Fehldatenmodellierung

Ein intuitives Informationsaustauschmodell für den rechnergestützten Bauwerkslebenszyklus

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS

von der Fakultät für Architektur
der Universität Karlsruhe (Technische Hochschule)
genehmigte Dissertation

von Dipl.-Inform. Dirk Henckels
aus Neuss

Referent: Prof. Dr. ès. sc. tech. Niklaus Kohler
Koreferent: Prof. Dr.-Ing. habil. Reinhard Hübler
Tag der mündlichen Prüfung: 17. Mai 2005

Fehldatenmodellierung

Ein intuitives Informationsaustauschmodell für den rechnergestützten Bauwerkslebenszyklus

Dirk Henckels

Institut für Industrielle Bauproduktion
Universität Karlsruhe (TH)

Erklärung

Ich versichere wahrheitsgemäß, die Dissertation bis auf die in der Abhandlung angegebenen Hilfen selbständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

Neuss, den 22.10.2004

Inhalt

Abbildungsverzeichnis

1 Einleitung

2 Funktionale Sicht

3 Bauwerkslebenszyklus-Anforderungen

4 Modellierung von Fehldaten

5 Implementierung

6 Validierung

Vorwort und Danksagung

Die vorliegende Arbeit entstand konzeptionell im Rahmen einer Forschungstätigkeit am Institut für Industrielle Bauproduktion (ifib) der Universität Karlsruhe von Januar 1996 bis März 1999. Das ifib forscht seit ca. 1980 auf dem Gebiet der computergestützten Planungssysteme und ist seit dem Wechsel der Institutsleitung von Prof. Fritz Haller zu Prof. Niklaus Kohler auch auf dem Gebiet der Energie- und Stoffflüsse tätig.

Nachdem auf beiden genannten Gebieten merkliche Fortschritte erzielt wurden, besteht Bedarf an einer Verknüpfung der Daten und Methoden dieser Bereiche. Hierzu ist eine konzeptionelle Grundlage nötig, die sowohl für Planungssysteme als auch für Energie- und Stoffflußberechnungen ausreichend tragfähig ist. Diese Arbeit soll einen Modellierungsbeitrag hierzu leisten, der sowohl für Forschungsprojekte als auch für dem Prototypenstadium entwachsene Softwarelösungen geeignet sein kann.

Diese Arbeit ist natürlich entscheidend durch das ifib geprägt: durch seine Forschungsthemen, meine Kolleginnen und Kollegen dort, die Gäste in Vorträgen und Kolloqiuen. Allen Angehörigen des ifib möchte ich für die kreative und kritische Aufmerksamkeit bei unzähligen Gespräche danken.

Besonders danke ich fachlich wie menschlich meiner Partnerin Gabriele Blodau. Hervorheben möchte ich ihre Fähigkeit, konzeptionelle Probleme im Detail bereits auf theoretischer Basis früh aufzudecken. Gabi widme ich diese Arbeit.

Sehr wichtig waren mir auch die Gespräche mit Dr. Christian Müller, der unter anderem durch seine farbigen Beispiele (“vorgehängte transparente Holzlattenfassade”) jede kontroverse Sachdiskussion aufzulockern vermag.

Nicht zuletzt schulde ich besonderen Dank meinen Referenten, Prof. Niklaus Kohler insbesondere für seine Geduld im Verlauf der Ausarbeitung und vielfältige Anregungen sowie Prof. Reinhard Hübler für seine detaillierten Hinweise.

Neuss, im Oktober 2004

Dirk Henckels

1 Einleitung

1.1 Kurzfassung

Der Prozeß der Gebäudeplanung und -verwaltung ist durch seine vielen Beteiligten mit unterschiedlichen Sichtweisen und Interessen sehr komplex und erfordert umfangreiche Kommunikation. Obwohl eine Rechnerunterstützung zur Kommunikationsbeschleunigung und Kooperationsverbesserung mit dem Ziel eines konsistenten Informationsstatus aller Beteiligten vielversprechend scheint, beschränkt sich der praktische Einsatz von Computern in der Architektur häufig auf spezialisierte Werkzeuge wie CAD, AVA (Ausschreibung-Vergabe-Abrechnung) und Statik. Systemen, die eine umfassende Unterstützung mehrerer Aufgaben im Bauwesen anstreben, ist bisher eine ausreichende Akzeptanz versagt geblieben, obwohl durch Überschneidung in den bearbeiteten Daten ein Nutzen zweifelsfrei vorhanden wäre.

Die vorliegende Arbeit versucht einführend, Ursachen dieser mangelnden Akzeptanz aufzudecken und entsprechend Anforderungen an eine zufriedenstellende, umfassende Rechnerunterstützung des Bauwerkslebenszyklusses abzuleiten. Im Hauptteil der Arbeit wird ein diese Anforderungen erfüllendes Modell als "Fehldatenmodell" eingeführt. Seine Besonderheit liegt in der Behandlung gewünschter, aber noch nicht spezifizierter Information: Diese kann intuitiv beschrieben werden, ihre Beschaffung kann fallweise automatisiert sein. Durch diese Art der Datenverwaltung reicht das Fehldatensystem näher als bisherige Systeme an die traditionellen Arbeitsweise in der Planungsphase heran und bewahrt dabei die Vorteile des rechnergestützen Arbeitens bezüglich Informationsverarbeitung und Kommunikation. Die prototypische Implementierung versucht, durch minimierten Aufwand für manuelle Konfiguration und die leichte Erweiterbarkeit und Modifizierbarkeit darüber hinaus die eingangs erwähnten Akzeptanzprobleme weiter zu verringern.

1.2 Schwerpunkt und Umfang

Situation der bauausführenden Gewerke

Der Hochbau ist einer der vielschichtigsten Produktionsbereiche in Deutschland. Durch den hohen Grad an Individualität, der für Gebäude erforderlich ist, entsteht eine große Varianz an Lösungen und dadurch auch an Organisationsformen und Lösungsverfahren:

- Aus Produktsicht: Fertighäuser, Baukastensysteme sowie individueller Massivbau vor Ort

- Aus Planungs- und Ausführungssicht: Generalunternehmen, Verbünde einzelner Architektur- und Ingenierbüros sowie freie und einzelne Auftragsvergabe nach Gewerken durch die Bauherrenschaft

Allen Varianten gemeinsam, wenn auch unterschiedlich stark ausgeprägt, ist das Problem der im Vergleich zu anderen Branchen relativ geringen Produktivität. Diese hat unter anderem folgende Ursachen:

- die hohe Arbeitsintensität des Produktes durch die oft geforderte hohe Individualität. Diese Individualitätsforderung wird im folgenden "Unikateigenschaft" genannt.
- das hohe Maß an individueller Nacharbeit auch bei vorgefertigten Baukomponenten, das häufig aus nicht ausreichend detaillierter Planung und unexakter Ausführung resultiert.
- der immer größere Anteil von Erneuerungsarbeiten an der Gesamtheit der Bauvorhaben.

Hieraus entwickelte sich eine in der Praxis gereifte und wenig formalisierte Arbeitsweise im Bauwesen als heutiger Normalzustand.

Zusammen mit den nationalen baurechtlichen Forderungen führen diese Ursachen zu einem außergewöhnlich hohen Kostenniveau für das Bauen in Deutschland. Nicht zuletzt deshalb ist der Konkurrenzdruck bei bauausführenden und planenden Unternehmen sehr hoch.

Vor diesem Hintergrund haben sich etwa bei großen Bauaufträgen Generalunternehmer-Verträge durchgesetzt, in denen ein Unternehmen gegenüber der Bauherrenschaft alleinverantwortlich zeichnet. Meist sind dies große Bauunternehmen, je nach Projekt auch spezielle Planungsbüros. Nur Gewerke, die nicht im eigenen Unternehmen bearbeitet werden können, werden projektweise an Subunternehmen vergeben. Dies hat zu existenziellen Problemen bis hin zu Insolvenzen in erster Linie bei Kleinfirmen geführt, die mit der Mischkalkulation und den kleineren Gewinnspannen dieser Großfirmen nicht konkurrieren können.

Situation im administrativen Bereich

Nachdem in den letzten 20 Jahren die Prozeßautomatisierung in der industriellen Produktion üblich geworden ist, setzen sich nun auch prozeßorientierte Lösungen im administrativen Bereich durch, die unter dem Begriff Groupware zusammengefaßt werden. Dies könnte die erforderliche Basis bilden, um auch im Bauwesen mehrere kleine Unternehmen projektweise organisatorisch zu einem "virtuellen Unternehmen" zu verbinden, das auch rechtlich gegenüber der Bauherrenschaft als Generalunternehmen auftreten kann. Faktisch ist dieser Fall trotz des nötigen Handlungsdrucks selten. Der gewinnbringende alleinige Einsatz solcher Groupwaresysteme, die individuell angepaßt werden müssen, erfordert sowohl technisches als auch Anwendungsverständnis, das in dieser Kombination in den wenigsten Kleinunternehmen zu finden ist. Hieraus ist ein Marktsegment entstanden, das von Beratung bis Durchführung kundige Unterstützung anbietet und sich dabei häufig internetgestützter Hilfsmittel (etwa Portale, Application Service Providing) bedient.

Neben diesen technischen Schwierigkeiten gibt es für das seltene Auftreten virtueller Unternehmen aber auch substantielle organisatorische und rechtliche Ursachen. In der Automobilfertigung etwa existiert eine klare Hierachie aus Herstellern und Zulieferern, in der ein oder wenige Hersteller bzw. Herstellerverbünde die Datenformate bestimmen. Im Gegensatz dazu lassen sich im Bauwesen mit seinen häufig (noch) gleichberechtigten Planungs-, Rohbau- und Ausbaufirmen einheitliche Formate zum Datenaustausch oder Verfahren zur Entscheidungsfindung kaum durchsetzen.

Erster Schritt: Bauwerkslebenszyklus-Anforderungen

Die vorliegende Arbeit versucht, aus der mangelnden Akzeptanz vorhandener Lösungen und aus Voraussetzungen für eine technische Unterstützung einen informationstechnischen Anforderungskatalog abzuleiten, der eine geeignete Basis für den projektweisen Zusammenschluß einzelner Personen oder kleiner Unternehmen zur gemeinsamen Arbeit legt.

Zweiter Schritt: Fehldatenmodell; Ziel: explizite Behandlung fehlender Information

Sowohl die verfügbaren Produkte als auch aktuelle Forschungsprototypen, die diese Anforderungen erfüllen, sind aufgrund der Verteilung und der Unterschiedlichkeit der Arbeitsplätze und der Beteiligten in vielen Eigenschaften veränderlich. Durch die häufige Änderung der Projektparameter ist eine ständige Pflege der Konfiguration solcher Systeme erforderlich, so daß neben der produktorientierten Arbeit eine Ebene der “Meta-Arbeit” entsteht, die als nicht direkt produktive, systemtechnisch orientierte Tätigkeit den Eindruck vermittelt, das Datenaustausch- und Kommunikationsproblem durch ein Systemadministrationsproblem zu tauschen. Die Grundthese des Modellierungsteils in dieser Arbeit ist, daß solche Konfigurationspflege möglichst unaufwendig sein, idealerweise sogar unbemerkt und transparent ablaufen sollte. Dies entspricht aus wirtschaftlicher Sicht der Reduktion der *Total cost of Ownership*.

Das hier vorgestellte Fehldatenmodell versucht, dieser These Rechnung zu tragen. Das benötigte semantische Wissen liegt in Bereichen der computerbasierten Zusammenarbeit zu einem wesentlichen Teil darin, welche Informationen von den Beteiligten zur Weiterarbeit benötigt werden. Sind diese benötigten Informationen noch nicht vorhanden, werden sie im Folgenden "Fehldaten" genannt. Wenn das Wissen über das Fehlen konkreter Informationen systeminhärent ist, kann die Datenanforderung automatisch gesteuert werden, ohne spezielle Kommunikationsstrukturen aufsetzen zu müssen. Dies ist das Ziel des Fehldatenmodells. Für eine beschleunigte weitere Bearbeitung können Fehldaten solange durch Daten schlechterer Qualität, etwa durch Vorgabewerte oder Schätzungen, ersetzt werden, bis hochwertigere Informationen vorliegen.

Dritter Schritt: Umsetzung im Fehldatensystem; Ziel: Minimale Konfiguration

Das “Fehldatensystem” als Umsetzung des Fehldatenmodells beruht auf einer Modulstrukur. Jede(r) Beteiligte benutzt je nach Kompetenz und Verantwortlichkeit ein oder mehrere Module (Bild 1.1: Struktur des Fehldatensystems). Jedes Modul kennt die für seinen Wissenbereich nötigen Eingangs- und Ausgangsinformationen, kann also benötigte Informationen anfordern und produzierte (auf Anforderung) weitergeben. Fehldaten können bei Modulen, die diese Informationen produzieren, im Vorhinein angefordert werden. Je nach Verarbeitungslogik des angefragten Moduls können nun wiederum hierzu benötigte Eingangsinformationen gezielt angefordert oder die manuelle bzw. automatisierte Informationsverarbeitung entsprechend priorisiert werden.

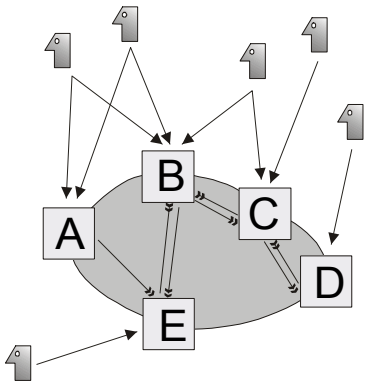

Bild 1.1: Struktur des Fehldatensystems

Beteiligte Personen nutzen im Fehldatensystem verschiedene Module A bis E. Diese kommunizieren Werte und Wertebedarf selbsttätig untereinander.

Die Minimierung des Konfigurationsaufwandes ist das zweite Ziel dieser Arbeit. Das Fehldatensystem kommt im Idealfall ohne manuelle Konfiguration zur Laufzeit aus. Die notwendigen, implizit existierenden Konfigurationsregeln sind demzufolge in jedem Modul implementiert. Dabei erfordert ein neues Fehldatenmodul nur die Kenntnis eines geringen Teils der Implementierung des Fehldatenmodells, da es aus einem bestehenden Template-Modul abgeleitet werden kann.

1.3 Gliederung

In Kapitel 2 ("Funktionale Sicht", Seite 7) wird zunächst das Anwendungsumfeld dieser Arbeit beschrieben. Der Schwerpunkt liegt hier auf dem zu unterstützenden Prozeß aus funktionaler Sicht. Kapitel 3 ("Bauwerkslebenszyklus-Anforderungen", Seite 21) betrachtet aus technischer Sicht Rahmenbedingungen und Möglichkeiten für eine Lösung der gestellten Aufgabe.

Die Grundlage für ein konkretes Bauwerkslebenszyklus-System legt Kapitel 4 ("Modellierung von Fehldaten", Seite 39). Dieses Kapitel behandelt die Datenmodellierung schwerpunktmäßig im Hinblick auf Erweiterbarkeit und Verteilung. Hierdurch ist bereits die Kooperation verschiedener Module in einem Fehldatensystem angesprochen, die Kapitel 5 ("Implementierung", Seite 57) unter dem Gesichtspunkt der einfachen Konfigurierbarkeit vertieft. Diese beiden Kapitel legen die Implementierungsgrundlagen.

Kapitel 6 ("Validierung", Seite 83) überprüft das modellierte Konzept anhand einer prototypischen Implementierung. Es beschreibt, daß das Fehldatensystem bei Veränderungen zur Laufzeit phänomenologisch ein deutlich anderes Verhalten als bestehende Softwarelösungen aufweist. Die Untersuchung der Unterschiede schließen diese Arbeit ab.

2 Funktionale Sicht

Diese Arbeit fußt im Bereich der informationstechnischen Unterstützung für den gesamten Prozeß von der Vorplanung und Ausschreibung eines Bauprojektes über Planung und Realisierung bis hin zu Nutzung und Umnutzung, das im folgenden verkürzt als "Bauwerkslebenszyklus-Problem" bezeichnet wird. Hierfür eine Lösung zu finden, läßt sich aus inhaltlicher wie auch historischer Perspektive unter folgenden grundlegenden Aspekten betrachten:

- Die Planung, speziell von Neubauten, ist das am besten untersuchte Problem im Bereich informationstechnischer Unterstützung des Bauwesens. Sie bildet daher den Ausgangspunkt der Prozeßbetrachtung.
- Informationstechnische Hilfsmittel stehen sowohl für branchenspezifische Anwendungen als auch in Form allgemeiner Werkzeuge für Datenaustausch und Kollaboration zur Verfügung. Verschiedene Standardisierungsbemühungen kommen hinzu.
- Die Betrachtung des Lebenszyklusses des Gebäudes hat noch wenig Eingang in den Arbeitsalltag der Planung gefunden, ist aber über viele Jahre theoretisch fundiert, so daß für die Modellierung des Lösungsraumes insgesamt eine zufriedenstellende Basis besteht.

Diese drei Blickwinkel gliedern dieses Kapitel.

