispiele für kooperative Zeichentools sind Commune [Bly 90], Xsketch [Lee 90], GroupSketch und GroupDraw [Greenberg93] sowie VideoDraw [Tang 90].

Sitzungs– und Entscheidungsunterstützungssysteme

Sitzungsunterstützungssysteme (Electronic Meeting Systems – EMS) dienen zur Verbesserung der Effektivität von face–to–face–Sitzungen. Dem Nutzen von Sitzungen wie Generierung von mehr Ideen, Synergieeffekten, verbesserter Informationsvalidierung und Lernprozessen bei Teammitgliedern stehen Schwierigkeiten wie ungleichmäßig verteilte Redezeit, Verminderungs– und Konzentrationsprobleme, Passivität einzelner Teammitglieder, kognitive Einbahnstraßen und Informationsüberfluss gegenüber. Sitzungsunterstützungssysteme vereinigen häufig verschiedene Typen von Applikationen wie E–Mail–Systeme, Videokonferenzsysteme, gemeinsame Informationsräume und Gruppeneditoren. Typischerweise unterstützen derartige Systeme Aufgaben wie das Sammeln von Ideen, Abstimmungen und das gemeinsame Edieren von Dokumenten. Häufig werden die Sitzungsunterstützungssysteme in speziellen Electronic Meeting Rooms installiert, in denen ein Großdisplay und ein Konferenztisch sowie oft vernetzte Personalcomputer für alle Teilnehmer vorhanden sind.

In Abhängigkeit von der Integration der Teilnehmer können folgende Modellarchitekturen unterschieden werden: Beim *Chauffeur–Modell* ist der Chauffeur[6] der einzige Sitzungsteilnehmer, der über einen Rechnerzugang verfügt. Somit besteht für die anderen Teilnehmer kein direkter Zugriff auf die Informationen des Großdisplays. Beim *rechnergestützten Modell* existiert ein ausgezeichneter Nutzer als Chauffeur, der für die Aktivierung der Displaysoftware, die Planung der Tagesordnung und die Moderation verantwortlich ist. Bei diesem Modell verfügen alle Sitzungsteilnehmer einen Rechner und haben durch diesen Zugriff auf die Darstellung auf dem Großdisplay. Das *interaktive Modell* besitzt keinen Chauffeur und gestattet dadurch eine gleichberechtigte, parallele Arbeit der Sitzungsteilnehmer.

Entscheidungsunterstützungssysteme (Group Decision Support Systems – GDSS) steigern die Effizienz von Gruppenentscheidungsprozessen im Rahmen teilweise strukturierter Aufgaben. Die Entscheidungsaufgabe wird dabei in Problembereiche zerlegt. Zu jedem Problembereich können

[6] In einem Teil der englischsprachigen Literatur, z.B. bei [Viller 91], wird der Chauffeur auch als ‚facilitator' bezeichnet.

verschiedene Meinungen zugeordnet werden sowie zu jeder Meinung mehrere Pro– und Contra–Argumente. Da Problembereiche mit anderen, nachgeordneten Problembereichen in Relation gesetzt werden können, ergibt sich eine baumförmige Graphstruktur. GDSS werden oft in Meeting–Rooms installiert und zusammen mit der elektronischen Sitzungsunterstützung verwendet. Unterstützt wird im allgemeinen die Modellierung der Entscheidungsstruktur, die endgültige Abstimmung und das Sammeln von Ideen und Meinungen sowie die Bewertung dieser. Diese Vorgänge erfolgen im allgemeinen anonym, durch diese Entpersonifizierung der Aussagen können Ideen unvoreingenommener beurteilt werden. Weitere Vorteile sind die gleichzeitige Ideeneinbringung durch alle Teilnehmer, die Rationalisierung der Bewertung und die oft sorgfältigere textuelle Formulierung der Ideen.

2.2.5 Bekannte Vertreter von Groupware–Systemen

Nachfolgend sollen einige ausgewählte Beispiele häufig eingesetzter Groupware–Lösungen kurz vorgestellt werden. Hierbei handelt es sich sowohl um kommerzielle Systeme als auch um akademische Lösungen, die einen gewissen Verbreitungsgrad erreicht haben. Diese Aufzählung kann und will keinen Anspruch auf Vollständigkeit erheben und nur einige interessante Exemplare eines breiten Spektrums streifen.

Lotus Notes

Lotus Notes ist ein Produkt der Lotus Development Corporation, die mittlerweile im Besitz von IBM ist. Lotus Notes vereinigt die Funktionalität von elektronischen Postsystemen und Bulletin–Board–Systemen. Das System besteht aus verteilten, gemeinsam nutzbaren Datenbanken, in denen semistrukturierte Dokumente durch Anwender abgelegt werden. Die Datenbanken können von Anwendern abonniert werden, diese erhalten eine Anzeige der noch nicht gelesenen Dokumente. Im Unterschied zu anderen Bulletin–Board– Systemen besitzt Lotus Notes eine automatisierte Aktualisierung der verteilten Datenbestände, sehr vielseitige Strukturierungsmöglichkeiten und eine Makrosprache zur Entwicklung eigener Applikationen. Anwender können lokal mit Notes arbeiten, wenn für sie notwendige Datenbanken auf ihren Rechner repliziert wurden sind. Lotus Notes bietet keine expliziten Koordinationsfunktionen an, ebenso fehlen komfortable Mittel zur Vorgangsdefinition. Durch Einsatz der Makrosprache und Zusatzwerkzeuge wie Action Workflow können diese jedoch implementiert werden.

BSCW

BSCW steht für ‚Basic Support for Cooperative Work' [Bentley 95] und ist ein Framework zur Realisierung gemeinsamer Informationsräume der Gesellschaft für Mathematik und Datenverarbeitung (GMD–FIRST). Der gemeinsame Informationsraum wird im WWW so realisiert, dass Nutzer ihre Informationen darin strukturiert ablegen können. Daneben gibt es Mechanismen, die Nutzer über Modifikationen an Dokumenten, die gegenwärtige Gruppenzusammensetzung sowie über aktuelle Aktivitäten informieren. BSCW befindet sich seit

1995 im Praxistest im Rahmen des POLITeam–Projektes [Prinz 96], in dem die verteilte Arbeit von Ministerien an den Standorten Bonn und Berlin untersucht wird.

Action Workflow

Action Workflow [Medina–Mora 92] ist ein Produkt von Action Technologies. Action Workflow ist ein auf der Sprechakttheorie basierendes Workflow Management System. Als Grundstruktur dient eine sogenannte Action Loop, welche ein 4–Phasen Interaktionsmodell zwischen einem *Customer* und einem *Performer* beschreibt. Die einzelnen Phasen umfassen Anbahnung, Verhandlung, Durchführung und Beurteilung. Jede Phase wird mit den möglichen Handlungsalternativen durch Strukturen der Sprechakttheorie modelliert. Durch Verkettung dieser Grundstruktur können komplexere Vorgänge modelliert werden, die eine größere Anzahl Beteiligter umfassen.

Flowmark

Flowmark ist ein Workflow Management System der IBM. Flowmark arbeitet vorgangsorientiert, da sich Ablaufbeschreibungen nicht auf zu bearbeitende Objekte, sondern auf den Gesamtvorgang beziehen. Die Modellstruktur einer Organisation ist fest vorgegeben und muß eine streng hierarchische Form haben. Dokumente lassen sich nicht integrieren, sondern nur Verweise oder Namen von Dokumenten. Damit sind Dokumente nicht automatisch von jedem Rechner zugreifbar.

GroupKit

GroupKit [Roseman 92] [Roseman 96] ist ein Toolkit zur Erstellung synchroner Groupware–Systeme der Universität Calgary. Eine bei mehreren Anwendern laufende synchrone Anwendung wird als Konferenz bezeichnet. Konferenzen können dynamisch erzeugt und gelöscht werden und eine einzige oder mehrere Anwendungen umfassen. Konferenzen müssen durch mindestens eine Audio–Verbindung ergänzt werden, die nicht in GroupKit selbst unterstützt wird. Die Verwaltung von Konferenzen geschieht nutzerseitig unabhängig von konkreten Konferenzanwendungen durch ‚Session Managers'. Ein Session Manager kann alle Konferenzen eines Anwenders verwalten. Die GroupKit – Architektur basiert auf einem zentralem Server, auf dem ein ‚Registrar'– Prozess läuft, der die Daten des Konferenz–Managements verwaltet und als Kommunikationszentrale der replizierten Session–Managers dient. Die Kommunikation zwischen den Komponenten erfolgt über TCP/IP. GroupKit ist in der Skriptsprache TCL implementiert und umfaßt den Registrar sowie Sessionmanager. Anwendungen müssen ebenfalls in TCL implementiert werden.

TeamWave Workplace

TeamWave Workplace ist die Nachfolgeentwicklung des GroupLabs der Universität Calgary zu GroupKit. Anstelle der Konferenz wird der Raum als zentrales Konzept verwendet. In einem Raum können mehrere Anwendungen laufen, die dann zu einem gemeinsamen Zweck kombiniert

werden. Räume können als Treffpunkt, als Container oder als Aufgabe dienen, ebenso ist der gemeinsame Arbeitsbereich ein Raum. Rauminhalte sind persistent in bezug auf die Anwendungen und deren Daten, damit wird auch eine asynchrone Kooperation ermöglicht. Benutzer können sich nur in einem Raum zu einem Zeitpunkt aufhalten, sollen mehrere Aufgaben parallel bearbeitet werden, muss ein spezieller Raum dafür erzeugt werden.

Die Architektur von TeamWave Workplace beruht auf einer reinen Client–Server – Architektur. Die persistente Speicherung der Rauminhalte erfolgt beim Server. Der Server fungiert weiterhin als Kommunikationsknoten, da Clients nicht direkt, sondern nur über den Server kommunizieren. Die Administration erfolgt über einen speziellen Client. Auch TeamWave Workplace ist in TCL realisiert. Die Anwender–Clients laufen als Prozess, der mehrere TCL–Programme parallel interpretiert und ausführt [Maher 00].

DOLPHIN

DOLPHIN [Streitz 94] [Mark 96] ist ein von GMD–IPSI entwickeltes System für die Vorbereitung und Unterstützung von Sitzungen. Die Teilnehmer verfügen über einen Arbeitsplatzrechner, außerdem wird ein Großdisplay verwendet. DOLPHIN unterstützt die kooperative Bearbeitung von hypermedialen Gruppendokumenten, die Erzeugung und das Manipulieren informaler Strukturen wie Freihandzeichnungen und die Manipulation von Hypertextdokumenten des Vorgängersystems SEPIA. DOLPHIN ist unter Smalltalk–80 (ObjectShare Visual Works) entwickelt worden und unterstützt eine TCP/IP basierte Kommunikation zwischen Rechnern der Sitzungsteilnehmer. DOLPHIN nutzt ab Version 2 eine Master– Client– Architektur. Es existiert kein spezieller Server, statt dessen übernimmt der erste Client diese Aufgabe und wird dadurch Master. DOLPHIN ist ein Anwendungsprogramm und kann daher nicht wie Toolkits für Eigenimplementationen genutzt werden. DOLPHIN ermöglicht mehrere virtuelle Konferenzen über separate Kommunikationskanäle.

2.3 CSCW-Funktions- und Strukturmodelle

2.3.1 Gruppenprozessmodelle

Gruppenarbeit ist stets auf die Realisierung definierter Ziele ausgerichtet. Um diese zu erreichen, sind (Teil–)Aufgaben zu lösen, was wiederum die Ausführung darauf orientierter Tätigkeiten impliziert. Unter Gruppenarbeit ist somit die ‚Summe aller aufgabenbezogenen Tätigkeiten, welche von Gruppenmitgliedern ausgeführt werden, um zielbezogene Aufgaben zu erfüllen und somit Gruppenziele zu erreichen' [Teufel 95] zu verstehen. Diese aufgabenbezogenen Tätigkeiten können drei Kategorien von Vorgängen zugeordnet werden: Kommunikations–, Koordinations– und Kooperationsvorgängen. Zwischen diesen Vorgangskategorien besteht ein hierarchisches Abhängigkeitsverhältnis, da Kooperationsprozesse im allgemeinen eine Koordination bedingen, während Koordinationsprozesse eine Kommunikation der Teammitglieder erfordern.

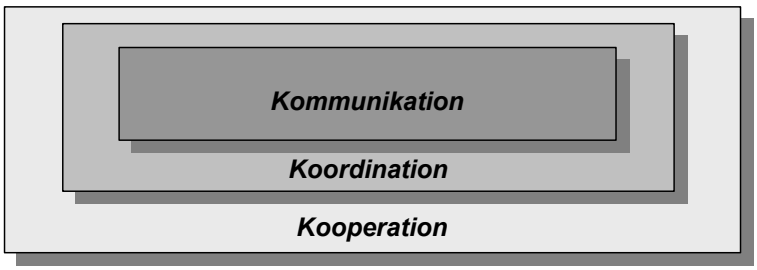

Abbildung 3: Gruppenprozesse bei der Teamarbeit

Teamarbeit beinhaltet im allgemeinen alternierende Phasen synchroner und asynchroner Kooperation. Die synchronen Phasen sind oft notwendig, um Kommunikations– und Koordinationsvorgänge beispielsweise in face–to–face–Sitzungen ausführen zu können, während die kooperative Arbeit an der eigentlichen Realisierung der Gruppenziele je nach Aufgabe synchron oder auch asynchron erfolgt.

Die Gruppenziele sowie die Abfolge von Tätigkeiten zur Realisierung dieser Ziele können als Prozess interpretiert werden. Unter dem Gruppenprozess versteht man somit die Spezifikation von Informationen, Aktivitäten und Eigenschaften einer elektronisch unterstützten Gruppe bei Festlegung des Rahmens, in dem die Gruppenarbeit stattfindet. Der Endzustand des Gruppenprozesses repräsentiert das Resultat der Gruppenarbeit. Der Gruppenprozess enthält die statischen Komponenten Gruppenziel, Gruppenorganisation, soziale und technische Kommunikationsprotokolle der Gruppe sowie die Gruppenumgebung, d.h. das organisatorische Umfeld und die technische Ausstattung der Gruppe. Dynamische Komponenten des Gruppenprozesses sind die während der Teamarbeit entstehenden Gruppendokumente, die Aktivitäten des Teams mit ihren kausalen und temporalen Abhängigkeiten untereinander, Gruppensitzungen als Folge von kooperativen Gruppenaktivitäten sowie der aktuelle Gruppenstatus [Borghoff 98].

Nach Kaplan et al. können Gruppenprozessen folgende weitere typische Eigenschaften zugeschrieben werden [Kaplan 91]:

- *Kollaborative Prozesse:* Diese grundlegende Eigenschaft muss beim Entwurf von Systemen zur Unterstützung der Prozesse ständig beachtet werden und kann als kritischer Faktor für den Erfolg der rechnerbasierten Prozessunterstützung angesehen werden.
- *Offene Prozesse*: Die Prozesse werden als offen bezeichnet, da es im allgemeinen nicht einen einzelnen, wohldefinierten Weg zur Erreichung der Ziele der Teamarbeit gibt. Unterschiedliche Teams erreichen das Ziel oft auf unterschiedlichen Wegen, d.h. die detaillierte Abfolge und Verteilung von Arbeitsschritten kann differieren.
- *Prozesse mit offenem Ende*: Teamarbeit wird häufig als *open–ended* bezeichnet, da vor Beginn der Arbeit das Kriterium der Zielerreichung nicht im Detail klar definiert werden kann,

sondern nur vage Vorstellungen von den Ergebnissen vorliegen. Häufig verfeinern sich die Vorstellungen über das Ergebnis der Teamarbeit während dieser.

- *bedingt automatisierbare Prozesse*: Durch den Charakter der Teamarbeit und vor allen die letztgenannten beiden Eigenschaften erscheint eine völlige Automatisierung der Prozesse als unrealistisch. Ziel kann es nur sein, einzelne Phasen oder Arbeitschritte zu automatisieren, Kommunikation, Koordination und Kooperation zu unterstützen und Werkzeuge zum besseren Management des Gesamtprozesses zur Verfügung zu stellen.

Gruppenprozessmodelle dienen einerseits als Hilfsmittel zur Analyse von Gruppenarbeit, können aber auch als konzeptionelle Basis für eine informationstechnische Unterstützung der Teamarbeit dienen. Nach Rapaport können drei Kategorien von Gruppenprozessmodellen unterschieden werden:

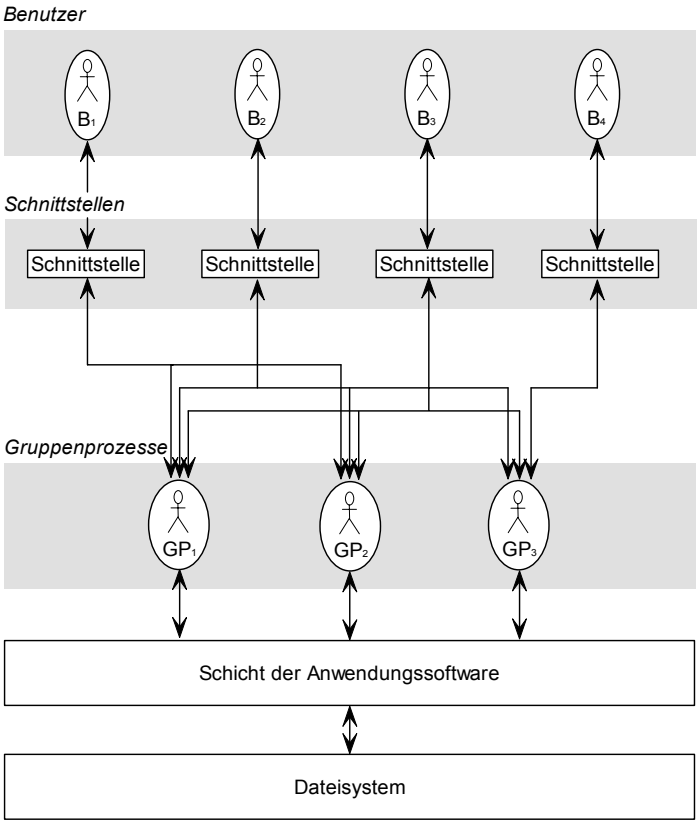

Abbildung 4: Zentrales Gruppenprozessmodell

Zentrales Gruppenprozessmodell:

Das zentrale Gruppenprozessmodell geht von der Verwaltung aller Informationen in einem Datenbestand aus. Die Applikationssoftware ist für die Separierung unterschiedlicher Prozesse verantwortlich. Es gibt im System nur den Originaldatenbestand, auf dem alle Teammitglieder operieren, von einer Notwendigkeit von privaten Informationsbereichen wird bei diesem Modell nicht ausgegangen.

Verteiltes, nichtrepliziertes Gruppenprozessmodell:

Bei diesem Modell werden die Informationen nach einzelnen Gruppenprozessen differenziert und nach diesem Kriterium verteilt verwaltet. Dadurch gibt es von jeder Information nur ein Original, das auf dem für den jeweiligen Gruppenprozess zuständigen Rechner abgelegt ist. Da auf allen Arbeitsplatzrechnern dieselbe Applikationssoftware eingesetzt wird, ist sowohl Zugriffs– als auch Ortstransparenz[7] gesichert.

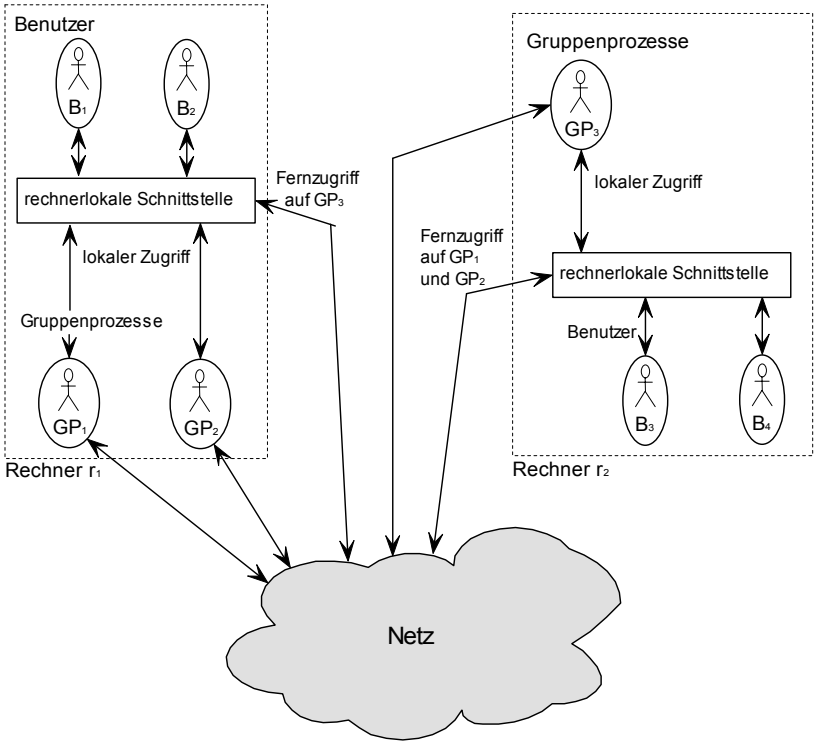

Abbildung 5: Verteiltes, nichtrepliziertes Gruppenprozessmodell

Verteiltes, replizierendes Gruppenprozessmodell:

Hier werden Repliken der Informationen eines Gruppenprozesses auf jedem Rechner, auf dem Mitglieder des jeweiligen Gruppenprozesses arbeiten, angelegt. Dieses Modell basiert auf Zugriffs–, Orts– und Replikationstransparenz. Vorteile dieses Modells sind die bessere Systemperformance durch kürzere Response–Zeiten und die höhere Robustheit, diese werden jedoch mit einer erschwerten Sicherung der Modellkonsistenz wegen denkbarer Race–Conditions erkauft.

[7] Transparenzeigenschaften Verteilter Systeme werden unter anderem in [Borghoff 98] ausführlich diskutiert.

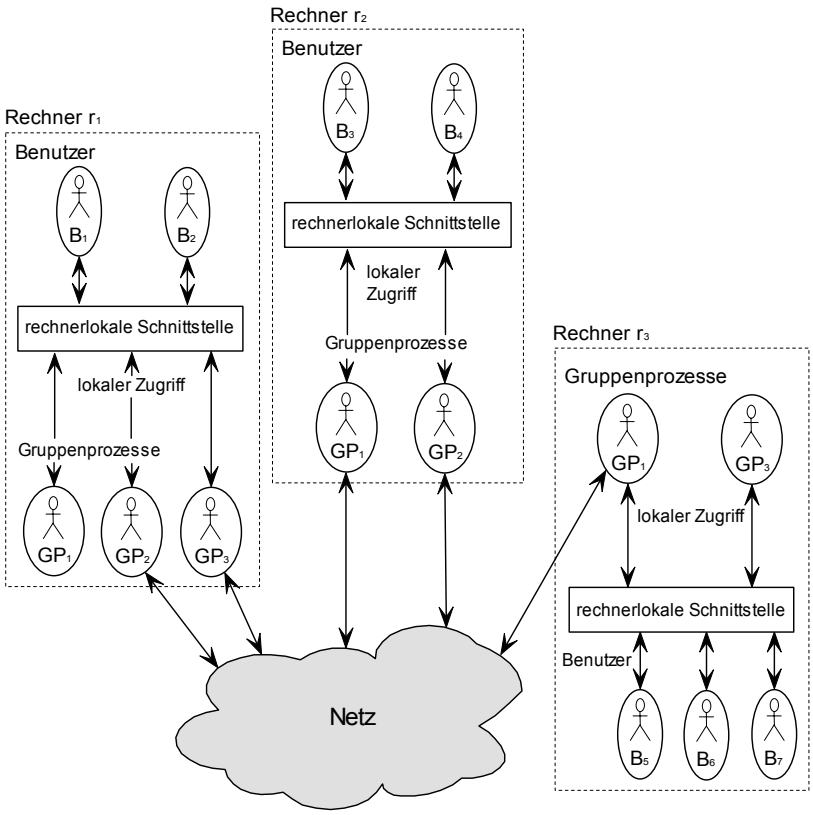

Abbildung 6: Verteiltes, repliziertes Gruppenprozessmodell

2.3.2 CSCW– Architekturen

Die Architektur eines CSCW–Systems ist von mehreren Faktoren abhängig. Von fundamentalem Einfluß auf die Architektur ist die Frage, ob eine Neuimplementierung einer gruppenbewußten Groupware–Applikation geplant ist oder ob existierende Einbenutzer–Anwendungen kooperativ nutzbar gemacht werden sollen. Ein weiterer wichtiger Aspekt ist der zugrundeliegende Gruppenprozess mit dessen Attributen Verteilung (zentral vs. verteilt) und Replikation (repliziert vs. nicht repliziert).

Aus logischer Sicht besitzen alle kooperativen Anwendungen eine zentralisierte Struktur [Rodden 91], d.h. für den Anwender erscheint der durch die kooperative Applikation verwaltete Informationsbestand als einzelner atomarer Block. Für die Implementierung existieren jedoch mehrere mögliche Architekturen zur Realisierung von Groupware–Applikationen. Eine häufig angewandte Klassifikation von CSCW–Architekturen ist jene nach dem Kriterium der implementierungsseitig genutzten Replikation von Information. Groupware–Applikationen können danach einer der folgenden drei Klassen zugeordnet werden: Systeme mit zentralen, replizierenden oder hybriden Architekturen [Lauwers 90] [Ahuja 90] [Rodden 91].

Zentralisierte CSCW– Architekturen

Typisch für zentrale CSCW–Architekturen ist die Existenz nur einer Instanz der Anwendung. Als Konsequenz müssen alle auf Nutzerinteraktionen bezogenen Ein– und Ausgaben vom Rechner des Anwenders an den Applikations–Rechner weitergeleitet werden. Nutzereingaben können gleichzeitig erfolgen und müssen daher sequentialisiert werden. Zentralisierte Architekturen sind im allgemeinen weniger komplex als replizierte Architekturen und gewährleisten eine einfachere Konsistenzsicherung. Nachteile sind die schlechte Performance und das oft unbefriedigende Antwortverhalten. Weiterhin sind Applikationen, die auf dieser Architektur basieren, nicht unbegrenzt skalierbar, da das Hinzufügen neuer Benutzer negative Auswirkungen auf die Performance hat.

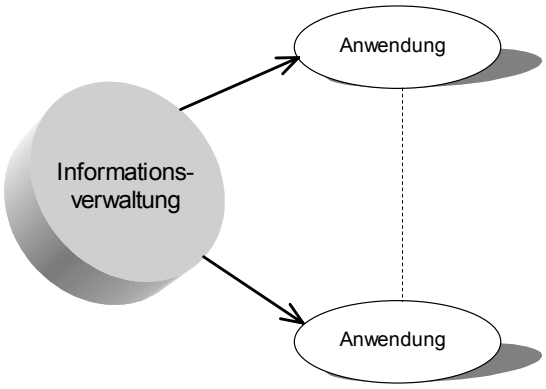

Abbildung 7: Information sharing

Information Sharing: Bei dieser Architektur verfügt jeder Nutzer über seine Instanz der Anwendung. Der Datenaustausch zwischen diesen erfolgt jedoch ausschließlich indirekt über ein zentrales Informationsverwaltungssystem. Da dieses im allgemeinen mit Zeitverzögerungen verbunden ist und Clients bei der Majorität der Systeme erst beim wiederholten Informationszugriff Änderungen wahrnehmen, ist diese CSCW–Architektur nur für asynchrone Kooperation geeignet. Als Beispiele können Lotus Notes, das World Wide Web und Basic Support for Cooperative Work (BSCW) genannt werden.

Window Sharing: Bei dieser Architektur, die durch gemeinsame GUI–Fenster mit identischem Inhalt gekennzeichnet ist, existiert nur eine Instanz der Anwendung. Zwischen die Anwendung und die graphische Nutzeroberfläche wird ein ‚Pseudo Window Server' geschaltet. Dessen Aufgaben umfassen das Sequentialisieren aller Benutzereingaben und das Senden aller Änderungen der Fensterinhalte an alle Windowserver der Anwender–Arbeitsstationen, womit gesichert wird, dass alle Anwender dieselben Fensterinhalte präsentiert bekommen[8]. Damit ist diese Architektur gut für sysnchrone Kooperation geeignet, jedoch beinhaltet diese die Gefahr

[8] Dieses Konzept wird ‚What You See Is What I See' (WYSIWIS) genannt und in Kapitel 1.4.1 ausführlicher betrachtet.

von sogenannten Window– und Scroll–Wars, bei denen sich Anwender gegenseitig durch Verändern der Fensterinhalte bei ihrer Arbeit behindern.

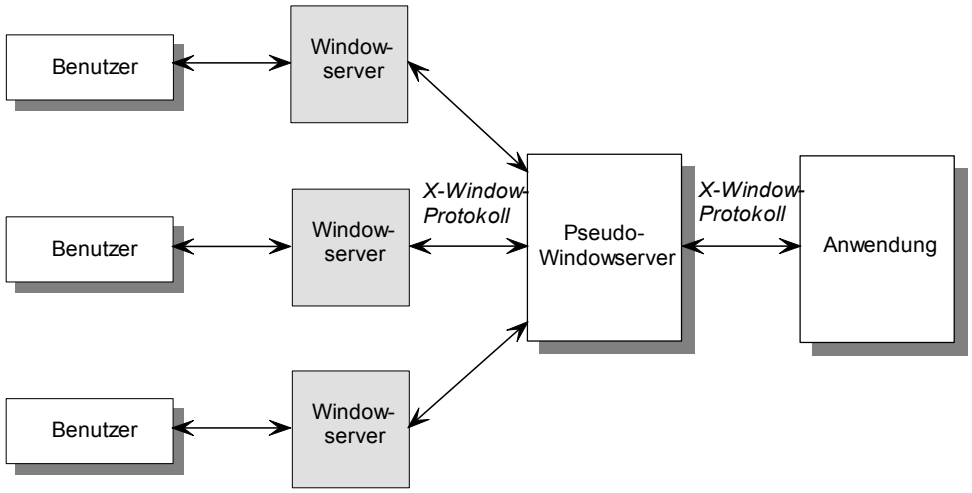

Abbildung 8 : Window sharing

Diese Gefahr ist um so mehr gegeben, da sich Window–Sharing ohne Änderungen an der Original–Applikation für bestehende Anwendungen umsetzen läßt. Solche Realisierungen besitzen dann meist keine weiteren Interaktionsmöglichkeiten für die Benutzer, mit denen Konflikte über soziale Protokolle gelöst werden können. Das klassische Beispiel für diese Architektur ist Shared–X, das auf X–Windows aufsetzt, welches die erforderliche Eigenschaft eines netzorientierten Windowsystems besitzt.

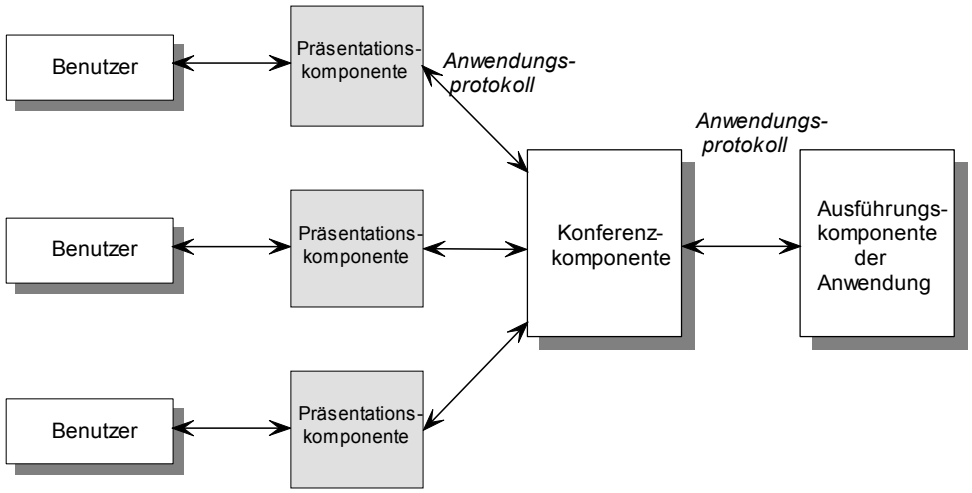

Abbildung 9: Konferenz–Komponente

Konferenz–Komponente: Bei der Anwendung von Konferenz–Komponenten wird die Anwendung in eine Ausführungs– und eine Präsentationskomponente zerlegt. Eine Instanz der Ausführungs– und der Präsentationskomponente laufen auf einem ausgezeichnetem Rechner, während jeder Anwender über eine Instanz der Präsentationskomponente verfügt. Der Konferenz–Komponente obliegt die Serialisierung der Nutzereingaben und das Propagieren von Änderungen der zu

präsentierenden Objekte. In Kontrast zum Window–Sharing wird jedoch nicht das Protokoll des Fenstersystems genutzt, sondern es werden Informationen mit anwendungsspezifischer Semantik übertragen. Damit muß diese Architektur nicht nur ausschließlich für striktes WYSIWIS angewandt werden, es ist sogar denkbar, dieselbe Informationsabstraktion in unterschiedlichen Views zu präsentieren. Auch diese Architektur ist für bestehende Applikationen anwendbar, indem eine entsprechende Aufteilung der Applikation auf Quelltext–Niveau vorgenommen wird. Ebenso können bei geeigneter Implementierung des Systems Scroll– und Window–Wars vermieden werden. Problematisch ist bei dieser Architektur, dass sich die Konferenz–Komponente zu einem Bottleneck entwickeln kann.

Replizierende CSCW– Architekturen

Für replizierende CSCW–Systemarchitekturen ist bezeichnend, dass auf jeder Arbeitsstation eines Teammitgliedes eine Kopie der Groupware–Anwendung ausgeführt wird. Benutzereingaben werden an alle Kopien der Anwendung gesandt und müssen lokal sequenzialisiert werden. Vorteilhaft ist das gute Antwortverhalten des Systems, da die Eingabepropagation parallel zur lokalen Verarbeitung erfolgen kann. Replizierende Architekturen skalieren besser als zentralisierte. Die Umsetzung ist jedoch wesentlich komplexer und die Konsistenz schwieriger zu sichern. So erfordert die Realisierung von Systemen erhöhte Anstrengungen zur Sicherung eines identischen Startzustandes, auch für unter Umständen später hinzukommende Teammitglieder. Weiterhin müssen die Anwendungen ein deterministisches Verhalten aufweisen und dürfen weder von ihrem Kontext noch von temporalen Parametern abhängig sein. Zur Sicherung der Konsistenz des Gesamtsystems ist es erforderlich, dass die Anwendungsinstanzen alle Eingaben in derselben Reihenfolge erhalten.

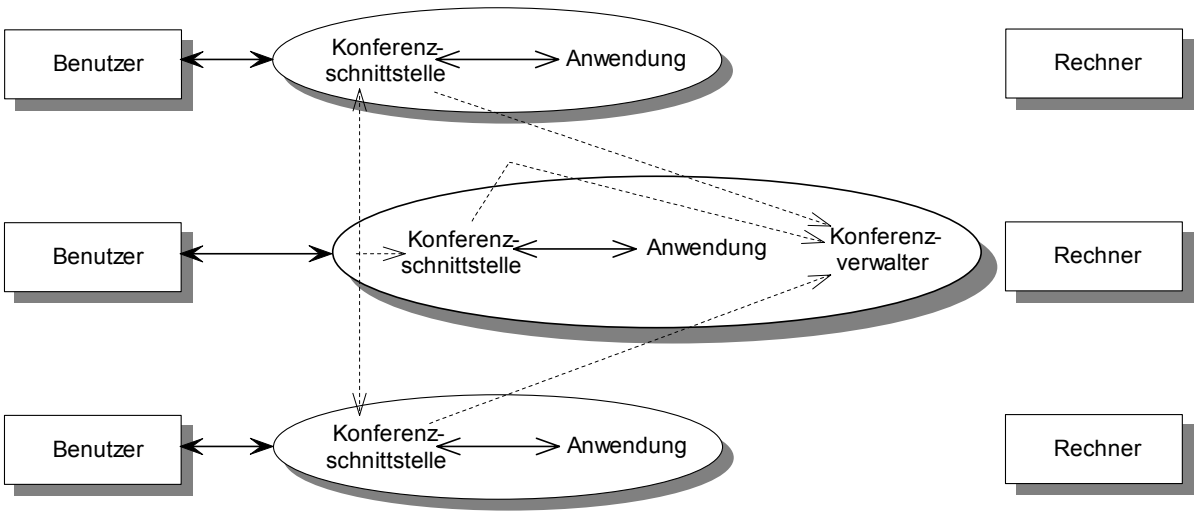

Abbildung 10: Konferenz–Verwalter

Konferenz–Verwalter: Bei dieser Architektur werden die gemeinsam genutzte Anwendung und eine Konferenzschnittstelle repliziert. Diese fungiert als Vermittler zwischen dem replizierten Groupware–System und den Anwendern. Deren Eingaben werden durch die Konferenz-

schnittstellen an alle Applikationen propagiert. Ebenso werden die Ausgaben des Systems durch die Konferenzschnittstelle dem Benutzer dargestellt, aber nicht an andere Rechner weitergeleitet. Auf einem definierten Rechner existiert ein Konferenzverwalter, der die Rechnerkonferenzen organisiert sowie die Kontrolle der Nebenläufigkeiten und weitere Synchronisationsaufgaben wie beispielsweise Zugriffe auf die gemeinsamen Informationen übernimmt. Diese Architektur hat besondere Anforderungen an die Synchronisation der Groupware–Applikation, weiterhin müssen die Anwendungen eingabetreu[9] und ständig in derselben Version auf allen Knoten installiert sein.

Gruppenbewußte Konferenzsysteme: Gruppenbewusste Konferenzsysteme sind CSCW–Applikationen, bei denen die Verwaltung der Benutzerschnittstelle, die Synchronisation sowie Concurrency Control in das System integriert sind. Für derartige Groupwaresysteme muss ein spezielles Design erstellt werden; es ist bei einer derartigen Architektur nicht möglich, existierende Einbenutzerapplikationen mit einzubeziehen.

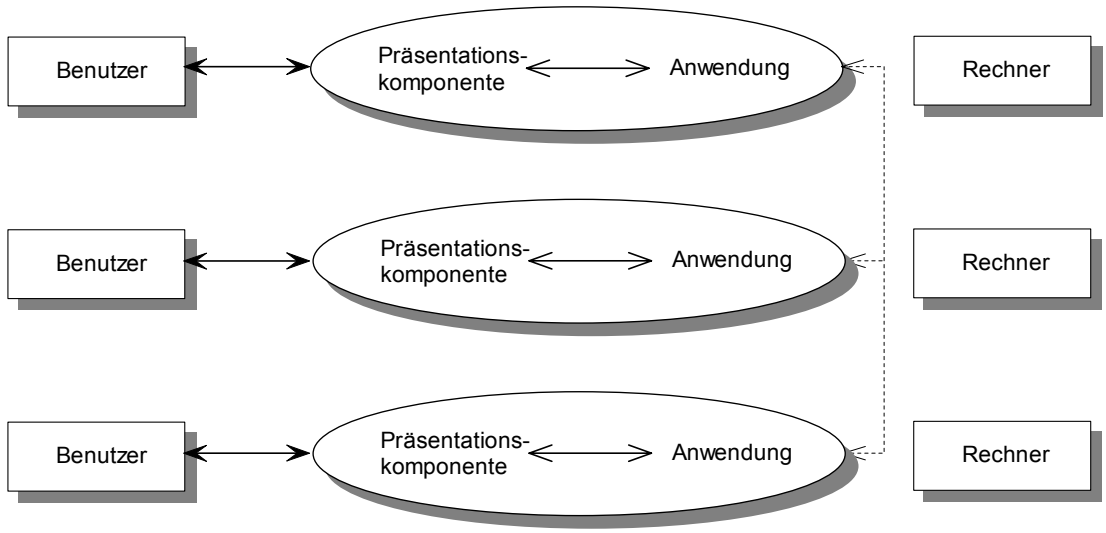

Abbildung 11: Gruppenbewusste Konferenzsysteme

Hybride CSCW– Architekturen

Hybride Architekturen versuchen, die Vorteile zentralisierter und replizierender Architekturen zu vereinigen, indem das CSCW–System Portionen beider Herangehensweisen, also zentrale als auch replizierte Anteile enthält. Vorteile sind das gute Antwortverhalten und die vereinfachte Konsistenzsicherung. Durch die in derartigen Systemen notwendige Aufgabenverteilung unter den Komponenten und der daraus folgenden unterschiedlichen Komponentenstrukur sind hybride Architekturen jedoch häufig komplexer als andere CSCW–Architekturen.

Ein bekanntes Beispiel eines Groupware–Systems mit hybrider Architektur ist *Rendezvous* [Brink 92], [Hill 94]. Rendezvous ist ein constraint–basiertes Toolkit zur Implementierung von

[9] Eine verteilte Anwendung wird als eingabetreu bezeichnet, wenn diese bei gleicher Eingabe gleiche Ergebnisse erzeugt, unabhängig davon, auf welchem Rechner die Anwendung ausgeführt wird.

Groupwaresystemen für geographisch verteilte Kooperation mit einer speziell entwickelten Architektur und Sprache. Es basiert auf einer zentralisierten Plattform, auf der mehrere unterschiedliche Applikationen aufgesetzt werden können. Auf Rendezvous beruhende Systeme arbeiten sowohl mit einem zentralisierten Datenmodell, welches als ‚Abstraktion' der kooperativ genutzten Informationen dient, als auch mit einer zentralisierten Darstellungskomponente. Im Gegensatz dazu werden aber die Basisdienste der Nutzerschnittstelle repliziert. In einer Weiterentwicklung des Rendezvous–Systems (Distributed multiprocessor ALV) kann ebenso die View–Komponente verteilt werden, was bei komplexen graphischen Präsentationen das Antwortverhalten verbessert. Hier ist zu beachten, dass Multi–User–Applikationen wesentlich höhere CPU–Last als Single–User–Applikationen verursachen, da jede Systeminteraktion eines Nutzers potentiell eine Aktualisierung aller Views erfordert.

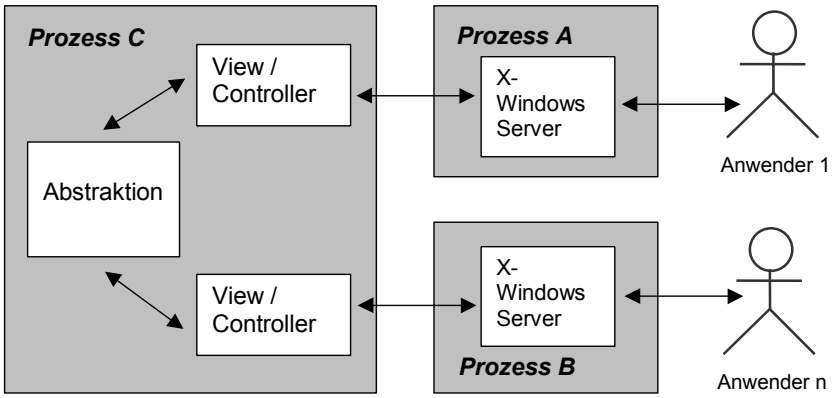

Abbildung 12: Rendezvouz–Architektur

Das Rendezvous–System nutzt Abstraction–Link–View (ALV), eine adaptierte Form des Model–View–Controller (MVC)–Paradigmas [Krasner 88], als Systemphilosophie. Als Abstraktion wird die Menge der für das Groupware–System relevanten geteilten Informationen bezeichnet, die bei der Bildung des Informationsmodells durch Abstraktionsprozesse als im Rahmen der Systemverantwortlichkeit betrachtet werden. Als View wird die Präsentation dieser Informationen und die dazugehörigen Interaktionsmöglichkeiten des Anwenders über die Nutzerschnittstelle des Systems bezeichnet. Anwender interagieren niemals direkt mit dem Datenbestand der Abstraktion, sondern immer indirekt über den View. Daher muss die Implementierung des Views in der Lage sein, Informationen der Abstraktion zu aktualisieren. Einer Datenabstraktion können mehrere Views zugeordnet werden. Dadurch können identische Informationen auf verschiedene Art und Weise dargestellt und entsprechend manipuliert werden. Typischerweise existiert ein View pro Anwender, möglich ist aber auch, dass Anwender präferieren, dass ihnen die Informationen gleichzeitig verschiedenenartig dargestellt werden. Der wesentliche Unterschied zwischen MVC und ALV besteht darin, dass bei ALV die View– und Controller–Funktionalität im View vereint wird und dass ein sogenanntes Link–Objekt zwischen Abstraktion und View geschaltet wird, in dem die Lisp/CLOS basierten Constraints zwischen diesen Komponenten verwaltet werden. Damit wird nach Hill, Brinck et al. eine flexiblere

Verwaltung der Abhängigkeiten zwischen Abstraktion und View erreicht und auf die sonst verwendete Kommunikation über Botschaften des View an das Modell sowie Callback–Methoden im View kann verzichtet werden [Hill 94].

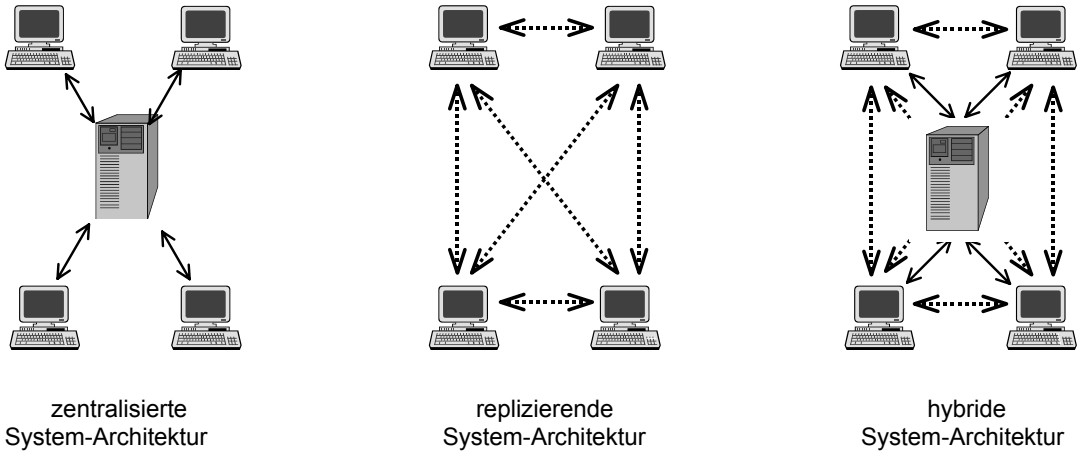

Abbildung 13: CSCW – Architekturen

2.3.3 Modelle der Gruppeninteraktion bei asynchroner Kooperation

Asynchrone Gruppenprozesse bestehen aus einer Anzahl temporal versetzter Interaktionen zwischen den Mitgliedern des Teams. Diese Interaktionen erzeugen oder modifizieren im Groupware–System neue Informationseinheiten, die je nach CSCW–Applikation als Dokumente, Texte, Nachrichten, etc. verwaltet werden. Dabei lassen sich drei allgemeine Modelle ableiten [Borghoff 98]:

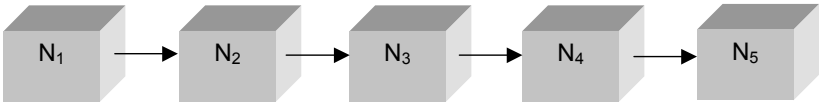

Abbildung 14: Asynchrone Gruppeninteraktion nach dem linearen Modell

Lineares Modell

Beim linearen Modell liegt eine strenge temporale Sequenz zwischen den Informationseinheiten vor. Nach Informationseinheit N_i folgt stets Informationseinheit N_{i+1}, wobei N_{i+1} eine Reaktion auf N_0 bis N_i sein kann oder die Einführung eines völlig neuen Informationskontextes. Beim linearen Modell ist es für die Teammitglieder sehr einfach, die Historie der Gruppeninteraktion nachzuvollziehen, da sowohl die Abfolge der Interaktionen als auch deren Ergebnisse in Form von Informationseinheiten erkennbar sind. Ein Vertreter dieses Typs ist das oben beschriebene Konferenzsystem *EMISARI*.

Kamm–Modell

Beim Kamm–Modell besteht der Gruppenprozess aus einer Reihe von spezifizierten Informationskontexten bzw.Unterthemen. Jeder Informationskontext ist klar definiert und intern linear strukturiert. Die Kamm–Struktur des Gesamtgruppenprozesses entsteht durch die aus sequentiell geordneten Informationseinheiten bestehender Informationskontexte. Die Navigation durch die Informationen wird durch diese Unterteilung erleichtert. Neue Gruppenteilnehmer können relativ leicht Informationen auffinden, indem sie den dazugehörigen Informationskontext identifizieren und dort nach den benötigten Daten suchen. Beim Kamm–Modell können leicht neue Informationskontexte durch privilegierte Anwender kreiert werden. Es ist möglich, die Informationskontexte hierarchisch zu strukturieren, indem zu den bestehenden Informationskontexten rekursiv neue Unterkontexte bzw. Unterthemen zugeordnet werden.

Der bekannteste Vertreter von asynchroner Kooperation nach dem Kamm–Modell ist das *Usenet*. Es existiert hier eine mehrstufige Hierarchie von Informationsthemen, die Newsgroups genannt werden. Die Nachrichten innerhalb einer Newsgroup werden in einer temporalen Sequenz verwaltet, so dass Anwender schnell zu den aktuellen Nachrichten navigieren können.

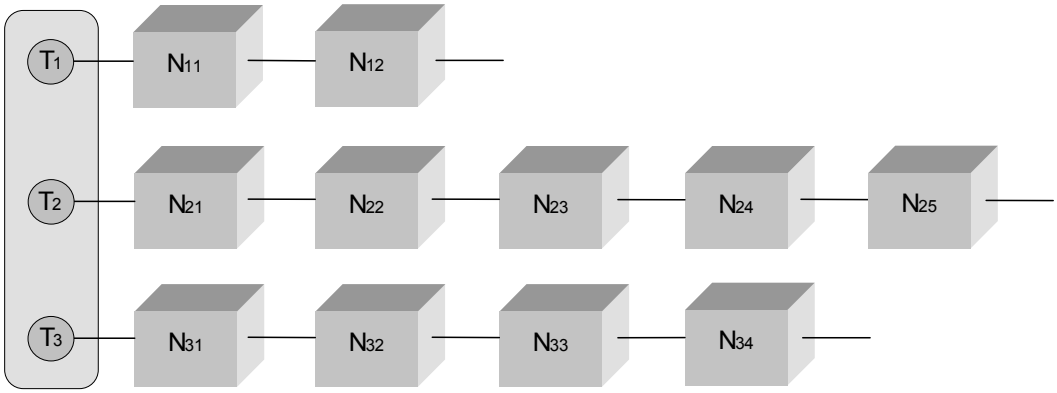

Abbildung 15: Asynchrone Gruppenkooperation nach dem Kammmodell

Verzweigungsmodell

Das Verzweigungsmodell basiert auf einer zyklenfreien Graphstruktur. Es entsteht, indem von einem initialen zentralen Informationskontext Unterthemen abgespalten werden, die dann sukzessive zu neuen Unterbäumen in der Struktur werden können. Neue Teilbäume können jederzeit durch alle Anwender initiiert werden, insofern ist das Verzweigungsmodell dynamischer als das Kamm–Modell. Innerhalb eines Teilbaumes liegt eine sequentielle Struktur zwischen den Verzweigungspunkten vor, diese werden in der Struktur markiert. Durch die Baumstruktur ist es nicht immer trivial, gewünschte Informationen aufzufinden. Daher ist bei Gruppeninteraktionen nach dem Verzweigungsmodell eine Navigationshilfe wichtig.

Bestimmte Systeme, die auf diesem Modell beruhen, besitzen weiterhin Features zum ‚Umhängen' von Teilbäumen, zum Verknüpfen der Sequenzen von Teilbäumen oder zum

Herauslösen von Unterkontexten, womit ein neuer Gruppenprozess initiiert wird. Ein Vertreter dieser Klasse ist das System *Parti* der Universität Stockholm.

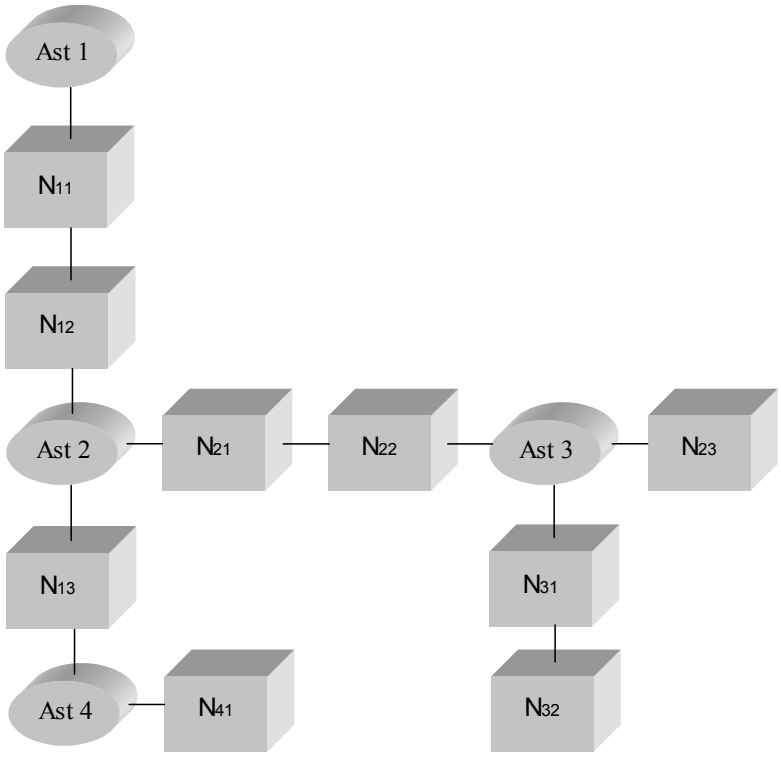

Abbildung 16: Asynchrone Gruppeninteraktion nach dem Verzweigungsmodell

2.4 CSCW–Basistechniken

Um eine rechnerbasierte Unterstützung synchroner und asynchroner Kooperation realisieren zu können, müssen spezifische Techniken eingesetzt werden. Dabei kann zwischen Techniken unterschieden werden, die direkt oder indirekt Teamarbeit unterstützen. Zur ersten Gruppe gehören Maßnahmen zur Gestaltung eines gemeinsamen Kontexts der Teammitglieder wie WYSIWIS und Telepresence–Features. Zu den Techniken zur indirekten Unterstützung von Teamarbeit zählen Koordinierungstechniken für synchrone Arbeit oder auf die Anforderungen von CSCW–Applikationen angepasste klassische Verfahren anderer Teilgebiete der Informatik, wie beispielsweise Multi–User–Undo–Mechanismen. Im folgenden soll vor allem auf für die CSCW–Systemklasse ‚Workgroup Computing' relevante Techniken eingegangen werden, da diese Systemklasse im Fokus der folgenden Kapitel stehen wird.

2.4.1 'What You See Is What I See' (WYSIWIS)

Unter ‚What You See Is What I See' (WYSIWIS)[10] wird eine Technik zu Unterstützung synchroner Kooperation verstanden, die versucht, die Funktionalität einer Wandtafel bei face–to–face–Meetings zu abstrahieren und auf eine rechnerbasierte Kooperationsunterstützung zu übertragen [Stefik 87]. Der gemeinsame Kontext des Teams wird also geschaffen, indem die gemeinsamen Informationen allen Teammitgliedern in einer konsistenten Darstellung präsentiert werden. WYSIWIS gibt Teammitgliedern den Eindruck einer echten Gruppeninteraktion mit den gemeinsamen Objekten, in derselben Form, als würde die Gruppe gemeinsam um ein Textdokument oder eine Zeichnung sitzen.

Bei einem striktem WYSIWIS wird für alle Teammitglieder ein absolut identischer Kontext durch eine uniforme Präsentation in Bezug sowohl auf die dargestellten Informationen und den dargestellten Ausschnitt als auch über Darstellung der Cursors aller Teammitglieder hergestellt. Dieses Vorgehen ist im Praxiseinsatz im allgemeinen zu einschränkend und wird daher meist relaxiert.

Zum Beispiel müssen zur Veränderung der Bildschirminformationen bei striktem WYSIWIS Konventionen im Team geschaffen und eingehalten werden, da sonst das Auftreten von Problemen wahrscheinlich ist. Es kann möglicherweise zum sogenannten ‚Scroll War' kommen, wenn ein Teilnehmer bestimmte Informationen betrachten und ein anderer im Text blättern oder einen anderen Zeichnungsausschnitt betrachten möchte. Von ‚Window War' spricht man, wenn ein Teilnehmer ein neues Fenster öffnet, das für ein anders Teammitglied interessante Informationen überdeckt. Daher ist eine synchrone WYSIWIS–Kooperation ohne einen informationellen Kanal [Dourish 92] wie beispielsweise eine parallele Audioverbindung zur Absprache der nächsten Aktivitäten kaum praktikabel.

Nach Stefik et al. können an den vier Dimensionen Raum, Zeit, Population und Kongruenz sinnvoll Beschränkungen des strikten WYSIWIS relaxiert werden [Stefik 87]. Einige häufig angewandte Varianten sollen im folgenden betrachtet werden:

- *Shared & private Views:* relaxiert das Raum–Constraint und gestattet, den Bildschirm der Teammitglieder in Fenster mit strikter WYSIWIS–Darstellung und in Fenster mit individuellen Inhalten aufzuteilen. Durch die Fenster mit gemeinsamer Darstellung wird der Gruppenkontext geschaffen; die privaten Fenster erlauben die individuelle Vorbereitung von neuen gemeinsamen Informationen, die bei einem gewissen Reifegrad in das Gruppenfenster übertragen werden oder die Ausführung von rollenspezifischen Applikationen. Ein bekanntes Beispiel für derartige Systeme ist RTCAL [Greif 88].

[10] Der Begriff WYSIWIS wurde von Stefik et al. in Anlehnung an WYSIWYG (‚What You See Is What You Get') geprägt, der eine Klasse von Texteditoren bezeichnet, bei denen das Textlayout während der Bearbeitung in der Form der Druckausgabe dargestellt wird.

- *Selektive Cursordarstellung:* erlaubt die Darstellung des Cursors nur ausgewählter Teammitglieder, verletzt aber das Raum–Constraint. Die Darstellung des Mauszeigers bzw. des Cursors jeder Person kann gerade bei großen Teams irritierend wirken. Ebenso wäre eine graphische Hervorhebung des eigenen Cursors nicht mit striktem WYSIWIS vereinbar.
- *Individuelle Bildschirm–Layouts:* sind ebenso eine Abschwächung des Raum–Constraints. Window Wars können vermieden werden, wenn Teammitglieder Fenster mit gemeinsamen bzw. individuellen Darstellungen frei am Bildschirm positionieren können. Damit müssen aber Cursordarstellungen anderer Teammitglieder fensterbezogen übertragen werden und räumliche Bezugnahmen bei der Kommunikation über die informationellen Kanäle werden erschwert.
- *Verzögerte Aktualisierung:* lockert das Zeit–Constraint und erlaubt eine verzögerte Aktualisierung oder Darstellung der gemeinsamen Inhalte. Das kann zu einer Darstellung von unterschiedlichen Versionen führen, vor allem dann, wenn die Synchronisierung nur nach größeren Zeiträumen oder auf Nutzeranforderung erfolgt. Werden lokale Änderungen erst dann propagiert, wenn der Nutzer diese freigegeben hat, können lokal Alternativen durchgespielt werden und eine endgültige Variante in den gemeinsamen Informationsbestand übertragen werden.
- *Subgruppenbildung:* ist eine Abschwächung des Populations–Constraints und bedeutet die Darstellung der gemeinsamen Informationspräsentation nur für eine Teilgruppe anstelle des gesamten Teams.
- *Alternative Views*: verletzen das Kongruenz–Constraint und gestatten, dass identische Informationen durch verschiedene Präsentationsformen, beispielsweise als Tabelle oder als Graphik, dargestellt werden können. Genauso kann die Darstellung durch unterschiedliche Ansichten, Perspektiven oder Ausschnitte differieren.

Für existierende Single–User–Applikationen existiert ein vergleichsweise einfaches und damit kostengünstiges Verfahren, diese in WYSIWIS–Multi–User–Applikationen mit zentralisierter Architektur umzuwandeln. Allerdings arbeiten diese dann für die Teammitglieder ‚collaboration transparent', die Applikation ‚bemerkt' ebenso die Nutzung durch mehrere Anwender nicht. Dazu muss im verwendeten Windowsystem ein Multiplexer/Demultiplexer[11] implementiert werden. Der Multiplexer sorgt dafür, dass die lokale Darstellung an die Windowmanager der anderen Teammitglieder weitergeleitet wird. Aufgabe des Demultiplexers ist es, die Eingaben von verschiedenen Workstations an die Applikation weiterzugeben. Zur Sicherung der logischen Konsistenz der gemeinsamen Daten ist es bei der Nutzung ehemaliger Single–User–Applikationen vor allem mit größeren Gruppen wünschenswert, dass nur eine Person Eingaben

[11] Greenberg beschreibt die notwendigen Implementierungsarbeiten der Umwandlung einer Applikation im Detail. In diesem Kontext wird der Multiplexer ‚View–Manager' und der Demultiplexer ‚Chair Manager' genannt.

vornehmen kann. Dazu können verschiedene automatisierte Floor–Control–Verfahren angewandt werden, oder ein Moderator steuert die Eingabeberechtigungen. Eine Registrar–Komponente ist für Konferenzaufbau und –beendigung, für die Maßnahmen beim Eintritt oder Ausscheiden von Personen aus der Konferenz und die Steuerung von Rechten innerhalb der Konferenz verantwortlich. Weiterhin kann ein Meta–Manager integriert werden, der die Verwaltung von Awareness–und Telepresence–Techniken übernimmt, die für die zugrundeliegende Applikation nicht sichtbar sind [Greenberg 90].

Ein bekanntes Beispiel eines für derartige Zwecke angepassten Windowsystems ist Hewlett–Packard's *SharedX* für das X–Window–System.

2.4.2 Tele–Presence– Techniken

Zur rechnernetzwerkbasierten Unterstützung synchroner Kooperation ist die visuelle Präsentation der gemeinsamen Informationen nicht hinreichend. Einerseits fehlen die oben beschriebenen positiven Auswirkungen der Group Awareness, andererseits ist eine effektive Unterstützung der für Teamarbeit wesentlichen Prozesse Kommunikation und Koordination nötig. Mit Tele–Presence–Techniken soll die geographisch verteilte Gruppeninteraktion unterstützt werden, indem versucht wird, den psychologisch–soziologischen Faktor der Präsenz der Teammitglieder durch CSCW–Techniken zu erfassen und zu übertragen [Buxton 92]. Dies betrifft sowohl die explizite Dynamik des Gruppenprozesses als auch subtilere Formen wie Gestik, Körpersprache, Augenkontakt, Stichwörter, das Lenken der Aufmerksamkeit etc. Tele–Presence–Techniken haben also das Ziel, bei geographisch verteiler, synchroner Arbeit die Nutzung der natürlichen Kommunikationstechniken zu befördern, die üblicherweise bei face–to–face–Meetings angewandt werden [Greenberg 92]. Naturgemäß sind Tele–Presence–Techniken ein aktueller und häufig betrachteter Forschungsgegenstand für viele CSCW–Applikationstypen wie Systeme für verteilte elektronische Sitzungen, Unterstützungssysteme für face–to–face–Sitzungen, Systeme für asynchrone Kommunikation, Gruppeneditoren usw. Im folgenden soll sich wiederum auf Techniken beschränkt werden, die für die CSCW–Systemklasse ‚Workgroup Computing' relevant sind.

Telepointer

Ein Weg, der Reduzierung des Gruppenbewußtseins bei abgeschwächten WYSIWIS entgegenzuwirken, ist der Einsatz von Telepointern bzw. die Darstellung des Cursors der anderen Teammitglieder. Ein Telepointer ist ein spezieller graphischer Cursor, der von mehreren Teammitgliedern bewegt werden kann und dessen Position in den Fenstern aller Teammitglieder identisch ist. Bestimmte Systeme wie Xerox's Colab können mehrere statische Pfeildarstellungen des Pointers in der Fensterfläche anbringen, indem während der Pointerbewegung Mausclicks ausgeführt werden [Stefik 87]. Daher können Telepointer zum Verweisen auf Inhalte von öffentlichen WYSIWIS–Fenstern bei der synchronen Teamkooperation verwendet werden. Daher

müssen sie wie eine gemeinsame Ressource verwaltet werden, um die konsistente Darstellung und die Zugriffsberechtigungen zu regeln.

Bei relaxiertem WYSIWIS–Darstellungen müssen sich Telepointer an der logischen Position orientieren. So sollte der Telepointer in einem kollaborativ nutzbaren Texteditor bei allen Teammitgliedern über demselben Wort bzw. Buchstaben befinden, unabhängig davon, welcher exakte Ausschnitt des Dokuments gerade dem einzelnen Teammitglied präsentiert wird.

Teleselections

Bei der kooperativen Dokumentenbearbeitung stellen Selektionen von Objekten häufig einen Zeitpunkt der Unterbrechung der Kooperation dar, da sie Operationen wie Löschen, Bewegen, Kopieren, etc. vorausgehen. Daher ist es wünschenswert, eine graphisch unterscheidbare Darstellung der Selektionen anderer Teammitglieder bereitzustellen, um Konflikten vorzubeugen [Stefik 87].

Eine weitere sinnvolle Anwendung von Teleselections ist der Ersatz von Telepointern in bestimmten Situationen. Wenn die WYSWIS–Darstellung soweit relaxiert wurde, dass Teammitgliedern die gemeinsame Information durch verschiedene Präsentationstypen dargestellt wird, können Telepointer nicht mehr sinnvoll eingesetzt werden, da sich dann die Positionierung der Pointers nicht an der physikalischen, sondern an der logischen Position orientieren muß. In diesem Fall besteht der gemeinsame Kontext nur noch aus denselben Objekten in einer unterschiedlichen Darstellung. Somit wäre die einzige übertragbare Verweisinformation, ob sich der Pointer über einem Objekt befindet und wenn dies der Fall ist, eine Referenz dieses Objekts. Bei einer schnellen Bewegung des Telepointers über den Bildschirm kann das zu schlechten Antwortverhalten der Applikation führen. Eine Lösungsmöglichkeit ist die Anwendung von Teleselections, wie dies im unten beschriebenen GroupPlan–System [Hauschild 00] der Fall ist

Informationelle Kanäle

Es ist wichtig, dass die Gruppenmitglieder zur Koordinierung ihrer gemeinsamen Tätigkeit über einen externen Kommunikationskanal verfügen. Neben Systemfeatures zur Herstellung von Awareness gehören diese Kanäle zu den Awareness–Mechanismen, die auf einer expliziten Generierung und Verteilung der Gruppenbewusstseins– Informationen beruhen. Im Gegensatz dazu gehören Telepointer und Teleselections zu den Mechanismen, die passiv arbeiten und vorhandene Informationen aufnehmen und weiterleiten, um Awareness zu erzielen [Dourish 92]. Durch diese kann eine Synchronisation der Tätigkeiten erfolgen und Konflikte bei Aktivitäten im Ansatz vermieden werden [Borghoff 98].

Dazu können beispielsweise elektronische Mail–Systeme, Chat–Systeme oder Mechanismen zum Anbringen von Anmerkungen und Kommentaren verwendet werden, wie dies in einigen kooperativen Editoren geschieht [Dourish 92]. Für eine engere Gruppenkooperation sollte auf

Video– oder Audiokanäle zurückgegriffen werden, um schnelle und neben der eigentlichen Tätigkeit nutzbare Kommunikationsmöglichkeiten zur Verfügung zu stellen [Posner 92].

Ein anderes Verfahren ist die von Gaver vorgestellte Nutzung der Übertragung nichtverbaler Audio–Informationen [Gaver 92]. Bei diesem Ansatz werden typische Geräusche, die für den Typus der aktuell ausgeführten Tätigkeit spezifisch ist, an die Teammitgleder übermittelt. So könnte beispielsweise das Team über Löschoperationen durch Reißgeräusche informiert werden. Außerdem ist der Einsatz von ambienten Audio– Hintergrundinformationen getestet worden, die als Hinweis auf gerade stattfindende Ereignisse oder zur Erinnerung der Teammitglieder an baldige Termine dienen. Es ist augenscheinlich, das die erste Klasse von nonverbalen Audioinformationen nur für kleine Teams geeignet ist, während die zweite auch bei einer großen Anzahl von Beteiligten gut nutzbar ist.

2.4.3 Mechanismen zur Nebenläufigkeitskontrolle (Concurrency Control)

Bei synchroner Kooperation auf Basis gemeinsamer Informationen ist die Kontrolle von Nebenläufigkeiten eine essentielle Aufgabe.

Der Begriff der Nebenläufigkeitskontrolle spielt auch im Bereich der Parallelverarbeitung und bei Verteilten Datenbanken eine große Rolle. In diesen Bereichen ist das Ziel jedoch, dass nur ein Prozess, der Elemente des Datenbestands modifiziert, zu einer Zeit aktiv sein kann. Unter Konsistenz wird in diesem Kontext ein Zustand verstanden, bei dem alle Modifikationen bereits erfolgt sind oder keine Änderungsanfragen vorliegen.

Diese Herangehensweise ist jedoch für CSCW–Applikationen nicht wünschenswert. Hier liegt das Ziel darin, eine echte synchrone Kooperation aller Teammitglieder zuzulassen und als Ergebnis der Teamarbeit ein Produkt zu erzeugen, dass durch asynchrone und synchrone Modifikationen entstanden ist und den Intentionen des Teams entspricht. Die klassischen Techniken der oben genannten Teildisziplinen der Informatik sind hier nur bedingt anwendbar, da hier eine Interaktion von Programmprozessen und Teammitgliedern vorliegt und die ‚Human Factors' entsprechend Beachtung finden müssen [Dourish 95] [Rodden91]. So würden unverhältnismäßig lange Wartezeiten auf Systemreaktionen oder das Verwerfen der Ergebnisse einer Anzahl von Arbeitsschritten infolge einer durch das System abgebrochenen Transaktion die Nutzerakzeptanz eines derartigen Systems völlig untergraben.

Bei der Anwendung von Concurrency–Control Verfahren in Groupware–Systemen müssen folgende Aspekte berücksichtigt werden [Ellis 91] [Greif 88]:

- *Laufzeitverhalten*: Die CSCW–Applikation muss auch mit hohen Interaktionsfrequenzen bei synchroner Kooperation Schritt halten können. Daher müssen einerseits kurze Antwortzeiten nach Operationen und andererseits kurze Synchronisationszeiten bei der Propagation von Zustandsänderungen an die Teammitglieder gesichert werden.

- *Gruppenschnittstelle*: Es ist wichtig, dass Effekte wie Sperrung oder Wartezeiten auf das Systemverhalten infolge des Concurrency–Control–Verfahrens allen Teammitgliedern und nicht nur den Verursachern über die Nutzerschnittstelle mitgeteilt werden. Ausnahmen von dieser Regel sollten ausschließlich bei nichtunterbrechenden Verfahren zugelassen werden.
- *Weiträumige physikalische Verteilung*: Synchrone Kooperation über Wide Area Networks (WAN's) steigert das Risiko von Verzögerungen oder gar Ausfällen der Kommunikationsverbindung. Die Auswirkungen auf das Antwortverhalten müssen ebenso wie die Reaktion auf Kommunikationsabbrüche betrachtet werden.
- *Datenreplikation*: Aus Performance–Gründen werden die gemeinsamen Daten oft repliziert. Die positiven Auswirkungen auf Lesezugriffe kehren sich jedoch bei Modifikationen um, da alle Repliken aktualisiert werden müssen. Die Verfahren müssen mit dem Fall umgehen können, dass synchron eine nicht notwendigerweise identische Modifikation an demselben atomaren Element in zwei oder mehr replizierten Datenbeständen vorgenommen wird.
- *Robustheit*: Die Verfahren müssen mit ungewöhnlichen Zuständen wie dem Absturz oder Ausfall einzelner Rechner, dem Ausfall der Kommunikationsverbindung evtl. sogar mit einer Partitionierung des Netzwerks und unvorhersehbaren Nutzerverhalten wie dem Abbruch einer Session ohne vorhergehende Abmeldung umgehen können. Daher ist ein leichtes Recovery von solchen Zuständen essentiell.
- *Lange Transaktionen*: Im CSCW–Bereich treten häufig lange oder geschachtelte Transaktionen auf. Dies impliziert einige Probleme: einerseits muss den anderen Teammitgliedern der Grund langer Sperren oder Antwortzeiten transparent gemacht werden, weiterhin erhöht sich die Gefahr, dass größere Mengen Arbeit durch Abbruch einer langen Transaktion verloren gehen und außerdem steigt das Risiko, dass ein Abbruch durch das Versagen von Systemmodulen oder Kommunikationsverbindungen erfolgt.