2.1 Gebäudeplanung und weitere Teilprozesse

Der Planungsprozeß ist eine Kombination aus kreativer, inspirationsgetriebener Arbeit und ingenieurmäßigem Vorgehen [Rechtin et al. 1997]. Als "Konstruktiver Entwicklungsprozeß" ist er in der ingenieurwissenschaftlichen Forschung seit längerem Thema. [Hansen 1974] beschreibt, daß das Finden einer optimalen Lösung nicht mit Sicherheit möglich ist, sondern jede Lösung *a posteriori* gegen die Anforderungen und Alternativlösungen bewertet werden muß. Hierin liegt ein wesentlicher Unterschied gegenüber technischen Prozessen, die auf jede Dateneingabe mit eindeutiger Verhaltensweise reagieren und damit in ihrer Ergebnisqualität *a priori* abschätzbar sind. Diese Klasse der Probleme, die erst nach ihrer Lösung komplett verstanden sind, wird auch als *Wicked Problems* bezeichnet [Churchman 1967].

Für die Unterstützung der kooperativen Architekturplanung gilt erschwerend, daß sowohl veränderliche Kooperationen [Fitzpatrick 1998] als auch die architektonische Planung selbst [Rittel et al. 1973] in diesem Sinne als *Wicked Problems* betrachtet werden können. Die Beherrschbarkeit dieser vorab nicht genau bekannten

Probleme ist damit die Hauptschwierigkeit bei der Unterstützung kooperativer Arbeit im Bauwesen.

Mit der Hinzunahme weiterer Lebenszyklusphasen steigt sowohl hinsichtlich der Kooperation als auch der Planungsanforderungen die Komplexität. Da eine kooperative Lösung jede einzelne Tätigkeit (in gewissen Grenzen) antizipieren muß, um einen geordneten Projektablauf sicherzustellen, wird im folgenden mit "Planung" der allgemeine Vorgang der Vorwegnahme projektbezogener Vorgänge bezeichnet, wohingegen "Bauplanung" oder "Gebäudeplanung" sich ausschließlich auf einen zukünftigen Zustand des Zielobjektes bezieht.

2.2 Heutiger Einsatz von Informationstechnik

Die Praxis in Planungsbüros ist heute der Einsatz von Rechnern zur Unterstützung des Zeichenvorganges unter Einsatz von CAD-Systemen. Deren Layertechnik, ursprünglich aus der Fotografie entlehnt, hat sich über alle technischen Fortschritte hinweg als sehr leistungsfähig erwiesen. Sie wird heute durch eine objektorientierte Arbeitsweise ergänzt.

Die Grenzen der CAD-Systeme liegen einerseits beim Datenaustausch mit anderen Applikationen, etwa zur Simulation oder Präsentation, andererseits beim Versuch der Anwendung in den frühen Planungsphasen mit geringem Detaillierungs- und Exaktheitsgrad.

2.2.1 Produktmodellierung und Datenaustausch

Die direkte Verknüpfung von Programmen (z.B. CAD und Simulation) ist in der Forschung bereits gelungen, solange die Einsatzgebiete hinsichtlich der Applikationsart [NN 1998: 4D-CAD] oder der baulichen Anwendung [Hovestadt et al. 1999] [Henckels et al. 1997] beschränkt sind. Da einige Ansätze zur Produktmodellierung aus früheren Lösungen zum Datenaustausch entstanden sind, faßt dieser Abschnitt beide Gebiete zusammen.

CAD-Programmen bieten seit etwa 1999 Funktionalität zur Unterstützung von Teamarbeit [Graphisoft 1999]. Allerdings sind dies bislang Funktionen zum Dokumentenaustausch und *Redlining*, so daß von einer wirklichen Integration nicht gesprochen werden kann. Das immernoch gültige Ideal einer allumfassenden Lösung des Entwurfsproblems durch Informationstechnik beschäftigt die Forschung bereits seit spätestens den 70er Jahren [Haller 1974] [Hansen 1974] [Bijl 89], seine Grundlagen reichen jedoch noch weiter zurück [Archer 1964] [Asimow 1964]. Bereits mit [Hovestadt 1994] wurde ein Schlußstrich unter selbständig planende und lösungsentwerfende Systeme gezogen, da diese durch direkte oder indirekte Eingriffe die Planungsfreiheit einschränken. Jüngere Arbeiten zielen auf assistierende

Systeme, die Werkzeuge zur Unterstützung und damit Effizienzsteigerung der Planungsbeteiligten zur Verfügung stellen.

Produktmodelle

Allen Applikationen liegt ein mehr oder minder komplexes Datenmodell zugrunde. Die Abbildung eines realen Produktes auf ein rechnerinternes Datenmodell heißt Produktmodell [Seiler 1985], wobei das Produkt nicht notwendigerweise gegenständlich sein muß. Die Notwendigkeit eines Produktmodells zur Lösung rein ingenieurwissenschaftlicher Entwicklungsaufgaben ist allgemein anerkannt [Böhms et al. 1994] [Augenbroe 1995] [Eastman et al. 1997]. Jedoch ist der Gebäudelebenszyklus speziell durch seine künstlerisch-kreative Komponente in der Planungsphase nicht rein ingenieurwissenschaftlich. Plattformen wie ArchE [Hovestadt et al. 1999] oder die neueren, kooperationsorientierten Plattformen[1] [Müller 1999] begnügen sich deshalb mit einfachen Datenstrukturen und versuchen teilweise, hieraus Vorteile für die kreative Seite des Prozesses zu ziehen. Auch gibt es viele Versuche, wissensbasierte Systeme zu gestalten, die das Menschheitswissen sammeln und in für Computer verarbeitbarer Form zur Verfügung stellen können (etwa [Boley 1996]), allerdings haben diese noch keinen großen Einfluß auf kollaborative Systeme im Bauwesen gewonnen.

Ein gemeinsames Produktmodell mehrerer Applikationen vereinfacht den Datenaustausch. Transformationen (auch Relationen genannt) zwischen unterschiedlichen Produktmodellen können dabei neben der reinen Datenübernahme auch die Konvertierung bzw. Berechnung von Werten leisten [Willenbacher et al. 1998]. Transformationen unterliegen aber denselben Problemen hinsichtlich des schwer vorherzusagenden zukünftig nötigen Umfangs wie Produktmodelle selbst. Da zudem keine Applikation alle möglichen Informationen liefern wird, muß ein unvollständiger Datenexport akzeptiert werden. Auf der Importseite kann wiederum nur ein eingeschränkter Teil des Datenmodells verarbeitet werden, so daß nur die Schnittmenge von Export- und Importmenge effektiv übertragen werden (Bild 2.1: Beschränkung des Datentransfers auf die Import-/Exportschnittmenge.). Datenverluste sind damit üblich [Gehrlein 1999] [Rudolph et al. 1999] [Willenbacher 2002].

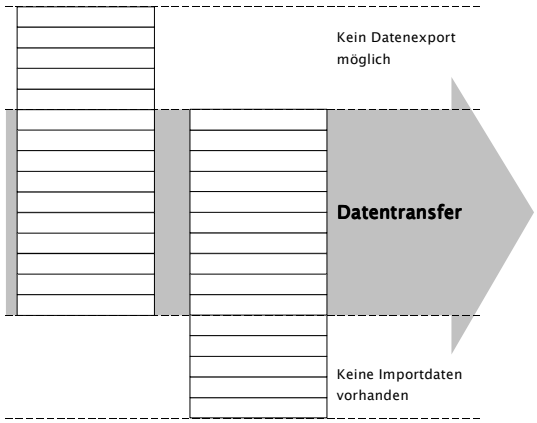

Bild 2.1: Beschränkung des Datentransfers auf die Import-/Exportschnittmenge.

1. aber auch das hier beschriebene Fehldatenmodell

nicht objektorientierte Modellierung: DXF und STEP

Zunächst der *de-facto*-Standard DXF [Rudolph et al. 1999], später STEP [ISO 1996: STEP] als die praktisch bedeutendsten Ansätze für Interoperabilität greifen die Transformation von Daten zwischen unterschiedlichen Produktmodellen auf, indem sie Datenmodelle definieren, die für den Informationsaustausch zwischen Applikationen entworfen wurden. Das rein darstellungsorientierte DXF bietet zur Modellierung keine ausreichende Anzahl Grundkonstrukte, da etwa Kreissegmente nur durch Polygone angenähert werden können. Es war ursprünglich jedoch auch nur für den Datenaustausch spezifiziert worden. Aktueller ist das *"Scalable Vector Graphics"*-Format SVG [W3C 2003], dessen Design jedoch auf Webanwendungen ausgerichtet ist. STEP zeigt bereits Aspekte der Objektorientierung und verfügt neben der Modellierung selbst mit EXPRESS-X über die Fähigkeit zur formalen Definition von Abbildungen zwischen Datenmodellen, dem *"Schema Mapping"*. Auch STEP konnte sich jedoch aufgrund mangelnden Umfangs im Bauwesen nicht durchsetzen, während es etwa im Maschinenbau weitgehend etabliert ist.

Objektorientierte Modellierung mit IFCs

Die objektorientierte Denkweise, die sich in den meisten aktuellen Applikationen als Modellierungsparadigma wiederfindet, mit Kapselung von Informationen und Vererbungsmechanismen erlaubt die Beherrschung weit komplexerer Sachverhalte in der Modellierung und Implementierung von Software. Die Industry Foundation Classes (IFCs) [IAI 1997] sind durch ihre vollständige Objektorientierung prinzipiell besser als STEP und EXPRESS auch zur applikationsinternen Informationsrepräsentation geeignet. Damit verringern sie potentiell auch die Transformationsverluste. Weiterhin finden auch Prozeßaspekte zunehmend Eingang in das Modell [IAI 2003].

Die objektorientierte Implementierung erfordert zur Strukturierung jedoch eine statische Hierarchie von Objekten, entlang derer Objekte Eigenschaften "erben". Sie hat damit eine systeminhärente Schwäche: Wenn Eigenschaften und Verhalten von Objekten zur Laufzeit änderbar und erweiterbar sein sollen, das heißt eine Dynamisierung des Modells [Laabs et al. 1997] erfolgt, ist diese statische Hierarchie kontraproduktiv, sobald die Änderungen und Erweiterung der bisherigen Vererbungshierarchie zuwider laufen. Die objektorientierte Implementierung setzt damit der Dynamisierung des Modells Grenzen. Somit erscheint es möglich, daß auch die *International Alliance for Interoperability* mit ihren IFCs am Vorhaben der allumfassenden Integration konzeptbedingt scheitern könnte.

2.2.2 Implizite Vorgänge

In der Forschung wenig diskutiert wird jedoch, daß bei heutigen Planungsplattformen nicht eine Verbesserung des baulichen Ergebnisses, sondern des Vorganges der Planung im Vordergrund steht. Die Folge sind Werkzeuge, die sich explizit mit dem bisher impliziten, stillschweigend funktionierenden Teil des Planungsvorganges beschäftigen. Die impliziten (üblicherweise "im Kopf" stattfindenden Teile) der Planung sind offenbar schwer formalisierbar. Die bisherigen Projekte haben stets auch versucht, in die frühen Planungsphasen vorzudringen und damit einen Teil der "impliziten" Gedankentätigkeit explizit zu dokumentieren. Die implizite Arbeit ist jedoch in den frühen Planungsphasen effektiver:

- Durch die aus der fehlenden Dokumentation resultierenden Geschwindigkeit, mit der Alternativen durchdacht und verworfen werden können, sind traditionelle Vorgehensweisen (z. B. Skizzen) bis zu einem gewissen Detaillierungsgrad sehr schnell.
- Die Objekte einer typischen architektonischen Skizze werden durch die Datenerfassung zwangsweise an exakte Maße gebunden wird, obwohl eine derart punktgenaue Festlegung noch nicht stattgefunden hat. Diese Fehlinformation wirkt irritierend.
- Die existierenden Mensch-Maschine-Schnittstellen mit 2D-Bildschirmen, Maus, Digitisier-Tablett und Tastatur sowie gegebenenfalls Shutter-Brille und Space-Maus erlauben keine "naturnahe", intuitive Eingabe von dreidimensional angeordneten Gebäudedaten. Der Umstand der nötigen Adaption wird erst für den Gewinn sauberer, leicht modifizierbarer Darstellungen in späten Planungsphasen gerne in Kauf genommen.

[Norman 1990] führt für im Alltagsbetrieb zu benutzende Gegenstände *"Principles of Design for Understandibility and Usability"* aus, unter anderem:

"provide a good conceptual model: A good conceptual model allows us to predict the effects of our actions. ... [Conceptual models] are part of an important concept in design: mental models*, the models people have of themselves, others, the environment, and the things with which they interact. People form mental models through experience, training, and instruction. The mental model of a device is formed largely by interpreting its perceived actions and its visible structure."*

Für ein geeignetes informationstechnisches System sind intuitive Betrachtungsmechanismen der Architektur im räumlichen Umfeld ebenso nötig wie nachvollziehbare und gegebenenfalls anpaßbare Vorgaben zum Planungsvorgang selbst. Beides spricht dafür, ein Vorgehen zu modellieren, das Assoziationen zum konventionellen Planungsablauf weckt. Wenn Abweichungen von der üblichen und gut beherrschten konventionellen Vorgehensweise erzwungen werden, müssen sich diese aus einer Verbesserung des Arbeitsablaufes rechtfertigen.

Insgesamt liegt die Vermutung nahe, daß explizite Lösungen für typischerweise implizit ablaufenden Phasen nur zufriedenstellen, wenn der bei Computerprogrammen häufige Konfigurationsaufwand bereits als selbstverständlich angenommen wird. Auch deshalb könnte die praktische Akzeptanz von Planungswerkzeugen im Bauwesen, das Computer eher spät einsetzte, bisher relativ gering sein. Die vorliegende Arbeit kehrt daher von der Optimierung der impliziten (und zumeist gut funktionierenden) Planungsaufgaben ab und arbeitet stattdessen auf die optimaler Handhabung der expliziten Aufgaben hin.

2.3 Gebäudelebenszyklus

Phasenstruktur

Die "Honorarordnung für Architekten und Ingenieure" (HOAI) [NN 1996: HOAI] bietet die in Deutschland rechtlich verbindliche Grundlage zur Strukturierung des Gebäudeplanungsprozesses. Sie umfaßt Phasen, deren Umfang und Honorarverteilung den Rückschluß zuläßt, daß beim Entwurf der HOAI eine sequentielle Abarbeitung unterstellt wurde. Da die Integration der Tätigkeiten zu einer Effizienz- und Qualitätsverbesserung führen kann, weil Zeit- und Personalaufwand gespart sowie Fehler durch manuelle Datenübertragung und Mißverständnisse vermieden werden können [Kohler et al. 1997], ist dieses Modell unzureichend. Für eine Parallelisierung der Bearbeitung ist eine sorgfältige Arbeitsvorbereitung nötig, die sich in einer stärkeren Gewichtung der frühen Phasen widerspiegeln müßte [SIA 1996].

Entsprechend sind weitere Modelle entstanden, die stärker als die HOAI alle Tätigkeiten im gesamten Bauwerkslebenszyklus beachten. Dieser Lebenszyklus unterscheidet sich nicht strukturell, wohl aber in der Gewichtung seiner Phasen vom Lebenszyklus anderer Produkte [Bundestag 1995]. Die Unterschiede zu anderen Produktentwicklungsprozessen sind durch die entscheidend höhere Komplexität und Individualität eines Gebäudes so deutlich, daß die naheliegende Einordnung als Sonderfall der Investitionsgüterentwicklung kaum weiterhilft. Insbesondere wird das Gebäude häufig aus individuell angefertigten Teilen erstellt, die zum Teil wegen der Immobilität des Produktes vor Ort fabriziert werden[2]. Insofern ist die Bauproduktion das Musterbeispiel der Unikatfertigung schlechthin.

Dieser Abschnitt diskutiert mit einer Erweiterung der Integralen Planung um den Detaillierungsgrad sowie dem Drei-Ebenen-Modell zwei unterschiedliche Ansätze zur Beschreibung des Lebenszyklusses.

2.3.1 Integrale Planung

Lebenszyklus

Die grundlegenden informationslogistischen Aspekte im Planungsprozeß sind:

- Transfer zwischen Personen: Welche Form des Austausch zwischen den Beteiligten ist gewünscht und erforderlich?
- Transfer über die Zeit: Welche Daten werden in welcher Phase benötigt?
- Transfer zwischen Betrachtungsebenen: Welcher Detaillierungsgrad ist wann und für wen nötig?

Diese drei Aspekte können als Charakteristika des Planungsprozesses betrachtet werden. Damit ist der Planungsprozeß keineswegs vollständig oder eindeutig klassifiziert; stattdessen bieten diese Fragen eine Hilfe für die Interpretation der Gebäudeplanung aus informationstechnischer Perspektive, wie die folgenden Abschnitte zeigen.

2. Beispielhaft seien zum einen Wohnbauten genannt, bei denen trotz der hohen Kosten meist individuell geplante Einfamilienhäuser als erstrebenswert gelten, zum anderen der immernoch hohe Ortbetonanteil in Industriebauten.

Begriff der Integralen Planung

Der Begriffe der “Integralen Planung” ist in der Architektur allgemein geläufig, aber selten definiert. Die erstmalige Definition erfolgte vermutlich in [Suter et al. 1986] und betont den Unterschied zwischen serieller Planung, wie sie die deutsche HOAI strukturell voraussetzt, und der Parallelisierung der Integralen Planung, die insbesondere Rückkopplungen nach Art des *Concurrent Engeneering* zuläßt.

[Stulz 1993] spricht von “Integraler Planung”, wenn der Führung eines Bauprojektes wegen der Projektkomplexität eine Person zur technischen Koordination zur Seite steht, um Bauherrschaft, Behörden, Planer(innen), Fachspezialisten, Benutzer(innen) und Unternehmen zu beteiligen. Die Bindung dieser Aufgabe an eine separate Position im Projekt verdeutlicht die Komplexität der Aufgabe, läßt aber zugleich die Definition ungeeignet erscheinen, wenn technische Hilfsmittel den manuellen Aufwand der Koordination verringern können.

Für eine informationstechnische Auswertung ausreichend dokumentierte Fallbeispiele Integraler Planung existieren bislang nicht. Allen Definitionen und Anwendungen ist jedoch die Voraussetzung eines interdisziplinären Teams gemeinsam, das von Anfang an zusammen und parallel arbeitet. Damit wäre eine Vergrößerung des Aufwandes für Datenaustausch und Kommunikationsabläufe zu erwarten, so daß besonderes Augenmerk auf den Abgleich der Informationen und Interessen der Beteiligten zu richten ist.

In [Stulz 1993] [Kohler et al. 1996: KOBEK] wird Integrale Planung anhand zweier Integrationsrichtungen definiert: Die “horizontale Integration” der beteiligten Personen innerhalb einer Lebensphase und die “vertikale Integration” der Informationen über verschiedene Phasen hinweg. Dies dient als Ausgangspunkt für die folgenden Ausführungen.

Personen

Fachwissen

Die Bauplanung wird nicht durch eine einzelne Person, sondern durch mehrere (bei großen Projekten sehr viele) Personen recht hohen Spezialisierungsgrades durchgeführt. Deren Zusammenarbeit ist durch folgende Punkte gekennzeichnet:

- Jede Person hat einen individuellen Kenntnisstand, der ihr Arbeitsfeld bestimmt. Es existiert eine Menge von Planungsbereichen, in denen jeweils eine oder mehrere Personen gemäß ihrem Arbeitsfeld tätig sind. Die zu einem Planungsbereich gehörenden Personen, die untereinander und mit Außenstehenden kooperieren, werden Team genannt [Teufel et al. 1995].
- Jedes Team benötigt für die Arbeit in seinem Planungsbereich unterschiedliche Informationen. Optimalerweise können Informationen einzelnen Baukomponenten oder Planungsvorgängen zugeordnet werden. Dies entspräche einer ergebnisorientierten Arbeitsweise und läßt eine gute Arbeitsunterstützung erhoffen. Dann jedoch benötigt jedes Team nur eine Teilmenge dieser Komponenten oder Vorgänge, und aus diesen wiederum nicht alle Daten (Bild 2.2: Horizontale Integration)

Organisationsstrukturen

Üblicherweise gehören die Projektbeteiligten mehreren kleinen bzw. mittelständischen Unternehmen an, die bauprojektweise zusammenarbeiten. In jüngerer Zeit

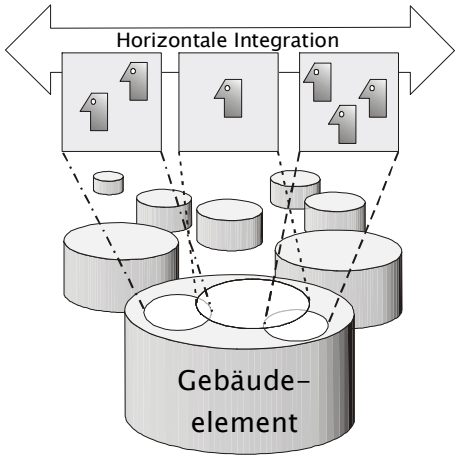

Bild 2.2: Horizontale Integration

Einzelne Teams oder Personen benötigen je nach Anwendung nur Teile der Daten, die über ein Element verfügbar sind. Die Integration dieser Sichten heißt "Horizontale Integration".

etablieren sich zur Steigerung der Arbeitseffizienz allerdings zwei neue Strukturen:

- Generalunternehmen erhalten den Gesamtauftrag und die Verantwortung für seine Abwicklung. Diese sind zum Teil auf Planung spezialisierte Ingenieurbüros, die Unteraufträge vergeben, häufig aber große Baufirmen, die mit entsprechenden Fachabteilungen einen Großteil der Wertschöpfung im eigenen Haus erwirtschaften und damit den gewachsenen Strukturen im Bauwesen "das Wasser abgraben".
- Festere Bindungen kleinerer Firmen versuchen die Konkurrenzfähigkeit zu großen Firmen zu bewahren. Dies sind entweder Bietergemeinschaften, die über längere Zeit zusammenarbeiten, oder "Virtuelle Unternehmen" [Pfaus 1998] [Scheer et al. 1996], die vertraglich gebunden sind und gegenüber den Auftraggebern als Generalunternehmen auftreten [Fink 1998].

Die Zusammenarbeit verschiedener Projektbeteiligter in einem Gebäudeplanungsprozeß wird im Fall des Generalunternehmungsvertrages von einem Projektsteuerer moderiert und geleitet. Verbünde kleiner Unternehmen werden meist durch den Architekten oder die Architektin koordiniert.

Horizontale Integration

Einerseits durch die unterschiedlichen Rollen in der Projektorganisation, andererseits durch verschiedene Fachrichtungen und Kenntnisstände der Beteiligten entstehen unterschiedliche Betrachtungsweisen der Planung. Die Zusammenführung dieser Betrachtungsweisen zu einem durchgängigen Planungs- und letztlich Gebäudekonzept wird als "Horizontale Integration" bezeichnet (Bild 2.2: Horizontale Integration).

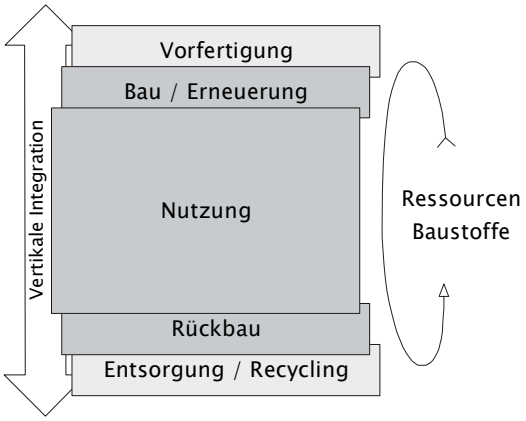

Bild 2.3: Vertikale Integration

Lebenszyklus eines Gebäudes bzw. Bauteils und dessen Vertikale Integration. Die Nutzungsphase ist die längste Phase.

Zeit

Die Beteiligten am Planungsprozeß müssen neben geeigneter Kommunikation untereinander auch die Aufgabe bewältigen, Informationen aus einer Phase in die nächste zu überführen – etwa aus einem Gespräch mit der Bauherrenschaft in ein funktionales oder Raumkonzept des Gebäudes oder aus einem derartigen Konzept und eventuell vorhandenem Bestand in einen Entwurf.

Lebensphasen

Hierzu gibt es verschiedene Varianten, den Lebenszyklus eines Gebäudes in Lebensphasen aufzuteilen. Bild 2.3: Vertikale Integration zeigt eine Aufteilung in Anlehnung an [Kohler et al. 1997]. Die Nutzungsphase ist betont, da sie die üblicherweise längste Phase ist. Für einzelne Teile eines Gebäudes können sich deren Lebensphasen überschneiden[3].

Der Wechsel zwischen Lebensphasen kann als Informationstransfer über die Zeitachse verstanden werden und darf sich auf diejenige Information beschränken, die für kommende Phasen verwendbar sind: Die Betrachtung der Gebäudeinformationen in unterschiedlichen Phasen erfolgt zu unterschiedlichen Zwecken, so daß für die für die aktuelle Phase unterschiedliche Teilmengen der Information interessant sind. Relativ gut beherrscht ist hier der Transfer geometrischer Informationen, etwa aus der Vorplanung in die Werkplanung, wenngleich der Transfer von dort in das Facility Management der Nutzungsphase noch selten ist.

Neben dieser Datenübernahme in nachfolgende Lebensphasen müssen fallweise Daten späterer Phasen im Vorhinein bekannt sein[4]. Diese Antizipation ist Bestandteil jeder Neubauplanung, wird jedoch erst in der Integralen Planung in der Modellierung berücksichtigt. Entsprechend sind Rückkopplungen zur Validierung der

3. So kann durch Renovierung zum Beispiel nach 30 Jahren die Dacheindeckung rückgebaut und erneuert werden, während der Rest des Gebäudes in der Nutzung ist.

4. Für die Planung etwa sind Informationen der Nutzungsphase relevant, etwa: Wie viele Personen werden in einem Gebäude gleichzeitig anwesend sein? Welcher Bedarf an Ver- und Entsorgung (Strom, Wasser, Klima) besteht?

Ergebnisse möglich, die wertvolle Hinweise zur Verbesserung der Prozesse liefern können.

Dieser Aspekt der Integration über die Zeitachse wird als “Vertikale Integration” bezeichnet (Bild 2.3: Vertikale Integration).

Detaillierungsgrad

iterativ-rekursive Planung

Betrachtet man den Planungsprozeß unter informationstechnischen Aspekten, kann man sich diesen vereinfacht aus einer beliebigen Kombination von Iteration und Rekursion von Planungsschritten zusammengesetzt denken [Henckels 1994] (Bild 2.4: Planungsprozeß), wobei jeder Planungsschritt grob in Konzeption, Entwurf und Bewertung aufteilbar ist.

Ein Iterationsschritt entspricht hierbei der Revision der Planung. Über die Iteration gibt es eine Rückkopplung, eine Bewertung des Planungsstandes gegen vorhandene Alternativen bzw. festgelegte Anforderungen aus Auftrag oder Gesetz. Die Bauplanung kann daher als manuell ausgeführte Regelung eines Prozesses verstanden werden, in dessen Verlauf das Planungsergebnis ausdetailliert wird. Ein Rekursionsschritt zerlegt die Planungsaufgabe in Teilaufgaben, die wieder als eigene Planungsschritte betrachtet werden. Jeder Wechsel der Rekursionstiefe impliziert zudem einen Wechsel des Detaillierungsgrades, da tiefere Rekursionsebenen Teilprobleme genauer, das heißt “gezoomt” oder weniger abstrakt betrachten.

Bestimmte Abstraktionsebenen können trotz der prinzipiell beliebig inviduellen Vorgehensweise in jedem Planungsprojekt identifiziert werden: Das Ergebnis der Anforderungsermittlung kann zum Beispiel in einem oder mehreren Raumbüchern festgehalten werden [Dingler 1999]. Sofern mehrere Raumbücher verwendet werden, unterscheiden sich diese in Sichtweise und Detaillierungsgrad. Je nach Variante entspricht das Raumbuch etwa dem Pflichtenheft in der Informationstechnik. Es folgt ein wenig formalisierter (und bislang auch wenig formalisierbarer) Prozeß

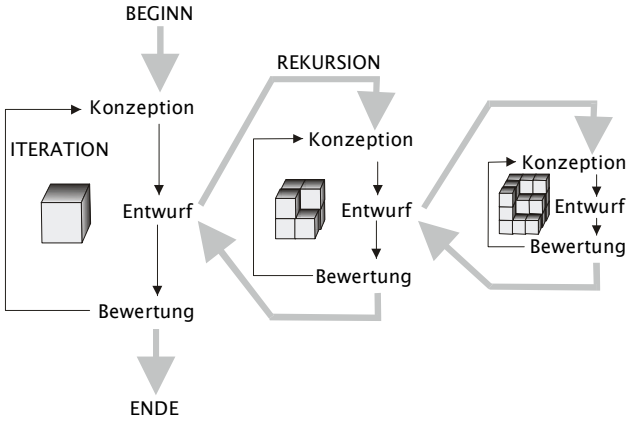

Bild 2.4: Planungsprozeß

Planung als kombinierte Iteration und Rekursion der Schritte Konzeption, Entwurf und Bewertung.

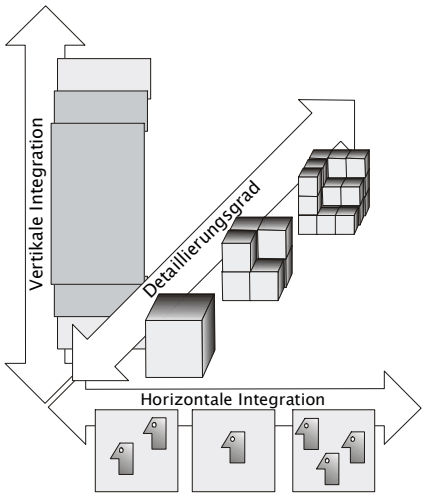

Bild 2.5: Detaillierungsgrad

Erweiterung der Integralen Planung (Vertikale u. Horizontale Integration) um den Detaillierungsgrad als kontextbestimmenden Aspekt.

der Festlegung von funktionalen Zusammenhängen und im Idealfall einer Entwurfsidee, die als Leitlinie die gestalterische Seite des Gebäudeentwurfs prägt. Anschließend wird die geometrische Umsetzung gestaltet, die ihrerseits auf verschiedenen Abstraktionsebenen erfolgt, bis die Detaillierung für die Ausfertigung von Werkplänen ausreicht.

In das Modell der Integralen Planung aus Horizontaler Integration (Bild 2.2: Horizontale Integration) und Vertikaler Integration (Bild 2.3: Vertikale Integration) einbezogen, ist der Detaillierungsgrad der aktuellen Betrachtung neben Personen und Zeit ein dritter Aspekt, der zur Einschränkung der notwendigen Informationen herangezogen werden kann (Bild 2.5: Detaillierungsgrad): Jeder Informationszugriff läßt sich nach diesen drei Aspekten einordnen, wird also in einem "Kontext" aus Beteiligten, Gebäude-Lebensphase und Detaillierung durchgeführt. Dieser Kontext entspricht einer "Sicht" im Sinne der Datenbanktechnik [Lockemann et al. 1993]. Diese Sicht kann weiter eingeschränkt werden, etwa durch die Auswahl eines geeigneten geometrischen Ausschnitts oder eine Einschränkung der betrachteten Informationen nach vorgegebenen Kriterien (etwa "nur statisch relevante Bauteile"). Da diese Sichten sich nur auf die Darstellung der Information zur ihrer leichteren Bearbeitung, nicht aber auf die grundlegende Relevanz der Information für die entsprechende Person im entsprechenden Moment auswirken, gehen sie jedoch nicht in den aktuellen Kontext nach obiger Definition ein.

2.3.2 Drei-Ebenen-Modell

[Kohler et al. 1997] zeigt ein Drei-Ebenen-Modell, das die komplexe Zusammenarbeit im Bauwesen so zusammenzufassen versucht, daß Tätigkeiten mit ähnlichem Informationsbedarf gruppiert werden (Bild 2.6: Drei-Ebenen-Modell zur Strukturierung von Integrationsumgebungen.). Ziel war die Gruppierung und

Schnittstellendefinition von verschiedenen Aufgaben innerhalb des Bauwerkslebenszyklusses.

- Die Kooperationsebene behandelt die Abwicklung der Mensch-zu-Mensch-Kommunikation. Aus technischer Perspektive sind auf dieser Ebene Fragen des Dokumenten- und Nachrichtenaustauschs, aus Anwendungsperspektive Probleme der Entscheidungsfindung und der Projektkoordination angesiedelt. Sie umfaßt die nicht-räumlichen Informationen des Projektes, das heißt jede Information, die nicht eindeutig einem räumlichen Ausschnitt des Gebäudes zugeordnet werden kann. Dazu gehören Kontaktinformationen für Projektpartner, deren Rollenbeschreibung im Projekt, Vereinbarung von Datenaustauschformaten etc. Diese Ebene ist vor allem in der Vorplanung von Bedeutung, während Art und Umfang der Zusammenarbeit ausgehandelt werden.
- Die Planungsebene basiert auf einem meist geometriebasierten Produktmodell. Simulations- und Visualisierungswerkzeuge, letztlich alle direkt bauobjektbezogene, aber von der realen Existenz des Objektes unabhängige Tätigkeiten sind hier angesiedelt. Mit zunehmender Menge an Daten über das Bauwerk gewinnt diese Ebene an Bedeutung und hat damit ihren Schwerpunkt in Phasen der Planung oder Umplanung.
- Die Betriebsebene befaßt sich mit der Fertigung und Nutzung des Gebäudes bis hin zum Abriß. Sie umfaßt damit Aspekte der Baustofftechnologie ebenso wie der Sensorik und Aktuatorik und behandelt natürgemäß den Teil der Daten, der eine reale Entsprechung besitzt oder dessen reale Umsetzung im Entstehen begriffen ist. Diese Daten teilt sie mit der Planungsebene. Ihre größte Bedeutung hat die Betriebsebene in der Nutzungsphase.