Klassifikation der Verfahren zur Nebenläufigkeitskontrolle

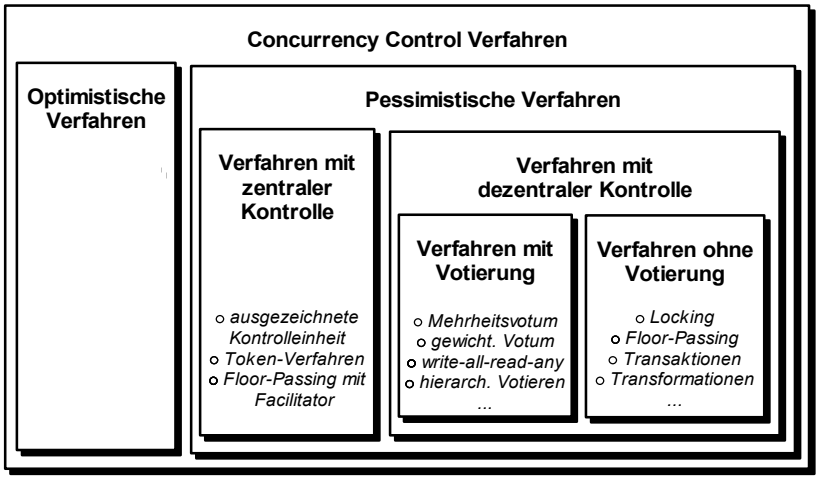

Abbildung 17: Verfahren zur Nebenläufigkeitskontrolle

Es gibt grundsätzlich zwei Klassen von CSCW–Verfahren zur Nebenläufigkeitskontrolle, nämlich optimistische und pessimistische Verfahren. Der Grundgedanke optimistischer Verfahren ist, alle Zugriffe auf den gemeinsamen Datenbestand zuzulassen, da Konflikte relativ selten auftreten. Tritt dennoch ein Konflikt auf, muss dieser erkannt werden, anschließend erfolgt eine Behebung der Auswirkungen des Parallelzugriffs. Während optimistische Concurrency Control Verfahren für Datenbanken immer konsistenzsichernd sind, existieren im CSCW–Bereich auch Methoden, die nicht die Konsistenz der gemeinsamen Daten garantieren [Borghoff 98]. Optimistische Verfahren sind besonders dann zu bevorzugen, wenn das Antwortverhalten des Systems nicht so schnell gestaltet werden kann, das in der Zeit zwischen Objektselektion und Objektnutzung in einem pessimistischen Verfahren über die Zugriffsberechtigung entschieden werden kann. Allerdings sind optimistische Verfahren häufig schwieriger zu implementieren als pessimistische.

Gibt es hohe Anforderungen an die Konsistenz der gemeinsamen Daten, müssen pessimistische Verfahren eingesetzt werden. Diese serialisieren über verschiedene Verfahren die Zugriffe auf die gemeinsamen Informationen oder bestimmen über verschiedene Protokolle eine systemweit gültige Version der Daten und garantieren somit einen konsistenten Datenbestand. Bei den Verfahren mit zentraler Kontrolle existiert entweder eine statische zentrale Kontrolleinheit, die für Zuordnung von Modifikationsrechten bzw. die Verwaltung eines ausgezeichneten Zustands verantwortlich ist oder diese Verantwortlichkeit wechselt dynamisch zwischen den Rechnern des Teams, wobei gesichert wird, dass nur ein Rechner zu einem Zeitpunkt die Kontrolle besitzt. Dabei muss nicht zwingend ein Rechner für den gesamten gemeinsamen Datenbestand verantwortlich sein, sondern die Granularität kann den Erfordernissen des Groupware–Systems angepasst werden.

Nachteile der zentralen Verfahren sind, dass diese empfindlich gegen Ausfälle des Kommunikationsmediums oder des Zentralrechners sind, was zum Erliegen der Teamarbeit führen würde und die hohen Ansprüche an die Verfügbarkeit dieser Maschine. Diese Probleme werden durch Anwendung dezentraler Verfahren beseitigt, die in Methoden mit und ohne Votierung unterschieden werden können.

Im folgenden sollen kurz einige wichtige Typen pessimistischer Concurrency Control Verfahren für CSCW–Applikationen vorgestellt werden. Eine umfassende Behandlung dieses Themas würde deutlich den Rahmen dieses Kapitels sprengen, daher sei auf die recht umfangreiche Literatur zu diesem Thema verwiesen.

Sperrverfahren (Locking)

Das Sperren von Ressourcen ist ein relativ einfaches Verfahren der Nebenläufigkeitskontrolle. Nur der Eigentümer der Sperre kann die gesprerrte Ressouce modifizieren, bei einigen Sperrverfahren werden nichtmodifizierende Zugriffe auf die Ressource erlaubt. Die Sperrung der Ressource kann explizit über eine Lock–Anforderung geschehen oder implizit beispielsweise durch Selektion des Datenfeldes in der Nutzeroberfläche erfolgen. Die Sperrung von Ressourcen wird häufig allen Teammitgliedern in der Nutzerschnittstelle bekannt gemacht, um die

Wahrscheinlichkeit von abgewiesenen Sperranfragen zu verringern. Dies kann beispielsweise durch (graue) Umfärbung der Informationen bzw. des Hintergrundes oder durch ikonische visuelle Indikatoren geschehen.

Ein Problem von CSCW–Applikationen ist die Möglichkeit, dass explizit angeforderte Sperren unverhältnismäßig lange aufrecht erhalten werden oder dass die Aufhebung der Sperre schlicht durch den Nutzer vergessen wird. Allgemeine Ansätze der Behebung dieses Problems sind Tickle–Locks und Probabilistic–Locks.

Tickle–Locks bleiben nur solange wirksam, wie die Anwendung des Eigentümers der Sperre aktiv ist. Bleibt die Anwendung des Eigentümers einen gewissen Zeitraum inaktiv, wird die Sperre durch das System aufgehoben und die gemeinsamen Informationen für die Teammitglieder wieder zugreifbar.

Bei Probabilistic–Locks wird versucht, innerhalb eines gewissen Zeitraums die gewünschte Information zu sperren. Gelingt dies nicht, kann die Bearbeitung auf Verantwortung des Benutzers fortgesetzt werden. Rechtzeitig zugeordnete Sperren bleiben nur für einen definierten Zeitraum wirksam, nach dem Timeout wird die Sperre entfernt und der Benutzer muss sich evtl. um die erneute Zuordnung der Sperre bemühen. Probabilistic Sperren sichern nicht in jedem Falle die Informationskonsistenz, außerdem kann es passieren, dass durch Kommunikations-verzögerungen mehrere Anwendungen glauben, im Besitz der Sperre zu sein.

Wichtig bei der Anwendung von Locking ist die Wahl einer geeigneten Granularität. Sofern es im Kontext des zu realisierenden Groupware–Systems realisierbar ist, sollte die Granularität in einer geeigneten Weise durch den Anwender definiert werden. Ist sie sehr fein, fordert die Lock–Verwaltung einen beträchtlichen Einsatz von Systemressourcen wie CPU–Zeit, Speicherplatz und Netzbandbreite, ist sie zu groß, werden Nutzeraktivitäten unnötig beschränkt. Ebenso ist zu hinterfragen, für welche gemeinsamen Informationen derartige Maßnahmen nötig sind und wo optimistische Verfahren nutzbar sind [Borghoff 98] [Ellis89] [Ellis91] [Greenberg 94] [Greif 88].

Floor–Passing

Die Floor–Control–Verfahren gehören zu den ‚Turn Taking Protocols' und realisieren somit eine im Team temporal wechselnde Zuordnung der Zugriffsrechte auf gemeinsame Ressourcen durch Bildung einer Reihenfolge. Wie die Bezeichnung ‚Floor–Control' schon sagt, ist das Vorbild dieser Verfahren die face–to–face Konferenz, bei der die Redner wünschenswerter Weise über die gemeinsame Ressource ‚Redezeit' nacheinander verfügen. Die Berechtigung für den Zugriff auf die gemeinsamen Ressourcen wird als *floor–token* bezeichnet, der aktuellen Besitzer dieser wird *floor–holder* genannt.

Diese können in Verfahren mit oder ohne Zeitbeschränkung sowie in explizite und implizite Methoden[12] klassifiziert werden. Explizite Verfahren arbeiten mit einer aktiven Weitergabe der Steuerung durch die Benutzer, d.h. der *floor–token* wird von Teammitglied T_i an Teammitglied T_{i+1} weitergereicht. Kommen neue Teilnehmer hinzu, werden sie in die Reihenfolge der potentiellen Token–Empfänger aufgenommen, abgemeldete Personen erhalten keine Token mehr.

Beim impliziten Floor–Passing wird die Weitergabe durch das System vorgenommen. Diese Verfahren sind normalerweise fairer, da die Ressourcenverteilung von einer unabhängigen Instanz vorgenommen wird bzw. nach Ablauf der Zeiteinheit der Token entzogen und dem Nachfolger übergeben wird. Für die Steuerung der Weitergabe des Tokens kann eine zentrale Stelle existieren, die *Facilitator* genannt wird. Der Facilitator findet sein Analogon beim Chairman von face–to–face–Konferenzen, allerdings stehen CSCW–Applikationen wenn gewünscht wesentlich durchgreifendere Möglichkeiten zur Verfügung, da nur die Eingabe– und Kommunikationskanäle gesperrt werden müssen. Dies bringt die Nachteile aller zentralisierten Verfahren nach sich, fällt die zentrale Instanz oder die Kommunikationsverbindung aus, werden sämtliche weiteren Zugriffe verhindert.

Vor der Anwendung von Floor–Control–Verfahren für informationelle Kanäle ist zu prüfen, ob für mehrere Fenster bei WYSIWIS–Applikationen oder mehrere Medien, z.B. Audio und Video, ein gemeinsamer ‚Floor' realisiert wird, oder ob separate ‚Floors' angemessen sind. Ebenso kann es gerade in bezug auf Telekonferenzsysteme wünschenswert sein, die Audio– und Videokanäle durch soziale Gruppenprotokolle steuern zu lassen und Floor– Control nur für die Fenster des Konferenzunterstützungssystems anzuwenden.

Weiterhin ist es nötig, auf das Antwortverhalten des verwendeten Verfahrens zu achten. Verfahren mit für den Anwender spürbaren Wartezeiten bei einer impliziten Anforderung des Floors sind nicht akzeptabel [Borghoff 98] [Burger 97] [Crowley 90] [Lauwers 90a].

Transaktionsverfahren

Die Serialisierung von Nebenläufigkeiten durch Transaktionsverfahren ist im Bereich der Verteilten Systeme eine häufig genutzte Herangehensweise. Gerade diese Technik ist aber im allgemeinen Kontext von CSCW–Applikationen nur bedingt und für interaktive Real–Time Anwendungen nahezu ungeeignet. Das ist wenig überraschend, da hier ein grundlegender philosophischer Unterschied zwischen Datenbanken und Groupware–Systemen deutlich zu Tage tritt. Während bei Datenbanken und vielen anderen Multi–User–Umgebungen das Ziel ist, jedem Anwender die Illusion zu vermitteln, dass er der einzige Benutzer des Systems ist, verfolgen CSCW–Applikationen erklärtermaßen das gegenteilige Ziel.

Wird beispielsweise das mehrstündige Edieren eines Dokuments als lange Transaktion implementiert, könnte das zu einer unzumutbar langen Aussperrung der anderen

[12] Explizite Verfahren werden in der Literatur auch als ‚Baton Mode' – Verfahren, implizite Verfahren dann als ‚Designation Mode' –Verfahren bezeichnet.

Teammitglieder führen. Änderungen, die sich während der Transaktion ergeben, würden sehr lange für andere Teammitglieder nicht sichtbar werden, bevor es zum Commit kommt. Der Abbruch einer langen Transaktion in einer CSCW–Umgebung ist teuer, da die Resultate einer größeren Menge Arbeitszeit verloren gehen können. Dies würde über einen nicht klar erkennbaren Transaktionsbeginn noch weiter verschärft werden, da nach dem Abbruch der Transaktion sich die betroffenen Teammitglieder einen Überblick über den von System hergestellten Zustand der gemeinsamen Informationen verschaffen müssten. Treten während einer langen Transaktion Netzpartitionierungen oder Rechnerabstürze auf, ist schwer zu entscheiden, ob die Transaktion erfolgreich beendet oder abgebrochen wurde.

Werden andererseits sehr feingranulare Transaktionen eingesetzt, beispielsweise pro Tastendruck bei einem kollaborativen Editor, würde sich eine immense Systemlast ergeben und die Responsivität des Groupware–Systems wäre wiederum in Frage gestellt. Außerdem kann sich nach Greenberg ein wenig erwartungskonformes Verhalten der Nutzerschnittstelle ergeben. Würden beispielsweise in einem kollaborativen Zeichenprogramm zwei Personen ein Zeichenobjekt eine bestimmte Distanz in unterschiedliche Richtungen bewegen wollen, würde es zum Hin– und Herspringen des Objektes kommen, bis die Objektbewegung beendet wird.

Häufig werden optimistische Transaktionsverfahren eingesetzt, die alle Zugriffe auf die gemeinsamen Informationen zulassen, aber gegen Ende der Transaktion testen, ob es Anzeichen für Konflikte gibt. Je nach Systemstrategie wird in diesem Fall die Transaktion durch das System abgebrochen oder der Konflikt wird den betroffenen Teammitgliedern mitgeteilt, die dann einen Abbruch auslösen können oder die Überprüfung und Behebung der Auswirkungen des Konflikts selbst übernehmen.

Transaktionsverfahren sind dann gut einsetzbar, wenn asynchrone Gruppenarbeit unterstützt werden soll, da dann keine kritischen Auswirkungen auf das Interaktionsverhalten der Applikation auftreten. Ebenso sind Transaktionen gut bei Systemen einsetzbar, bei denen die Teammitglieder synchron in privaten Kontexten arbeiten, die später mit einer Merge– Operation zusammengefügt werden. [Borghoff 98][Ellis 91][Greenberg 94][Greif 88][Rodden 91]

Transformationsverfahren

Transformationsverfahren erlauben ein sehr gutes Interaktionsverhalten auch bei einer sehr feingranularen Kontrolle von Nebenläufigkeiten. Daher werden diese Verfahren häufig für eng gekoppelte, synchrone Kooperation bei völlig replizierenden Architekturen verwendet. Häufig kann bei diesen Verfahren ohne Sperren ausgekommen werden, weiterhin sind diese Verfahren robust gegen Ausfälle von einzelnen Rechnern.

Ein bekanntes Verfahren ist die ‚distributed Operational Transformation' (dOPT) des oben eingeführten kollaborativen Texteditors GROVE [Ellis 91], bei dem das Verfahren mit der Granularität eines einzelnen Eingabezeichens ausgeführt wird, um beim gemeinsamen Edieren von Textpassagen jede Änderung sofort auf allen Workstations darstellen zu können. Jeder

Benutzer verfügt hier über eine Kopie des Editors. Wird eine Operation durch den Anwender beispielsweise durch Tastendruck angefordert, wird diese sofort lokal ausgeführt. Danach findet ein Broadcast der Operation gemeinsam mit einem Statusvektor statt, der angibt, wieviele Operationen von anderen Workstations schon lokal ausgeführt worden sind. Da jede Workstation über einen eigenen derartigen Statusvektor verfügt, ist ein Vergleich des mit der Operationsanforderung empfangenen Statusvektors mit dem lokalen Zustand möglich. Sind die Vektoren äquivalent, wird die empfangene Operation sofort ausgeführt. Hat die Empfänger–Workstation Operationen noch nicht ausgeführt, wird die Operationsanforderung als ‚zukünftige Operation' in eine Warteschlange gestellt. Hat die Empfänger–Site Operationen schon ausgeführt, die der Sender noch nicht ausgeführt hat, erfolgt eine Transformation der ‚zurückliegenden' Operation vor der Ausführung. Diese Transformation ist abhängig vom Typus der Operation und von der Historie der bereits ausgeführten Operationen. Ziel der Transformation ist es, Abhängigkeiten zwischen den Operationen auszugleichen, um allen Teammitgliedern ein einheitliches Resultat unabhängig von der Ausführungsreihenfolge der einzelnen Operationen präsentieren zu können. Dazu dient eine quadratische Transformationsmatrix, deren Rang der Kardinalität der Menge der Operationen entspricht. Die Elemente der Matrix sind Funktionen, die Operationen in andere Operationen genau so transformieren, dass bei zwei überlappenden Operationen i und j die Ausführung der transformierten Operation j auf die Operation i dasselbe Resultat liefert wie Anwendung der transformierten Operation i auf die Operation j.

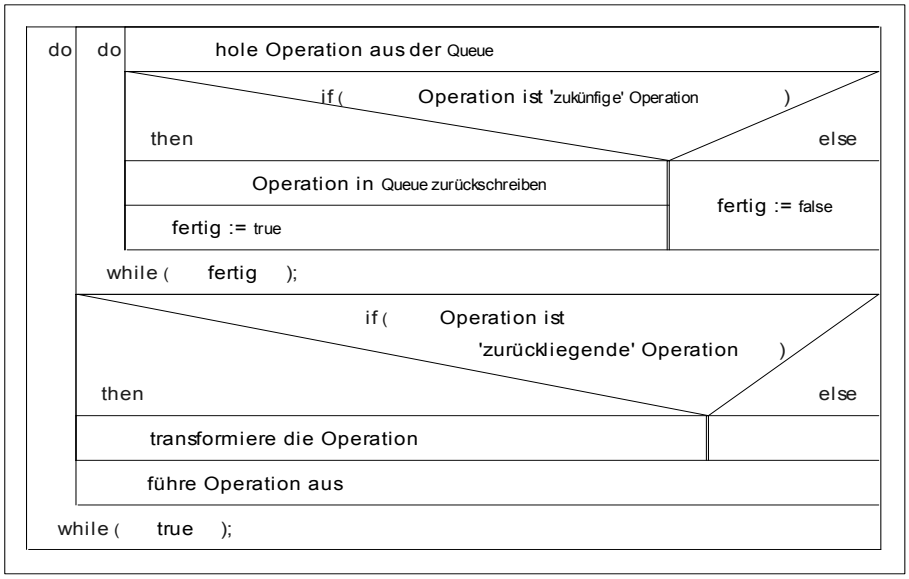

Abbildung 18: dOPT-Transformationsalgorithmus von GROVE

Transformationsverfahren haben jedoch auch wesentliche Nachteile. Einerseits ist die Erstellung einer Transformationsmatrix in keiner Weise trivial, vor allem, wenn die Menge der Operationen groß ist. Ein anderer Nachteil ist, dass eine ständig wachsende Historie mitgeführt werden muss, die bei allen Transformationen durchmustert wird und ab einer bestimmten Größe zur

Degradation der Systemleistung führt. Bei GROVE musste deshalb etwa pro Minute eine Pause von zehn Sekunden eingelegt werden, um alle Operationen in den Queues abzuarbeiten, damit anschließend alle Historien auf leer zurückgesetzt werden konnten. Die seit der Entwicklung von GROVE deutlich angestiegene verfügbare CPU–Leistung dürfte dieses Problem für ‚kollaborative Texteditoren' mildern. Bei Domänen, die kompliziertere Transformationen erfordern, sollte beim Systementwurf betrachtet werden, ob mit dem Einsatz derartiger Verfahren eine flüssige Teamarbeit gewährleistet werden kann [Ellis 89] [Ellis 91] [Greenberg 94] [Borghoff 98].

Votierungsverfahren

Votierungsverfahren stammen ursprünglich aus dem Bereich der verteilten Dateisysteme, werden aber auch zunehmend im Kontext von CSCW–Applikationen eingesetzt. Sie sind vor allem dann gut geeignet, wenn größere Datenblöcke[13] oder ganze Dateien als Zugriffseinheiten behandelt werden sollen, weil dann der Aufwand des nötigen Abstimmungsverfahrens gerechtfertigt wird. Vorteile der Votierungsverfahren sind die hohe Robustheit des Verfahrens und die hohe Verfügbarkeit der so verwalteten Ressourcen, die nicht auf Kosten der Konsistenz der Daten erreicht wird. Damit eignen sich diese Verfahren vor allem für die Unterstützung asynchroner Kooperation.

Es wird davon ausgegangen, dass von Datenblöcken mit gemeinsamen Informationen eine Anzahl Repliken auf verschiedenen Rechnern existieren. Für die Datenblöcke werden Versionsnummern und der Sperrzustand verwaltet. Gibt es Zugriffswünsche auf einen Datenblock, wird eine Abstimmung unter den Rechnern vorgenommen, die eine Kopie des Datenblocks oder nähere Informationen über den Datenblock besitzen. Der Zugriff wird dann gestattet, wenn das Votum eine vorgegebene untere Schranke erreicht oder übertrifft und mindestens einer der permissiv votierenden Rechner die aktuelle Version des Datenblocks besitzt. Im Ergebnis des Votums wird die nach dem Zugriff entstehende Version als allgemein verbindlich betrachtet und auf die entsprechenden Rechner repliziert. Es existieren unterschiedliche Verfahren zur Durchführung des Votums; verschiedene Votierungsverfahren werden in [Borghoff 98] detailliert betrachtet.

2.4.4 Weitere CSCW–Techniken

Im vorangehenden Abschnitt wurde dargestellt, dass es im Bereich des Concurrency Control eine Anzahl von klassisch zu nennenden Methoden in anderen Teilbereichen der Informatik gibt, die aber aufgrund der Spezifik von Groupware–Systemen bei solchen nicht oder nur adaptiert eingesetzt werden können. Die Problematik, dass Standardmethoden der Informatik nicht uneingeschränkt für Computer Supported Cooperative Work übernommen werden können, tritt auch außerhalb der Kontrolle von Nebenläufigkeiten auf. Exemplarisch sollen hier die Bereiche Zugriffskontrolle und Undo/Redo genannt werden, die im Laufe der weiteren Kapitel von Interesse sein werden.

[13] Borghoff und Schlichter verstehen darunter Datenblöcke ab mehreren KBytes.

Zugriffskontrolle (Access Control)

Mechanismen zur Zugriffskontrolle dienen dazu, Anwendern den Zugriff auf diejenigen Informationen und Ressourcen entsprechend ihrer Zugriffsrechte zu gestatten oder zu verweigern. Diese Rechte können sowohl personen– als auch rollengebunden sein, ebenso spezifizieren bestimmte Groupware–Systeme bestimmte Zugriffsrechte auch dynamisch in Abhängigkeit der auszuführenden Arbeitsschritte. Traditionelle Verfahren mit Lese–, Schreib– und Ausführungsrechten für Eigentümer, Gruppe und sonstige Benutzer, wie sie in vielen UNIX®–Derivaten zu finden sind, können weder in Hinblick auf ihre Flexibilität noch auf ihre Ausdruckskraft als hinreichend betrachtet werden. Das in modernen Betriebssystemen genutzte Access Control Lists(ACL)– Verfahren ist zwar ausdrucksstärker, erfüllt aber ohne Anpassungen ebenso nicht in jedem Falle die Anforderungen von Groupware–Systemen.

Beispielsweise ist es bei diesen Verfahren schwer möglich, Modifikationen eines gemeinsamen Datenbestandes zu verweigern, aber gleichzeitig Anfügungen an diesen zuzulassen. Rechte zum Aufbau oder zur Modifikation einer gemeinsamen WYSIWIS–Darstellung oder zur Erstellung einer Bezugnahme auf Informationen aus dem Datenbestand eines Partners sind ebenso nicht direkt abbildbar.

An eine effektive Zugriffskontrolle für CSCW–Applikationen sind nach Trevor und Rodden sowie Dewan und Shen folgende Anforderungen zu stellen:

- dynamische Zuordnung von Rechten wegen häufiger Änderungsfrequenz
- Ableitung der Zugriffsrechte aus dem aktuell auszugebenden Arbeitsschritt oder der aktuell ausgeübten Rolle, dabei können auch multiple Rollen einer Person zu einem Zeitpunkt auftreten
- einfaches Management der Zugriffsrechte durch eine Gruppe von Nutzern anstelle eines im allgemeinen nicht zum Team gehörenden Systemadministrator
- Realisierung einer feinen Granularität der Zugriffskontrolle mit der Möglichkeit der Spezifizierung der Rechte jedes einzelnen Nutzers auf jedes einzelne Objekt
- effiziente Speicherung und Auswertung der Zugriffsrechte
- einfache Implementierbarkeit des Zugriffsmodells in Multi–User–Umgebungen

Da die durch das Betriebssystem bereitgestellten Access Control Mechanismen im allgemeinen nicht hinreichend sind, muss die CSCW– Applikation oft eine eigene Zugriffskontrolle[14] realisieren. Dafür sind je nach den Anforderungen der Applikation adaptierte ACL– Verfahren einsetzbar, sehr häufig werden Zugriffsmatrix– basierte Access Control Mechanismen eingesetzt.

---

[14] Eine technische Voraussetzung der Realisierung einer umfassenden eigenen Zugriffskontrolle unter aktuellen Betriebssystemen ist, dass die CSCW–Applikation als *trusted application* über das *setuid*–Privileg besitzt, d.h. über die globalen Zugriffsrechte wie die Kernprozesse des Betriebbsystem verfügt.

Das Matrixverfahren nach Lampson, Graham und Denning basiert auf der Definition von Zugriffsrechtszuständen und von Zustandstransitionen. Ein Zugriffsrechtszustand (protection state) ist definiert als Tripel (S,O,A), wobei unter S die Subjektmenge, also Personen und Prozesse mit Zugriffsanforderungen und O die Menge der zuzugreifenden Objekte ist. A stellt eine Matrix dar, deren Zeilen die Subjekte und deren Spalten die Objekte repräsentieren. Die Matrixelemente A[s,o] stellen die Zugriffsrechte der Subjekte auf die Objekte dar. Von Shen und Dewan wurde dieses Modell für das Suite–Framework um eine Menge von Rechten zur Kollaboration, um negative Rechte zum expliziten Zugriffsabweis und um eine Rechte–Vererbung erweitert. Darunter wird in diesem Kontext die Übertragung der Zugriffsdefinitionen eines nicht spezifizierten Zugriffsmatrixelements von der zugeordneten Subjekt– und/oder Objekt– Gruppe verstanden.

Eine weitere, allgemeine Erweiterung des Matrixverfahrens wurde von Smith und Rodden vorgenommen. Bei deren Verfahren werden die User U mit ihren Rollen R sowie die Objekte O und die Zugriffsrechte P der Nutzer auf die Objekte verwaltet. Für jedes Objekt wird eine dynamische Zugriffsmatrix erstellt, die zwei synchron erweiterbare Spalten mit Einträgen von Zugriffsrechten auf spezifische Eigenschaften oder Operationen des Objekts enthält. Die erste Spalte repräsentiert die Zugriffsrechte der Nutzer auf die Eigenschaften oder Operationen des Objekts. Die zweite Spalte enthält Informationen darüber, ob die Ergebnisse des Zugriffs an andere Nutzer propagiert werden dürfen. Smith und Rodden schlagen für jede Nutzer– Objekt– Kombination eine derartige Matrix mit bitweisen Einträgen pro Spalte vor. Den Matrixelementen wird weiterhin einer von vier möglichen Zuständen zugeordnet: (i) der Zugriff ist immer erlaubt, (ii) der Zugriff ist niemals erlaubt, (iii) der Zugriff ist standardmäßig erlaubt, aber entziehbar oder (iv) der Zugriff ist als Default verboten, kann aber durch den Nutzer genehmigt werden.

Neben der Realisierung eines effektiven Mechanismus zur Zugriffskontrolle ist es aber anzustreben, bei Systemen mit anwenderdefinierbaren Rechten auch diese Rechtedefinition zu überwachen. Diese Systemkomponente wird als ‚Meta Access Control' bezeichnet. Hauptaufgabe ist, die Rechte der Teammitglieder in bezug auf das Einräumen oder Entziehen der Zugriffsrechte auf ihre Informationen zu regeln und nebenher eine komfortable Oberfläche zur Rechtezuordnung bereitzustellen. Die technische Umsetzung der Metazugriffskontrolle basiert auf der Anwendung geeigneter Kontrollmechanismen wie Zugriffsmatrizen oder Access Control Lists auf die Informationen, mit denen die Zugriffsrechte der gemeinsamen oder privaten Informationen der Teammitglieder verwaltet werden [Shen 92] [Dewan 98] [Dewan 98a] [Ellis 91] [Greif 88] [Trevor 94] [Smith 93].

Undo/ Redo

Die Möglichkeit, dass ein Anwender seine letzten Aktionen durch die Applikation rückgängig machen (undo) und diese danach wieder neu ausführen (redo) lassen kann, hat sich als nützliches und der Systemakzeptanz sehr zuträgliches Feature erwiesen. Neben der eigentlichen Funktion dieses Features zur Rückgängigmachung von Fehlern sollten einerseits die Möglichkeiten zur

Exploration von verschiedenen Varianten und andererseits die positiven psychologischen Auswirkungen auf unerfahrene Anwender mit Ängsten vor rechnergestützten Verfahren beachtet werden.

Bei der Integration von Undo–Mechanismen in CSCW–Applikationen zur Unterstützung synchroner Teamarbeit treten nach Prakash und Knister Schwierigkeiten in drei Bereichen auf. Erstens muss die korrekte Operation gefunden werden, die rückgängig zu machen ist. In einer kollaborativen Umgebung ist die letzte Operation einer Person nicht zwangsläufig die global letzte Aktion, da das vorliegende Teamdokument aus den Eingabeströmen aller Teammitglieder erstellt wurde. Nutzer erwarten aber häufig, dass ihre letzte Operation zurückgenommen wird. Somit muss ein Mechanismus zur Selektion der betroffenen Operation realisiert werden. Zweitens muss der Effekt der Rückgängigmachung betrachtet werden. Eine einfache Ausführung der inversen Operation führt in den Fällen nicht zum gewünschten Resultat, in denen Modifikationen des Dokuments durch andere Nutzer auch diese Operation betreffen. Drittens ist es nicht in jedem Falle möglich, Operationen eines Nutzers rückgängig zu machen, ohne gleichzeitig Operationen anderer Nutzer ebenso zu annulieren. Dies tritt vor allem dann auf, wenn Undo–Operationen in überlappenden Arbeitsbereichen mehrerer Teammitglieder ausgeführt werden.

Häufig genutzte traditionelle Verfahren für Undo–Operationen sind Single–Step–Undo, Undo nach dem linearen oder US&R–Modell sowie History–Undo. Single–Step–Undo gestattet nur das Annullieren der letzten Operation und unter Umständen ein Redo, was häufig als Undo–Operation der Undo–Operation realisiert wird. Das lineare Modell erlaubt die Rückgängigmachung mehrere Operationen beginnend bei der letzten Operation und ein Redo von Operationen in der Undo–Reihenfolge. Das Undo–Skip&Redo (US&R)–Modell ähnelt dem linearen Modell, erlaubt aber das Überspringen (Skip) von Operationen beim Redo. Ein Nachteil des linearen und US&R– Modells ist, dass zum Annullieren einer zurückliegenden Operation eine Anzahl danach ausgeführter Operationen erst rückgängig und anschließend wieder neu ausgeführt werden müssen, was in Gruppenumgebungen als störend empfunden würde. Beim History–Undo werden im Unterschied zu den beiden letztgenannten Modellen auch Undo–Operationen in eine Historie eingetragen, was in jedem Falle eine Wiederherstellung vorangegangener Zustände erlaubt.

Prakash und Knister schlagen für kollaborative Umgebungen *Selective Undo* vor, das auf History–Undo beruht, aber mit den genannte Problemen derartiger Techniken in kooperativen Umgebungen umgehen kann. Selective Undo erlaubt die selektive Rückgängigmachung von Operationen und behandelt die mögliche Verschiebung des Operanden sowie Konflikte explizit. Zur Umsetzung werden History–Listen genutzt, in deren Einträgen der ausführende Nutzer mitgeführt wird. Diese Nutzerkennung wird zum Annulieren der letzten Aktion eines Anwenders genutzt, das Verfahren kann aber auch mit Attributen mit anderer Semantik arbeiten. Die Applikation muss die Funktionen *Inverse*, *Conflict* und *Transpose* bereitstellen. Die Funktion

Inverse(A) liefert zu jeder Operation A die inverse Operation ~A. Die Funktion *Conflict(A,B)* liefert ein boolsches Resultat, das ausssagt, ob B von A abhängig und damit ohne A nicht sinnvoll ist. *Transpose(A,B)* wird im Falle der Konfliktfreiheit genutzt und liefert zwei neue Operationen A‘ und B‘, die auf bestehende Informationen in der Reihenfolge B’A‘ ausgeführt dasselbe Resultat haben wie AB. Diese *Transpose*– Operation ist mit der Transformation im GROVE–Algorithmus verwandt, sie muss jedoch nur dann definiert sein, wenn keine Konflikte vorliegen. *Selective Undo* ist ein allgemein einsetzbares Verfahren, überlässt aber die Realisierung der drei genannten Funktionen dem Designer des Groupware–Systems. Weiterhin kommt es hin und wieder zu für den Anwender nicht erwartungskonformen Abweisungen der Undo–Aufforderung aufgrund von Konflikten, daher sollten die Möglichkeiten des Auftretens von Konflikten schon ab der Designphase minimiert werden [Prakash 92] [Prakash 94].

2.5 Implementierungsbasis: Verteilte Systeme

2.5.1 Verteilte Systeme: Definitionen und Klassifikation

Verteilte Systeme bestehen aus einer Sammlung autonomer Computer, die durch ein Netzwerk verbunden sind und mit einer Software zur Kommunikation, Kooperation oder Lastminderung ausgestattet sind [Coulouris 94] [Borghoff 98]. In Verteilten Systemen können sowohl Hardware–Komponenten, Last, Daten, Kontrolle oder Verarbeitung verteilt vorliegen.

Soll eine Integration der Applikationen und Modelle in der Bauwerksplanung auf Systemebene vollständig auf dateibasierten Informationsaustausch verzichten, ist die Nutzung Verteilter Systeme unabdingbar. Aber auch bei einer filebasierten Integrationsumgebung kann der Einsatz Verteilter Systeme beispielsweise zur Übertragung der Dateien an Projektpartner, z.B. über http- oder ftp- Protokolle, interessant sein. Die Bedeutung von Verteilten Systemen hat in den vergangenen zwei Jahrzehnten stark zugenommen. Infolge der Verfügbarkeit preisgünstiger CPU's hat sich die betriebliche Datenverarbeitung weg von zentralisierten Lösungen hin zu dezentralen Strukturen entwickelt. Um aber die betrieblichen Informationsflüsse zu gewährleisten, war das Vorhandensein hinreichend leistungsfähiger Netzwerktechnologien eine weitere Voraussetzung der 'Client-Server-Revolution'. Es kann konstatiert werden, dass mittlerweile eine hohe Verbreitung von lokalen Netzwerken und breitbandigen Zugängen zum Internet oder anderen Weitverkehrsnetzen sowohl in Firmen aller Größen bis hin zu Privathaushalten vorliegt.

Basis der Nutzung Verteilter Systeme ist neben der Rechnerhardware eine geeignete Rechnernetzinfrastruktur und die zum Betrieb notwendige Software. Hier soll im folgenden die Middleware genannte Betriebssoftware für Verteilte Systeme thematisiert werden, für weitergehende Betrachtung zur Hardware, Topologien und Protokollen von Netzwerken sei auf die umfangreiche Literatur zu diesen Punkten verwiesen. Die Middleware erbringt eine Transport– und Verwaltungsschicht für Informationen und Prozesse über ein Netzwerk hinweg.

Typischerweise sind Middleware–Systeme plattform– und protokollunabhängig und gestatten die Nutzung einer Anzahl von Programmiersprachen zur Realisierung von Applikationen.

Coulouris et al. nennen sechs charakteristische Eigenschaften, die für die Effektivität der Nutzung Verteilter Systeme verantwortlich sind, diese sind:

- Teilung von Ressourcen: Verteilte Systeme teilen Hardware- oder Informationsressourcen auf mehrere Knoten auf, dies trifft letztendlich auch auf Systeme mit Last– und Verarbeitungsverteilung zu. CSCW–Techniken beruhen sehr stark auf einer Verteilung von Informationen durch Applikationen auf einer Anzahl Workstations.
- Offenheit: Die Offenheit eines Systems definiert den Grad dessen späterer hardware– oder softwareseitiger Erweiterbarkeit. In Verteilten Systemen bezieht sie sich meist auf den Schwierigkeitsgrad der Hinzufügung weiterer Dienste. Durch die Entwicklung von Interprocess Communication (IPC) –Mechanismen in UNIX–Systemen wie BSD begann allgemein der Öffnungsprozess der vorher geschlossenen, monolithischen Systemsoftwaresysteme, der zu neuen Blickwinkeln beim Design von Rechnersystemen führte. Im nächsten Schritt wurde es möglich, auch über Rechnergrenzen hinweg und später auch zwischen heterogener Hardware zu kommunizieren.
- Nebenläufigkeit: Verteilte Systeme überdecken im Normalfall mehrere Hardwareknoten. Da einerseits mehrere Nutzer oder Applikationen gleichzeitig Dienste anfordern können und andererseits auf Servern mehrere Prozesse zur Abarbeitung der Anfragen (quasi–)parallel bearbeit werden können, sind Nebenläufigkeiten in Verteilten Systemen die natürliche Form der Prozessorganisation. Infolge dessen sind in Verteilten Systemen Vorkehrungen zur Synchronisation der Zugriffe auf gemeinsame Ressourcen notwendig.
- Skalierbarkeit: Verteilte Systeme können in verschiedenen Größenordnungen betrieben werden, die Spanne reicht von kleinen LAN's mit zwei bis drei Knoten bis hin zu Internetservern mit extrem hohen Clientzahlen. Gut skalierbare Verteilte Systeme erfordern keinen Wechsel der Middleware, wenn die Clientzahlen im Laufe der Zeit drastisch ansteigen.
- Fehlertoleranz: Mit dem Ansteigen der Komplexität der Verarbeitungsprozesse wächst die Wahrscheinlichkeit des Auftretens von Fehlern verschiedener Art. In Verteilten Systemen können zur Kommunikation benötigte Netzwerke genauso wie einzelne Knoten oder einzelne Hardwarekomponenten ausfallen, verschiedene Ebenen der Software können inkorrekte Resultate liefern oder den Betrieb einstellen, ebenso können Verzögerungen an verschiedenen Stellen auftreten. Verteilte Systeme können diesen verschiedenartigen Fehler– und Ausfallquellen mit zwei Strategien begegnen, einerseits kann Hardware redundant ausgelegt werden, andererseits müssen alle Softwaresysteme so entworfen werden, dass ein Recovery nach dem Auftreten von Fehlern möglich ist.

- Transparenz: Die Transparenzeigenschaften eines Verteilten Systems sind essentiell für dessen einfachen, sicheren und ökonomischen Betrieb. Der Grad der Erfüllung der einzelnen Transparenzeigenschaften variiert in Abhängigkeit von den genutzten Kommunikationsphilosophien und der eingesetzten Middleware und Hardware. Einige besonders wichtige Arten von Transparenz sind:
    - Zugriffstransparenz – die Zugriffsmechanismen für sämtliche lokalen und entfernten Ressourcen und Dienste sind identisch,
    - Ortstransparenz – der genaue Ort von Ressourcen und Diensten muss einem Client nicht bekannt sein, da diese über Namen angesprochen werden können, die keine näheren Informationen über den Aufenthaltsort des Dienstes beinhalten,
    - Nebenläufigkeitstransparenz – die gleichzeitige Nutzung des Systems durch mehrere Anwender oder Prozesse bleibt diesen verborgen, indem für diese durch transparente Synchronisation die Illusion eines exklusiven Zugriffs geschaffen wird,
    - Replikationstransparenz – für Clients ist nicht erkennbar, ob sie auf die Kopie oder das Original eines Informationsbestandes oder eines Dienstes zugreifen, ebenso ist nicht erkennbar, ob und wie oft das Original repliziert wurde,
    - Fehlertransparenz – Ausfälle des Systems oder des Kommunikationsmediums werden vor dem Anwender verborgen,
    - Migrationstransparenz – eine Relokation von Ressourcen und Dienste ist möglich, ohne dass Clients oder Anwender davon Kenntnis erlangen,
    - Leistungstransparenz – dynamische Rekonfigurationen des verteilten Systems sind möglich, um Lastausgleich und damit optimale Performance zu erreichen,
    - Skalierungstransparenz – eine hard- oder softwareseitige Erweiterung oder Modifikation des Systems ist möglich, ohne dass Änderungen an der Systemstruktur erfolgen müssen,
    - Sprachtransparenz – verschiedene Komponenten des Verteilten Systems können in unterschiedlichen Programmiersprachen implementiert werden.

Das Gebiet der Verteilten Systeme überdeckt einen sehr großen Bereich von Systemen mit unterschiedlichsten Philosophien, Mechanismen und Einsatzzwecken. Es ist kein einheitliches und allgemeingültiges Klassifizierungsschema für das Gesamtgebiet bekannt, obwohl diese in abgegrenzten Teilgebieten durchaus existieren. Aufgrund der verflochtenen Beziehungen zwischen den einzelnen Teilgebieten würde die Suche nach einem allgemeingültigen Ansatz problematisch erscheinen.

Für Integrationszwecke in der Bauwerksplanung sind vor allem diejenigen Typen von Verteilten Systemen interessant, die sich primär für Kommunikationszwecke und das Abfordern entfernter Verarbeitungsleistungen eignen. Andere Typen Verteilter Systeme mögen auch für spezielle

Aufgaben im Bauwerksentwurf relevant sein, beispielsweise Verteilte Clusterbetriebssysteme bei der Ausführung numerisch anspruchsvoller Berechnungen oder Verteilte Filesysteme für die Ablage von Projektdokumenten, sollen aber hier nicht weiter betrachtet werden.

Für die Kommunikation und Koordination in Verteilten Systemen sind folgende Techniken gebräuchlich:

- Nachrichten–basierte Systemkommunikation (Message Passing): als einfachste Form der Interaktion kann eine asynchrone Nachrichtenübertragung zwischen Clients und Server genutzt werden, eine typische derartige Technik ist die Nutzung der BSD-Socket-API. Lösungen, die auf derartigen Mechanismen beruhen, können eine hohe Performance aufweisen, jedoch ist aufgrund der sehr hardwarenahen Programmierung und des geringen Abstraktionsgrades ein Einsatz für die Realisierung komplexer verteilter Systeme aus softwaretechnologischen Gründen abzulehnen.
- Entfernte Prozeduraufrufe (Remote Procedure Calls): Entfernte Prozeduraufrufe sind eine Übertragung synchroner Prozeduraufrufe auf verteilte Systeme. Damit lassen sich bestehende, monolithische, nach dem prozeduralen Paradigma implementierte Applikationen vergleichsweise leicht nachträglich in verteilte Applikationen umwandeln. Für eine Nutzung als Basis der Neuimplementierung komplexer verteilter Systeme erscheinen RPC-Mechanismen aufgrund des zugrundeliegenden Paradigmas nur bedingt geeignet.
- Verteilte Objekte (Remote Method Calls, Distributed Objects Computing): Verteilte Objekte sind die Grundlage objektorientierter verteilter Systeme, deren Verbreitung vor allem durch die Notwendigkeit der Handhabung von Komplexität den Wunsch nach Wiederverwendung motiviert ist. Für Systeme, deren Module nach dem objektorientierten Paradigma realisiert werden, erscheinen derartige objektorientierte Mechanismen die adäquate Middleware, da hier der konzeptuelle Abstand am geringsten ist. Verteilte Objekte werden im nächsten Abschnitt näher betrachtet.
- Webbasierte Techniken: Die Nutzung des World Wide Web hat eine breite Verbreitung gefunden. Das Web ist ein verteiltes Hypermediasystem und als solches gut für die Informationspräsentation und eine implizite und asynchrone Kommunikation geeignet. Es sind Mehrschichtenarchitekturen umsetzbar, allerdings sind die Funktionalitäten der terminalen Clients auf jene von Thin–Clients oder gegebenenfalls Applet–Clients beschränkt, ferner existieren verglichen zu Verteilten Objekten eine Reihe anderer Restriktionen bei der Realisierung Verteilter Systeme.

Ein sehr häufig genutztes Modell für Verteilte Systeme ist das Client-Server-Modell. Unter einem Client versteht man eine Applikation, die auf einem Clientrechner ausgeführt wird und eine Anforderung an einen Service stellt. Man geht im allgemeinen davon aus, dass Clients a priori nicht bekannt sind. Ein Service ist ein Softwaremodul, welches eine bestimmte Dienstleistung erbringt und dabei auf einer oder mehreren Maschinen abläuft. Ein Server ist eine spezielle

Anwendung, die Services für Clients zur Verfügung stellt und dazu auf einer Servermaschine ausgeführt wird und dort eine Anzahl von Operationen anbietet. Das Client-Server-Modell geht davon aus, dass ein Client einen Service beim Server anfordert. Diese führt nach Empfang der entsprechenden Botschaft die dazugehörige Operation aus und gibt anschließend dem Client ein Resultat zurück. Einfach strukturierte Verteilte Systeme bestehen aus einem Server und einer Anzahl Clients. Dabei können je nach vorliegender Umsetzung eine variable Menge von Funktionalitäten den Clients oder dem Server zugeordnet werden. Die Extremfälle liegen hier bei einer Reduktion des Clients auf reine GUI–Präsentation (Null Client) bzw. der Beschränkung des Servers auf reines Datenmanagement (Fat Client); gebräuchlich ist auch die Zuweisung aller Interaktionsaufgaben an den Client (Thin Client). Derartig einfache Verteilte Systeme werden Zweischichtenmodelle (two tier models) genannt. Bei komplexeren Aufgaben tritt häufig der Fall ein, dass ein Server zur Erbringung seiner Dienste auf die Mitwirkung anderer Server angewiesen ist, der Server also auch gleichzeitig als Client fungiert. Entsprechend der Anzahl von Verteilungsschichten spricht man dann von n-tier model. In vielen Branchen weit verbreitet sind Dreischichtenmodelle (three tier models), die eine Unterteilung der Aufgaben in der betrieblichen Datenverarbeitung in das Datenmanagement, die Ausführung der Geschäfts-prozesslogik und die Anwenderinteraktion vornehmen.

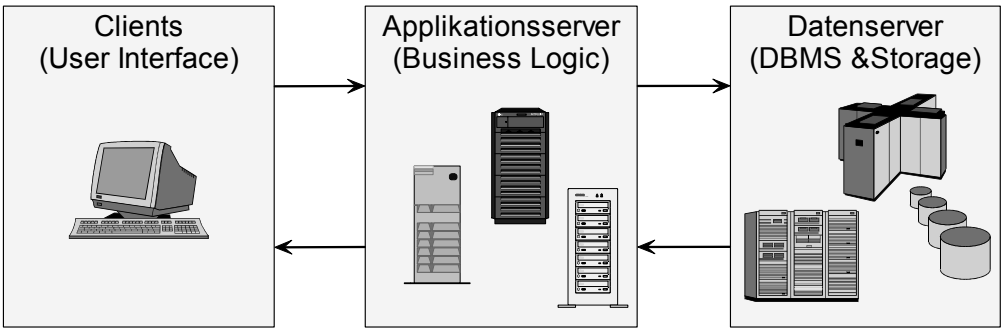

Abbildung 19: C/S-System mit Three Tier Model

Typische und weit verbreitete Umgebungen und Spezifikationen für die Realisierung konventioneller, RPC–basierter Verteilter Systeme nach dem Client–Server–Modell sind:

- Sun Open Network Computing (ONC): Open Network Computing ist eine Familie von Diensten zur Erstellung RPC–basierter Anwendungen, sie besteht im wesentlichen aus einem von der Netzwerktransportschicht unabhängigen Mechanismus für Remote Procedure Calls, einer Bibliothek für eine hardwareunabhängige externe Datendarstellung und den Network File System (NFS)– und Network Information Service (NIS)– Diensten. ONC ist eine vergleichsweise schlanke und hochperformante Umsetzung für RPC's. Es existieren eine Reihe von ONC–Umsetzungen für eine Vielzahl von Plattformen.

- Distributed Computing Environment (DCE): OSF DCE ist eine 1992 veröffentlichte Spezifikation für Verteilte Umgebungen der Open Software Foundation (heute: Open Group). Ein primäres Ziel der Entwicklung von DCE war die Unabhängigkeit von Hardware, Software und einzelnen Herstellern, entsprechend existieren Umsetzungen für nahezu jede verbreitete Plattform und für verschiedene Netzwerkprotokolle. Neben Spezifikationen für Remote Procedure Calls existieren Komponenten für leichtgewichtige Prozesse (Threads), Zeit–, Security– und Namensdienste und ein auf darauf aufsetzendes verteiltes Filesystem. DCE skaliert sehr gut, derartige Umgebungen können durchaus vier– bis fünfstellige Computerzahlen und sieben– bis achtstellige Anwenderanzahlen umfassen, leider gilt aber die Administration großer DCE–Systeme als äußerst komplexe und zeitaufwändige Aufgabe.
- Open Distributed Processing (ODP) ist eine durch die ISO genormte Spezifikation für verteilte Systeme, die ein Referenzmodell (ODP–RM) zur Verfügung stellt, das von einer Design–Modellierung aus den Blickwinkeln Unternehmen, Information, Verarbeitung, Engineering und Technologie ausgeht.

Die oben genannten Umgebungen für die Realisierung Verteilter Systeme beruhen auf dem Paradigma der strukturierten, imperativen Programmierung. Da der Realisierung Verteilter Systeme dieselben Vorteile der Nutzung des Objektorientierten Paradigmas wie in anderen Bereichen des Software Engineerings existieren, sollen im folgenden Objektorientierte Verteilte Systeme betrachtet werden.

2.5.2 Verteilte Objekte: Techniken und Implementierungen

Die Nutzung des Objektorientierten Paradigmas innerhalb der Technik der Verteilten Systeme ist, wie oben bereits erwähnt, durch den Wunsch nach Mechanismen zur Handhabung der bei Verteilten Systeme aufgrund der hohen Parallelität und Vielzahl von Teilkomponenten besonders hohen Komplexität und zur Wiederverwendbarkeit von Softwaremodulen motiviert.

Klassische Objekte, wie sie zum Beispiel in den Programmiersprachen Java, Smalltalk und C++ genutzt werden, bestehen aus gekapselten Daten und Implementierungen, auf die über eine Schnittstelle der Programmiersprache zugegriffen werden kann. Allerdings leben diese Objekte im allgemeinen nur zur Laufzeit eines Programms. Der Compiler erzeugt Code, der diese Objekte zur Laufzeit generiert und außerhalb der Applikation ist typischerweise nichts über die Existenz der konkreten Objekte bekannt.

Im Gegensatz dazu besitzen Verteilte Objekte (Distributed Objects) eine auch außerhalb des ausführbaren Moduls sichtbare Identität und ihre Dienste können auch von anderen Applikationen, eventuell sogar entfernt über Netzwerke in Anspruch genommen werden. Verteilte Objekte besitzen die Fähigkeit, folgende vier Grenzen von traditionellen Applikationen zu überwinden:

- Adressraum – Objekte der objektorientierten Programmiersprachen können nur mit Objekten im Adressraum ihrer Applikation interagieren,

- Maschine – compilierte Codesequenzen sind an Hardwareeigenschaften des Prozessors gebunden,
- Betriebssystem – herkömmliche Objekte können nicht über mehrere Betriebssysteme hinweg kommunizieren, diese Einschränkung ist besonders in heterogenen Umgebungen problematisch,
- Programmiersprache – Objekte, die in einer OOPL geschrieben sind, können im allgemeinen nicht mit Objekten aus einer anderen OOPL interagieren.

Innerhalb der Technologie der Verteilten Objekte sind verschiedene Ebenen zu unterscheiden. Als unterste Schicht agieren (Interprocess) Object Models, die als verteilbarer Objektbus beschrieben werden können. Diese Ebene löst vor allem das Problem der binären Kopplung der Applikationen mit den zu nutzenden Objekten und führt Fernaufrufe von Objekten aus. Um dies zu ermöglichen, haben die Objekte Schnittstellenbeschreibungen, die in einer geeigneten Interface Definition Language (IDL) notiert werden.

Als Ebene zwischen den Applikationen und den Object Models existieren die Component (Integration) Models. Diese Ebene macht die Einführung von komponentenbasierten Softwarelösungen, also Componentware möglich. Weiterhin sind Komponentenmodelle die Basis für einen dokumentenzentrierten Ansatz für Applikationen.

Verbreitete Ansätze für die Realisierung Verteilter Systeme auf Basis Verteilter Objekte (Distributed Object Computing – DOC) sind:

- Distributed Component Object Model (DCOM): Das Distributed Component Object Model ist eine proprietäre Protokoll-Spezifikation der Firma Microsoft, die eine relativ transportschichtunabhängige Kommunikation von Softwarekomponenten über Netzwerke hinweg gewährleisten soll. DCOM basiert partiell auf der DCE–Spezifikation der OSF und erlaubt durch die Nutzung von COM eine Interoperabilität mit ActiveX–Komponenten. Das Common Object Model (COM) ist ein Framework, das eine binäre Integration erlaubt und als Objektbus zwischen den Komponenten fungiert. COM stellt die technische Basis für die Spezifikation für Compound Documents OLE2 und ActiveX bereit.
- Java Remote Method Invocation (RMI): Java Remote Method Invocation ist eine spezielle Programmierschnittstelle (API) zur Erzeugung einfacher verteilter Java-Anwendungen, die in der homogenen Umgebung von Java Virtual Machines (JVM's) ablaufen. Vorteilhaft ist die leicht überschaubare und sich natürlich in die Sprache Java einfügende Umgebung, allerdings ist RMI kaum als Basis für die Realisierung komplexer verteilter Anwendungen geeignet. Gründe dafür sind die stark eingeschränkten Dienste der Middleware–Umgebung und die Beschränkung auf eine reine Funktionalität zum entfernten Methodenaufruf.
- Sun Enterprise Java Beans (EJB): Enterprise Java Beans sind eine Komponentenarchitektur für die Entwicklung und Nutzung von auf n-tier-Architekturen beruhenden, verteilten, skalierbaren, objektorientierten Java–Applikationen. EJB erleichtern die Implementierung

von Business-Komponenten durch die Bereitstellung automatischer Dienste für skalierbare Komponenten für Appplication–Server. EJB kann auf Grundlage existierender transaktionsverarbeitender Systeme wie beispielsweise TP–Monitore, Webserver, DBMS-Server, Application-Server eingesetzt werden. EJB ist die technische Basis der Java 2 Enterprise Edition (J2EE), welche die Java 2 Standard Edition (J2SE) um Funktionalitäten für die Realisierung vollständiger, stabiler, sicherer und performanter Systeme für unternehmensweite Informationsverarbeitung auf Basis der Java–Plattform erweitert.

- Common Object Request Broker Architecture (CORBA): CORBA ist eine durch die ISO genormte, hersteller- und plattformunabhängige Spezifikation für objektorientierte verteilte Systeme, die nachfolgend detaillierter behandelt werden soll.

Die **Common Object Request Broker Architecture (CORBA)** ist eine Spezifikation der 1989 gegründeten Object Management Group (OMG). Gründungsmitglieder waren acht namhafte IT-Firmen, darunter DEC, HP und Sun; mittlerweile besitzt die OMG über 700 Mitglieder aus Industrie, Forschung und staatlichen bzw. internationalen Organisationen.

Erstes Ziel der OMG war die Schaffung der Object Managment Architecture (OMA), die seither als abstrakte objektorientierte Referenzarchitektur und Grundlage von CORBA dient [OMG 95]. Neuere Versionen der OMA betrachten vor allem standisierte Interfaces für CORBA–Objekte und umfassen die CORBAservices, CORBAfacilities und CORBAdomain–Spezifikationen. Weitere OMG-Spezifikationen befassen sich mit Model Driven Architectures (MDA), der Meta Object Facility (MOF), dem Common Warehouse Metamodel (CWM) sowie der von Rational Software initial entworfenen und von der OMG weiterentwickelten und genormten Unified Modeling Language (UML).

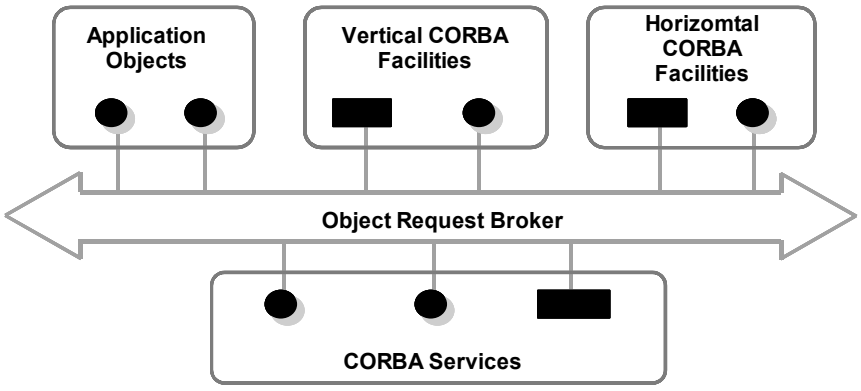

Abbildung 20: OMG Object Management Architecture

CORBA beruht auf einer vergleichsweise konsequenten Nutzung des objektorientierten Paradigmas. Die Schnittstelle der Serverobjekte wird zur Erzielung der Sprachunabhängigkeit durch eine Interface Definition Language beschrieben; dieselbe Schnittstellendefinition wird clientseitig zum Zugriff auf die verteilte Infrastruktur benutzt. Neben den Schnittstellen zum statischen CORBA–Methodenaufruf, deren Proxies und Stubs üblicherweise aus den IDL–

Informationen generiert werden, existiert auch eine Infrastruktur zur dynamischen Ausführung von Aufrufen (Dynamic Invocation Interface – DII). CORBA gestattet die Ausführung synchroner und (nichtblockierender) aynchroner Aufrufe. Für technische Details der CORBA–Infrastruktur sei an dieser Stelle auf die CORBA–Spezifikationen und die Sekundärliteratur verwiesen [OMG 01], [Pope98], [Siegel 97].

Die Version 1 der CORBA–Spezifikation definierte die allgemeine CORBA–Architektur. Durch Version 2 wurde die Interoperabilität zwischen ORB's verschiedener Hersteller gesichert. Dazu wurden Brige–API's sowie das obligatorische General Inter-ORB Protocol (GIOP) und die optionalen Environment Specific Inter ORB Protocol (ESIOP) beschrieben. Die 2002 verabschiedete Version 3 behandelt vor allem Internet Integration, Definitionen zur Sicherung von Dienstgüte (QoS – Quality of Service)– Eigenschaften und Spezifikationen für Komponenten-Frameworks.

In den vergangenen Jahren hat die Object Management Group verstärkt Anstrengungen zur Verabschiedung von branchenspezifischen Spezifikationen, sogenannten CORBAdomains, unternommen. Mit den CORBAdomains–Spezifikationen liegen gegenwärtig hauptsächlich Interface-Definitionen für das Finanzwesen, das Gesundheitswesen, die automatisierte Fertigung sowie für Telekommunikation und Luftverkehrskontrolle vor. Leider liegen abgesehen von recht allgemeinen Spezifikationen für CAD–Dienste in der Manufacturing–Norm keine auf den Bauwerksentwurf übertragbaren Festlegungen vor, ebenso sind auch keine Anstrengungen in Vorbereitung derartiger Normen wie beispielsweise Request for Proposals (RFP) bekannt.

Für die Realisierung von verteilten Systemen sind die in Vergleich zu anderen aktuellen Middleware–Lösungen sehr gut ausgebauten CORBAservices genannten Infrastruktur–Dienste sehr hilfreich. Derzeit sind folgende CORBAservices beschrieben [OMG 98] [OMG 00]:

- Naming service: Dienste zur Zuordnung von Objekten zu Namen in Kontexten,
- Lifecycle service: Dienste zum Erzeugen, Verschieben, Kopieren und Löschen von Objekten,
- Event service: asynchrone Kommunikation zwischen Objekten über Ereigniskanäle,
- Persistent State Service/Persistent Object service: Schnittstelle für Persistenzmechanismen,
- Relationship service: Dienste für Beziehungen zwischen CORBA–Objekten,
- Externalization service: Externalisierung und Internalisierung von Objekten,
- Concurrency service: Kontrolle des Zugriffs auf gemeinsam genutzte Ressourcen,
- Object Query service: Dienste für Anfragen an Objekt–Collections auf SQL– und OQL–Basis,
- Properties service: Dienste zur dynamischen Zuweisung von Eigenschaften an Objekte,
- Licensing service: Kontrolle der Nutzung durch lizensierte Clients,
- Time service: Bereistellung der aktuellen Zeit und einer Fehlerapproximation,

- Trader service: Dienst zur Suche passender Serverobjekte anhand bestimmter Anforderungen,
- Collections service: Verwaltung von Objektgruppen in verschiedenen Umsetzungen von ADT's,
- Security service: Funktionalitäten zur Authentifikation, Autorisierung und Auditing,
- Common Secure Interoperability: Dienst zur Nutzung von IIOP über sichere Verbindungen,
- Transaction service: Schnittstelle zur Unterstützung verschiedener Transaktionstypen,
- Notification service: asynchroner Kommunikationsmechanismus mit QoS–Eigenschaften.

Abschließend ist zu konstatieren, dass mit CORBA eine leistungsfähige und standardisierte Spezifikation für Infrastrukturen objektorientierter verteilter Systeme bereitsteht, die gut als Basis für die Umsetzung verteilter Bauwerksentwurfsumgebungen geeignet ist. Durch die Vielzahl der CORBA–konformen ORB's, unter denen sehr performante Umsetzungen existieren, kann ferner auch die Problematik des heterogenen Systemumfelds gelöst werden.

3 Bauwerksmodelle als gemeinsamer Informationsraum

3.1 Planung als Modellbildungs- und Nutzungsprozess

3.1.1 Ausgangsbasis

Im Ergebnis der einzelnen Phasen der Bauplanung entstehen in Abhängigkeit vom Typ und der Komplexität des in Planung befindlichen Bauwerks bestimmte Informationsbestände auf unterschiedlichsten Medien wie zum Beispiel Zeichnungen, Erläuterungstexte, Nachweise, Materiallisten, physische Architekturmodelle, etc. Diese Informationen sind eine abstrakte Darstellung des später zu realisierenden Bauwerks und stellen externe Repräsentationen der mentalen Modelle der Entwerfenden sowie Dokumentationen der durchgeführten Arbeitsschritte dar. Der Sinn dieser Informationen ist es, als Kommunikationsmittel zwischen den Beteiligten am Bauwerksentwurf genutzt zu werden, als Vorlage für die Bauausführung zu dienen, das Bauwerk über seine Lebensdauer zu dokumentieren und bestimmte Aussagen über Eigenschaften des Bauwerks treffen zu können, bevor dieses realisiert wurde. Diese Informationsbestände erfüllen die notwendigen Eigenschaften von Modellen, da sie ein reales oder künstliches Original in der Weise abbilden, dass nur relevante Aspekte erfasst werden und die dabei genutzte Abstraktion den subjektiven Erfordernissen des Modellierenden entspricht. Somit können die Entwurfs– und Konstruktionsprozesse der Bauwerksplanung als Modellierungsprozesse im weiteren Sinne aufgefasst werden. Unabhängig vom Repräsentationsmedium der Modelle treten dabei als grundlegende Modellformen beschreibende, analoge, symbolische, ikonische und physisch– nachbildende Modelle auf [Liebich 93] [Fischbach 94] [Kowalczyk 97]. So wird bei mathematisch basierten Nachweisverfahren auf ein symbolisches Modell des Bauwerks oder seiner Teile zurückgegriffen. Eine am Reißbrett entstandene Zeichnung oder die interne Repräsentation eines Bauwerks in einem traditionellen 2D– oder 2½D– CAD– System stellt dagegen ein ikonisches Modell des Entwurfsgegenstands dar.

Der Typ und der Detaillierungsgrad der beim Bauplanungsprozess genutzten Modelle haben immense Auswirkungen auf die Kommunikationsmöglichkeiten zwischen den Beteiligten am Bauplanungsprozess sowie auf Quantität und Qualität der im voraus über das Bauwerk zu treffenden Aussagen.

Im traditionellen, nicht rechnergestützten Bauwerksentwurf sind Zeichnungen, also ikonisch–analoge Modelle, das wichtigste Ausdrucks– und Kommunikationsmittel. Mit der Einführung von CAD–Systemen hat sich dies nicht sofort grundlegend geändert. Das Zeichnungskonzept von der traditionellen Arbeitsweise wurde übernommen und spielt bis heute bei einer Vielzahl von CAD–Systemen eine zentrale Rolle in der Systemphilosophie. Diese Systeme stellen effizientere elektronische Zeichenmaschinen dar, für die der Begriff CAD besser als *Computer Aided Drafting*

gedeutet werden sollte. Diese Systemphilosophie impliziert jedoch eine ganze Reihe von Beschränkungen. Diese rühren daher, dass das systeminterne Informationsmodell für eine Verarbeitung von meist zwei– oder zweieinhalbdimensionalen graphischen Primitiven wie Linien, Polygonen, Rechtecken, Quadern, etc. sowie Beschriftungen und bestimmten Symbolen ausgelegt ist. Die systeminterne Darstellung des in der Zeichnung als Projektion in den zweidimensionalen Raum abgebildeten, im Originalzustand räumlichen Entwurfsobjektes ist folglich eine Ansammlung derartiger Primitive und enthält keinerlei Informationen über die Semantik der Grafikprimitive im Kontext des durch die Zeichnung beschriebenen Entwurfsgegenstands. Der über die reine Anordnung graphischer Primitive hinausgehende intelligible Gehalt einer Zeichnung, also der gewünschte Endzustand des Entwurfsgegenstands und die auszuführenden Schritte zur Erzielung dessen kann jedoch nur durch über entsprechende Expertise[1] verfügende Fachleute vollständig interpretiert werden [Fischbach 94b]. Derzeit ist keine Technik bekannt, alle Aspekte dieser Expertise durch AI–Techniken so in rechnergestützte Systeme zu übertragen, dass eine vollständige und korrekte Interpretation von Bauzeichnungen möglich wäre[2]. Somit ist es bei diesen Systemen weder möglich, Funktionalitäten wie Analysen oder Auswertungen zu implementieren, die Wissen über die Semantik der Entwurfsobjekte enthält, noch kann ein Informationsaustausch auf dem semantischen Niveau der Entwurfsobjekte erzielt werden. Ein durchgängiger Informationsfluss unter Einschluss aller Phasen des Bauwerkslebenszyklus und aller Beteiligten in Entwurfs– und Revitalisierungsprozessen kann mit derartigen Systemen nicht erreicht werden.

Einige CAD– Systeme der nachfolgenden Generation erlauben die parametrisierte Erzeugung von bestimmten Entwurfselementen wie Wänden, Türen, Fenstern, etc. Oft ist es möglich, diese Elemente zu verschieben, zu löschen und Verschneidungen an Grenzbereichen mehrerer derartiger Elemente vorzunehmen. Diese ‚intelligenten Bausteine' beruhen jedoch in der Regel

[1] Expertise umfasst nach Zimbardo Wissen im Fachgebiet über effiziente Systeme von Regeln und Schemata, heuristische ‚Abkürzungen' für Suchprozesse, die Fähigkeit Top–Down– und Bottom–Up–Prozesse gleichzeitig auszuführen, die Speicherung einer beträchtlichen Menge von Fakten– und Handlungswissen, die Fähigkeit, allgemeines Wissen auf das Fachgebiet anzuwenden sowie metakognitives Wissen. Unter metakognitivem Wissen wird die Einschätzung des Grades der Verständnis des eigenen Fachgebiets im allgemeinen und der aktuell zu einem Objekt vorliegenden Informationen im speziellen verstanden [Zimbardo 92].

[2] Ein derartiges Vorhaben würde derzeit auf eine Vielzahl von Hindernissen stoßen. Einerseits ist es schwierig, Expertenwissen z.B. durch Interviewtechniken zu erfassen und formalisiert in Rechnersystemen zu speichern. Zimbardo schreibt dazu: "Sogar Experten hochtechnisierter Berufe können wahrscheinlich nicht erklären, wieso sie wissen, was sie wissen." Weiterhin müsste zur Auswertung dieser immensen Informationsmenge eine massive Parallelverarbeitung angewendet werden, um zu ähnlichen Antwortzeiten wie bei der Interpretation im menschlichen Gehirn zu kommen. Weiterhin wirkt erschwerend, dass einzelne Linien nur im Kontext der Gesamtzeichnung sicher interpretierbar sind und dass mit Inkonsistenzen wie mit nicht exakt abschließenden Linien, falschen bzw. uneinheitlichen Linientypen und unkorrekten Bemaßungen umgegangen werden muss.

aufgrund der Programmsystemhistorie intern auf einer Menge der oben beschriebenen Grafikprimitive. Das Hinzufügen neuer Elemente ist nur für den Systementwickler möglich oder praktikabel. Weitergehende Funktionalitäten lassen sich auch bei diesen Systemen nicht implementieren, ebenso bleibt die Problematik des Datenaustauschs zwischen den Entwerfenden bestehen [Kowalczyk 97].

Unter Produktmodellierung im Bauwesen wird die Entwicklung und Nutzung eines komplexen und möglichst konsistenten Modells des Entwurfsgegenstands verstanden, der mehrere Modellgrundtypen inkorporiert und diejenige Informationsmenge enthält, welche zur Beschreibung der strukturellen Prinzipien des Entwurfsgegenstands benötigt wird sowie die im Lebenszyklus eines Bauwerks für dessen Planung, Baudurchführung und Nutzung, Umnutzung und umweltgerechte Entsorgung notwendig ist. Dazu gehören die Geometrieinformationen, die funktionale Beschreibung des Bauwerks, Angaben über zu verwendende bzw. verwendete Materialien sowie technische und sonstige Angaben. Unter einem Produktmodell wird ein durch Abstraktion entstandenes, generisches Modellschema verstanden, das versucht, die oben genannten Informationen abzubilden und zur Beschreibung einer potentiell unendlichen Klassen von Entwurfsgegenständen geeignet ist [Kretzschmar 94] [Fischbach 94a]. Dient ein Produktmodell der Beschreibung von Bauwerken, wird es auch Bauwerksmodell[3] genannt.

Bauwerksmodelle sind seit mehreren Dekaden Gegenstand intensiver Forschungsarbeiten. Die Intentionen dieser Forschungsarbeiten und der Tätigkeit von Gremien mit Normungsabsichten waren einerseits auf die oben angerissenen wesentlich erweiterten Möglichkeiten von auf Bauwerksmodellen beruhenden Entwurfssystemen ausgerichtet. Andererseits zielten eine Vielzahl der Bemühungen auf die Lösung der Problematik des Produktsdatenaustauschs zwischen CAE–Systemen mit dem Ziel der Schaffung von durchgängigen Datenflüssen über alle oder einen großen Teil der Phasen des Bauwerkslebenszyklus ab. Mit neutralen Austauschmodellen sollen genormte, nicht proprietäre Formate zur Speicherung, dem Austausch, der Verarbeitung und zur Archivierung von Produktdaten geschaffen werden. Jedoch muss konstatiert werden, dass die derzeitige Relevanz von derartigen Bauwerksmodellen in der Baupraxis weit hinter den hohen Erwartungen zurückgeblieben ist. Einige Gründe dafür sollen in den folgenden Abschnitten erörtert werden.

[3] Die Termini Bauwerksmodell bzw. Produktmodell werden in der Literatur nicht einheitlich benutzt. Insbesondere werden je nach Autor ähnlich zur mehrdeutigen Anwendung des Begriffs ‚Datenmodell' verschiedene Modell– Abstraktionsstufen als Bauwerks–/Produktmodell bezeichnet. Da im folgenden mit dem Begriff ‚Bauwerksmodell' ein allgemeingültiges Schema zur Beschreibung von entsprechenden Objekten bezeichnet werden soll, wird die Beschreibung eines einzelnen konkreten Bauwerks ‚Bauwerksdaten' genannt.