Die Ausbildung der Betriebsebene als eigene informationstechnische Ebene in diesem Modell rechtfertigt sich neben der Sicht auch aus der räumlichen Bindung an den Bauplatz. Diese bewirkt eine Trennung von dem Teil eines Gesamtsystems, der mit rein virtuellen Objekten arbeitet und daher zumeist am Arbeitsplatz der datenbearbeitenden Personen angesiedelt ist.

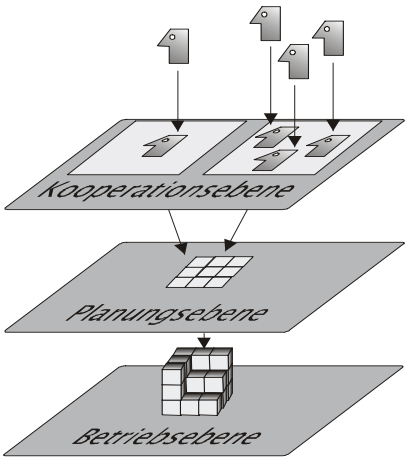

Bild 2.6: Drei-Ebenen-Modell zur Strukturierung von Integrationsumgebungen.

Während die Trennung zwischen Planungs- und Betriebsebene durch die Existenz oder Nichtexistenz von Bauteilen determiniert ist, läßt die Trennung zwischen Kooperations- und Planungsebene einige Freiheiten. Zum Teil entscheidet die Struktur der Zusammenarbeit über die Interaktion zwischen den Beteiligten, zum Teil können diese zwei Ebenen nach frei festlegbaren Präferenzen modelliert werden. Die bisherigen Forschungsarbeiten erwähnen dies meist nicht explizit, beziehen ihre unterschiedlichen Modellierungen aber zu einem wesentlichen Teil aus divergenten Vorstellungen dieser beiden Ebenen.

Für den Allgemeinfall des gesamten Bauwerkslebenszyklusses ist ein Überwinden oder Zusammenführen dieser drei Ebenen wegen der unterschiedlichen Datenerfordernisse in der Regel nicht sinnvoll.

In der aktuellen Forschung des ifib [Kohler 2002] existiert auch eine Benennung, nach der die Kooperationsebene der "Welt der Akteure", die Planungsebene etwa der "Welt der Geometrie" und die Betriebsebene etwa der "Welt der Elemente" entspricht.

2.3.3 Vergleich der Modelle

Das Modell der Integralen Planung strukturiert anhand der anfallenden Daten und hat somit Einfluß auf die Datenmodellierung, während beim Drei Ebenen-Modell die Systemstruktur der angestrebten Integrationslösung Ausgangspunkt der Betrachtung ist. Zwischen beiden Modellen läßt sich eine Zuordnung treffen, die für die folgenden Kapitel nützlich ist: (Bild 2.7: Überführung von Integraler Planung in das Drei-Ebenen-Modell.)

- Die Vertikale Integration der verschiedenen Phasen im Gebäude-Lebenszyklus betrifft vorrangig die Planungsebene. Speziell die Überführung der Gebäudedaten aus der Vorplanung in die Planung und schließlich in die

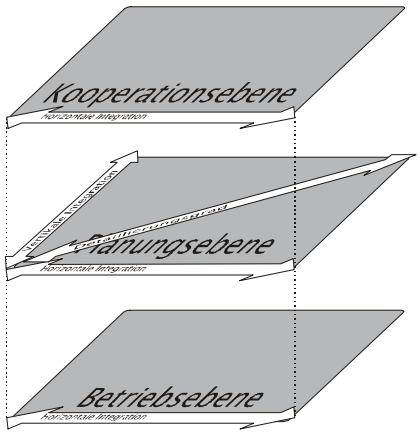

Bild 2.7: Überführung von Integraler Planung in das Drei-Ebenen-Modell.

Hierbei ist die Erweiterung der integralen Planung um den Detaillierungsgrad konsistent mit überführbar.

Betriebsphase sowie die Rückführung der Betriebsdaten in die Planung von Renovierung und Umbau stellen hohe Anforderungen an den Datenaustausch.

- Ebenso vorrangig in der Planungsebene angesiedelt ist die Integration über den Detaillierungsgrad. Das Problem der Überführung der Gebäudedaten etwa von einem abstrakten Raumprogramm in die Ausführungsplanung gewinnt noch an Komplexität, wenn die Konsistenz der Anfangsdaten (hier: des Raumprogramms) über alle Änderungen in der Planung hinweg gewahrt bleiben soll.
- Die Horizontale Integration findet sich in allen Ebenen des Drei-Ebenen-Modells wieder. In der Kooperations- und Planungsebene umfaßt sie vor allem die Koordination der verschiedenen Gewerke während Planung und Bau, während in der Betriebsebene die Koordination verschiedener Nutzerinnen und Nutzer im Vordergrund steht.

Die Planungsebene zeichnet sich demzufolge durch ihre zentrale Rolle zur Integration aus, während die Betriebsebene als Schittstelle zwischen computergestützter Arbeit und Baurealität unabdingbar ist. Interessant ist die Rolle swe Kooperationsebene, die nach diesem Abgleich klar eine Metaebene darstellt, die für den Bauablauf nur mittelbar zielführend ist. In anderen Worten: Würde eine einzige Person alle für ein Bauprojekt nötigen Kompetenzen auf sich vereinen, wäre diese Ebene unstreitig überflüssig. Ob bzw. wann eine separate Darstellung der Kooperation nötig ist, bleibt bisher offen – die Vorteile einer solchen Kooperationsdarstellung werden schließlich mit einer höheren Komplexität der Rechnerunterstützung selbst und deren Anwendung erkauft.

2.4 Schlußfolgerung

Generell läßt sich aus der Formulierung vieler Arbeiten zur Kooperation entnehmen, daß der hohen Komplexität der kollaborativen Bauprozesses mit einer entsprechend komplexen Modellierung der Kollaboration begegnet wird – je komplexer das gewählte Ausgangsszenario, desto komplexer die resultierende Modellierung.

Diesen Ansatz möchte die vorliegende Arbeit hinterfragen. Aufgrund des Ergebnisses des Modellvergleichs und der offenkundigen Akzeptanz von Einzelplatzlösungen ergibt sich die Frage, inwieweit eine Minimalisierung der Metaebene zur Kooperation eine umfassende kooperative Lösung im Bauwesen sogar verbessern kann.´Das hier zu beschreibende Modell beschreitet einen neuen, durchaus extremen Weg, indem es keine explizite Modellierung der Kooperation vorsieht. Der Beantwortung dieser Frage soll in dieser Arbeit nähergekommen werden, indem zugleich ein Gegenentwurf vorgelegt wird, anhand dessen sich die Diskussion leichter fortsetzen läßt, und dieser Gegenentwurf auf Machbarkeit überprüft wird.

3 Bauwerkslebenszyklus-Anforderungen

Aus der Diskussion von zwei Modellen für den Gebäude-Lebenszyklus im vorangegangenen Kapitel lassen sich zwei Problembereiche für eine IT-Unterstützung ableiten:

- Daten: Die Planungs- und Betriebsebene des Drei-Ebenen-Modells basieren beide auf einem Produktmodell, das eine Vielzahl von Anforderungen erfüllen muß. Vor allem die Integration über den Detaillierungsgrad verdeutlicht, daß dieselbe Information in mehreren Ausprägungen vorliegen kann, deren inhaltliche Übereinstimmung kontrolliert werden muß.
- Prozesse: Die Koordinationsebene des Drei-Ebenen-Modells sowie die Horizontale und Vertikale Integration erfordern die Weitergabe von Informationen. Sie beschreiben aus Ihrer jeweiligen Sichtweise die Zusammenarbeit der Beteiligten.

In den folgenden Abschnitten werden für diese beiden Problembereiche Anforderungen für die informationstechnische Unterstützung des Bauwerkslebenszyklusses beschrieben. Die gesammelten Anforderungen werden durch Modellierung und Systemstruktur eines idealen vollständigen Systems zur Unterstützung des Bauwerkslebenszyklusses zu erfüllen sein. Mit Erfahrungen aus anderen Fachgebieten wird diese Anforderungsliste wahrscheinlich erweitert und überarbeitet werden müssen, so daß die interdisziplinäre Ausarbeitung eines Gesamt-Anforderungsprofils für Arbeiten wie die vorliegende nützlich wäre.

3.1 Daten

Jede Informationsverarbeitung im Bauwesen fußt auf der Modellierung der Gebäude- und Vorgangsdaten sowie der zugehörigen Prozesse. Je umfassender der Anwendungsfall, desto umfassender ist die notwendigerweise zugrundeliegende Modellierung, desto mehr (stillschweigend oder explizit vorhandene) strukturelle Vorgaben existieren. Strukturelle Vorgaben aber sind eine Einschränkung der Entwurfsfreiheit und daher in der Regel von den Anwenderinnen und Anwendern unerwünscht. Besonders tritt dies zutage, wenn die Planung mit Hilfe sogenannter “Baukästen” erfolgt, die mit Vorgaben für die verwendbaren Bauteile weitere, offen sichtbare implizite Planungseinschränkungen festlegen.[5]

5. Beispielhaft seien hier die Baukästen MINI, MIDI [Haller 1974] und MAXI für Gebäude unterschiedlicher Größe und Installationsdichte genannt, die von Professor Fritz Haller entwickelt wurden und trotz großen Nutzens für Bauvorgang und Bauherrenschaft keine hohe Verbreitung erzielen konnten.

Ziel der Modellierung ist daher, die Freiheitsgrade etwa in der Planung beim Umgang mit dem Informationssystem so weit wie möglich zu erhalten, dabei jedoch eine Arbeitserleichterung zu bieten.

3.1.1 Datenmodell

Komplexität

Kern der Rechnerunterstützung für den Bauwerks-Lebenszyklus ist die Darstellung des Gebäudes und seiner im Lebenszyklus relevanten Informationen innerhalb eines Rechnersystems, wie sie Abschnitt 2.2.1 ("Produktmodellierung und Datenaustausch", Seite 8) als "Produktmodell" beschreibt. Die Formulierung eines Gebäude-Produktmodells gelingt jedoch bisher nur dann zufriedenstellend, sofern ein entsprechend enger Teilbereich innerhalb der Architektur betrachtet wird. Dies ist etwa bei Bauelementkatalogen der Fall. Die Komplexität (in Anzahl und Umfang) der im Modell abzubildenden Gegenstände ist daher das Kernproblem der Produktmodellierung.

Räumliche Information

Die Planung von Gebäuden verwenden mehrheitlich Grundrisse und Schnitte. In diese Pläne, die einen horizontalen oder vertikalen Schnitt durch das Gebäude wiedergeben, werden bei Bedarf Zusatzinformationen eingetragen, die für bestimmte Gewerke von Bedeutung sind. Die Informationsfülle erfordert, daß für verschiedene Gewerke unterschiedliche Pläne erstellt werden. In solchen Entwurfs-, Genehmigungs- oder Werkplänen können somit alle relevanten Informationen angemessen dargestellt werden, sofern sie räumlich orientiert sind, das heißt bestimmten Bereichen im Gebäude zugeordnet werden können.

Diese Aufteilung der Gesamtaufgabe reduziert die Komplexität. In der Bauplanung ist häufig die Trennung des inspirativen Teils, der typischen Architektenaufgabe der Funktionsdefinition und Gestaltung, vom transpirativen Teil, der Umsetzung funktionaler und gestalterischer Ideen durch Fachingenieure oder bauausführende Gewerke, anzutreffen. Sie führt jedoch zu erhöhtem Kommunikationsaufwand, der nur zu rechtfertigen ist, solange niemand beide Teile beherrscht. Klassischerweiser erbringen Architekt oder Architektin eine Entwurfsleistung und koordinieren die Projektarbeit im Sinne eines Dolmetschers zwischen Bauherrnschaft und den umsetzenden Ingenieurbüros und Bauunternehmen.

Nicht-räumliche Information

Darüberhinaus existieren in jedem Planungsprozeß Informationen, die nicht räumlich orientiert sind. Insbesondere sind dies Zusatzinformationen zu räumlich orientierten Daten (etwa Bauelemente) sowie Informationen bezüglich der Arbeitsprozesse (Teamzusammensetzung, Planungsablauf). Hierzu gehören Kostenaufstellungen, Raumprogramme, konzeptionelle Erläuterungen auf der einen Seite, prozeßrelevante Daten wie Terminpläne, Besprechungsprotokolle auf der anderen Seite. In der Forschungsarbeit des Instituts für Industrielle Bauproduktion sind dies etwa schwerpunktmäßig Daten der Energie- und Stoffflußbetrachtung, die mit einem rechnergestützten Planungssystem verbunden werden kann. Produktmodelle für das Bauwesen verwenden derzeit bereits neben den geometrischen Attributen weitere, je nach Modell untergeordnete oder gleichwertige Attribute zur Beschreibung der Realität. Diese unterschiedlichen Formen von Erweiterungen der dreidimensionalen Modellierung von Gebäudekomponenten ver-

bindet das Bestreben, alle Aspekte des Planungs- und ggf. auch Bau- oder Nutzungsvorgangs zu integrieren.

Beispielhaft sei hier das A4-Modell [Hovestadt 1994] angeführt. Dieses Modell führt die gleichwertige Betrachtung quasi beliebiger Attribute neben den geometrischen ein, um eine Entwurfsentscheidung (d.h. eine Änderung zu einem bestimmten Zeitpunkt) zu speichern. Es wird z.B. im ArchE-Prototyp angewendet, wo bereits komplexe Attribute enthalten sind, die die Postscript-Beschreibung der Bildschirmdarstellung eines Objektes oder auch Programmcode enthalten können. Beziehungen zwischen Objekten sind hier allein über Nachbarschaft und Kollision möglich, die über die Ordnung der Werte für eine Eigenschaft definiert werden. Nachbarschaft bezeichnet aufeinander folgende (sozusagen "aneinander angrenzende") Werte, während Kollision eine Überschneidung von Werten oder Wertebereichen meint. Weitere Beziehungen werden über eigens definierte Bedingungen, sogenannte "*Constraints*" abgewickelt [Sturm 1997]; eine direkte Objekt-Objekt-Kommunikation existiert nicht.

Dynamisierung des Modells

Da bei Entwurf der Modellierung nicht alle Anforderungen zukünftiger Bauaufgaben bekannt sind, ist die Wahrscheinlichkeit groß, daß benötigte Aspekte zunächst nicht modelliert werden. Mehr noch, eine vollständige *a priori*-Modellierung und -Modelloptimierung ist nicht möglich [Beucke 2001]. Daraus ergibt sich die Notwendigkeit, das Produktmodell während der Laufzeit des Rechnersystems anpassen zu können; dies ist auch als "Dynamisierung des Modells" bekannt [Laabs et al. 1997]. Diese wird entweder über generische Basisobjekte oder Verknüpfung von Partialmodellen [Hübler et al. 2001] [Willenbacher 2002] realisiert, durch die das Gesamtmodell nachträglich erweiterbar wird. Auch für den Fall, daß die initialen Daten nicht aus einer Neuplanung, sondern aus der Bauaufnahme vorhandenen Bestandes stammen und somit auf den ersten Blick weniger Veränderungen unterworfen sein könnten, ist diese Dynamisierung essentiell [Donath 2001].

Die Ansätze zur Dynamisierung variieren zwischen einem einzigen sehr komplexen Produktmodell [Schmitt et al. 1996] und der Definiton von "Partialmodellen". entweder mittels einem gemeinsamen Basismodell oder mit geeigneten Transformationen zwischen lokalen Datenmodellen [Rezgui et al. 1996]. Zieht man die unterschiedlichen Sichtweisen auf denselben Planungsgegenstand in Betracht, wird die fehlende Flexibilität eines einzelnen umfassendenden Modells deutlich [Willenbacher 2002]. Alle derartigen Projekte gelten heute als gescheitert.

Persistenz

Selbstredend müssen die so gewonnenen Daten vor Datenverlust geschützt werden. Datenverlust entsteht, wenn Informationen nicht persistent sind. Persistenz bezeichnet die Dauerhaftigkeit der Information über das Ende der Laufzeit einer bestimmten Programminstanz hinaus. Während Daten im Hauptspeicher einer Applikation durch die Speichertechnik in RAM-Bausteinen flüchtig sind, läßt die Auslagerung dieser Daten auf einen Datenträger (wie Festplatte, Band, CD, Diskette) die Information persistent werden. Art und Menge der Daten sowie Merkmale der Bearbeitung entscheiden über geeignete Speicherverfahren und Datenträger. Sehr häufig werden im Bauwesen derzeit relationale Datenbanken eingesetzt, da diese gut beherrschbar sind und somit bei vertretbaren Kosten auch im Mehrbenutzerbetrieb zuverlässig und schnell arbeiten.

Ein geeignetes Modell sollte daher folgende Anforderungen erfüllen:

I. Das Modell muß die Erstellung und Verwaltung räumlich gebundener, aber auch nicht-räumlicher Information anbieten.
II. Das Modell muß dynamisch erweiterbar sein.
III. Das Modell muß die persistente Datenspeicherung erlauben.

3.1.2 Konsistenz und Versionen

Die umfangreichen Datenmodelle und Datenbestände bergen viele Möglichkeiten für fehlerhafte Datenzustände. Je nach verwendetem Modell können diese unter anderem entstehen durch

1. unvollständige Informationen
2. Fehlplanungen, das heißt Fehlleistungen der Beteiligten
3. Fehlende Übereinstimmung zwischen Instanzen unterschiedlicher Datenmodelle, die dieselbe reale Information repräsentieren

Alle derartigen Datenfehler heißen Inkonsistenz. Sie können prinzipiell über Regeln aufgedeckt werden [Türker et al. 1996] [Sturm 1997], sofern das zugrunde liegende Datenmodell ausreichende Informationen beinhaltet. Je nach Anwendungsfall und Erfordernissen an die Antwortzeit des Systems können Konsistenzregeln beim Erzeugen der Daten oder ereignisgesteuert zu einem späteren Zeitpunkt überprüft werden.

Der sehr häufige Fall 1 muß über einen langen Zeitraum der Datenbearbeitung toleriert werden [Sturm 1997]. Er ist aber nur mit vollständigem semantischen Wissen von Fall 2 zu unterscheiden, so daß in praktischen Anwendungen auch dieser Fall toleriert bleibt[6]. Speziell die Aufdeckung menschlicher Fehlleistung ist nur theoretisch vollständig möglich, da ein entsprechendes Regelsystem sehr hohe Komplexität (und damit entsprechende Laufzeit- und Fehleranfälligkeitsprobleme) haben würde. Die an sich wünschenswerte Parallelisierung der Arbeit, bei der verschiedene Personen unabhängig Daten bearbeiten, impliziert, daß dieselben Daten zur selben Zeit in unterschiedlicher und inkompatibler Weise geändert werden können, womit Fall 3 eintritt. Es gibt verschiedene Varianten, dies *a priori* zu verhindern oder *a posteriori* zu behandeln:

- Sperrprotokolle: Wer Daten bearbeitet, sperrt diese zugleich gegen Bearbeitung durch andere Personen. Damit ist das Problem von Vorneherein vermieden.

- Konsistenzsicherung im Rahmen der Transaktionssteuerung: Nach Abschluß der Bearbeitung wird eine Überprüfung auf Kompatibilität mit dem aktuellen Datenbestand vorgenommen. Eine Speicherung ist nur möglich, wenn die Daten “passen”, ansonsten wird die Bearbeitung zurückgewiesen.

6. Umgekehrt bewirkt nicht jede Fehlleistung einen fehlerhaften Datenzustand.

- separate Konsistenzprüfung: Auf Benutzerwunsch oder zeit- bzw. ereignisgesteuert wird eine Prüfung des gesamten Datenbestandes veranlaßt. Inkompatibilitäten werden beanstandet und müssen manuell nachbearbeitet werden.

Sperren

Die Sperrprotokolle erkaufen den Vorteil, das Problem gänzlich zu umgehen, mit dem Nachteil, die erwünschte Parallelität der Arbeit einzuschränken. Sie verhindern Inkompatibilität nur, wenn tatsächlich alle Bauteile gesperrt werden, die in Abhängigkeit von A stehen. Damit verlagert sich das Problem auf die nicht allgemein lösbare Frage der Definition von Datenabhängigkeiten, die sich wiederum nur durch das präventive Sperren eines großen Datenanteils, das heißt mit großer Sperrgranularität umgehen läßt. Weiterhin geht diese Variante davon aus, daß der bearbeitende Mensch einen in sich konsistenten Datenzustand erzeugt, was keineswegs sicher ist. Damit hat diese Variante ein ungünstiges Kosten-Nutzen-Verhältnis.

Konsistenzprüfung

Die Konsistenzprüfung ist ein probates Mittel zur Überprüfung auch großer und komplex abhängiger Datenmengen [Sturm 1997]. Bei der Bearbeitung wird der jeweils aktuelle Stand von Gebäudedaten (beziehungsweise einer Version von Gebäudedaten) von den Beteiligten bewertet und editiert. Während für den gemeinsamen Editiervorgang verschiedene Arten von technischen Hilfsmitteln bereits üblich und in komplexe Systeme integriert sind, erfolgt die Rechnerunterstützung bei der Bewertung und Konsistenzprüfung, soweit vorhanden, in der Regel in Form von Einzelplatz-Lösungen, da die nötige Software meist aufwendig ist und das Interesse an einer Integration entsprechend gering ist.

Für die Bewertung von Gebäudezuständen kommen zum einen Regelbasierte Systeme zum Einsatz, deren Arbeitsweise der Prüfung von *Constraints* ähnelt: Auf Basis der Daten können diskrete Aussagen über den Gebäudezustand zu einem gegebenen Zeitpunkt oder die Gebäudeentwicklung getroffen werden, so daß im Ergebnis üblicherweise qualitative Aussagen stehen. Damit ist, je nach zur Verfügung stehenden Daten, mit jedem Datenmodell grundsätzlich der Einsatz Regelbasierter Systeme möglich.

Für die Umsetzung der Konsistenzprüfung innerhalb der Transaktionssteuerung muß jedoch eine akzeptable Antwortzeit garantiert werden, da die Prüfung im Hintergrund und ohne explizite Auslösung zur Benutzerin oder Benutzer erfolgt. Somit scheiden komplexe Prüfungen aus. Weitere Probleme bereiten hier die Antwortzeiten und Laufzeiten im Netzwerk.

Damit erscheint zunächst der dritte Fall der separaten Konsistenzprüfung am praktikabelsten, allerdings hat dieser den Nachteil, daß prinzipiell beliebig lange ein inkonsistenter Zustand der Gesamtdaten bestehen kann. Dieser kann den bearbeitenden Personen überlassen bleiben, da insbesondere bei grafischer Bearbeitung räumlicher Daten viele Inkonsistenzen auch bei konventioneller Planung relativ schnell aufgedeckt werden. Eine Möglichkeit der Übermittlung von entfernt arbeitenden Beteiligten neu bearbeiteter Daten in die lokale Arbeitsumgebung ist die farbliche Hervorhebung, wie dies etwa als “Redlining”-Funktion in CAD- und Textverarbeitungsprogrammen bereits üblich ist.

Versionen

Speziell Fall 3 kann auch als Versionenproblem verstanden werden, in dem mehrere, zueinander inkonsistente Kopien der Datenbestände zur selben realen Information existieren. Die horizontale Kooperation mehrerer Beteiligter verschärft das Versionenproblem, da zum Teil durch das Erlauben von Parallelbearbeitung, zum Teil durch Laufzeiteffekte innerhalb von Rechnernetzen Versionen entstehen können. Nur unter geeigneten technischen Randbedingungen (etwa der ständigen Verbindung aller Arbeitsstationen im Netzwerk) können Versionen aus Laufzeiteffekten durch Sperrprotokolle vermieden werden. Hieraus und aus dem normalen Vorgang, während der Planungsphase Alternativen abzuwägen, ergibt sich die Notwendigkeit zur Akzeptanz von Versionen in jeder Art kollaborativem System für Ingenieursanwendungen [Amor et al. 1997].

Die Schwierigkeit, in der Gebäudeplanung Versionen zu akzeptieren, liegt im Aufwand der Konsolidierung mehrerer Versionen zu einer gemeinsamen Lösung [Eiermann et al. 1994]. Diese ist bislang nicht automatisierbar, sondern kann nur durch Funktionen wie Redlining oder Differenzbildung unterstützt werden. [Blodau 1999] zeigt jedoch, daß Versionierung und Konsistenz ineinander überführbar sind und damit zur selben Problemklasse der “Kollision” zusammenfallen. Daher ist eine gemeinsame Strategie zur Bearbeitung sinnvoll. Parallel bearbeiteten Daten können eine neue Version eröffnen, wobei die Übersicht und der Wechsel der Versionen an der Benutzungsoberfläche unterstützt werden muß. In Kombination mit einer Konsistenzprüfung nach Fall 2 kann eine Vorauswahl getroffen werden, welche Datenelemente bei zwei gleichzeitigen Bearbeitungen etwa gemeinsam sind und daher nicht nach Versionen unterschieden werden müssen.

Auch hier verschiebt sich jedoch das Problem teilweise nur auf die Konsolidierung der Versionen, die heute wegen der Problemkomplexität üblicherweise nicht oder, wie in [Eiermann et al. 1994] beschrieben, unzureichend gelöst ist. Letztlich bleibt die Konsistenz der Daten daher eine nur durch den Menschen zu lösende und zu überwachende Aufgabe, die zum einen durch entsprechende Überwachungs- und Analysewerkzeuge sowie Editoren unterstützt, zum anderen durch ausreichende Dokumentation der Daten entsprechend erleichtert werden muß. “Multiple Repräsentationen” derselben Daten konsistent zu halten, ist daher auch Gegenstand der Forschung.

Die Anforderungen bezüglich Versionen und Konsistenz sind damit:

IV. Das Modell muß Versionen zulassen. Diese können explizit oder implizit entstehen.
V. Das Modell muß Mechanismen zur Abfrage bieten, die eine Konsistenzprüfung und Konsolidierung ermöglichen.
VI. Um Implementierungsfreiheiten offenzuhalten, müssen prinzipiell zu jedem Zeitpunkt alle Daten zugreifbar sein.

3.1.3 Navigation

Heutige Zeichen- oder Planungssysteme ordnen Objekte in der Regel im dreidimensionalen Raum an. Die naheliegende und folgerichtige Methode zur Navigation in der Datenmenge ist daher zunächst die Bewegung im Raum.

Zeitachse zur Navigation

Betrachtet man die zwei Zeitachsen im A4-Modell [Hovestadt 1994], so ist die eine Zeitachse den Bauteilen, die andere dem Planungsverlauf zugeordnet. Dies hat in der Anwendung immer wieder zu Nachfragen geführt. Um diese Semantik verständlich zu machen, ist daher eine transparente Umsetzung dieser Trennung erforderlich: Der Gegenstand der Gebäudeplanung und die Kooperation im Planungsumfeld werden somit zwei unabhängige, getrennte Teile des Planungssystems. Andere Systeme (etwa Intesol [Kohler et al. 1998]) trennen diese Betrachtungsweisen komplett voneinander, so daß eine Ebene zur Planungskooperation und eine Ebene der Planungsmodellierung entsteht, die sich in das Drei-Ebenen-Modell einfügt.

Wenn auf die Kooperationsebene verzichtet werden kann, entfallen die unterschiedlichen Berachtungsebenen und mit ihnen ein Modellierungs- und Benutzbarkeitsproblem. Trotzdem muß neben der räumlichen Navigation die Navigation über die Zeit vorgesehen werden, wenn zur Berücksichtigung des Lebenszyklusses mehr als nur der Moment der Gebäudefertigstellung geplant werden soll.

Da das Modell dynamisch erweiterbar sein soll, können über die grundlegende Orientierung in Raum und Zeit grundsätzlich weitere Kriterien zur Navigation hinzukommen. Diese sind jedoch auf heutigem Kenntnisstand methodisch als Mechanismen zur Datensuche und -filterung einzuordnen. Sofern grundsätzlich alle Daten abgefragt werden können, ist eine Unterstützung auf Modellebene lediglich eine Frage der Optimierung hinsichtlich Datenzugriffen und Antwortzeit. Als Unterschied gegenüber der Navigation in Raum und Zeit bleibt weiterhin festzuhalten, daß diese zusätzlichen Kriterien neben kontinuierlichen Werten auch beliebige andere (diskrete) Werte enthalten können, etwa eine Herstellerangabe.

Es ergeben sich die folgenden Anforderungen:

VII. Das Modell muß Abfragemechanismen zur Navigation über kontinuierlichen Datenwerten in Raum und Zeit bieten.
VIII. Das Modell sollte Such- und Filtermechanismen für diskrete wie kontinuierliche Datenwerte optimierend unterstützen.

3.1.4 Simulation

Im Bereich der Gebäudequalität rücken neben der Eignung eines Gebäudes für die vorgesehene Nutzung und der Einhaltung des Zeit- und Geldbudgets für Bauvorgänge durch veränderte gesellschaftliche Wahrnehmung und daraus resultierender Gesetze zunehmend ins Blickfeld:

- die Umweltbelastung beim Bauen und beim Baubetrieb

- die Lebensdauer der die eingesetzten Materialien sowie deren funktionale und stoffliche Recyclingfähigkeit

Diese muß für vor der Umsetzung einer Baumaßnahme vorab ermittelt werden, wobei Informationen aus der Planung genauso wie Erfahrungswerte über das Verhalten vorgesehener Bauteile und den Zustand des Bauplatzes sowie ggf. des Baubestandes einfließen können. Simulationen und Berechnungen eignen sich für solche quantitativen Bewertungsprobleme[7], da sie typischerweise kontinuierliche Daten verarbeiten. Da die Einsatzgebiete ineinander übergehen, werden im folgenden beide Begriffe unter einem subsumiert: Eine "Simulation" bestimmt das Zutreffen bestimmter Kriterien (damit die Tauglichkeit einer Annahme) anhand eines mathematischen Modells.

Bereits ohne das Vorliegen von geometrischen Information sind einige interessante Simulationen möglich, so etwa

- Kostenberechnungen
- Abschätzungen zur Umweltbelastung einer Baumaßnahme (Alterung von Bauteilen)

Die Bauteilalterung profitiert für außenliegende Bauteile bereits von geometrischen Informationen über die Lage des Bauteils (etwa Wetterseite).

Sofern Geometrieinformationen vorliegen, kommen vielfältige Anwendungsgebiete hinzu:

- Modellierung des Raum- und Zeitbedarfs beim Bauvorgang: Können bestimmte Bauteile noch in den kompletten Rohbau eingebracht werden oder wäre dafür ein unvollendeter Rohbau nötig (etwa Badewanne durch 60cm-Tür)? Reicht der Platz für den Montagevorgang (etwa großflächige Leichtbauwände an engen Fluren)?
- Im Nutzungskontext veränderliche räumliche Anordnungen. So können etwa bestimmte Vorgänge während der Benutzung simuliert werden, die die Möblierung verändern. Kollisionen in diesem Vorgang können wie Kollisionen in der Planung erkannt werden [Blodau 1999].
- Auch komplexe Steuerungsprozesse mit wechselseitigen Abhängigkeiten sind nun prinzipiell simulierbar, etwa das Anspringen der Heizung, nachdem die Verschattung automatisch aktiviert wurde. In diesen Kontext gehören daher auch Energiesimulationen.

Die Beherrschung der Komplexität realer Erzeugnisse ist hierbei das Kernproblem, das durch die Quantifizierung von Merkmalen (etwa "ökologische Verträglichkeit") angegangen werden muß. Die Erfassung und Bereitstellung von Daten über Baustoffe und Bauteile spielt die Schlüsselrolle in diesem Bereich. Neben (heute meist fehlenden) geeigneten Informationen über Montageprozesse einzelner Teile ist dafür Voraussetzung, daß Daten über Teile des Bestandes gekennzeichnet werden, da sie im Gegensatz zu geplanten Teilen nicht ohne weiteres verändert wer-

7. Aus dem aggregierten Ergebnis der quantitativen Bewertung können selbstverständlich qualitative Aussagen abgeleitet werden – das Ziel einer Simulation ist häufig eine Aussage der Art "geeignet" / "nicht geeignet".

den können - soll ein reales Gebäudeteil verändert werden, verursachen Demontage- und Montagevorgang zeitliche, räumlichen, personelle, finanzielle und ökologische Aufwände.

Es ist leicht zu erkennen, daß solche Simulationswerkzeuge bei stärkerer Integration in den Planungsvorgang zu wichtigen Entscheidungshilfen in komplexen Zusammenhängen werden können, etwa bei der Prüfung von Planungszuständen.

Bei der Simulation wird die Veränderung von Datenwerten über die Zeit vorausberechnet. Die Weiterverarbeitung von derart kontinuierlich veränderbaren Daten erfordert, daß Simulationswerkzeuge Werte in Abhängigkeit vom Zeitpunkt abfragen, um hiermit weitere Berechnungen durchzuführen.

Objektverhalten

Für die folgende Betrachtung wird eine objektorientierte Denkweise benutzt. Da die Objektorientierung gleichmächtig ist wie andere Paradigmen, beschränkt dies die Allgemeinheit der Aussage nicht, vereinfacht aber ihr Verständnis.