Mit der Einführung von nichtproprietärenen Bauwerksmodellen werden folgende Erwartungen verbunden [Beucke 97]:

- Konsistenz der Bauwerksdaten in allen Phasen des Lebenszyklus
- Verringerung des Aufwandes für die nur einmalig zu erfolgende Datenerfassung
- Leichtere Änderungsmöglichkeiten der Planungsunterlagen
- Qualifiziertere und schnellere Entscheidungsfindung durch Ganzheitsbetrachtungen
- Direkte Verfügbarkeit der erforderlichen Bauwerksdaten.

Weitere Erwartungen betreffen die Eineindeutigkeit von Objekten im Entwurfsprozess. Diese sichert unter anderem nach bidirektionalen Übertragungen von Bauwerksinformationen die korrekte Interpretation von geänderten oder neu eingefügten Objekten.

Jedoch sind mit der Einführung neutraler Bauwerksmodelle auch Gefahren und Probleme verbunden. Einerseits besteht die Gefahr von Informationsverlusten durch Normierung der Daten. Dieses Problem tritt auf, wenn durch die Quell– und Zielapplikation interpretierbare Semantik nicht durch die Schnittstelle übertragen werden kann. Der Aufbau kompletter Bauwerksmodell–Datenstrukturen erfordert zusätzlichen Aufwand von einem Teil der Beteiligten am Bauwerksentwurf. Obwohl sich dieser Aufwand durchaus lohnt, wenn der Gesamtlebenszyklus des Bauwerks betrachtet wird, existieren Widerstände seitens der vom Mehraufwand betroffenen Beteiligten. Diese rühren daher, dass derzeit diese Tätigkeiten aufgrund der Festlegungen in VOB und HOAI nicht in hinreichendem Umfang honoriert werden. Für Softwarehersteller bedeutet die Unterstützung neutraler Modelle, dass deren Kunden Produkte für bestimmte Aktivitäten freier kombinieren können. Dies kann einerseits neue Kundenkreise erschließen, läuft aber der in Vergangenheit häufig zu beobachtenden Politik des „proprietary lock–in's" zuwider, bei der Ergänzungsprodukte nur mit denen desselben Herstellers harmonieren. Für Anwender als auch für Softwarehersteller sind konkurrierende Modelle aus Gründen des Investitionsschutzes problematisch, da die Umstellung bzw. Realisierung der entsprechenden Software mit hohen Kosten verbunden ist.

Als erste nationale Ansätze für Produktmodellierung im Bauwesen können das Standardleistungsbuch StLB und darauf aufbauende[4] sowie ähnliche, aber anders strukturierte[5] Entwicklungen angesehen werden. Das Standardleistungsbuch wurde Anfang der siebziger Jahre auf Initiative der Bauindustrie und des Bundesministeriums für Raumordnung, Bauwesen und Städtebau entwickelt und enthält standardisierte Ausschreibungstexte in einer nach

[4] Das Standardleistungsbuch ist Grundlage vieler AVA–Programme und wird auch in verschiedenen erweiterten Formen kommerziell vertrieben.

[5] Eine alternative bauteilorientierte Gliederung wurde 1978 von Piel unter der Bezeichnung SFB/CRB–Code entwickelt. Trotz semantisch reichhaltigerer Beschreibungsmöglichkeiten hat sich dieser Ansatz nicht durchgesetzt.

Baugewerken strukturierten Form. Ziel dieser Entwicklung war die Erleichterung von Ausschreibungen und eine bessere Zuordnungsmöglichkeiten von Bauwerksdaten.

Anfang der neunziger Jahre wurde versucht, einen eigenen finnischen Standard für den Produktdatenaustausch im Bauwesen zu etablieren. Das Modell RATAS beruht auf Konzepten von Objekten mit Attributen und Relationen zwischen Objekten. Bauwerke werden in einer mehrstufigen Abstraktionshierarchie von Gebäuden, Systemen, Teilsystemen, Teilen und Details beschrieben. RATAS geht von einem zentralen Modell aus, auf dessen Informationen mehrere Sichten (‚Viewpoints') erzeugt werden [Björk 89] [Björk 92]. Obwohl keine breite praktische Anwendung dieses Modells bekannt ist, wurde jedoch eine Vielzahl von Arbeiten in den neunziger Jahren von diesem Ansatz beeinflusst.

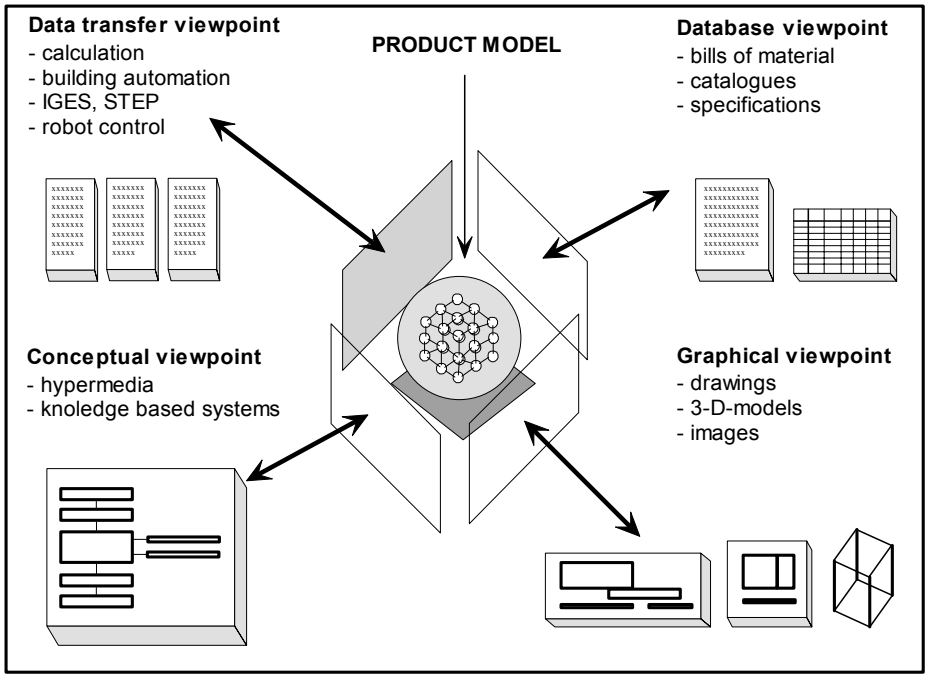

Abbildung 1: RATAS

Die RATAS– Philosophie [Hannus 96] ist in Folgeprojekten weiterentwickelt wurden. Die aktuelle RATAS–Vision beruht auf den folgenden vier Säulen:

- Öffentliche Datenbanken: projektunabhängige Informationen sollen von der A/E/C–Industrie bereitgestellt werden. Diese Datenbanken sollen Informationen über Produkte, Zulieferer, Kunden, Berufsverbände, Baunormen und Referenzlösungen enthalten.
- Datenaustausch–Standards: geeignete Datenaustausch–Standards für unterschiedliche Informationen werden entwickelt werden.
- Bauwerksprodukt–Modellierung: um den Informationsaustausch in Bauprojekten zu sichern werden die Bauwerksinformationen auf hohem semantischen Niveau als Produktmodell gespeichert. Die RATAS–Vision geht von einem Übergang von zeichnungsorientierten CAD–Systemen zu modell–orientierten Systemen aus.

- Anfrage–basierte Spezifikationen: Gegenwärtig dienen Entwurfsdokumente als vertragsgemäße Produkte des Bauwerksentwurfs und berücksichtigen nicht den Informationsbedarf unterschiedlicher Nutzer. Der RATAS–Ansatz zur Bauwerksmodellierung geht von einer Trennung von Informationserzeugung und –nutzung aus. Von einer Produktmodell–Datenbank werden durch DBMS–Queries spezifische Informationen extrahiert und dem Anwender in geeigneter Weise präsentiert.

Die RATAS–Vision umreißt die Idee einer voll integrierten Bauplanung mit durchgängigen Informationsflüssen. Als problematisch muss allerdings die Ausrichtung auf einzelne globale Modelle angesehen werden. Weiterhin ist die Nutzung filebasierter Austauschformate für die Unterstützung der Gruppenarbeit im Bauwerksentwurf sehr einschränkend.

Frühe neutrale Austauschformate waren nahezu ausschließlich für die Übertragung von Zeichnungs– oder Geometrieinformationen konzipiert und für vollständigen Produktdaten-austausch nicht anwendbar. Hier sind vor allem die Formate IGES und VDAIS als spezielles Subset für die Automobilindustrie zu nennen. Dasselbe trifft auch für die proprietäre Zeichnungsdaten–Schnittstelle DXF zu, welche die Bedeutung eines Quasistandards erlangt hat. Dieses Format besitzt zwar Möglichkeiten zur Übertragung nichtgeometrischer Informationen, jedoch sind diese sehr eingeschränkt und die Semantik der zu übertragenen Informationen ergibt sich aus der Einhaltung individueller Konventionen zwischen Sender und Empfänger. Effektive Möglichkeiten zur Übermittlung von Informationen über Gegenstände und Konzepte im Bauwerksentwurf besitzen diese genannten Formate nicht.

Ein durch die Internationale Standardisierungs–Organisation ISO genormtes neutrales Format zum Austausch von Produktmodelldaten ist STEP (Standard for the Exchange of Product Model Data). Obwohl ursprünglich für filebasierten Produktdatenaustausch zwischen CAD–Systemen konzipiert, umfasst die Funktionalität mittlerweile wesentlich mehr Features. STEP und die dazugehörige Spezifikationssprache EXPRESS werden im Abschnitt 3.4 näher betrachtet.

3.1.2 Techniken und Probleme der rechnergestützten Modellintegration

Eine notwendige Grundlage der Unterstützung von Teamarbeit im Bauwerksentwurf ist ein durchgängiger Informationsfluss zwischen den im Bauplanungsprozess genutzten Applikationen. Diese durchgängigen Informationsflüsse sind derzeit in der Baupraxis im allgemeinen nicht gegeben. Die Ursache dessen liegt darin, dass ursprünglich viele dieser Applikationen ausschließlich für die Unterstützung feingranularer Arbeitsaufgaben konzipiert wurden; die Idee eines integrierten Systems war wenig verbreitet [Fisher 97]. Daher ist in der Vergangenheit eine Vielzahl von Insellösungen entstanden, und bis heute muss eine Dominanz dieses

Applikationstyps konstatiert werden. Die Entwicklung der Applikationen im Bauwerksentwurf wird durch die ‚Islands of Automation'[6] von Matti Hannus [Hannus 98] illustriert.

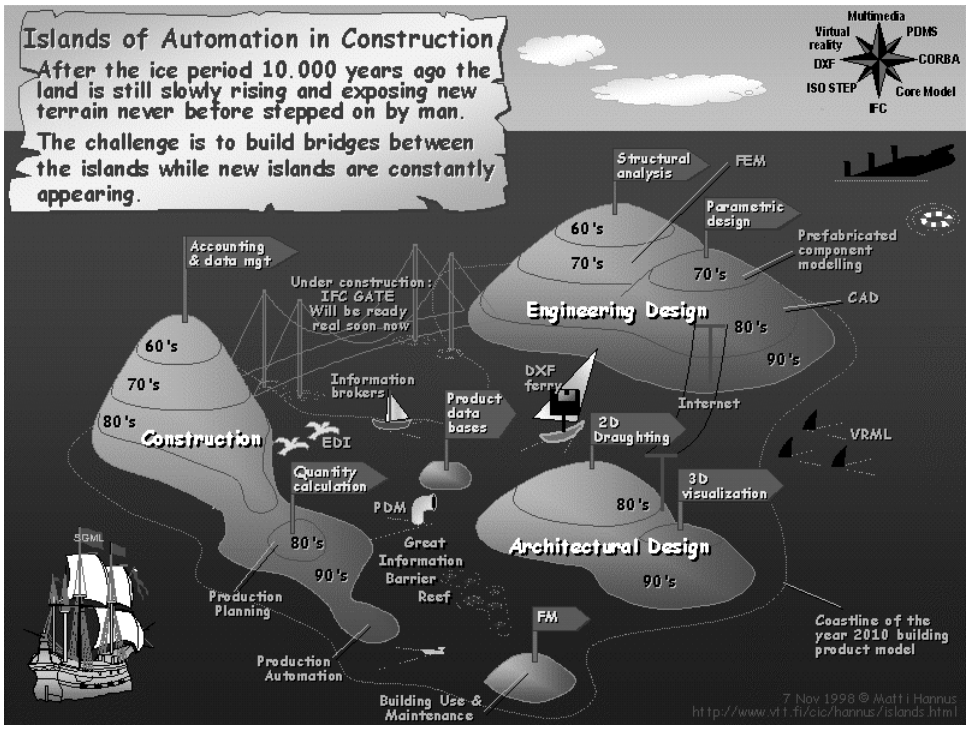

Abbildung 2: Islands of Automation nach Hannus [Hannus 98]

Es existieren verschiedene Ansätze für die Realisierung der Systemintegration der Applikationen im Bauwerksentwurf, ebenso muss zwischen drei verschiedenen Integrationsebenen für CAE–Werkzeuge unterschieden werden [Gausemeier 96]:

- Prozessebene: Gegenstand der Integration auf Prozessebene ist das Prozess– bzw. Workflow–Management. Es ist für die Ermittlung des aktuellen Stands des Entwurfs, für die Definition von Aufgaben und die Kontrolle von deren Durchführung verantwortlich. Ebenso ist der Einsatz der integrierten Werkzeuge zu steuern und zu planen.
- Modellebene: Integration auf Modellebene beschäftigt sich mit der Übermittlung von Informationen über den Entwurfsgegenstand zwischen den beteiligten Applikationen. Dabei ist eine explizite Kommunikation mit Datenaustausch zwischen Applikationen oder eine implizite Kommunikation über gemeinsame Modelle möglich. Die modelltechnische Integration verfolgt das Ziel, für Anwender einen transparenten Umgang mit verschiedenen Produktmodellrepräsentationen zu schaffen.

[6] Die Entwicklung dieser Graphik ist Ausdruck der Schwierigkeiten bei der Realisierung von durchgängigen Informationsflüssen im Bauwerksentwurf. In einer Version von 1996, die in [Fisher 97] veröffentlich wurde, wird die Existenz eines integrierenden Bauwerksproduktmodells für das Jahr 2000 angenommen.

- Systemebene: Die Bereitstellung eines geeigneten Kommunikationsmediums für den Informationsaustausch zwischen den Entwurfs–Applikationen ist Gegenstand der systemtechnischen Integration.

In einer Vielzahl von Publikationen wird unter Systemintegration ausschließlich die modelltechnische Integration verstanden, da auf dieser Ebene der Austausch von Informationen mit Entwurfssemantik zwischen den Applikationen stattfindet. Es kann zwischen folgenden Verfahren zur Integration auf der Modellebene grundsätzlich unterschieden werden:

- Sequentielle Verfahren: sind durch Applikationen mit separater Datenhaltung gekennzeichnet, zwischen denen ein Datenaustausch stattfindet. Sequentielle Verfahren können nach der Art der Schnittstelle charakterisiert werden:

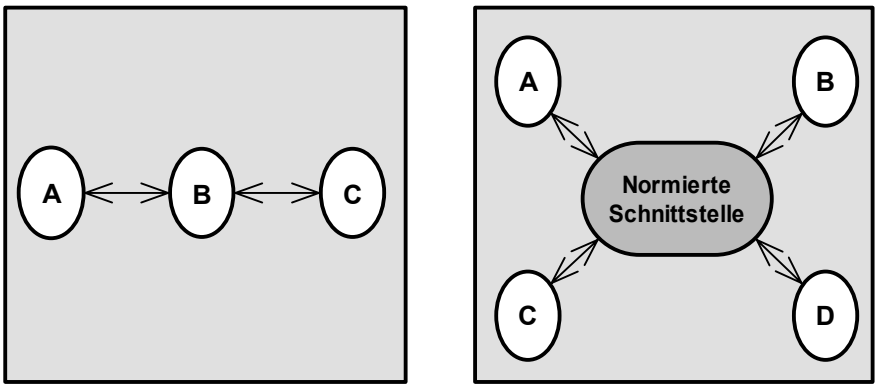

Abbildung 3: Sequentielle Integration mit (nicht-) normierten Schnittstellen

- Verknüpfung durch menschlichen Eingriff: gehört als nicht automatisierte Technik nur formell zu den Integrationsverfahren, da hier ein Bearbeiter Informationen aus einem System in ein anderes überträgt.
- Verknüpfung durch systemeigene Austauschformate: bei diesem Verfahren schreibt eine beteiligte Applikation Daten in einem spezifischen Format in eine Datei, die durch eine weitere Applikation interpretiert wird. Der Vorteil dieser Herangehensweise ist, dass auf diese Weise ein sehr hoher semantischer Gehalt des Informationsaustauschs realisiert werden kann. Für jede Kombination aus beteiligten Applikationen und jede Austauschrichtung muss jedoch ein Konverter–Modul geschaffen werden. Damit müssen für n Applikationen $n(n–1)$ Konverter realisiert werden; bei einer neu hinzukommenden Applikation A_{n+1} müssen $2n$ neue Konverter implementiert werden. Dieser erforderliche Implementierungsaufwand ist durch seine Komplexität kaum beherrschbar und wäre mit immensen Kosten verbunden, besonders wenn Applikationen beteiligt sind, die bei neuen Versionen neue Konverter[7] erfordern.

[7] Beispielsweise ist das DXF–Format eine vollständige externe Repräsentation der Zeichnungsdatenbank von AutoCAD. Es ist daher erklärte Strategie von AutoDesk, das DXF–Format sukzessive zu erweitern und an den Funktionsumfang der aktuellen AutoCAD–Version anzupassen.

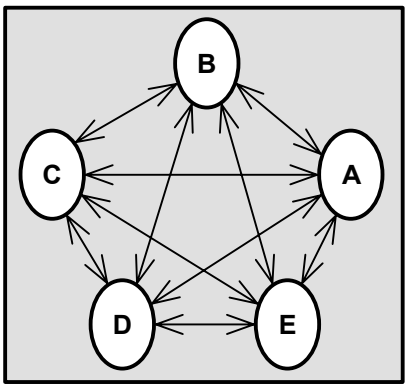

Abbildung 4: Verknüpfung durch systemeigene Austauschformate

- Verknüpfung durch standardisierte Austauschformate: Bei diesem Verfahren tauschen zwei oder mehr Applikationen Informationen in einem gemeinsamen, möglicherweise neutralen Format aus. Vorteilhaft ist dabei, dass pro Applikation nur ein Lese– und ein Ausgabemodul zu entwickeln ist, also insgesamt *2n* Prozessoren zu realisieren sind. Probleme dieser Lösung beruhen auf Informationsverlusten durch die ‚Normierung' der auszutauschenden Daten. Dies kann dazu führen, dass bestimmte Informationen nicht übertragbar sind, obwohl deren Semantik sowohl in der sendenden als auch in der empfangenden Applikation interpretierbar ist. Weitere Probleme standardisierter Austauschformate sind nach Duffy die Entwicklungsdauer des Austauschformats, deren Restriktionen und das schnelle Hinauswachsen der CAE–Systeme. Zwangsläufig können nicht alle Aspekte aller CAE–Systeme behandelt, ebenso kann die Laufzeit–Effizienz direkter Konverter im allgemeinen nicht erreicht werden [Duffy 95].

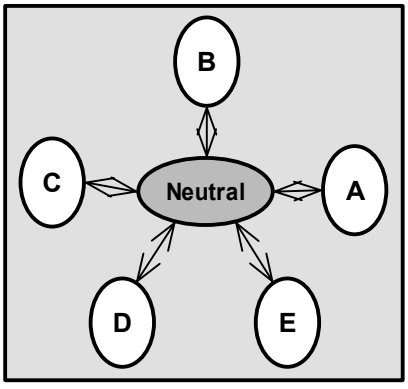

Abbildung 5: Verknüpfung durch neutrale Austauschformate

- Datenaustausch mittels Applikationsdomänenmodellen: Wenn die Existenz spezifischer Appliaktionsdomänen vorausgesetzt wird und diese einen definierbaren Scope besitzen, kann Standardisierung auf Applikationsdomänen zielen, die nicht notwendigerweise gemeinsame Definitionen aufweisen. Dies stellt die grundlegende Idee des Applikationsprotokoll–Ansatzes von STEP dar. Problematisch für diesen Ansatz ist jedoch der Fakt, dass im

Bauwesen Datenaustausch vor allem zwischen unterschiedlichen und schwer definierbaren Applikationsdomänen auftritt [Hannus 94].

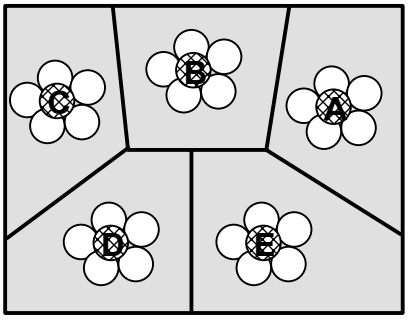

Abbildung 6: Verknüpfung über Applikations-Domänenmodelle

- Austausch über wechselseitig exklusive Partialmodelle: In bestimmten Fällen besitzen zwei Applikationen gemeinsame Daten, die nicht in weiteren Applikationen auftreten. Trotz einer engen Beziehung zwischen den beiden Applikationen erfordert die Verknüpfung tiefe Kenntnisse. Letztendlich kann nach Hannus die Menge der gemeinsamen Informationen als eigene Domäne angesehen werden [Hannus 94]. Es ist fragwürdig, ob irgendeine Form von Standardisierung für diese Art von Schnittstellen sinnvoll ist.

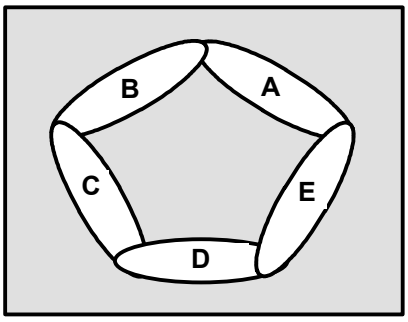

Abbildung 7: Verknüpfung über gegenseitig exklusive Partialmodelle

- Vertikale Verfahren der Systemintegration: sind durch eine gemeinsame Datenhaltung gekennzeichnet, auf die alle beteiligten Applikationen zugreifen.
  - Verknüpfung durch Speicherung in einer gemeinsamen Datenbank: ist dadurch gekennzeichnet, dass alle Applikationen ihre Daten in einer zentralen, gemeinsamen Datenbank ablegen. Vorteile dieses Ansatzes sind, dass alle Applikationen ständig über aktuelle Informationen verfügen können und dass keine aufwendigen Konverter implementiert werden müssen. Nachteilig ist, dass Maßnahmen zur Sicherung der Datenintegrität getroffen werden müssen, da diese durch den potentiell parallelen Zugriff nicht mehr grundsätzlich gegeben ist. Weiterhin ist die Komplexität des zu schaffenden zentralen Datenmodells sehr hoch, was sich als wesentliches Hindernis für diesen Ansatz erweist.

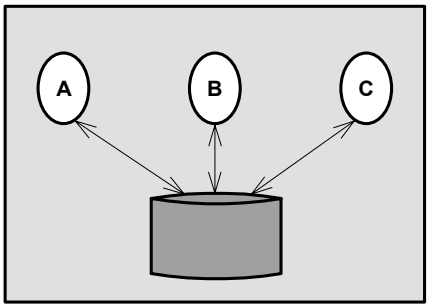

Abbildung 8: Vertikale Integration mit gemeinsamer Datenhaltung

- Verknüpfung über gemeinsamen Kern: Mehrere Applikationsmodelle besitzen unter Umständen gemeinsame Entities. Diese können dann als gemeinsamen Kern der Modelle genutzt werden. Häufig treten jedoch unterschiedliche Spezialisierungen dieser gemeinsamen Entities in unterschiedlichen Domänenmodellen auf. Diese applikationsspezifischen Erweiterungen außerhalb des Kerns gehen allerdings beim Datenaustausch verloren.

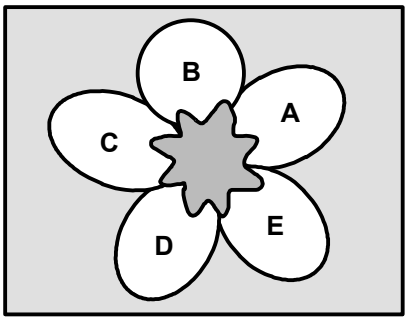

Abbildung 9: Verknüpfung über gemeinsamen Informationskern

- Datenaustausch über gemeinsame Ressourcen: Um Interoperabilität zwischen Applikations–Protokollen zu sichern, wurde bei STEP das Konzept der gemeinsamen Ressourcen eingeführt, die von unterschiedlichen Domänen geteilt werden. Über diese gemeinsamen Ressourcen ist zumindest ein Informationsaustausch auf Low–Level Niveau möglich, die applikationsspezifische Semantik geht im allgemeinen verloren.

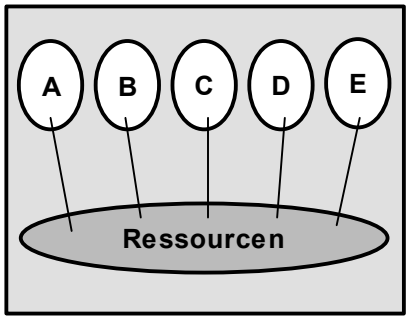

Abbildung 10: Verknüpfung über gemeinsame Ressourcen

- Horizontale Verfahren der Systemintegration: realisieren nicht nur den Zugriff auf eine gemeinsame Datenbasis, sondern gewährleisten auch den Austausch von Informationen auf konzeptueller Ebene.
    - Horizontale Integration mit Blackboards: Neben der gemeinsamen Datenbasis existiert ein Blackboard, dessen Informationen zur Konsistenzkontrolle genutzt werden. Dadurch können Entwurfsentscheidungen zu einem bestimmten Grad validiert werden.

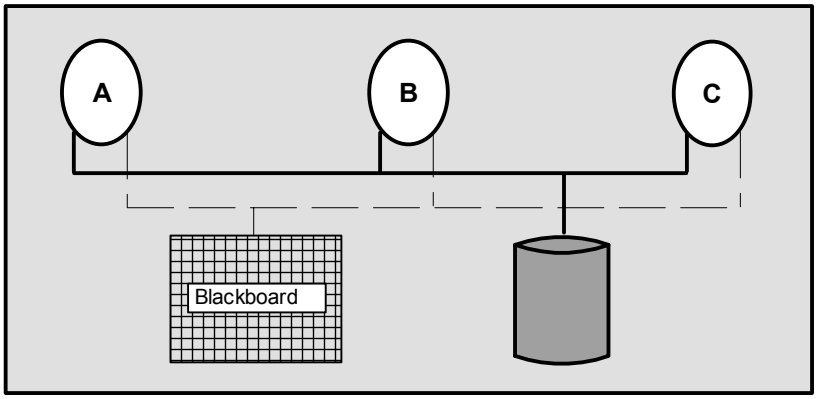

Abbildung 11: Horizontale Integation mit Austausch über Blackboards

- Integration über gemeinsamen Kern und Metamodell: Es ist wünschenswert, den Fokus der Standardisierung von spezifischen, applikationsabhängigen Definitionen zu fundamentalen Architekturen von Bauwerksmodellen mit beispielsweise generischen Datenstrukturen zu verschieben. Ein gemeinsames Kernmodell sollte definiert werden, der aus einer kleinen Menge von Entities besteht, die für einige Applikationen relevant sind. Ein integrierter Verknüpfungsmechanismus wird benötigt, um applikationsspezifische Informationen mit dem gemeinsamen Kern auszutauschen. Die Domänenmodelle und der Kern sollten auf einem gemeinsamen Metamodell mit gemeinsamen generischen Datenstrukturen basieren.

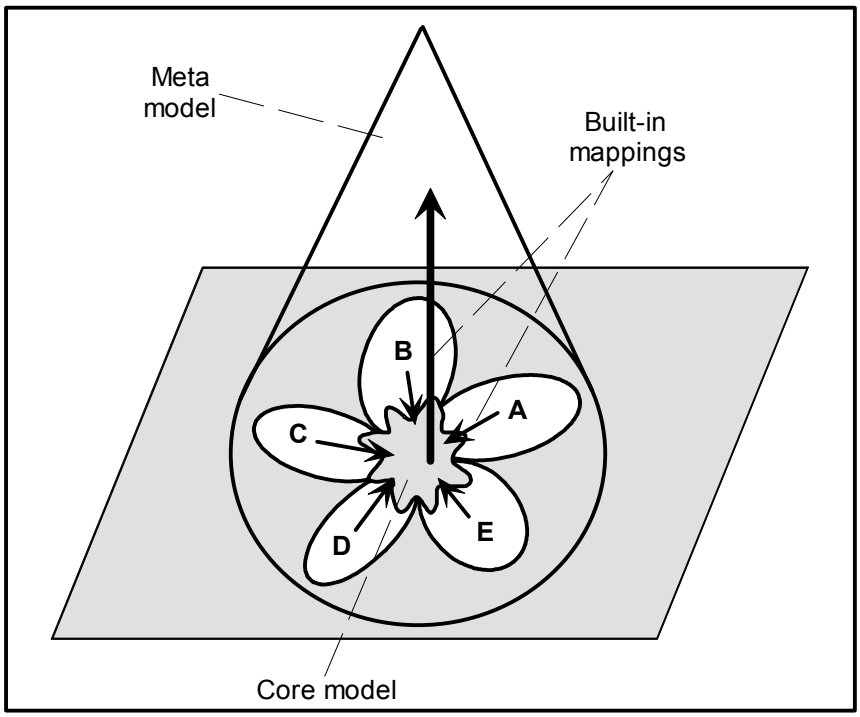

Abbildung 12: Integration über gemeinsamen Kern und Metamodell

Die herkömmliche Vorgehensweise zur Übermittlung der Bauwerksmodelldaten beruht auf einem filebasierten Informationsaustausch. Dies kann durch Übergabe eines Datenträgers geschehen, der mit der Hardware und dem Betriebssystem sowohl des sendenden als auch des empfangenden Systems kompatibel ist. Ebenso ist eine elektronische Übermittlung durch direkte temporäre Rechnerkopplung über Telekommunikationsnetze oder durch Nutzung von Internet–Basisdiensten wie FTP, E–Mail, etc. möglich. Jedoch ist bei sämtlichen filebasierten Verfahren zum Informationsaustausch die Zugriffsgranularität auf die Produktmodellinformationen sehr grob und eine Unterstützung einer echten Interaktion mit parallel laufenden Entwurfsprozessen ist kaum realisierbar [Gausemeier 96]. Generell kann ein filebasierter Informationsaustausch als zu restriktiv für die Realisierung von Applikationen der CSCW–Systemklasse ‚Workgroup Computing' angesehen werden [Borghoff 98]. Gründe dafür sind im schlechten Interaktionsverhalten einer filebasierten Kopplung zu suchen. Diese Problematik wird noch durch wenig befriedigende Möglichkeiten der Benachrichtigung zwischen den Applikationen und die im allgemeinen stattfindende Übertragung gesamter Modelle anstelle von Versionsdifferenzen vergrößert.

Einige Techniken der systemtechnischen Integration, die diese Beschränkungen nicht aufweisen, sind zum Beispiel:

- Funktionale Kopplung: Eine systemtechnische Integration ist durch Nutzung von Application Programming Interfaces (API's) möglich, wenn beide Applikationen synchron auf einer Maschine in einem gemeinsamen Speicherbereich laufen. Diese direkte Kopplung besitzt ein gutes Interaktionsverhalten, ist aber nur realisierbar, wenn offene Systeme vorliegen, die für

die Kopplung notwendige Kompatibilitätseigenschaften aufweisen. Zur Unterstützung von Teamarbeit ist diese Technik nur sehr bedingt geeignet, da dann die Applikationen in einer multi–threading fähigen Multi–User Umgebung laufen müssten.

- Nutzung von OS–IPC–Features: Applikationen können auf einer Maschine über Mittel der Inter Process Communication (IPC) des Betriebssystems wie Shared Memory, Pipes, Semaphore, Message Queues, usw. kommunizieren. Problematisch ist an diesem Ansatz, dass die konkrete Umsetzung der IPC–Features betriebssystemabhängig ist und bestehende Normen wie ISO POSIX nur auf bestimmten Plattformen konsequent umgesetzt wurden.
- Socket–Programmierung: Applikationen können durch direkte Nutzung der TCP–/UDP–Sockets über Rechnergrenzen hinweg kommunizieren. Der hohen Kommunikations–Performance stehen ein hoher Programmieraufwand, fehlende Transparenzeigenschaften, die Nichtverfügbarkeit höherwertiger Dienste und eine geringe Portabilität der Lösung gegenüber. Das Programmier–Paradigma der Socket–Nutzung kann als analog zur Assembler–Programmierung betrachtet werden.
- Remote Procedure Call's (RPC's): Die klassische Kommunikationsform des Client–Server–Computings sind entfernte Prozeduraufrufe. Die Anwendung von Remote Procedure Calls entspricht der Überführung des Paradigmas der imperativen Programmierung in die Umgebung der netzwerkbasierten Systeme. Die verbreitetste Umgebung für RPC–basierte Verarbeitung nach dem Client Server Modell ist das Distributed Computing Environment (DCE), das im Kapitel 2 vorgestellt wurde.
- Verteilte Objekte: Verteilte Objekte gewährleisten eine netzwerkweite Interaktion von Objekten. Die Hauptgründe für die Nutzung der Objektorientierung in Verteilten Systemen liegen in der besseren Beherrschbarkeit der Komplexität der Systeme, dem Wunsch nach wiederverwendbaren Komponenten und günstigeren Transparenzeigenschaften dieses Ansatzes. Das Konzept Verteilter Objekte und Systemumgebungen für Distributed Object Computing (DOC) wurden ebenfalls im Kapitel 2 eingehend besprochen.

Für Systeme zur Unterstützung von Teamarbeit im Bauwerksentwurf sind besondere Anforderungen an die zugrundeliegenden Integrationstechniken zu stellen. Dabei spielt die Zugriffsgranularität eine wesentliche Rolle. Darunter wird die Größe des für einen Produktentwickler für Modifikationen reservierten Teil des Bauwerksmodells verstanden. Dies kann im Minimalfall ein einzelnes atomares Attribut eines Objekts sein, im Maximalfall ein gesamtes Domänenmodell. Je kleiner der reservierte Teil eines Domänenmodells ist und je flexibler der Sperrmechanismus ist, desto günstiger sind die Voraussetzungen für die Unterstützung synchroner Kooperation. An die systemtechnische Integration sind Anforderungen wie hohe Performance, netzwerkbasiertes Arbeiten, hohe Flexibilität und Kompatibilität zu den Domänenapplikationen zu stellen.

Mit der zunehmenden Komplexität eines Produktes wächst auch die zu dessen Beschreibung notwendige Informationsmenge. Zur Organisation dieser Informationen können Produktdaten–Management (PDM)–Systeme eingesetzt werden. Diese Systeme stellen dem Entwerfenden die benötigten Informationen zur Verfügung und verwalten dessen Arbeitsergebnisse, allerdings im allgemeinen nur in einer filebasierten Form. Damit ist die Zugriffsgranularität allerdings sehr grob. Dadurch sind PDM–Systeme nur bedingt zur Unterstützung von Teamarbeit geeignet.

3.2 Planungsadäquate Bauwerksmodelle

3.2.1 Objektorientierte Bauwerksmodelle

Im letzten Jahrzehnt hat die Modellierung unter Nutzung des objektorientierten Paradigmas[8] stark an Bedeutung gewonnen [Alhir 98]. Gegenwärtig stellt diese Modellierungsphilosophie die hauptgenutzte Form der Modellbildung bei neuen Projekten in der Bauinformatik dar [Olbrich 98]. Dies ist wenig überraschend, da die Anwendung dieses Paradigmas den gegenwärtigen Stand der Kunst in der Softwareentwicklung repräsentiert [Booch 99].

Objektorientierte Modelle weisen die allgemeinen Eigenschaften von Modellen auf, sind also Vereinfachungen der Realität und werden erzeugt, um das System, das entwickelt werden soll, besser zu verstehen. Komplexe Systeme können nicht vollständig ohne abstrahierende Modelle verstanden werden. Booch nennt vier grundlegende Prinzipien der Modellierung [Booch 99], die besonders für die Nutzung der Objekttechnologie gelten:

- Einerseits hat die Entscheidung, welche Modelle zu erzeugen sind, weitreichende Auswirkungen auf den Lösungsweg und die Strukturierung der Lösung.
- Alle Modelle können auf unterschiedlichen Detaillierungsgraden ausgedrückt werden.
- Die besten Modelle sind mit der Realität verknüpft.
- Kein einzelnes Modell ist hinreichend. Jedes nichttriviale System kann am besten durch eine kleine Menge nahezu unabhängiger Modelle beschrieben werden.

[8] Paradigmen beschreiben die Art und Weise, mit der kognitiv fassbare Modellrepräsentationen der (undurchdringlich komplexen) realen Welt erzeugt und genutzt werden. Unter einem Paradigma wird daher eine organisierte Menge von in Beziehung stehenden Komponenten verstanden, welche die Basis der Modellierung bilden. Paradigmen besitzen durch eine Menge von Termini ein spezifisches Vokabular und definieren durch eine Menge von Konzepten Ideen und Begrifflichkeiten, die mit den Subjekten verknüpft sind. Paradigmen bilden die Basis für Metamodelle, Modelliersprachen und Modelliermethoden durch die Definition des in der Modellierung nutzbaren Vorrats an Elementen, Termini und Konzepten. Ein Teil der in Paradigmen eingeführten Konzepte und damit verbundenen Termini stellen fundamentale, essentielle und universell akzeptierte Wahrheiten und Prinzipien innerhalb des Paradigmas dar. Auf dieser Grundlage setzen die anderen Konzepte des Paradigmas auf und werden Systeme modelliert, analysiert, dargestellt, interpretiert und manipuliert.

Das Paradigma der Objektorientierung basiert im wesentlichen auf drei fundamentalen Säulen[9]: Kapselung (Encapsulation), Erbschaft (Inheritance) und Polymorphismus (Polymorphism). Unter Kapselung[10] wird das Verbergen der Interna von Objekten wie Datenstrukturen und Implementierungsdetails gemeinsam mit der Bereitstellung einer öffentlichen Schnittstelle verstanden. Die im Objekt enthaltene Informationsmenge stellt die Repräsentation des Objekts im zu modellierenden System dar, dabei werden algorithmische und attributive Elemente gemeinsam erfasst. Die Auswahl der Elemente der Klassen und ihrer Exemplare stellt eine Abstraktion der zu betrachtenden Entität innerhalb der Modellwelt dar, da hier aus dieser Perspektive die essentiellen Eigenschaften und Operationen erfasst, während unwesentliche Details ignoriert werden. Aus diesem Grund werden in einem Teil der Literatur die Konzepte ‚Abstraktion' und ‚Kapselung' in Bezug auf genanntes Paradigma als zusammenhängend betrachtet. Bei wohlentworfenen Klassen und deren Exemplaren können Statusänderungen bzw. Abfragen des Objektstatus nur durch das Senden von in der Schnittstelle aufgeführten Botschaften ausgeführt werden. Daher können durch diese Mittel des objektorientierten Paradigmas Objekte dem Prinzip der Abstrakten Datentypen (ADT's) genügen.

Polymorphismus[11] (griechisch für ‚Vielgestaltigkeit') bezeichnet die Möglichkeit, die selbe Bezeichnung für unterschiedliche Operationen zu verwenden. Aus Objektsicht betrachtet bedeutet Polymorphismus, dass Exemplare je nach Typ individuell auf den Empfang derselben Botschaft reagieren können, da die zum jeweiligen Objekttyp gehörende Implementation der Operation ausgeführt wird.

Der Terminus ‚Erbschaft' bezieht sich auf zwei grundlegende Mechanismen: die Übertragung der Attribute und Operationen von Klassen an deren Exemplare sowie von Superklassen an abgeleitete Klassen. Der erste Mechanismus, die Zuordnung von Objekten zu ihrem Typ, wird

---

[9] In dieser Arbeit soll das Paradigma der Objektorientierung nur kurz und überblicksartig behandelt werden. Für weitergehende Informationen soll auf die zahlreiche ausführliche Literatur zum Thema wie u.a. [Graham 94], [Booch 94], [Rumbaugh 91], [Wirfs–Brock 93], [Alhir 98] usw. verwiesen werden. Die mathematischen Grundlagen des objektorientierten Modellierens im Bauwerksentwurf werden unter anderem in [Olbrich 98] dargestellt.

[10] Ein äquivalenter gebrauchter Terminus für Kapselung ist ‚Information Hiding'.

[11] Cardelli und Wegner beschreiben verschiedene Arten von Polymorphismus: Grundsätzlich kann in universellen und nicht–universellen oder ad–hoc Polymorphismus unterschieden werden. Universeller Polymorphismus ist potentiell auf eine unendliche Anzahl von Typen (mit einer gegebenen gemeinsamen Struktur) anwendbar, währenddessen ad–hoc Polymorphismus nur für eine endliche Anzahl von Typen (welche keine gemeinsame Struktur aufweisen müssen) anwendbar ist und die Nutzung desselben Symbols für potentiell semantisch verschiedene Operationen beschreibt. Universeller Polymorphismus kann in parametrischen Polymorphismus (eine Funktion kann einheitlich mit verschiedenen Typen arbeiten) und Inclusion–Polymorphismus (modelliert Subtypen und Erbschaft) unterschieden werden. Zum ad–hoc Polymorphismus gehören Overloading (Function & Operator Overloading) und Coercion (implizite Typumwandlungen von Parametern und in Ausdrücken, z.B. zwischen numerischen Datentypen) [Cardelli 85].

durch die Klassifikations–Relation *is–a* modelliert, wobei Exemplare demselben Typ angehören, wenn sie dieselbe Schnittstelle aufweisen. Erbschaftsbeziehungen zwischen Klassen werden durch die Generalisierungs–/Spezialisierungs–Relation *a–kind–of* modelliert. Bei Erbschafts–Relationen zwischen Klassen kann zwischen Schnittstellen– und Implementationsvererbung unterschieden werden. Während ein Großteil der aktuellen objektorientierten Programmiersprachen Implementationsvererbung unterstützt, bringt eine Beschränkung auf reine Schnittstellenerbschaft Vorteile in Bezug auf die Erfüllung des open–close–Prinzips [Meyer 88] und einer guten Umsetzbarkeit des Liskovschen Substitutionsprinzips [Martin 96] [Liskov 87].

Bei der objektorientierten Modellierung treten im allgemeinen zwei grundlegende Typen von gerichteten zyklenfreien Graphstrukturen[12] auf, einerseits Klassifikation und Erbschaft in Form von Hierarchien oder Heterarchien und andererseits die (strukturellen) Kompositions– und Aggregationshierarchien. Letztere beschreiben die Gesamtheit–Teil–Struktur (*part–of*) von zu modellierenden Objekten. Relationen zwischen Klassen bzw. Objekten, die nicht auf Generalisierung/Spezialisierung/Klassifikation oder Aggregation/Komposition zurückzuführen sind, werden als Assoziationen modelliert. Assoziationen können daher zur Repräsentation von Beziehungen mit unterschiedlicher Semantik genutzt werden.

Das Paradigma der Objektorientierung hat sich im letzten Jahrzehnt als geeignete Grundlage für Modellierung und Softwareentwicklung erwiesen. Jedoch löst dieses Paradigma nicht sämtliche bestehende Probleme, beispielsweise bleibt die umfassende Wiederverwendung von Softwarekomponenten hinter den Erwartungen der frühen Arbeiten auf diesem Gebiet wie beispielsweise [Meyer 88] zurück; weiterhin kann das Laufzeitverhalten negativ beeinflusst werden und die Anfangskosten können über denen konventioneller Projekte liegen. Bedeutende Vorteile der Objektorientierung sind die im Vergleich zu anderen Paradigmen einfachere Abbildbarkeit von Entitäten der realen Welt als Modellobjekte und die besseren Möglichkeiten, komplexe Sachverhalte darzustellen. Durch die Abstraktion der realen Welt wird es möglich, größere Ausschnitte bzw. komplexere Sachverhalte kognitiv zu erfassen und zu modellieren. Dies ist besonders wichtig, da Untersuchungen der kognitiven Psychologie auf dem Gebiet der visuellen Wahrnehmung und der Verarbeitung im Kurzzeitgedächtnis zeigen, dass die Anzahl gleichzeitig im Gehirn verarbeitbarer Informationseinheiten beschränkt[13] ist. Die Behandlung von

[12] Neben diesen beiden Strukturtypen sind auch andere denkbar [Graham 94], jedoch werden diese in den gängigen objektorientierten Methoden aufgrund ihrer grundlegenden Bedeutung der *is–a* bzw. *a–kind–of* und *part–of* Relationen direkt durch Elemente des Modellierungsvokabulars unterstützt.

[13] Der schottische Philosoph Sir William Hamilton (1788–1856) sowie William Stanley Jevons (1835–1882) beschäftigten sich mit den Aufnahmegrenzen visueller Wahrnehmung. Der amerikanische Psychologe George Miller führte Untersuchungen zu Chunking (Rekodierung einzelner Items durch Gruppierung oder durch Kombination in größere Muster) im Kurzzeitgedächtnis durch. Allen Untersuchungen ist gemeinsam, dass als Grenze der gleichzeitigen visuellen Wahrnehmung bzw. der Kapazität des Kurzzeitgedächtnisses die ‚magische Zahl sieben plus/minus zwei' nicht verbundener Informationseinheiten gefunden wurde [Zimbardo 92].

Komplexität durch rekursives Aufteilen eines Problems in Teilprobleme (*divide–et–impera*) wird in hilfreicher Weise durch Aufteilung in zur Realität korrespondierende Komponenten befördert, während das anderen Paradigmen inhärente Splitting von Informationen und Funktionalität eher geeignet ist, die Komplexität des Sachverhalts noch weiter zu erhöhen. Weiterhin gewährleistet das objektorientierte Paradigma einen geringen konzeptuellen Abstand zwischen realer und Modellwelt sowie eine Methodik der Modellierung, die den Prozessen menschlicher Wahrnehmung recht nahe kommt. Weitere Vorteile der Objekttechnologie sind Kostenreduktionen durch Design– und Codewiederverwendung sowie eine höhere Entwicklungsproduktivität. Durch die Modularität objektorientierter Entwürfe können Softwarekomponenten umfassend getestet werden, was zu einer höheren Qualität der Produkte führen kann. Ferner sind derartige Systeme besser zu warten, flexibler, unempfindlicher gegen Änderungen, besitzen geringere Entwicklungsrisiken und sind besser skalierbar [Booch 94] [Graham 94].

Für objektorientierte Bauwerksmodelle gelten im Vergleich zu konventionellen Modellen die genannten Vorteile des objektorientierten Paradigmas. Bauwerksmodelle besitzen naturgemäß eine hohe Komplexität, die sich mit Hilfe der objektorientierten Techniken besser handhaben lässt. Die potentielle Komplexität dieser Modelle wird beispielsweise durch die aktuelle Version der Industry Foundation Classes (IFC) verdeutlicht, welche eine vierstellige Anzahl von Klassen beinhaltet.

Ebenso wesentlich ist der vergleichsweise geringe konzeptuelle Abstand zwischen Modellwelt und Realität bei Anwendung des objektorientierten Paradigmas sowie dessen Analogie zu linguistischen Modellen. Da das rechnerinterne Modell eine externalisierte Darstellung des mentalen Modells des Entwerfenden ist, welche wiederum eine abstrahierte Vorwegnahme des gewünschten Endzustandes des Bauprozesses darstellt, kommt einer möglichst einfachen Überführbarkeit eine große Bedeutung zu. Folglich können objektorientierte Bauwerksmodelle eine geeignete Basis für die Unterstützung verschiedener Entwurfsaufgaben der Klasse des Innovativ-Designs bilden. Dies gilt unter anderem für frühe Phasen des architektonischen Entwurfs [Liebich 93] [Steinmann 97a]. Um diese Entwurfsphasen adäquat unterstützen zu können, sollten Entwurfsaktivitäten sowohl in Bottom–Up– als auch in Top–Down– Richtung möglich sein. Letzteres ist erforderlich, um besonders im architektonischen Vorentwurf das Bauwerksmodell korrespondierend zum mentalen Modell des Entwerfenden wachsen lassen zu können. Bei einer Betrachtung des Designprozesses im Sinne linguistischer Modelle werden Entwurfsgegenstände anfänglich unter Berücksichtigung ihrer Funktion im Bauwerk nur als Exemplar ihres Typs verstanden. Im weiteren Entwurfsprozess wird sukzessive durch Problemdekomposition und Lösungssynthese die konkrete Idee des Entwurfsgegenstands mit verfeinerten Attribut–Informationen und möglicherweise der Spezialisierung des Typs entwickelt. Zeitweise treten im Entwurfsprozess auch Bottom–Up–Phasen auf, beispielsweise wenn diese leicht lösbar oder vorgegebene Detailinformationen mit den eigenen Entwurfsintensionen in Einklang zu bringen sind. Weiterhin wäre der Entwerfende bei strengem Top–Down–Vorgehen gezwungen, auch triviale oder unbewusste Entwurfsobjekte zu

externalisieren. Die Methodologie der objektorientierten Modellierung in der Informationstechnologie steht in Analogie zu derartigen kognitiven Entwurfsmodellen.

Bei Entwurfsprozessen treten häufige Wechsel des Abstraktionsgrades des aktuell betrachteten Objekts auf. Diese Wechsel sind bei Applikationen, die auf objektorientierten Modellen beruhen, gut unterstützbar, indem eine andere Tiefe des Taxonomie– bzw. Aggregationsbaumes gewählt wird. Ein weiterer Vorteil von Objektmodellen in der Bauwerksplanung ist die mögliche Durchgängigkeit der Modelle über den gesamten Bauwerkslebenszyklus von ersten Entwurfsschritten bis hin zu archivierten Dokumentationen nach Abbruch des Gebäudes. Verschiedene Anforderungen an Inhalte und Präsentation von Modellen lassen sich durch spezifische Modellsichten oder -subsysteme bzw. dedizierte Attribute lösen. Neben den reinen Objekt– und Attributinformationen sind Relationen zwischen Objekten untereinander und mit Modellklassen für verschiedene Phasen sehr wichtige Informationen, die aber bei anderen Paradigmen nur schwer bzw. aufwendiger abbildbar sind. Dieses Wissen kann gut je nach Art in Klassifizierungs–, Erbschafts–, Aggregations– oder Assoziationsstrukturen abgebildet werden und bleibt über die gesamte Modelllebensdauer erhalten. Ferner dürfte die Kombination des zum Entwurfszeitpunkt vorhandenen und in Klassen abgebildeten Wissen mit den Informationen über das Bauwerk als konkrete Ausprägung über sehr lange Zeiträume interpretierbar bleiben [Steinmann 97a].

In [Fisher 97] werden Erwartungen an die durchgängige Nutzung von objektorientierten Modellen in der britischen Bauwirtschaft benannt und begründet. Diese dürften zumindest auf die meisten Staaten der Europäischen Union übertragbar sein. In der Makro–Sicht, also auf die gesamte Bauwirtschaft bezogen, sorgen genannte Modellierungstechniken für:

- besseres Preis–Leistungsverhältnis durch umfassenderen Entwurf aufgrund der Nutzbarmachung von Entwurfswissen und –erfahrung
- für jeweilige Entwurfsbelange optimale Modellpräsentation in allen Phasen
- besseres Informationsmanagement und bessere Entwurfskoordination
- Einsatz von mehr standardisierten Teilen und insgesamt weniger Komponenten
- schnellerer Bauprozess durch bessere Bemessung, bessere Konstruktion von Verbindungselementen und bessere Montageplanung
- verbesserte Einbindung von Herstellern von Komponenten durch Informationskoordination und –austausch, gemeinsame digitale Probemontagen und Planung sowie Training der Arbeitskräfte

In der Mikro–Sicht, also auf den einzelnen Entwerfenden bezogen, benennen Fisher und Koautoren folgende Vorteile der Nutzung des objektorientierten Paradigmas:

- Unterstützung von Kreativität durch bessere Chancen zur Exploration von Entwurfsalternativen aufgrund drastisch verringertem Zeitaufwand für Änderungen

- bessere Abschätzbarkeit der Auswirkungen von Entwurfsänderungen auf die Gesamtbaukosten durch Kostenmodelle
- (teil–)automatische Generierung von Virtual Reality–Visualisierungen des Bauwerks
- verbesserte Kommunikation mit anderen Entwerfenden und Bauherren
- Möglichkeiten zum Test der Funktionsfähigkeit des Bauwerks durch Simulation
- bessere Ressourcenplanung im Bauablauf sowie erhöhte Genauigkeit der Material– und Arbeitskräftebedarfsplanung
- Möglichkeiten zur Suche von negativen Wechselwirkungen bei der Planung von gravierenden strukturellen Eingriffen in einen existierenden Baubestand; derartige Probleme werden oft erst im Bauablauf erkannt und müssen dann kostenintensiv ausgeräumt werden

Zusammenfassend kann konstatiert werden, dass eine durchgängige Nutzung des objektorientierten Paradigmas in der Bauwerksmodellierung vielfältige Potentiale für den Entwurf und die Bauausführung als auch für volkswirtschaftlichen Nutzen enthält.

3.2.2 Dynamische Bauwerksmodelle

Die Majorität der Bemühungen auf dem Gebiet der Bauwerksmodellierung basiert auf statischen Modellen. Bei diesen konventionellen Ansätzen wird zum Zeitpunkt der Systementwicklung ein Bauwerksmodell definiert und ein Kompilat dessen fest mit der Entwurfsapplikation verbunden, so dass Erweiterungen oder Modifikationen des Modells nicht oder nur äußerst eingeschränkt möglich sind. Dabei kann das Modell direkt durch den Entwickler in einer Programmiersprache erfasst oder mittels objektorientierten CASE–Tools erzeugt bzw. in einer Beschreibungssprache wie EXPRESS (textuell) oder EXPRESS–G (graphisch) definiert und durch einen entsprechenden Generator umgewandelt werden.

Leeuwen und Wagter formulieren in [Leeuwen 97] mit besonderem Hinblick auf den architektonischen Entwurf folgende Anforderungen an Modelle und Modellierkomponenten in Entwurfsumgebungen, die aber auf alle nicht primär numerisch orientierte Entwurfsdomänen verallgemeinerbar sein dürften:

- Erweiterbarkeit: erlaubt den Entwerfenden die Erweiterung der Menge der Definitionen, die das konzeptuelle Modell des Entwurfsgegenstands konstituieren. Dies ist erforderlich, um beispielsweise den Erfordernissen spezifischer Techniken bzw. Stile, Regeln oder Konventionen eines Projekts genügen zu können. Ebenso ist wünschenswert, Informationen, die für die Repräsentation der Charakteristika neuer Bautechnologien oder neuer Bauteile, –produkte oder –materialien erforderlich sind, abbilden zu können.
- Flexibilität: Die Notwendigkeit der Bereitstellung einer flexiblen Modellierbasis hat mehrere Teilaspekte. Einerseits wird mit der Möglichkeit der Definition von Informationseinheiten für spezifische Entwurfsaufgaben der dynamischen Natur von Entwurfsprozessen Rechnung

> getragen. Ebenso erfordert die Erweiterung von Modellen Flexibilität in diesen, da mit dem Hinzufügen neuer oder der Erweiterung bestehender Konzepte häufig auch neue Relationen zu anderen Konzepten im Modell bestehen und repräsentiert werden sollen. Dies gilt auch umgekehrt für Relationen von 'alten' Modellanteilen zu den neu hinzugefügten Informationen. Flexibilität ist auch für eine adäquate Unterstützung von Entwurfsprozessen notwendig, da diese als spezielle Problemlösungsprozesse eine häufige Interpretation und Umstrukturierung (oder Umklassifikation, Anm. d.A.) der Informationsbestände erfordern.

Neben der recht eingeschränkten Erfüllung der obigen Anforderungen sind mit statischen Modellen eine Reihe von weiteren Nachteilen verbunden. Diese sind meist Folgen der Grundproblematik, dass es als unmöglich angesehen wird, ein für alle Situationen und Entwurfsgegenstände umfassendes Bauwerksmodell zu definieren.

Ramscar charakterisiert den Prozess der Erstellung von Produktmodellen als unumgänglicherweise empirischen induktiven Prozess, bei dem von der analytisch ermittelten Existenz von Instanzen notwendigerweise im Modell abzubildenden Konzepten durch Klassifikation dieser auf abzubildende Modellklassen geschlussfolgert wird. Ebenso ist kein Weg einer formalen Verifikation von Bauwerksmodellen in Bezug auf deren Konsistenz und Umfassendheit bekannt, da es sich hier um eine Abbildung eines Ausschnitts der realen Welt in eine abgeschlossene Modellwelt handelt. Ramscar stellt in [Ramscar 94] einen mathematisch basierten, formalen Beweis vor, dass es nicht möglich ist, die Vollständigkeit eines Produktmodells zu beweisen. Die Unmöglichkeit eines derartigen Beweises korrespondiert mit dem Problem, dass äußerst wahrscheinlich jedes Produktmodell nicht umfassend definiert werden kann. Ramscar fordert für Entwurfsumgebungen Komponenten zur Unterstützung von induktiver Interpretation und Klassifikation zur Laufzeit, um mit neuen Elementen und Aspekten im Entwurf umgehen zu können, also wiederum erweiterbare und flexible Modelle.

Allgemein kann konstatiert werden, dass es als äußerst fraglich angesehen werden muss, ob eine einzelne und allgemeingültige Ontologie zur Beschreibung aller Aspekte von Bauwerken und ihrer Komponenten aufgestellt werden kann.

Die Nutzung von vordefinierten Klassen erfordert eine implizite Kommunikation zwischen Modellentwickler und Modellanwender über das Medium Software (d.h. hier Modell und Modellverwaltungskomponente). Daher besteht hier die Gefahr semantischer Lücken zwischen der Intention der Klassennutzung und ihrer eigentlichen Definition in statischen Systemen. Steinmann empfiehlt zur Unterstützung von Entwurfsaktivitäten in typischen CAD–Domänen mit Ill–Defined–Problems, in der Literatur auch als Innovative–Design bezeichnet, die Nutzung von dynamischen Verwaltungssystemen für deskriptive Modelle und betrachtet deren Anwendung ausführlich für frühe Phasen des architektonischen Entwurfs [Steinmann 97b].

Neben den genannten Problemen mit statischen Modellen sind unter anderem folgende weitere Vorteile dynamischer bzw. evolutionärer Modellieransätze zu nennen:

- Lebenszyklus–Aspekt: Es ist wünschenswert, Modelle zu schaffen, die ein Bauwerk von den ersten Entwurfshandlungen über mehrere Nutzungs– und Umbauphasen bis zum Abbruch begleiten können. Unter Berücksichtigung der langen Lebensdauer von Bauwerken erscheinen dynamische, deklarative Modelle, welche auch die inhärente Semantik der Modellinformationen in sich mit abbildenden, als der erfolgversprechendste Ansatz.
- Wechsel der Bauwerksnutzung: Nach einer Revitalisierung von Bauwerken und einem möglicherweise damit verbundenem Wechsel der Nutzungsart können teilweise andere Informationen für das Facility Management oder auch später nachfolgende Revititalisierungsmaßnahmen relevant sein. Dies unterstreicht die Forderungen nach erweiterbaren und flexiblen Modellen.
- Know How–Aspekt: Es ist wünschenswert, personalisierte Entwurfsumgebungen zu schaffen, in die der Entwerfende eigenes Entwurfswissen ablegen kann. Als formalisierte Modell-erweiterungen sind hier die Einfügung neuer Attribute und die Definition von Subtypen vordefinierter Modelltypen denkbar. Schwerer formalisierbare Informationen können als Verweise auf Bestände verschiedener Medien– oder Dokumenttypen im Modell abgelegt werden, um beispielsweise persönliche Anmerkungen oder Notizen, Gesetzeswerke, Entwurfsvorschriften– oder Richtlinien abzubilden. Ein Angebot zur Präsentation derartiger Informationen zum Zeitpunkt der Nutzung entsprechender Modellklassen kann als dem Entwurfsprozess adäquate Art des Informationsretrievals angesehen werden.
- Aufgaben–Aspekt und Vollständigkeits–Aspekt: Der Unikatcharakter von Bauwerken und der im Vergleich zu anderen Ingenieurdisziplinen geringe Standardisierungsgrad der eingesetzten Bauteile erhöhen wiederum die Wahrscheinlichkeit, mit vordefinierten statischen Modellen nicht alle vorhandenen, relevanten Informationen abbilden zu können. Dies betrifft vor allem durch ihre Größe oder spezifische Nutzung komplexe Bauwerke, bei denen eine Entwurfsunterstützung besonders wichtig ist, und alte/historische Bausubstanz. Bietet das Bauwerksmodell keine Möglichkeit, alle anfallenden Informationen abbilden zu können, werden Informationen in der Entwurfspraxis auf verschiedensten modellexternen Medien von Files, PDA–Einträgen bis hin zu Haftnotizzetteln abgebildet werden. Ein Informationsretrieval auf derartigen Medientypen ist naturgemäß nur schwer durch eine Entwurfsumgebung zu unterstützen.
- Softwarelebenszyklus–Aspekt: Flexible und erweiterbare Modelliersysteme besitzen zumindest potentiell eine längere Nutzungsdauer als Applikationen mit statischen Modellen, da diese leichter an neue Technologien oder Materialien im Entwurfsbereich, neue Versionen anzubindender konventioneller Softwaresysteme und Änderungen von Bauvorschriften, Normen etc. beispielsweise durch Import neuer Taxonomien angepasst werden.

Zur Anwendung von laufzeitdynamischen Bauwerksmodellen bietet sich die Nutzung der Modellmetaebene[14] an, da hier mit der Mächtigkeit des zugrundeliegenden Paradigmas der Funktionsumfang des Modellierkerns statisch beschrieben und damit in automatisierbare Form gebracht werden kann. Das Spektrum der Anwendungsmöglichkeiten reicht hier von weitgehend umfassenden Interfacedefinitionen wie die AKO–Schnittstelle [AKO 97] bis hin zu auf Tabellen relationaler Datenbanken beruhenden Datenstrukturen mit einem gewissen Grad ihrer Generizität und der Möglichkeit, Erweiterungen und eingeschränkt Modifikationen an Attributmengen von Entitäten und Typen vornehmen zu können [Richter 94].

Wichtig ist bei der Nutzung dynamischer Modelle die Definition und spätere Beachtung von Invarianten–Bedingungen, um bei Modifikationen des Modells dessen Konsistenz sichern zu können. Dies betrifft den Grad erlaubter Modifikationen von Klassen und Relationen, Regeln zur Instanzmigration, die Reichweite und Freigabepolitik bei Löschoperationen etc. [Eastman 99].

Trotz der Erweiterbarkeit und Flexibilität von Bauwerksmodellen erscheint es sinnvoll, für jede Bearbeiterrolle Kernbereiche der Taxonomie ihrer Domänen als Ausschnitt des allgemein akzeptierten Begriffsapparates des Fachgebietes bereitzustellen. Der Entwerfende kann sowohl während der Projektbearbeitung als auch unabhängig von aktuellen Projekten seine Taxonomie erweitern, d.h. neue Konzepte durch Spezialisierung vorhandener Informationen einbringen und in begrenztem Rahmen Modifikationen bestehender Klassen vornehmen. Ohne diese vordefinierten Kernbereiche wäre die Nutzung von speziellen Planungstools wie auch die Kommunikation zwischen Fachleuten derselben Teildisziplin gefährdet (sh. auch [Leeuwen 97]).

Ein auch unter den Befürwortern dynamischer Modelle häufig diskutierter Punkt ist die Frage, ob dem Fachplaner die Erweiterung der Modelle übertragen und zugemutet werden kann. Zweifelsohne ist eine geeignete Mensch–Maschine–Schnittstelle, die bei der Ausführung derartiger Aktivitäten assistiert, ein wichtiger Faktor für den Erfolg dynamischer Modelle. In dieser HCI–Fragestellung besteht dringlicher weiterer Forschungsbedarf.

Arbeiten lokal mehrere Entwerfende gemeinsam an der Lösung einer Teilaufgabe des Entwurfs für ein Bauprojekt, erscheinen Differenzen zwischen den Taxonomien dieser Bearbeiter nicht wünschenswert. Folglich sollten dynamische Modellmodifikationen lokal durch einen verantwortlichen, entsprechend geschulten Fachplaner oder für größere Organisationen durch einen Wissensingenieur vorgenommen werden. Augenbroe spricht in diesem Kontext von "local tiger teams", die derartige Modellanpassungen vornehmen könnten und verweist auf das Potential von Java–basierten Frameworks zur Einbindung benötigter Semantik in Modelle [Augenbroe 98].

Neben der oben erwähnten Problematik der Nutzerinteraktion bestehen bei der Nutzung dynamischer Modelle in weiteren Punkten Schwierigkeiten, die in dieser Form bei statischen Modellen nicht existieren:

[14] Metaebene: Die Begriffe Metaebene und Metaklasse werden ausführlich im Abschnitt 5.2 behandelt.

- Darstellung: Zur Visualisierung nutzerdefinierter oder –modifizierter Klassen müssen besondere Vorkehrungen getroffen werden
- Informationsweitergabe: Bei Austausch von Projektinformationen müssen die Taxonomieinformationen mit übertragen werden; der Empfänger der Informationen muss in die Lage versetzt werden, die Semantik der erhaltenen Informationen zu interpretieren.
- Applikationsanbindung: auf den Modellen aufsetzende Applikationen müssen mit Instanzen dynamisch definierter Klassen sinnvoll umgehen können.

Ein Teil der genannten Probleme lässt sich durch konsequente Nutzung der objektorientierten Paradigmas lösen, da beispielsweise Instanzen einer durch Hinzufügung eines spezifischen Attributs gebildeten Subklasse nicht zwangsweise durch sämtliche Fachapplikationen anders behandelt werden müssen als diejenigen ihrer Basisklasse.

Für einen weiteren Teil der Probleme werden in den Kapiteln 4 und 5 der vorliegenden Arbeit Lösungsansätze vorgestellt, ebenso sollen verschiedene Technologien zur Realisierung von dynamischen Modellverwaltungssystemen später im Kapitel 4 ausführlich betrachtet werden.

Aus der Literatur sind sehr wenige funktionsfähige und umfassende Umsetzungen dynamischer Modellierkerne für Bauwerksinformationen bekannt.

Kowalczyk beschreibt einen evolutionären Modellierkern, der einen Teil der Modelliermächtigkeit des objektorientierten Paradigmas abbildet. Allerdings sind einige Funktionalitäten durch Implementierungsgrenzen eingeschränkt (z.B. die Zahl der Oberklassen einer Klassse), die Möglichkeiten von Modellmodifikationen begrenzt und bestimmte andere nicht direkt in der verwendeten Programmiersprache abbildbare Modellierelemente wie Packages nicht vorgesehen. Die Implementierung erfolgte in einer unportablen frühen C++–Version und ist an die verwendete Betriebssystemumgebung HPUX V.9 gebunden. Mechanismen zum entfernten oder zur Kontrolle des synchronen Modellzugriffs konzipiert Kowalczyk nicht [Kowalczyk 97] [Werner 94].

Im Kontext der Esprit–Projekte VEGA und COMBI entstand die O.P.E.N.–Plattform (Object–oriented Product model Engineerig Network) als industrielle Implementation einer Middleware für Virtual Enterprises [Beetz 98]. Der Grundansatz von O.P.E.N. liegt in der Intergration durch gemeinsam genutzte Daten, um Datenaustausch zwischen den Applikationen zu vermeiden. Folglich wird eine Verwaltung der gemeinsamen Datenbestände des holistischen Produktmodells mit Präsentationstechniken nach dem MVC–Paradigma und Netzwerkzugriff mittels der proprietären Schnittstelle für verteilte Objekte DCOM. Um Flexibilität der Produktmodellinformationen zu gewährleisten, wurde ein 'Dynamic Product Model Kernel' (DPM Kernel) realisiert. Dieser verwaltet die aktuellen Projektinformationen ebenso wie das im Schema gespeicherte Wissen. Der DPM Kernel soll die Möglichkeit zur Dynamic Schma Evolution (DSE) bieten, also Erzeugung, Löschung und Modifikation von Schemainformationen zur Laufzeit erlauben. Informationen über die zugrundeliegende Realisierungstechnologie und den konkreten

Funktionsumfang sind nicht bekannt, ebenso liegen keine Veröffentlichungen über Praxisreife und –einsatz vor.

Auch in aktuellen Standardisierungsbemühungen ist eine gewisse Tendenz zu erweiterbaren und flexiblen Modellen zu erkennen. Einen ersten Schritt in diese Richtung bedeuten die Property Sets (Psets) der Industry Foundation Classes (IFC) der International Alliance for Interoperability (IAI), die eine Erweiterung von Instanzen um eine vordefinierte Menge von Attributen mit Informationen für bestimmte Planungsaufgaben oder –applikationen erlauben.

3.3 Verteilte Bauwerksmodelle

3.3.1 Zentrale vs. verteilte Modelle

Frühe Ansätze der Bauwerksmodellierung verfolgten häufig das Ziel, alle anfallenden Informationen in einem einzelnen globalen, physikalisch kohärenten und zentral gespeicherten Gesamtdatenbestand abzubilden, auf den eine Anzahl von speziell angepassten oder entworfenen Applikationen zugreifen [Eastman 99]. Dies trifft auf erste Arbeiten auf diesem Gebiet aus den siebziger Jahren zu, ebenso verfolgten einzelne Projekte bis in die neunziger Jahre diesen Ansatz. Ein bekannter und jüngerer Vertreter des genannten Ansatzes war das EU–Projekt COMBINE–1 [Augenbroe 94] [Dubois 94], das unter anderem ergab, dass derartige Ansätze vor allem für Domänen geringer Komplexität und Dynamik geeignet sind [Augenbroe 98].

Sehr häufig ist der Ansatz der Repräsentation der gesamten relevanten Bauwerksinformationen in einem einzelnen zentralen Modell und Speicherung dessen in einem physikalischen Datenbestand nicht zweckmäßig oder durch Randbedingungen der vorgefundenen Umgebung nicht möglich. Gründe dafür sind oft mit den vorgefundenen Erfordernissen von Fachapplikationen oder der Notwendigkeit der Integration bereits bestehender Systemmodule mit eigenen Partialmodellen verbunden. Ebenso sind in der Vergangenheit häufig Versuche der Aufstellung globaler Bauwerksmodelle an der extremen Komplexität des entstehenden Modells gescheitert. Ferner impliziert ein derartiger Ansatz den Anspruch, sämtliche im Lebenszyklus eines Bauwerks auftretenden Informationen in allen möglicherweise eintretenden Situationen abbilden zu können, was zu Problemen bei der Modellerstellung führt, da ein solches endgültiges ‚Vorausdenken' kaum möglich sein dürfte. Auch in der gegenwärtigen Praxis führt ein möglicher Wunsch nach globalen, zentralistischen Modellen zu Schwierigkeiten, da sich bis heute kein derartiges genormtes und allgemein akzeptiertes und verbreitetes Bauwerksmodell durchgesetzt hat.

Ein weiteres Problem ist, dass bei bestimmten, komplexeren Planungsvorhaben wie beispielsweise das Bauen im Bestand die Fachgebiete der zu beteiligenden Spezialisten nicht von vornherein zu spezifizieren sind. Beim Vorliegen schwieriger Planungssituationen werden entsprechende Experten in die Planungstätigkeiten mit einbezogen. Daher ist die Möglichkeit des

nachträglichen Zufügens der Partialmodelle mit relevanten Informationen für Spezialgebiete ein wichtiger Faktor für kooperativ nutzbare Entwurfsumgebungen.

Neben den genannten konzeptuellen Schwierigkeiten existieren auch technische Probleme mit globalen, zentralistischen Ansätzen. Bauplanungsprozesse werden heute zunehmend in ‚Virtual Enterprises'[15] ausgeführt, die für die Dauer des Planungs– und Bauprozesses existieren. Da die beteiligten Projektpartner im allgemeinen eine geographische Verteilung aufweisen, müssten dann sämtliche Modellzugriffe über Weitverkehrsnetze wie dem Internet geschehen. Ausfälle des Kommunikationsmediums und die in paketorientiert arbeitenden Netzen oft auftretenden Verzögerungen der Datenübermittlung würden die Nutzbarkeit der darauf aufsetzenden Applikationen stark beeinträchtigen.