In der Simulationstauglichkeit liegt eine wesentliche neue Anforderung gegenüber reinen Planungssystemen: Übliche Planungsentscheidungen sind Veränderungen eines Zustandes in einen anderen Zustand zu einem beliebigen Zeitpunkt. Diese Information kann als kontinuierliche Information betrachtet werden, wenn das Objektverhalten als “stabil bis zur nächsten Änderung” definiert wird. Für kontinuierlich veränderliche Information, wie sie für Simulation benötigt wird, muß ein entsprechendes Verhalten (etwa “Festigkeit nimmt jährlich degressiv um 5% ab”) objektspezifisch definierbar sein. Ein Datenwert hängt somit nicht nur von expliziten Datenänderungen (Editieraktionen) ab, sondern auch vom Objektverhalten selbst.

Nutzerverhalten

Dies bietet zunächst noch keine explizite Unterstützung für eine Einbeziehung des Nutzerverhaltens in die Simulation. Dieser ohnehin sehr komplexe Punkt wird in den meisten Simulationen außer acht gelassen, kann aber das Ergebnis massiv beeinflussen. Im Bereich der Energiesimulation etwa ist von großer Wichtigkeit, ob sich in einem Raum von 30 m² zwei oder fünfzehn Personen befinden (die jeweils eine typische Wärmeabstrahlung von 100 Watt haben). Dem Nutzerverhalten liegt für Simulationen eine eigene Modellbildung zugrunde, die in die jeweiligen Simulationswerkzeuge einfließen muß. Deshalb sollten Personen im Gebäude für Simulationen modellierbar sein. Das Verhalten der Personen fließt dann wie die Eigenschaften der Planungsobjekte in Zustandänderungen über die Zeit ein. Insbesondere ist hierfür erforderlich, daß Personen replaziert werden können, also ihr Wert auf geometrischen Achsen veränderlich und das Verfahren der Werteänderung modellierbar ist. Damit sind etwa Raumauslastungen zu Stoßzeiten simulierbar, indem eine Anzahl von Menschen mit unterschiedlichem Verhalten modelliert und eine Simulation über die Gebäudebelegung durchgeführt wird.

Datenqualität

Mit ausreichenden Daten über Teile und Bauprozesse wäre damit eine automatische Optimierung des Bauprozesses auf bestimmte Zielgrößen hin möglich. Die hierfür notwendige Datenqualität liegt jedoch in der Regel erst zu einem recht späten Zeitpunkt innerhalb der Planung vor, da erst bei hohem Detaillierungsgrad ausreichend exakte Angaben gemacht werden können. Das Simulationsergebnis wäre aber häufig interessant, um die Planungsdetaillierung zielgerichtet voranzu-

treiben. Damit kann auch das Problem der Simulation zu Planungszwecken als *Wicked Problem* eingeordnet werden. Dieses Problem läßt sich mit *Default*-Werten entschärfen [Kohler 1998]. Danach können für Objekte Vorgabewerte angenommen werden, auf deren Ergebnis bereits eingeschränkt Simulationen möglich sind, bis objektspezifische Informationen zur Verfügung stehen. Die Qualität des Simulationsergebnisses variiert dabei mit der Qualität der Ausgangsdaten, so daß eine Einschätzung der Ergebnisqualität nur möglich ist, wenn die Qualität der Ausgangsdaten bekannt ist.

Da für die Unterstützung des gesamten Bauwerkslebenszyklusses die Simulation unabdingbar ist, gelten folgende Anforderungen:

IX. Das Modell muß "Objektverhalten" erlauben, das heißt die nicht-interaktive Zustandänderung während eines beliebigen Zeitraums.
X. Das Modell sollte die Modellierung von Nutzerverhalten ermöglichen, etwa indem Menschen modelliert werden.
XI. Das Modell sollte die Speicherung und Verarbeitung unterschiedlicher Datenqualitäten vorsehen.

3.2 Prozesse

Effizienz

Bauland, Bausubstanz und Bauausführung sind in Deutschland vergleichsweise teuer, so daß gerade hierzulande die Notwendigkeit einer Effizienzsteigerung besteht, um die Kosten (etwa durch Finanzierung von Baumaßnahmen, während das im Umbau befindliche Gebäude nicht nutzbar ist) zu senken.

Der zunehmenden Anteil von Instandhaltungsaufgaben rückt neben den Investitionskosten in der Bauphase die Unterhaltungsaufwendungen und Erneuerungsinvestitionen in das Blickfeld. Mit der so zunehmenden Komplexität des Gebäudes und der Bauaufgabe müssen zur Wahrung der Übersicht eine stärkere Arbeitsteilung und hierzu mehr Abstraktionsebenen eingeführt werden, zumal je nach Gebäude unterschiedliche von Gesetz, Bauherrnschaft oder Materialien / Bauteilen verursachte implizite Planungseinschränkungen existieren. Hierbei ist das Lösen der Prozesse von Bauteilen, das heißt die Absage an feste Vorgehensschritte, zur Effizienzsteigerung von Vorteil.

Organisatorisch betrachtet kommunizieren im Bauprozeß in der Regel wenige Personen auf der Bauherrenseite mit wenigen Personen aus dem Bereich der Architektur oder Projektsteuerung. Letztere koordinieren zugleich den Bauprozeß, so daß wegen der Überschaubarkeit des Teams bereits mit relativ schlecht strukturierten Vorgehensweisen eine erfolgreiche Baudurchführung möglich ist, wenn eine kritische Masse an Gesamterfahrung der Beteiligten erreicht wird. Obwohl dies durch die beschriebene zunehmende Komplexität der Bauaufgabe immer weniger gilt, prägt eine entsprechend wenig organisierte Kooperation den Bauprozeß. Läßt sich die jeweilige Kompetenz der Beteiligten effizienter nutzen, kann die

Qualität des Ergebnisses zugleich mit anfallenden Kosten und erforderlicher Zeit für Planung und Ausführung optimiert werden.

Für den Fall, daß eine komplette Kooperationsmodellierung mit vollständiger Unterstützung von Workflows und Entscheidungsfindung gewünscht ist, muß daher eine Modellierung der Beteiligten, ihrer fachlichen Kompetenzen und Entscheidungsbefugnisse sowie dedizierte Protokolle zur Entscheidungsfindung vorgesehen werden.

Ein Ziel der Projektsteuerung ist ein möglichst hohes Parallelitätsniveau zwischen Planung und Bau, das die Gesamt-Projektdauer bis zur Fertigstellung verkürzt und damit die Baukosten reduziert. Diese Optimierungsaufgabe in bezug auf die Parallelität bietet einen Ansatzpunkt zur Effizienzsteigerung durch rechnergestützte Planungssysteme, da Prozeßoptimierung ein im Produktionsbereich bereits bekanntes Feld der Rechneranwendung ist. Entwicklungspotential für die zeitliche Optimierung der Arbeit liegt offensichtlich darin,

- die Planung zu verteilen und gleichzeitig zu bearbeiten
- die Bauausführung bei noch laufender Detailplanung zu beginnen
- die Totzeiten in Planungs- und Bauablauf zu minimieren, unter anderem durch Beschleunigung der Kommunikation der Beteiligten

Hierfür ist eine Fokussierung des Ablauf auf den Vorgang der Planung und Umsetzung, mithin auf die anstehenden Entscheidungen anstelle der betrachteten Bauteile nötig: Was sind die Zielvorgaben? Zu welchem Bauzeitpunkt müssen welche Entscheidungen getroffen sein? Welche Vorgaben stehen hierfür evtl. zur Disposition? Wer hat welche Kompetenzen? Diese Vorgehensweise ist charakteristisch für die Integrale Planung.

3.2.1 Kompetenzen und Zugriffssteuerung

Kollaboration

Die Zusammenarbeit mehrerer Personen an einer gemeinsamen Aufgabe heißt "Kollaboration". Die Zusammenarbeit mehrere Personen mithilfe einer computerunterstützten Arbeitsumgebung ist unter dem Begriff *Computer Supported Cooperative Work* (CSCW) eingeführt. Softwarelösungen, die die erforderliche Prozeßunterstützung bieten, werden allgemein als *Groupware* bezeichnet. Das Spektrum reicht hierbei von der Unterstützung weitgehend statischer Prozesse in Workflow-Systemen bis zu dynamischen Umgebungen wie Videokonferenz- und Chatsystemen [Bergmann et al. 2000]. Im engeren Sinne firmieren im Wesentlichen Workflowsysteme unter diesem Begriff, von proprietären branchenübergreifenden Standardprodukten bis hin zu Branchen- und Individuallösungen für das Bauwesen [Kohler et al. 1998] [Müller et al. 1998], die zunehmend auf Internettechniken basieren [Bentley et al. 1997]. Da sich die organisatorischen Rahmenbedingungen häufig und ohne direkten Bezug zu rechtlichen Rahmenbedingungen ändern, gibt es Bestrebungen zur Ad-hoc-Planung wie in [Müller 1999], die eine möglichst flexible und rasche Anpassung an geänderte Gegebenheiten erreichen will.

Vielen Systemen, die branchenübergreifend eingesetzt werden sollen und damit ohne implizites Wissen über den Inhalt der Daten auskommen müssen, ist eine dokumentenbasierte Arbeitsweise gemeinsam: Die kleinste vom System unterstützte und semantisch bekannte Informationseinheit ist ein Dokument. Die "Granularität" der Information ist damit im Vergleich zu Systemen, die Informationen in Gebäudebauteile strukturieren, recht grob. Eine kleinere Granularität der Information kann innerhalb der jeweiligen Systemgrenzen programmiert werden, Gegenstand des Datenaustauschs bleibt jedoch das Dokument. Damit unterstützen diese Lösungen weniger Ziel der Bearbeitung eines Gebäudes in seinen Teilen als vielmehr durch den Dokumentenaustausch den Arbeitsprozeß selbst. Daher basieren geeignete Arbeiten für das Bauwesen auf dynamischen Objektstrukturen [Hauschild et al. 2000] oder beinhalten neben der Dokumentenebene zusätzliche Modellierungsebenen [Scherer 2000].

Verwaltung der Prozeßstruktur

Systeme, die den Arbeitsprozeß modellieren, nehmen eine explizite Teammodellierung vor, um die reale Umgebung möglichst umfassend in ein virtuelles Modell zu überführen. Dies schließt neben den Zielobjekten auch die Akteure einer Kollaboration ein und umfaßt in seiner vollständigen Ausprägung folglich eine Modellierung der Kompetenzen der beteiligten Personen. Diese Kompetenzen können zur Suche geeigneter Ansprech- oder Kooperationspartner eingesetzt werden (Skill Management). Dieser Bereich ist als Teil des Wissensmanagements ein eigenständiger Forschungsbereich, der nicht Gegenstand dieser Arbeit ist.

Liegt der Fokus auf der Betrachtung des Arbeitsprozesses selbst, werden Anwendungen zur Festlegung von Kommunikationsabläufen, Teamzusammensetzung, technischen Rahmenbedingungen und Ähnlichem mehr oder minder eigenständig. Hierbei bestehen jedoch zwei Risiken:

- Die dafür eingesetzte Zeit kann von den Benutzerinnen und Benutzern als unproduktiv empfunden werden, da die Arbeit nicht offensichtlich zielführend ist.
- Umgekehrt besteht die Gefahr, Zeit in vermeintlich produktive Konfiguration(soptimierung) zu investieren, da die in sich geschlossene Metaebene bei Vollendung der Arbeit das Erreichen eines Zwischenergebnisses suggeriert.

Beides führt potentiell zu einer verringerten Akzeptanz des Systems. Arbeiten über Kollaboration optimieren so eher die technischen Abwicklung der Kommunikation als die Arbeit auf ein qualitativ hochwertiges Ergebnis hin auszurichten. Es ließe sich spekulieren, ob dies dazu beigetragen hat, daß umfassende Integrationslösungen im Bauwesen auf Basis der Kollaborationsoptimierung bisher nicht marktrelevant sind.

Umgekehrt läßt sich aus der Existenz solcher Arbeiten und Softwarelösungen sowie dem Aufwand für Kommunikationsverwaltung ablesen, daß Wechsel der Kooperationsstruktur wenn auch nicht alltäglich, so doch zumindest kein Ausnahmefall sind. Daher sollte dieser Wechsel mit minimalem manuellem Auf-

wand erfolgen. Ein geeignetes System muß daher die Rekonfiguration zur Laufzeit unterstützen:

XII. Das Modell muß erlauben, daß die Struktur der Beteiligten mit geringem Aufwand rekonfiguriert wird.

3.2.2 Dokumentation und Rücksetzvorgang

Die nötige Zusammenarbeit über Unternehmensgrenzen hinweg fordert ebenso wie die Qualitätsmanagement-Normen DIN-ISO 9000-9004 eine gute Dokumentation, da die korrekte bzw. fehlerhafte Erbringung von Leistungen gegebenenfalls rechtlich bindend belegt werden muß. Dies wird auch praktisch im Bauwesen angewendet und gilt somit auch für Informationssysteme im Bauwesen. Jede Entscheidung im Bauwerkslebenszyklus muß mitprotokolliert werden. Da möglicherweise automatisierte Teile des Systems bestimmte Entscheidungen automatisch treffen dürfen, bezieht sich der Begriff "Entscheidung" im folgenden sowohl auf solchermaßen automatisiert berechnete Ergebnisse als auch auf menschliche Festlegungen. Berechnete Werte müssen auf Rechenverfahren und Annahmen sowie menschliche Festlegungen bis zur Person und Zeitpunkt zurückführbar sein. Die "Rückverfolgbarkeit" setzt daher eine ausreichende Speicherung der Umstände der Entscheidung voraus. Dies könnte in Fortführung der "Planungshistorie" aus dem A4-Modell geschehen [Hovestadt 1994], würde dann aber zu so großen Datenmengen führen, daß Netzlast und Antwortzeiten eines verteilten Systems wahrscheinlich nicht mehr vertretbar wären. Zur Reduktion der Datenmenge kann die Granularität der dokumentierten Entscheidung erhöht werden, indem das Transaktionskonzept zur Zusammenfassung mehrerer Datenänderungen in eine Entscheidung angewendet wird.

Graph der Entscheidungen

Da die Ergebnisse einer Entscheidung als Annahmen in die nächste einfließen, lassen sich aller Entscheidungen als gerichteten Graphen, als "Entscheidungsnetz" darstellen. Theoretisch ist damit jede Entscheidung und jeder Planungszustand durch eine beliebige Anzahl "Entscheidungspfade" bestimmt, die sich bis an den Projektanfang zurückverfolgen lassen.

Für die Ergebnisqualität ist es unabdingbar, daß nach der Änderung einer Entscheidung diejenigen Bereiche überarbeitet werden, für die diese Entscheidung Voraussetzung war. Dieser Fall tritt auch ein, wenn die Kundenanforderungen während des Projektes wechseln oder sich sonstige Rahmenumstände ändern. Daher müssen diese Entscheidungspfade dokumentiert und nachgehalten werden können.

Rücksetzen

In besonderem Maße interessant ist die Vorgehensweise beim Stornieren von Entscheidungen, die von Softwareanwendungen selbsttätig getroffen werden. Für diesen Fall sollte immer die Möglichkeit des Rücksetzens in den Ausgangszustand vorgesehen werden. Falls etwa eine Anwendung Werte A aus Werten B und C ermitteln und diese Ergebnisse weitergeben kann, gelten hierfür die gleichen Gesetzmäßigkeiten wie für durch Menschen getroffene Entscheidungen: Unter welchen Umständen ist dies zulässig und erwünscht? Wie ist dies im konkreten Fall

nachvollziehbar und gegebenenfalls revidierbar? Im folgenden wird das Rückgängigmachen einer (menschlichen wie maschinellen) Entscheidung “Revision” genannt unabhängig davon, ob es ein Rücksetzen in den Ausgangsuzustand oder ein Ersetzen durch einen neuen Zustand ist.

Im Vergleich dazu ist in ArchE [Henckels et al. 1997] ein bereichsweises Rücksetzen des Planungsgegenstands auf einen früheren Zeitpunkt möglich. Das Rücksetzen nur eines Teils der Planung widerspricht jedoch dem dort vertretenen Konzept, Planungsentscheidungen statt Planungsgegenstände zu speichern, da auch die Entscheidung des Rücksetzens als Entscheidung speicherbar sein sollte. Damit ist das bereichsweise Rücksetzen schwer verständlich zu machen. Wie sich zeigte, wirft es zudem schwerwiegende Konsistenzprobleme für vergangene und nun nachträglich revidierte Planungszeitpunkte auf, die wiederum zu komplexen Erklärungsproblemen an der Benutzungsschnittstelle führen.

Automatische oder manuelle Anpassung an geänderte Gegebenheiten können als neue Entscheidungen dokumentiert werden. Dies gilt auch für den Sonderfall der Wiederherstellung eines zu einem früheren Zeitpunnkt bereits existierenden Planungszustandes.

Sowohl das bereichsweise Rücksetzen als auch das Revidieren automatisch getroffener Entscheidungen sind hinsichtlich der Darstellbarkeit problematisch. Alle Rücksetzvorgänge sind hinsichtlich der Überprüfung des neuen Zustandes verwandt zum Problem der Konsistenzüberprüfung nach 3.1.2 ("Konsistenz und Versionen", Seite 24). Es bleiben Anforderungen bezüglich der Dokumentation:

XIII. Das Modell muß relevante Entscheidungen dokumentieren, das heißt speichern und abrufbar halten.
XIV. Das Modell sollte die Abhängigkeiten dieser Entscheidungen mit dokumentieren, um die Rückverfolgbarkeit und Konsistenzsicherung zu erleichtern.