Ferner sind globale Modelle mit eng angekoppelten Applikationen in Bezug auf deren Wartungsfreundlichkeit problematisch, da Änderungen an diesen durch die darauf aufsetzenden Applikationen reflektiert werden müssen, was zu einem enormen Aufwand und dadurch zu hohen Betriebskosten führen würde [Fenves 94].

Jedoch sind nicht nur zentralistische Ansätze mit Schwierigkeiten verbunden. Zum ersten tritt mit einer erschwerten Sicherung der logischen Konsistenz das klassische Problem verteilter Datenbestände auf. Weiterhin erfordern entfernte Modellzugriffe zum Datenaustausch und Modellabgleich Vorkehrungen zur Erhöhung der Robustheit der Systeme. Entwurfshandlungen in Weitverkehrsnetzen erfordern Autorisierungs– und Authentisierungsmechanismen sowie Vorkehrungen zur Sicherung des Dokumentencharakters der Ergebnisse des Bauplanungsprozesses. Für diese Probleme erscheint jedoch im Gegensatz zu den prinzipbedingten Schwierigkeiten zentraler Modelle eine Lösung durch Anwendung aktueller Verfahren der Informatik greifbar.

3.3.2 Verteilungs- und Verknüpfungstechnologien

Die Motivation der Untersuchung und des Einsatzes verteilter Modelle ergibt sich direkt aus der in den vorangegangenen Abschnitten betrachteten Problematik der Modellierung von Bauwerksdaten: einerseits ist die Abbildung aller relevanten Aspekte des in Planung befindlichen Bauwerks in einem zumindest logisch existierenden, kohärenten Gesamtmodell sehr wünschenswert, andererseits ist ein einzelnes physisches Bauwerksmodell aus Gründen der Verteilung, Arbeitsorganisation, Server– und Datenverfügbarkeit, Softwareeinsatz, etc. in sehr vielen Fällen nicht möglich.

In den oben gezeigten sequentiellen Ansätzen auf Modellebene erfolgt der Datenaustausch zwischen den verschiedenen Bearbeitern bzw. Rollen und den Partialmodellen häufig zu definierten Zeitpunkten, an denen eine Konvertierung kompletter Modelle bzw. Modellteile vorgenommen wird. Bei vertikalen und horizontalen Ansätzen der Integration wird von ständig

[15] Anforderungen des Concurrent Engineerings und der Planung in Virtual Enterprises an die Modellierung von Bauwerken werden ausführlich im Kapitel 3 behandelt.

verfügbaren gemeinsamen Informationsräumen in Form von Datenbanken, Blackboards, etc. ausgegangen. Werden nun Vorkehrungen getroffen, um einen kontinuierlichen oder relativ hochfrequenten Abgleich zwischen den Modellen zuzulassen, mit dem auch Änderungsmengen übertragen werden können und welcher die im vorhergehenden Abschnitt genannten Anforderungen weitgehend erfüllt, kann die Gesamtheit der Bauwerksinformationen als ein zusammenhängendes, logisch existentes Bauwerksmodell angesehen werden.

Bei der Betrachtung Verteilter Modelle ist zu unterscheiden, ob die Verteilung in einer einzelnen Sicht bzw. einem einzelnen Partialmodell auftritt oder ob die einzelnen Teilmodelle eines gesamten Bauwerksmodells verteilt verwaltet werden. Der erste Fall ist in der Bauwerksplanung oder auch allgemein in Virtual Enterprises nur eingeschränkt von Bedeutung, da eine dauerhafte geographische Verteilung innerhalb einer Rolle kein typisches Charakteristikum von Bauplanungsprozessen ist. Für temporäre entfernte Zugriffe auf Partialmodelle, wie sie beispielsweise zur Unterstützung von Telearbeit oder zur Ausführung von Aktivitäten in situ benötigt werden, erscheint eine Unterstützung durch Modellverteilung kein adäquater Ansatz zu sein, da unter anderem die hierfür benötigte technische Infrastruktur überdimensioniert wäre und diese Problematik durch einfachere Mechanismen behandelt werden kann. Folglich soll im folgenden vorrangig die in Virtual Enterprises häufig auftretende Problematik verteilter Partialmodelle eines logischen umfassenden Bauwerksmodells betrachtet werden.

Neben der genannten Frage der Ebene der Verteilung sind bei derartigen Modellieransätzen vor allem deren

- Struktur des sich ergebenden logischen Bauwerksmodells,
- die Funktionsweise und der Funktionsumfang der verbindenden Elemente,
- das technische Medium und die Schnittstellen zur Realisierung der Verbindung der Modelle,
- die Aufgaben und Nutzungsdauer im Bauwerksentwurfsprozess sowie
- die Semantik und der Grad der Heterogenität der Teilmodelle

bei einer näheren Betrachtung von Interesse.

Naturgemäß eignen sich zur Verwendung in Verteilten Modellen vor allem Strukturen, die über einen gemeinsamen Kern oder über eine Vielzahl von Relationen zwischen Teilmodellen verfügen, während Modelle für eine sequentielle Integration ohne neutralen Kern (siehe Abb. 3) ebenso wie Ansätze mit gegenseitig exklusiven Partialmodellen (siehe Abb. 7) eher schlecht geeignet sind, da Informationsflüsse durch mehrere Teilmodelle hindurch propagiert werden müssen, Daten u.U. mehrfach konvertiert werden und damit die Wahrscheinlichkeit von Normierungsverlusten recht hoch wird. Gut geeignet für Verteilte Modelle erscheinen Strukturen mit gemeinsamem Informationskern und möglicherweise gemeinsamen Metamodell (siehe Abb. 9 und 12), sofern es möglich ist, den gemeinsamen Kern zu definieren, was sich in der Praxis als schwierig erwiesen hat.

Im Falle eines einzelnen zentralen und applikations– und sichten–neutralen Modells[16] sind alle Relationen (auch Mappings genannt) zwischen dem Modell und den aufsetzenden Applikationen angeordnet und werden daher auch externe Mappings genannt.

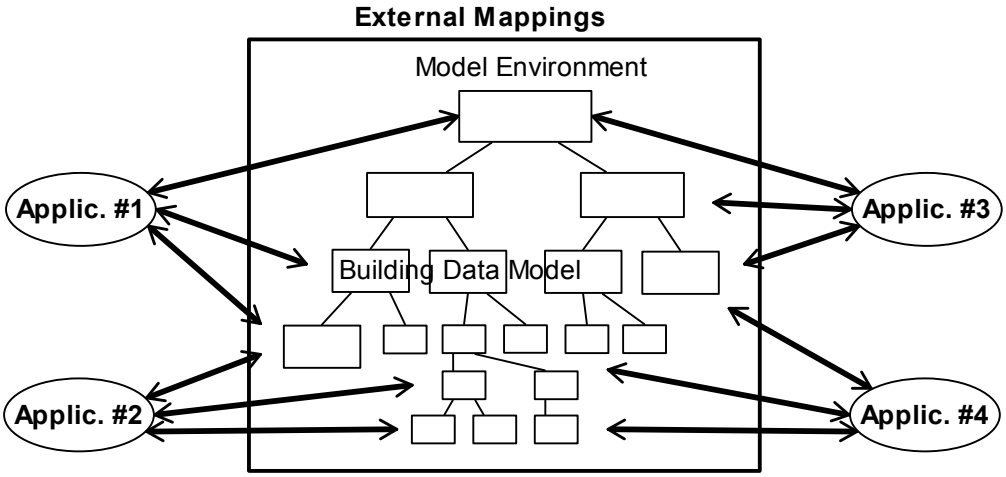

Abbildung 13: Einzelnes Zentrales Bauwerksmodell nach Eastman

Im Falle eines aus multiplen Sichten zusammengesetzten Gesamtmodells kann zwischen internen Mappings, welche die einzelnen Sichten in Relation setzen und externen Mappings zwischen Teilen des Modells und den Applikationen unterschieden werden [Eastman 99].

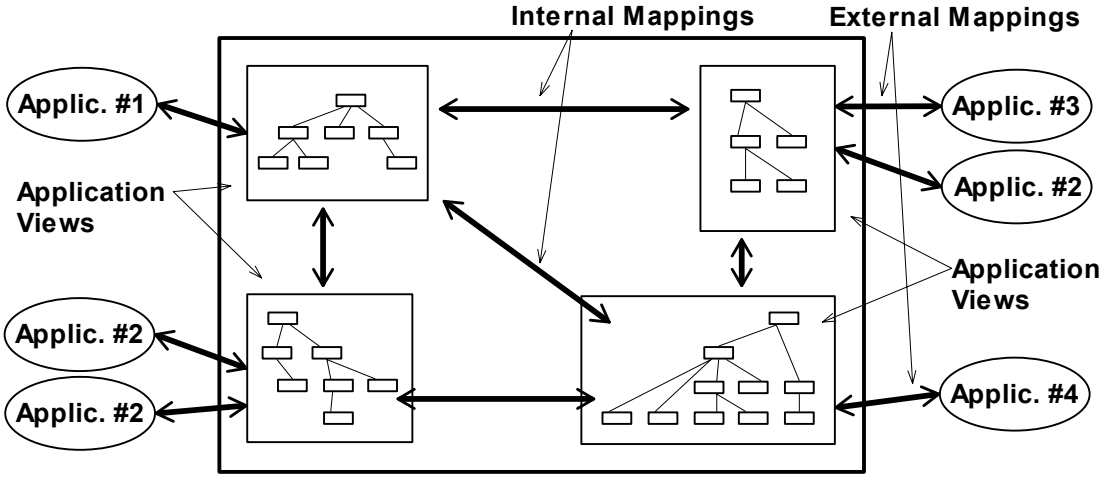

Abbildung 14: Separate Modelle mit externen und internen Mappings nach Eastman

[16] Diese Modellarchitektur wird bei Fenves et al. als "Tightly coupled database approch" eingeführt [Fenves 94].

Unter den Architekturen, die geeignet sind, aus verschiedenen Sichten bestehende Bauwerksmodelle abzubilden, kann grundsätzlich zwischen solchen mit und ohne zentralen Integrator[17] unterschieden werden. Dieses zentrale Integrationsmodell beinhaltet Daten, die für mehr als eine Domäne relevant sind, also potentiell einem Informationsaustausch unterworfen sind. Damit wird es im Idealfall möglich, dass sämtliche interne Mappings ausschließlich zwischen dem zentralen Integrationsmodell und den anderen Domänenmodellen verlaufen. Allerdings dürfte es schwer sein, von vornherein den Inhalt und eine in sich geschlossene Semantik dieses Modells zu definieren. Ebenso dürfte das nachträgliche Verschieben von Informationen, deren Bedeutung für andere Domänen nicht bei der Modelldefinition erkannt worden ist, aus einem Sichtenmodell in das zentrale Integrationsmodell sehr schwerwiegende Anforderungen an die Modellverwaltung und die Applikationen stellen. Ein Vorteil dieses Ansatzes ist aber nach Eastman, dass die Domänenmodellstrukturen relativ ähnlich zum Fachinformationsmodell der Applikationen sein können. Dann können die Mappings in beiden Richtungen so gestaltet werden, dass keinerlei Informationsverluste auftreten. Derartige Mappings werden isomorphisch genannt [Eastman 99] [Eastman97].

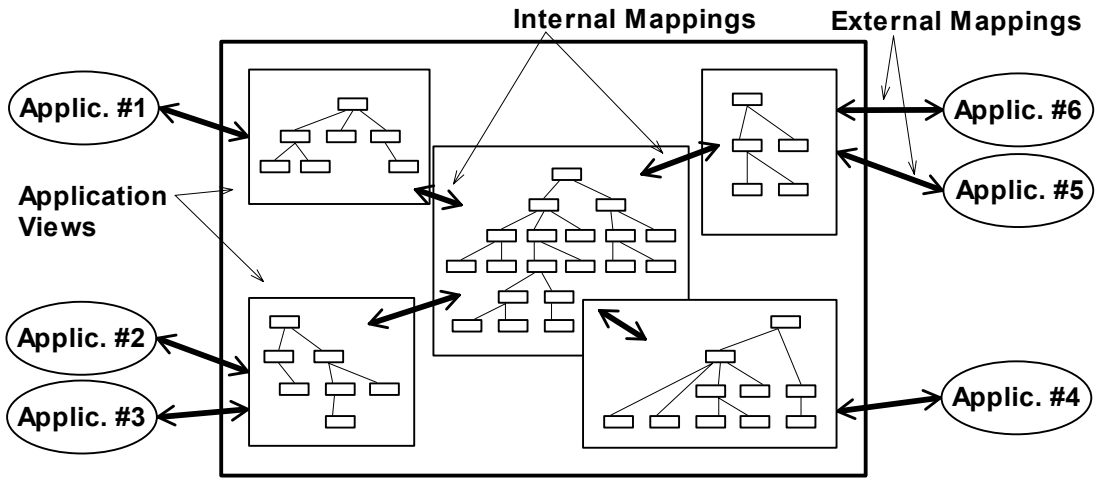

Abbildung 15: Separate Modelle mit zentralen Integrator nach Eastman

Je nach vorliegender Modellarchitektur und Systemimplementierung kann die Funktionsweise und der Funktionsumfang der Mappingmechanismen zwischen den einzelnen Teilmodellen differieren. Einerseits ist eine aktive Informationsübertragung vom Quell– zum Ziel–Teilmodell mittels eines push–Mechanismus denkbar, andererseits ist ein Abfordern der Informationen durch den Empfänger durch einen pull–Mechanismus denkbar. Der Wahl dieses Mechanismus

[17] Bauwerksmodelle ohne zentrale Integrationskomponente entsprechen den "Linked chain approach", jene mit zentraler Komponente dem "Hub-and-spokes approach" nach Fenves et al.

hat Einfluss darauf, ob ein Änderungsmanagement realisierbar ist, da hier nur aktive oder aus Benachrichtigung und pull–Übertragung kombinierte Mechanismen in Frage kommen.

Die Funktionalitäten der Mapping–Mechanismen können ebenfalls je nach Architektur und Realisierung verschiedenartig ausgeprägt sein. Einige mögliche Funktionalitäten von Mapping–Mechanismen sind beispielsweise:

- Modell–Übertragung: Bei der Übergabe von Planungsergebnissen eines vorgelagerten Arbeitschrittes werden sämtliche relevanten Informationen eines Partialmodells übertragen. Diese Vorgehensweise liegt beispielsweise bei der Übertragung statischer Domänenmodelle bei Fileformaten wie STEP zugrunde.
- Notifikation: Im Falle des Eintretens von Änderungen an Informationen eines Domänenmodells erfolgt eine Benachrichtigung an die Verantwortlichen für Teilmodelle, für die diese Informationen relevant sind bzw. deren Informationen vom geänderten Datum abgeleitet sind. Dem Vorteil der gesicherten Übertragung steht hier die mögliche Informationsüberflutung des Empfängers gegenüber. Bei Einsatz dieser Technik sollten daher dem Anwender durch die Entwurfsumgebung beispielsweise über intelligente Filter– und Kategorisierungstechniken assistiert werden; ebenso ist eine Unterstützung des Anwenders beim Erstellen der Mappings wünschenswert.
- Propagation: Neben der Benachrichtigung über eingetretene Änderungen an Modelldaten können diese auch automatisch in die betroffenen Partialmodelle übertragen werden, insofern die Zielsemantik der Mappings hinreichend exakt spezifiziert ist. Bei diesem Mechanismus muss jedoch muss gesichert werden, dass die innere fachlich logische Konsistenz des Empfänger–Modells durch die von außen initiierten Informationsänderungen nicht gefährdet wird. Ebenso sollten die Fachplaner über automatisch vorgenommene Änderungen an ihren Modellen informiert werden, da im allgemeinen Falle nur mit deren Know How die Implikationen für das Empfängermodell abgeschätzt werden können. Somit bestehen ähnliche Anforderungen an eine Unterstützung durch die Entwurfsumgebung wie im obigen Falle des Notifikationsmechanismus.
- Generierung: Durch die numerische Verknüpfung mehrerer Informationen aus einem oder auch mehreren Domänenmodellen können durch entsprechende Mappingvorschriften und –mechanismen neue Informationen in einem Partialmodell oder gesamte Modelle des Bauwerksmodells generiert werden.

Die oben genannten Beispiele für Mapping–Funktionalitäten schließen sich nicht gegenseitig aus, daher kann eine Implementierung durchaus mehrere der genannten Mechanismen aufweisen.

Ein weiteres Klassifizierungskriterium für Mappingmechanismen ist die Lebensdauer und der Zeitpunkt der Spezifikation der Relationen zwischen den Domänenmodellen. In statischen Modellstrukturen können Mappings zwischen Modellklassen schon zum Zeitpunkt deren Definition angegeben werden. Liegen dagegen dynamische Modellstrukturen vor, können die

Mappings naturgemäß ebenso nur dynamisch erzeugt werden. Bezüglich der Mappings zwischen Instanzen von Modellklassen können die Beziehungen je nach Mappingmechanismus und Grad der Modelldynamik spezifiziert werden. Bei statischen Modellen und kompletter Modellübertragung liegen die Mappings zwischen Modellinstanzen implizit vor und sind im allgemeinen in den Algorithmen der Konvertermodule fest verdrahtet. Bei anderen Mechanismen können die Relationen je nach Grad der Komplexität der Modelle durch die Entwurfsumgebung zur Laufzeit erzeugt werden, bei dynamischen Modellen kann dies nur durch zumindest teilweise Beteiligung der Fachspezialisten geschehen.

In Anhängigkeit der oben genannten Kriterien können technische Medien ausgewählt werden, auf denen die Umsetzung der Mappings beruht. Im Falle des Vorliegens einer Konverterlösung oder eines ähnlichen Offline–Mechanismus ist kein Einsatz eines Kommunikationsmediums erforderlich. Bei einer Online–Arbeitsweise der Mechanismen muss hingegen auf geeignete Netzwerktechnologien zurückgegriffen werden. Hier kommen sowohl nachrichtenbasierte Techniken Verteilter Systeme wie entfernte Prozedur– oder Methodenaufrufe, asynchrone Kommunikationstechniken aktueller Middleware–Technologien oder beispielsweise Software-agenten in Frage (siehe Kap. 4). Die Gestaltung der Applikations–Schnittstellen (und bedingt auch der evtl. vorhandenen Nutzerschnittstelle) zum Mapping–Mechanismus hängt wesentlich von der Art der Schnittstelle zum Modell ab. Liegt ein beispielsweise statisches Modell vor, kann auf die einzelnen Quell– und Zielobjekte sowie Mechanismen des Mappings direkt zugegriffen werden, liegt dagegen ein dynamisches Modell mit einer durch die Modellverwaltungs-komponente bereitgestellten Schnittstelle auf Meta–Niveau vor, wird auch der Mapping-mechanismus auf Metaebene arbeiten müssen.

Zur Repräsentation der Relationen zwischen Klassen mit Mitteln der objektorientierten Modellierung existieren verschiedene Ansätze. Beispielsweise wird von Olbrich ein auf den mathematischen Eigenschaften der Objektrelationen basiertes Modellierungs–Konzept ‚RelationShipModel' vorgestellt. In diesem Modellierungsschema bestehen Ausdrucksmittel in Form spezieller Klassen für Assoziationen, Relationen, attributierte Relationen, Rollen und bipartite Graphen. Grundlage des Entwurfs dieser Implementierung sind Untersuchungen differenzierter Entwurfsmuster für Assoziationsrelationen, welche zur Zusammenfassung von Kriterien für Objektbeziehungen in einem Schema und zur Beschreibung der Implementierungs-klassen durch möglichst orthogonale Invarianten dienen [Olbrich 98].

Im obigen Abschnitt wurde vorrangig der Online–Informationsaustausch durch Mappings betrachtet. Die Mapping–Problematik tritt jedoch auch bei der Zusammenführung oder Übertragung existierender Schemata beliebiger Art auf. Beispielsweise sind derartige Fragen ebenso wie jene des Umgangs mit semantischer bzw. logischer Heterogenität für Verteilte

Datenbanken[18] und Federated Databases[19] [Heimbigner 85] [Sheth 90] zu klären. Unter semantischer Heterogenität versteht man, dass identische oder überlappende Informationen häufig über verschiedene Schemata repliziert sind, aber in diesen nicht identisch repräsentiert werden oder dass in mehreren Quellen identisch benannte Informationen verschiedene Bedeutungen und Intentionen besitzen [Hakimpour 01]. Dabei sind folgende fundamentale Aufgaben in Bezug auf die Verwaltung der Informationsbestände zu erfüllen [Hull 97]:

- Schemaintegration – Schaffung einer einheitlichen Sichtweise auf den logischen globalen Informationsbestand
- Updates – Aktualisierung semantisch heterogener Daten in mehreren Modellen
- Identifikation und Mapping – Spezifikation der Beziehung zwischen zwei oder mehr Instanziierungen der replizierten Information
- Synchronisation – Gewährleistung der logischen Konsistenz der replizierten Information.

Die oben genannten Aufgaben können als generell relevant für Informationen verschiedenen Strukturierungsgrads unabhängig von der Form ihrer Speicherung angesehen werden.

Bei der Zusammenführung mehrerer Schemata können ebenso strukturelle Konflikte auf Modell- und Datenebene auftreten. Diese lassen sich nach Won Kim in fünf folgende Kategorien klassifizieren, die hier nachfolgend sinngemäß von der relationalen Terminologie auf objektorientierte Systeme übertragen wurden, dabei betreffen die ersten drei die Klassen und die folgenden zwei Punkte die Daten [Kim 91] [Kim 95]:

1. Klasse zu Klasse Konflikt – kann 1:1 wie auch m:n auftreten
    - Klassennamenkonflikte: unterschiedliche Namen für semantisch gleiche Konzepte oder gleiche Namen für unterschiedliche Konzepte
    - Klassenstrukturkonflikte: fehlende Attribute oder implizit (z.B. durch die Typinformation) modellierte Informationen

[18] Unter Verteilten Datenbanken (Distributed Databases – DDBS) werden Architekturen verstanden, in denen ein einzelnes Distributed Data Base Management System (DDBMS) mehrere verteilte Datenbank-Bestände verwalten.

[19] Unter 'Federated Database Systems' (FDBS) wird eine Anzahl kooperierender, aber autonomer 'Komponenten-Datenbanksystemen' verstanden. Dabei sind verschiedene Integrationsgrade (enge bzw. lose Kopplung) möglich. Die zentrale Verwaltungskomponente wird als Federated Database Management System (FDBMS) bezeichnet und ist für die Verwaltung der Verbund–Topologie, die Einfügung neuer Komponenten und die kontrollierte Manipulation der verwalteten Informationen verantwortlich. Die einzelnen Komponenten-Datenbanken steuern ihre Interaktionen mit anderen Komponenten häufig über Import- und Export–Schemata. Export–Schemata spezifizieren hier die Informationen, die eine Komponente anderen Komponenten zur Verfügung zu stellen bereit ist, Import-Schemata beschreiben die Menge der nichtlokalen Informationen, die eine Datenbank manipulieren möchte [Sheth 93] [Heimbigner 85].

- Klassen–Constraint–Konflikte: Einschränkung der Menge der semantisch korrekt zuordenbaren Instanzen der Quellklasse zu denen der Zielklasse, dieser tritt häufig dann auf, wenn Quell– und Zielklasse auf unterschiedlichem Taxonomieniveau modelliert werden
- Klassen–Einschluss–Konflikt – ist mit dem Klassen–Constraint–Konflikt entgegengesetzt, aber ursächlich verwandt und tritt auf, wenn die Menge der Instanzen einer Quellklasse mit mehreren disjunkten Zielklassen in Beziehung steht

2. Attribut zu Attribut Konflikt – kann 1:1 wie auch m:n auftreten
   - Attributnamenkonflikte
   - Defaultwertkonflikte
   - Attribut–Constraint–Konflikte
   - Attribut–Einschluss–Konflikt
3. Klasse zu Attribut Konflikt: Modellierung einer Information wechselseitig als eigene Klasse bzw. als Attribut
4. Inkonsistente / Falsche Daten
   - Verletzung der Datenkonsistenz: semantisch äquivalente Informationen besitzen unterschiedliche Attributwertbelegungen
   - Veraltete Daten infolge von Nichtausführung von Updates in einem bzw. mehreren Modellen
5. Unterschiedliche Repräsentation derselben Daten
   - Unterschiedliche Ausdrücke – Darstellung semantisch äquivalenter Informationen durch unterschiedliche Datentypen, Codes, Symboliken, Abkürzungen etc., beispielsweise können Prüfungsnoten sowohl durch numerische Skalen mit unterschiedlichem Wertebereich bei besseren Ergebnissen absteigend oder aufsteigend als auch durch natürlichsprachliche Prädikate z.B. in Deutsch oder Latein ausgedrückt werden
   - Unterschiedliche Maßeinheiten
   - Unterschiedliche Genauigkeiten der Werte

Diese allgemeinen Aussagen lassen sich auch auf Probleme des Modellmappings innerhalb von Planungsumgebungen für den Bauwerksentwurf übertragen.

Um Mappings zwischen Bauwerksmodellen beschreiben zu können, wurde von Amor et al. die Mapping–Definitionssprache View Mapping Language (VML) entwickelt. Durch diese sollen Mappings beschrieben werden, die zur Generierung der Bauwerksdaten neuer Modelle aus den Instanz–Informationen bestehender Domänenmodelle, zum Update abhängiger Modelle durch Änderungspropagation und zur Überprüfung der Konsistenz der Informationen im aktualisierten

Modell genutzt werden können. Dabei wurde auf eine deklarative Sprachdefinition zurückgegriffen, um bidirektionale Mappings mit einem einzelnen Ausdruck beschreiben zu können; prozedurale Ansätze erfordern eine separate Beschreibung für jede Richtung. Mit VML können direkte Übertragungen von Werten, einfache numerische Abhängigkeiten unter Nutzung der Grundrechenarten und einer Anzahl von im VML Programming Environment (VPE)–System vorhandenen Funktionen, sowie für komplexere Fälle unter Angabe zweier Mapping–Prozeduren mit gegenseitig invertierter Funktionalität für beide Mapping–Richtungen textuell oder graphisch beschrieben werden. Jedoch muss konstatiert werden, dass einerseits die VML relativ komplex ist und daher kein adäquates Arbeitsmittel für einen Fachplaner darstellen dürfte und das die bestenfalls semi–dynamische Arbeitsweise von derartigen Mapping–Sprachen nur bedingt für die dynamischen Organisationsformen in Virtual Enterprises anwendbar ist [Amor 94] [Amor 97].

Darüber hinaus existieren eine Anzahl weiterer Sprachen zur Definition von Mappings, wie beispielsweise EXPRESS–M, EXPRESS-X, EDM–2 usw. Diese Sprachen zielen ebenso auf die Beschreibung von Relationen zwischen statischen Domänenmodellen und sind vorrangig für die Nutzung im EXPRESS/STEP–Kontext konzipiert [Eastman 99] [Amor97].

Eine relativ ausführliche Behandlung der Mapping–Problematik für (statische) EXPRESS–Modelle sowie die Beschreibung einer Anzahl notwendiger Tools zugeschnitten auf die Belange des Facilty Managements, findet sich ferner in [Kolbe 97].

Weiterhin sei auf aktuelle, parallel in Bearbeitung befindliche Forschungsarbeiten innerhalb des Projektbereichs D "Informationsverarbeitung" des Sonderforschungsbereichs 524 "Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken" verwiesen, bei denen ausführlichere und weiterführende Aussagen zur Definition, Verwaltung und Anwendung von Mappingtechniken zu erwarten sind.

3.4 Entwicklungsstand integrierter Bauwerksmodelle

3.4.1 Datenaustauschformate

Im folgenden soll ein kurzer Überblick der Bemühungen auf dem Gebiet der Bauwerks-modellierung gegeben werden. Dabei soll die Entwicklung dieser Technik beleuchtet werden und kurz auf bestimmte, bekannte Vertreter der Entwicklungsstufen mit ihrer Grundphilosophie und ihrer Leistungsfähigkeit eingegangen werden. Für ausführliche Beschreibungen der einzelnen Bauwerksmodellierungsprojekte und –spezifikationen soll hier jedoch auf die teilweise reichlich vorhandene Literatur (z.B. [Eastman 99]) und auf die (leider in einigen Fällen recht kostenintensiven) Originaldokumentationen verwiesen werden.

Die Bemühungen um die Schaffung von Bauwerksmodellen lassen sich bis in den Zeitraum Ende der siebziger bis Anfang der achtziger Jahre zurückverfolgen, als entsprechende Termini noch unbekannt waren. Im allgemeinen werden Systeme wie die Krankenhaus–Entwurfsumgebung

OXSYS (Anfang 70er Jahre) des Instituts ARC der Universität Cambridge, dessen kommerzieller Nachfolger BDS (Building Design System – 1978) und die Designsprache GLIDE (1974) mit der dazugehörigen Umgebung CAEADS (1980) als die ersten Bemühungen auf dem Gebiet der Bauwerksmodellierung angesehen. Als zweite Phase der Entwicklung von Bauwerksmodellen können die Bemühungen zur Abbildung von Bauwerken in Expertensystemumgebungen angesehen werden. Die späteren Entwicklungsphasen waren fundamentale Produktdaten-modelle, generische Produktmodelle, Sichten und Schichten–Modelle. Zusammen mit Applikationsmodellen bilden diese Ansätze häufig die Komponenten umfassender Bauwerks-Produktdatenmodelle (Terminologie nach Björk: Comprehensive Building Product Data Models). Einen gewissen Konsens über Inhalte und Strukturierungsansätze konnte jedoch erst Anfang der neunziger Jahre erreicht werden [Junge 98] [Eastman 99], nichtsdestotrotz sind Fragen der Gestaltung von Bauwerksmodellen auch noch gegenwärtig ein aktueller Forschungsgegenstand.

Dabei werden Techniken der Produktmodellierung sowohl für die Schaffung neutraler Austauschformate als auch für die Realisierung von integrativen Komponenten verschiedenen Typs in Entwurfsunterstützungssystemen verwendet.

Der internationale Standard STEP (ISO1303) wurde als Reaktion auf verschiedene nationale Standardisierungsansätze in Europa wie das französische SET und das deutsche CAD*I und VDA–FS entwickelt, um die betreffenden Aktivitäten zu bündeln und später mit den entsprechenden Bemühungen in den Vereinigten Staaten (PDES) verschmolzen.

STEP basiert auf dem ANSI/SPARC–Standard für Datenbanken, der eine Schichtenarchitektur von Abstraktionen und Mappings definiert. Daher besitzt auch STEP

- ein 'physical layer' mit der physikalischen Repräsentation des Schemas beispielsweise in einem Filesystem,
- ein 'logical layer' für die abzubildenden Informationen und
- ein 'application layer' zur Repräsentation der anwendungsspezifischen Sichten.

Zur STEP–Architektur gehören fünf Klassen von Komponenten, die im STEP–Kontext 'Tools' genannt werden:

- Beschreibungsmethoden (description methods) zur Spezifikation der Informationen in der gesamten Architektur von STEP, also von Intergrated Ressources, Application Reference Models und Application Interpreted Models. Beispiele sind NIAM, IDEF1x, EXPRESS.
- Integrierte Ressourcen (integrated ressources) sind häufig bei der Definition von Applikationsmodellen verwendete Modellbestandteile, generische IR werden in den unterschiedlichsten Domänen genutzt, branchenspezifische IR werden dagegen 'appication integrated ressources' genannt.

- Anwendungsprotokolle (application protocols) sind Spezifikationen, die für den Kontext bestimmter Applikationen entwickelt werden. Diese besitzen ein 'Application Reference Model', welche die notwendigen Informationen des AP unter Anwendung der Terminologie der Domäne häufig grafisch mittels NIAM, IDEF1x oder EXPRESS–G beschreibt. Das auf Basis des ARM zu entwickelnde 'Application Interpreted Model' soll sowohl für entsprechende Applikationen als für Anwender verständlich sein und eine vollständige Definition der abzubildenden Informationen enthalten. AIM's werden häufig in EXPRESS formuliert.
- Implementations–Methoden (implementation methods) bilden zusammen mit den AP's die Basis einer STEP–Umsetzung. Beispiele dafür sind STEP physical file (SPF) und STEP Data Access Interface (SDAI).
- Konformitätsprüfung (conformance testing) ist ebenso erforderlich für Application Protocols und Implementierungen, um deren Übereinstimmung mit der ARM's und AIM's zu testen.

Im Kontext der Entwicklung von STEP wurden unter anderem die Sprachen EXPRESS und EXPRESS–G entwickelt. EXPRESS ist eine Sprache zur Definition der Informationsstrukturen in Domänenmodellen, EXPRESS–G ist deren graphische Variante [Eastman 99] [Froese 96]. Leider muss festgestellt werden, dass die Praxisrelevanz von STEP hinter den hohen Erwartungen zurückgeblieben ist. Gründe dafür können im späten Erscheinen der betreffenden Standards, im Aufwand des Umgangs mit entsprechenden Formaten, im mangelnden Engagement großer Firmen der IT–Branche und in der Notwendigkeit der hochqualitativen Strukturierung der Informationen vor deren Austausch. In Bezug auf die Modellierung von Bauwerksinformationen ist zu beobachten, dass sich UML für die Definition von Informationsstrukturen durchsetzt [Augenbroe 98].

Eine Anzahl von Forschungsprojekten und spezifischen Normierungsbemühungen sind auf Basis von STEP durchgeführt worden. So wurde durch den Deutschen Stahlbauverband die modellbasierte Produktschnittstelle Stahlbau auf Basis von STEP definiert [DStV 00] [Haller 95].

Das EUREKA–Projekt CIMsteel beschäftigte sich seit 1987 mit Spezifikationen für den Datenaustausch unter besonderer Ausrichtung auf Rahmentragwerke im Stahlbau. Der hauptsächliche Fokus lag auf Effizienzsteigerung im Stahlbau und der Stahlindustrie durch Anwendung von CIM–Techniken. Dabei wurden im Unterthema 'CIMsteel Integration Standards' (CIS) Fragen der Modellierung und des Datenaustauschs betrachtet und ein produktmodell-basiertes Austauschformat unter Nutzung von STEP definiert, dass für alle Applikationen im Stahlrahmenbau und für einen großen Bereich des Lebenszyklus von Planung bis zum Abbruch nutzbar sein soll. Dazu wurde ein 'Logical Product Model' (LPM) entwickelt, dass die Informationsbedürfnisse von Applikationen für Tragwerksanalyse, (Teil–)Entwurf, Verbindungsentwurf und konstruktive Detaillierung abbildet und für diese Teilaufgaben vier Data Exchange Protocols (DEP's) besitzt. In einer zweiten Version der Spezifikation 'CIS/2', die durch das Teilteam der University of Leeds betreut wird, wurde ein deutlich vergrößertes LPM mit über 600 Klassen definiert und ferner Funktionalitäten zum inkrementellen Datenaustausch

sowie zur Unterstützung von Concurrent Engineering hinzugefügt. CIS/2 kann als Vorgänger des Stahlbau Applikationsprotokolls STEP AP 230 angesehen werden [Watson 94] [CIS 00] [Eastman 99].

3.4.2 Systeme mit Modellmanagementfunktionalitäten

Das EU–ESPRIT Projekt ATLAS 'Architectures Methodologies and Tools for Computer Integrated Large Scale Engineering' (EP 7280) beschäftigte sich von 1992 bis 1995 ebenfalls mit dem Bauwerksdatenaustausch auf semantischer Ebene für Entwurfs–, Konstruktions– und Management–Aufgaben innerhalb von Large–Scale Engineering (LSE) Projekten. Konkret wurden die Domänen Architektur, HLS, Akustik und Tragwerksplanung untersucht. Als Basis der Datenaustauschformate wurde STEP und EDIFACT verwendet, weiterhin wurde die Entwicklung von Konvertern ("Integration Tools") zum Zugriff auf die Informationen geplant. Das Konzept umfasst eine mehrschichtige Architektur von einem 'Large Scale Engineering Project Type Model' bis zu 'Application Level Models'. Es wurden 'Interrelation Models' mit Beziehungen zwischen den Systemteilen definiert, trotzdem ist kein Konzept einer über Nutzung von neutralen Modellen und Konvertereinsatz hinausgehenden Systemarchitektur aus den vorliegenden Veröffentlichungen zu erkennen. Die Integration wissensbasierter Komponenten war ebenso geplant, die Bemühungen im ATLAS fokussierten jedoch eher auf der Kopplungsproblematik als auf Fragen der Inhalte und Strukturierung der Wissensbasen [ATLAS 93] [Tolman 94] [Poyet 94].

Das Objektorientierte Modellmanagement–System OOMM des Instituts für Numerische Methoden und Informatik im Bauwesen der TU Darmstadt unterstützt für den Entwurf von Stahlbeton–Geschossbauten die Integration von Architekturentwurf und statischer und numerischer Modellierung. Ziel war die durchgängige Unterstützung des Ingenieurs im Planungsprozess und die Bereitstellung einer ingenieurgerechten, bauteilorientierten Sichtweise auf das Gesamttragwerk. Dazu wurden separate Teilmodelle und Applikationen für die jeweiligen Prozesse entwickelt, die Modelle werden aber uniform durch OOMM verwaltet. Der Datenaustausch zwischen den Modulen erfolgt unter Nutzung von STEP–2DBS[20] [Meißner 95] [Peters 97].

Das EU–Projekt COMBINE (Computer Models for the Building Indutry in Europe) wurde mit 11 Partnern aus 7 Ländern im Rahmen des JOULE–Programms für effektive Energienutzung durchgeführt. Phase 1 des Projekts lief von 1990 bis 1992, die zweite Phase von 1992 bis 1995, insgesamt betrug der Aufwand siebzig Mannjahre. Grundaufgabe war die Demonstration der Mächtigkeit existierender Technologien bei der Integration von Applikation für bauphysikalische

[20] STEP–2DBS (STEP–2D–Building Subset) [Haas 93] ist im Rahmen früher Bemühungen des DIN Arbeitskreises DIN–NAM96.4.3.–Bau entstanden und sollte zur Nutzung von STEP für den Datenaustausch im Bauwerksentwurf beitragen. Seit der Verabschiedung späterer ISO–STEP Applikationsprotokolle kann STEP–2DBS als obsolet angesehen werden, es wurde leider auch keine nennenswerte Praxiswirksamkeit erreicht.

bzw. bauenergetische Berechnungen und für die HLS–Planung. In der ersten Phase wurde ein IDM (Integrated Data Model) genanntes einzelnes zentrales Bauwerksmodell mit über 400 Klassen–Entities aufgestellt. Dieses IDM umfasste alle Informationen der angebundenen sechs Applikationen, welche über Schnittstellen mit Views auf das globale Modell versorgt wurden. Dies führte letztendlich schon zu Problemen im Umgang mit derartig komplexen Modellen, die noch nicht die Semantik aller bei durchschnittlichen Bauplanungsaufgaben benötigten Werkzeuge und Domänen beinhalten. Der Datenaustausch erfolgt in der ersten Phase über Datenträger mit STEP–Dateien, in der zweiten Phase wurde das Datenmodell in einer zentralen objektorientierten Datenbank (unter ObjectStore) verwaltet. Diese diente als Kern einer DES (Data Exchange System) – Architektur, die über Data Interaction Managers offline und online Datenaustausch unterstützen sollte. Der Online–Datenaustausch wurde lokal über C++–API's der Komponenten ausgeführt. Nach Abschluss der Arbeit an COMBINE wurde von den Projektdurchführenden konstatiert, dass die Integration aller bzw. vieler Sichten und Tools in ein einzelnes, zentrales Modell nicht empfehlenswert ist, und wenn unumgänglich, im Umfang begrenzt werden muss. Die semantische Integration durch ein einzelnes zentrales Modell führt weiterhin zu transaktionszentrischen Systemen, denen Prozesskontext fehlt und daher schwer nutzbar sind [Augenbroe 94] [Dubois 94] [Lockley 94] [Augenbroe98] [Eastman 99].

Das Projekt ANICA (Analysis of Interfaces of various CAD/CAM–Systems) der Universität Kaiserslautern untersuchte auf Initiative der deutschen Automobilindustrie die Schnittstellen zu den Systemkernen einiger Cax–Hersteller analysiert und ein Konzept für kooperierende Cax–Systeme entworfen. Zur Lösung wurde ein Cax–Objektbus auf Basis von CORBA und STEP AP 214 realisiert, der auch erfolgreich in einer praxisnahen Umgebung getestet wurde. Aufgrund der starken branchenspezifischen Unterschiede in Bezug auf Entwurfsgegenstände und Organisation des Entwurfsprozesses sind diese Ergebnisse nur bedingt übertragbar [Dankwort 96] [Iselborn 98].

Das ESPRIT III–Projekt COMBI (Computer–Integrated Object–Oriented Product Modeling Framework for the Building Industry) beschäftigte sich von 1992 bis 1995 unter der Grundphilosophie "Integration durch Kommunikation" mit Concurrent Engineering und kooperativen Design im Bauwerksentwurf. Dazu wurde ein flexibles Framework zur Repräsentation der Bauwerksmodelldaten geschaffen, mit dem verschiedene, auch teilweise neu geschaffene Applikationen mit ihren dazugehörenden rollenspezifischen Sichten integriert werden können. Als Anforderungen an dieses Framework wurde die Möglichkeit des Mappings zwischen verschiedenen Aspekten desselben physikalischen Objekts, die Transformation zwischen verschiedenen Objektrepräsentationen und eine dynamische Objektevolution und –reklassifikation gestellt. Als Lösungsansatz wurde ein hybrider Ansatz eines zentralen Modells mit Satelliten gewählt, dieses besteht aus einem zentralen domänenunabhängigen neutralen teilgenerischen Modellkern, einigen Domänen–Aspektmodellen und daran angelagerten Applikationsdatenmodellen [Scherer 94] [Katranuschkov 94].

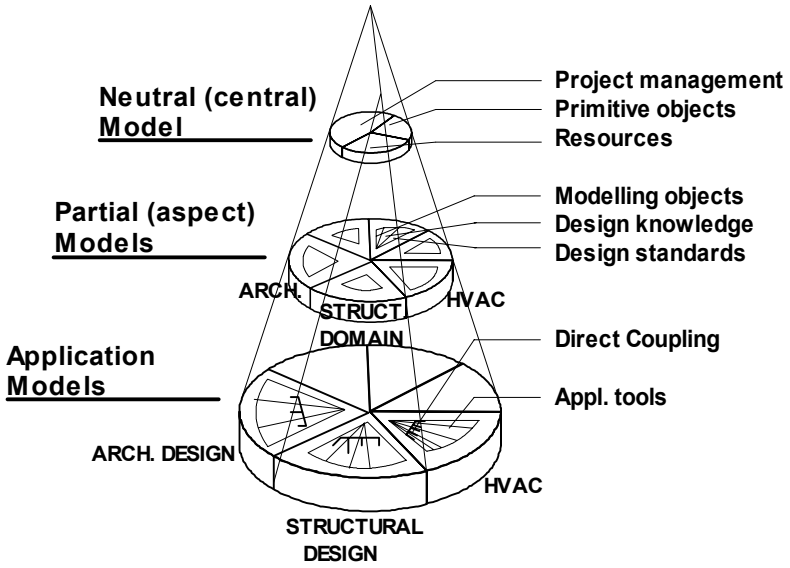

Abbildung 16: COMBI- Modellarchitektur

Die Interaktion zwischen den einzelnen Komponenten wurde in COMBI durch den Austausch von STEP–Files gelöst. Die Transformation zwischen Objekten des neutralen Kernmodels und denen der Domänenmodelle wird durch Steuerungskomponenten eines Blackboard–Systems vorgenommen. Als Nachteil des Ansatzes von COMBI gilt die hohe Komplexität der zentralen Komponenten, die sich auch in einem sehr hohen Verwaltungsaufwand niederschlägt sowie des explosionsartigen Anwuchses seines Inhaltes.

Das COMBI–Nachfolgerprojekt ToCEE (Towards a Concurrent Engineering Environment in the Building and Engineering Structures Industry) baute auf vergleichbaren Konzepten auf, die Architektur wurde jedoch erweitert und aktualisiert. ToCEE beruht auf einer physikalischen verteilten Client–Server–Architektur und einer fünfschichtigen Datenmodellarchitektur.

Die obersten Schichten sind sogenannte Ontologie–Layer für die Projekt- und die Virtual Enterprise–Sichten zur Prozessintegration. Sie dienen zur logischen Integration der Server und der serverspezifischen Datenstrukturen in ein einzelnes projekt– bzw. virtual–enterprise–weites Modell. Die drei Basisebenen werden über alle Projektserver repliziert und entsprechen der COMBI–Datenstruktur Neutral Layer, Aspect Layer und Application Layer [Scherer 98].

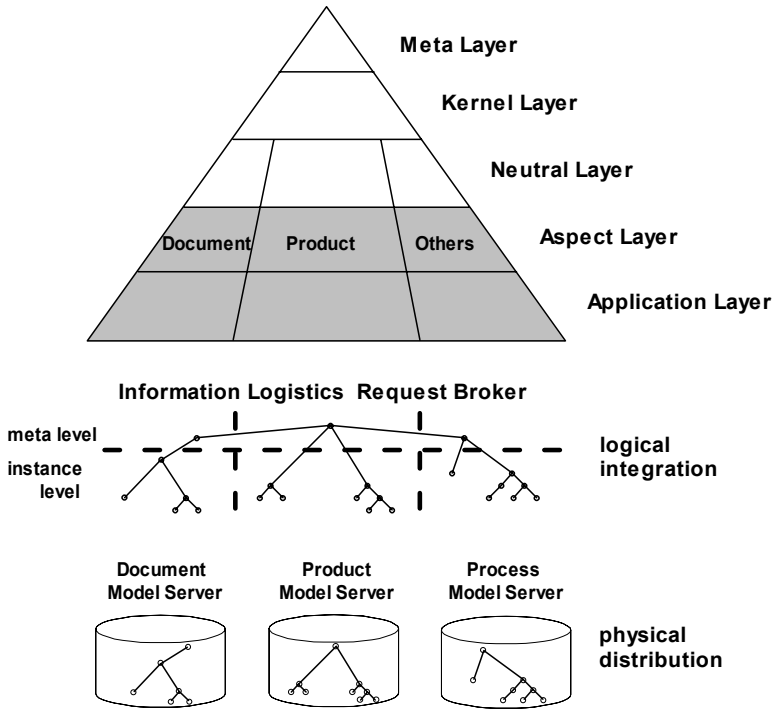

Abbildung 17: ToCEE- Modellarchitektur

Im VEGA–Projekt (Virtual Enterprises using Groupware Tools and distributed Architectures) wird ein sogenanntes konsolidiertes Modell genutzt, das aus fünf Schichten besteht. Die drei Basisschichten entsprechen wiederum COMBI und ToCEE, sind also Application Models, Aspect Models und ein Building Product Data Model Kernel. In Ebene 4 nutzt VEGA ein Generic Product Description Model und in Ebene 5 ein Fundamental Data Model. VEGA führt in der Modellarchitektur sogenannte 'Common Concepts' ein. Diese werden auf einer Ebene oberhalb der Domänenmodelle definiert und beinhalten Konzepte, die in einer Anzahl von Partialmodellen eine domänenspezifische Definition besitzen. Da die vertikalen Mappings zwischen der Ebene der 'Common Concepts' und den Domänenmodellen nach Ansicht der VEGA–Entwickler wesentlich einfacher zu definieren sind als jene zwischen einzelnen Domänenmodellen, können diese zur Vermeidung dieser horizontalen Mappings genutzt werden. Neben dem o.g. Konzept für die O.P.E.N.–Platform für dynamische Modelle verfügt VEGA über COAST–Stores zur Speicherung EXPRESS–definierter Produktmodelldaten in STEP–Formaten [Junge 97] [Junge 97a] [Stephens 98].

Die International Alliance for Interoperability (IAI – gegründet 1994 als Industry Alliance for Interoperabilty) ist ein Zusammenschluss von Industrieunternehmen, Forschungseinrichtungen und staatlichen Institutionen verschiedener Länder. Sie verfolgt vorrangig das Ziel, Interoperabilität zwischen verschiedenen in der A/E/C–Industrie relevanten Applikationen durch Produktmodellspezifikationen wie die IFC zu gewährleisten. Die Basis der Spezifikationen ist ein vierphasiges IAI–Bauwerkslebenszyklusmodell, das Machbarkeitsstudie, Entwurf, Konstruktion

und Bauausführung umfasst. Für jede dieser Phasen wurden Prozessmodelle definiert, dabei wurde Schwerpunkt auf die Aktivitäten gelegt, die durch Rechnereinsatz unterstützbar sind.

Die Industry Foundation Classes (IFC)– Spezifikationen definieren die Bauwerksmodell-informationen, die ausgetauscht werden können. Dabei werden drei verschiedene Definitionsviews verwendet: EXPRESS und EXPRESS–G für die Definition von Datenbank-Schemas und Austauschfiles, CORBA–IDL für Laufzeitobjekt–Definitionen aufgrund der Vielzahl von Language Bindings und Textdefinitionen für Informationen, die sich auf formalisiertem Wege nicht oder nur schwer ausdrücken lassen.

Die Version 1.5 der IAI–Industry Foundation Classes wurde 1997 vorgestellt, eine leicht aktualisierte Version 1.5.1 1998. Frühere Versionen der IFC wurden nicht durch für produktiven Einsatz geeignete Software unterstützt. Diese Versionen unterstützen im wesentlichen den Informationsaustausch im Umfang von Rohbaudaten; austauschbare Informationen sind:

- die logische Struktur des Gebäudes (Gebäude, Geschossgliederung)
- die Raumplanung (Räume und ihre Zuordnung zur Gebäudestruktur)
- die tragenden Bauelemente (Wände, Stützen, Balken, Decken, Dächer)
- die Öffnungen in Bauelementen (Fenster– und Türöffnungen, Durchbrüche, etc.)
- die Details zu Fenstern und Türen (Aufschlagsrichtung, Öffnungsart, etc.)
- allgemeine Geometrieelemente (Geometriedaten aller Planungsobjekte)
- Beziehungen zwischen den Bauelementen (z.B. Wandverschneidungen)
- Beziehungen zwischen den Bauelementen und der Gebäudestruktur (Zuordnung der Bauelemente zu den einzelnen Räumen und Geschossen, etc.)
- freie Attribute und Zusatzinformationen (wie eindeutige Kennung) zu allen Objekten

Die Version 2 der IFC Version 2.0 wurde 1999 freigegeben und beinhaltet neue Objekte für die Haustechnik, das Umzugsmanagement, für Facilities Management, die Kosten– und Ablaufplanung und für thermische Lastberechnungen. Aufgrund des Zeitdrucks bei der Fertigstellung der Release 2.0 wurde 2000 eine Maintainance Release 2x nötig, welche vorrangig zur Qualitätssicherung diente. Diese dient aber auch als Plattform für Erweiterungsprojekte der IFC und enthält neue Definitionen unter anderem für Organisationsstrukturen und eine verbesserte Referenzbibliothek.

Interessant ist eine parallel zur Version 2 entwickelte XML–Informationsaustauschversion der IFC 'IfcXML'. Die Bauwerksmodelle und Strukturen entsprechen der IFC 2x, d.h. die Semantik der zugrundeliegenden Definitionen entsprechen einander. Zur Nutzung dieser Definitionen gibt es zwei Varianten, einerseits kann eine XML–Datei wahlweise zu einer existierenden STEP–Datei durch eine Toolbox erzeugt werden, andererseits können die ifcXML Definitionen als Repository für eigene XML Austauschdefinitionen genutzt werden.

Trotz des fundierten Ansatzes muss auch den IFC–Spezifikationen eine gewisse Skepsis entgegengebracht werden. Die Spezifikation ist äußerst komplex, nur ein Teil der Softwarefirmen engagiert sich für diese Variante des Informationsaustauschs und die Abbildung der Informationen nach Nutzung bestimmter gängiger Entwurfsoperationen (z.B. nach Teilung eines Raums) ist nicht endgültig geklärt. Die sehr begrenzten Möglichkeiten von eigenen Schemaerweiterungen und die recht eng gefasste Auffassung zum Bauwerkslebenszyklus dürfte die Anwendungsfelder weiter einschränken[21].

[21] Inwieweit umfassende Interoperabilität und Standards wirklich konsequent und durchgängig verfolgte Ziele der IAI sind, wird nach einem Besuch der Website ungewiss, die im Gegensatz zu verbreiteten propretären Lösungen einige W3C–standardkonforme Browser nicht unterstützt, wodurch viele Nutzer verschiedener Betriebssysteme Schwierigkeiten beim Besuch der Präsentation haben dürften.

B: GroupPlan: Ein Groupwaresystem für die Bauwerksplanung

Übersicht:

4 Konzept des Bauwerksmodelliersystems

In den folgenden Kapiteln sollen das Konzept und die Implementierung eines verteilten, die Teamarbeit unterstützenden Bauwerksmodelliersystems beschrieben werden. Dabei soll im vorliegenden Kapitel 4 die Modellierung und die Verwaltung des Modells behandelt werden, im Kapitel 5 soll auf die Problematik der Verteilung der Modellier- und Entwurfsumgebung eingegangen werden, während im Kapitel 6 die Groupware – Eigenschaften des Systems dargestellt werden.

4.1 Modelliersystem

4.1.1 Objektsystem

Die Abbildung des formalisierbaren Anteils des Entwurfwissens der einzelnen Domänen bzw. Fachdisziplinen erfolgt in Domänen-Modellen. Die Modellierung erfolgt unter Nutzung des objektorientierten Paradigmas, dessen Vorteile im Kapitel 3 dargelegt wurden. Die grundsätzliche Herangehensweise an die Modellierung von Bauwerken des hier beschriebenen Projekts findet ihre Parallelen in den am selben Lehrstuhl bearbeiteten Projekten PREPLAN und AKO, erweitert diese aber wesentlich durch Konzepte und Architekturen zur verteilten und asynchronen sowie synchronen kollaborativen Projektbearbeitung.

Die Aktivitäten des Entwurfs von Bauwerken gehören je nach aktueller Aufgabe in die Kategorie des Innovativ- sowie des Routine–Designs und stellen somit besondere Ansprüche an computerbasierte Entwurfsunterstützungssysteme. Eine mögliche konzeptuelle Basis für derartige Systeme stellen linguistische Modelle dar [Liebich 93]. Entwurfsgegenstände werden unter Bezugnahme auf die Funktion, die sie im Bauwerk realisieren, anfangs lediglich als Ausprägung ihres Typs verstanden. Im weiteren Entwurfsprozess wird sukzessive die konkrete Idee des Entwurfsgegenstandes entwickelt. Das mentale Bauwerksmodell des Entwerfenden entsteht also in Top–Down–Richtung von abstrakten Vorstellungen hin zu konkreten Entwürfen.

Die Systemphilosophie konventioneller CAD–Systeme entspricht jedoch nur bedingt den Erfordernissen aller anstehenden Aufgaben, vor allem der architektonische Entwurf kann kaum durch derartige Systeme adäquat unterstützt werden. Durch das Bottom–Up–Konzept konventioneller CAD–Systeme ist der Entwerfende gezwungen, das mentale Modell bis zu einem gewissen Grad fertigzustellen, bevor externe Repräsentationen des Bauwerks erzeugt werden können. Zeitweise treten im Entwurfsprozess auch Bottom–Up–Phasen auf, beispielsweise wenn diese leicht lösbar oder vorgegebene Detailinformationen mit den eigenen Entwurfsintensionen in Einklang zu bringen sind. Weiterhin wäre der Entwerfende bei strengem Top–Down–Vorgehen gezwungen, auch triviale oder unbewusste Entwurfsobjekte zu externalisieren. Demzufolge sollen die Entwurfsaktivitäten am Bauwerksmodell sowohl in Bottom–Up– als auch in Top–Down–

Richtung möglich sein. Bei Letzterem wird der Entwurfsgegenstand initial lediglich als reine Ausprägung seines Typs charakterisiert, deren Zustandsinformationen sukzessive angegeben und verfeinert werden. Weiterhin wird die Typzugehörigkeit des Entwurfsobjekts möglicherweise weiter spezialisiert. Diese kognitiven Aspekte des Entwurfsprozesses finden ihr Pendant in der Informationstechnologie bei der objektorientierten Modellierung [Steinmann 95].

Die Unterstützung von Top–Down–Entwurf bedeutet die Abbildung von abstrakten mentalen Entwurfskonzepten in einer rechnergestützten Entwurfsumgebung. Dies impliziert wiederum die Notwendigkeit, in dieser zeitweise mit vagen, unvollständigen oder sogar inkonsistenten Informationszuständen umgehen zu können. Ein Ansatz des Umgangs mit dieser zeitweisen Unschärfe ist die Nutzung von facettierten Attributen, die Angaben über relaxierbare Restriktionen und Default-Werte beinhalten. Bei der Lösung dieser Problematik adaptiert GroupPlan Konzepte des FlexOb–Systems. Die Basis der Arbeit mit GroupPlan stellen domänenspezifische Taxonomien entsprechend der Rolle des jeweiligen Bearbeiters dar, in denen formalisiertes Wissen über den jeweiligen Entwurfsbereich abgelegt ist.

Für jede Bearbeiterrolle wird eine Taxonomie bereitgestellt, die wesentliche Elemente des allgemein anerkannten Begriffsapparates seiner Domäne enthält. Diese Klassenhierarchie kann durch den Bearbeiter sowohl während der Projektbearbeitung als auch unabhängig von aktuellen Projekten modifiziert und erweitert werden, um sie an spezielle Bedürfnisse anzupassen oder persönliches Know–How abzulegen.

Zur Realisierung der genannten Anforderungen beinhaltet das Bauwerksmodelliersystem folgende Modellierelemente:

- Objekte: Objekte des Modelliersystems bilden die vorhandenen planungsrelevanten Informationen über den aktuellen Entwurfsgegenstand, also über spezielle reale Entitäten, aus der Perspektive der jeweiligen Fachdisziplin ab. Alle Objekte im Modellverwaltungssystem besitzen eine eindeutige Identität durch einen uniquen Identifikator; ihr Zustand wird durch die gegenwärtige Belegung der definierten Klassen-eigenschaften und die aktuell bestehenden Relationen festgelegt. In rein deskriptiven Modellen besitzen die Objekte kein spezifiziertes Verhalten, d. h. es sind keinerlei Methoden bzw. Operationen angegeben. Dies ist für die reine modellhafte Beschreibung eines Bauwerkes oder anderen Produktes im Sinne einer Informationsabstraktion (Semantik-objekte) durchaus sinnvoll. In dynamischen Modellen ist jedoch eine Möglichkeit zur dynamischen Definition von spezifischen Objektverhalten für den Umgang mit Modellobjekten, wie z.B. deren Präsentation zu Dialogzwecken oder deren Visualisierung durch Präsentations– oder View–Objekte wünschenswert.
- Klassen: Wie bei objektorientierter Modellierung üblich, fassen Klassen jeweils eine Menge von Objekten mit identischen Attributen und Methoden zusammen. Durch diesen Prozess der Generalisierung wird von Kenntnissen über die Eigenschaften und Struktur einer Menge gegenwärtig bekannter Objekte Wissen über den allgemeinen Aufbau von Exemplaren der

jeweiligen Klasse bzw. Typs[1] gewonnen; diese Erkenntnisse werden dann auch auf nicht bekannte oder zukünftige Instanzen übertragen. Dieses Wissen und seine Abbildung ist für den Entwurfsprozess essentiell, da hier Erkenntnisse über die Gesamtheit der Exemplare einer Klasse benötigt werden, die auf die konkreten Objekte im Entwurfsbereich übertragen werden, welche initial nur in der Imagination des Planers, dann im virtuellen Bauwerksmodell und erst später in der gebauten Realität existieren. Neben der Funktion der Klasse als 'Bauplan' oder 'Schablone' für die dazugehörigen Exemplare existieren Klassenobjekte mit eigenem Speicherbereich, die neben Instanz–Lebenszyklusaufgaben für die Verwaltung von Defaultwerten für Attributbelegungen zuständig sind. Diese Attribute entsprechen technisch den *class attributes* von Smalltalk–80 bzw. *static class members* von C++ oder Java und werden genutzt, um solange auf die Defaultwerte zuzugreifen, bis für ein konkretes Objekt der jeweilige Attributwert belegt worden ist.

- Taxonomien / Generalisierungshierarchien: Durch Generalisierungshierarchien werden taxonomische Relationen, also *is-a-kind-of-* Beziehungen dargestellt. Die Generalisierungsbeziehung verläuft in Richtung des allgemeineren Konzepts, also in Richtung Superklasse (Ober–, Basisklasse). Die Klassenrelation der Gegenrichtung wird mit Spezialisierung bezeichnet. Entlang der Spezialisierungspfade erfolgt die Übertragung von Eigenschaften und Verhalten der Superklasse zur Subklasse. Im hier beschriebenen Modellverwaltungssystem sind einer Klasse im Unterschied zu vielen objektorientierten Sprachen und Systemen sowohl deren Super– als auch Subklassen bekannt. Momentan sind im Modellverwaltungssystem Vererbungshierarchien in der Form eines gerichteten Wurzelbaums, also eines zusammenhängenden azyklischen Graphen mit ausgezeichnetem Wurzelknoten, also Einfachvererbung implementiert. Auf *Multiple Inheritance* wurde aufgrund der Vielzahl der bekannten Probleme in der Modellierungsphilosophie als auch in der Umsetzung vorerst verzichtet.
- Attribute: Durch Attribute[2] werden bestimmte Eigenschaften von Klassen und deren Exemplaren beschrieben. Im hier zu beschreibenden Modellverwaltungssystem es ist möglich,

[1] Die Termini Klasse und Typ werden in der einschlägigen Literatur oft synonym benutzt und sind als eng verwandt zu betrachten. Typen beschreiben Mengen von Objekten, die gleiche Eigenschaften besitzen, für die gemeinsame Operationen definiert sind und die auf den Empfang derselben Botschaft einheitlich reagieren. Die trifft auch auf Klassen zu, diese besitzen jedoch eine konkrete Semantik, üblicherweise eine Implementierung und treffen Aussagen über den internen Aufbau und die möglichen internen Zustände der Objekte. Somit ist es denkbar, dass zu einem Typ mehrere Klassen existieren.

[2] Im konkreten Modellverwaltungssystem wird aus projektgeschichtlichen Gründen der Begriff Slot verwendet. Bestimmte objektorientierte Systeme zur Wissensrepräsentation wie Frame-Systeme nutzen Slots zur Abbildung von Attributen. Die gelegentlich anzutreffende Verallgemeinerung des Begriffs Slot auf Attribute, Methoden und Relationen wird im hier beschriebenen Modellverwaltungssystem nicht verwendet. Aus Gründen der Abwärtskompatibilität wurde nach der Einführung von speziellen Konzepten zur Abbildung von Relationen der ursprünglich genutzte Terminus beibehalten.

facettierte Attribute zu verwenden. Facetten sind die 'atomaren' bzw. terminalen Modellierelemente und repräsentieren einzelne Teile einer Menge zusammengehörender Informationseinheiten einer einzelnen Eigenschaft. Beispielsweise könnte ein Attribut Länge eine Wertfacette, eine Facette für die Maßeinheit und Facette für relaxierbare sowie strenge Minimum- bzw. Maximum-Restriktionen enthalten. Wie in der Mehrzahl der Objektmodelliersysteme und OOPL – eine prominente Ausnahme ist Smalltalk – besitzen Attribute eine für alle Exemplare gültige Attributtypdefinition, die den Datentyp, der im Attribut gespeicherten Informationen spezifiziert und letztendlich den Typ der Wert–Facette und den unter Umständen vorhandenen Restriktionen festlegt.

- Aggregationen: Aggregationen dienen zur Abbildung von *a-part-of-* bzw. *has-a-* Relationen, also Gesamtheit–Teil–Strukturen, durch die eine Menge von Einzelteilen zu einem kompositen, komplexen Objekt zusammengefasst werden. Umgekehrt kann ebenso die Zerlegung eines Objekts in seine Komponenten modelliert werden. Wird eine nähere Unterscheidung in Aggregationen, Komposition etc. in der Modellverwaltung vorgenommen, müssen als Konsequenz verschiedene Lebenszyklusoperationen an die Operationen zur Aggregationsverwaltung gekoppelt werden.

- Assoziationen: Neben der Generalisierungs– und der Aggregations–Relation mit ihrer jeweils vordefinierten Semantik existiert die Assoziation zur Abbildung von allgemeinen Beziehungen. Eine konkrete Semantik ist hier nicht vorgegeben, sollte aber durch die Rollennamen bzw. den Assoziationsnamen angegeben werden. Assoziationsbeziehungen können sowohl zwischen Klassen als auch zwischen Objekten existieren, in bestimmten Fällen sind auch Assoziationen zwischen Klassen und Objekten sinnvoll.

- Packages: Packages, oder je nach Modelliermethodik auch Subjekte oder Kategorien genannt, dienen zur logischen Gruppierung von Klassen bzw. Objekten im Modell. Modelle mit einer hohen Anzahl von Klassen werden schnell unübersichtlich, durch Einsatz von Packages können diese Modelle in übersichtliche, verständliche und wartbare Teilmodelle aufgeteilt werden, dabei sind Bezüge auf Modellelemente aus anderen Packages möglich.

- Domänenmodell: Das hier vorgestellte Modellverwaltungssystem ist in der Lage, auch mit einem MVS–Kern mehrere Domänenmodelle zu verwalten.

- Methoden: Methoden sind über die Schnittstelle des Modellverwaltungssystems derzeit nicht direkt definierbar. Bei einem Teil der später im Verlauf des Kapitels vorzustellenden Techniken zur Modellverwaltung ist aber eine Integration von direkt implementierten Methoden oder anderen klassenspezifischen, zur Laufzeit durch Interpretation ausführbaren Codesequenzen durchaus denkbar. Es soll hier noch einmal darauf hingewiesen werden, dass die hier verwendeten deskriptiven Bauwerksmodelle keine Methoden benötigen, dass aber Methoden an Hilfsklassen für bestimmte Zwecke wie Interaktion, Präsentation, einfache Konsistenzprüfungen und sehr eingeschränkt auch für die nachträgliche Anpassung der Verarbeitungslogik hilfreich sein können.

Die oben beschriebenen Informationen können als formalisiertes Entwurfswissen im Modellverwaltungssystem abgelegt werden. Weiterhin müssen jedoch auch schwer formalisierbare Beschreibungsanteile gespeichert und im Kontext bereitgestellt werden. Dazu gehören CAD–Zeichnungsformate, Textverarbeitungsdokumente, Files (quasi–)standardisierter Dokumentenformate wie PDF, Spreadsheets sowie multimediale Informationen diverser Medientypen, also beispielsweise Audiosequenzen und Bilder bzw. Fotos. Die Informationen können z.B. existierende Bausubstanz beschreiben, persönliche Notizen beinhalten, aber auch allgemeine Informationen zum Fachgebiet, wie Vorschriften, Gesetzlichkeiten, Kataloge, Normen, Richtlinien umfassen. Um eine Anbindung an das Modell zu erreichen, stehen spezifische nichtstatische sowie statische Attributtypen zur Verfügung. Dabei sollten allgemeine Informationen bezüglich Konzepten der Disziplin an Klassen abgelegt werden, projektbezogene Daten dagegen an Instanzen im Modell. Der Vorteil der Verbindung von formalisierbaren und nichtformalisierbaren Informationen liegt in der Möglichkeit des einfacheren Informationsretrievals und der möglichen Präsentation vorhandener Informationen im Kontext von Entwurfshandlungen mit Bezug auf die betreffende Klasse bzw. das aktuelle Objekt.

Wie eingangs erwähnt, wurden in diesem zur Vorbereitung der Unterstützung verteilter, kooperativer Arbeit notwendigem Punkt durch das GroupPlan–System und die Ergebnisse der Arbeit innerhalb des Teilprojekts D3 "Digitales Bauwerksmodell" des SFB 524 "Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken" wesentliche Erweiterungen und Aktualisierungen bezüglich der Modellierungstechniken von PREPLAN und AKO erbracht, deren grundlegende Philosophie aber beibehalten. Daher ist im obigen Anschnitt nur kurz auf die einzelnen Elemente des Modelliervorrats eingegangen worden. Weitere detaillierte Erläuterungen finden sich in den Dokumenten und Veröffentlichungen zu den genannten Projekten wie z.B. [Steinmann 97a] [AKO 97] [Willenbacher 98] [Hübler 94a] [Steinmann 95].

4.1.2 Metamodell

Für eine Bauwerksmodellierung auf Basis laufzeitdynamischer Domänenmodelle müssen besondere Vorkehrungen getroffen werden. Herkömmliche, traditionelle Bauwerksmodelle werden entweder durch Tools aus Beschreibungssprachen wie EXPRESS, EXPRESS–G, UML, etc. generiert oder direkt in einer geeigneten Programmiersprache implementiert. Anschließend muss im allgemeinen das Modell kompiliert und je nach Programmiersprache die Applikation gelinkt werden. Sobald Änderungen am Domänenmodell notwendig werden, muss dieser Entwicklungszyklus neu durchlaufen werden, unter Umständen hat eine Modellanpassung auch Auswirkungen auf die Datenbank und die Nutzerinteraktionsmodule der Applikation. Nach einem Neustart der Applikation oder gegebenenfalls nach einer Neuinstallation sind die Änderungen verfügbar.

Da in einem korrekt modellierten Domänenmodell für den Bauwerksentwurf sich keinerlei Ausdrucksmittel für eine Anpassung des Modells selbst finden sollten, muss für das Management eines dynamischen Domänenmodells eine spezielle Menge von Operationen bereitgestellt werden. Derartige Modelle, die Eigenschaften und Verhalten bzw. gültige Operationen mit Modellen beschreiben, nennt man Metamodelle. Metamodelle sind also für den Lebenszyklus und das Handling der durch sie beschriebenen Modelle verantwortlich. Metamodelle können, aber müssen nicht zwingend nach demselben Paradigma strukturiert werden. Systeme mit einer Metaebene behandeln Klassen auch als Objekte, Metaklassen beschreiben hier das Verhalten von (regulären) Klassen.

Das hier beschriebene Modellverwaltungssystem nutzt zur Verwaltung der objektorientierten Domänenmodelle auch ein objektorientiertes Metamodell, d.h. im Metamodell existieren Klassen zur Beschreibung z.B. von Klassen, Objekten, Packages und Slots. Nur wenige Programmiersprachen besitzen eine echte Metaebene. Beispielsweise ist in Smalltalk-80 eine vollwertige Metaebene integriert, in Java sind bestimmte Features wie Introspektion integriert, während direkte Möglichkeiten zur Veränderung von Klassen fehlen. C++ besitzt mit RTTI einige wenige Metaebenenfunktionalitäten.

Folglich muss für eine Bauwerksentwurfsumgebung, die obige Anforderungen erfüllen soll, eine Metaebene realisiert werden, um Entwurfsaktivitäten nach einem kognitiven Entwurfsmodell zu unterstützen. Die Nutzung einer Metaebene in der Bauwerksmodellierung ist ein relativ junger Ansatz für Modellarchitekturen. Obwohl dieser Ansatz mit einer Anzahl von Vorteilen verbunden ist, ist die Anzahl der publizierten Projekte mit derartigen Modellarchitekturen überschaubar [Hannus 94] [Hannus 95] [Junge 97] [Scherer 98] [Katranuschkov 94]. Dabei existieren verschiedene Motivationen für derartige Architekturen. Neben der Unterstützung dynamischer Bauwerksmodellierung und einfacher Modellerweiterbarkeit werden folgende Vorzüge genannt bzw. erwartet:

- Die Metaebene kann einheitliche, geeigneterweise objektorientierte Elemente für die Modellierung von Bauwerken bereitstellen, dies erleichtert den Informationsaustausch zwischen verschiedenen Domänen.
- Mappings zwischen Modellen können aufgrund der identischen Modellierfunktionalität einfacher realisiert werden, ebenso kann ein gemeinsam genutzter zentraler Modellkern mit geringerem Aufwand realisiert werden (siehe Kap. 3, Abb. 12).
- Ein generischer Mechanismus zum Versionsmanagement kann realisiert werden.
- Eine höhere Modellflexibilität kann erreicht werden, diese betrifft auch den Detaillierungsgrad der repräsentierten Objekte, die Organisationsstruktur des Informationsaustauschs und die Menge der Domänen, deren Informationen im Gesamtmodell verwaltet werden.

- Die Modellarchitektur trägt zur Vermeidung von Redundanzen durch mehrfache Datenduplikation bei.
- Eine unique, eindeutige Identifikation aller Objekte und Versionen ist notwendig; diese kann durch die Module der Metaebene bereitgestellt werden.
- Das Bauwerksmodell kann weitgehend unabhängig von konkreten Fachapplikationen, Implementierungsplattformen und –umgebungen sowie spezifischer Hardware und Betriebssystemen gehalten werden.

Im SFB 524 sowie im GroupPlan–Projekt wird eine erweiterte und aktualisierte Version der AKO–Schnittstelle als Definition der Metaebene der Modellverwaltung genutzt. Die AKO-Schnittstelle ist ursprünglich als C++-Interface zur Entkopplung von Applikationen und darunter liegenden Modellmanagementmechanismen von Partnern der Bauhaus-Universität Weimar, der TU Berlin, der TU Dresden und der Firma Hochtief Software geschaffen worden. Um eine Verteilung von Modell und Modellverwaltung zu erreichen, wurde ca. 1998 in der Bauhaus-Universität Weimar eine CORBA-IDL-Version der AKO-Schnittstelle [Willenbacher 98] definiert, die vorrangig zur Anbindung von Smalltalk-80-basierten Modellverwaltungssystemen (Visual Works mit Distributed Smalltalk) genutzt wurde. Diese Version wurde um Relationen, Packages, ein verbessertes Exception Handling, neue vordefinierte Datentypen und eine Reihe kleiner Funktionalitäten erweitert und in der Umsetzung auf eine neu konzipierte, leistungsfähigere, im folgenden beschriebene Basis gesetzt. Die Metaebene des Modellverwaltungssystems, also dessen Vorrat an Modellierelementen, ist in wesentlichen Teilen in folgender Abbildung dargestellt.

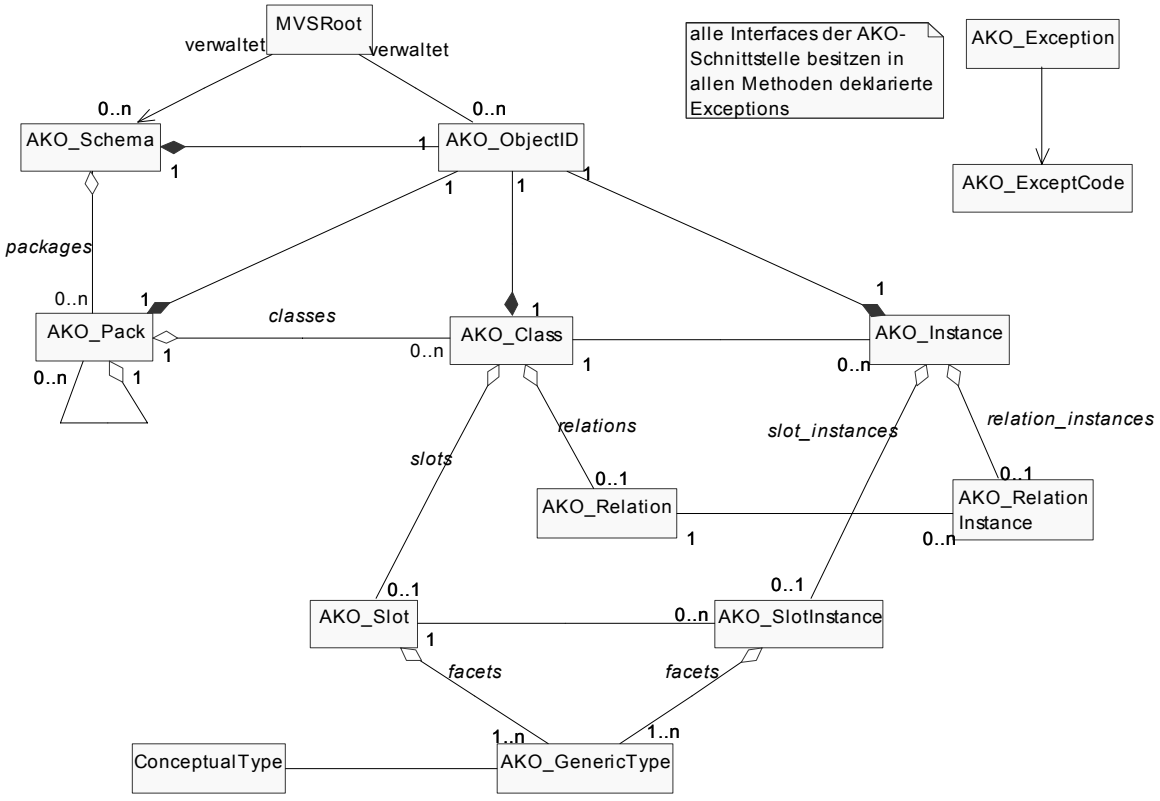

Abbildung 1: UML-Darstellung der aktualisierten AKO-Schnittstelle (Auszug)

Dynamische Modellverwaltungssysteme ermöglichen Änderungen des Bauwerksmodells zur Laufzeit. Beim Entwurf der Funktionalitäten des Metamodells müssen die Auswirkungen der Änderungen am Modell auf eventuell schon vorhandene Instanzen der betroffenen Klasse oder deren Subklassen beachtet werden. Relativ unkritisch sind Hinzufügungen von Attributen oder Relationen zu Klassen, die in leeren bzw. mit Defaults belegten Wertbelegungen an den Instanzen resultieren. Werden Löschungen uneingeschränkt zugelassen, wird dies zu Informationsverlusten der betreffenden Attribute führen. Sehr problematisch sind jedoch uneingeschränkt erlaubte Attributtypänderungen, die unter Umständen beim Konvertieren von Wertbelegungen zu Datenveränderungen führen können. Die folgende Tabelle listet ausgewählte Operationen der Metaebene und deren Auswirkung auf existierende Daten auf:

Operation	**Auswirkung**	**Erläuterung**
Erzeugen einer Klasse	verlustfrei	keine Informationsverluste
Löschen einer Klasse	nicht verlustfrei	Instanzen werden gelöscht bzw. zu Oberklassentyp konvertiert, dabei Verlust aller zur löschenden Klasse gehörender Attribute
Hinzufügen eines Attributs	verlustfrei	keine Informationsverluste
Hinzufügen einer Relation	verlustfrei	keine Informationsverluste
Löschen eines Attributs	nicht verlustfrei	die Werte aller Instanzen des betroffenen Attributs werden gelöscht
Löschen einer Relation	verlustfrei	keine Verluste an Bauwerksinformationen, nur Löschen von Traversierungspfaden
Namensänderung Attribut	verlustfrei	keine Informationsverluste
Typänderung Attribut	bedingt (nicht) verlustfrei	im Falle der Konvertierbarkeit der Werte keine Verluste
Hinzufügen einer Facette	verlustfrei	keine Informationsverluste
Typänderung Facet	bedingt verlustfrei	im Falle der Konvertierbarkeit der Werte keine Verluste
Namensänderung Relation	verlustfrei	keine Informationsverluste
Namensänderung Rolle in Relation	verlustfrei	keine Informationsverluste
Hinzufügen Superklasse	verlustfrei	keine Informationsverluste
Entfernen Superklasse	nicht verlustfrei	die Werte aller ererbten Attribute sämtlicher Instanzen werden gelöscht
Zuweisen einer Klasse zu Package	verlustfrei	keine Informationsverluste
Änderung des Namens einer Package	verlustfrei	keine Informationsverluste
Zuweisung eines neuen Parent-Packages zu Package	verlustfrei	keine Informationsverluste
Löschen Package	bedingt verlustfrei	nur dann verlustfrei, wenn mit Package keine Klassen gelöscht werden

Tabelle 1: Auswirkungen von Schemamodifikationen auf Instanzen

Die folgende Abbildung zeigt eine Taxonomie von möglichen Modifikationen am Bauwerksmodell durch Operationen der Metaebene. Dabei kann zuerst zwischen Modifikationen von Klassendefinitionen und Modifikationen an der hierarchischen Anordnung der Klassen unterschieden werden. Dabei sind nicht–kursive Operationen im Modellverwaltungssystem realisiert, kursive Operationen sind gegenwärtig nicht implementiert.

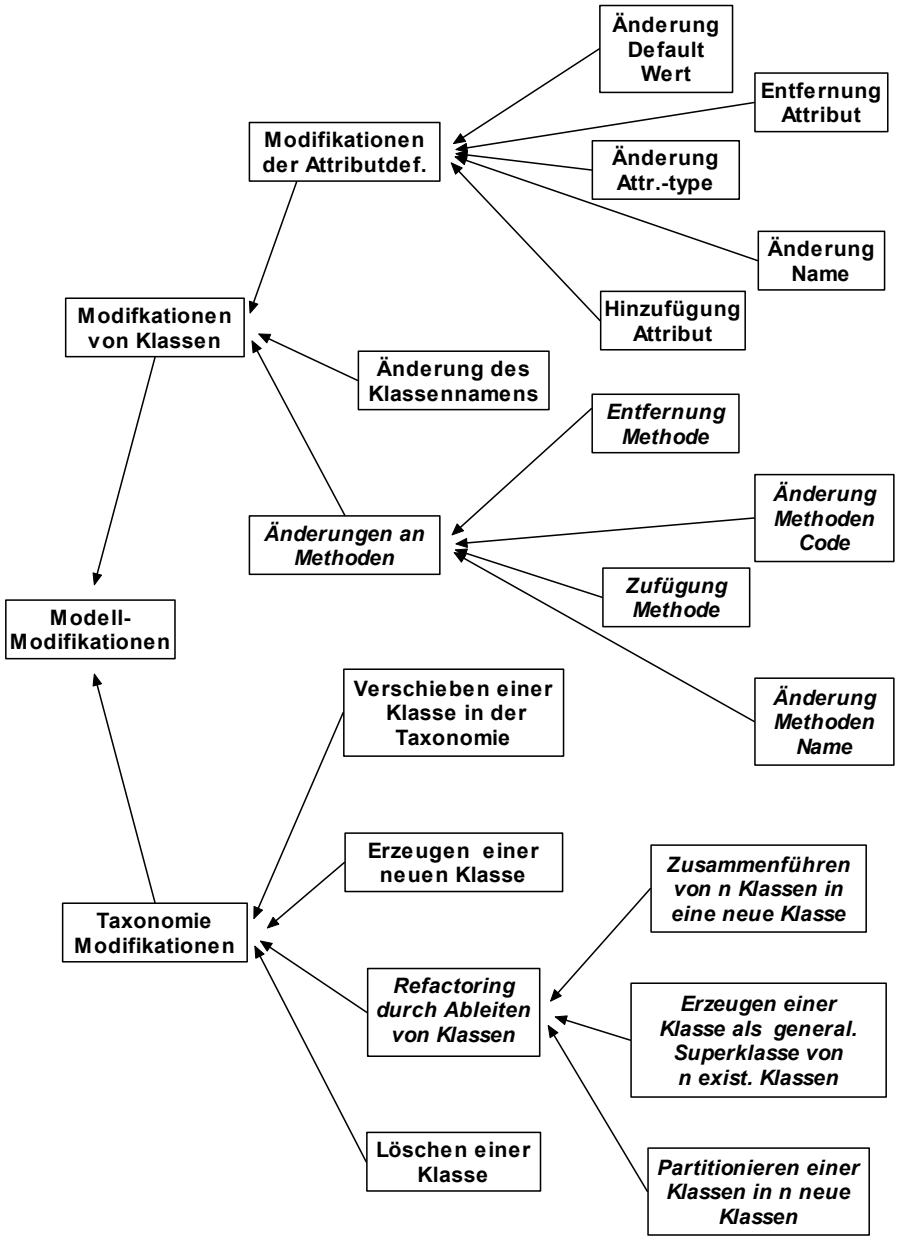

Abbildung 2: Taxonomie von Modifikationen an Domänenmodellen

Die Operationen der Metaebene können additiv, subtraktiv oder sowohl additiv und subtraktiv sein. Additive Änderungen fügen neue Komponenten in das Bauwerksmodell ein, ohne existierende Informationen zu ändern. Subtraktive Modifikationen entfernen dagegen Modellinformationen aus den Bauwerksmodellen. Bestimmte Modifikationen können aber auch

sowohl additiv als auch subtraktiv sein, zu diesen gehören Modifikationen der Taxonomie als auch Namenswechsel. Eine Teilmenge der gleichzeitig additiven und subtraktiven Modifikationen lässt keine Einwirkungen an der äußeren Schnittstelle des Bauwerksmodells zurück, zu diesen gehören beispielsweise Änderungen von Defaultwerten.

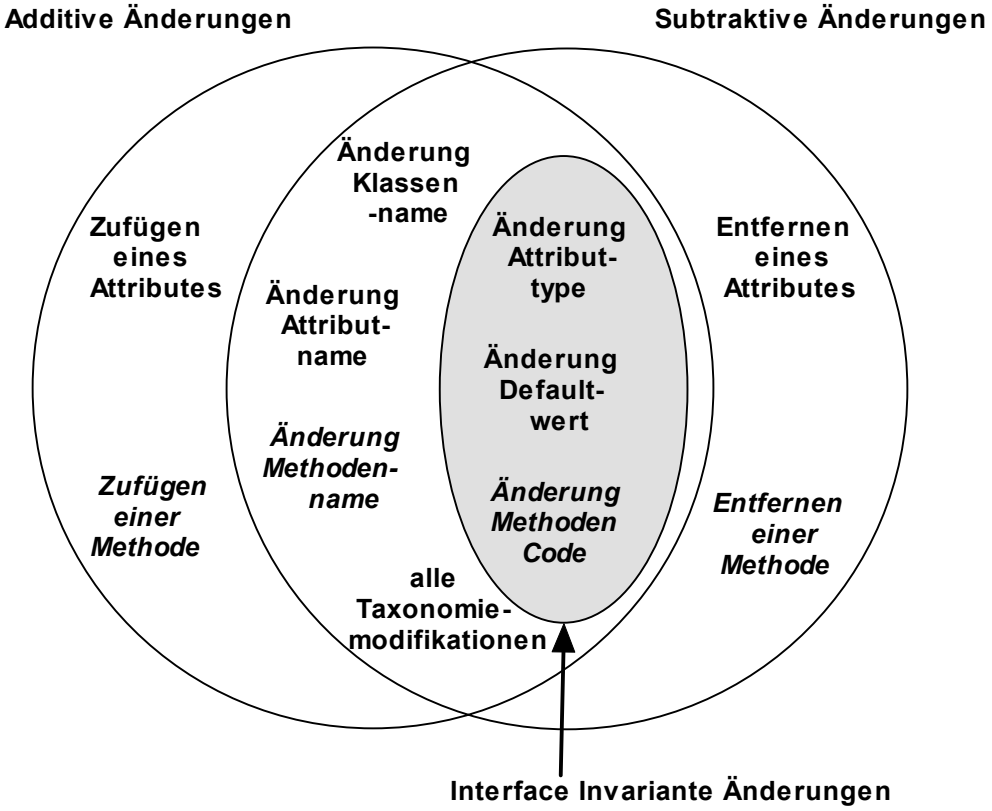

Abbildung 3: Kategorien von Modifikationen von Domänenmodellen

Diese Kategorien von Modifikationen haben Einfluss auf die Anwendbarkeit von Abfragen und (Alt–)Applikationen mit vorgegebenen bzw. statischen Bauwerksmodellen. Subtraktive Änderungen sind rückwärtskompatibel, d.h. Anfragen oder Applikationen, die auf dem Domänenmodell nach der Modifikation ausgeführt werden können, wären auch mit dem Domänenmodell vor der Änderung lauffähig. Additive Modifikationen sind vorwärtskompatibel. Dadurch können Applikationen, die vor der Änderung implementiert wurden oder das entsprechende Domänenmodell erwarten, auch nach einer additiven Modifikation genutzt werden.

Bei Modifikationen die sowohl additiv als auch subtraktiv sind, muss zwischen schnittstelleninvarianten und nicht schnittstelleninvarianten Modifikationen unterschieden werden. Schnittstelleninvariante Änderungen sind sowohl vorwärts– als auch rückwärtskompatibel. Diejenigen Modifikationen, die dieses Kriterium nicht erfüllen, sind weder vorwärts– noch rückwärtskompatibel.

4.2 Entwurf und Realisierung von Modellverwaltungssystemen (MVS)

4.2.1 Funktionalitäten von Modellverwaltungssystemen

Digitale Bauwerksmodelle werden zweckmäßigerweise in speziellen Modellverwaltungssystemen (teilweise auch Dynamic Modeler genannt) gespeichert und durch deren Hilfe für Interaktionen mit Anwendern und Fachapplikationen bereitgestellt. Modellverwaltungssysteme dienen daher als Basis der Prozessintegration, indem sie den beteiligten Planern die benötigten adäquaten Informationsmengen während der Planungstätigkeit zur Verfügung stellen. Folglich sind speziell entworfene Modellverwaltungssysteme, die eine Reihe erweiterter Funktionalitäten besitzen, eine geeignete Basis für die Unterstützung der Kooperation im Bauwerksentwurf nach dem CSCW–Prinzipen des 'Workgroup Computings' und der 'Gemeinsamen Informationsräume'.

Durch den begründeten Wunsch der Bereitstellung einer laufzeitdynamischen Bauwerksmodellierung (siehe Kap. 3) müssen durch das MVS einerseits Taxonomien des Begriffsapparats des jeweiligen Fachgebietes der Beteiligten am Revitalisierungsvorhaben und andererseits die Informationen des konkreten Revitalisierungsvorhabens verwaltet werden.

Zum möglichen Einsatz in den genannten Aufgabenfeldern müssen Modellverwaltungssysteme die folgenden Anforderungen vollständig (oder bei einigen Punkten zumindest ansatzweise) erfüllen [Hauschild 02] [Hauschild 01] [Steinmann 97a] [Eastman 97]:

- Dynamische Modellierung nach dem objektorientierten Paradigma: Bereitstellung erweiterbarer und bedingt modifizierbarer Bauwerksmodelle. Die Modellierung des Diskursbereichs soll dabei auf dem üblichen "Repertoire“ deskriptiver objektorientierter Modelle basieren, dieses umfasst:
    - Klassen mit (facettierten) Attributen,
    - deren Instanzen,
    - Generalisierungsstrukturen, wobei im allgemeinen Einfachvererbung als hinreichend angesehen werden kann,
    - Aggregations– und Assoziationsstrukturen und
    - Kategorien (Packages) zur Strukturierung komplexer Taxonomien
- Informales Wissen: Neben den formalisierten Informationen in den Domänenmodellen und den Instanzen von deren Klassen müssen auch schwer formalisierbare Beschreibungsanteile gespeichert und im Planungskontext bereitgestellt werden. Dazu gehören multimediale Informationen, CAD–Zeichnungsformate, Textverarbeitungsdokumente, Spreadsheets, u.ä.. Diese Informationen können sowohl an Klassen als auch an Objekten gespeichert werden, um sowohl allgemeine Informationen zu Begrifflichkeiten des Fachgebiets, wie Vorschriften,

Gesetzlichkeiten, Kataloge, etc., als auch Informationen zum konkreten Revitalisierungsobjekt, wie beispielsweise Schadensdokumentationen, verwalten zu können.

- Persistente gemeinsame Speicherung von Bauwerksmodell und Bauwerksinformationen über aktuelle Entwurfsprojekte in einem Modellverwaltungskern: Die häufig praktizierte Trennung von Modell und Projektdaten ist im Kontext dynamischer Modelliersysteme kaum als adäquater Ansatz anzusehen, da Erweiterungen oder Modifikationen an den Domänenmodellen sofort auf die Instanzen übertragen werden müssen, bzw. bestimmte Modifikationen nach der Erzeugung von Instanzen nicht mehr sinnvoll sind. Die in bestimmten Arbeiten vertretene Ansicht, dass die persistente Speicherung der Informationen nicht in Binärformaten oder DBMS erfolgen sollte und statt dessen ASCII–Formate verwendet werden sollten [Steinmann 97a], wird vom Autor nicht geteilt. Dies führt zwangsläufig zu schlechten Laufzeitverhalten; der eigentlichen Intention hinter dieser Ansicht kann ebenso gut durch geeignete Exportformate Rechnung getragen werden. Allerdings ist zur persistenten Speicherung die Funktionalität von konventionellen DBMS nicht ausreichend, ein MVS kann aber serverseitig durchaus auf die Dienste von entsprechenden DBMS–Produkten oder vorgefertigten Frameworks zurückgreifen, die teilweise auch Exportfunktionalitäten mitbringen.
- Verteiltes, nicht proprietäres MVS–Interface: Bereitstellung der im MVS verwalteten Klassen und Objekte an die darauf aufsetzenden Applikationen über eine verteilt nutzbare, weitgehend programmiersprach-unabhängige, in heterogenen Umgebungen anwendbare Schnittstelle, die auf Nutzung anerkannter Normen für Middleware beruht. Über diese Schnittstelle sollten alle Interaktionen mit Fachapplikationen ausgeführt werden. Ferner ist vorrangig für Administrationszwecke auch eine grafische Nutzerschnittstelle zum Modellverwaltungskern sinnvoll, während alle anderen Nutzerinteraktionen mit den im MVS gespeicherten Datenbeständen über die GUI's von Fachapplikationen ausgeführt werden sollten.
- Eignung für die Unterstützung von Kooperation im Bauwerksentwurf: Möglichkeit des „Andockens“ von Groupware–Applikationen für synchrone und asynchrone Kooperation, dies impliziert unter anderem auch das Vorhandensein einer event–gesteuerten Kommunikation zwischen MVS und Applikationen sowohl nach dem Push– als auch unter Umständen nach dem Pull–Modell. Erforderlich ist ferner eine Integrierbarkeit der CSCW–Unterstützung in der gewohnten Arbeitsumgebung bzw. Fachapplikationen.
- Repositorium–Funktionalität: Bereitstellung aller für den Gesamtplanungsprozess relevanten Entwurfsinformationen in einem gemeinsamen Informationsraum (Repository) für alle Anwender und angebundenen Applikationen, dies impliziert die Notwendigkeit von Navigationsfunktionalitäten in den Informationsbeständen.
- Inkrementelle Updates: Die Übermittlung der Daten von Entwurfsfortschritten und –modifikationen an das MVS ist u.a. zur Unterstützung synchroner Kooperation erforderlich,

ein ausschließliches Update über den Austausch der kompletten Daten einer Fachdisziplin ist nicht akzeptabel. Infolge dessen müssen alle Objekte in MVS und allen Applikationen unique und eineindeutig identifiziert werden können. Dies stellt allerdings bei der Nutzung von Legacy–Applikationen und bestimmten Formaten ein Problem dar.

- Dokumentencharakter: Die Bereitstellung von Funktionalitäten zur Sicherung des Dokumentencharakters und damit der Rechtsverbindlichkeit der Modellinformationen z.B. durch digitale Signaturen ist erforderlich, wenn auf den Austausch abgezeichneter Papierdokumente verzichtet werden soll.
- Autorisierung und Authentisierung: Kooperatives Arbeiten in offenen Netzwerken erfordert grundsätzlich die Einführung von Sicherheitsvorkehrungen. Dies bedeutet hier die Kontrolle des autorisierten Zugriffs auf Informationen durch für kooperative Entwurfsumgebungen geeignete Mechanismen. Daher sind Schnittstellen zur Authentisierung, zur Autorisierung des Zugriffs sowie zur Rechte- und Rollenverwaltung erforderlich (siehe Kap. 5).
- Sicherung der physikalischen Datenintegrität: Synchrones kooperatives Arbeiten erfordert Mechanismen zur Kontrolle von Nebenläufigkeiten und des gemeinsamen Ressourcenzugriffs, die aber eine Eignung für die Mensch–Maschine–Interaktion in Entwurfsprozessen (siehe Kap. 6) besitzen müssen.
- Konsistenzprüfung: Wünschenswert sind Mechanismen zur Prüfung der Konsistenz der Planungsdaten innerhalb eines und unter Umständen zwischen mehreren Domänenmodellen. Werden hierzu constraintbasierte Regeln aufgestellt, können diese auch assistierend bei der Wiederherstellung der Konsistenz wirken.
- Versionierung: Mechanismen zur Versionierung von Modellen und Projektdaten sind erforderlich, dabei sollte sowohl eine zeitorientierte als auch eine variantenorientierte Versionierung möglich sein. Ebenso sollten kooperative Undo– und Redo–Mechanismen durch das MVS unterstützt werden. Allerdings existiert bis zur vollständigen Lösung dieses Fragenkomplex noch Klärungsbedarf in einigen Punkten.
- Abbildbarkeit von Soll– und Ist– Informationen eines Bauwerks: Bei Revitalisierungsvorhaben ist es nötig, neben der üblichen Abbildung des Zielzustandes auch den gegenwärtigen Zustand im Modell abzubilden. Dieser Ist–Zustand definiert einen ersten, teilweise vagen und unvollständigen Initialzustand des zu verwaltenden Bauwerksmodells. Folglich muss eine Modellierungstechnik angewandt werden, die temporal inkonsistente oder unvollständige Informationsbestände verwalten kann. Digitale, dynamische deskriptive Bauwerksmodelle sind grundsätzlich in der Lage, derartige Informationen zu repräsentieren. Letztendlich kann diese Funktionalität durch Nutzung einer variantenorientierten Versionierung oder durch simple Verwaltung paralleler Informationsbestände erfolgen, im letzteren Fall sind die Interaktionsmöglichkeiten naturgemäß eingeschränkt.

- Teilmengenbildung für Instanzen: Identifizierbarkeit von Objekt–Teilmengen (Kennzeichnung von Objekten) nach verschiedenen Kriterien wie Ort, Funktion, Material/Produkt, damit werden Applikationen in die Lage versetzt, nur diejenigen für den aktuellen Arbeitsschritt relevanten Objekte vom MVS abzufragen; Applikationen zum Prozessmanagement nutzen diese Funktionalität zur Zuordnung von Verantwortlichkeiten und Planungsständen zu Instanzmengen.
- Modellierungsaspekt: Modellverwaltungssysteme sollten geeignete Werkzeuge zur Entwicklung von Domänenmodellen bereitstellen, alternativ wären für die Entwicklung großer Modelle auch Schnittstellen zu CASE–Tools eine Lösung.
- Analysefunktionalität: MVS–Kerne müssen Mechanismen zur Modellanalyse und zur Traversierung von Relationsgraphen, wie z.B. zur Analyse von Taxonomien oder Aggregationsstrukturen bereitstellen.
- Unschärfe–Abbildung: Entwurfsprozesse wie auch bestimmte Entwurfsgegenstände beinhalten verschiedene Formen von Unschärfe und Vagheit. Neben der Modellierung über statistische/stochastische Größen hat sich hier die Nutzung von speziellen Attributfacetten sowie von Defaultwerten für den Umgang mit noch nicht vorliegenden Informationen bei der Unterstützung von Top–Down–Entwurfsphasen bewährt [Steinmann 97a].
- Langlebige Speicherungsform: Bauwerke haben einen sehr langen Lebenszyklus und lange Nutzungsphasen zwischen Planungstätigkeiten in Vorbereitung einer Umnutzung oder eines Umbaus. Damit sind für den Datenexport im Gegensatz zur hochfrequenten persistenten Datenspeicherung keine Binärdaten oder Applikationsformate wünschenswert. Hier sollten ASCII–Files mit einer einfach interpretierbaren Struktur verwendet werden, zum Beispiel ist die Nutzung von XML–Files denkbar.

Eine Anzahl von weiteren Anforderungen aus Sicht der Unterstützung des architektonischen Entwurfs werden in [Steinmann 97a] formuliert.

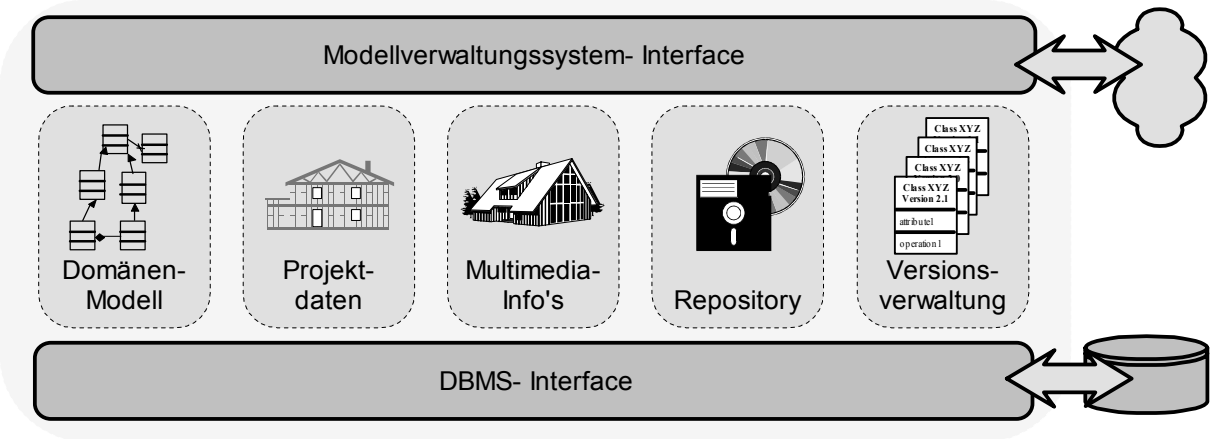

Abbildung 4: Notwendige Komponenten eines Modellverwaltungssystems

Klassische Ansätze des Managements von Bauwerksdaten wie die direkte Nutzung von Datenbanken erfüllen diese Anforderungen nur bedingt. Demzufolge müssen die genannten Funktionalitäten durch ein Modellverwaltungssystem bereitgestellt werden, was speziell für derartige Zwecke zu entwerfen und umzusetzen ist. Der für die Bereitstellung der genannten Basisfunktionalitäten verantwortliche Modellverwaltungskern hat daher die in Abb. 4 veranschaulichte Grundstruktur.

4.2.2 Alternative Implementierungsansätze für Modellverwaltungssysteme

Da durch den direkten Einsatz konventioneller Datenbankverwaltungssysteme ohne weitere Vorkehrungen nicht die Anforderungen an eine Verwaltungskomponente für Bauwerksmodelle erfüllt werden und kommerzielle Bibliotheken zur Produktmodellverwaltung neben anderen fehlenden wünschenswerten Features keine oder nur eingeschränkte dynamische Modellierfunktionalitäten bieten, müssen zur Erfüllung der oben genannten Anforderungen Entwicklungsarbeiten zur Anpassung und Erweiterung vorhandener Systeme oder eine Neuimplementierung erfolgen.

Für die Realisierung von Modellverwaltungssystemen, die den genannten Anforderungen genügen, gibt es verschiedene Ansätze, zu diesen gehören:

- Es ist eine Eigenimplementierung aller Modellverwaltungsfeatures möglich, bei der sämtliche relevanten Eigenschaften des objektorientierten Paradigmas vollständig durch die Applikation bereitgestellt werden,
- die Realisierung kann für die Funktionalitäten der dynamischen Modellverwaltung relevante Eigenschaften objektorientierter Programmiersprachen oder Datenbankverwaltungssysteme mit (quasi–) dynamischen Typsystemen ausnutzen. Hierbei muss zwischen einer Realisierung auf Basis von Sprachen mit voller Dynamik wie Smalltalk–80 und Sprachen mit Semidynamik wie Java unterschieden werden,
- die Implementierung könnte auch auf geeigneten Tools, Frameworks oder Bibliotheken basieren.

Zum letztgenannten Punkt wurden empirische Untersuchungen durchgeführt, teilweise konnte auch auf Ergebnisse ähnlicher Recherchen im Rahmen des FlexOb–Projekts zurückgegriffen werden. Grundsätzlich ist der Einsatz von bestimmten, zur Abbildung von Objektsystemen geeigneten KI–Tools denkbar, jedoch sind diese äußerst kostenintensiv, besitzen diverse nicht benötigte Funktionalitäten, die im vorliegenden Kontext eher als Overhead zu betrachten sind und werden üblicherweise nach anderen Kriterien optimiert.

In Bezug auf Frameworks oder Bibliotheken, die entweder auf die Verwaltung von Produktmodellen oder CAD–Modellierkernen ausgerichtet sind, ist zu konstatieren, dass kein bekanntes System ohne gravierende Schwächen bei einem oder mehreren wesentlichen Punkten

der oben genannten Anforderungen bekannt ist. Auch diese Produkte sind häufig äußerst kostenintensiv.

Ferner existieren Applikationen (z.B. ALFABET), die für die Modellierung von Geschäftsprozessen entworfen wurden und daher Ansätze dynamischer Modellierung aufweisen. Deren Programmierschnittstellen sind jedoch nur bedingt für den Einsatz in verteilten Systemen geeignet, vor allem sind diese nicht für das Management von sechs– bis achtstelligen Objektmengen ausgelegt.

Daher sollen im folgenden der erste und beide Varianten des zweiten Punktes näher betrachtet werden.

4.2.3 Realisierung von MVS auf Basis einer vollständigen Eigenimplementierung der objektorientierten Modellierfunktionalität

Der erstgenannte Implementierungsansatz basiert auf einer vollständigen Eigenimplementierung der Metaebenenfunktionalitäten durch die Module des Modellverwaltungskerns. Daher werden sämtliche Informationen über Modellklassen, Instanzen mit ihren Attributen und Relationen usw. in geeigneten Datenstrukturen des Modellverwaltungskerns abgebildet. Ebenso müssen alle erforderlichen Operationen der definierten Metaebene durch die eigenen Module implementiert werden.

Vorteilhaft an diesem Weg ist, dass die Auswahl der Implementierungssprache sich ausschließlich an Erfordernissen des Software Engineerings und an Randbedingungen wie Bibliotheksverfügbarkeit und dem Grad der Beherrschung von Programmiersprachen orientieren muss, dass aber keine weiteren Anforderungen an das Typsystem der Programmiersprache gestellt werden müssen. Insbesondere ist eine Implementierung auf Basis von Sprachen mit statischen Typsystem wie z.B. C++ möglich.

Erfahrungsgemäß führt dieser Weg zu Modellverwaltungskernen, die ein vergleichsweise gutes Laufzeitverhalten bei Schemaänderungen und –erweiterungen aufweisen. Jedoch ist die Optimierung des Zugriffs auf Instanzen sehr komplex und aufwendig, ferner muss auch der Gesamt–Implementierungsaufwand aufgrund der ernormen strukturellen und algorithmischen Komplexität als hoch eingeschätzt werden. Gemäß der Erfahrungen der zurückliegenden Projekte tendieren derartige MVS zu einem Verhalten, bei dem sich infolge dessen die Interaktion mit den verwalteten Instanzen als Bottleneck erweisen, sobald größere Objektmengen verwaltet werden. Die Beschleunigung dieser Operationen ist technisch möglich, aber mit der Einführung hochkomplexer und implementierungsintensiver Datenstrukturen verbunden, die auf einer Anzahl verschiedener Traversierungspfade den Zugriff auf die Speicherstelle von Objekt– und Attributinformationen mit wenigen Indirektionen erlaubt. Durch die Notwendigkeit der Reorganisiererbarkeit der Datenstruktur nach Schemaänderungen wird dieses Vorhaben noch erschwert. Dieser Nachteil im Laufzeitverhalten relativiert die Vorteilte des Ansatzes stark, da

die Häufigkeit von Instanzenzugriffen um mehrere Größenordnungen über der von Schemaänderungen liegt.

Der hier beschriebene Ansatz wurde von FlexOb–Modelliersystem des PREPLAN–Projektes gewählt. Mittels C++ wurde ein Modellierkern realisiert, der die Funktionalität einer frühen Version der AKO–Schnittstelle bereitstellt. Mit einem Implementierungsaufwand von circa einem Mannjahr konnte auf Basis des beschriebenen Ansatzes eine funktionsfähige Implementierung erstellt werden, die den Vorgängerapplikationen auf Lisp– und KappaPC–Basis deutlich überlegen war, aber die genannten Defizite aufwies.

4.2.4 Realisierung von MVS auf Basis von Sprachen mit volldynamischem Typsystem

Werden bei der Implementierung von Modellverwaltungskernen geeignete objektorientierte Programmiersprachen mit nichtstatischen Typsystemen genutzt, können diese Eigenschaften bei der Implementierung direkt genutzt werden. Voraussetzung dafür ist eine mindestens ansatzweise vorhandene Metaebene in der Programmiersprache und die Möglichkeit des Einfügens von neuen Klassen in das Typsystem der Sprache zur Laufzeit. Sprachmittel zur Abfrage von Typinformationen zur Laufzeit wie RTTI (run time type information) von C++ allein sind dafür nicht hinreichend. Damit eignen sich Programmiersprachen, deren Kompilate als fest gebundene Executables ausgeführt werden, nicht für eine derartige Unterstützung der Entwicklung von Modellierkernen, stattdessen sind Sprachen, die als Ausführungsumgebungen Virtuelle Maschinen benutzen, prinzipiell gut geeignet. Natürlich muss über die Eignung jeder konkreten Programmiersprache im Einzelfall nach Prüfung der Metaebene und der Laufzeitumgebung entschieden werden. Der Grad der möglichen Nutzung von Mitteln der Sprache für die Abbildung von Bauwerksmodellen variiert mit der Ausstattung der Metaebene der genutzten Sprache.

Objektorientierte Programmiersprachen mit vollständig dynamischen Typsystemen wie Smalltalk-80 besitzen eine vollständige Metaebene, die Funktionalitäten für Reflection[3], Introspection[4], Intercession[5] und damit auch zur Behandlung neuer Klassen enthalten.

---

[3] Unter Reflection wird die Fähigkeit eines Programms verstanden, Informationen, die den Zustand des Programms zur Laufzeit repräsentieren, wie Daten des Programms zu manipulieren. Reflection besitzt die beiden Teilaspekte Introspection (Selbstreflektion) und Intercession (Selbstmodifikation).

[4] Introspection ist die Fähigkeit eines Programms, seinen eigenen Zustand zu überwachen und daher über seinen Status zu schlussfolgern. Strukturelle Reflection erfordert die Fähigkeit einer Sprache, Informationen über das momentan ausgeführte Programm und sämtliche (abstrakten) Datentypen des Programms oder der Ausführungsumgebungen in den Ausdrucksmitteln der Sprache selbst vergegenständlicht bereitzustellen.

[5] Unter Intercession wird die Fähigkeit eines Programms verstanden, seinen eigenen Ausführungsstatus zu modifizieren oder seine eigene Interpretation oder Bedeutung zu ändern.

Smalltalk–80 (ST–80) ist zu wesentlichen Teilen selbst in Smalltalk–80 implementiert, nur ein sehr geringer Teil des Gesamtsystems machen 'Primitive Methoden' aus, die in anderen Sprachen (meist C, möglicherweise auch Makroassembler) realisiert wurden. Üblicherweise liegt der Sourcecode des ST–80–Systems in diesem für Anwendungsprogrammierer vor. ST–80 ist nicht nur eine Programmiersprache im klassischen Sinn, sondern gleichzeitig auch Entwicklungs– und Ausführungsumgebung, ferner werden auch bestimmte Betriebssystemdienste bereitgestellt. Im Unterschied zu anderen Sprachen werden keine einzelnen Executables erzeugt. Neu erstellte Klassen werden mit in das Smalltalk–Image aufgenommen, das üblicherweise auch die ST–80–Klassenbibliothek sowie die Entwicklungs– und Ausführungsumgebung beinhaltet. Nachteilig an dieser Lösung ist allerdings, das spezielle Vorkehrungen für eine konflikt– und kollisionsfreie Softwareentwicklung im Team getroffen werden müssen.

Klassen werden in Smalltalk auch als Objekte aufgefasst, deren Verhalten und Eigenschaften von den Klassen der Metaebene[6] beschrieben werden. Somit ist es technisch auch möglich, durch Modifikationen dieser Klassen das Verhalten des gesamten Smalltalk–Systems zu ändern. Dies ist allerdings für Modellverwaltungszwecke nicht nötig, da die Metaebene das objektorientierte Paradigma korrekt abbildet.

Smalltalk–80–Systeme enthalten in ihrem Image bei vielen gängigen Implementierungen einen in der Sprache selbst realisierten Compiler. Durch diesen Übersetzer kompilierter Code steht nach dessen Verarbeitung sofort im Image zur Ausführung und Nutzung bereit. Darauf baut das Konzept zur Nutzung einer Smalltalk–Umgebung als Basis eines Modellverwaltungskerns. Bauwerksmodellklassen werden als gewöhnliche Klassen des Smalltalk–Systems mit Sourcecode und kompilierten Bytecodes verwaltetet. Methodenaufrufe der AKO–Schnittstelle werden in solche des Smalltalk–Systems transformiert. Hat der Methodenaufruf eine bauwerksmodell-modifizierende Semantik, wird der entsprechende Smalltalk–Sourcecode geändert und der eingebaute Compiler aktiviert. Modifikationen an Instanzen werden weitgehend durch das ST–80-System übernommen. AKO–Aufrufe zur Erzeugung von Instanzen oder zur Modifikation dieser resultieren entsprechend.

Es sei hier darauf hingewiesen, dass bei Nutzung einer geeigneten objektorientierten Sprache als Implementierungsbasis theoretisch keine Laufzeitverbesserung im Vergleich zur oben betrachteten Variante erzielbar sind, weil mit der Nutzung derartiger Systeme ein gewisser Overhead mit in Kauf genommen werden muss. In der Praxis darf jedoch nicht ignoriert werden, dass derartige Systeme äußerst effektive Verfahren zum Zugriff auf Objekte besitzen, da sie über Jahre und teilweise Jahrzehnte mit hohem Aufwand entwickelt und optimiert worden sind und

[6] Die Metaebene von Smalltalk-80 besteht aus den Klassen *Object*, *Behaviour*, *ClassDescription*, *Class* und *Metaclass*. Alle genannten Klassen sind Instanzen von *Metaclass*. Da dies auch für *Metaclass class* gilt, wird die sonst unendliche Instanziierung gestoppt. Das wesentliche Verhalten von Klassen und Metaklassen wird durch *Class* und *Metaclass* definiert, *Behaviour* spezifiziert minimales Verhalten wie die physische Repräsentation in der Virtuellen Maschine, *Object* trifft Festlegungen über das Verhalten von Objekten.

dass dieser hohe Aufwand für die meisten Projekte für Bauwerksmodelliersysteme nicht geleistet werden kann.

Ein weiterer Vorteil der Nutzung von Smalltalk ist die Gewährleistung von Persistenz für Modell und Projektinformationen im Smalltalk–Image. Dieser Mechanismus kann bei überschaubaren Objektzahlen durchaus sinnvoll genutzt werden. Ferner existieren ausgefeilte Mechanismen zur Serialisierung und zum Streaming von Objekten (BOSS–Streams), die bei mittleren Objektmengen gut geeignet sind. Für sehr komplexe Modelle oder sehr hohe Objektzahlen sollten DBMS verwendet werden. Je nach ST–80– und DBMS–Implementierung werden diese direkt ins Smalltalk–System integriert oder über Schnittstellen extern angesprochen. Hier muss allerdings konstatiert werden, dass derartige Lösungen entweder sehr teuer sind und eine kommerziell unsichere Zukunft haben (z.B. GemStone/S) oder ein unbefriedigendes Laufzeitverhalten (z.B. ObjectStore–Anbindung) besitzen.

Der Modellverwaltungskern des GroupPlan–Projekts wurde nach dieser Strategie implementiert. Vorrangig die Defizite der vorhandenen Middleware, die Probleme mit Datenbankanbindungen und Unsicherheiten in Bezug auf dauerhafte Sicherung der Wartung und Pflege der Smalltalk-Umgebung durch deren Hersteller motivierten die Suche nach alternativen Implementierungs-plattformen.

Obwohl keine praktischen Erfahrungen des Autors dazu vorliegen, dürfte die beschriebene Implementierungsstrategie auf Umgebungen mit vergleichbaren Funktionalitäten, wie beispielsweise CLOS und LOOPS übertragbar sein.

4.2.5 Realisierung von MVS auf Basis von Sprachen mit semidynamischem Typsystem

Die Herangehensweise der Smalltalk–80–Implementierung lässt sich auch partiell auf Sprachen übertragen, deren Metaebene nicht eine vollständige Abbildung aller Operationen einer objektorientierten Modellierumgebung aufweisen.

Die Sprache Java besitzt ein semidynamisches Typsystem und Ansätze einer Metaebene, die aber nicht die Modelliermächtigkeit der entsprechenden Module von Smalltalk–80 oder CLOS bietet. Konkret bietet Java nur Möglichkeiten der strukturellen Introspection, direkte Mechanismen für Intercession sind nicht integriert. Dies ist auch die Ursache für eine Reihe von Java-Dialekten- oder Erweiterungen wie Kava, ReflectiveJava, MetaJava, REFLEX, etc.. Die Java-Bibliothek besitzt im Package java.lang eine Anzahl von Klassen[7] mit Metaebenen-Funktionalitäten, ein über die Java Core Reflection API hinausgehendes klar abgegrenztes Metamodell existiert nicht. Ebenso ist die transparente Hinzufügung von Klassen zur Laufzeit à la ST–80 nicht möglich,

7 Die Klassen *Class*, *Classloader*, *Object* und *Package* im Package java.lang besitzen Methoden mit einer reflektiven Semantik. Das Package java.lang.reflect beinhaltet die Klassen *AccessibleObject*, *Array*, *Constructor*, *Field*, *Method*, *Modifier*, *Proxy* (ab JDK–Version 1.3) und *ReflectPermission* und stellt Mechanismen zur strukturellen Introspection bereit.

sämtliche Klassen, auf die mit regulären Bezeichnern zugegriffen werden soll, werden durch die Java–Umgebung zum Zeitpunkt des Ladens der Java–Applikation bzw. des Applets mitgeladen.

Trotzdem kann Java als Basis der Implementierung von Modellverwaltungskernen unter weitergehender Ausnutzung der Sprachmittel verglichen zum im Absatz 4.2.3 beschrieben Ansatz genutzt werden. Durch den Java–Classloader zur Laufzeit nachgeladene Klassen können unter Nutzung des Reflection–API benutzt werden, d.h. ein Zugriff auf beispielsweise Attribute oder Methoden ist nur indirekt über Instanzen von Class Field und Class Method möglich.

Die Übersetzung von Java–Sourcecode in Java–Bytecode ist zur Laufzeit auf mindestens zwei Wegen möglich. Einerseits kann in einem separaten Thread der reguläre externe Java-Compiler javac gestartet und ausgeführt werden. Andererseits existiert im Package sun.tools.javac ein interner Compiler, der ein wesentlich besseres Laufzeitverhalten aufweist, aber nicht im Kontext einer anderen Java–Laufzeitumgebung als derjenigen von SunSoft verfügbar sein muss. Daher sollten beide Varianten nutzbar sein, im Falle der Nichtverfügbarkeit des internen Compilers muss der externe genutzt werden.

Somit ist es möglich, Bauwerksmodellklassen als Java–Klassen zu repräsentieren, Instanzen der Modellklassen werden als normale Java-Objekte verwaltet. AKO–Methodenaufrufe mit modellmodifizierender Semantik resultieren in der Generierung oder dem Update von Java–Klassensourcecode. Dabei müssen sämtliche direkt und indirekt von einer Änderung betroffenen Klassen modifiziert oder zumindest neu übersetzt werden. Dies trifft auf alle Klassen zu, die direkt auf die modifizierten Klassen Bezug nehmen, wie beispielsweise direkte Subklassen, oder die in einem indirekten Abhängigkeitsverhältnis mit der geänderten Klasse stehen, wie z.B. entferntere Subklassen. Ist die Modellmodifikation abgeschlossen, wird eine neue Version der betroffenen Klassen erzeugt. Auf diese Klassen kann nun durch ein Mapping von AKO–Bezeichnern auf Java–Class Class zugegriffen werden, ebenso können diese instanziiert werden. Die erzeugten Instanzen können ihrerseits wie gewöhnlich in Collections verwaltet werden. AKO–Aufrufe mit Modifikationen an Instanzen des Bauwerksmodells werden durch AKO–ObjectID's den entsprechenden Java-Objekten zugeordnet. Da die Klassen dynamisch zur Laufzeit erzeugt wurden, können Attributänderungen wiederum nicht durch Angabe des Attributbezeichners ausgeführt werden, sondern müssen indirekt über Mittel der Reflection API ausgeführt werden.

Als Fazit kann an dieser Stelle konstatiert werden, das der Aufwand einer Implementierung eines Modellverwaltungskerns in Java höher als in Smalltalk ist, aber glücklicherweise deutlich geringer verglichen zum C++–Ansatz aus Absatz 5.4.1 ist.

Weiterhin erweisen sich die feingranularen, portablen und nachladbaren Java-Klassenmodule an weiteren Stellen als vorteilhaft, da sie eine gute Basis für diverse Erweiterungen bieten. Fertig kompilierte Bytecodefiles können gut über Netzwerke auf andere Maschinen übertragen werden. Damit ist unter der Voraussetzung des Vorhandenseins entsprechender Frameworks die Unterstützung synchroner Kooperationsphasen durch relaxierte WYSIWIS–Modell-

präsentationen denkbar. Ferner eignen sich die Bytecodemodule in Java gut für eine Generierung von Methoden in dynamisch erzeugten Modellklassen. Methoden sind zwar in den Klassen zur Abbildung der Domänensemantik von deskriptiven Bauwerksmodellen nicht erforderlich, sind aber in Hilfsklassen zur Generierung von Views auf diese Informationen hilfreich, da damit eigene Interpretationslösungen von Darstellungsvorschriften (s.u.) eingespart werden können.

Für Java existieren diverse Anbindungen an OODBMS und RDBMS, die über ein gutes Laufzeitverhalten und gute Ausstattung verfügen. Ein Teil dieser Lösungen ist sehr kostengünstig. Ebenso existieren eine Vielzahl von Middlewareumgebungen, die sich gut für den Aufbau komplexerer Verteilter Systeme eignen und weit über den Funktionsumfang der in der Laufzeitumgebung integrierten CORBA–Umgebung hinausgehen.

Abschließend sollen in der folgenden Tabelle die Vor– und Nachteile der besprochenen drei Implementierungsstrategien zusammengefasst werden:

Implementierungsstrategie Kriterium	Vollständige Eigenimplent., siehe 4.2.3	OOPL mit dynamischem Typsystem, siehe 4.2.4	OOPL mit semidynam. Typsystem, siehe 4.2.5
Abbildbarkeit objektorientierter Modellelemente	sehr gut	gut, abhängig von OOPL	gut, abhängig von OOPL
Implementierungsaufwand	sehr hoch	mittel	mittel...hoch
Abbildung Metaebene Modell	Klassen der Applikation	Metaebene OOPL	Metaebene OOPL +zusätzl. Methoden
Abbildung Klassen Modell	Instanzen Appl.	Klassen OOPL	Klassen OOPL
Abbildung Instanzen Modell	Instanzen Appl.	Instanzen OOPL	Instanzen OOPL
Rel. Perform. Modellmodifikation	gut	mäßig	mäßig
Rel. Performance Instanzenzugriff	mäßig	gut	gut
Verfügbarkeit geeign. (OO)-DBMS	gut	begrenzt	gut
Verfügbarkeit geeign. Middleware	gut	begrenzt	gut
Beispiel-Modellierkern	FlexOb	GroupPlan	SFB524

Tabelle 2: Implementierungsansätze für Modellverwaltungskerne

4.2.6 Weitere Aspekte der Implementierung von Modellverwaltungskernen

Neben der Auswahl einer geeigneten Implementierungsstrategie für den Modellverwaltungskern sind eine Anzahl weiterer wesentlicher Entscheidungen für die Implementierung eines Modellverwaltungskerns zu treffen. Zu diesen gehören die Auswahl oder das Design einer

geeigneten Schnittstelle, die Auswahl eines Mechanismus für die persistente Speicherung der Modellinformationen und die Entscheidung für eine geeignete Middleware für die Sicherung des entfernten Zugriffs auf die Informationen. Dabei ist zu beachten, dass die genannten Teilaspekte mit der Implementierungsstrategie zusammen im Komplex zu betrachten sind, da sich aus einer Zusammenfügung der besten Varianten der einzelnen Teilaspekte nicht zwangsläufig die günstigste Gesamtstrategie ergibt.

Zur Anbindung der Fachapplikationen an das Modellverwaltungssystem muss eine Schnittstelle bereitgestellt werden. Diese kann neu definiert werden, im Falle der Modellverwaltungskerne des GroupPlan–Projekts und des SFB 524 konnte auf die existierende AKO–Schnittstelle zurückgegriffen werden, die aktualisiert und erweitert wurde. Diese Schnittstelle hat zumindest folgende Ansprüche zu erfüllen:

- Abbildung der Funktionalitäten der Metaebene: Bereitstellung der Funktionalität zur Modellierung der Metaebene sowie zur Interaktion mit den Informationen über Klassen und Instanzen im MVS.
- Verteilung, Interoperabilität: Gewährleistung des entfernten Zugriffs auf das MVS, Bereitstellung einer für mehrere Programmiersprachen nutzbaren Schnittstelle, die auch in heterogenen Umgebungen anwendbar ist, Sicherung der Interaktion mit Objekten, die durch verschiedene Middleware–Technologien verwaltet werden.
- Infrastruktur für Verteilte Systeme: Vorhandensein einer geeigneten Infrastruktur für verteilte Anwendungen mit verschiedenen Modi der Kommunikation und genormten Diensten.
- Transparenz: Akzeptable Transparenzeigenschaften der Middleware.
- Standardisierung: Konformität mit Standards zur Sicherstellung der Unabhängigkeit von einzelnen Middleware–Produkten.
- Akzeptables Verhältnis von Performance zu Implementierungsaufwand und Wartbarkeit der Software.

Für die Modellverwaltungssysteme von GroupPlan und des SFB 524 wurde die Common Object Request Broker Architecture (CORBA) der Object Management Group (OMG) als geeignete Middleware gewählt. Aus heutiger Sicht wären Enterprise Java Beans (EJB) ähnlich gut geeignet, zum Zeitpunkt der Entscheidungsfindung war der Reifegrad dieser Technologie noch nicht als hinreichend fortgeschritten anzusehen. Daher wurde als Schnittstelle eine in CORBA–IDL definierte erweiterte Version der AKO–Schnittstelle genutzt. Bei der Realisierung des MVS erwies es sich als notwendig, einen ORB, der mindestens mit CORBA Version 2.5 konform ist, einzusetzen, um bestimmte CORBAservices nutzen zu können und Möglichkeiten der Einflussnahme auf das Verhalten der Objektadapter (siehe Kap. 5) des ORB's nehmen zu können.

Zur persistenten Datenhaltung muss das MVS entweder auf den Einsatz eines Datenbankmanagementsystems oder auf geeignete Frameworks oder Bibliotheken zurückgreifen. Im Vorfeld der Implementierung ausgeführte Tests haben gezeigt, dass sich Database Management Systems mit weitgehenden Möglichkeiten zur laufzeitdynamischen Schemaänderung aus Sicht des Systemdesigns und des Software Engineerings gut in diesem Kontext eignen würden. Als kommerzielle Systeme sind hier die DBMS–Produkte Gemstone/S und Itasca in Betracht gezogen worden, die jedoch für Einsatz unter Forschungskonditionen nicht zur Verfügung standen. Ausführliche Tests mit einer Demoversion offenbarten ferner schwerwiegende Probleme bezüglich der Stabilität zumindest der Java-Version von Itasca. Somit muss auf Datenbanksysteme mit begrenzten Möglichkeiten der Laufzeit-Schemaevolution zurückgegriffen werden.

Grundsätzlich können zur Realisierung von MVS sowohl relationale als auch objektorientierte DBMS genutzt werden, insofern geeignete Programmierschnittstellen verfügbar sind. Untersuchungen im Vorfeld der Implementation sind praktisch mit DB/2, Poet, ObjectStore und Ozone ausgeführt worden, ferner sind Recherchen über die Eigenschaften von ORACLE, INFORMIX, MySQL, Gemstone/S, Gemstone/J und Versant ausgeführt worden. Die wesentlichen Entscheidungskriterien waren dabei:

- die grundsätzliche Nutzbarkeit des DBMS im genannten Kontext,
- der durchschnittliche Einarbeitungsaufwand,
- die Möglichkeiten der Ausführung und der Auswahl des Typs von Transaktionen,
- die vorgefundenen Mittel zur feingranularen Anbringung von Sperren,
- die Konformität zu existierenden Standards,
- das Laufzeitverhalten und der Ressourcenbedarf,
- die Verträglichkeit mit andern MVS–Komponenten und der Middleware,
- der Preis des Systems sowie die Lizenzierungs– und Wartungsbedingungen,
- die Verfügbarkeit für verschiedene Hardware- und Betriebssystemplattformen.

Während in der ursprünglichen Modellverwaltungsversion von GroupPlan der BOSS–Mechanismus zur binären Objektserialisierung von Smalltalk–80 genutzt wurde, ist nach Abwägung der oben genannten Kriterien für den Einsatz des zum Standard der Object Database Management Group (ODMG) konformen objektorientierten OODBMS Poet[8] entschieden worden.

Objektorientierte Datenbanken für Java stellen die Persistenz für Java-Objekte im allgemeinen durch einen Enhancement–Prozess sicher. Dabei wird entweder der Quelltext der Java–Klasse oder deren Bytecode durch einen Prozessor bearbeitet, der notwendige Änderungen und Ergänzungen zur Verwaltung der Objekte in der Datenbank einfügt. Beide Verfahren haben ihre

[8] Mittlerweile ist durch Poet Software das Poet-DBMS Produkt in FastObjects umbenannt wurden. Im Rahmen der Implementierungsarbeiten wurde die gut ausgebaute Version FastObjects t7 genutzt.

Vor– und Nachteile, der besseren Effizienz des Inplace–/Bytecode–Enhancement steht häufig Kritik über die in bestimmten Fällen nachteilige Black-Box-Vorgehensweise (z.B. bei Bezeichnerkollisionen) dieses Verfahrens gegenüber. Poet nutzt Bytecode–Enhancement, das im Kontext des Modellverwaltungskerns sofort nach Abschluss der Kompilation der Quelltexte durch den internen bzw. u.U. externen Java–Compiler aktiviert wird. Sollte eine modifizierte Version einer Klasse erzeugt worden sein, von der schon Instanzen in der Datenbank existieren, wird eine interne Versionierung der Datenbank durchgeführt. Derselbe Prozess wird ebenfalls für alle direkt und indirekt von einer Änderung betroffenen Klassen mit ausgeführt.

Neben den hier näher besprochenen Aspekten der Implementierung von Modellverwaltungskernen müssen natürlich ebenfalls die üblichen Entwurfsentscheidungen des Software Engineerings (z.B. taktisches Design: Bibliothekseinsatz, Fehlerbehandlungsphilosophie, vereinheitlichte Konzepte zum Ressourcenzugriff, Modularchitekturen, Architektonische Planung, interne Schnittstellen, UI's, Deployment, Optimierungskriterien, etc.) und der Projektplanung (Versionsfolge, Test-Releases, Prototypen, usw.) getroffen werden, auf die hier nicht näher eingegangen werden soll.

4.3 Modell- Präsentation: Konzept & Realisierung

4.3.1 Konzept rollenspezifischer Präsentationstypen

Für eine weitgehende Unterstützung der synchronen Phasen der Bauplanung ist es wichtig, den einzelnen Mitgliedern des Teams entsprechend ihrer Rolle adäquate Präsentationen der im System gespeicherten Informationen über den gemeinsamen Arbeitsgegenstand anzubieten. Für den Bauwerksentwurf bedeutet das, dass beispielsweise ein Architekt während des Gestaltentwurfs im allgemeinen eine dreidimensionale Darstellung bevorzugen wird, während für die Erstellung von Dokumenten für Konstruktionsdetails durch den Tragwerksplaner häufig eine zweidimensionale Darstellung angemessen ist bzw. ein Baukalkulator eine tabellarische Sicht auf bestimmte Informationen wünschen würde.

Aufgrund des dynamischen Charakters der in den Modellverwaltungssystemen von GroupPlan und des SFB 524 gespeicherten Bauwerksmodelle und der daraus folgenden Möglichkeit für Bearbeiter, ihre Taxonomien zu erweitern oder zu modifizieren, kann eine adäquate Präsentation der Modellinformationen nicht auf traditionellem Weg vorimplementiert werden. Auf dem Wege einer statischen Implementierung sind ausschließlich generische Präsentationen wie Ansichten der Taxonomie–Bäume und der dazugehörigen Objekte sowie Darstellungen der Aggregations– und Assoziationenstrukturen des Domänenmodells in textueller, graphenorientierter oder graphischer Form zum Beispiel als UML–Diagramm realisierbar. Diese Modellrepräsentationen allein entsprechen aber in keiner Weise den Erfordernissen einer computerbasierten Unterstützung des Entwurfs von Bauwerken.

Eine weitere Anforderung an die Modellvisualisierung ergibt sich ferner aus dem Einsatz in einer verteilten Umgebung zur Kooperationsunterstützung im Bauwerksentwurf. Die Modelldaten eines Bearbeiters müssen in Phasen entfernter synchroner Kooperation auf der Maschine eines anderen Teammitglieds dargestellt und aktualisiert werden können, um auf Basis der relaxierten WYSIWIS- Darstellung die Koordinierung der folgenden Arbeitsschritte oder die kollaborative Lösung eines aktuellen Entwurfsproblems zu ermöglichen.

Beim gewählten Lösungsansatz wird wiederum davon ausgegangen, dass Erweiterungen des Domänenmodells vorrangig in den Blattbereichen der Taxonomie erfolgen. Der Grund dieser Annahme ist, dass der Anwender gegebenenfalls Know How über bestimmte Teilmengen von Entwurfsgegenstandstypen in die Modellverwaltung einbringen möchte, aber grundsätzlich neue Konzepte nur in sehr seltenen Ausnahmefällen abbilden wird. Bei Anwendern, die sich nicht mit den objektorientierten Modellierungskonzepten beschäftigen möchten, ist es auch durchaus denkbar, dass diese eigene Subklassen nur zur Speicherung von Dokumenten über Spezialfälle ihrer Domäne nutzen, also ausschließlich mediale oder BLOB–Daten in die neuen Subklassen einfügen. Weiterhin ist festzustellen, dass bei der Abbildung von Spezialfällen im Domänenmodell nicht zwingend signifikante Differenzen der Visualisierung im Vergleich zur Basisklasse auftreten müssen, vor allem wenn nur nichtgraphische Attribute modifiziert oder erweitert wurden.

Daher bietet sich eine objektorientierte Lösung der Visualisierungsproblematik an. Es erfolgt eine Trennung der Datenabstraktionen (Model) und der dazugehörigen Präsentation (View). Unter der Daten–Abstraktion werden die für den Bauwerksentwurf relevanten und vom realen Entwurfsgegenstand abstrahierten Eigenschaften verstanden. Unter dem View versteht man hingegen eine Präsentation der Daten mit entsprechenden Interaktionsmöglichkeiten. Zu einer Abstraktion können mehrere Typen von Views existieren. Sind mehrere Views zu einem Zeitpunkt aktiv, müssen nach Zustandsänderungen der Abstraktion alle Views aktualisiert werden.

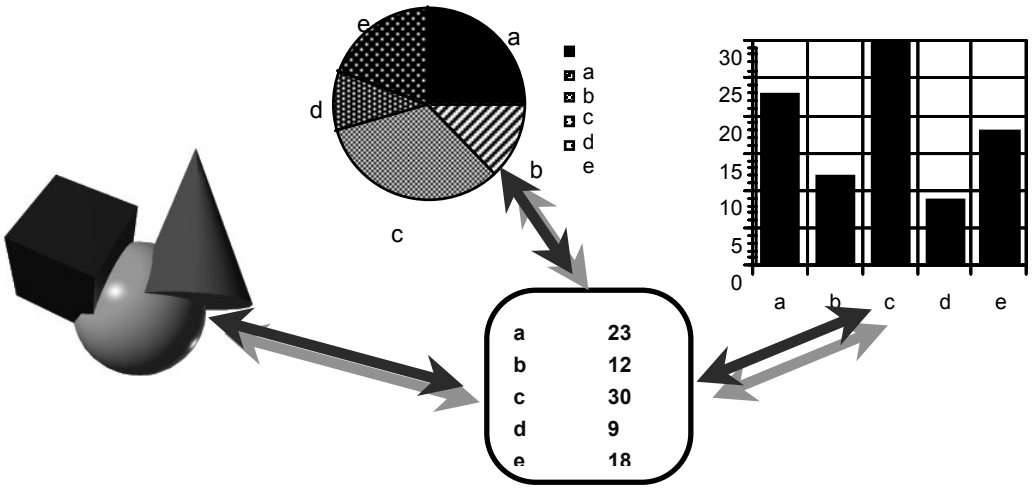

Abbildung 5: Trennung von Informations-Abstraktion und -Präsentation

Diese Präsentations–Mikroarchitektur appliziert das Model–View–Controller(MVC)– Paradigma von Smalltalk–80 [Krasner 88] auf Informationen in laufzeitdynamischen Bauwerksmodelliersystemen. Model und View nehmen die beschriebenen Aufgaben wahr, während das Controller–Objekt für die Auswertung der Nutzerinteraktionen verantwortlich ist.

Betrachtet man diese Vorgehensweise verallgemeinert als Technik zur Konsistenthaltung von entkoppelten Objekten in einer one–to–many–Relation, so spricht man auch vom Observer–Pattern [Gamma 95].

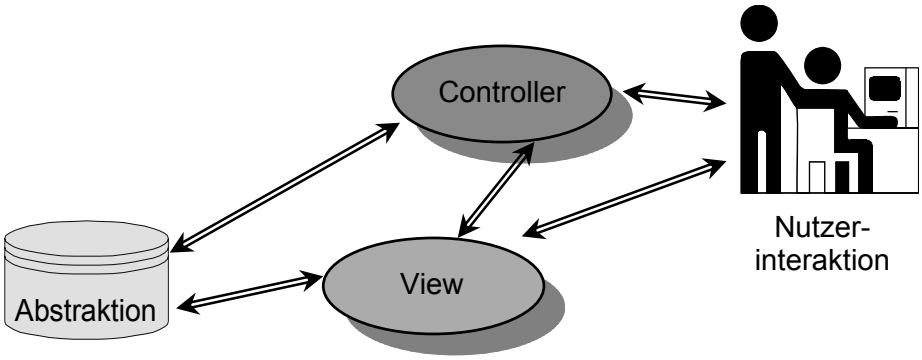

Abbildung 6: Model View Controller (MVC) - Paradigma

Ein weiterer Vorteil dieser Lösung ist die Möglichkeit, View– und Controller–Objekte austauschen oder warten zu können, ohne Modifikationen am Modell oder an der dazugehörigen Metaebene vornehmen zu müssen. Ferner lässt sich eine verteilte Präsentation von Modellinformationen in einer CORBA–basierten Umgebung leicht realisieren, da die Kommunikation von View– und Controller–Objekten mit dem Modell ortstransparent implementiert werden kann. Man spricht dann von ‚Semantic Presentation Split'.

4.3.2 Umsetzung in GroupPlan

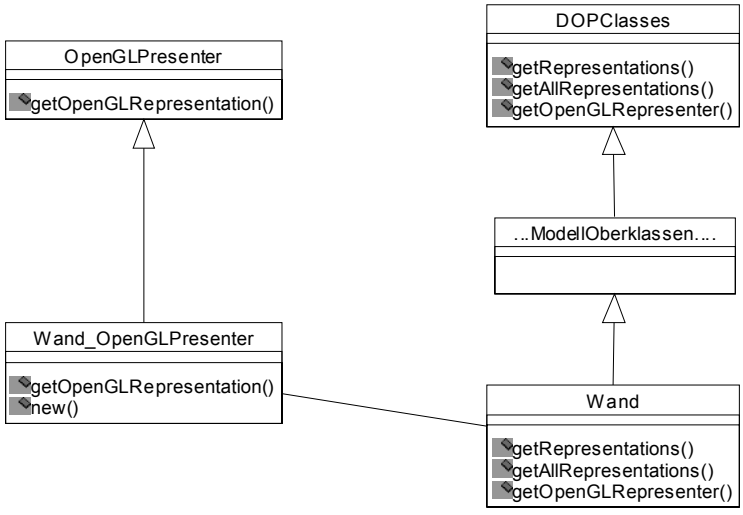

Abbildung 7: Prinzipdarstellung der Methoden für Objektpräsentationen

Jede Modellklasse kann in GroupPlan befragt werden, welche Präsentationstypen für sie anwendbar sind. Dabei kann zwischen direkt für diese Klasse definierten Präsentationen und ererbten Darstellungen unterschieden werden. Bei der Darstellung der Modellinformationen greifen die Präsentationen auf die relevanten Instanzen des Projektes zu. Existiert für eine Klasse dabei keine Darstellungsvorschrift, wird auf eine ererbte Vorschrift zurückgegriffen.

In GroupPlan liefert eine Klasse als Ergebnis der Methode *getRepresentations* eine Liste mit den Symbolen der für diese Klasse definierten Repräsentationen, z.B. (*#Text*, *#OpenGL*, *#Window*), dagegen liefert *getAllRepresentations* eine Symbolliste, die sowohl neu definierte als auch ererbte Darstellungen enthält. In der Wurzelklasse aller Modellklassen in GroupPlan *DOPClasses* sind abstrakte Methoden definiert, welche die für die jeweilige Repräsentation verantwortliche Klasse liefern; diese Methoden werden in den konkreten Subklassen redefiniert. Wird für eine Klasse keine neue Darstellung definiert, wird über 'normale' Erbschaft auf die einer Basisklasse Bezug genommen.

Zum Beispiel könnte als Resultat der Botschaft *Wand>> getOpenGLRepresenter* die Klasse *Wand_OpenGLRepresenter* geliefert werden. Von dieser Klasse können nun Präsentationsobjekte erzeugt werden, dabei wird als Parameter das darzustellende Objekt übergeben. Dazu wird diesem Objekt nun eine Botschaft, die nach dem Schema *get*`Presentationsymbol`*Presentation* benannt ist, gesendet. An die Stelle des Platzhalters `Presentationsymbol` ist hier eine implementierte Darstellungsart einzusetzen. Im Beispielfall der Wand würde die Botschaft *Wand_OpenGLRepresenter>>getOpenGLPresentation: aWand* für die Generierung einer OpenGL-Präsentation gesendet werden. Da der Repräsentationsklasse das darzustellende Objekt übergeben wird, ist eine Rückbezugnahme auf dieses möglich, beispielsweise wäre ein graphischer Element- Pick und anschließende Attributänderung im Modell möglich.

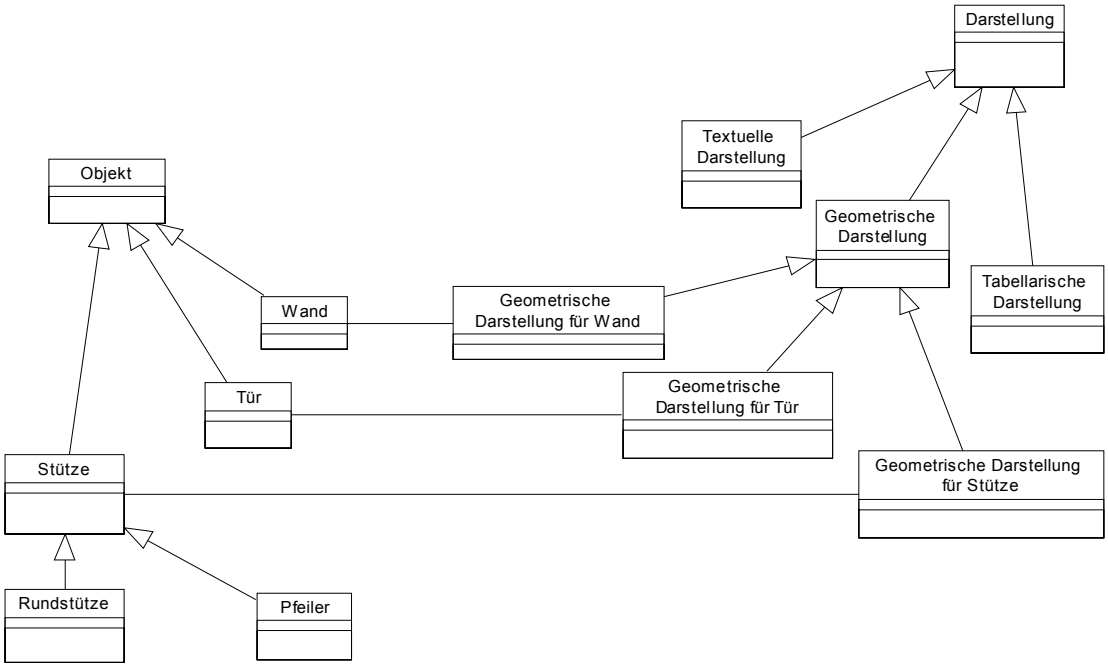

Abbildung 8: Relationen zwischen Modellklassen und Präsentationsklassen

Somit ist es für den Anwender nur gelegentlich erforderlich, selbst neue Darstellungsvorschriften erstellen zu müssen. Für die übrigen Fälle kann dies interaktiv mit Hilfe eines geeigneten Editors erfolgen. Für die dreidimensionale Präsentation kann dies zum Beispiel interaktiv in einem graphischen Editor analog zur CSG–Modellierung erfolgen (siehe Abbildung 9). Die Nutzung anderer Ansätze zur Geometriemodellierung ist ebenso denkbar, jedoch ist bei Anwendung in diesem Kontext mit vergleichsweise höherem Implementierungsaufwand zu rechnen.