3.2.3 Kooperation und Datenaustausch

Jedes System zur Kooperationsunterstützung zwischen menschlichen Beteiligten erfordert generell Mehrbenutzerfähigkeit. Aufgrund der organisatorischen Struktur der Nutzerinnen und Nutzer im Bauwesen mit einem hohen Anteil kleiner Unternehmen und Freiberufler ist eine zentralistisch-hierarchische Struktur nur in den Fällen naheliegend, wo ein Generalunternehmen die Bedingungen der Zusammenarbeit festschreiben kann. Der Nachteil dezentraler Lösungen ist der hohe Aufwand, um eine gute *Quality of Service* (Dienstverfügbarkeit, Antwortzeit und Bandbreite) im gesamten Netz sicherzustellen. Speziell bei großen Datenmengen, wie sie im Bauwesen anfallen, ist die Antwortzeit des Systems kritisch und von Ort und Art der Datenspeicherung abhängig[8].

Zentrale Datenhaltung

Der Versuch, alle anfallenden Daten in einem gemeinsamen Datenmodell abzulegen, bedingt die Forderung nach maximaler Flexibilität des Datenmodells. Produktmodelle oder der A4-Raum arbeiten mit diesem Ansatz. Unabhängig davon, ob eine zentrale Datenhaltung auf einem Großrechnersystem oder in einer Client-Server-Umgebung stattfindet, sind folgende Eigenschaften der zentralen Datenverwaltung für die weitere Diskussion von besonderer Bedeutung:

- Es gibt eine hierarchische Trennung zwischen Datenserver- und Anwendungssoftware, die diese Daten verwendet.
- Jede Anwendungssoftware hat eine eigene, individuelle Sicht auf die Gesamtdaten. Diese repräsentiert die Sicht des Personenkreises, der diese Applikation benutzt. Da die Zusammensetzung der Arbeitsgruppen in der Regel von Planungsprojekt zu Planungsprojekt wechselt und daher die individuellen Kenntnisse der Beteiligten nicht im Vorhinein bekannt sind, besteht der Benutzungskreis einer Applikation potentiell aus Personen verschiedener Arbeitsgruppen[9].
- Soll eine neue Applikation in das System eingebracht werden, die zusätzliche Daten benötigt, so müssen diese zusätzlich in das Datenmodell aufgenommen werden oder bestehende Daten im Datenmodell müssen entsprechend angepaßt werden. Beides verursacht Aufwand für die Veränderung des Datenmodells, im Falle der Anpassung bestehender Daten müssen die vorhandenen Sichten an das veränderte Datenmodell angepaßt werden; der Aufwand hierfür steigt mit der Anzahl vorhandener Sichten.

Diese oben beschriebene Zerfaserung des Systems in Einzelbausteine führt zu einem System, dessen Struktur nicht auf dem architektonischen Planungsprozeß basiert, sondern in erster Linie informatischen Gesichtspunkten wie dem Finden geeigneter Sichten und einfacher Strukturierung in Module genügt. Dies entspricht nicht dem Ziel dieser Arbeit.

Verteilte Datenhaltung und Transformationen

Die diametral entgegengesetzte Methode der Datenverwaltung ist der Verzicht auf ein gemeinsames Datenmodell zugunsten lokaler Datenmodelle. Dies ist interessant, wenn selbständige Applikationen über vorhandene Schnittstellen zum Datenaustausch verbunden werden sollen. Hierbei werden die Daten von einer in die andere Applikation transformiert; siehe Abschnitt 2.2.1 ("Produktmodellierung und Datenaustausch", Seite 8).

Die objektorientierte Implementierung erfordert zur Strukturierung jedoch eine statische Hierarchie von Objekten, entlang derer Objekte Eigenschaften “erben”.

8. Die Datenübertragungsleistung zwischen kleinen Unternehmen liegt heute immernoch häufig bei dem Maximalwert einer ISDN-Verbindung (64 kBit/s, ohne Protokolloverhead ca. 7 KByte/s Nutzdaten), was keinen Austausch komplexer Planteile während der Bearbeitung zuläßt. Auch als "Business DSL" gehandelte Anbindungen erlauben in Senderichtung mit 256 kBit/s typisch das Vierfache, was von steigenden Datenvolumina aufgezehrt wird. Erst mit SDSL oder andernen symmetrischen Breitbandlösungen ist hier eine substanzielle Besserung zu erwarten.

9. Die Sicht der Applikation ist also der größte gemeinsame Ausschnitt der Benutzenden. Hieran wird deutlich, daß in einer integrierten Planungsumgebung mehr Applikationen und damit mehr einzelne Sichten entstehen werden, als es Arbeitsgruppen gibt. Dieser Effekt ist etwa beim ArchE-Projekt bereits zu beobachten gewesen.

Sie hat damit eine systeminhärente Schwäche bezüglich der Dynamisierung des Modells: Wenn Eigenschaften und Verhalten von Objekten zur Laufzeit änderbar und erweiterbar sein sollen, ist diese statische Hierarchie kontraproduktiv, sobald die Änderungen und Erweiterung der bisherigen Vererbungshierarchie zuwider laufen. Die objektorientierte Implementierung setzt damit der Dynamisierung des Modells Grenzen.

Föderierte Datenhaltung

Die Föderierte Datenhaltung [Türker et al. 1996] modelliert von vorneherein lokale Datenmodelle und ihre Transformation für die Nutzung auch innerhalb einer Applikation. Sie verteilt die Daten so, daß sie idealerweise vom Ort der nächsten Bearbeitung mit geringem Transaktionsaufwand erreichbar sind. Dies bedingt, das entweder alle denkbaren Transformationen implementiert werden müssen oder Datentransfer und damit Planung nur auf eng eingegrenzten, vorgegebenen Pfaden möglich ist. Der erste Fall erzeugt hohen Programmieraufwand und damit große Fehleranfälligkeit, der zweite Fall eine unerwünschte Einschränkung der Planungsfreiheit. Das Zufügen neuer Applikationen ist in der Föderation einfach, da das Erstellen geeigneter Möglichkeiten der Transformation zur Einbringung in das Gesamtsystem reicht (Bild 3.1: Hinzufügen eines Datenmodells in eine Föderation durch Hinzufügen von Transformationen.), allerdings zieht jede Änderung eines einzelnen Datenmodells Aufwand zur Korrektur aller zugehörigen Transformationen nach sich. Die Verwaltung der zu einem Datenmodell zugehörigen Transformationen ist komplex und führt zu dem Wunsch, möglichst wenig Transformationen zu benötigen und pflegen zu müssen. Transformationen können außerdem zu Informationsverlusten führen, die ebenso manuell entdeckt und korrigiert werden müssen.

Diese Lösung biete sich also an, wenn eine klare räumliche Strukturvorgabe den Ablauf des Planungsprozesses bestimmt. Dies ist etwa im Schiffsbau der Fall, wo

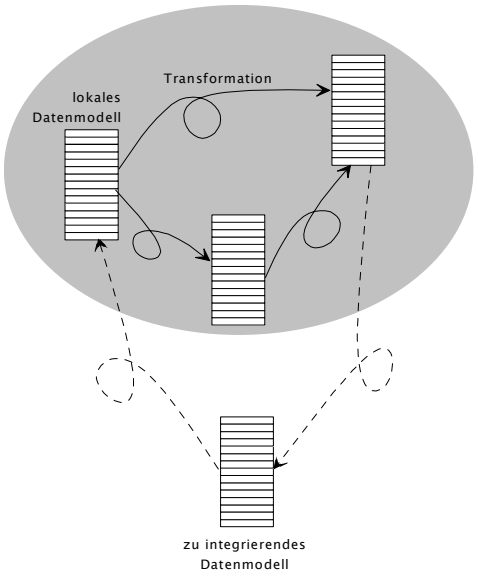

Bild 3.1: Hinzufügen eines Datenmodells in eine Föderation durch Hinzufügen von Transformationen.

die Struktur der Werft (Kalkulations- und Konstruktionsbüro, Dock, Materialwirtschaft) und enge Zeitvorgaben den Ablauf bestimmen.

Mischformen

Zwischen einer zentralen und einer föderativen Datenhaltung sind Mischformen denkbar, in denen gewisse Applikationen ein Datenmodell teilen und die Daten zu anderen Applikationen tranformiert werden, wie dies auch bei der Systemintegration verschiedener Anwendungen entsteht. Diese Lösungen erzeugen Kopien von Informationen. Um Inkonsistenzen und hieraus resultierende Bearbeitungsfehler zu vermeiden, müssen diese Kopien als Duplikate auch dann identifiziert werden können, wenn ihr Wert geändert wurde. Eine Werteänderung muß auf alle Systeme, die Kopien vorhalten, propagiert werden. Dies entspräche im Fall eines gemeinsamen Datenmodells einer Replikation.

Die Betrachtung der verschiedenen existierenden und möglichen Strukturen ergibt folgende Anforderungen:

XV. Das Modell muß eine effiziente Kommunikation theoretisch beliebiger Partner an beliebigen Orten erlauben.
XVI. Das Modell muß sicherstellen, daß der Datenaustausch nicht zu wesentlichen Informationsverlusten führt.

4 Modellierung von Fehldaten

Dieses Kapitel entwirft auf Basis der extrahierten Anforderungen ein Daten- und Kommunikationsmodell. In diesem Kapitel wird hierzu ein Protokoll definiert, dem die Kommunikation zwischen Systemteilen (Modulen) folgt.

4.1 Modellierungsziel

Neben den Anforderungen gehen in dieses Modell praktische Erfahrungen mit den Forschungsprototypen des Instituts für Industrielle Bauproduktion sowie mit Groupware und branchenspezifischen Anwendungen ein. Folgende für das Modell prägenden Einflüsse ergaben sich aus diesen Erfahrungen:

- der in Abschnitt 2.4 ("Schlußfolgerung", Seite 20) genannten Entschluß, keine Kooperationsebene vorzusehen.
- das hieraus abgeleitete Nebenziel, den Aufwand für Konfiguration des Systems möglichst gering zu halten.
- die gleichwertige Behandlung menschlicher, teilautomatisierter und Vollautomatisierter Einflüsse auf den Planungsverlauf.
- die Entscheidung für ein dezentral organisiertes System.

Alle diese Einflüsse dienen dem Ziel, ein möglichst flexibles, mit möglichst wenig technischen Vorkenntnissen während der Arbeit leicht an neue Erfordernisse anzupassendes System zu erschaffen. Sie sind als Entscheidung diskutierbar und letztlich subjektiv getroffen.

Einschränkung der Modellierung

Aus dem in Kapitel 3 ("Bauwerkslebenszyklus-Anforderungen", Seite 21) eingangs genannten Bedarf an Erweiterung und Revision wird deutlich, dass die Gesamtaufgabe zu komplex für eine Einzelleistung ist. Die vorliegende Arbeit setzt daher den Schwerpunkt dort, wo strukturell Neues erwartet wird: Im Bereich der Prozeßabbildung und der (dafür notwendigen) passenden Datenmodellierung. Nicht vollständig umgesetzt werden daher die Anforderungen der Abschnitte 3.1.2 ("Konsistenz und Versionen", Seite 24), 3.1.3 ("Navigation", Seite 27) und 3.1.4 ("Simulation", Seite 27):

IV. Das Modell muß Versionen zulassen. Diese können explizit oder implizit entstehen.
V. Das Modell muß Mechanismen zur Abfrage bieten, die eine Konsistenzprüfung und Konsolidierung ermöglichen.
VI. Um Implementierungsfreiheiten offenzuhalten, müssen prinzipiell zu jedem Zeitpunkt alle Daten zugreifbar sein.

VII. Das Modell muß Abfragemechanismen zur Navigation über kontinuierlichen Datenwerten in Raum und Zeit bieten.
VIII. Das Modell sollte Such- und Filtermechanismen für diskrete wie kontinuierliche Datenwerte optimierend unterstützen.
IX. Das Modell muß "Objektverhalten" erlauben, das heißt die nicht-interaktive Zustandänderung während eines beliebigen Zeitraums.
X. Das Modell sollte die Modellierung von Nutzerverhalten ermöglichen, etwa indem Menschen modelliert werden.
XI. Das Modell sollte die Speicherung und Verarbeitung unterschiedlicher Datenqualitäten vorsehen.

Dennoch darf die resultierende Modellierung diese Anforderungen nicht konterkarieren, sondern muß Möglichkeiten zur Lösungsintegration bieten. Daher verdienen die Anforderungen VI. und XI. besondere Aufmerksamkeit:

- Die vollständige, generische Datenverfügbarkeit nach Anforderung VI. erlaubt in Verbindung mit der Erweiterbarkeit aus Anforderung II. mit hoher Wahrscheinlichkeit eine adäquate Umsetzung der Daten-Anforderungen IV., V. und VII. bis X.
- Datenqualitäten nach Anforderung XI. greifen so tief in die Daten- und Prozeßmodellierung ein, wie dies sonst nur Versionsunterstützung nach Anforderung IV. tut, die jedoch weit besser untersucht ist.

Beide Anforderungen werden den Kanon aufgenommen, die in dieser Arbeit umzusetzen sind. Damit bleibt die Beschränkung signifikant bezüglich der sofortigen praktischen Nutzung des Modells, schmälert aber nicht die Möglichkeit zur Einschätzung des Kommunikations- und Datenaustauschverfahrens als Hauptziel dieser Arbeit. Insgesamt ergibt sich folgender Liste der umzusetzenden Anforderungen:

I. Das Modell muß die Erstellung und Verwaltung räumlich gebundener, aber auch nicht-räumlicher Information anbieten.
II. Das Modell muß dynamisch erweiterbar sein.
III. Das Modell muß die persistente Datenspeicherung erlauben.
VI. Um Implementierungsfreiheiten offenzuhalten, müssen prinzipiell zu jedem Zeitpunkt alle Daten zugreifbar sein.
XI. Das Modell sollte die Speicherung und Verarbeitung unterschiedlicher Datenqualitäten vorsehen.
XII. Das Modell muß erlauben, daß die Struktur der Beteiligten mit geringem Aufwand rekonfiguriert wird.
XIII. Das Modell muß relevante Entscheidungen dokumentieren, das heißt speichern und abrufbar halten.
XIV. Das Modell sollte die Abhängigkeiten dieser Entscheidungen mit dokumentieren, um die Rückverfolgbarkeit und Konsistenzsicherung zu erleichtern.
XV. Das Modell muß eine effiziente Kommunikation theoretisch beliebiger Partner an beliebigen Orten erlauben.
XVI. Das Modell muß sicherstellen, daß der Datenaustausch nicht zu wesentlichen Informationsverlusten führt.

4.2 Systemstruktur

Die Beherrschbarkeit, sowohl hinsichtlich der Verarbeitungslogik als auch der Performanz mit realistischen Datenmengen, entscheidet über den realen Nutzen eines Informationssystems. Dabei ist das Szenario von Freiberuflern und Kleinunternehmen zu beachten, die projektweise zusammenarbeiten und hierzu Computer einsetzen, die per Einwahlverbindung miteinander vernetzt werden können[10].

Dieser Abschnitt widmet sich daher zunächst der Systemstruktur. Der innerhalb der Informatik gebräuchliche Terminus der "Systemarchitektur" wird im folgenden vermieden und durch "Systemstruktur" ersetzt, um Verwechslungen mit der Architektur als Anwendungsgebiet dieser Arbeit zu vermeiden [Rechtin et al. 1997].

Mit dieser Zielgruppe der Kleinunternehmen und Freiberufler scheidet eine beliebige Erhöhung der Hardwareanforderungen als Lösung von Performanzengpässen aus. Auch beim Datenaustausch sind Verbindungen mit 64 bis 128 kBit/s (idealerweise ca. 7 bis 14 KByte pro Sekunde oder ca. 1 bis 2,5 Minuten pro Megabyte), in Senderichtung üblich, was für den Austausch kompletter Daten eines Projektes bei weitem nicht ausreicht.

Damit wird es erforderlich, Daten im Vorgriff auf zukünftige Bearbeitung lokal bereitzuhalten, das heißt in jedem beteiligten Unternehmen wird es mindestens einen Rechner für die Datenhaltung geben müssen. Dies erlaubt folgende Varianten in der Systemstruktur:

- Mehrere inhaltsgleiche Server mit Replikation: Dies ist die einfachste Variante, da keine Vorhersage bezüglich der benötigten Daten notwendig ist. Gleichzeitig entsteht ein Problem durch Synchronisation von zeitgleich an verschiedenen Orten offline bearbeiteten Daten.
- Ein zentraler Server, ausschließlich online-Bearbeitung: Dies setzt die Einigung auf einen Standort für den Server voraus. Da dies zugleich Kenntnisse in Serveradministration und größeren Datenbanksystemen voraussetzt, ist dieser Ansatz nur bei größeren Unternehmen zum Beispiel im Rahmen von Generalunternehmer-Verträgen einsetzbar. Zugleich entsteht ein Problem durch die ständig verfügbare Verbindung zum Server.
- Verteiltes System: Alle beteiligten Rechner fungieren als Datenserver für die Teile der Daten, die lokal nötig sind, und tauschen fallweise Daten aus. Soweit ist dieses das Idealmodell, da es vielfältig auf die konkreten Bedürfnisse hin zugeschnitten werden kann. Allerdings setzt dies eine hervorragende Kenntnis der Abläufe im System voraus, um die richtigen Datenaustauschaktionen durchzuführen. Auch schließt dies die gleichzeitige offline-Bearbeitung der Daten nicht zwangsläufig aus.