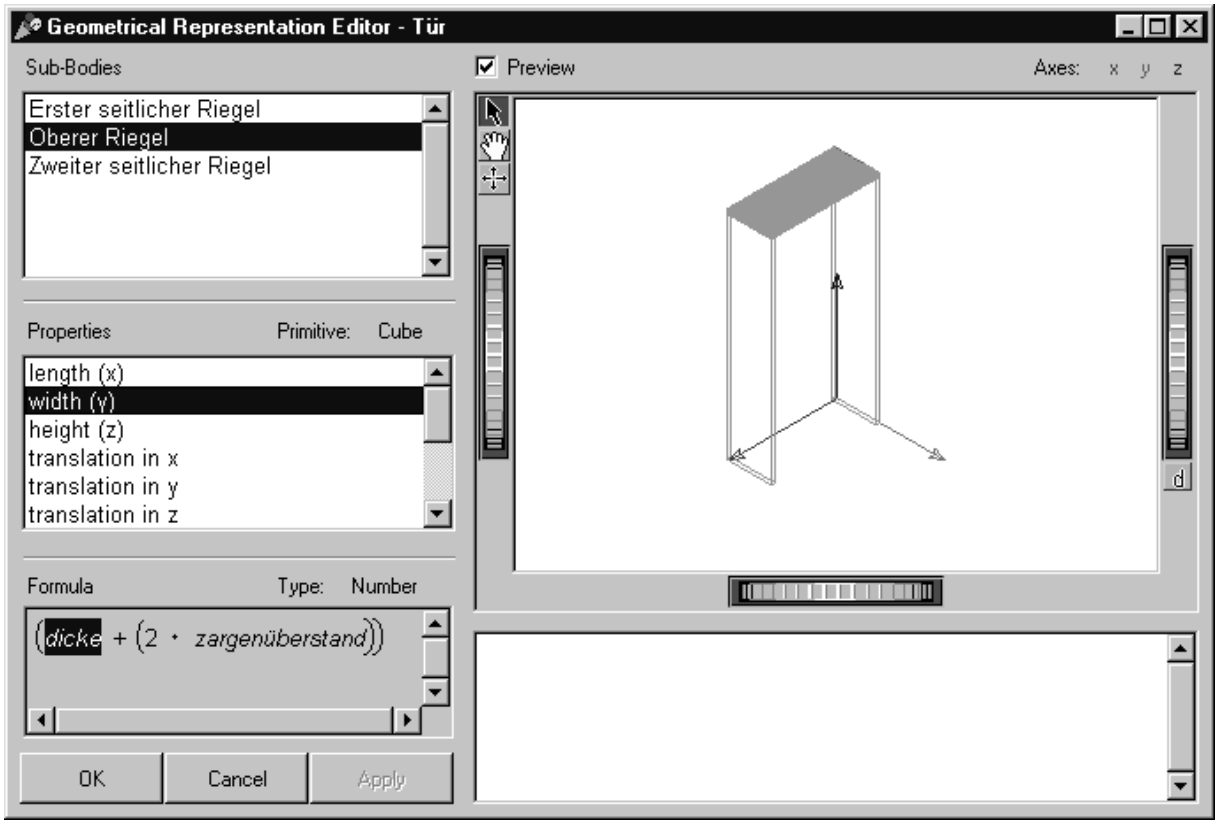

Abbildung 9: Editor für 3D-Darstellungsvorschriften von GroupPlan

Bei der Realisierung der dreidimensionalen Präsentation in GroupPlan wird eine Beschreibung der visuellen Eigenschaften der Modellklassen durch Aufbau aus geometrischen Primitiven analog zur CSG–Modellierung erstellt. Deren Parameter korrelieren direkt oder über numerische Ausdrücke mit Attributen der Modellklassen. Bei der Darstellung der Projektinformationen wird auf die Attributbelegung zurückgegriffen und somit eine korrekte Visualisierung erzielt. Bei der Erstellung oder Bearbeitung der Visualisierungsvorschrift wird auf die in der Klasse verwalteten Defaultwerte Bezug genommen. Es ist möglich, von der visuellen Darstellung interaktiv zu den Objekten und ihren Klassen der Modellwelt zu gelangen. Das ist sowohl für unten beschriebene Groupware–Funktionalitäten als auch für einfache Änderungen von Attributwerten durch den Anwender erforderlich.

Es ist möglich, für aggregierte Objekte Darstellungsvorschriften zu erstellen. Dadurch lassen sich Visualisierungen in verschiedenen Feinheitsgraden erzeugen, indem wahlweise die Darstellungsvorschrift des Aggregatobjekts oder die aller dazugehörigen Komponenten benutzt wird. Das Auflösen von Aggregationen in der Darstellung kann solange rekursiv fortgesetzt werden, bis der gewünschte Detaillierungsgrad erreicht ist. Leider sind bei sehr detaillierten Darstellungen negative Auswirkungen auf das Interaktionsverhalten der Entwurfsumgebung festzustellen, die sich aber weitgehend durch den Einsatz geeigneter Grafikhardware abbauen lassen dürften.

Die graphische Darstellung wurde durch Anbindung einer OpenGL–Bibliothek (Jun) an das unter Smalltalk–80 realisierte GroupPlan–System umgesetzt. Bei der Implementierung wurde bewusst nicht versucht, eine vollständige CAD–Funktionalität zu erzielen, da der Schwerpunkt des Projektinteresses auf den kooperativen Aspekten des Entwerfens liegt. Eine Darstellung der Bauwerksmodellinformationen in einem CAD–System ist eine weitere denkbare Repräsentation.

Die graphische dreidimensionale Darstellung einer Modellklasse wird definiert, indem der Anwender sie analog zur CSG–Modellierung interaktiv aus geometrischen Primitiven (Quadern, Prismen etc.) zusammensetzt. Diese sogenannten Subkörper besitzen Eigenschaften, deren Werte mit den Attributen der Modellklasse direkt oder über einfache numerische Ausdrücke korrelieren.

Bei der Darstellung werden für jede Instanz der Modellklasse die jeweiligen Subkörper erstellt, indem die Darstellungsvorschrift auf die Instanz angewendet wird. Dabei werden die einzelnen Dimensionen der Subkörper entsprechend der Attributbelegung der Instanz gesetzt. Bei der Erstellung und Bearbeitung der Darstellungsvorschrift wird auf die durch die Modellklasse verwalteten Default–Werte der Attribute Bezug genommen.

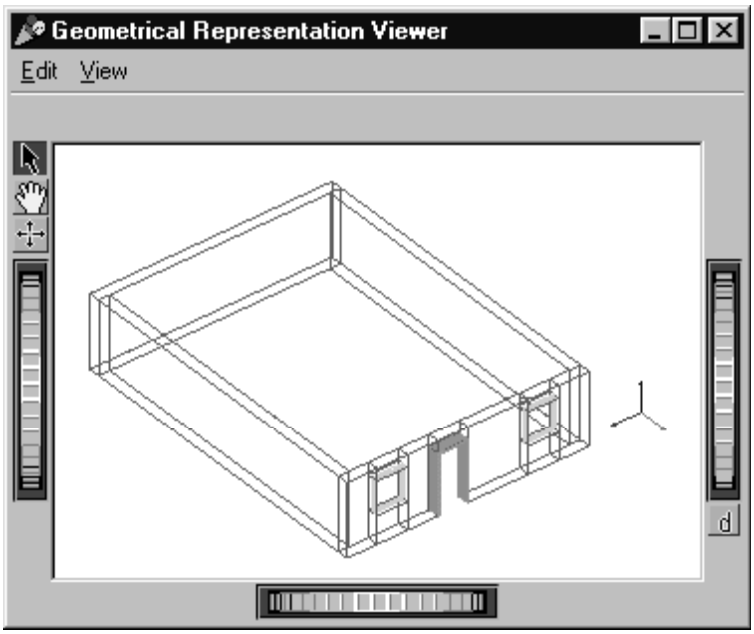

Abbildung 10: Darstellung eines von Bauwerksteilen im Project-Viewer

Die geometrische Darstellung eines bestimmten Projektes ist die Summe aller Subkörper für alle darstellbaren Objekte des Modellbereiches. Von jedem Subkörper kann der Anwender interaktiv zu dem zugrundeliegenden Objekt der Modellwelt gelangen sowie zur verwendeten Darstellungsvorschrift. Aggregationen im Bauwerksmodell können sowohl über eine vereinfachende Darstellung, die für die entsprechende Klasse im Bauwerksmodell definiert ist, als auch über die Darstellung ihrer Bestandteile repräsentiert werden. Damit kann die Granularität der Repräsentation schrittweise verfeinert werden.

Taxonomien und Projekte können in einer Applikation betrachtet und verändert werden. Dem Bearbeiter stehen neben der Baumdarstellung der Vererbungsstruktur auch seine domänenspezifische Darstellungsart zur Verfügung. In beiden Darstellungen können einzelne Objekte selektiert werden, wobei die Auswahl auf die jeweilig andere Darstellung übertragen wird. Darüber hinaus können in der Baumansicht auch Klassen ausgewählt werden (siehe Abbildung 11).

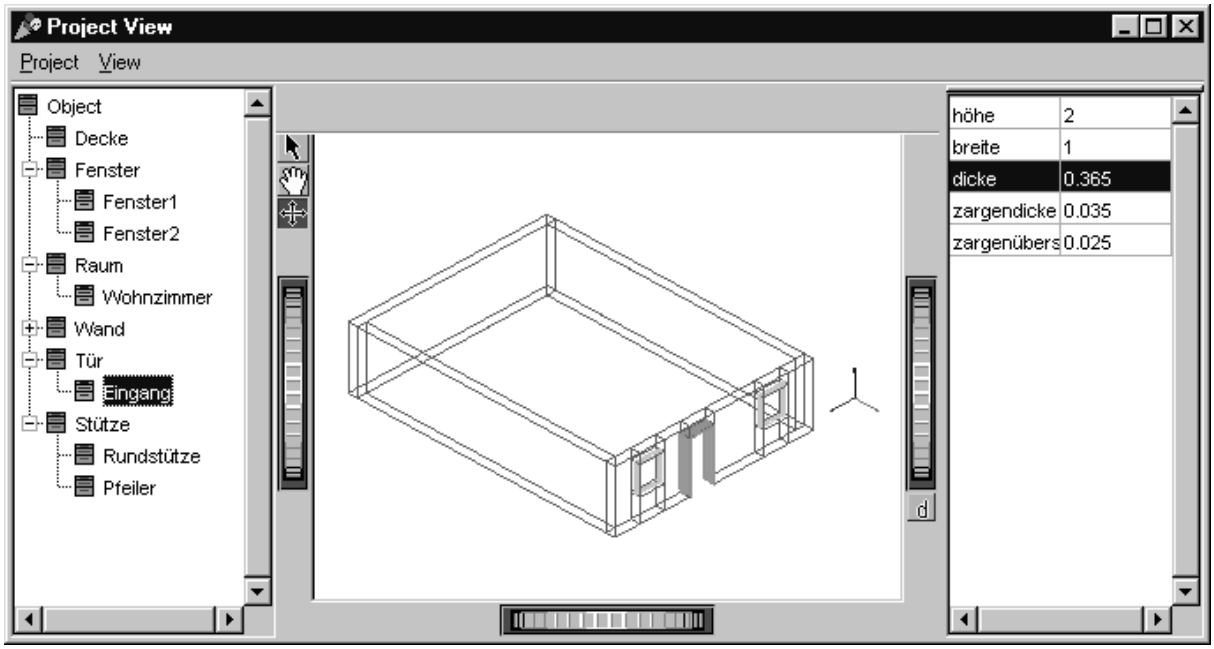

Abbildung 11: Präsentation eines Ausschnitts von Projektinformationen in GroupPlan

Die Attribute selektierter Objekte und Klassen werden im rechten Bildschirmbereich angezeigt und können dort auch editiert werden. Außerdem werden alle Objekte angezeigt, mit denen das selektierte Objekt über Aggregationen oder Assoziationen in Verbindung steht. Auf diese Weise kann das geöffnete Projekt effizient durchsucht bzw. betrachtet werden.

Die Realisierung einer vergleichbaren Präsentationskomponente für laufzeitdynamisch erzeugte Modellanteile ist auch unter Java denkbar. Dabei können die Grundgedanken der Architektur für die Darstellung der Bauwerksinformationen mit mehreren Darstellungstypen beibehalten werden. Werden nun anstelle der später zu interpretierenden Darstellungsvorschriften Java-Methodenquelltexte der Presenterklassen erzeugt, die mit den unter 4.4.3 beschriebenen Mechanismen kompiliert in die Informationsbestände des Modellverwaltungskerns übertragen werden, sollten Modelldarstellungen mit guter Responsibilität realisierbar sein. Die Realisierung derartiger Komponenten ist geplant und daher derzeit noch nicht in der Implementierungsphase.

Ferner ist anzumerken, dass derartige Darstellungsarchitekturen auch sinngemäß auf die Ausführung einfacher Operationen auf Modellinstanzen wie beispielsweise simpler Berechnungen übertragen werden können. So könnten beispielsweise einfache Regeln zur Prüfung der logischen Konsistenz von Attributbelegungen definiert und anhand konkreter Werte

ausgeführt werden oder bei redundanten Attributen abhängige Werte ermittelt werden. Naturgemäß eignen sich die beschriebenen Mechanismen aber weniger für numerisch komplexe Aufgabenstellungen.

5 Verteilte Bauwerksmodelle mit GroupPlan

5.1 Modellstruktur

5.1.1 Modellpartitionierung

Planungsprozesse im Bauwerksentwurf sind durch die kooperative Arbeit einer Anzahl von Beteiligten mit verschiedenen Rollen charakterisiert. Für eine effektive Unterstützung des Planungsprozesses ist es wesentlich, das alle Fachplaner mit den für sie adäquaten Informationsmengen im Kontext ihrer Aktivitäten versorgt werden. Somit kommt der Strukturierung der Gesamtinformationsbestände zu einem gemeinsamen Planungsgegenstand und der Integration der Fachapplikationen eine große Bedeutung zu (siehe oben).

Aus der Sicht der Anwender und der Entwickler können unter anderem folgende Wünsche an die Organisation der Gesamtinformationsbestände gestellt werden:

- Gewährleistung durchgängiger Datenflüsse zwischen den Fachapplikationen entlang der Kooperationsbeziehungen,
- keine Kommunikationsverluste durch Datennormierung oder Nichtübertragbarkeit von Informationen,
- Vermeidung sowohl der 1:n – als auch der n:n – Konverterproblematik (siehe oben),
- Unterstützung synchroner und asynchroner Phasen der Teamarbeit,
- gutes Antwortverhalten aller Applikationen,
- Übertragbarkeit von Informationen über Modelländerungen sowie von vollständigen Domänenmodellen,
- Unterstützung eines Änderungsmanagements durch die Entwurfsumgebung,
- Gewährleistung einer systemweit eindeutigen Objektidentifikation,
- Wahl einer System– und Modellarchitektur, die eine relativ einfache Konsistenzsicherung auf der physikalischen Ebene zulässt,
- Wahrung der Robustheit des Systems, unter anderem gegen Ausfälle von Komponenten und Kommunikationsmedien,
- Schaffung von externen Schnittstellen für einen Informationsaustausch über genormte Formate oder Protokolle,
- Anbindung von Legacy–Applikationen über Kapselung in Wrappern,

- Wahl einer System– und Modellarchitektur, die eine geringe Netzlast bei der Durchführung der Planungsprozesse hervorruft.

Diese Forderungen sind jedoch derzeit kaum vollständig zu erfüllen; weiterhin schließen sich einige Punkte wechselseitig aus, so dass eine konsequente Umsetzung eines Aspekts massive Kompromisse bei anderen Aspekten erfordern.

Im vorangegangenen Kapitel ist die Strukturierung der eingesetzten objektorientierten Domänenmodelle, die lokale Sicht von Anwendern, also Fachplanern auf die Modelle und technische Umsetzung der Modellverwaltung beschrieben worden. Hier soll nun der Ansatz zur Lösung der Integrationsproblematik der Informationen verschiedener Fachdisziplinen, also des Umgangs mit mehreren Domänenmodellen betrachtet werden. Grundsätzlich bestehen hier die Alternativen der Auswahl zentralisierter, replizierter oder hybrider Architekturen.

Zentralisierte Architekturen sind dadurch gekennzeichnet, dass alle verteilten Applikationen auf eine gemeinsame Datenbasis zugreifen. Somit würde bei einer konsequenten Verfolgung dieses Ansatzes versucht werden, sämtliche relevanten Informationen aller Disziplinen im Bauwerksentwurf in ein einzelnes globales Modell einzubringen und dieses Modell zentral für alle Bearbeiter zu verwalten. Ebenso sind relaxierte Varianten dieses Ansatzes denkbar, die aber je nach Abschwächungsgrad ähnliche Eigenschaften aufweisen. Vorteile der zentralisierten Variante sind die vergleichsweise einfache Konsistenzsicherung der gemeinsamen Datenbasis und die relativ unproblematische Synchronisation der verteilten Prozesse, die allerdings im Falle der Anwendung grobgranularer Sperren oder pessimistischer Transaktionskonzepte zu Problemen bei der Gestaltung responsiver Mensch–Maschine–Schnittstellen führen würden (siehe unten). Nachteilig ist die geringe Robustheit der Lösung, beispielsweise in Bezug auf Ausfall des Kommunikationsmediums, und der hohe Kommunikationsbedarf zwischen den Applikationen, welcher sich häufig in unbefriedigendem Laufzeitverhalten der Systeme niederschlägt. Im Falle einer weiten geografischen Verteilung der Entwurfspartner würden sämtliche Entwurfsaktivitäten durch Netzwerkkommunikation in Weitverkehrsnetzen erfolgen müssen; dies würde zumindest derzeit zu instabilen Systemen mit schlechtem Interaktionsverhalten führen.

Der Grundgedanke replizierender Architekturen ist, auf allen beteiligten Rechnern lokale Kopien aller relevanten Applikationen und deren Informationen bereitzustellen. Bei einer derartig verteilten Architektur würden eine Anzahl unabhängiger Domänenmodelle existieren. Sämtliche notwendige Kommunikation würde direkt zwischen den jeweiligen Modellverwaltungssystemen stattfinden, so dass eine gewisse Analogie zur m:n– Konverterlösung entsteht. Diese Lösung ist bezüglich des Ausfalls einzelner Komponenten relativ robust und erfordert nur einen Kommunikationsbedarf zur Übertragung von Änderungen relevanter Informationen, bringt aber Probleme bei der Synchronisierung des Zugriffs auf Informationen in den Modellverwaltungskernen mit sich. Somit kehren sich bei diesem Ansatz die wesentlichen Vor- und Nachteile im Vergleich zu zentralisierten Architekturen um.

Der hybride Ansatz versucht, die Vorteile der beiden anderen Möglichkeiten weitgehend zu verbinden und zeichnet sich durch das Vorkommen selbständiger, domänenabhängiger Partialmodelle und eine diese Partialmodelle integrierende Komponente aus. Die konkreten Spezifika einer hybriden Modellarchitektur sind von der konkreten Entwurfsumgebung abhängig und haben sowohl auf das Modellverwaltungssystem als auch auf die gesamte Systemarchitektur Auswirkungen.

In den hier beschriebenen Forschungsprojekten wird der logische Gesamtinformationsbestand zur Repräsentation des Bauwerks in getrennt verwaltete Domänenmodelle unterteilt. Diese werden jedoch auf semantischer Ebene durch Verknüpfungen und ein Navigationsmodell sowie technisch durch einen zentralen Projektinformationsserver zusammenhängend gehalten. Weiterhin verwenden sämtliche Domänenmodelle identisch das objektorientierte Paradigma mit denselben Modellierelementen, es existiert also eine gemeinsame Metaebene. Mit dieser Systemarchitektur wird einerseits die Konsistenzsicherung des Gesamtmodells und die Problematik der Synchronisation vergleichsweise einfach gehalten, aber andererseits die Robustheit und die Performance des Gesamtsystems nicht wesentlich beeinträchtigt. Weiterhin wird durch diese Architektur die Netzlast reduziert und für ein angemessenes Interaktionsverhalten gesorgt.

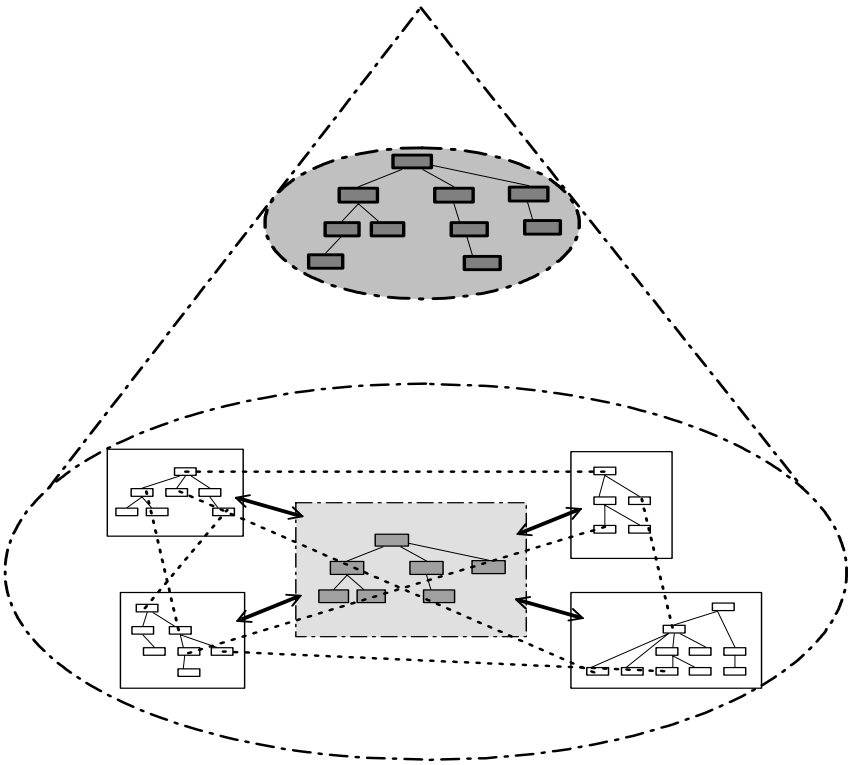

Abbildung 1: Verknüpfte Domänenmodelle mit Metamodell und Vermittlungsmodell

Das sowohl in GroupPlan als auch für die Modellverwaltung im SFB 524 verwendete Metamodell ist ebenso wie die lokale Sicht auf Domänenmodelle im vorangegangen Kapitel näher behandelt

worden. Die Techniken zum entfernten Zugriff auf Domänenmodelle sollen in den folgenden Abschnitten dieses Kapitels besprochen werden.

Bezüglich zentraler Modellkomponenten ist in GroupPlan der Ansatz der Nutzung eines zentralen, gemeinsamen Geometriemodells verfolgt worden. Grundgedanke der Bereitstellung eines derartigen Modells war, Informationen, die für alle Beteiligten am Planungsprozess relevant sind, in einem gemeinsamen Modell zu speichern. Dieses sollte aber aus Handhabungsgründen möglichst überschaubar gehalten werden; aus den Erfahrungen vorangegangener Projekte erscheint es nicht zweckmäßig, alle Informationen, die für mehr als eine Fachdisziplin relevant sind, zentral zu verwalten. Denkbar wäre aber eine Erweiterung dieses Modell um topologische Informationen wie verschiedene Nachbarschaftsbeziehungen zwischen Bauelementen, wie zum Beispiel 'trägt', 'grenzt an' oder 'liegt innerhalb', da hierdurch die Navigation und das Traversieren des Modells vereinfacht werden würde. Neben den Vorteilen durch die Verringerung der Modellkomplexität treten die Nachteile einer erschwerten Verwaltung von Änderungs– und Erweiterungsrechten im gemeinsamen, zentralen Modell. Weiterhin existieren Probleme bei der technischen Umsetzung des Zugriffs auf die gemeinsamen Informationen. Da die Häufigkeit des Zugriffs auf Geometrieinformationen für viele Fachdisziplinen sehr hoch ist, erscheint ein dauerhafter Zugriff über Weitverkehrsnetze ungünstig, da dadurch die Robustheit der Entwurfsumgebung beeinträchtigt wird. Alternativ ist eine Replikation der gemeinsamen Informationen möglich, dadurch werden aber Probleme der Synchronisierung, physischen Konsistenzsicherung und vor allem nach Netzpartitionierungen des Festlegens der gültigen aktuellen Version impliziert. Die zu erwartende hohe Änderungswahrscheinlichkeit an Geometrieinformationen würde die genannte Problematik noch weiter forcieren und die Notwendigkeit eines effektiven Update–Mechanismus unterstreichen.

Im den Bauwerksmodellen des Projektbereichs "Informationsverarbeitung" des SFB 524 wurde auf ein zentrales Geometriemodell verzichtet. Grund dafür ist, dass die Anforderungen an die Repräsentation von Geometrieinformationen zwischen den einzelnen Fachdisziplinen stark differieren und weit über die Frage der Nutzung bauteilorientierter versus raumorientierter Geometrien hinausgehen, so dass die gemeinsame Nutzbarkeit fragwürdig erscheint. Stattdessen wird der Ansatz der Schaffung eines Navigationsmodells verfolgt, das sowohl der Suche nach Informationen durch Applikationen (Data Mining) als auch der Navigation durch Anwender (3D–Visulisierung) dienen soll. Dabei soll das Navigationsmodell vorrangig Verweise auf Informationsbestände in den einzelnen Domänenmodellen verwalten, eine redundante Spiegelung von Informationsbeständen soll vermieden werden. Der Autor ist der Ansicht, dass für diesen Kontext die Anwendbarkeit oder Erweiterbarkeit von flexiblen Geometriemodellen, die sowohl eine raumorientierte als auch eine bauteilorientierte Interpretation der Informationsbestände erlauben[1], geprüft werden sollte.

[1] Ein Beispiel eines derartigen flexiblen Modells wird von Khemlani und Kalay in [Khemlani 97] beschrieben. Die weiteren Vorschläge des genannten Papers, ein derartiges Modell als alleinige Basis der

Es bleibt zu konstatieren, dass zum jetzigen Zeitpunkt noch weiterer Forschungsbedarf zu dieser Thematik besteht. Im Rahmen der geplanten Bearbeitung der Forschungsthemen des Projektbereichs "Informationsverarbeitung" des SFB 524 sind weitere Aussagen zur genannten Problematik zu erwarten.

Die Ausführung der Aktivitäten des Bauwerksentwurfs erfolgt im gemeinsamen Kontext der kooperativen Lösung der Entwurfsaufgaben bezüglich desselben Bauwerks. Daher werden keine Domänenmodelle existieren, die keinerlei Beziehungen zu anderen Domänenmodellen oder gemeinsamen Informationsbeständen haben. Alle Fachdisziplinen werden bei einzelnen Instanzen oder Attributwerten auf solche aus anderen Modellen zugreifen oder diese für andere Disziplinen bereitstellen. Somit kann die Kohärenz des logischen Gesamt–Informationsbestandes als gesichert angesehen werden. Diese Kohärenz wird durch Verknüpfungen zwischen den einzelnen Domänenmodellen abgebildet, die propagierend oder notifizierend genutzt werden können [Hauschild 98][Hauschild 00a]. Die Philosophie der Modellverknüpfungen und Ansätze zur Realisierung der entsprechenden Modelle sollen hier nicht weiter behandelt werden. Es kann hier auf Veröffentlichungen aus parallellaufenden Forschungsaktivitäten innerhalb des SFB 524 verwiesen werden, die sich dediziert mit diesem Thema befassen [Willenbacher 00] [Willenbacher 01].

Die genannten Bestandteile der Entwurfsumgebung wie die Domänenmodellserver und der zentrale Projektserver werden in diesem Kapitel als Komponenten der Systemarchitektur näher betrachtet.

5.1.2 Modellverteilung und entfernter Modellzugriff

Die Unterstützung der kooperativen Aspekte des Bauwerksentwurfs und die Integration der Informationsbestände der Bearbeiter der jeweiligen Fachdisziplinen erfordert die Etablierung einer Kommunikation zwischen den für die Speicherung der Bauwerksinformationen verantwortlichen Systemen. Der in GroupPlan und im Modellverwaltungssystem des SFB 524 gewählte Integrationsansatz auf Modellebene ist oben beschrieben worden, die Alternativen der systemtechnischen Integration wurden im ersten Teil der vorliegenden Arbeit beleuchtet. Für eine Unterstützung asynchroner und auch synchroner Kooperationsphasen sind filebasierte Lösungen wenig vorteilhaft, aufgrund der häufig heterogenen Systemstrukturen in Virtual Enterprises sind auch proprietäre bzw. betriebssystemspezifische Kommunikationslösungen oder IPC–Mechanismen keine geeignete Basis. Verteilte Objekte weisen im Vergleich zur Socketprogrammierung oder zu einfachen Remote Procedure Calls wesentlich bessere Eigenschaften bezüglich des Software Engineerings in Verteilten Systemen auf, die auch die Nachteile durch die schlechtere Performance und den geringeren Durchsatz zumindest im Kontext der beschriebenen Applikationsklasse aufwiegen.

Kooperation im Bauwerksentwurf zu nutzen, betont jedoch überproportional die Erfordernisse des architektonischen Entwurfs und ignoriert diejenigen einiger anderer beteiligter Fachdisziplinen.

Zur Auswahl einer geeigneten Middleware gelten dieselben Kriterien, die im Absatz 4.2.6 bei der Implementierung von Modellverwaltungskernen genannt wurden. Unter den offenen, nicht–proprietären Lösungen für Verteilte Objekte sind derzeit CORBA und Enterprise Java Beans (EJB) sowohl im akademischen als auch im industriellen Umfeld weit verbreitet. Der Grad der Standardisierung ist bei CORBA am weitesten fortgeschritten, durch die Übernahme wesentlicher Teile der Spezifikationen der Object Management Group (OMG) als ISO–Normen sind diese nicht nur de-facto–, sondern auch de-jure– Standards. Bei der Entwicklung von Enterprise Java Beans wurde in wesentlichen Punkten auf Kompatibilität zu CORBA geachtet, so dass eine Interaktion zwischen Objekten beider Technologien leicht realisierbar ist.

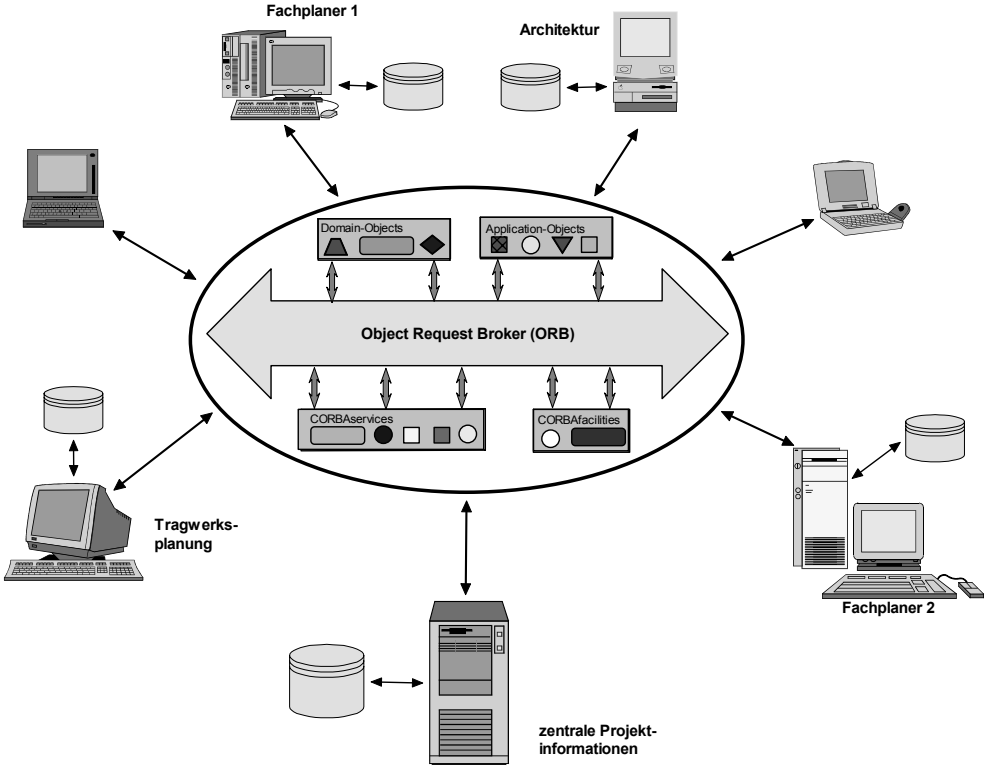

Abbildung 2: CORBA-basierte Kommunikation der GroupPlan-Komponenten

Zum Zeitpunkt der Entscheidung für die Nutzung von CORBA wies die Enterprise Java Beans Technologie noch nicht die notwendige Stabilität und Verbreitung auf. Aufgrund der positiven Weiterentwicklung von EJB wäre zu erwarten, dass Recherchen und Untersuchungen zur Auswahl der Implementierungsbasis zum heutigem Zeitpunkt mit einer ähnlich guten Eignung beider Technologien abschließen würden.

Durch die Nutzung von CORBA als Middleware wird sichergestellt, dass sowohl ein lokaler als auch ein entfernter Zugriff über LAN's, WAN's, VPN's und das Internet erfolgen kann. Die für derartig flexible Zugriffsmöglichkeiten erforderlichen Sicherheitsmassnahmen werden nachfolgend behandelt. Clients von CORBA–basierten Systemen können unter Anwendung einer Anzahl prozedural–imperativer sowie objektorientierter Programmiersprachen implementiert werden. Es existiert eine Vielzahl teilweise auch freier CORBA–Implementierungen für nahezu

alle verbreiteten Hardwareplattformen und Betriebssysteme, deren Interoperabilität durch die einschlägigen OMG–Spezifikationen (CORBA 2.x) gesichert wird. Für sehr zeitkritische Anwendungen stehen auch realtime–fähige ORB's zur Verfügung. Weiterhin werden durch die CORBAservices–Spezifikation leistungsfähige Dienste einer verteilten Infrastruktur zur Verfügung gestellt.

Zur Kommunikation zwischen den einzelnen Modellverwaltungskernen und den Applikationen wird die AKO–Schnittstelle (siehe Kap. 4) verwendet. Trotz der Möglichkeit der laufzeit-dynamischen Erweiterung und Modifikation der Domänenmodelle können alle Klassen und Instanzen im Modellverwaltungskern über eine statische, in CORBA-IDL definierte Schnittstelle zugreifen, da diese Schnittstelle auf der Metaebene der Domänenmodelle operiert. Somit kann auf die Nutzung des CORBA Dynamic Invocation Interface (DII) verzichtet werden, dessen Nutzung aufgrund des deutlich höheren client- und serverseitigen Programmieraufwandes und des größeren Ressourcenbedarfs des Modellverwaltungskerns vermieden werden sollte.

Allerdings können nicht alle Typen von MVS–Implementierungen direkt ohne weitere Vorkehrungen die Dienste des eingesetzten ORB's nutzen. Der Grund dafür ist in der Notwendigkeit des effektiven Umgangs mit einer hohen Zahl von CORBA–Objekten zu suchen. Da in der aktualisierten und erweiterten Version der AKO–Schnittstelle alle Schemata, Packages, Klassen und Instanzen einen Objektidentifikator besitzen, über den die Modellierelemente referenziert werden können, müssen diese Objekte auch in der CORBA–Infrastruktur deklariert werden. Ferner sind alle Slots, Relationen, GenericType–Objekte und RelationInstances als CORBA–Interfaces in der AKO–Definition spezifiziert und müssen daher über CORBA–Mittel zur Interaktion ansprechbar sein. Damit können in Abhängigkeit vom Abstraktheitsgrad der Bauwerksmodellierung, von der Komplexität des Bauwerks und der Granularität der Erfassung relevanter Informationen Repräsentationen von durchschnittlichen Bauwerken nach den vorliegenden Erfahrungen fünf- bis siebenstellige Objektanzahlen mit sich bringen. Bei sehr komplexen Gebäuden oder einer extrem detaillierten Modellierung sind aber auch durchaus noch deutlich größere Informationsmengen zur Abbildung der Projektinformationen denkbar. CORBA–Umgebungen besitzen zwar eine gute Skalierbarkeit, die Verwaltung derartig großer CORBA–Objektmengen durch einen ORB auf einer Servermaschine ist aber bei einer derzeitig üblichen, Supercomputer oder große Cluster explizit ausgenommen, oder kurzfristig zu erwartenden Hardwareausstattung nur sehr eingeschränkt möglich, da für jedes echte Objekt in einer Distributed Objects Computing (DOC)–Middleware ein gewisser Speicherbedarf und mindestens ein eigener Thread benötigt werden (siehe Abbildung 4).

Daher müssen in Abhängigkeit von der zugrundeliegenden Implementierungsstrategie des Modellverwaltungskerns Vorkehrungen getroffen werden, um eine effiziente und responsive Verwaltung der Modellelemente in der Middleware–Infrastruktur zu ermöglichen.

Die interne Strategie der Objektverwaltung und die Details der technischen Umsetzung eines ORB's ist im Gegensatz zum Verhalten der API's, den Language Mappings und den

Kommunikationsprotokollen nicht genormt. Dadurch ergibt sich je nach Programmiersprache und Plattform ein mehr oder minder großer Spielraum bei der Realisierung derartiger Middleware. Eine gängige Variante der Implementierung eines ORB's für die Nutzung unter Smalltalk–80 besteht in dem Ansatz, Programmiersprach–Objekte, die durch die Middleware angesprochen werden sollen, von einer speziellen Oberklasse erben zu lassen. Die dabei ererbten Attribute werden für die Verwaltung der Middleware genutzt, die nötigen Methoden finden sich im wesentlichen an den Oberklassen. Zur Gewährleistung der quasiparallelen Abarbeitung kann auf den Threadmechanismus der Programmierumgebung zurückgegriffen werden. Der ORB ist für das Auffinden des zu einer IOR gehörigen Smalltalk–Objekts verantwortlich, was durch eine effizient strukturierte Mapping–Tabelle leicht möglich ist und muss die übliche Kommunikationsinfrastruktur bereitstellen. So ist es auch möglich, dass trotz des dynamischen Typssystems von Smalltalk alle in dieser Sprache verwalteten CORBA–Objekte über die statische Aufrufschnittstelle angesprochen werden können. Somit können in Smalltalk CORBA–Objekte mit vergleichsweise geringem Ressourcenbedarf verwaltet werden und ein Modellverwaltungskern mit der in Abschnitt 4.2.4 beschriebenen Implementierungsstrategie muss diesbezüglich keine weiteren Vorkehrungen treffen.

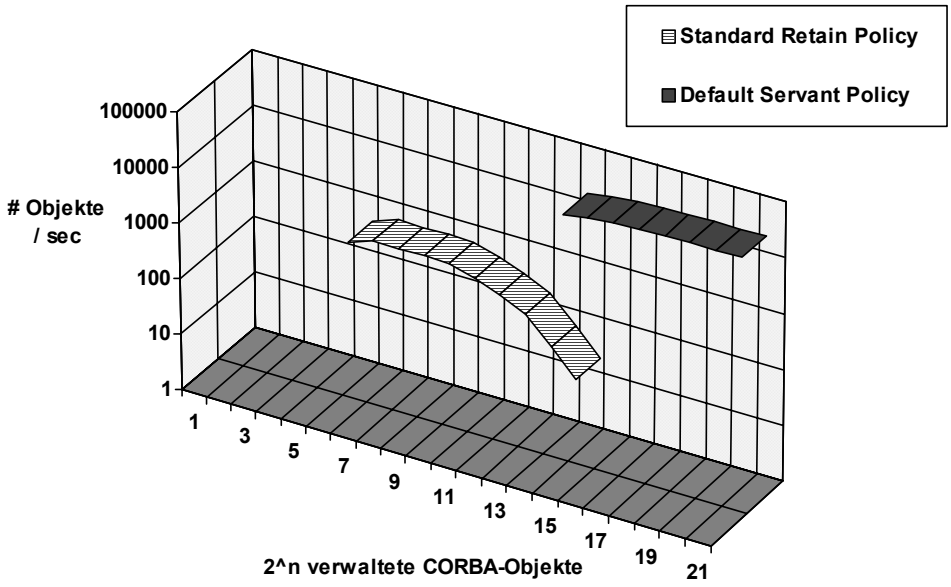

Abbildung 3: Verhalten von CORBA-Policies zur Objektverwaltung

Dies gilt aber nicht für die in Abschnitt 4.2.5 beschriebene, für Sprachen wie Java typische Implementierungstechnik. Abbildung 3 zeigt das Verhalten eines ORB's beim Erzeugen von CORBA–Objekten. Die hell gefüllte Linie zeigt die Erzeugung von Objekten mit der Standard Objektverwaltungs-Policy, d.h. jedem Objekt wird ein eigener CORBA–Servantprozess zugeordnet. Bis zu einem gewissen Punkt lässt sich eine weitgehend konstante Anzahl von Objekten erzeugen, danach –hier im vierstelligen Bereich– geht die Quote dramatisch zurück. Die exakte Anzahl erzeugbarer Objekte und die Dauer des linearen Verlaufs ist von der vorliegenden

Hardware abhängig, der Grundcharakter des degressiven Kurvenverlaufs ist jedoch dieser Art der Objektverwaltung immanent. Aufgrund der zu verwaltenden Objektmengen ist daher die Standard–Objektverwaltungstechnik von CORBA nicht für den Einsatz als Middleware für den verteilten Zugriff auf Informationen in Modellverwaltungssystemen geeignet.

Die CORBA Version 2.5 standardisiert verschiedene Taktiken (*policies*) zum Umgang mit den durch den Portable Object Adapter (POA) verwalteten Objekten. Zu diesen gehört die Default–Servant–Policy, die erlaubt, einer Anzahl von CORBA–Referenzen einen einzelnen Servant zuzuweisen (siehe Abbildung 3, dunkle Linie). Bei Empfang einer Botschaft an eine derartige Referenz kann vom Portable Object Adapter ein Bytestring abgefragt werden, der auf den eigentlichen Empfänger der Botschaft verweist. Somit kann für jeden CORBA–Interfacetyp der AKO–Spezifikation ein einzelner DefaultServant verwaltet werden, der in Zusammenarbeit mit dem POA und dem IORManager des Modellverwaltungssystems die Aufrufe an Java-Objekte weiterleitet, die im Modellverwaltungskern mit Hilfe der Datenbank persistent verwaltet werden.

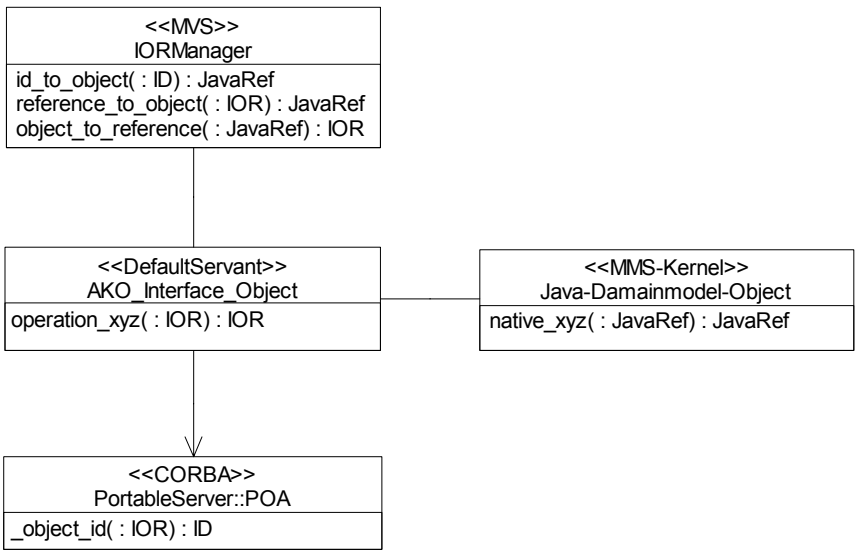

Abbildung 4: Default-Servant-Mechanismus im MVS (vereinfacht)

Der beschriebene Mechanismus zeichnet sich durch sehr sparsamen Ressourcenverbrauch aus. Wie Abbildung 4 zeigt, weist der implementierte Default–Servant–Mechanismus mindestens bis in den sieben– bis achtstelligen Bereich der Objektanzahlen ein lineares, konstantes Laufzeitverhalten auf, außerdem ist die Lösung circa achtmal schneller. Deutlich außerhalb der Grenzen ist auch hier durch den Speicherbedarf eine leichte Degression zu erwarten, jedoch sollte diese mit Hardwareaufrüstungen und besonders RAM–Erweiterungen der Servermaschine ausgleichbar sein, so dass Bauwerksmodelle von Entwurfsgegenständen normaler bis mäßig erhöhter Komplexität im MVS verwaltbar sind.

Da in Modellverwaltungssystemen mit dem unter 4.2.3 beschriebenen C++–typischen Implementierungsansatz ohnehin keine Programmiersprachobjekte mit Bauwerksmodellsemantik existieren, ist eine direkte Abbildung der Modellelemente ohnehin nicht möglich. Da hier ein

Mapping von CORBA–Referenzen auf interne Datenstrukturen erfolgen muss, bietet sich auch hier der Default–Servant–Mechanismus als Lösung der Problematik an. Der bei der POA–Abfrage resultierende Bytestring kann dann als Referenz in der Datenstruktur genutzt werden.

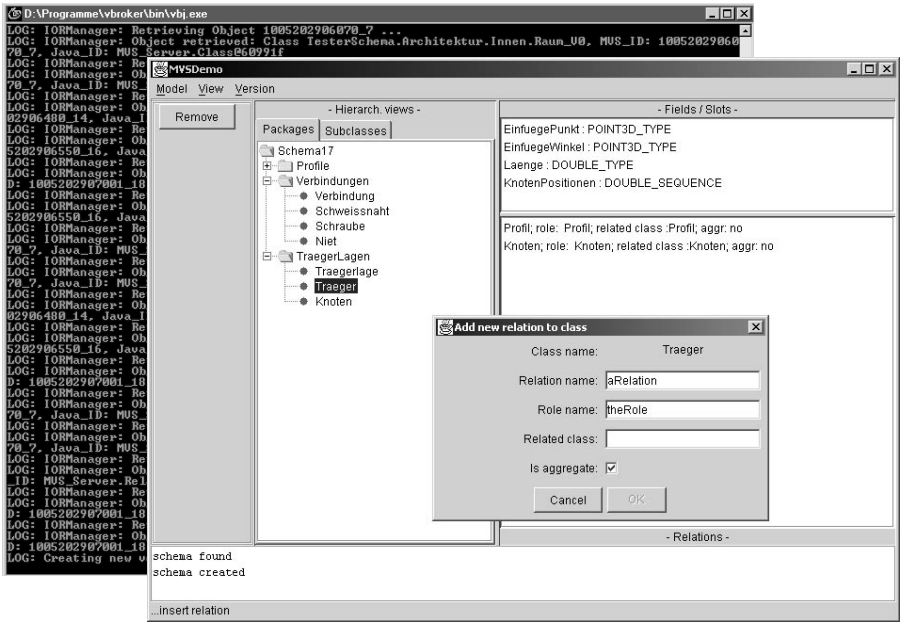

Abbildung 5: Modellverwaltungssystem und Modellbrowser für Administrationsaufgaben

Während des Bauwerksentwurfs muss die Modellkonsistenz auf mehreren Ebenen überwacht werden. Auf unterster, physischer Ebene muss die Bereitstellung der korrekten und aktuellen Attributwerte gesichert werden. Daher haben Techniken zur Synchronisation paralleler Zugriffe auf Modellinformationen eine große Bedeutung. Diese werden im nächsten Kapitel besprochen.

Auf Partialmodellebene ist zu sichern, dass die Entwurfsdaten (Instanzen) mit ihren Klassen korrelieren. Dies ist nicht grundsätzlich gegeben, da Klassen zur Beschreibung von Nutzerwissen zur Laufzeit definiert und vor allem redefiniert werden können. Außerdem ist die Integration von in UML–Object Constraint Language (OCL) formulierten Regeln in die Domänenmodelle als Aktivität im SFB 524 geplant. Damit lassen sich beispielsweise Minima, Maxima und numerische Abhängigkeiten von Attributwerbelegungen überwachen. Die Überwachung der Constraints erfolgt durch das Modellverwaltungssystem, welches die aufsetzenden Applikationen über Regelverletzungen informiert.

Auf oberstem Niveau ist einerseits die Konsistenz der Relationen zwischen den Partialmodellen zu überwachen und andererseits definierte Zustände nach bestimmten Phasen des Bauwerksentwurfes zu verwalten. Diese Zustände markieren den Abschluss gewisser Phasen bzw. das Erreichen bestimmter Zwischenziele im Bauwerksentwurf (Freigaben). Die Sicherung der Konsistenz obliegt in unteren Niveaus dem Systemkern, in höheren Schichten muss eine Aufgabenteilung zwischen Systemkern und den Client-Anwendungen stattfinden, da hier Wissen

über die Semantik der Informationen im Objektsystem vorauszusetzen ist. Bezüglich der Konsistenzsicherung auf den höheren Levels ist zu konstatieren, dass zum heutigen Zeitpunkt noch Forschungsbedarf existiert und entsprechende Aktivitäten geplant sind.

5.1.3 Rollenkonzept

Vor der Betrachtung der technischen Aspekte der Umsetzung ist das zentrale Konzept der Rolle zu behandeln, da dieses unter anderem für taktische Festlegungen bezüglich der Systemarchitektur, der Zugriffskontrollmechanismen und der Concurrency Control relevant ist. Die Semantik der Rollen im Bauwerksentwurf, die zugeordneten Aktivitäten und Privilegien sind Gegenstand parallellaufender Forschungsaktivitäten der Prozessmodellierung bzw. der Prozessplanung und sollen hier nicht näher betrachtet werden.

Der Begriff der Rolle findet sich in psychologischen sowie soziologischen Betrachtungen zu Gruppenprozessen und kann auf Systeme zur Unterstützung von kooperativer Arbeit übertragen werden. Nach Zimbardo wird unter dem Begriff der Rolle im Kontext von Gruppenprozessen ein sozial definiertes Verhaltensmuster verstanden, das von einer Person, die eine bestimmte Funktion in einer Gruppe hat, erwartet wird. Rollen sind größtenteils von dem bestimmten Individuum, das sie innehat, unabhängig; die erwarteten Verhaltensweisen sind die gleichen, gleichgültig, über welche persönlichen Merkmale der Rolleninhaber verfügt [Zimbardo 92].

Nach Borghoff und Schlichter ermöglichen Rollen, die Interaktionen zwischen den Teilnehmern zu strukturieren und die Funktionalitäten unabhängig von der Rollenverteilung zu definieren. Eine Rolle definiert die soziale Funktion eines Einzelnen in Beziehung zum Gruppenprozess, zur Organisation und zu anderen Gruppenteilnehmern [Borghoff 98]. Dies schließt insbesondere auch die Berücksichtigung der Kompetenzen, Kenntnisse, Fähigkeiten und Fertigkeiten der Teilnehmer ein. Ferner definiert eine Rolle die Rechte und Pflichten im Rahmen des Gruppenprozesses sowie die Kontrolle über die Informationseinheiten (Lese- und Schreibrechte) und die Aktivitäten, welche die einzelnen Beteiligten ausführen dürfen bzw. müssen.

Auf den Bauwerksentwurf übertragen bedeutet dies, dass die Ausführung bestimmter Aktivitäten im Planungsprozess Rollen zugeordnet werden kann und umgekehrt diese Aktivitäten den Charakter der Rolle bestimmen. Für das Übernehmen einer Rolle im Planungsprozess ist eine Spezialisierung, Ausbildung oder sonstige Befähigung notwendig. Durch die Ausübung einer Rolle können Berechtigungen zu Modifikationen, Erweiterungen, aber auch zur Kenntnisnahme von bestimmten Informationen im Bauwerksmodell abgeleitet werden.

Erweitert man das Rollenkonzept über reine soziologische Gruppenprozesse hinaus, kann man Rollen auch als spezielle Akteurtypen ansehen, die Generalisierungen von Akteuren mit denselben Fähigkeiten, Kompetenzen und Zugriffsrechten auf Aktivitäten und Informationen darstellen. Akteure können dabei Personen, Gruppen oder Softwaresysteme sein. Diese Erweiterung ist auch im Bauwerksentwurf sinnvoll, da beispielsweise bestimmte Aktivitäten an eine Firma gebunden werden können. Welcher konkrete Mitarbeiter der Firma dann mit der

Bearbeitung betraut wird, liegt im Ermessen der Verantwortlichen der Firma und ist im allgemeinen für die Projektpartner nicht relevant. Ebenso ist die Kontrolle der Zugriffs auf Bauwerksmodellinformationen durch bestimmte Applikationen sinnvoll.

Bei der Definition von Rollen sind folgende Aspekte zu berücksichtigen:

- Rollen sollen in ihrer Definition die auszuführenden Aktivitäten, wenigstens für die zentralen Tätigkeitsbereiche, enthalten.
- Die Aktivitäten der Rollenpartner sind komplementär zueinander und die Resultate der Aktivitäten entsprechen den Erfordernissen des Planungsprozesses.
- Die Zuordnung von Personen zu Rollen hat die Struktur des Virtual Enterprise und den Organisationsablauf zu berücksichtigen. Rollen identifizieren Tätigkeiten, die gegebenenfalls nur einen Teil, unter Umständen aber auch das Mehrfache der Arbeitszeit eines Mitarbeiters beanspruchen, deshalb kann ein Mitarbeiter durchaus mehrere Rollen innehaben bzw. eine Rolle von mehreren Mitarbeitern wahrgenommen werden.
- Einige Rollen, die für das Projekt relevant sind, können auch über die Dauer des Projektes an eine Organisationseinheit oder einen Partner im Virtual Enterprise gebunden sein. Dies ist vor allem dann sinnvoll, wenn die Aufgabeninhalte der Rollen sehr anspruchsvoll sind.
- Es können Rollen im Projekt wegfallen, wenn ihnen durch den Projektzuschnitt keine Aktivitäten mehr zugeordnet sind.
- Bei kleinen Projekten müssen zwangsläufig mehrere Rollen durch eine Person abgedeckt werden. Bei großen Projekten wird in der Regel jede Rolle durch verschiedene Personen wahrgenommen.

Bei der Bearbeitung komplexer Planungsaufgaben kann also der Fall auftreten, dass mehrere Entwurfsbearbeiter dieselbe Rolle und dieselbe Sicht auf das Bauwerksmodell haben und für Teile desselben Planungsabschnitts verantwortlich sind. Dies trifft zum Beispiel zu, wenn mehrere Mitarbeiter eines Planungsbüros die Bearbeitung einer Teilaufgabe gemeinsam übernehmen.

Wie eingangs erwähnt wurde, bleibt die Definition exakter Rollen und deren Inhalte im Verantwortungsbereich anderer Forschungsaktivitäten und soll hier nicht betrachtet werden. Ein Modellverwaltungssystem muss jedoch entsprechende Mechanismen zur Umsetzung der den Rollen zugeordneten Privilegien zur Verfügung stellen, welche die oben genannten Aspekte berücksichtigen.

Für Werkzeuge zur Planung der Entwurfsprozesse und zum Prozessmanagement ist es wünschenswert, Objektmengen im Bauwerksmodell einzelnen Aktivitäten und Verantwortlichen, also Personen, die eine bestimmte Rolle ausüben, zuordnen zu können. Zu diesem Zweck wurde ein Mechanismus in das Modellverwaltungssystem des SFB 524 integriert, der es erlaubt, Modellobjekte mit einer Kennzeichnung bezüglich ihrer Position im Bauwerk, ihrer Funktion

sowie ihrem Material oder ihrem Bauteiltyp zu signieren. Die Signaturen bestehen aus entsprechenden Strings, die durch die Konkatenation der Knotennamen beim Traversieren des Wurzelbaums der weitgehend vorzukonfigurierenden Kriterientaxonomie beginnend von der Wurzel bis zur konkreten Signaturteilinformation gebildet werden. Durch die boolsche Verknüpfung von beliebig langen und damit beliebig konkreten Suchstrings lassen sich nun Objektmengen unabhängig von deren Typ oder Zugehörigkeit zu Kompositionsstrukturen ermitteln, die den gegebenen Kriterien genügen. Dabei müssen nicht zwingend alle drei Teilsignaturen genutzt werden, natürlich lassen sich andererseits mit detaillierten Suchanfragen auch sehr kleine und exakte Treffermengen erzielen, wenn eine sinnvolle Vorkonfiguration der Kriterientaxonomien vorausgesetzt wird. Es soll hier betont werden, dass durch diesen Mechanismus keine eineindeutige Zuordnung von Objekten und Suchstringkombinationen erzielt werden soll und kann, eine Zuordnung zu Planungsschritten ist aber möglich. Beispielsweise lassen sich auf diesem Wege alle Stahlbetonbauteile, die zur Tragstruktur gehören und sich im Erdgeschoss des Nordflügels des dritten Bauabschnitts befinden, ermitteln. Ebenso sind alle Heizkörper der Bauart 'Logatrend VK–Profil', Typ 22, Breite 1,20 m, Höhe 0,80m im Dachgeschoss eines Mehrfamilienhauses zu ermitteln. Derartige Informationen sind nicht nur für den ersten Planungsprozess, sondern ebenso für Revitalisierungsmaßnahmen oder das Facility Management des Bauwerks von Interesse.

5.2 Gesamtsystemarchitektur

Beim Entwurf der Systemarchitektur waren einige Prämissen zu beachten, die im folgenden vor der Behandlung der einzelnen Komponenten der Systemarchitektur betrachtet werden sollen.

Für eine effektive Unterstützung der Kommunikation, Koordination und Kooperation in Virtual Enterprises müssen aufgrund der engen Kooperationsbeziehungen zwischen einer Vielzahl von Beteiligten sowie deren zunehmender geographischer Verteilung als Kommunikationsmedien das Internet, Virtual Private Networks (VPN's) oder sonstige Weitverkehrsnetze (WAN's) genutzt werden. Dies impliziert, dass die Systemkomponenten eine hohe Robustheit gegenüber Verzögerungen oder Ausfällen des Kommunikationsmediums aufweisen, da die Nutzbarkeit der Entwurfsumgebungen der einzelnen Entwurfsbearbeiter nicht von Ausfällen oder Verzögerungen des Mediums beeinträchtigt werden darf.

Wie oben ausgeführt worden ist, können durchaus mehrere Bearbeiter mit derselben Rolle betraut oder eine Rolle an eine Firma oder sonstige Organisationseinheit übergeben werden. Beim Entwurf der Systemarchitektur wurde davon ausgegangen, dass in solchen Fällen keine geographische Verteilung vorliegt und über die zuverlässige und vergleichsweise sehr störunanfällige LAN–Technik als technisches Kommunikationsmedium auf die gemeinsamen Datenbestände lokal zugegriffen werden kann. In Virtual Enterprises werden die einzelnen Planungsaufgaben üblicherweise an Planungsbüros oder sonstige Firmen übertragen. Falls eine Firma andere Partner zur Erfüllung von Teilaufgaben beauftragt, kann wiederum auf die übliche

Kommunikationstechnik im Virtual Enterprise zurückgegriffen werden. Somit stellt diese Entscheidung für die Mehrzahl der Projekte keine Einschränkung dar.

Wie eingangs beschrieben wurde, ist bei Architekturen für Systeme zur Kooperationsunterstützung eine Entscheidung zwischen zentralen und dezentralen bzw. replizierenden Lösungen zu treffen, dies bedeutet, dass letztendlich ein Kompromiss zwischen den Anforderungen an die Verfügbarkeit des Servers und der Konsistenzsicherung des Bauwerksmodells notwendig wird. Diese Problematik trifft sowohl im großen auf Gesamtbauwerksmodelle mit allen dazugehörigen Partialmodellen als auch im kleinen auf einzelne Partialmodelle zu.

Klassische Ansätze der Verteilten Systeme, wie Client/Server–Techniken, lassen sich nicht uneingeschränkt auf Systemarchitekturen für verteilte Planungsumgebungen übertragen. Mehrschichtenarchitekturen (z.B. three tier architectures), die in vielen Bereichen erfolgreich angewendet werden, sind kaum für Planungsprozesse in Virtual Enterprises (VE) geeignet. Die Ursachen dafür liegen in der Dynamik der Organisationsstrukturen von Virtual Enterprises, für die während der Projektbearbeitung ein– und austretende Partner charakteristisch sind. Durch die zeitbefristete Kooperation im VE und die mögliche gleichzeitige Mitgliedschaft in mehreren derartigen Organisationsstrukturen kommt im allgemeinen nicht die für Mehrschichtenarchitekturen notwendige Normierung zustande. Ebenso lässt sich aus dem Planungsprozess und Bauwerkslebenszyklus kein dauerhaft Verantwortlicher für die zentralisierten Komponenten einer Mehrschichtenarchitektur ableiten.

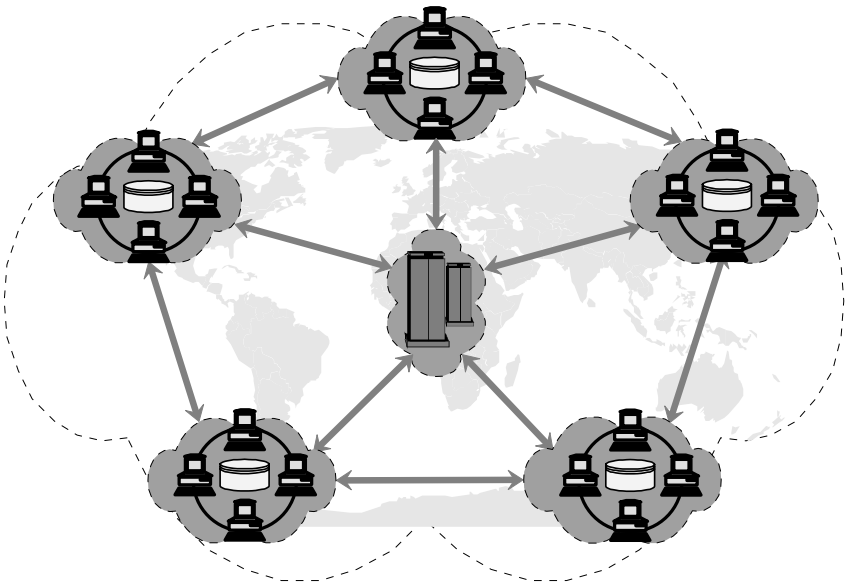

Abbildung 6: Grobdarstellung der Systemarchitektur

Stattdessen wird der Ansatz verfolgt, Bauwerksinformationen in diesem stark heterogenen Umfeld in für die einzelnen Fachdisziplinen verantwortlichen Bauwerksmodellservern dezentral zu verwalten und eine gemeinsame Infrastruktur zur Kommunikation, Koordination und Kooperation bereitzustellen. Damit wird die Verwaltung der Domänenmodelle im Bereich

derjenigen Personen oder Organisationen belassen, die auch die Bearbeitung der Planungsinformationen durchführen und die juristische Verantwortung für diese Informationen tragen. Jedoch müssen Informationen über die Infrastruktur des Planungsprojekts auf einem zentralen Knoten verfügbar sein, ebenso bietet sich eine zentrale Speicherung von Komponenten zur Navigation und zur Suche über den logischen Gesamtinformationsbestand an.

Daher ergibt sich eine Systemarchitektur, die aus den drei Grundkomponenten:

- zentraler Projektserver,
- Domänenmodellserver und
- Domänenclient

besteht (Abbildung 7). Die einzelnen Komponenten der vorgeschlagenen Systemarchitektur, mit der beabsichtigt wird, dem Charakter der kooperativen Bauwerksplanung in Virtual Enterprises Rechnung zu tragen, sollen in folgenden vorgestellt werden.

5.2.1 Zentraler Projektserver

Der zentrale Projektserver ist für die Verwaltung bestimmter projektglobaler, infrastruktureller oder navigationsbezogener Informationen zuständig.

Dies betrifft einerseits notwendige Informationen über die technische Infrastruktur des Planungsprojekts. Zu diesen Informationen gehören die Hostnamen und die genaue DNS-Domainbezeichnung der Server mit Modellinformationen oder deren IP–Adresse bzw. gegebenenfalls NetBIOS–Name, die CORBA– Interoperable Object References (IOR's) der Modellverwaltungskern–Servants und diejenigen der Infrastruktur–Dienste des Bauprojekts. Diese Komponente ist erforderlich, um die Ressourcen der einzelnen Partner im Virtual Enterprise entsprechend den auftretenden Beziehungen dynamisch verwalten zu können. Die Komponente verfügt über einen Zuordnungsmechanismus von logischen Namen, beispielsweise "Tragwerksmodell" zu entsprechenden IOR's. Über eine CORBA–Schnittstelle ist es möglich, Einträge zu erzeugen, zu löschen und abzufragen. Damit besitzt diese Komponente funktionelle Ähnlichkeiten zu CORBA–Nameservices, wichtig war aber bei der Realisierung eine sofortige sichere persistente Speicherung der Informationen, und Vorkehrungen zum sofortigen Wiederanlauf des Dienstes nach Störungen oder Ausfällen. Die Referenz auf diesen Dienst ist die einzige Referenz in der Systemarchitektur, die über eine statische Adresse beschafft werden muss. Dazu kann das Projektdirektory so konfiguriert werden, dass eine IOR in der externen ASCII–Stringdarstellung wahlweise auf einem http-, ftp- oder Fileserver (NetBIOS–Pfad) abgelegt werden kann. Dieser Einsprungspunkt wird allen Partnern mitgeteilt und deren Modellverwaltungssysteme entsprechend konfiguriert. Durch diesen Mechanismus wird es möglich, jederzeit auf die verfügbaren und freigegebenen Informationen des Projekts zuzugreifen, auch wenn beispielsweise Migrationen von Diensten oder Hinzufügungen neuer Partner aufgetreten sind.

Weiterhin existiert eine Komponente zum Management der Authentisierungs– und Autorisierungsdaten, welche die Verwaltung der Lese- und Schreibrechte der einzelnen Projektpartner für die verschiedenen Bauwerksmodellinformationsbestände der Fachdisziplinen projektglobal bereitstellt. Die zentralisierten Komponenten der Verknüpfungsverwaltung sind ebenso Bestandteile des zentralen Projektservers. Gleiches gilt für Schnittstellen zur Kooperationsunterstützung. Die damit verbundenen Mechanismen werden im folgenden Kapitel betrachtet.

Eine weitere notwendige Komponente des zentralen Projektservers ist ein zentrales Navigationsmodell, welches Verweise auf spezielle Informationsbestände, die bestimmten Sichten zugeordnet sind, verwaltet. Durch eine Teilkomponente ist dieses auch für die visuelle Navigation von Bearbeitern durch das zu revitalisierende Bauwerk verantwortlich; ebenso ist eine Komponente zur Suche in den Datenbeständen zu realisieren. Wie oben ausgeführt, existiert noch Forschungsbedarf bezüglich des Navigationsmodells.

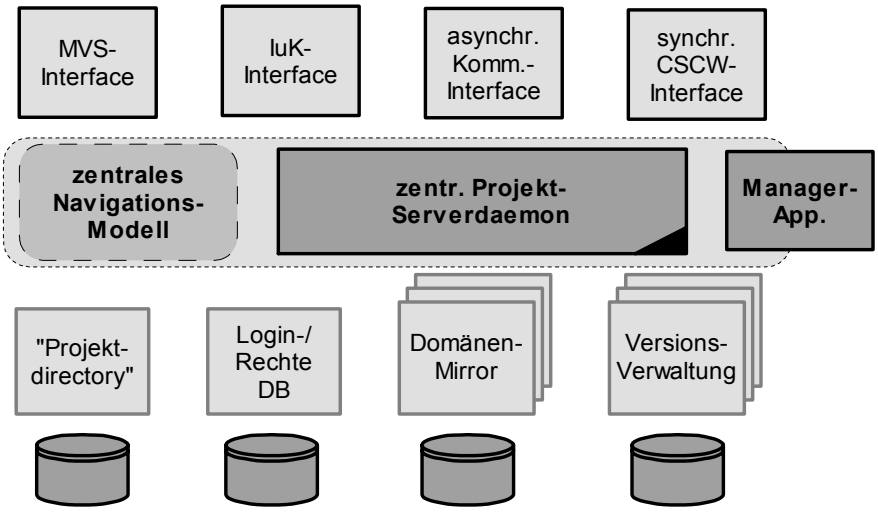

Abbildung 7: Struktur des zentralen Projektservers

5.2.2 Domänenserver

Die Hauptaufgabe der Domänenmodellserver besteht in der Verwaltung der zur jeweiligen Fachdisziplin gehörigen Klassen– und Objektinformationen. Diese Aufgaben werden im wesentlichen durch deren wichtigste Komponente, den im vorangegangenen Kapitel ausführlich beschriebenen Modellverwaltungskern, erfüllt. Die verwalteten Modellinformationen des Domänenmodellservers werden über die beschriebenen CORBA–basierten Schnittstellen mit Metaebenen–Semantik für die Domänenclients und externe Server zur Verfügung gestellt.

Ein Modul zur domäneninternen Verwaltung von Zugriffsrechten ist ebenso erforderlich, welches allerdings für Belange des Domänenmodells und für Berechtigungen zu Anfragen an externe Server zuständig ist. Ferner müssen Vorkehrungen zur Sicherung der physikalischen Datenintegrität und zur Vermeidung von Problemen durch Konkurrenzsituationen (race

conditions) infolge synchronen Modellzugriffs getroffen werden. Dies kann über Sperrmechanismen und geeignete Transaktionskonzepte erreicht werden, die nachfolgend beschrieben werden. Die entsprechenden Techniken können in den Modellverwaltungskern integriert werden, ebenso werden derartige Mechanismen durch das darunter liegende Datenbankverwaltungssystem und durch verschiedene Middleware–Lösungen bereitgestellt.

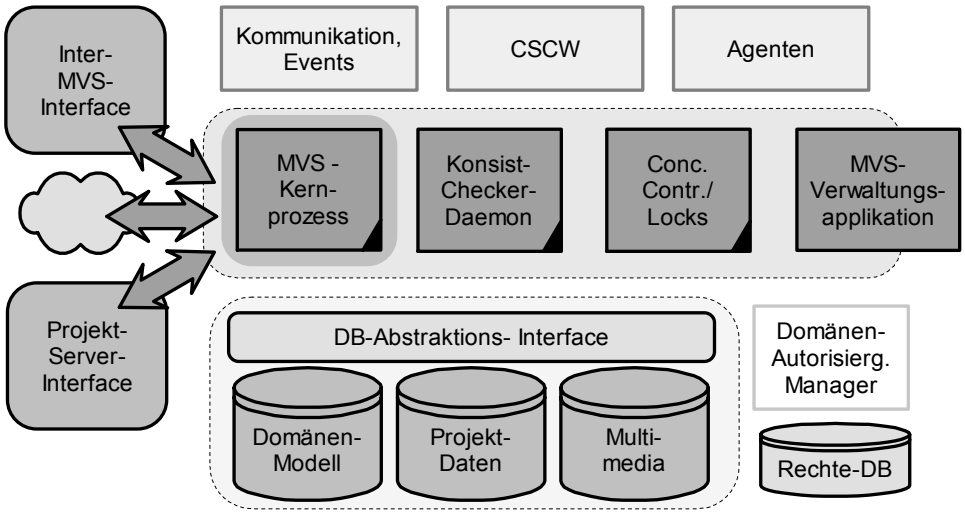

Abbildung 8: Struktur des Domänenservers

Die logische Konsistenz kann bis zu einem gewissen Grad durch die Integration von konsistenzbeschreibenden Regeln in die Domänenmodelle geprüft werden. Der Domänenmodellserver ist verantwortlich für die Auswertung dieser Constraints, Maßnahmen zur Wiederherstellung eines korrekten Zustands müssen jedoch den aufsetzenden Applikationen und den Anwendern der Entwurfsunterstützungsumgebung überlassen werden. Daher existiert ein Benachrichtigungsmechanismus, über den der Modellverwaltungskern durch einen CORBAservices–basierten asynchronen Benachrichtigungsdienst (CORBAservices Notification Service) die für den Empfang derartiger Nachrichten angemeldeten Domänenclienten informiert.

Weiterhin ist der Domänenserver für jenen Teil der Groupware–Funktionalitäten verantwortlich, der sich nicht auf reine Informationspräsentation und Systeminteraktion beschränkt.

5.2.3 Domänenclient

Die genannten Präsentations- und Interaktionsaufgaben werden vom Domänenclient übernommen. Auf allen Rechnern der beteiligten Bearbeiter im Planungsprozess befindet sich ein Exemplar dieser leichtgewichtigen Interfaceimplementierung, die für Kommunikation mit dem zuständigen Domänenmodellserver verantwortlich ist. Deren Hauptaufgabe ist es jedoch, als Schnittstelle zwischen Modellverwaltung und Fachapplikation zu fungieren, um den Entwerfenden bzw. den von ihnen genutzten Applikationen eine Interaktion mit dem

gemeinsamen, verteilten Bauwerksinformationsbestand zu ermöglichen. Weiterhin werden die über den Kommunikationskanal vom Domänenmodellserver empfangenen Benachrichtigungen an die Applikationen und den Anwender weitergeleitet, ebenso müssen die Authentisierungsinformationen des Nutzers entgegengenommen werden.

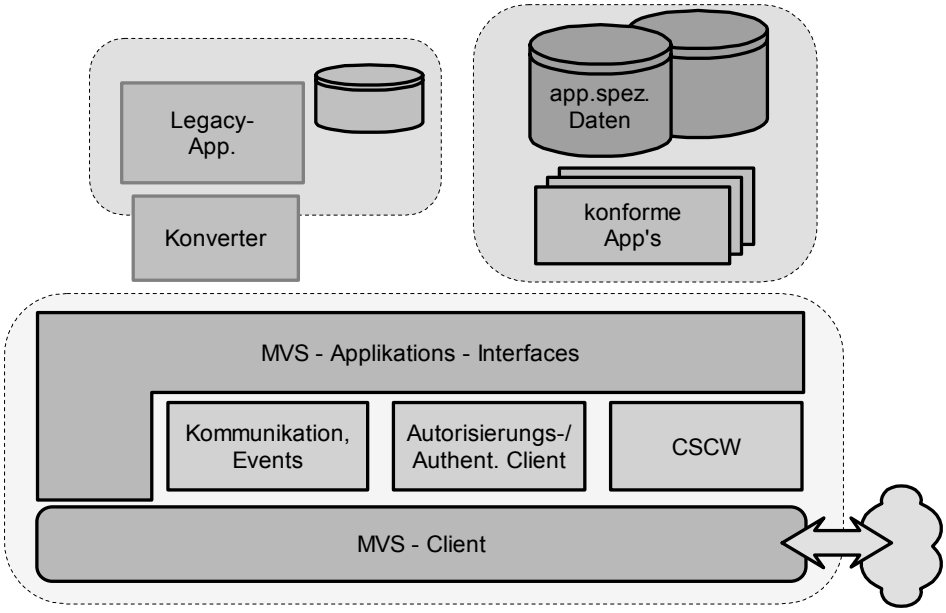

Abbildung 9: Struktur des Domänenclients

Falls es sich als notwendig erweist, ältere monolithische Applikationen ohne offene Application Programming Interfaces (API's) anzubinden, ist es möglich, auf Basis der clientseitigen MVS–Schnittstellen einen Konverter zur filebasierten Anbindung der Applikation zu realisieren, der als Wrapperapplikation in der Verteilten Umgebung fungiert. Je nach Aufbau der Fileschnittstelle gehen dabei jedoch bestimmte Funktionalitäten verloren. So ist dann die Unterstützung synchroner Kooperation durch WYSIWIS–Darstellungen durch Mittel der Entwurfsumgebung nicht möglich. Ist es nicht möglich, die uniquen Identifikatoren im MVS mit an die Applikation zu übergeben und diese mit erzeugten Informationen wieder ausgeben zu lassen, wird eine bidirektionale Kommunikation mit der Applikation stark erschwert, insofern Modifikationen an übergebenen Informationen vorgenommen wurden, da eine automatische Zuordnung nicht mehr möglich ist. In der Konsequenz muss in diesen Fällen entweder auf eine bidirektionale Kommunikation verzichtet werden oder der Anwender muss assistiert durch die Wrapperapplikation die Zuordnung der geänderten Objekte herstellen oder mindestens prüfen.

5.2.4 Fazit

Die oben vorgestellte Systemarchitektur unternimmt den Versuch, die Anwendbarkeit aktueller Technologien der Verteilten Objekte als Kommunikationsbasis in Virtual Enterprises in der Bauplanung zu untersuchen und eine adäquate Strukturierung der Architekturkomponenten

aufgrund der stark eingeschränkten Übertragbarkeit klassischer Lösungen vorzuschlagen. Die wesentlichen Module der drei Architekturkomponenten zentraler Projektserver, Domänenserver und Domänenclient wurden (teils bis zu einem hohem Reifegrad, teils prototypisch) umgesetzt, so dass die technologische Tragfähigkeit des genannten Konzepts als gesichert gelten kann. Aufgrund verschiedener Faktoren, wie der aus Gründen der Entwicklungskapazität teils recht prototypischen Umsetzung und des für kleine Teams sehr hohen Aufwands der Anbindung voll-funktionaler, industrieller Fachapplikationen, war jedoch ein Praxistest unter Real-World-Bedingungen mindestens über die volle Dauer eines Planungsprojekts nicht möglich.

Ferner sind für den Einsatz unter industriellen Planungsbedingungen mindestens noch Module zur Erhöhung der Daten- und Ausfallsicherheit zu realisieren, die hier nicht Gegenstand der Betrachtung sein sollen. Ein Datenverlust auf einem Domänenserver oder dem zentralen Projektserver würde zu sehr hohen Folgekosten führen. Eine Spiegelung der Daten auf einem Backupserver, verbunden mit einer sehr häufigen Aktualisierung, kann zur Lösung dieser Problematik beitragen. Ob ein hardwareseitiges redundantes Speicherkonzept allein ohne weitere Vorkehrungen hinreichend ist, wird bezweifelt, da nach einem Systemausfall nicht in jedem Fall ein logisch konsistenter Informationsbestand vorliegt. Ebenso wäre zu untersuchen, ob eine entfernte redundante Speicherung von Domänenmodellinformationen (Caching) als Maßnahme zur Steigerung der Zugriffseffizienz von Partnern im VE sinnvoll ist, da sich daraus wiederum die üblichen Probleme replizierter Datenbestände ergeben. Die Ausführung derartiger Optimierungsaufgaben sollte nach dem Vorliegen praxisrelevanter Testdaten zur Performance der Gesamtsystemarchitektur sowie zur Art und Stelle aufgetretener Bottlenecks erfolgen.

5.3 Modellüberwachung

5.3.1 Authentisierung und Autorisierung

Bei einer kooperativen Arbeit in Netzwerken ist eine Überwachung des berechtigten Zugriffs auf Informationsbestände grundsätzlich erforderlich. Da im Ergebnis von Bauplanungsprozessen Dokumente mit rechtsverbindlichem Charakter entstehen, kommt dieser Anforderung eine besondere Bedeutung zu.

Damit sind einerseits Authentisierungsmechanismen zu integrieren, um den Zugang zu den Informationsbeständen und Ressourcen auf die Menge der am Planungsprozess beteiligten Personen und Systeme zu beschränken. Im Ergebnis des Authentisierungsverfahrens ist der Entwurfsumgebung bekannt, durch wen der Zugriff erfolgt. Damit ist es ebenso möglich, die Rollen eines auf gemeinsame Ressourcen zugreifenden Beteiligten zu ermitteln.

Andererseits kommt der Realisierung geeigneter Autorisierungsmechanismen eine große Bedeutung zu, um bestimmte Rechte des Zugriffs auf gemeinsame Informationen oder auf Informationen aus Verantwortungsbereichen anderer Beteiligter personenkonkret bzw. nach Rollenzugehörigkeit vergeben zu können.

Für die Realisierung von Autorisierungsmechanismen sind die meisten traditionellen Verfahren nur bedingt geeignet. Verfahren mit der Vergabe von Schreib–, Lese– und Ausführungsrechten für Individuen und Gruppen besitzen weder die notwendige Ausdruckskraft noch die notwendige Flexibilität. Verfahren, die auf Access Control Lists (ACL's) basieren, erfüllen ohne Anpassungen ebenso nicht die Anforderungen für die Realisierung von Groupware–Systemen. So ist es beispielsweise schwierig, Modifikationen eines gemeinsamen Datenbestands zu verweigern, aber Hinzufüge–Operationen zu gestatten. Rechte zur Verwaltung gemeinsamer WYSIWIS–Präsentationen oder zur Erstellung einer Bezugnahme auf Informationsbestände eines anderen Beteiligten sind ebenso schwer abbildbar.

Wichtige Anforderungen an eine effektive Zugriffskontrolle in CSCW–Applikationen sind die Möglichkeit einer dynamischen Zuordnung von Rechten infolge der hohen Änderungsfrequenz sowie eine feingranulare Vergabe von Rechten. Wünschenswert ist ebenso eine Ableitung der aktuellen Rechte aus dem momentan ausgeführten Arbeitsschritt oder der gegenwärtig ausgeübten Rolle. Günstig ist ferner ein einfaches Management der Rechte durch definierte Teammitglieder anstelle externer Systemadministratoren.

Aufgrund der genannten Anforderungen ist es oft erforderlich, dass Groupware–Systeme eine eigene Zugriffskontrolle realisieren. Je nach den konkreten Anforderungen der umzusetzenden Applikation können hierfür adaptierte ACL–Verfahren oder auf Zugriffsmatrizen basierende Mechanismen eingesetzt werden.

Das Matrixverfahren nach Lampson sowie Graham und Denning basiert im wesentlichen auf den Zugriffsrechtszustand beschreibenden Tripeln aus Subjektmenge, Objektmenge und einer Zugriffsrechtematrix sowie Operationen zur Anpassung dieser Tripel.

Dieses Modell wurde von Shen und Dewan [Dewan 98] [Dewan 98a] um Kollaborationsrechte, negative Rechte und um eine Übertragung von Rechten der zugehörigen Subjekt– bzw. Objektgruppe für nichtspezifizierte Matrixelemente erweitert.

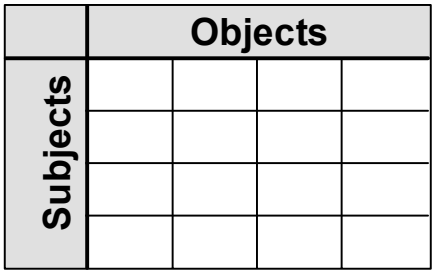

Abbildung 20: Zugriffsmatrix nach Lampson et al.

Eine noch allgemeinere und umfassendere Erweiterung des Matrixverfahrens schlagen Smith und Rodden [Rodden 91] [Smith 93] vor. Deren Zugriffsmodell beinhaltet Nutzer, Rollen, Objekte und Rechte. Die Menge der Rechte ist nicht vordefiniert, sondern kann an die Bedürfnisse der

Applikation bzw. des zu überwachenden Objekts angepasst werden. Für jede Zuordnung von Objekten zu Nutzern bzw. Rollen existiert eine zweispaltige erweiterbare Zugriffsrechte–Matrix. Die erste Spalte repräsentiert die spezifizierten Rechte des Zugriffs durch den Nutzer. Mit der zweiten Spalte wird verwaltet, ob die Ergebnisse des Zugriffs propagiert werden dürfen. Für die Elemente der Matrix schlagen Smith und Rodden bitweise Einträge vor, welche die aktuellen Zugriffsrechte sowie Defaultbelegungen bzw. generelle Verbote oder Erlaubnisse ausdrücken.

Ähnliche Ansätze finden sich ebenfalls in anderen aktuellen Arbeiten, die sich mit der Einführung von Roled Base Access Control (RBAC) in kooperativen Umgebungen befassen [Jaeger 96] [Hoeven 94]. Andere gegenwärtige Forschungsvorhaben beschäftigen sich mit Task based access control (TBAC) [Kang 01]. Derartige aktivitätsorientierte Ansätze gewährleisten eine hohe Flexibilität und können sehr feingranular konfiguriert werden, da Rechte an die Ausführung von Aktivitäten und Teilaktivitäten geknüpft werden. Damit liegen die Stärken dieser Verfahren aber in Umgebungen, in denen ein Workflow Management umgesetzt ist oder gut realisiert werden könnte. Die Anwendung derartiger Verfahren in der Bauwerksplanung, die unter anderem durch eine schwache Prozessstruktur und einen geringen Wiederholgrad charakterisiert wird, erscheint nicht als günstige und prozessadäquate Lösung.

Das Verfahren von Smith und Rodden weist die notwendige Flexibilität und Ausdruckskraft auf, um für in Modellverwaltungssystemen gespeicherte Bauwerksinformationen eine Zugriffs-kontrolle zu realisieren.

Einerseits lassen sich die nötigen Basisrechte wie Lesen, Modifizieren, Anfügen, Löschen ausdrücken, dasselbe gilt ebenso für durch die Groupware–Umgebung erforderlichen Rechte zur Verwaltung synchroner und asynchroner Kooperation.

Ein skalierter Einsatz des Zugriffskontrollmechanismus ist ebenso möglich. Dieser kann sich zwischen den Extremen der globalen Verwaltung der Informationen eines Domänenmodells und der Verwaltung einzelner Objekte der Bauwerksmodellebene bewegen. Der erste Fall ist für Anwendungen als zu grob und unflexibel einzuschätzen, der zweite Extremfall würde einen enormen Verwaltungsaufwand mit hohem Ressourcenbedarf und negativen Auswirkungen auf das Laufzeitverhalten nach sich ziehen.

Die hier vorgeschlagene Lösung basiert auf einer Default–Zugriffskontrollmatrix pro Domänenmodell. Diese Defaultwerte sind für Modellklassen oder deren Objekte überschreibbar. Werden Referenzen auf diese überschriebenen Zugriffsmatrizen entlang von Vererbungs– und Aggregationsbeziehungen und auf die Instanzen dieser Klassen propagiert, ergibt sich eine Kontrolle der Berechtigungen in einer auf die Modellsemantik anpassbaren Form. Um eine flexible, noch feingranularere Verwaltung der Berechtigungen zu erzielen, können den Attributen der Klassen bei deren Definition jeweils ein Grad auf einer abgestuften Permissivitätsskala zugewiesen werden. Dadurch wird es in Verbindung mit dem adaptierten Matrixverfahren möglich, den Zugriff anderer Beteiligter auf kritische Attribute restriktiv zu gestalten, während

die Interaktionsmöglichkeiten mit anderen Attributen offener gehalten werden können. Ebenso ist die Flexibilität für eine häufige Änderung der Rechte vorhanden.

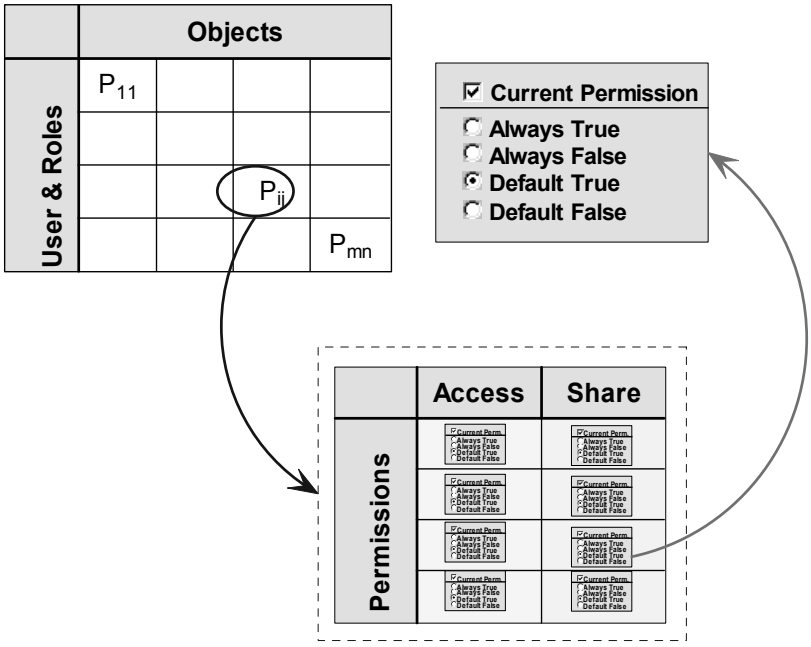

Abbildung 31: Zugriffsverfahren nach Rodden und Smith

Neben einer effektiven Zugriffskontrolle ist es wünschenswert, eine flexible Meta–Zugriffskontrolle zu integrieren, welche die Rechte der Anwender bei der Zuweisung und Weitergabe von Rechten überwacht. Durch die ‚Meta Access Control'–Komponente werden daher die Berechtigungen zum Einräumen und Entziehen von Rechten überwacht und eine komfortable Oberfläche zur Verwaltung der Zugriffskontrolle bereitgestellt. Eine technische Umsetzung ist wiederum durch adaptierte ACL's oder Zugriffsmatrizen möglich.

5.3.2 Sicherung des Dokumentencharakters

Durch die Entwicklungen in der Bauinformatik, den Informations- und Kommunikationstechnologien und der Produktdatentechnik wird heute die überwiegende Anzahl der Planungsaufgaben rechnergestützt bearbeitet. Ebenso sind die technischen Voraussetzungen für eine elektronische Übertragung der Entwurfsresultate gegeben. Besonders im Kontext kooperativer Planungsumgebungen wie der hier beschriebenen wird angestrebt, die synchrone und asynchrone Zusammenarbeit durch die oben genannten Techniken weitgehend zu unterstützen und eine Basis auch für geographisch weit verteilte Partner bereitzustellen.

Ein allgemeines Problem rechner- und rechnernetzwerkbasierter Entwurfsunterstützungssysteme ist dabei die Authentisierung der Entwurfsergebnisse. Eine Authentisierung von Dokumenten ist notwendig, um einerseits die Verbindlichkeit von Informationen zu sichern und andererseits im Konfliktfalle gerichtsverwertbare Beweise über den Inhalt der übermittelten

Dokumente zu besitzen. Bei konventionellen Bauplanungsprozessen werden üblicherweise Papierdokumente vom Bearbeiter oder einem Verantwortlichen der planungsausführenden Firma unterzeichnet, um die Authentizität der Dokumente zu beurkunden. Auch derartige Dokumente sind keineswegs fälschungssicher, beispielsweise können in Zeichnungen leicht Informationen zugefügt werden. Jedoch haben unterzeichnete Dokumente eine lange Tradition im Rechtssystem, so dass sie zumindest als Augenscheinsbeweis und möglicherweise auch als (Privat–)Urkunde genutzt werden können.

Wird auf eine Authentisierung verzichtet, können nach Ott beispielsweise folgende Probleme auftreten [Ott 98]:

- der Empfang von Informationen könnte bestritten werden, daher sind Empfangs-bestätigungen notwendig,
- der Empfänger könnte die Verbindlichkeit oder die Authentizität der Information bezweifeln und sie daher ignorieren,
- der Inhalt der Information könnte bestritten werden, da eine Fälschung des Dokumenteninhalts während der Übertragung nicht endgültig auszuschließen wäre.

Vor allem wäre aber der Nachweis der Verantwortung für Fehler im Planungsprozess oder andererseits für eine korrekte Abwicklung der Aufgaben gefährdet.

Somit sind Maßnahmen von Interesse, welche die Authentizität von elektronisch übermittelten Signaturen sicherstellen. Die rechtliche Seite wird durch das Gesetz zur digitalen Signatur (SigG) gesichert, welches in das Gesetz zur Regelung der Rahmenbedingungen für Informations- und Kommunikationsdienste (IuKDG) eingebunden ist.

Hier soll jedoch kurz die technische Seite der Erzeugung digitaler Signaturen betrachtet werden. Diese werden über Kryptoalgorithmen erzeugt, d.h. zur Authentifizierung werden die Dokumente ver- und entschlüsselt. Die sonst häufig angewandten symmetrischen Verfahren mit einem Schlüssel sind für diesen Zweck ungeeignet, da damit der Empfänger auch die Möglichkeit zur Modifikation der Information hätte. Asymmetrische Kryptosysteme arbeiten hingegen mit zwei verschiedenen Schlüsseln für die Codierung und Decodierung. Häufig wird ein Schlüssel öffentlich zugänglich gemacht, während der andere Schlüssel geheim gehalten wird; diese Verfahren werden Public–Key–Methoden genannt. Die zu übertragende Information wird durch den Sender mit dem geheimen Schlüssel codiert. Der Empfänger kann nun mit dem öffentlichen Schlüssel des Senders das Dokument dekodieren, gelingt dies, ist die Authentizität des Dokuments gesichert.

Aufgrund der ineffizienten Verschlüsselung großer Datenmengen durch Public–Key–Verfahren wird von größeren Dokumenten über eine Einweg–Hashfunktion ein 'Fingerprint' genannter binärer Identifikatorstring des Dokuments erzeugt, der anschließend verschlüsselt wird. Bei üblichen Stringlängen von 100 bis 200 Bit ist die Wahrscheinlichkeit einer unbemerkten

Modifikation des Dokuments oder der Erzeugung eines zweiten Dokuments mit gleichem Fingerprint extrem gering.

Die Erzeugung von digitalen Signaturen für Bauwerkmodellinformationen ist nur für freigegebene Planungsstände, die der Entwerfende als fertiggestellt und konsistent deklariert hat, sinnvoll. Einerseits können nun die digitalen Bauwerksinformationen in einem externen Format, beispielsweise als ASCII–basierte STEP– oder XML–Datei gespeichert werden. Von dieser wird nun ein Fingerprint ermittelt, dieser chiffriert und die so gewonnene Signatur mit dem File übermitelt. Der Empfänger ermittelt nun seinerseits den Fingerprint der übermittelten Datei, vergleicht diesen mit dem übermittelten Fingerprint und decodiert ihn mit dem öffentlichen Schlüssel. Ist dieser Vorgang erfolgreich, ist das Dokument echt und rechtsverbindlich [Anderl 97]. Andererseits ist auch durch Traversieren der Modellstruktur die Erzeugung eines Fingerprints möglich, wenn sichergestellt wird, dass die gleiche Reihenfolge des Besuchs der einzelnen Knoten eingehalten wird. Dies ist im vorgestellten Modellverwaltungssystem realisierbar, da alle Klassen und Instanzen über einen eineindeutigen Identifikationsstring verfügen, der konstant bleibt und somit zur Erzeugung einer lexikographischen Ordnung genutzt werden könnte. Dieser Fingerprint kann analog obigen Vorgehen für die Überprüfung eines gespiegelten Bestandes von Bauwerksmodellinformationen genutzt werden. Der authentisierte Planungsstand muss dann als freigegebener Zustand in die Versionsverwaltung übertragen werden oder darf nur noch lesend zugegriffen werden, um später Zugriff auf genau diese Instanzmenge zu haben.

Auch hier besteht jedoch noch Klärungsbedarf in bestimmten Punkten. Das oben beschriebene Vorgehen basiert auf gesetzlich zugelassenen Vorgehensweisen, eine Kryptoanalyse durch entsprechende Fachleute erscheint trotzdem wünschenswert. Ferner besteht die Problematik von Modifikationen ohne relevante semantische Auswirkungen, beispielsweise beim Ändern einer Schreibweise von alter in neue Rechtschreibung. Derartige Fragen müssen jedoch neben der technischen Untersuchung auch juristisch geklärt werden.

6 Kooperative Bauwerksmodellierung

CSCW-basierte Applikationen im allgemeinen, und damit auch kooperativ nutzbare Bauwerks–Entwurfsumgebungen im besonderen, basieren naturgemäß auf Verteilten Systemen, da der Informationsaustausch zwischen den Groupware–Applikationen auf den Rechnern der Beteiligten unter Nutzung von verschiedenen Schichten der Infrastruktur von Rechnernetzwerken erfolgt. Eine CORBA–basierte Lösung der netzwerkbasierten Kommunikationsproblematik sowie zur Gewährleistung des entfernten Zugriffs auf die relevanten Informationsbestände geographisch verteilter Teams wurde im vorangegangenen Kapitel vorgestellt.

Die Nutzung von Rechernetzwerken, das Auftreten von parallel ablaufenden Prozessen im System sowie die Kooperation einer Anzahl von Beteiligten impliziert die Notwendigkeit der Realisierung bestimmter Mechanismen, um eine korrekte Funktionsweise des Systems zu gewährleisten. Zu diesen gehören unter anderem Techniken zur Überwachung des potentiell parallelen Zugriffs auf gemeinsame Ressourcen und Mechanismen zur Authentisierung der beteiligten Personen sowie zur Autorisierung von deren Zugriffen auf Teile des gemeinsamen Informationsbestandes.

Für viele dieser Aufgaben existieren Lösungen, die für Standardprobleme von Datenbanken, Verteilten Systemen, Mehrbenutzersystemen oder Parallelen Prozessen anwendbar sind. Diese Techniken können jedoch nicht uneingeschränkt auf Groupware–Systeme übertragen werden, da diese teilweise die Spezifik von Groupware–Lösungen nicht hinreichend beachten. Eine Ursache dafür ist, dass Techniken, die für die Interaktion von Systemen geeignet sind, nicht grundsätzlich auf Probleme der Mensch–Maschine–Interaktion übertragbar sind. Die Konsequenz einer Anwendung herkömmlicher Techniken könnte eine unbefriedigende Responsivität der Nutzerschnittstelle oder ein für die Anwender und den Arbeitsprozess inakzeptables Systemverhalten sein.

Weiterhin ist es wichtig, dass die verwendeten Systeme und Techniken in einer CSCW–Umgebung „cooperation aware“ arbeiten. Darunter werden Systeme verstanden, die durch verschiedene Mittel der Schnittstellengestaltung dem Anwender die Präsenz und Aktivität anderer Teammitglieder verdeutlicht. Die Awareness–Eigenschaften der Groupware–Umgebung sind essentiell für den Gruppenprozess und haben damit Auswirkungen auf die Resultate der Teamarbeit, da letztendlich den Beteiligten eine bewusstere Entscheidung über die eigenen, nächsten Aktivitäten im Sinne der Erreichung des Teamzieles ermöglicht wird.

Klassische Mehrbenutzer–Systeme wie DBMS und beispielsweise diverse Betriebssysteme weisen dagegen im allgemeinen die „Kooperations–Transparenz“ genannte entgegensetzte Eigenschaft auf. Diese bedeutet, dass durch das System versucht wird, beim Anwender die Illusion der alleinigen Nutzung des Systems zu erzeugen. Häufig bekommt der Anwender solcher Systeme

nur durch negative Auswirkungen auf seine Aktivitäten wie abfallende Systemleistung oder gesperrte Ressourcen Kenntnis über die Nutzung durch Teammitglieder. Eine Teamarbeit bei solchen Systemen ist meist nur durch implizite Kooperation über das System in Verbindung mit expliziter Kommunikation über externe Kanäle möglich.

Nachfolgend sollen in diesem Kapitel Techniken zur Unterstützung der kooperativen Aspekte des Bauwerksentwurfs betrachtet werden und Techniken zur Umsetzung dieser im Kontext dynamischer Modellverwaltungssysteme vorgestellt werden.

6.1 Basistechniken der Kooperationsunterstützung

6.1.1 Ereignisorientierte asynchrone Kommunikationsmechanismen

Die Kommunikation zwischen den Modellverwaltungskernen und den Domänenclients bzw. den Applikationen wird durch das AKO–Interface definiert und erfolgt daher auf Basis synchroner CORBA–Methodenaufrufe. Analog zur üblichen Vorgehensweise in Verteilten Systemen bzw. speziell beim Distributed Objects Computing hat ein Serverprozess keine Kenntnis über seine konkreten Clients, insofern man die für die puren Aufrufmechanismen nebensächlichen Authentisierungsmaßnahmen außer acht lässt. Infolgedessen kann der Modellverwaltungskern über die AKO–Schnittstelle keine Informationen über zugehörige Clients oder Applikationen erlangen.

Für eine Anzahl von Zwecken, wie beispielsweise beim Modifizieren von Klassen oder Instanzen, beim Aufdecken inkonsistenter Modellzustände, bei Änderungen von WYSIWIS–Präsentationen bei synchroner Kooperation, zur Übermittlung von Nachrichten an Bearbeiter oder zur Übermittlung von Awareness–Informationen ist aber auch eine Übermittlung von Informationen an den Client wünschenswert.

Zur Umsetzung eines derartigen Kommunikationsmechanismus ist der Einsatz verschiedener Basistechnologien denkbar:

- Synchrone CORBA–Methodenaufrufe: Im Grunde können die benötigten Funktionalitäten durch Nutzung von synchronen CORBA–Methodenaufrufen realisiert werden. Nachteilig ist hier jedoch, dass die Empfängerseite im diesem Falle eine vollständige CORBA–Objektinfrastruktur besitzen muss, also dann als Server für den Nachrichtenempfang fungiert. Ein synchroner Aufruf ist ferner ungünstig, da der entsprechende Thread bis zum Empfang der Übermittlungsbestätigung oder einem Timeout blockiert bleibt, was vor allen bei inaktiven Clients zu Problemen mit weiteren zu übermittelnden Nachrichten führen würde, da eine Lösung mit einer Vielzahl von Threads aus Softwareentwurfsgründen abzulehnen ist. Ferner müsste auf jedem Domänenserver eine Liste aller aktiven Domänenclients mit dem dort relevanten Informationsbedarf verwaltet werden.

- Asynchrone CORBA–Methodenaufrufe: CORBA bietet auch die Möglichkeit, asynchrone Methodenaufrufe auszuführen. Bis auf die wegfallende Problematik der Blockierung des sendenden Prozesses bleiben aber die oben genannten Nachteile bestehen.
- Event–Service: Die CORBAservices–Spezifikation beschreibt einen Event–Service genannten asynchronen Kommunikationsdienst, der auf Basis standardisierter Ereigniskanäle arbeitet und verschiedene definierte Variationen beispielweise bezüglich des Kommunikationsverfahrens (pull bzw. push) und der Art der Nachrichtenobjekte (typisiert bzw typfrei) besitzt. Am durch die ORB–Infrastruktur bereitgestellten Eventchannel melden sich beliebig viele Sender und Empfänger an, so dass alle Meldungen an alle angemeldeten Empfänger geschickt werden. Sämtliche Interaktionen erfolgen nach dem Producer–Consumer–Modell, im Falle des Event–Service wird aber die Übermittlung der Botschaften nicht garantiert.
- Notification–Service: Der CORBA–Notification–Service ist eine Erweiterung des Event–Service. Wichtigste Neuerungen sind verbesserte Filtermöglichkeiten für Nachrichten, die Möglichkeit der Definition von als Meldung zu übertragenden Datentypen und Quality of Service (QoS)–Eigenschaften, welche die Übermittlung der Meldungen garantieren. Die Mechanismen der Nachrichtenübermittlung orientieren sich am Eventservice.
- CORBA–fremde Protokolle oder Mechanismen: Es existieren eine Vielzahl weiterer Protokolle zur Nachrichtenübermittlung außerhalb des CORBA–Kontexts wie z.B. SOAP. Eine aus der Nutzung resultierende Mischung von Konzepten für Verteilte Objekte erscheint wenig wünschenswert, ferner besitzen diese Protokolle oft ein ungünstiges Verhältnis der übertragenen Informationsmenge zum erforderlichen Verwaltungsdatenoverhead, so dass keine Vorteile von der Nutzung im gegebenen Kontext zu erwarten wären.

Für das Modellverwaltungssystem des SFB 524 wurde nach Abwägung der genannten Alternativen ein ereignisorientierter asynchroner Kommunikationsmechanismus realisiert, der auf dem CORBA–Notification–Service basiert. Bei jedem zustandsändernden Aufruf an Modellen, Packages, Klassen, Slots oder Relationen bzw. deren Instanzen im Modellverwaltungskern wird geprüft, ob Beobachterobjekte für das jeweilige Modellelement existieren. Ist dies der Fall, wird ein Benachrichtigungsobjekt erzeugt und dem durch den Notitikationsmanager bereitgestellten Ereigniskanal übermittelt. Clientseitig wird die Nachricht an einen Beobachtermanager weitergeleitet, der alle Nachrichten bezüglich des entsprechenden Modellelementetyps empfängt und an die entsprechenden Objekte der Applikation weiterleitet.

Für Clients mit statischen Modellen bietet sich die Generierung von Proxyobjekten (Konnektoren) des überwachten Objekts in den Applikationen an. Diese vermitteln zwischen den applikationsspezifischen Objektstrukturen und der auf Metaebene agierenden AKO–Schnittstelle durch die Bereitstellung von Zugriffspfaden auf die Modellelemente in der üblichen Syntax der genutzten Programmiersprache.

Clients, die auf dynamischen Modellen basieren, nutzen die Schnittstelle des Beobachtermanagers, um über Ereignisse informiert zu werden. Da derartige Applikationen ohnehin zumindest partiell auf Metaniveau agieren müssen, entfällt hier die Motivation zur Schaffung von Konnektoren.

Der hier vorgestellte Mechanismus ist neben der beschriebenen Funktionalität zur Aktualisierung abhängiger Informationspräsentationen oder Applikationen auch für die anderen oben genannten Zwecke einsetzbar. Dazu müssen entsprechend den zu übermittelnden Nachrichten neue Ereignislieferanten und Beobachtermanager in das Modellverwaltungssystem und die Domänenclients eingefügt werden.

6.1.2 Bereitstellung von Group Awareness – Informationen

Wie bereits im Kapitel über CSCW–Techniken dargelegt worden ist, hat die Bereitstellung von Awareness– Informationen positive Auswirkungen auf den Gruppenprozess. Weiterhin unterstützen diese die Teammitglieder häufig bei anstehenden Entwurfsentscheidungen dergestalt, dass das Wissen über Aktivitäten im eigenen Kontext eine bessere Abwägung der Vor- und Nachteile verschiedener Alternativen gestattet. Allerdings muss bei Einsatz von Widgets oder GUI–Windows mit Awarenessinformationen strikt darauf geachtet werden, dass keine Informationsüberflutung auftritt, woraus Restriktionen für die Widgets bezüglich ihrer Anzahl und ihres Typs folgen. Daraus folgt für die Teamarbeit beim Bauwerksentwurf und die Arbeit in Virtual Enterprises, dass die Eignung konkreter Awarenessmaßnahmen nur in Abhängigkeit von der Organisationsstruktur und den Teamgrößen im Virtual Enterprise bestimmt werden kann. Tendenziell wird es sinnvoll sein, bei sehr großen Projekten mit vielen Bearbeitern derartige Informationen nur innerhalb einer Domäne anzubieten. Andererseits kann bei kleinen und gut überschaubaren Projekten die Bereitstellung von Awarenessinformationen bezüglicher aller Projektbeteiligten hilfreich sein.

Für den Bauwerksentwurf erscheinen vor allem Maßnahmen zur Verbesserung der informellen sowie der Workplace– Awareness sinnvoll. Informal Awareness, also Wissen über den Gruppenzustand, sollte über Präsenzindikatoren, also ikonische Darstellungen der angemeldeten Teammitglieder, vermittelt werden. Diese sind einfach zu realisieren, verbrauchen nur sehr wenig zusätzliche Ressourcen und können auf Basis des oben beschriebenen Kommunikationsmechanismus realisiert werden. Eine einfache Darstellung des Login–Zustandes im Team verbunden mit einer Rasterbild-Darstellung der Teammitglieder veranschaulichen Gelegenheiten zu synchroner Kommunikation oder Kooperation. Insofern die Hardwarevoraussetzungen im Form von Webcams verfügbar sind, ist auch eine periodische Aktualisierung des dargestellten Bildes leicht realisierbar.

Für die Unterstützung der kooperativen Arbeit innerhalb einer Domäne erscheinen Radar–Views, also Darstellungen des gesamten Arbeitsbereichs unter Hervorhebung der für einzelne Teammitglieder sichtbaren Bereiche zweckdienlich. Wird die Planung eines Bauwerks durch

Zuweisung von Zonen an einzelne Teammitglieder parallel bearbeitet, können bei Aktivitäten in den jeweiligen Randbereichen Kollisionen vermieden werden. Außerdem lassen sich die Urheber von mit Sperren belegten Teilen des Bauwerksmodells auf diesem Weg leicht ablesen. Alternativ kann auch die "What–You–See–Is–What–I–Do" genannte Technik der vereinfachten Darstellung des Arbeitskontextes mit einer ständigen mittigen Platzierung des Cursors des Anwenders eingesetzt werden. Diese Technik ist auch gut geeignet, wenn bei Aufgaben mit begrenzter Komplexität und kleinen Planungsteams Awarenessinformationen auch über Domänengrenzen hinweg verbreitet werden sollen.

Ein weiteres innerhalb einer Domäne für Rundrufzwecke einsetzbares Awareness–Widget ist das Tickertape, ein Lauftext zur Darstellung kurzer Textnachrichten. Diese Technik ist leicht realisierbar, sollte aber nicht extensiv genutzt werden, da eine zu häufige Darstellung von Nachrichten ablenkend wirken würde.

6.2 Asynchrone Kooperation

Der überwiegende Zeitanteil der Aktivitäten im Bauplanungsprozess wird in asynchroner Kooperation verbracht. Das heißt, die Bearbeiter einzelner Planungsaktivitäten arbeiten selbständig und üblicherweise allein an der Erfüllung ihrer Aufgaben, bis diese abgeschlossen sind. Unterbrechungen der Tätigkeiten treten auf, wenn Konfliktklärungen bzw. Absprachen zu treffen sind oder Informationen von anderen Teammitgliedern beschafft werden müssen.

Daher ist es zur Unterstützung asynchroner Kommunikationsphasen vor allem wichtig, geeignete Techniken zur Kommunikation und Koordination bereitzustellen und die Bearbeiter über Änderungen, Änderungsanfragen und divergierende Entwurfsintentionen bezüglich der Objekte ihres Verantwortungsbereichs zu informieren.

6.2.1 Asynchrone Kommunikations– und Koordinierungstechniken

Allgemein anwendbare, nicht domänenspezifische, asynchrone Kommunikationstechniken haben einen fortgeschrittenen Reifegrad erreicht und gelten als die weitverbreitetsten Groupware–Applikationen. Daher steht die Untersuchung und Entwicklung neuer derartiger Techniken nicht im Blickpunkt der vorgestellten Forschungsprojekte. Da aber das Vorhandensein effektiver asynchroner Kommunikationsmechanismen eine unabdingbare Voraussetzung für eine effektive Unterstützung des Bauwerksentwurf durch CSCW–Methoden ist, soll kurz auf diese Thematik eingegangen werden.

Die komplexe Problematik von Bauplanungsprozessen und die Arbeitsweise in Virtual Enterprises implizieren einen hohen Kommunikations– und Koordinierungsbedarf zwischen den Beteiligten. Auch hier gilt, dass Kooperation auf Koordination beruht und für letztere Kommunikation notwendig ist. Einige Beispiele für Motivationen zur Nutzung der Kommunikationsinfrastruktur sind:

- das Treffen von Absprachen mit anderen Beteiligten zur Entwurfskoordinierung,
- das Verbreiten von Vorschlägen und Ideen für den Planungsprozess,
- das Hinweisen auf Probleme bei der Planungsausführung,
- die Lösung von Planungskonflikten aufgrund divergierender Intentionen,
- die Bitte um Erläuterung spezieller Entwurfsentscheidungen,
- die Bitte um Bereitstellung nicht im Bauwerksmodell enthaltener Hintergrundinformationen,
- die Bekanntgabe des Erreichens von Meilensteinen im Planungsfortschritt,
- die Archivsuche zur Nachverfolgung der Entstehungsgeschichte und der durchlaufenen Zwischenversionen im Team gewählter Lösungen.

Die obigen Beispiele zeigen, dass zur Lösung der Kommunikations- und Koordinierungsaufgaben sowohl explizite als auch implizite Kommunikationsmechanismen benötigt werden. Während für implizite Kommunikation ein asynchrones Modell die natürliche Vorgehensweise ist, kann ein Teil der expliziten Kommunikationsaufgaben sowohl durch synchrone als auch asynchrone Modelle durchgeführt werden.

Bei der Auswahl des Kommunikationsmodells für explizite Kommunikation müssen deren Vor- und Nachteile abgewogen werden. Synchrone Kommunikation führt in vielen Fällen zu einer schnelleren Problemlösung und entspricht eher der traditionellen Arbeitsweise im Planungsprozess. Asynchrone Kommunikationsformen haben aber deutlich geringere negative Auswirkungen auf die Kerntätigkeit des Empfängers, da dieser den aktuellen Arbeitsschritt beenden kann, bevor er sich den Kommunikationsaufgaben widmet. Bei geographisch verteilten Teams kann eine synchrone Kommunikation aufgrund von Zeitverschiebung nicht immer möglich sein, auch dann müssen asynchrone Techniken genutzt werden.

Zur Unterstützung asynchroner Kommunikation ist die Bereitstellung eines leistungsfähigen Mailsystems erforderlich. Dieses wird üblicherweise als hauptsächliches Medium der expliziten peer–to–peer Kommunikation genutzt. Es ist anzustreben, in den zentralen Projektserver einen Mailserver zu integrieren. Dieser wäre in der Lage, eine Zuordnung von Rollen zu Personen vorzunehmen, so dass Emails auch an Rollenbezeichnungen, z.B. 'Baugrundgutachter' oder 'Ansprechpartner Tragwerksplanung' adressiert werden können. Dies hat den großen Vorteil, dass auch bei komplexen Projekten im Falle der Änderung der Zuordnung von Personen zu Rollen infolge Vertretung oder Teamänderung Mails schnell und zuverlässig zugestellt werden. Ferner ist hier auch der Betrieb eines Listservers möglich, der je nach Listenadresse Mails an eine Gruppe von Interessenten, z.B. alle Elektroplaner, alle Beteiligten an der Planung eines spezifischen Bauabschnitts oder alle Interessenten an den Ergebnissen der HLS–Planung zu senden.

Implizite Kommunikation im Planungsprozess wird über verschiedene Kanäle und Medien ausgeübt. So gehört die Einspeisung von Bauwerksinformationen in die Domänenmodelle und die

Kenntnisnahme dieser Daten durchaus zu dieser Kommunikationsform. Die Gesamtheit der Komponenten der Systemarchitektur gemeinsam mit den darin verwalteten Informationen stellen einen gemeinsamen Informationsraum im CSCW–Sinne dar.

Darüber hinaus ist die Bereitstellung eines textbasierten impliziten Kommunikationsmediums durch den zentralen Projektserver sinnvoll. Dies kann durch ein geeignetes Bulletin Board System (BBS) oder verteiltes Hypermediasystem, also eine Intranet-Lösung, geschehen. Dadurch können Planungsfragen im Team diskutiert und der Lösungsverlauf später nachvollzogen werden. Ebenso können durch dieses System allgemeine Informationen, die sich schwer einzelnen Klassen der Domänenmodelle zuordnen lassen, verwaltet werden. Derartige Lösungen gewinnen als Projektplattformen oder internetbasierte Projekträume auch in der Planungspraxis zunehmend an Bedeutung.

6.2.2 Modellmapping: Konzept und Mechanismen

Neben den oben betrachteten allgemein nutzbaren asynchronen Kommunikationsmechanismen ist es für eine effektive Unterstützung der kooperativen Arbeit im Bauwerksentwurf notwendig, Bearbeitern Mechanismen zur Verfügung zu stellen, die bei der Behandlung von Modifikationen an Informationen, auf denen bereits erbrachte Planungsleistungen basieren, assistieren. Diese Techniken stellen im CSCW–Sinne planungsspezifische asynchrone Groupware–Lösungen dar. Weiterhin besitzen diese Techniken Ansätze einer teamweiten Überwachung und Wiederherstellungshilfe der Konsistenz der Gesamtinformationsbestände bezüglich des aktuellen Bauplanungsvorhabens.

Wie im vorangegangenen Kapitel ausgeführt worden ist, bestehen zwischen den Partialmodellen der Bearbeiter häufig Verknüpfungen genannte Beziehungen in der Art, dass Objekte des einen Modells eine Entsprechung in anderen Domänenmodellen finden. Derartige Beziehungen werden als Verknüpfungstypen explizit zwischen den Klassen der domänenspezifischen Taxonomien angegeben. Diese Relationen werden durch Erbschaft an die Instanzen übertragen. Beispielsweise kann ein Attribut für den Architekten ein lichtes Raummaß bedeuten, während für den Tragwerksplaner dieser Wert die Spannweite einer Decke beschreibt und für den Baukalkulator die Länge einer Wand angibt. In diesen Beispiel würde also eine entsprechende Verknüpfung zwischen einer Decken– oder Trägerklasse des Tragwerksmodells und einer Klasse des raumorientierten Architekturmodells bestehen.

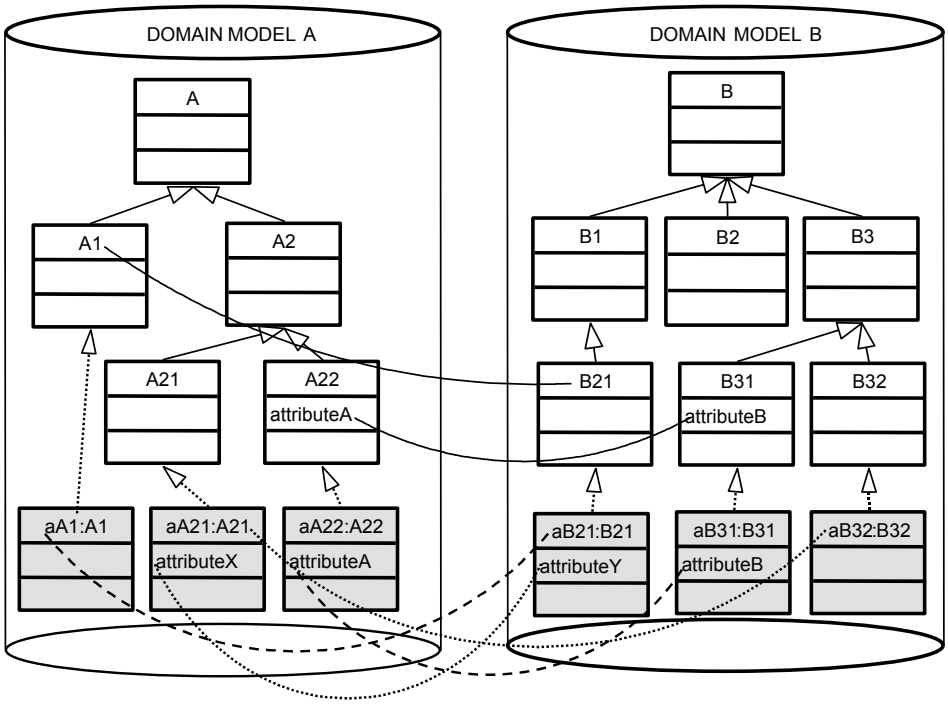

Abbildung 1: Relationen zwischen Domänenmodellen

In einigen Fällen kann bei der Instanziierung der Objekte des Bauwerksmodells die Verknüpfung durch das System aufgebaut werden. Dieses Vorgehen ist vor allem dann möglich, wenn die Eindeutigkeit der Zuordnung der Relationen gesichert ist. In den anderen Fällen muss das 'Mapping' zwischen den einzelnen Teilmodellen durch die Bearbeiter im Planungsprozess assistiert durch die Planungsumgebung bewerkstelligt werden, indem diese die assoziierten Attribute und Objekte in einer geeigneten graphischen oder alphanumerischen Präsentation auswählen. Eine Definition von Verknüpfungen in den Domänenmodellen (Verknüpfungstypen) ist während des Planungsprozesses, also zur Laufzeit möglich. Dies ist notwenig, um auf Informationsanforderungen an ebenfalls dynamisch definierte Klassen reagieren zu können.

Die beschriebenen Relationen können für verschiedene Mechanismen der Informationsübertragung zwischen Domänenmodellen genutzt werden. Welcher Modus angewendet wird, kann pro Verknüpfung festgelegt werden, daher können in einem Domänenmodell mehrere der nachfolgend beschriebenen Modi genutzt werden:

- Modifikations–Notifikation: Der Notifikationsmodus ist eine passive Vorgehensweise, bei denen das System Bearbeiter über Änderungen an fremden, aber für sie relevanten Domänenmodellen informiert. Diese Modifikationen können sowohl die Domänenmodelle betreffen, also beispielsweise das Erzeugen, Löschen oder Modifizieren von Klassen bedeuten, als auch hauptsächlich über Modifikationen der aktuellen Projektinformationen, wie das Erzeugen oder Löschen von Instanzen oder die Änderung von Attributwertbelegungen, informieren. Dem Empfänger der Modifikations–Notifikation steht nach der Kenntnisnahme des Inhalts der Meldung und der Abwägung entsprechender fachlicher Aspekte frei,

entsprechende Änderungen in seinem Verantwortungsbereich vorzunehmen oder die Meldung wegen Irrelevanz oder Geringfügigkeit zu ignorieren. Wichtig ist hier eine adäquate Präsentation der auftretenden Informationen. Günstig erscheint hier eine listenförmige Darstellung in einem speziellem GUI–Window, eine Präsentation durch häufig auftretende Pop–Up–Windows wäre aufgrund der Störung des Bearbeiters und des Fehlens einer späteren Nachvollziehbarkeit nicht geeignet.

- Modifikations–Propagation: Der Propagationsmodus ist eine aktive, wertmodifizierende Technik. Diese überträgt Änderungen aus fremden, aber für die eigene Planungstätigkeit relevanten Domänenmodellen in die Projektinformationen des Bearbeiters. Dieser Mechanismus kann für Änderungen an Attributwertbelegungen sowie für Lebenszyklusoperationen an Instanzen wie Instanz-Erzeugung und –Löschung genutzt werden. Neben einer einfachen Wertübertragung ist auch eine Wertüberführung durch eine entsprechende Vorschrift, welche die numerischen Abhängigkeiten beschreibt, möglich. Aufgrund möglicher Auswirkungen darf dieser Mechanismus durch das System nur ausgeführt werden, wenn der verantwortliche Bearbeiter diesen Modus explizit freigegeben hat. Weiterhin muss der Bearbeiter über ausgeführte Informationspropagationen analog zum vorher beschriebenen Mechanismus informiert werden.

- Informationsgenerierung: Der Informationsgenerierungsmodus ist ein aktiver Mechanismus zur Herstellung eines Initialzustandes der Instanziierung eines Domänenmodells. Nach Abschluss und Freigabe der Planungstätigkeiten einer Fachdisziplin können relevante Informationen in die Projektdatenbestände nachgelagerter Aktivitäten überführt werden, bevor diese Tätigkeiten starten. Somit kann der Informationsgenerierungsmodus mit einer Online-Variante des CAD–Filekonverters in der konventionellen rechnergestützten Bauplanung verglichen werden.

An dieser Stelle soll nicht tiefer auf die Details von Verknüpfungsmechanismen eingegangen werden, da diese der dedizierte Gegenstand parallellaufender Forschungsarbeiten und dazugehöriger Publikationen [Willenbacher 00] [Willenbacher 01] im Kontext des SFB 524 sind.

Im Rahmen der genannten Forschungsarbeiten sind Verknüpfungstechniken auf der Basis eines Multiagentensystems (MAS) realisiert worden. Aufgrund der Eigenschaften von mobilen und stationären Agenten ist diese Technik aus der Sicht des Systemdesigns gut geeignet. Aus Sicht des Autors steht den unbestritten Vorteilen der MAS–Technik ein außerordentlich hoher Ressourcenbedarf als leichtes Manko gegenüber.

Eine nachfolgende Untersuchung des CORBA–Notification–Service auf Anwendbarkeit im genannten Kontext erscheint zweckmäßig, da dieser aufgrund seiner Herkunft für die Verwaltung von Telekommunikationsnetzen für die performante Übertragung kleinerer Informationseinheiten zwischen einer Vielzahl von Sendern und Empfängern entworfen ist und die Anzahl der zusätzlich erforderlichen Prozesse überschaubar bleibt. Ferner besteht derzeit auch weiterer Forschungsbedarf bezüglich eines intelligenten Assistierens der

Bauplanungsumgebung beim Erstellen von Verknüpfungen, hier sollten zukünftig zum Beispiel Techniken, die auf Heuristiken wie Nachbarschaftsbeziehungen, einem CBR–Repository für frühere Verknüpfungsentscheidungen bzw. Dictionaries zum Auffinden von synonymen oder ähnlichen Begriffen in Domänemodellen beruhen, näher betrachtet werden.

6.3 Synchrone Kooperation

Für die Koordination der kooperativen Arbeit oder zur Lösung von Konflikten infolge gegenläufiger Entwurfsintensionen sind Phasen synchroner Kooperation notwendig. Häufig liegt bei Arbeiten in Virtual Enterrprises eine geographische Verteilung des Entwurfsteams vor. Die Organisation von Face–to–face–Meetings von Verantwortlichen mehrerer Fachdisziplinen ist sowohl mit Kosten als auch mit Zeitaufwand für die Reise verbunden und kann kaum spontan beim Auftauchen von Problemen erfolgen. Somit ist eine Unterstützung von entfernter synchroner Kooperation durch die rechner- und netzwerkbasierte Entwurfsumgebung als wünschenswert anzusehen. Dabei ist es erforderlich, den beteiligten Teammitgliedern in angemessener Weise den gemeinsamen Kontext der Arbeit am Bauwerksentwurf darzustellen. Soweit realisierbar, sollte das direkt in der Entwurfsumgebung geschehen, da einerseits der Umgang mit diesem System den Anwendern vertraut ist und andererseits Resultate der verteilten Teamsitzung sofort in das Bauwerksmodell einfließen können.

Neben den genannten CSCW–Techniken zur Unterstützung entfernter synchroner Kooperation ist der Einsatz lokaler synchroner Groupware in bestimmten Situationen, wie bei initialen face–to–face–Meetings zum Beginn der Planungstätigkeiten günstig. Derartige Techniken sollen jedoch hier im folgenden nicht weiter im Blickpunkt stehen.

6.3.1 Synchrone Kommunikationstechniken

Für adäquate Groupware–Applikationen zur Unterstützung der Phasen synchroner Kooperation im Bauwerksentwurf sollten die in den nächsten Abschnitten beschriebenen, entwurfs-spezifischen Techniken zur Schaffung eines gemeinsamen Kontexts genutzt werden.

Zur Bereitstellung einer Umgebung, in der sich entfernte synchrone Kooperation möglichst analog zu einem face–to–face–Meeting ausführen lässt, sind mehrere Kommunikationskanäle notwendig (siehe Kapitel CSCW–Techniken). Für diesen Zweck müssen nicht unbedingt neue, entwurfsspezifische Mechanismen realisiert werden, da bewusst ausgewählte Standardlösungen für diesen Zweck hinreichend sind. Im Falle der Nichtverfügbarkeit entwurfsspezifischer Lösungen ist eine auf Standardtechniken beruhende Lösung höherwertiger als überhaupt keine synchrone Groupware einzuschätzen, jedoch geht die dann gegebene Unterstützung nicht über das Stadium eines Notbehelfs hinaus, da entweder notwendige Kommunikationskanäle fehlen oder die Interaktion im Team nur sehr eingeschränkt oder starren Regeln unterworfen ist.

Zu den Standardtechniken für synchrone CSCW–basierte Systeme, die beim Vorhandensein entsprechender Hardware und Steuersoftware leicht in eine Entwurfsumgebung integriert und nutzbringend angewendet werden können, gehören:

- Audiokanäle: Eine synchrone Audioverbindung ist ein unverzichtbares Hilfsmittel bei der synchronen Kooperation. Einerseits bildet dieses Medium bei gemeinsamen Interaktionen mittels einer gemeinsamen WYSIWIS–Präsentation des Entwurfssubjekts die Basis der Ausbildung einer durch soziale Protokolle gesteuerten Tätigkeit. Andererseits wird durch diesen Kanal auch verbal übermittelte Semantik bezüglich des Entwurfssubjekt parallel übermittelt, die durch Gestikulieren mit dem Telepointer oder durch Eingriffe auf die Präsentation bzw. den Entwurfsgegenstand nur umständlich zu transportieren wäre.
- Desktop-Videokonferenztechniken: Anstelle eines Audiokanals kann auch ein Desktop-Videokonferenzsystem genutzt werden. Dieses besitzt typischerweise einen oder mehrere Audiokanäle und in GUI–Windows dargestellte Videodarstellungen der Teilnehmer an der Entwurfssitzung. Die übermittelbare fachbezogene Semantik bezüglich des Bauplanungs-projekts unterscheidet sich nicht wesentlich von der eines Audiokanals. Sehr vorteilhaft ist jedoch die parallele Übermittlung nonverbaler Informationen, die in einer face–to–face–Umgebung ebenfalls meist unbewusst ausgetauscht und ausgewertet werden. Diese nonverbalen Informationen bedeuten einen enormen Zuwachs an social awareness, also Hintergrundinformationen über Motivation und emotionalen Zustand im Team, fördern die sozialen Protokolle bei der Interaktion am Planungsgegenstand und tragen dadurch zu einer besseren Akzeptanz der kooperativen Entwurfsumgebung bei.
- Textbasierte Kommunikationskanäle: Kanäle zur textbasierten Kommunikation orientieren sich am Beispiel des Internet–Relay–Chats (IRC). Der Interaktionsmodus nimmt eine Zwischenstellung zwischen asynchroner und synchroner Kommunikation ein, da Reaktionen auf Anfragen üblicherweise im Bereich von einigen Sekunden bis zu mehreren Minuten erfolgen. Damit ist die Interaktionsgeschwindigkeit spürbar langsamer als bei verbaler Kommunikation, aber schneller als bei einer typischen Nutzung von Email. Aufgrund der einfachen Realisierbarkeit eignet sich die Technik gut als weiterer Mechanismus zur Steigerung der Teamawareness, andererseits hat sich diese Technik im GroupPlan–Projekt auch für Testzwecke und die Konfiguration der Verteilten Umgebung bewährt.
- Shared Whiteboards: Shared Whiteboards sind kooperativ nutzbare rasterbasierte Zeichentools, die nach dem Vorbild einer Wandtafel gemeinsam zum Zeichnen von Skizzen genutzt werden. Da diese Techniken für einfache Prinzipskizzen schneller und weniger formell als in der Bauwerksmodellierumgebung nutzbar sind, können Shared Whiteboards in Situationen wie beispielweise der Koordinierung späterer Entwurfsaktivitäten eingesetzt werden.
- Window Sharing Systeme: Es existieren aktuelle Projekte zur Untersuchung der Kooperation in Virtual Enterprises, die auf der Nutzung von Window Sharing Systemen wie Microsoft

NetMeeting basieren. Diese Techniken replizieren den Inhalt einzelner Fenster oder den gesamten Desktop auf die Bildschirme der Sitzungsteilnehmer. Nachteilig bei dieser Vorgehensweise ist, dass alle Teilnehmer nur eine gemeinsame Darstellung einer Applikation nutzen können und die Übergabe der Kontrolle nur durch ein Protokoll des Window Sharing Systems erfolgen kann. Damit sind die Interaktionsmöglichkeiten im Team stark eingeschränkt.

6.3.2 Techniken zur Synchronisation der Modellzugriffe

Synchrone kooperative Arbeit erfordert effektive Concurrency–Control Mechanismen, die einerseits die Integrität des gemeinsamen Informationsbestandes sichert, andererseits die Tätigkeiten am Bauwerksmodell nicht unnötig beeinträchtigt. Traditionelle Techniken sind meist für die Nebenläufigkeitskontrolle für parallele Prozesse in Rechnersystemen konzipiert und beabsichtigen häufig, nur einen aktiven Prozess mit Modifikationsabsichten am gemeinsamen Datenbestand zuzulassen. An Techniken, die in CSCW–Umgebungen zum Einsatz kommen, sind spezielle Anforderungen zu stellen, die nicht grundsätzlich durch sämtliche traditionelle Verfahren erfüllt werden. Die hier zum Einsatz kommenden Verfahren sollen unter anderem eine echte synchrone Arbeit zulassen, eine gute Responsivität des Systems garantieren, eine hohe Robustheit besitzen, auch für Wide Area Networks geeignet und auf die Bedürfnisse der ablaufenden Teamprozesse ausgerichtet sein. Beispielsweise wäre eine zeitweise Blockierung sämtlicher Aktivitäten von Benutzern nicht wünschenswert. Längere Wartezeiten auf Systemreaktionen beispielsweise infolge komplizierter Mechanismen zur Anforderung von Sperren tragen nicht zu einer guten Systemakzeptanz beim Anwender bei. Das Zurücksetzen der Resultate einer größeren Anzahl von Arbeitsschritten infolge einer abgebrochenen langen Transaktion ist ebenfalls inakzeptabel.

Wie im Kapitel über CSCW–Techniken dargelegt wurde, kann grundsätzlich zwischen optimistischen und pessimistischen Verfahren zur Nebenläufigkeitskontrolle unterschieden werden. Optimistische Verfahren lassen zunächst alle Zugriffe auf gemeinsame Ressourcen zu. Tritt dabei ein Konflikt auf, muss dieser erkannt und anschließend die entstandenen Auswirkungen behoben werden. Optimistische Verfahren sind besonders für CSCW–Anwendungen geeignet, in denen Konflikte relativ selten auftreten und sich mit herkömmlichen Verfahren kein befriedigendes Interaktionsverhalten erzielen lässt, was durchaus auf kooperative Entwurfsumgebungen zutrifft. Nachteilig sind bei diesen Verfahren allerdings der höhere Realisierungsaufwand und der geringere garantierbare Grad der Datenkonsistenz. In kooperativen Entwurfsumgebungen, also CSCW–Systemen für "Workgroup Computing" und "Gemeinsame Informationsräume", sollten optimistische Transaktionsverfahren eingesetzt werden, da pessimistische Verfahren nicht für lange Transaktionen wie beispielsweise kooperative Dokumentenbearbeitung geeignet sind. Diese Verfahren prüfen erst vor Abschluss der Transaktion, ob Anzeichen für Konflikte vorliegen. Dies wird den betroffenen

Teammitgliedern mitgeteilt, die dann über Abbruch der Transaktion oder manuelle Behebung der Auswirkungen entscheiden können.

Für die Unterstützung verschiedener Aspekte der asynchronen Kooperation im Bauwerksentwurf sind Lock–Verfahren gut geeignet. Diese bewirken eine Sperrung von Ressourcen, die sich bereits in Benutzung befinden. Je nach konkreter Implementierung werden sämtliche Zugriffe auf die betroffene Informationseinheit oder Ressource verweigert oder nur nichtmodifizierende Zugriffe zugelassen. Lock–Verfahren besitzen die Eigenschaft der „Group–Awareness", da die Verursacher von Sperren durch andere Teammitglieder ermittelbar sind und gesperrte Informationen je nach genutztem Präsentationstyp vergleichsweise einfach an der Nutzerschnittstelle beispielsweise durch die Nutzung definierter Farben oder Icons kenntlich gemacht werden können. Ein Problem der Anwendung von Lock–Verfahren sind irrtümlich aufrecht erhaltene Sperren, die andere Teammitglieder über längere Zeiträume behindern können. Es existieren verschiedene Ansätze zu dessen Lösung, so ist eine Aufhebung nach einer verstrichenen Zeit der Inaktivität des Lock–Inhabers möglich. Ebenso können aber auch Sperren grundsätzlich nur für festgelegte Zeiträume vergeben werden, nach denen sich der ehemalige Inhaber neu um eine Lock–Zuweisung bemühen muss. Lock–Verfahren sind für verschiedene Grade der Granularität anwendbar, als Extrema können wiederum gesamte Domänenmodelle für eine äußerst grobe Sperrung bzw. einzelne Objekte oder Attribute in diesen für eine außerordentlich feingranulare Sperrung angesehen werden. Für die Nebenläufigkeitskontrolle für in Modellverwaltungssystemen gespeicherte Bauwerksinformationen ist eine Propagation von Locks entlang von Aggregationsbeziehungen ein geeignetes Mittel, um bestimmte Modellbereiche zu sperren. Beinhaltet das Domänenmodell beispielsweise die konstruktive Bauwerksgliederung, lassen sich Gebäudeteile, Etagen aber auch kleinere Bauteile sperren. Eine weitere zweckmäßige Vorgehensweise ist das Anbringen von Sperren an Objektmengen, die sich durch Anfragen an den im vorangegangenen Mechanismus zur Auswertung von Positions–, Material– und Funktionssignaturen ergeben, so dass sich Objekte, die durch eine Fachdisziplin bearbeitet werden, beschränkt auf definierte Bauwerksteile sichern lassen.

Für Entwurfskonferenzen, also Phasen der verteilten synchronen kooperativen Arbeit am gemeinsamen Entwurfsgegenstand, sind ebenfalls effektive Concurrency–Control–Techniken erforderlich. In diesen Phasen müssen derartige Mechanismen sowohl für gemeinsame Informationsbestände, als auch für andere gemeinsam genutzte Ressourcen realisiert werden. Insofern die vorliegende Relaxierung der WYSIWIS–Präsentation der aktuellen gemeinsamen Entwurfsgegenstände das Kongruenz–Constraint nicht oder nicht signifikant verletzt, müssen auch Operationen, die diese betreffen, serialisiert werden.

Für derartige Aufgaben können Floor–Passing–Verfahren eingesetzt werden. Diese gehören zu den sogenannten ‚Turn–Taking–Protocols' und sorgen für eine temporal wechselnde Zuordnung der Rechte, indem das Vorbild von face–to–face–Konferenzen auf verteilte Konferenzen übertragen wird. Wie bei diesen hat nur ein Teilnehmer die Möglichkeit der aktiven Nutzung der

Ressourcen, wobei ihm diese nach einer gewissen Zeit oder durch einen Chairman, der in diesem Kontext ‚Facilitator' genannt wird, entzogen werden kann. Die Berechtigung zur Benutzung der geschützten Ressourcen wird durch den Besitz eines ‚floor–tokens' ausgedrückt, der bei expliziten Verfahren durch die Teilnehmer und bei impliziten Verfahren durch das System weitergereicht wird. Floor–Passing–Verfahren sind relativ leicht implementierbar, wirken aber durch die Beschränkung synchroner Tätigkeit im Team einschränkend.

Diese Einschränkungen können unter Umständen durch den Einsatz von Transformations-verfahren überwunden werden. Diese Verfahren sind anwendbar, wenn die Daten des aktuellen Arbeitsgegenstandes repliziert werden können und eine eng gekoppelte, synchrone Teamarbeit, wie bei kooperativer Dokumentenbearbeitung, im weitesten Sinne vorliegt. Der Grundgedanke dieser Verfahren ist, dass alle Operationen zuerst lokal ausgeführt werden und dann zusammen mit einem Statusvektor an alle anderen Teilnehmer gesendet werden. Die Empfänger können durch einen Vergleich mit einem lokalen Vektor den eigenen Abarbeitungszustand ermitteln. Sind die Statusvektoren äquivalent, wird die Operation sofort ausgeführt, handelt es sich um eine ‚zukünftige' Operation, wird sie vorerst in eine Warteschlange zurückgestellt. Wurde eine ‚zurückliegende' Operation empfangen, wird diese transformiert, um die Auswirkungen der veränderten Reihenfolge auszugleichen. Dies geschieht unter Nutzung einer quadratischen Transformationsmatrix, deren Elemente die anzuwendenden Transformationsfunktionen darstellen, nach deren Anwendung auf die empfangene Operation und die anschließende Ausführung der transformierten Operation dasselbe Ergebnis wie bei der Ausführung der Operationen in ihrer korrekten Reihenfolge entsteht. Der Nachteil dieser Verfahren liegt in der Tatsache, dass die Aufstellung dieser Transformationsmatrix nicht trivial ist, besonders wenn eine hohe Anzahl von Operationen vorliegt. Inwiefern dieses Problem mit vertretbarem Aufwand für CSCW–Applikationen im Bauwerksentwurf gelöst werden kann, sollte in Zukunft näher untersucht werden.

6.3.3 Realisierung von WYSIWIS

Eine entfernte Präsentation von Projektinformationen, also von Bauwerksteilen in einer für bestimmte Fachdisziplinen adäquaten Darstellungsweise für mehrere Teammitglieder ist zur Schaffung eines gemeinsamen Kontexts in Phasen synchroner Kooperation zu realisieren.

Sehen dabei sämtliche Meeting–Teilnehmer absolut exakt denselben Bildschirminhalt, spricht man von striktem 'What You See Is What I See' (WYSIWIS). Dieser identische Meeting–Kontext entspricht zwar weitgehend einem konventionellen Face–To–Face–Meeting, führt aber häufig zu Problemen. Wie oben dargelegt wurde, kann es beim strikten WYSIWIS leicht zum sogenannten ‚Scroll War' oder zum ‚Window War' kommen, wenn die Teilnehmer der entfernten Sitzung unterschiedliche Informationen bzw. GUI–Windows betrachten möchten. Eine Unterstützung von verteilten Sitzungen im Bauwerksentwurf nach WYSIWIS ist ohne eine parallele Audio– oder Viedeokonferenz–Verbindung zur Absprache der nächsten Aktivitäten nicht praktikabel.

Weiterhin wird oft der identische Kontext als einschränkend und nicht den rollenspezifischen Anforderungen entsprechend empfunden. Abschwächungen vom strikten WYSIWIS führen individuelle Views auf die Informationen oder private Arbeitsbereiche ein. Nachteilig ist, dass damit eine Reduzierung der Group Awareness einhergeht.

Für den Bauwerksentwurf sind relaxierte WYSIWIS–Präsentationen ein adäquates Mittel zur Schaffung eines gemeinsamen Kontexts in synchron ausgeführten Planungsphasen. In GroupPlan wurden WYSIWIS–Views realisiert, welche die gemeinsame Darstellung in einem frei plazierbaren GUI–Window präsentiert. Als Fensterinhalt kann wahlweise eine identische dreidimensionale Visualisierung von Bauwerksinformationen oder verschiedene, eigenständig aus den realisierten Präsentationstypen gewählte Darstellungen genutzt werden. Dabei ist der Grad des geschaffenen gemeinsamen Kontexts in der ersten Variante als hoch und in der zweiten Variante als geringer einzuschätzen, allerdings wirkt die gemeinsame Präsentation auch beschränkender, da keine individuellen Interaktionen mit der Visualisierung wie das kurzfristige Betrachten der Darstellung mit einem veränderten Zoomgrad oder Kamerastandpunkt durch nur einen Sitzungsteilnehmer möglich ist. Das zugrundeliegende Konzept lässt grundsätzlich auch andere Typen von Präsentationen für die beschriebene weniger relaxierte Form des WYSIWIS zu. Die Motivation für die Umsetzung der auf alternativen Views beruhenden stark gelockerten synchronen WYSIWIS–Darstellung liegt in der Tatsache, dass im allgemeinen Personen mit unterschiedlichen Rollen auch unterschiedliche Präsentationstypen bevorzugen.

Die in GroupPlan realisierten synchronen Modellviews relaxieren in jedem Fall das Kongruenz–Constraint, da im Normalfall nur Subgruppen der Menge der Beteiligten am Bauwerksentwurf synchron an einer Koordinierungs- oder Problemlösungsaufgabe arbeiten. Durch die individuellen Screenlayouts und eine selektierbare Cursordarstellung wird bei einer Betrachtung im engeren Sinne auch das Raum–Constraint verletzt. Im Falle der alternativen Views wird auch das Kongruenz- Constraint aufgegeben. Der oben genannten Problematik des Verlustes an Awareness wird durch den Einsatz der im nächsten Kapitel genannten Telepresence–Techniken entgegengesteuert.

Die genutzte Systemarchitektur besitzt Parallelen zur oben vorgestellten synchronen zentralisierten CSCW–Architektur mit Konferenzkomponente. Die darzustellenden Bauwerks-informationen sind im Modellverwaltungssystem der Domänen auf dem Domänenserver gespeichert. Durch ein CORBA–basiertes, aber letztendlich für diesen Anwendungszweck spezifisches Protokoll werden die darzustellenden Informationen zu den interessierten Domänen-clients übertragen, ebenso werden auf diesem Wege die notwendigen Informationen für die Synchronisierung der Visualisierung übermittelt.

Bei der Umsetzung der WYSWIS–Darstellungen wird die oben beschriebene Modell–View–Controller (MVC)–basierte Präsentation genutzt. Die damit verbundene Trennung von Datenabstraktion und Informationspräsentation ist eine geeignete Grundlage für die Schaffung multipler, entfernter und gegebenenfalls alternativer Präsentationen.

Im GroupPlan–-Projekt konnte als technische Basis der Semantic Presentation Split-Mechanismus [ObjectShare 98] von Distributed Smalltalk genutzt werden. Dieser Mechanismus stellt eine Erweiterung des MVC–Paradigmas auf Verteilte Umgebungen dar. Das Presentation – Object übernimmt die anwenderseitigen Aufgaben des Views sowie des Controllers, Semantic–Objekte entsprechen weitgehend dem Modell bei MVC. Intern nutzt die Smalltalk-Umgebung für die Umsetzung die CORBA–basierten Link– und EventNotification–Dienste von Distributed Smalltalk.

Für das Modellverwaltungssystem des SFB 524 ist ein ähnlicher Mechanismus realisierbar. Dazu müssen, wie im Modellverwaltungs–Kapitel beschrieben, Präsentationsklassen für die Domänenmodellklassen erzeugt werden, die Java–Bytecode–Module können im Repository des Modellverwaltungskerns verwaltet werden. Auf Seiten der Domänenclients müssen Frameworks für alle Präsentationstypen vorliegen, die lokal genutzt werden sollen, um eine Ausführungsumgebung für die Präsentationsklassen bereitzustellen. Die Verwaltung der Modellinformationen erfolgt ohnehin durch das Modellverwaltungssystem, der Domänenserver übernimmt als weitere Aufgabe die Benachrichtigung der interessierten Domänenclients über Änderungen der Modell– und Präsentationsinformationen. Dazu ist der oben beschriebene ereignisbasierte asynchrone Benachrichtigungsmechanismus gut geeignet.

Voraussetzung für die Darstellung synchroner WYSIWIS–Präsentationen in den Fachapplikationen ist die beschriebene enge Anbindung der Applikationen an das Modellverwaltungssystem. Legacy–Applikationen, die über Wrappertechniken mit der Infrastruktur verbunden sind und über filebasierte Schnittstellen mit der Modellierumgebung kommunizieren, sind naturgemäß nicht für die beschriebenen kooperativen synchronen Arbeitstechniken geeignet.

6.3.4 Synchrone CSCW– Techniken für Telepresence in GroupPlan

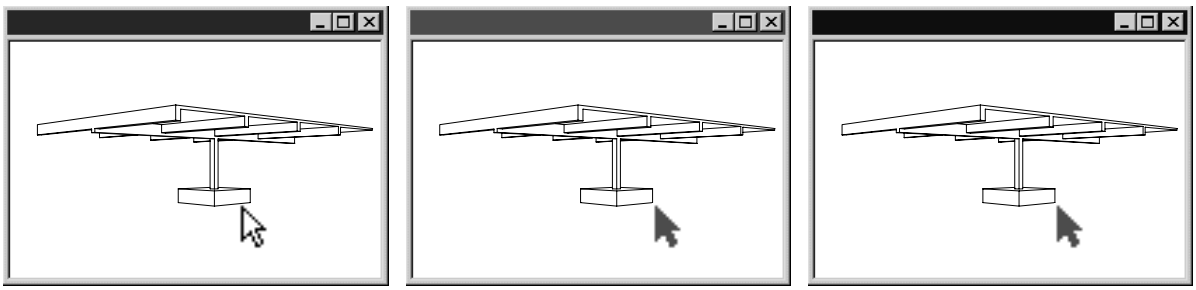

Abbildung 1: Multiple Views mit Telepointern

Ein Weg, der Reduzierung der Group Awareness bei abgeschwächten WYSIWIS entgegenzuwirken, ist der Einsatz von Telepointern bzw. die Darstellung des Cursors der anderen Teammitglieder. Ein Telepointer ist ein spezieller graphischer Cursor, der von mehreren Teammitgliedern bewegt werden kann und in Phasen synchroner Kooperation zum Verweisen

auf Inhalte von öffentlichen Fenstern dient. Im GroupPlan–System ist die Anwendung von Telepointern auf Fenster mit identischen Inhalten, also solche bei denen das Kongruenz–Constrait nicht verletzt wird, beschränkt.

Eine Anwendung im Bauwerksentwurf außerhalb von Darstellungen mit demselben Präsentationstyp und den gleichen visualisierten Objekten ist mit Problemen verbunden. Der Grund dafür ist, dass bei Nutzung verschiedener Views eine sinnvolle Telepointerdarstellung zu den Zeitpunkten kaum realisierbar ist, in denen infolge von Pointerbewegungen durch das Fenster keine Zuordnung zu Modellobjekten möglich ist, da sich dann die Positionierung der Pointers nicht an der physikalischen, sondern an der logischen Position im Fenster orientieren müsste. Dies würde der üblichen Arbeitsweise von Cursors und Telepointern widersprechen, die sich üblicherweise am Rasterkoordinatensystem des Fensters orientieren.

Diese Problematik ist auch die Ursache dafür, dass im GroupPlan–System bei der Nutzung von verschiedenen Präsentationstypen auf die Darstellung des Cursors anderer Teammitglieder vorerst verzichtet wurde. Statt dessen ist es möglich, die selektierten Objekte anderer Teammitglieder darzustellen. Das kann sowohl durch spezifische Darstellung, also die Nutzung von Darstellungsattributen wie partielle Wireframe–Darstellung und Farben, als auch durch Kennzeichnung mit dem Benutzernamen oder Icons bzw. sogenannte 'Handles', also kleine Quader an den Eckpunkten des umhüllenden Quaders geschehen. In GroupPlan wurde aufgrund der einfachen Realisierbarkeit für die Nutzung spezifischer Farbdarstellungen entschieden. Eine Voraussetzung für die Nutzung von Teleselections ist, dass die Selektionen des Teammitglieds zum Export freigegeben und dass die Darstellung der Selektionen einzelner Teammitglieder explizit angefordert wurde.

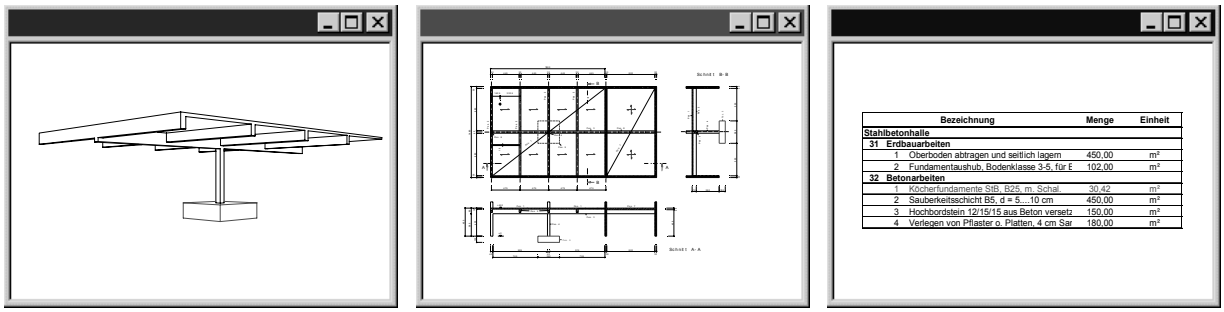

Abbildung 2: Multiple Views mit Teleselections

Wie oben bereits erwähnt, erfordert synchrone Kooperation das Vorhandensein externer Kommunikationskanäle. In GroupPlan sind eine Audioverbindung und ein textbasierter Kommunikationskanal realisiert worden. Zweckmäßig wäre eine Erweiterung um Video–Kanäle mit der Darstellung der Meeting–Teilnehmer in separaten Fenstern, was zu positiven Auswirkungen auf die Group Awareness führen dürfte.

Mit den beschriebenen Mechanismen ist es möglich, verschiedene Typen von graphischen, interaktiven und in synchroner Kooperation nutzbaren Bauwerksmodelleditoren bzw.

Entwurfstools zu implementieren. Die Realisierung von Systemen mit einer vollen CAD–Funktionalität ist jedoch mit einem sehr hohem Implementierungsaufwand verbunden und in einem universitärem Forschungsprojekt mit begrenzten Ressourcen nicht zu leisten. Daher musste im GroupPlan–Projekt die grundsätzliche Umsetzbarkeit der beschriebenen Konzepte an einfachen Prototypen nachgewiesen werden.

Auch bezüglich der Techniken zur Unterstützung synchroner Kooperation besteht punktuell weiterer Forschungsbedarf. So müssen spezifische Undo– und Redo– Mechanismen konzipiert und implementiert werden, da in kooperativen Umgebungen überlappende Arbeitsbereiche der Teammitglieder auftreten können. Die Anwendung traditioneller Verfahren würde hier zu Nebenwirkungen auf die Resultate anderer Teilnehmer führen können. Ebenso ist es beispielsweise notwendig, CSCW–fähige Mechanismen zum Zusammenführen unterschiedlicher Versionen von Modellinformationen zu realisieren, da im Ergebnis asynchroner Kooperation oder optimistischer Concurrency–Control–Verfahren Differenzen zwischen lokalen Informationsbeständen auftreten können.

7 Zusammenfassung und Ausblick

Durch Wandlungen in den Organisationsstrukturen der Bauwerksplanung ergeben sich zunehmend Änderungen im Arbeitsumfeld der an Planungsprojekten beteiligten Fachleute. Da die Planung innerhalb von Virtual Enterprises und Kooperation mit geographisch weit entfernten Partnern immer häufiger auftritt, erlangen effektive Technologien des Informationsaustauschs immer mehr an Bedeutung. Es ist zu erwarten, dass ähnlich zur weitgehenden Ablösung der Planungstätigkeiten am Reißbrett durch CAE–Techniken effiziente Technologien der Online-Informationsübertragung an die Stelle des Austauschs von Papierdokumenten treten werden.

Die vorliegende Arbeit leistet zu dieser Problematik einen Beitrag durch die Untersuchung der technologischen Möglichkeiten, die Konzeption und prototypische Evaluierung von Techniken zur Unterstützung asynchroner und synchroner Kooperation in der Bauwerksplanung auf Basis dynamischer deskriptiver objektorientierter Bauwerksmodelle, die als gemeinsame Informationsbestände genutzt werden.

Die Eignung laufzeitdynamischer deskriptiver objektorientierter Bauwerksmodelle für verschiedene Aufgaben in der Bauwerksplanung hat sich in einer Anzahl vorangegangener Projekte erwiesen. Modellverwaltungssysteme stellen die technische Basis zum Management derartiger digitaler Bauwerksmodelle und den dazugehörigen Projektinformationen, also von Taxonomieklassen und deren Instanzen dar. In der vorliegenden Arbeit wurden notwendige Anforderungen an Modellverwaltungssysteme sowie an in der Umsetzung anzuwendende Basisssoftware wie DBMS und Middleware erörtert. Ebenso wurden Implementierungsansätze für Modellverwaltungssysteme systematisiert, grundsätzlich kann die Umsetzung von Modellverwaltungssystemen auf geeigneten Tools, einer kompletten Implementation der benötigten Funktionalitäten oder auf der Ausnutzung von relevanten Eigenschaften objektorientierter Programmiersprachen oder Datenbankverwaltungssysteme beruhen. Eine Präsentation von Projektinformationen, die zu laufzeitdynamisch erzeugten Modellklassen gehören, ist durch auf dem Model View Controller beruhenden Präsentationsarchitekturen möglich. Dazu werden Definitionen von Darstellungsvorschriften unter Zuhilfenahme geeigneter Werkzeuge und möglichst unter Rückgriff auf ererbte Vorschriften erzeugt, die unter Auswertung der Instanzinformationen zur Präsentation der Projektinformationen ausgeführt werden.

In GroupPlan wie im SFB 524 wird eine hybride Modellarchitektur verwendet. Diese besteht aus separaten Domänenmodellen, die auf einem gemeinsamen Metamodell beruhen sowie einer Abbildung der Kohärenz der Gesamtinformationsbestände durch Modellverknüpfungen. Ferner bestehen zentrale Komponenten zur Verwaltung der technischen Infrastruktur des Planungsvorhabens, ebenso sind zentrale Komponenten zur Navigation in den Informationsbeständen erforderlich.

Die physikalische Gesamtsystemarchitektur muss die Besonderheiten der Planung in Virtual Enterprises beachten. Klassische Architekturen für Verteilte Systeme wie die in anderen Branchen erfolgreichen Three-Tier-Models erfüllen diese Anforderungen nur unbefriedigend. Daher wurde in der vorliegenden Arbeit eine Gesamtsystemarchitektur vorgestellt, welche die notwendige Flexibilität aufweist. Die Grundkomponenten der Systemarchitektur sind ein zentraler Projektserver, mehrere Domänenserver und eine Anzahl zu diesen gehöriger Domänenclients.

Zur Kommunikation der Domänenserver untereinander und mit den Domänenclients werden statische CORBA–Operationen genutzt. Dazu besitzen die Bauwerksmodellierkerne eine in CORBA-IDL definierte Schnittstelle, welche die Funktionalität der Metaebene der Bauwerksmodelle verfügbar macht. Zur Autorisierung der Zugriffe auf die Informationen im Modellverwaltungssystem kann das matrixbasierte Verfahren zur Access Control nach Rodden und Smith genutzt werden. Basis der Zugriffskontrolle ist eine Authentisierung der Anwender verbunden mit einer Zuordnung zu Rollen.

Als Mechanismus zur Übermittlung von Informationen zwischen sämtlichen Komponenten der Gesamtsystemarchitektur, also auch an die Domänenclients, wurde ein asynchroner, ereignisbasierter Kommunikationsmechanismus geschaffen. Dieser kann für die Übermittlung von Informationen über Modellmodifikationen oder beispielsweise als Basis für CSCW–Techniken genutzt werden. Ebenso kann der Mechanismus für die Verbreitung von Awareness–Informationen angewandt werden, die essentiell für eine effektive Gruppenarbeit sind.

Die Verwaltung der gemeinsamen Informationen in Modellverwaltungssystemen ist im CSCW–Sinne die Schaffung eine gemeinsamen Informationsraums. Die implizite Kommunikation durch das auf Information Sharing beruhende Medium Modellverwaltungssystem stellt die wesentlichste Unterstützung der asynchronen Gruppenarbeit dar. Ebenso stellen die Mapping–Mechanismen zwischen den Modellen einen wesentliche Groupware–Technik für asynchrone Kooperationsphasen dar, da durch diese die Anwender über Modifikationen an für sie relevanten Informationsbeständen informiert, Änderungen propagiert bzw. Informationen generiert werden können. Neben den genannten bauwerksentwurfsspezifischen CSCW–Techniken ist für eine effektive verteilte Teamarbeit die Bereitstellung asynchroner Kommunikationsverfahren notwendig.

Synchrone Kooperationsphasen treten vor allem zur Koordination künftiger Aktivitäten und zur Klärung von Entwurfskonflikten auf. Diese Aktivitäten können bei verteilter Arbeit durch eine rollenspezifische Darstellung des gemeinsamen Entwurfskontexts ermöglicht werden. Dazu sind WYSIWIS–Präsentationen geeignet, die auch auf Basis laufzeitdynamischer deskriptiver Bauwerksmodelle erzeugt werden können. Telepresence–Techniken wie Telepointer und Teleselections sind geeignet, den Verlust an Group Awareness bei relaxierten Formen von WYSIWIS auszugleichen, ferner sind unbedingt externe Audio– bzw. Videokommunikations-kanäle zur Verfügung zu stellen.

Bei synchroner Aktivität in gemeinsamen Informationsbeständen müssen Techniken zur Kontrolle von Nebenläufigkeiten umgesetzt werden. Die Majorität der klassischen Concurrency Control Techniken sind für die Steuerung paralleler Systemkomponenten entworfen wurden und können kaum auf Teamarbeit oder Human Computer Interaction übertragen werden, da die Konsequenzen möglicher Abbrüche von Aktivitäten inakzeptabel sind und diese Techniken den Entwurfsfortschritt durch Blockierung von Bearbeitern stark beeinträchtigen können. Im Kontext dynamischer Modelliersysteme stellen Locktechniken variabler Granularität und optimistische Transaktionskonzepte adäquate Technologien dar.

In der vorliegenden Arbeit wurde für die relevanten Bereiche der Fachgebiete Bauwerksmodellierung, CAE–Systemintegration, CSCW und Verteilte Systeme der aktuelle Stand der Forschung dargelegt. Für die Realisierung von Bauwerksmodellverwaltungssystemen, die Modellverteilung und den entfernten Zugriff auf Modellinformationen sowie für die Unterstützung kooperativer Arbeit in der Bauwerksplanung wurden neue Konzepte entwickelt und vorgestellt bzw. teilweise existierende Techniken untersucht, ausgewählt und für den Einsatz im beschriebenen Kontext angepasst. Im Rahmen der Bearbeitung wurden Prototypen für Modellverwaltungssysteme auf Java- und Smalltalk-Basis, für den zentralem Projektserver, für die Anbindung von Applikationen, für die vorgestellte Präsentationsarchitektur und für die beschriebenen WYSIWIS–Präsentationen zur Unterstützung synchroner Kooperation erstellt.

Die getroffenen Aussagen sind prinzipiell auch auf Systeme mit statischen Modellen übertragbar, da statische Modelle durchaus als Spezialfall dynamischer Modelle angesehen werden können. Je nach konkreter Aufgabe kann es jedoch für statische Modelle effizientere Lösungen geben, da hierfür ein Teil der verfügbaren Funktionalitäten nicht benötigt wird. Die Ergebnisse bezüglich der Kooperationsunterstützung und des Einsatzes von CSCW–Techniken wurden zwar für das genutzte Metamodell sowie die beschriebene Modellarchitektur und Gesamtsystemarchitektur konzipiert, dürften aber auch für andere auf Gemeinsamen Informationsräumen und Workgroup Computing beruhenden Bauwerksentwurfsumgebungen übertragbar sein.

Trotz der erzielten Resultate verbleiben noch eine Anzahl ungelöster Fragen, ebenso haben sich einige neue Fragen im Rahmen der Bearbeitung ergeben.

Die Versionierung von Projektinformationen bei statischen Domänenmodellen ist ein weitgehend geklärtes Problem. Bei laufzeitdynamischen Domänenmodellen müssen sowohl Klassen– und als Instanz-Informationen versioniert werden. Es existiert noch Klärungsbedarf zu Techniken variantenorientierter und zeitlicher Versionierung sowie zur Zusammenführung von Instanzmengen, die zu verschiedenen Versionen einer Taxonomie gehören.

Zur Überwachung der Modellkonsistenz von der physikalischen bis hin zur gesamtprojektbezogenen Ebene sollten weitere Forschungsaktivitäten unternommen werden. Auf Ebene einzelner Domänenmodelle sollte die Ausdruckkraft in UML–Object Constraint Language formulierter Regeln untersucht werden. Ferner besteht weiterer Handlungsbedarf im Bereich der Verwaltung der Modellverknüpfungen, die letztendlich auch als Mechanismus der Wahrung der

Konsistenz der domänenübergreifenden Planungsbemühungen fungieren. Hier sind beispielsweise noch effektivere Methoden zur Definition von Verknüpfungen, zur Übertragung von Definitionen auf Instanzen und zur Übermittlung größerer Ereignismengen wünschenswert. Nach Klärung der Fragen zur Konsistenzüberwachung sollte pro Ebene der Grad der (teil–) automatischen Wiederherstellbarkeit geprüft werden und Methoden zur Unterstützung der Anwender bei der Überführung der Informationen in einen konsistenten Zustand bzw. Mechanismen zur Ausführung dieser Aufgaben durch die Entwurfsumgebung entwickelt werden.

Bezüglich der Modellierung der Bauwerksinformationen existieren Bemühungen zur Klärung von Struktur und Inhalt zentraler Modellbestandteile. So sollte ein Navigationsmodell geschaffen werden, dass für die automatisierte Suche nach Informationen wie auch für die Navigation der Anwender durch die Datenbestände geeignet ist. Diese Modellanteile sollten einerseits nicht die oben beschriebenen Nachteile zentraler Komponenten aufweisen und andererseits die nötige Flexibilität für Planungsaktivitäten in Virtual Enterprises aufweisen.

Für die effektive Unterstützung der kooperativen Aspekte in der Bauwerksplanung sind eine Anzahl offener Fragen verblieben. So sind Techniken zur Verbesserung der Group Awareness und zur synchronen Kooperation nach wie vor ein aktuelles Forschungsfeld im CSCW–Bereich. Speziell für Entwurfstätigkeiten sind weiterhin gruppenarbeitsfähige Techniken zur Rückgängigmachung von Entwurfsentscheidungen (Undo) und zur Wiederherstellung dieser (Redo) wünschenswert. Da von diesen Aktionen die Ergebnisse anderer Teammitglieder betroffen sein können, sind diese Fragen sehr wahrscheinlich nicht trivial beispielsweise über Stacks vergangener Operationen zu lösen.

Ebenso besteht auch weiterer Handlungsbedarf an verschiedenen Punkten, der nicht ausschließlich durch Aktivitäten innerhalb der Informatik, der Bauinformatik, der Architektur oder dem Bauingenieurwesen zu lösen sind. Zu diesen gehören beispielsweise rechtliche Probleme wie die Klärung, inwiefern binäre Darstellungen mit derselben Semantik als äquivalent anzusehen sind. Ebenso sollten durch die Berufsverbände und den Gesetzgeber die organisatorischen und rechtlichen Randbedingen wie beispielsweise Gebührenordnungen an die Erfordernisse neuer Formen der Bauwerksmodellierung und des Informationsaustauschs angepasst werden, da diese derzeit noch von konventionellen Formen der Bauwerkplanung ausgehen.

Abkürzungsverzeichnis

ADSL	Asymmetric Digital Subscriber Line
A/E/C	Architecture/ Engineering/ Construction
AI	Artifical Intelligence
AKO	Arbeitskreis Objekte
AP	Application Protocol
API	Application Program Interface
ARM	Application Reference Model
BSD	Berkeley System Distribution
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CHI	Computer Human Interaction
CIM	Computer Integrated Manufacturing
CLOS	Common Lisp Object System
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CSCW	Computer Supported Cooperative Work
DCOM	Distributed Component Object Model
DDBS	Distributed Database Systems
DFG	Deutsche Forschungsgemeinschaft
DNS	Domain Name Service
DPM	Dynamic Product Model
DSE	Dynamic Schema Evolution
DSTV	Deutscher Stahlbauverband
DXF	Data Interchange Format
FDBMS	Federated Database Management System
FDBS	Federated Database Systems
FM	Facilty Management

FTP	File Transfer Protocol
GUI	Graphical User Interface
HCI	Human Computer Interaction
HLS	Heizung–Lüftung–Sanitär
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IAI	International Alliance for Interoperability
IBDE	Integrated Building Design Environment
IDL	Interface Definition Language
IFC	Industry Foundation Classes
IOR	Interoperable Object Reference
IP	Internet Protocol
IRC	Internet Relay Chat
ISO	International Organization for Standardization
IT	Information Technology
KI	Künstliche Intelligenz
LAN	Local Area Network
LOOPS	Lisp Object-Oriented Programming System
MAN	Metropolitan Area Network
MVC	Model–View–Controler
MVS	Modellverwaltungssystem
NetBEUI	NetBIOS Extended User Interface
NetBIOS	Network Basic Input Output System
O.P.E.N.	Objectoriented Product model Engineering Network
OMG	Object Management Group
OOPL	Object oriented programming language
ORB	Object Request Broker
OSI	Open System Interconnetion
PDT	Product Data Technology
POTS	Plain Old Telephone Service

QoS	Qualtiy of Service
RPC	Remote Procedure Call
SDAI	Standard Data Access Interface
SDSL	Single line Digital Subscriber Line
SFB	Sonderforschungsbreich
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
STEP	Standard for the Exchange of Product Model Data
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
VE	Virtual Enterprise
VML	View Mapping Language
VPE	VML Programming Environment
VPN	Virtual Private Network
WAN	Wide Area Network
WWW	World Wide Web
WYSIWIS	'What You See Is What I See'
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations

Literaturverzeichnis

[Abbot 94] Abbott, K.; Sarin, S.: Experiences with Workflow Management: Issues for the Next Generation, in: Furuta, R.; Neuwirth,C. (Hrsg.): Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '94, Chapel Hill, United States, 1994

[Ahuja 90] Ahuja, S. R.; Ensor, J. R.; Lucco, S. E.: ‚A Comparison of Application Sharing Mechanisms In Real–Time Desktop Conferencing Systems' in: Proceedings of the Conference on Office Information Systems, April 1990, Boston, 1990

[AKO 97] Ranglack, D.; Kolbe, P.; Steinmann, F.: Eine Schnittstelle für dynamische Objektstrukturen für Entwurfsanwendungen, IKM 1997, Weimar, 1997

[Alhir 98] Alhir, S. S.: UML in A Nutshell, O'Reilly & Associates Inc., Sebastopol CA, 1998

[Amor 94] Amor, R. W.; Hosking, J.: Mappings: The glue in an integrated system, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Amor 97] Amor, R. W.: A Generalised Framework for the Design and Construction of Integrated Design Systems, Dissertation, University of Auckland, New Zealand, 1997

[Anderl 92] Anderl, R.; Malle, B.; Schmidt, B.: Concurrent Engineering based on a Product Data Model, Proceedings of CALS EUROPE '92, Paris, Hermes, 1992

[Anderl 97] Anderl, R.; Momberg, M.: Authentisierung von Produktdaten, in: Produktdaten Journal Nr. 2, 4. Jahrgang, Dezember 1997, ProSTEP e.V., 1997

[ATLAS 93] ATLAS ESPRIT Projekt 7280 WWW Dokumentation: Architectures methodologies and Tools for computer integrated Large Scale engineering, URL=http://cic.cstb.fr/ILC/ECPROJEC/ATLAS/atlas.htm sowie URL=http://www.newcastle.research.ec.org/esp-syn/text/7280.html

[Augenbroe 94] Augenbroe, G.: An overview of the COMBINE project, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Augenbroe 98] Augenbroe, G.; Rombouts, W.; Verhoerls, M.: Product Data Technology in Integrated A/E/C Systems: Past, Present & Future, in: Amor, R. (ed.): Proc. of the 2nd European Conference On Product and Process Modelling in the Building Industry – ECPPM '98, Watford UK 1998, BRE Ltd, 1998

[Augenbroe 98a] Augenbroe, G.; Lockley. S.: CaribCAD: a Technology to Outsource CAD Production Work , in: Amor, R. (ed.): Proc. of the 2nd European Conference On Product and Process Modelling in the Building Industry – ECPPM '98, Watford UK 1998, BRE Ltd, 1998

[Baecker 93] Baecker, R. M. (Hrsg.): Groupware and computer supported cooperative work' Morgan Kaufmann, San Mateo CA, 1993

[Bannon 91] Bannon, L. J.; Schmidt, K.: CSCW: Four Characters in Search a Context, in: Bowers, J. M.; Benford, S. D. (Editors): Studies in Computer Supported Cooperative Work, Elsevier Science Publishers B.V. (North Holland), 1991

[Beck 97] Beck, K.: Smalltalk best practice patterns, Prentice Hall, Upper Saddle River NJ, 1997

[Beetz 97] Beetz, K.; Junge, R.; Steinmann, R.: The O.P.E.N. Platform, in: Amor, R. (ed.) : Proc. of the 2nd European Conference On Product and Process Modelling in the Building Industry – ECPPM '98, Watford UK 1998, BRE Ltd, 1998

[Bengel 00] Bengel, G.: Verteilte Systeme – Client Server Computing, Friedrich Vieweg & Sohn Verlagsgesellschaft, Braunschweig, Wiesbaden, 2000

[Ben–Natan 95] Ben–Natan, R.: CORBA – A guide to the common object request broker architecture, McGraw–Hill, New York, 1995

[Bentley 95] Bentley, R.; Dourish, P.: Medium versus mechanism: Supporting collaboration through customisation, in: Marmolin, H.; Sundblad, Y.; Schmidt, K. (ed.): Proc. of the 4th European Conference on Computer Supported Cooperative Work 1995 (ECSCW95), Kluwer Academic Publishers, Dordrecht, 1995

[Betz 94] Betz, M.: Interoperable Objects, in: Dr. Dobb's Journal, October 1994

[Betz 95] Betz, M.: Networking Objects with CORBA, in: Dr. Dobb's Journal, November 1995

[Björk 89] Björk, B.–C.: Basic Structure of a Proposed Building Product Model, Computer Aided Design, Vol. 21 (1989), No. 2, pp. 71–78, March 1989

[Björk 92] Björk, B.–C.: A conceptual model of spaces, space boundaries and enclosing structures. Automation in construction, Vol.1 (1992), No. 3, December 1992

[Beucke 97] Beucke, K.: Schlussbericht zum Forschungsvorhaben DBV 188 "CAE–Bauwerksmodelle", Fraunhofer IRB Verlag, 1997

[Bly 90] Bly, S. A.; Minneman, S. L.: Commune: A Shared Drawing Surface, ACM '90

[Booch 94] Booch, G.: Object–oriented Analysis and Design With Applications, 2nd Edition, Benjamin / Cummings Publishing Company, Redwood City CA, 1994

[Booch 99] Booch, G.; Rumbaugh, J.; Jacobson, I.: The Unified Modelin Language User Guide, Addison Wesley, Reading MA, 1999

[Borghoff 95] Borghoff, U.; Schlichter, J.: Rechnergestützte Gruppenarbeit. Eine Einführung in Verteilte Anwendungen, Springer Verlag Berlin 1995

[Borghoff 98] Borghoff, U.; Schlichter, J.: Rechnergestützte Gruppenarbeit. Eine Einführung in Verteilte Anwendungen, 2. vollst. überarbeitete und erweiterte Auflage, Springer Verlag Berlin 1998

[Bretschneider 98] Bretschneider, D.: Modellierung rechnergestützter, kooperativer Arbeit in der Tragwerksplanung, Dissertation Ruhr-Universität Bochum, VDI-Fortschrittsberichte Reihe 4 Nr. 151, VDI-Verlag GmbH, Düsseldorf, 1998

[Brinck 92] Brinck, T., Gomez, L. M.: A Collaborative Medium for the Support of Conversational Props, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '92, Toronto, Canada, 1992

[Burger 97] Burger, C.: GroupWare – Kooperationsunterstützung für verteilte Anwendungen, dpunkt–Verlag, Heidelberg, 1997

[Buxton 92] Telepresence: Integrating Shared Task and Person Spaces, in: Proceedings of Graphic Interface 92, pp.123–129, Morgan Kaufman Publishers 1992, Abdruck in [Baecker 93]

[Cardelli 85] Cardelli, L.; Wegner, P.: On Understanding Types, Data Abstraction, and Polymorphism, ACM Computing Surveys, Vol.17, No.4, December 1985, pp.471–522, Association for Computing Machinery, 1985

[CIS 00] Crowley, A. J.; Watson A. S. (eds.): CIMsteel Integration Standards–Release 2, University of Leeds & The Steel Construction Institute, Ascot UK, 2000

[Coulouris 94] Coulouris, G.; Dollimore, J.; Kondberg, T.: Distributed Systems – Concepts and Design, 2nd edition, Addison Wesley Publishers Ltd., Wokingham, UK, Reading MA, 1994

[Crowley 90] Crowley, T.; Milazzo, P.; Baker, E.; Forsdick, H.; Tomlinson, R.: MMConf: A Infrastructure for Building Shared Multimedia Applications, Proc. 3rd Int. Conf. On Computer–Supported Cooperative Work, Los Angeles, CA. New York: SIGCHI / SIGOIS ACM, 1990

[Dankwort 96] Dankwort, W.; Janocha, A.: Von Monolithen zu Komponenten: CAx – Architekturen im Wandel, in: Ruland, D. (ed.): Tagungsband GI–Fachtagung CAD '96, DFKI 1996

[Dewan 98] Dewan, P., Shen, H.: Controlling Access in Multiuser Interfaces, ACM Transactions on Computer Human Interaction 1998, Vol. 5:1, pp34–62, ACM Press '98

[Dewan 98a] Dewan, P., Shen, H.: Flexible Meta Access–Control for Collaborative Applications, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '98, Seattle Washington, USA, 1998

[Diaper 93] Diaper, D.; Sanger, C. (eds.): CSCW in Practice: An Introduction and Case Studies, Springer Verlag, London 1993

[Dinger 99] Dinger, F.; Forgber, U.: Graphische Kommunikationsunterstützung internetbasierter Projekträume, in: 11. Forum Bauinformatik, Darmstadt '99, VDI Verlag Düsseldorf, 1999

[Dourish 92] Dourish, P.; Belotti, V.: Awareness and Coordination in Shared Workspaces, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '92, Toronto, Canada, 1992

[Dourish 92a] Dourish,P.; Bly, S.: Portholes: Supporting Awareness in a Distributed Work Group, Proceedings of the ACM Conference on Human Factors and Computing Systems CHI '92, in: [Baecker 93]

[Dourish 95] Dourish, P.: The Parting of the Ways: Divergence, Data Management and Collaborative Work, in: Marmolin, H.; Sundblad, Y.; Schmidt, K. (ed.): Proc. of the Fourth European Conference on Computer Supported Cooperative Work 1995 (ECSCW '95), Kluwer Academic Publishers, Dordrecht, 1995

[Drach 94] Drach, A.: Flexible Werkzeuge für die integrierte Gebäudeplanung, VDI–Verlag GmbH, Düsseldorf 1994

[DStV 00] Deutscher Stahlbau–Verband – DStV Arbeitsauschuss EDV: Standard-beschreibung Produktschnittstelle Stahlbau, Version April 2000, Düsseldorf 2000

[Dubois 94] Dubois, A.; Flynn, J.; Verhoef, M.; Augenbroe, G.: Conceptual modelling approaches in the COMBINE project, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM'94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Duffy 95] Duffy, D. J.: Interoperability Issues in CAD Software Development, Objects in Europe 1995: 4

[Eastman 97] Eastman, C. et al: Integration of Design Applications with Building Models, in: Junge, R. (ed.): CAAD futures 1997 – Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures, Kluwer Academic Publishers, Dordrecht, 1997

[Eastman 99] Eastman, C. M.: Building Product Models: Computer Environments Supporting Design and Construction, CRC Press, Boca Raton, FL, 1999

[Edwards 97] Edwards, W. K.; Mynatt, E. D.: Timewarp: Techniques for Autonomous Collaboration, Proceedings of the ACM Conference on Human Factors and Computing Systems CHI '97, Atlanta, GA, USA, 1997

[Egger 93] Egger, E.; Wagner, I.: Negotiating Temporal Orders- The Case of Collaborative Time Management in a Surgery Clinic, Computer Supported Cooperative Work (CSCW), Vol.1, Nr. 4, pp.255-277, Kluwer academic publishers, 1993

[Egido 88] Egido, C.: Video Conferencing as a technology to support group work: A review of ist failures, Proceedings of the Conference on Computer Supported Cooperative Work (CSCW '88), Portland 1988, ACM Press.

[Ellis 89] Ellis, C. A.; Gibbs, S.: Concurrency Control in Groupware Systems, in: Clifford, J.; Lindsay, B.; Maier, D. (Hrsg.): Proc. ACM SIGMOD Int. Conf. On Management of Data, Portland, OR, 1989

[Ellis 91] Ellis, C. A.; Gibbs, S. J.; Rein, G. L.: 'Groupware – Some Issues and Experiences', Communications of the ACM 34:1,38–58,1991

[Ellis 94] Ellis, C.; Wainer, J.: A Conceptual Model of Groupware, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '94, Chapel Hill, United States, 1994

[Fenves 94] Fenves, S.; Flemming, U.; Hendrickson, C.; Maher, M.; Quadrel, R.; Terk, M.; Woodbury, R.: Concurrent Computer Integrated Building Design, Prentice Hall, Englewood Cliffs NJ, 1994

[Fischbach 94] Fischbach, R.: Modellbildung: Grundlagen und industrielle Bedeutung, Teil 1 – Grundlagen, in: Bauinformatik, Heft 2/94, Verlag R. Müller, Köln, 1994

[Fischbach 94b] Fischbach, R.: Modellbildung: Grundlagen und industrielle Bedeutung, Teil 2 – Industrielle Bedeutung, in: Bauinformatik, Heft 3/94, Verlag R. Müller, Köln, 1994

[Fischbach 94b] Fischbach, R.: Produktmodellierung im Bauwesen, in: Bauinformatik, Heft 6/94, Verlag R. Müller, Köln, 1994

[Fish 88] Fish, R. S.; Kraut, R. E.; Leland, M. D. P.: Quilt: A Collaborative Tool for Cooperative Writing, in: Allen, R. B. (Hrsg.): Proc. ACM SIGOIS / IEEE TC–OA Conf. on Office Information Systems, Palo Alto, CA, 1988

[Fisher 97] Fisher, N.; Barlow, R.; Garnett, N.; Finch, E.; Newcombe, R.: Project modelling in construction: ...seeing is believing, Thomas Telford Services Ltd., London, 1997

[Fitzpatrick 98] Fitzpatrick, G.; Parsowith, S.; Segall, B.; Kaplan,S.: Tickertape: Awareness in a Single Line, Proceedings of the ACM Conference on Human Factors and Computing Systems CHI '98, Los Angeles, CA, USA, 1998

[Forgber 97] Forgber, U.; Müller, C.: A planning Process model for Computer Suported Cooperative Work in Building Construction, URL=http://www.ifib.uni-karlsruhe.de, 1997

[Forgber 98] Forger, U.: Dynamische Entscheidungsunterstützung in Computerbasierten Kooperativen Planungsumgebungen, in: 10. Forum Bauinformatik – Junge Wissenschaftler forschen, Weimar '98, VDI Verlag Düsseldorf, 1998

[Fowler 97] Fowler, M.; Scott, K.: UML Distilled–Applying the Standard Object Modeling Language, Addison Wesley, Reading MA, 1997

[Froese 96] Froese, T.: STEP and the Building Construction Core Model, International Conference on Computing in Civil Engineering, Proc. of the Third Congress, pp. 445–451, ASCE, June 1996, Anaheim, 1996

[Fruchter 98] Fruchter, R.: Internet–Based Web–Mediated Collaborative Design and Learning Environment, in: Smith, I. (ed.); Artificial Intelligence in Structural Engineering–Information Technology for Design, Collaboration, Maintenance and Monitoring, Proceedings of EG–SEA–AI Lausanne 1998, Springer, Berlin, Heidelberg, New York, 1998

[Fuchs 95] Fuchs, L.; Pankoke–Babbatz, U.; Prinz, W.: Supporting Cooperative Awareness with Local Event Mechanism: The Groupdesk System, Proc. 4th European Conference on Computer–Supported Cooperative Work 1995 (ECSCW '95), Kluwer Academic Publishers, Dordrecht, 1995

[Gamma 95] Gamma, E.: 'Design patterns: Elements of reusable object–oriented software', Addison Wesley, 1995

[Gausemeier 96] Gausemeier, J.; Hahn, A.; Schneider, W.: Kooperatives Modellieren auf Basis transienter Objekte, in: Ruland, D. (ed.): Tagungsband GI–Fachtagung CAD '96, DFKI 1996

[Gaver 91] Gaver, W.: Sound Support For Collaboration, Proc of the 2nd European Conference on Computer Supported Cooperative Work (ECSCW '91), Amsterdam 1991

[Goldberg 89] Goldberg, A.; Robson, D.: Smalltalk–80: the language, Addison Wesley, 1989

[Grabowski 94] Grabowski, H.; Schreiner, P.: Multi Agent Driven Cooperative Product Development, in Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Graham 94] Graham, I.: Object Oriented Methods, 2nd Edition, Addison Wesley Publishing Company, Wokingham, England, Reprint 1994

[Greenberg 90] Greenberg, S.: Sharing Views and Interactions with single–user applications, ACM '90

[Greenberg 92] Greenberg, S.; Roseman, M.; Webster, D.; Bohnet, R.: Issues and Experiences Designing and Implementing Two Group Drawing Tools, Proceedings of the 25th Annual Hawaii International Conference on the System Sciences, Hawaii, January 1992, Vol.4, pp. 139–150

[Greenberg 94] Greenberg, S., Marwood, D.: Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '94, Chapel Hill, United States, 1994

[Greenberg 96] Greenberg, S.; Gutwin, C.; Cockburn, A.: Using Distortion–Oriented Displays to Support Workspace Awareness, University of Calgary, Canada, 1996

[Greenberg 96a] Greenberg, S.: Peepholes low cost awareness of one´s community, Proceedings of the ACM Conference on Human Factors and Computing Systems CHI '96, Vancouver, Canada, 1996

[Greenberg 96b] Greenberg, S.: A Fisheye Text Editor for Relaxed–WYSIWIS Groupware, Proceedings of the ACM Conference on Human Factors and Computing Systems CHI '96, Vancouver, Canada, 1996

[Greif 88] Greif, I.; Sarin, S.: 'Data Sharing in Group Work', 2nd Version in [Greif88a]

[Greif 88a] Greif, I. (ed.): 'Computer–Supported Cooperative Work: A Book of Readings' Morgan Kaufmann, San Mateo CA, 1988

[Griffel 98] Griffel, F.: Componentware – Konzepte und Techniken eines Softwareparadigmas, dpunkt–Verlag, Heidelberg 1998

[Grosche 00] Grosche, A.; Hauschild, Th.; Heinrich, T.: Unterstützung von Kommunikation und Kooperation bei Revitalisierungsvorhaben, in: 12. Forum Bauinformatik, Berlin, 2000

[Gutwin 96] Gutwin, C.; Rosema, M.; Greenberg, S.: A Usability Study of Awareness Widgets in a Shared Workspace Groupware System, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '96, Cambridge, MA, USA, 1996

[Gutwin 96a] Gutwin, C.; Greenberg, S; Roseman, M.: Supporting Awareness of Others in Groupware, Proceedings of the ACM Conference on Human Factors and Computing Systems CHI '96, Vancouver, Canada, 1996

[Gutwin 96b] Gutwin, C.; Greenberg, S: Workspace Awareness for Groupware, Proceedings of the ACM Conference on Human Factors and Computing Systems CHI '96, Vancouver, Canada, 1996

[Gutwin 96c] Gutwin, C.; Greenberg, S.; Roseman, M.: Workspace awareness support with radar views, Proceedings of the ACM Conference on Human Factors and Computing Systems CHI '96, Vancouver, Canada, 1996

[Gutwin 99] Gutwin, C.; Greenberg, S.: The Effects of Workspace Awareness Support on the Usability of Real–Time Distributed Groupware, ACM Transactions on Computer-Human Interaction, Vol. 6., No.3, September 1999, Pages 243-281, ACM Press, 2000

[Haas 93] Haas, W. (Hrsg.): 'CAD–Datenaustausch–Knigge: STEP–2DBS für Architekten und Bauingenieure', Springer Verlag Berlin, 1993

[Hakimpour 01] Hakimpour, F.; Geppert, A.: Resolving semantic heterogeneity in schema integration , Proceedings of the international conference on Formal Ontology in Information Systems, Ogunquit, Maine, October 2001

[Haller 95] Haller, H. et al: Product models für structural steelwork, in: Pahl, P. (ed.): Proceedings 6.ICCBE, Berlin, July 1995, Balkema, Rotterdam 1995

[Hannus 94] Hannus, M., Karstila, K.: Requirements on Standardised Building Product Models, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Hannus 96] Hannus, M.: RATAS–Homepage, VTT Finnland, URL= http://cic.vtt.fi/ projects/ ratas/index.html

[Hannus 98] Hannus, M.: The Island of Automation, URL= http://cic.vtt.fi/projects/ratas/ islands.html sowie URL= http://cic.vtt.fi/hannus/islands.html

[Hauschild 97] Hauschild, Th.; Hübler, R.: Entwicklung eines verteilbaren und kooperativ nutzbaren objektorientierten CAD–Produktmodellierkerns, in: IKM 1997, Weimar, 1997

[Hauschild 98] Hauschild, Th.; Hübler, R.: GroupPlan – ein Groupware–System für den Bauwerksentwurf, in: GI-Fachtagung CAD 1998 Tagungsband, Darmstadt, 1998

[Hauschild 98a] Hauschild, Th.; Hübler, R.: GroupPlan – Groupware für den Entwurf von Bauwerken, in: 10. Forum Bauinformatik – Junge Wissenschaftler forschen, Weimar '98, VDI Verlag Düsseldorf, 1998

[Hauschild 99] Hauschild, Th.; Hübler, R.; Schleinitz, M.: Realisierung rollenspezifischer Präsentationstypen von Bauwerksmodell–Informationen im GroupPlan–System, in: 11. Forum Bauinformatik, Darmstadt '99, VDI Verlag Düsseldorf , 1999

[Hauschild 00] Hauschild, Th.; Hübler, R.; Schleinitz, M.: Unterstützung von Gruppenarbeit im Bauwerksentwurf auf Basis eines dynamischen objektorientierten Bauwerks–Modelliersystems, in: GI–Fachtagung CAD 2000, Berlin, 2000

[Hauschild 00a] Hauschild, Th.; Hübler, R.: Teamwork support for building design based on object–oriented model management systems, in: VIII. International Conference on Computing in Civil and Building Engineering (ICCCBE), Stanford CA, 2000

[Hauschild 00b] Hauschild, Th.; Hübler, R.: Aspekte der verteilten Bauwerksmodellierung in kooperativen Entwurfsumgebungen auf Basis dynamischer Objektstrukturen, in: IKM 2000, Weimar

[Hauschild 00c] Hauschild, Th.: CORBA als Middleware für die Realisierung von CSCW–Applikationen für den Bauwerksentwurf (Workshop–Beitrag), in: 12. Forum Bauinformatik, Berlin, 2000

[Hauschild 00d] Hauschild, Th.: CORBA als Middleware für die Realisierung von CSCW–Applikationen für den Bauwerksentwurf – Beispiel für CORBA–Implementierungen (Tutorium im Rahmen der Workshops des Forums 2000), in: 12. Forum Bauinformatik, Berlin, 2000

[Hauschild 01] Hauschild, Th.; Hübler, R.: Verwaltung verteilter digitaler Bauwerksmodelle für Revitalisierungsvorhaben, in: 13. Forum Bauinformatik, München, VDI Verlag Düsseldorf 2001

[Hauschild 02] Hauschild, Th.; Hübler, R.: Distributed, Collaborative Management of Building Models for Revification Projects, in: IX. International Conference on Computing in Civil and Building Engineering (ICCCBE), Taipeh, 2002

[Hauschild 02a] Hauschild, Th., Hübler, R.: Collaborative, Dynamic Building Model Management Systems for Revivification Projects, GI– Fachtagung CAD 2002, Dresden, 2002

[Heimbigner 85] Heimbigner, A.; McLeod, D.: A Federated Architecture for Information Management, ACM Transactions on Office Information Systems, Vol. 3, No. 3, July 1985, pp 253–278, 1985

[Hill 94] Hill, R. D.; Brinck, T.; Rohall, S. L.; Patterson, J. F.; Willner, W.: The Rendezvous architecture and language for constructing multiuser applications, ToCHI Vol. 1, No. 2, June 1994

[Hoeven 94] Hoeven, A.; et al.: A Flexible Control Mechanism for CAD Frameworks, Proceedings of the conference on European design automation conference, 1994, Grenoble, France, IEEE Computer Society Press, Los Alamitos, CA, 1994

[Hofte 98] Hofte, H. G.: Working Apart Together – Foundations for Component Groupware, Telematica Instituut Fundamental Research Series (TI/FRS/001), Telematica Instituut, Enschede, The Netherlands, 1998

[Hudson 96] Hudson, S. E.; Smith. I.: 'Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems', Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '96, Cambridge, MA, USA, 1996

[Hübler 94] Hübler, R.; Steinmann, F.: Beitrag zur Computerunterstützung früher Phasen des architektonischen Entwurfs, in: IKM 1994, Weimar

[Hübler 94a] Hübler, R.; Kolbe, P.; Steinmann, F.: Wissensbasierte Computerstützung früher Phasen des architektonischen Entwurfs, Teil 1 – Konzeption und Realisierung des Systems PREPLAN, in: Wissenschaftliche Zeitschrift der Hochschule für Architektur und Bauwesen, Weimar Heft 4 (1994)

[Hübler 95] Hübler, R.; Steinmann, F.; Hauschild, Th.: Knowledge Support for Functional Design of Buildings. in: IABSE Colloquium1995 Report, Bergamo, 1995

[Hübler 95a] Hübler, R.; Steinmann, F.: Knowledge–based computer assistance for functionality and shape oriented building design, in: 6th International Conference on Computing in Civil and Building Engineering: Procedings, Berlin, 1995

[Hübler 02] Hübler, R.; Hauschild, Th.; Willenbacher, H: Distributed Cooperative Building Models for Revivification of Buildings, International Association for Bridge and Structural Engineering (IABSE) Symposium 2002, Melbourne 2002

[Hull 97] Hull, R.: Managing Semantic Heterogeneity in Databases: A Theoretical Perspective, Proceedings of the 16th ACM SIGACT–SIGMOD–SIGART symposium on Principles of database systems, Tucson AZ, 1997

[Iselborn 98] Iselborn, B.: Entwurf eines CAx – Objektbusses, in: Anderl, R.; Encarnacao, J.; Rix, J. (eds.): Tagungsband CAD '98, TeleCAD – Produktentwicklung in Netzwerken, Fachtagung der Gesellschaft für Informatik, TU Darmstadt DiK & Fraunhofer IGD, 1998

[ISO 94] International Organization for Standardization 'Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual' ISO 10303–11:1994(E) Genf, 1994

[Jablonski 95] Jablonski, S.: Workflow–Management–Systeme: Modellierung und Architektur, International Thomson Publishing, Bonn 1995

[Jaeger 96] Jaeger, T.; Prakash, A.: Requirements of Role-Based Access Control for Collaborative Systems, Proceedings of the first ACM Workshop on Role-based access control 1996 , Gaithersburg, Maryland, ACM Press, 1996

[Johansen 88] Johansen, R.: Groupware – Computer Support for Business Teams, Free Press, New York, 1988

[Junge 97] Junge, R.; Köthe, M.; et al.: The VEGA Platform, in: Junge, R. (ed.): CAAD futures 1997 – Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures, Kluwer Academic Publishers, Dordrecht, 1997

[Junge 97a] Junge, R.; Liebich, T.: Product Data Model for Interoperability in an Distributed Environment, in: Junge, R. (ed.): CAAD futures 1997 – Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures, Kluwer Academic Publishers, Dordrecht, 1997

[Junge 98] Junge, R.: Product Modeling Technology – the Foundation of Industy Foundation Classes, in: Amor, R. (ed.): Proc. of the 2nd European Conference On Product and Process Modelling in the Building Industry – ECPPM '98, Watford UK 1998, BRE Ltd, 1998

[Kang 01] Kang, M. H.; Park, J. S.; Froscher, J. N.: Access Control Mechanisms for Inter-Organizational Workflow, Proceedings of the Sixth ACM Symposium on Access control models and technologies, SACMAT '01, Chantilly, Virginia, ACM Press, 2001

[Kaplan 91] Kaplan, S. M.; Carrol, A. M.; MacGregor, K.: Suppurting Collaborative Processes with Conversation Builder, Proceedings of the ACM Conference on Organizational computing systems COCS '91, Atlanta, GA, USA, 1991

[Katranuschkov 94] Katranuschkov, P.: COMBI: Integrated product model, in: Scherer, R. J.(ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Khemlani 97] Khemlani, L.; Kalay, Y. E.: An Integrated Computing Environment for Collaborative, Multi–Disciplinary Building Design, in: Junge, R. (ed.): CAAD futures 1997 – Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures, Kluwer Academic Publishers, Dordrecht, 1997

[Kim 90] Kim, W.: Introduction to Object Oriented Database Systems, MIT Press, Cambridge MA, 1990

[Kim 91] Kim, W.; Seo, J.: Classifying schematic and data heterogeneity in multidatabase systems. IEEE Computer, 24(12):12–l& December 1991

[Kim 95] Kim, W.: Modern Database Systems – The Object Model, Interoperability and Beyond, Addison Wesley, New York, 1995

[Kolbe 97] Kolbe, P.; Pfennigschmidt, S.; Pahl, P. J.: FATIMA – Verteiltes Facility Management in Telekommunikationsnetzen, Projekt–Abschlußbericht, Technische Universität Berlin, Institut für Bauingenieurwesen, 1997

[Kowalczyk 97] Kowalczyk, W.: Ein interaktiver Modellierer für evolutionäre Produktmodelle, Dissertation TU München, 1997

[Knister 90] Knister, M. J.; Prakash, A.: DistEdit: A Distributed Toolkit for Supporting Multiple Group Editors, Proc. 3rd Int. Conf. On Computer–Supported Cooperative Work, Los Angeles, CA, New York: SIGCHI / SIGOIS ACM, 1990

[Krasner 88] Krasner ,G.E.; Pope, S.T.: A cookbook for using the model–view controller user interface paradigm in Smalltalk–80, Journal of Object–Oriented Programming Volume 1, Issue 3, Aug./Sept. 1988, SIGS Publications, Denville, NJ, 1988

[Kretzschmar 94] Kretzschmar, H.; Donath, D.; et al.: Computergestützte Bauplanung, Verlag für das Bauwesen GmbH, Berlin, 1994

[Lauwers 90] Lauwers, J. C.; Joseph, T .A.; Lantz, K. A.; Romanow, A. L.: Replicated Architectures for Shared Window Systems: A Critique, in: Proceedings of the Conference on Office Information Systems, April 1990, Boston, 1990

[Lauwers 90a] Lauwers, J.; Lantz, K.: Collaboration Awareness in Support of Collaboration Transparency: Requirements for the next Generation of Shared Window Systems, Proceedings of the ACM Conference on Human Factors and Computing Systems CHI '90, in: [Baecker 93]

[Lee 90] Lee, J. J.: Xsketch: A Multiuser Sketching Tool for X11, ACM '90

[Leeuwen 97] Leeuwen , J. P. van; Wagter, H.: Architectural Design–by–Features, in: Junge, R. (ed.): CAAD futures 1997– Proceedings of the 7th International Conference on Computer Aided Architectural Design Futures, Kluwer Academic Publishers, Dordrecht, 1997

[Liebich 93] Liebich, T.: Wissensbasierter Architekturentwurf – von den Modellen des Entwurfs zu einer intelligenten Computerunterstützung, Dissertation Hochschule für Architektur und Bauwesen Weimar 1993

[Liskov 87] Data Abstraction and Hierarchy, Proc. of OOPSLA 1987, in: ACM SIGPLAN Notices 23(5):17–34, ACM, 1988

[Lockley 94] Lockley, S.; Plokker, W.: The COMBINE Date Exchange System, in: Scherer, R. J.(ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM'94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[McKinney 98] McKinney, K.; Fischer, M.: Visualization of Construction Planning Information, Proceedings of the 3rd international conference on Intelligent user interfaces, San Francisco 1998, ACM Press, 1998

[McKinney 00] McKinney Liston, K.; Fischer, M.; Kunz, J.: Designing and evaluating visualization techniques for construction planning in: VIII. International Conference on Computing in Civil and Building Engineering (ICCCBE-VIII), Stanford CA, 2000

[Maher 00] Maher, M.; Simoff, S.; Cicognani, A.: Understanding Virtual Design Studios, Springer Verlag London, 2000

[Mark 96] Mark, G.; Haake, J.; Streitz, N. A.: Hypermedia Structures and the Division of Labor in Meeting Room Collaboration, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '96, Cambridge, MA, USA, 1996

[Martin 96] Martin, R.: The Liskov Substitution Priciple, C++–Report, 1996

[Medina–Mora 92] Medina–Mora, R.; Winograd, T.; Flores, R.; Flores, F.: The Action Workflow Approach to Workflow Management Technology, in: Turner, J.; Kraut, R. E. (Hrsg.): Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '92, Toronto, Canada, 1992

[Meißner 95] Meißner, U.; Peters, F.; Rüppel, U.: Graphically interactive, object–oriented Product Modeling of Structures, in: Pahl, P. ed.): Proceedings 6th ICCBE, Berlin, July 1995, Balkema, Rotterdam 1995

[Meißner 00] Meißner, U.; et al.: Objektorientierte Tragwerksmodelle für die Systemintegration von Planungs- und Konstruktionsvorgängen im Bauwesen. In: Hartmann D. (ed.): Deutsche Forschungsgemeinschaft - Objektorientierte Modellierung in Planung und Konstruktion, Wiley GmbH, 2000

[Meyer 88] Meyer, B.: Object–oriniented Software Construction, Prentice Hall, Hemel Hempstead, 1988

[Müller 98] Müller, C.: InteGrA- Eine integrierende Groupware-Anwendung für ein Architekturbüro, in: 10. Forum Bauinformatik – Junge Wissenschaftler forschen, Weimar '98, VDI Verlag Düsseldorf, 1998

[Munson 94] Munson, J. P.; Dewan, P.: Flexible Object Merging Framework, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '94, Chapel Hill, United States, 1994

[Neuwirth 90} Neuwirth, C. M.; Kaufer, D. S.; Chandhok, R.; Morris, J. H. (Hrsg.): Issues in the Design of Computer Support for Co–Authoring and Commenting, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '90, Los Angeles, CA, USA, 1990

[Noy 02] Noy, N. F.; Klein, M.: Ontology evolution: Not the same as schema evolution, Technical Report SMI–2002–0926, Stanford Medical Informatics, Stanford University, CA, 2002

[ObjectShare 98] ObjectShare, Inc.: Visual Works Distributed Smalltalk Programmer's Reference, Part-No. DS615-0298, ObjectShare Inc, Irvine CA, 1998

[Olbrich 98] Olbrich, M.: Relationenorientiertes Modellieren mit Objekten in der Bauinformatik, Dissertation Universität Hannover – Institut für Bauinformatik, 1998

[OMG 92] The Object Management Group 'The Common Object Request Broker: Architecture and Specification ' OMG Document Number 91.12.1, Revision 1.1 Framingham MA 1992

[OMG 95] The Object Management Group 'Object Management Architecture Guide' 3rd Edition, John Wiley & Sons, 1995

[OMG 96] The Object Management Group: The Common Object Request Broker: Architecture and Specification, OMG Document Number 96.03.04, Revision 2.0 Framingham MA 1996

[OMG 97] The Object Management Group: CORBAservices: Common Object Services Specification, OMG – Dokument 97–02–24, Framingham MA, 1997

[OMG 98] The Object Management Group: CORBAservices: Common Object Services Specification, OMG Document 98–12–09, Framingham MA, 1998

[OMG 00] The Object Management Group: Notification Service Specification, Version 1.0, OMG Document 00-06-20, Framingham MA, 2000

[OMG 01] The Object Management Group: The Common Object Request Broker: Architecture and Specification, Revision 2.5, OMG Document Number 01-09-01, Framingham MA, 2001

[Orfali 96] Orfali, R.; Harkey, D.; Edwards, J.: 'The Essential Distributed Objects Survival Guide', John Wiley & Sons, Inc., New York, 1996

[Ott 98] Ott, E.: The Building Site Without Paper- Legal Aspects in an IT-Environment, URL=http://www.e-ott.com/stockholm_jun98.htm, 1998

[Patterson 90] Patterson, J. F.; Hill, R. D.; Rohall, S. L.; Meeks, S. W.: ´Rendezvous: an architecture for synchronous multi–user applications´, ACM Press New York, p. 317–328 Sereies–Proceeding–Article, 1990

[Peters 97] Peters, F.; Petersen, M.; Meißner, U.: Bestandsverwaltung und –bewirtschaftung durch integratives Informationsmanagement, IKM '97: Berichte, Bauhaus–Universität Weimar, 1997

[Pope 98] Pope, A.: The CORBA Reference Guide – Undestanding the Common Object Request Broker Architecture, Addison Wesley Longman Inc., Reading MA, 1998

[Posner 92] Posner, I.; Baecker, R.: How People Write Together, Proceedings of the 25th Annual Hawaii International Conference on the System Sciences, Hawaii, January 1992, IEEE '92

[Poyet 94] Poyet, P.: ATLAS integration tools, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Prakash 92] Prakash, A.; Knister, M.: Undoing Actions in Collaborative Work, in: Turner, J.; Kraut, R. E. (Hrsg): Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '92, Toronto, Canada, 1992

[Prakash 94] Prakash, A.; Knister, M.: A Framework for Undoing Actions in Collaborative Systems, ACM Transactions on Computer–Human–Interaction, Vol.1:4, December 1994, pp295–330, ACM Press 1994.

[Pree 97] Pree, W.: Komponentenbasierte Softwareentwicklung mit Frameworks, dpunkt–Verlag, Heidelberg, 1997

[Prinz 96] Prinz, W.; Kolvenbach, S.: Support for Workflows in a Ministerial Environment, in: Ackerman, M. S. (Hrsg.): proc. 7th Int. Conf. On Computer–Supported Cooperative Work, Boston, MA, New York: SIGCHI / SIGOIS ACM, 1996

[Ramscar 94] Ramscar, M.: Static models and dynamic designs – An empirical impasse vs. an inductive solution, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Richter 94] Richter, P.; Richter, C.: Meta–model for buildings, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Rodden 91] Rodden, T.; Blair, G.: CSCW and Distributed Systems: The Problem of Control, Proc. of the 2nd European Conference on Computer–Supported Cooperative Work (ECSCW '91) , Amsterdam 1991, Kluwer Academic Publishers 1991

[Rodden 98] Rodden, T.: CSCW Technologie: an overview, Computing Department Faculty of Applied Sciences, UK, Lancaster University, 1998

[Rodden 96] Rodden, T.: Populating the Application: A Model of Awareness for Cooperative Applications, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '96, Camebridge, MA, USA, 1996

[Roseman 92] Roseman, M.; Greenberg, S.: GroupKit – A Groupware Toolkit for Building Real–Time Conferencing Applications, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '92, Toronto, Canada, 1992

[Roseman 96] Roseman, M.; Greenberg,S.: Building Real–Time Groupware with GroupKit, A Groupware Toolkit, ToCHI Vol. 3, No. 1, Mar. 1996

[Rosenman 98] Rosenman, M. A.; Gero, J. S.: CAD Modelling in Multidisziplinary Desing Domains, in: Smith, I. (ed.);Artificial Intelligence in Structural Engineering–Information Technology for Design, Collaboration, Maintenance and Monitoring, Proceedings of EG–SEA–AI Lausanne 1998, Springer, Berlin, Heidelberg, New York, 1998

[Rüdebusch 93] Rüdebusch,T.: CSCW: Generische Unterstützung von Teamarbeit in verteilten DV–Systemen, Deutscher Universitäts–Verlag, Wiesbaden, 1993

[Rumbaugh 91] Rumbaugh, J.: Object–oriented modeling and design, Prentice Hall, Englewood Cliffs NJ, 1991

[Rumbaugh 99] Rumbaugh, J.; Jacobson, I.; Booch, G.: The Unified Modeling Language Reference Manual, Addison Wesley, Reading MA, 1999

[Sachs 95] Sachs, P.: Transforming work: Collaboration, Learning and Design, Communications of the ACM, Sept. 95, Vol.38, No.9, pp. 36–45

[Sauter 95] Sauter, C.; Morger, O.; Mühlherr, T.; Hutchinson, A.; Teufel, S.: CSCW for Strategic Management in Swiss Enterprises: an Empirical Study, in: Marmolin, H.; Sundblad, Y.; Schmidt, K. (ed.): Proc. of the Fourth European Conference on Computer Supported Cooperative Work 1995 (ECSCW '95), Kluwer Academic Publishers, Dordrecht, 1995

[Scherer 94] Scherer, R. J.: EU–project COMBI – Objectives and overview, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Scherer 98] Scherer, R. J.: A Framework for the Concurrent Engineering Environment, in: Amor, R. (ed.): Proc. of the 2nd European Conference On Product and Process

Modelling in the Building Industry – ECPPM '98, Watford UK 1998, BRE Ltd, 1998

[Schmitt 98] Schmitt, G.: A New Collaborative Design Environment for Engineers and Architects, in: Smith, I. (ed.): Artificial Intelligence in Structural Engineering– Information Technology for Design, Collaboration, Maintenance and Monitoring, Proceedings of EG–SEA–AI Lausanne 1998, Springer, Berlin, Heidelberg, New York, 1998

[Sharp 97] Sharp, A.: Smalltalk by example: the developers guide, McGraw–Hill, NewYork, 1997

[Shen 92] Shen, H., Dewan, P.: Access Control for Collaborative Environments, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '92, Toronto, Canada, 1992

[Sheth 90] Sheth, A. P.; Larson, J. A.: Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, ACM Computing Surveys, Vol. 22, No. 3, September 1990

[Siegel 97] Siegel, J.: CORBA fundamentals and programming, John Wiley & Sons Inc, 1996

[Simone 99] Simone, C., Mark, G.; Giubbilei, D.: Interoperability as a Means of Articulation Work, Proceedings of the ACM Conference on Work activities Coordination and Collaboration WACC '99, San Francisco, CA, USA, 1999

[Smith 93] Smith, G.; Rodden, T.: Access as a Means of Configuring Cooperative Interfaces, Proceedings of the ACM Conference on Organizational computing systems COCS '93, Milpitas, CA, USA, 1993

[Specht 01] Specht, M.; et al.: Authoring Adaptive Educational Hypermedia in WINDS, ABIS-Workshop 2001,8.-10. Oktober 2001 im Rahmen der Workshopwoche "Lernen–Lehren–Wissen–Adaptivität" (LLWA 01), Dortmund, GI-Fachgruppen 1.1.4 u. 2.3.3 'Adaptivität und Benutzermodellierung in Interaktiven Softwaresystemen', 2001

[Stefik 87] Stefik, M.; Bobrow, D. G.; Foster, G.; Lanning, S; Tatar, D.: WYSIWIS Revised: Early Experiences with Multiuser Interfaces, in: [Baecker 93], ACM Transactions on office Information Sysems, 1987

[Stefik 88] Stefik, M.; Foster, G.; Bobrow, D.; Kahn, K.; Lanning, S.; Suchman, L.: Beyond the Chalkboard: Computer Support for Collaboration an Problem Solving in Meetings, in [Greif 88a]

[Steinmann 95] Steinmann, F.; Hübler, R.: Knowledge support for functional and shape oriented design of buildings, in: Pahl, P. (ed.): Proceedings 6.ICCBE, Berlin, July 1995, Balkema, Rotterdam 1995

[Steinmann 95a] Steinmann, F.; Hübler, R.; Hauschild, Th.: Knowledge Support for Functional Design of Buildings, Proceedings IABSE Colloquium Bergamo, March 1995

[Steinmann 97a] Steinmann, F.: Modellbildung und computergestütztes Modellieren in frühen Phasen des architektonischen Entwurfs, Dissertation, Bauhaus–Universität Weimar, 1997

[Stephens 98] Stephens, J.; Böhms, M.; et al: Virtual Enterprises using Groupware tools and distributed Architektures, in: Amor, R. (ed.): Proc. of the 2nd European Conference On Product and Process Modelling in the Building Industry – ECPPM '98, Watford UK 1998, BRE Ltd, 1998

[Streitz 94] Streitz, N.; Geißler, J.; Haake, J. M.; Hol, J.: DOLPHIN: Integrated Meeting Support accross LiveBoards, Local and Remote Desktop Environments, in: Furuta, R.; Neuwirth, C. (Hrsg.): Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '94, Chapel Hill, United States, 1994

[Tang 90] Tang, J.; Minneman, S. L.: VideoDraw: A Video Interface for Collaborative Drawing, ACM CHI '90

[Teege 97] Teege, G.: CSCW–Systeme: Unterstützung von Gruppenarbeit, Skript Technische Universität München, 1997

[Teufel et al.95] Teufel, S.; Sauter, C.; Mühlherr, T.; Bauknecht, K.: Computerunterstützung für die Gruppenarbeit, Addison–Wesley, Bonn, 1995

[Tolman 94] Tolman, F.; Poyet, P.: The ATLAS models, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Trevor 94] Trevor, J.; Rodden, T.; Mariani, J.: The Use of Adapters to Support Cooperative Sharing, Proceedings of the ACM Conference on Computer–Supported Cooperative Work CSCW '94, Chapel Hill, United States, 1994

[Viller 91] Viller, S.: The Group Facilitator: A CSCW–Perspektive, Proceedings of the 2nd European Conference on Computer Supported Cooperative Work 1991 – ECSCW 91, in: [Baecker 93]

[Watson 94] Watson, A.; Crowley, A.: CIMsteel integration standards, in: Scherer, R. J. (ed.): Proc. of the 1st European Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[Wegner 86] Wegner, P.: Classification in Object–Oriented Systems, ACM SIGPLAN Notices October 86, 21(3):173–182, ACM, 1986

[Wehner 97] Wehner, R.; Steinmann, F.; Hübler, R.: FLEXOB–Entwicklungstool für dynamische modellbasierte CAD–Systeme, IKM '97: Berichte, Bauhaus–Universität Weimar, 1997

[Weikum 88] Weikum,G.: Transaktionen in Datenbanksystemen – Fehlertolerate Steuerung paralleler Abläufe, Addison Wesley Verlag, Bonn, 1988

[Werner 94] Werner, H.; Kowalczyk, W.: Representation of Product data Models that evolve during the design process, in: Scherer, R. J. (ed.): Proc. of the 1st European

Conference On Product and Process Modelling in the Building Industry – ECPPM '94, Dresden 1994, A.A. Balkema Rotterdam, 1995

[WfMC 95] The Workflow Management Coalition: The Workflow Reference Model, Document Number TC00–1003

[Willenbacher 98] Willenbacher, H.; et al. : AKO–Schnittstelle – Kopplung objektorientierter dynamischer Modellverwaltungssysteme; Dokumentation der AKO–Schnittstelle – Level 1, Bauhaus–Universität Weimar, Informations– und Wissensverarbeitung, 1998 URL=http://www.uni-weimar.de/Bauing/iwv/forschung/ako/akodokmain.html

[Willenbacher 99] Willenbacher, H.; Hübler, R.: Notification– und Securityservice als Bestandteil einer Schnittstelle für Modellverwaltungssysteme, in: Forum Bauinformatik, Darmstadt, 1999

[Willenbacher 00] Willenbacher, H.; Hübler, R.: Relationen zwischen Domänenmodellen – Ansatz zur Schaffung einer integrierenden computergestützten Bauplanungs-umgebung, in: IKM 2000, Weimar, 2000

[Willenbacher 01] Willenbacher, H.; Bubner, A.; Friedrich, T.: Verknüpfungsbasierter Bauwerksmodellansatz, in: 13. Forum Bauinformatik, München, 2001

[Wilson 91] Wilson, P.: 'Computer Supported Cooperative Work' Intellect Books, Oxford, UK, 1991

[WINDS 00] WINDS-Projekt : WINDS – Web-based Intelligent Design and Tutoring System, URL=http://www.winds-university.org und URL= http://www.fit.fraunhofer.de/projekte/winds/index_en.xml, 2000

[Winograd 88] Winograd, T.: A Language / Action Perspective on the Design of Cooperative Work. Human Computer Interaction, 1987–1988, Volume 3, Lawrence Erlbaum Associates Inc.

[Winograd 89] Winograd, T.: Groupware: The Next Wave or Just Another Advertising Slogan, IEEE Intellectual Leverage Digest of Papers, COMPCON 1989

[Wirfs–Brock 93] Wirfs–Brock, R.; Wilkerson, B.; Wiener, L.: Objektorientiertes Softwaredesign, Hanser Verlag München, 1993

[Woo 90] Woo, C.C.: SACT: a tool for automating semi-structured organizational communication, Proceedings of the ACM Conference on Office information Systems 1990 , Cambridge MA, ACM Press, 1990

[Zimbardo 92] Zimbardo, P.: Lehrbuch der Psychologie, 5.Auflage, Springer–Verlag Berlin Heidelberg New York, 1992