Die ersten beiden Ansätze haben gravierende Nachteile, die aus der mangelnden Berücksichtigung des Kontextes der Datenbearbeitung resultieren. Dennoch ist

10. Dies schließt Szenarien mit Standleitungsverbindung und größeren Unternehmen keineswegs aus. Umgekehrt ist wegen der geringeren Investiotionsmöglichkeiten kleinerer Firmen ein auf größere Unternehmen zugeschnittenes Verfahren jedoch nicht ohne Weiteres übertragbar.

die Lösung eines zentralen Servers interessant, wenn ein Unternehmen die Projektsteuerung übernimmt und hierfür die Infrastruktur stellt. Umgekehrt können die Risiken des dritten, verteilten Ansatzes durch eine geeignete Definition des Datenaustausch-Modells beschränkt werden, wenn dieses Modell von vorneherein auf verteilte Systemstrukturen hin definiert wird. Daher setzt das Fehldaten-Modell auf diese dritte, verteilte Struktur, die durch seine Hierarchiefreiheit immanent sicherstellt, daß alle Daten prinzipiell allen Modulen zur Verfügung stehen, wie es Anforderung VI. vorsieht.

Aus der verteilten Systemstruktur ergibt sich ein weiterer Vorteil: Da es nicht möglich scheint, alle denkbaren Anforderungen an die verwalteten Daten in einem einzigen Datenmodell zu erfüllen und dabei wenig Redundanz (im Sinne von Daten, die für eine spezielle Sicht irrelevant sind) zu erzeugen, müssen für Aufgabenbereiche mit stark unterschiedlichen Datenanforderungen unterschiedliche Datenmodelle und zugehörige Transformationen geschaffen werden. Diese lassen sich auf die ohnehin nötigen Datenaustausch-Aktivitäten abbilden, so daß sich der Mehraufwand durch Transformationen rein auf die Datenkonvertierung beschränkt. Dies ist die Grundlage für die Erfüllung der Anforderung XV.

4.3 Fehldaten-Begriff

Das erstellte Modell wird neu einführen, daß die vom Gesamtsystem aller Beteiligten benötigte, aber noch nicht vorhandene Information ebenso wie die vorhandene modelliert und explizit repräsentiert wird. Damit läßt sich die weitere Arbeit so steuern, daß zum einen Blockaden vermieden werden und zum anderen die benötigte Information gezielt angefordert werden kann.

Die Erweiterung um die Übermittlung von Information über fehlende Daten legt nahe, zur Unterscheidung von Modellen zur Handhabung ausschließlich vorhandener Daten den Begriff “Fehldaten-Modell” zu wählen.

Über die Konnotation des Kunstwortes “Fehldaten” lieferte eine (nicht repräsentative) Recherche Aufschluß, deren Ergebnisse im Detail in [Henckels 1999] enthalten sind. Hiernach ist der Begriff selten eingesetzt und im Bereich der Datenmodellierung nicht verwendet. Die Interpretation des Begriffes ist jeweils eine der folgenden:

- Fehldaten im Sinne von fehlerhaften Daten.
- Fehldaten im Sinne von Daten über Fehler.
- Fehldaten im Sinne von Daten über Fehlzeiten oder fehlende Leistungen, zum Beispiel von Personen.

Der letztgenannten Interpretation liegt der Gedankengang zugrunde, der auch zur Auswahl des Begriffes “Fehldaten” für diese Arbeit geführt hat: Der Terminus bezeichnet einen Bedarf an einer Leistung (hier: Information), die grundsätzlich er-

bringbar ist, aber zum gegebenen Zeitpunkt in der geforderten Weise (noch) nicht zur Verfügung steht.

Quantitiv läßt sich diese kurze Recherche nicht sinnvoll auswerten, da der Begriff "Fehldaten" insgesamt selten benutzt wird. Ein geeignetes englischsprachiges Pendant für "Fehldatenmodellierung" wäre *"Data Deficiency Modelling"*.

4.4 Datenhaltung und -austausch

4.4.1 Datenverteilung

Die Reduktion der Volumen in Datenhaltung und Datenaustausch ist ein wichtiges Ziel der Modellierung für die genannte Zielgruppe. Hierzu seien Informationen in drei Kategorien unterschieden:

- Nutzdaten: Informationen, die im aktuellen Arbeitskontext als Ergebnis einer Anfrage, als Resultat einer Planung etc. direkt von Benutzerin oder Benutzer gewünscht sind.
- Hilfsdaten: Informationen, die indirekt, zum Beispiel als Zwischenergebnis einer Berechnung, zur Erstellung der Nutzdaten unbedingt erforderlich sind. Hierfür spielt die Anzahl der Berechnungsschritte (“Entfernung” der Hilfsdaten von den zugehörigen Nutzdaten) keine Rolle.
- Restdaten: Informationen, die im aktuellen Arbeitskontext nicht verwendet werden. Sie werden in der Regel in anderen Arbeitskontexten als Nutz- oder Hilfsdaten von Interesse sein.

Die Minimierung der Komplexität bei voller Funktionalität erfordert, daß Nutzdaten und Hilfsdaten verfügbar sind, während Restdaten ignoriert werden. Dafür müssen Module bekanntgeben, welche Daten für das jeweilige Modul Nutz- und Hilfsdaten darstellen. Damit können diese gezielt an der Ort der nächsten Verarbeitung transferiert werden, während auf den Transfer von Restdaten verzichtet werden kann. Erst wenn ein neu aktiviertes Modul Daten aus dem Restdatenkontingent verwendet, werden diese transferiert (und nach obiger Definition zu Nutz- oder Hilfsdaten). Unabhängig von Anzahl und Art laufender Module sind im Fehldatensystem insgesamt immer auch alle Restdaten präsent, jedoch ruhen diese bis zur Weiterverarbeitung am Ort ihrer Erzeugung.

Dieses Verfahren trägt der Forderung nach geringer erforderlicher Bandbreite Rechnung und kann als automatisiertes Pull-Verfahren klassifiziert werden, da die Zielanwendung den Umfang des Datenaustauschs bestimmt. Gleichzeitig ist es die Reinform einer verteilten Datenhaltung, die einerseits den unterschiedlichen Interessenslagen bei vielen selbständigen Beteiligten bezüglich des Datenschutzes Rechnung trägt, andererseits hinsichtlich Datensicherheit und Verfügbarkeit

Nachteile gegenüber verhältnismäßig leicht zu pflegenden Servern hat. Da diese Struktur der Arbeitsweise des Fehldatenmodells entspricht, soll sie trotz dieser Nachteile beibehalten werden. Gegebenenfalls kann ein Modul zur persistenten Datenhaltung an zentraler Stelle wie in Abschnitt 5.5.3 ("Systemmodul Datenspeicherung", Seite 80) beschrieben diese Nachteile ausgleichen, ohne das Fehldaten-Paradigma zu durchbrechen.

4.4.2 Austausch einzelner Attributsdaten

Die erforderliche Datenmenge wird wesentlich von der Granulariät der Daten, der kleinsten betrachteten Informationseinheit, beeinflußt. Setzt man bei der Datenmodellierung auf das objektorentierte Paradigma auf, ist die naheliegendste Informationseinheit das Objekt. Die Objektorientierung ist der gedankliche Ausgangspunkt der Fehldatenmodellierung, weil sie die natürlichste Abbildung der Realität erlaubt. Sie eignet sich für die Darstellung realer Gegenstände, wie sie als Bauteile in Gebäuden angeordnet werden, aber ebenso für die Modellierung abstrakter Information. Auf diese Weise kann auch die Modellierung von Nutzerinnen und Nutzern erfolgen, indem einzelne Personen oder Personengruppen als Objekte modelliert werden.

Aufgrund der Arbeitsteilung bei kooperativer Arbeit im Bauwesen ist jedoch das Objekt als Informationseinheit ungeeignet: Jede(r) Beteilige benötigt zur Erfüllung der persönlichen Aufgaben bestimmte Aspekte über alle Objekte: Zur Summierung der Baukosten sind zu allen Bauteilen und Arbeitsvorgängen und immateriellen Aspekten wie Vorfinanzierungen die jeweiligen Kosten interessant, während zur Bauzeitplanung die Lieferzeiten derselben Bauteile und die Dauer derselben Arbeitsvorgänge benötigt werden. Diese Aspekte werden in der objektorienten Modellierung durch Attribute von Objekten repräsentiert. Entsprechend wird die Fehldatenmodellierung als Granularität der Daten nicht das Objekt, sondern das Attribut wählen (Bild 4.1: Datengranularität).

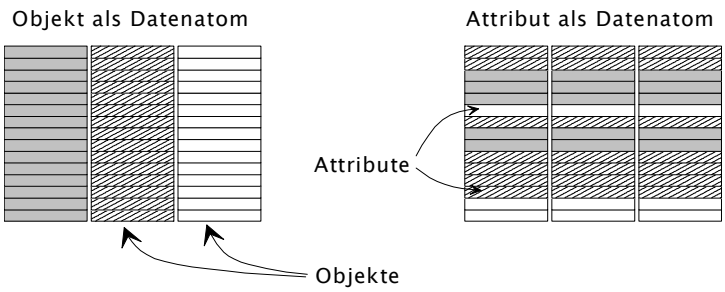

Bild 4.1: Datengranularität

Datengranularität auf Basis von Objekten und Attributen sowie Verteilung der Daten auf drei Module (hier weiß, grau und schraffiert).

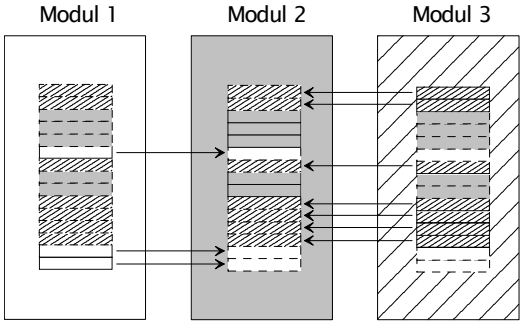

Bild 4.2: Verteilte Modularchitektur

Module 1 und 3 als Server für Attribut-Stubs in Modul 2.

Läßt man diese feine Granularität im Datenaustausch verschiedener Anwendungen zu, so wird nicht in jeder Anwendung jedes Objekt vollständig repräsentiert sein. Während üblicherweise verteilte Systeme entweder eine disjunkte Verteilung kompletter Objekte vorsehen, kann durch die feinere Granularität der Fall eintreten, daß ein Objekt auf keinem Rechner vollständig repräsentiert ist. Dies läuft scheinbar der objektorientierten Konzeption zuwider, alle Informationen gemeinsam zu kapseln. Bei näherer Betrachtung läßt sich dies jedoch als Stub-/Server-Konzept mit verteilter Datenhaltung betrachten, so daß jede Repräsentation eines Objektes partiell Server (das heißt lokal vorhanden) und partiell Stub ist (das heißt, Datenwerte werden von einer anderen Repräsentation bezogen; Bild 4.2: Verteilte Modularchitektur). Damit ist ein klassische Konzept der Datenverteilung zunächst lediglich auf die Ebene der Attribute eines Objektes übertragen. Somit ist auch die recht intuitive Datenmodellierung nach dem Paradigma der Objektorientierung möglich, die auch die Anforderung I.[11] erfüllt.

4.4.3 Modelldynamik

Ein wesentlicher Teil des Kommunikationsaufwandes nicht nur in Bauprojekten ist durch die Anforderung fehlender Information verursacht. Die fehlende Information kann hierbei eine unbekannte Menge von Objekten (etwa ein räumlich begrenzter Planungsteil) oder aber ein oder mehrere Attribute einer bereits vorhandenen Menge von Objekten (etwa Bauteilinformationen) sein. Üblicherweise nimmt dies in Verbünden zur Kooperation (mit oder ohne Computerunterstützung) grob folgenden Verlauf (Bild 4.3: Datenanforderung und Erfüllung):

1. Personen P1 und P2 bearbeiten den Datenbestand D innerhalb ihrer jeweiligen Sicht. A1 entdeckt nun, daß das Fehlen des Datenelements D1 die Forsetzung der Arbeit auf dem aktuellen "Bearbeitungspfad" stoppt.
2. P1 ermittelt durch Wissen oder Nachschlagen die Zuständigkeit von P2 für das Erstellen von D1, kontaktiert diese Person und beschreibt die Notwendigkeit von D1.

11. und Anforderung X., die in dieser Arbeit nicht weiter behandelt wird

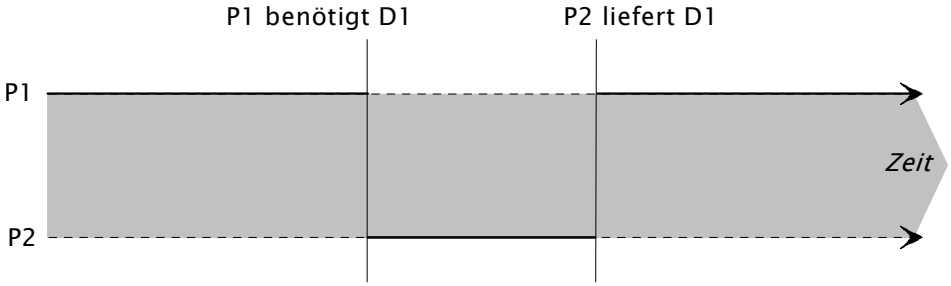

Bild 4.3: Datenanforderung und Erfüllung

Zwei Personen P1 und P2 bearbeiten die Datenmenge kooperativ. Für die Zeitspanne der unerfüllten Datenanforderung liegt der Bearbeitungsbedarf bei P2.

3. P1 wechselt den Bearbeitungspfad, um weiterarbeiten zu können, oder pausiert.

4. P2 liefert D1. P1 setzt die Arbeit am gestoppten Bearbeitungpfad fort.

Einerseits erfordert die wenig formalisierte Anforderung einen vergleichsweise hohen Zeitaufwand, selbst wenn mittels Kenntnis der Person P2, unformellen Arbeitsgepflogenheiten und elektronischer Unterstützung optimale Bedingungen vorherrschen. Andererseits hemmt das Stoppen des Bearbeitungspfades massiv die gewünschte Parallelität der Bearbeitung; diesen Stop zu vermeiden erfordert sehr gute Arbeitsorganisation und Flexibilität auf seiten der Beteiligten.

automatische Datenanforderung

Gesucht ist also ein Verfahren, das fehlende Daten automatisch ermittelt und anfordert. Dies würde den manuellen Kommunikationsaufwand für Datenanforderung idealerweise auf null reduzieren und so Anforderung XII. erfüllen.

Der Datentransfer auf Attributebene erlaubt die gewünschte gezielte Anforderung von Informationen. Im Falle von fehlenden Attributwerten für vorhandene Objekte wird diese Datenanforderung im System explizit bekanntgegeben und kann so wie Daten selbst gespeichert werden. Die Datenbearbeitung muß dabei nicht angehalten werden, sondern kann an anderer Stelle weiter erfolgen, bis das System in der Lage ist, nach Vorliegen der angeforderten Daten die Bearbeitung fortzusetzen. Dieses hohe Maß an Nebenläufigkeit ist nur praktikabel, wenn durch das Nachhalten der Anfragen keine übermäßiger Verwaltungsaufwand für den Menschen entsteht. Ist dies erfüllt, bildet die Modellierung von Datenanfragen einen wichtigen Baustein zur effizienten Kommunikation nach Anforderung XV.

Sofern es sich bei der fehlenden Information um die eingangs erwähnte unbekannte Objektmenge handelt, wurzelt der Bedarf an den angefragten Daten in einem der folgenden Fälle:

- im Fehlen notwendiger Attributwerten vorhandener Objekte, die die Schnittstelle zu den unbekannten Objekten bilden. In diesem Fall können diese Attributwerte angefordert werden.
- im Bedarf an abstrahierter Information über die unbekannte Objektmenge. In diesem Fall existiert entweder ein Objekt, das die entsprechende Ab-

straktion beschreibt, oder ein solches wird zum Zweck der Informationsanforderung angelegt. Dessen Attribute werden dann angefordert.

Modelldynamik

In der beschriebenen Form erlaubt der Anforderungsmechanismus sowohl unbekannte Objekte als auch unbekannte Attribute in Objekten. Da auf eine statische Objekthierarchie bisher verzichtet werden kann, ist insbesondere die vollständige Kenntnis aller vorhandenen Attribute eines Objektes nicht notwendig. Dies eröffnet die Möglichkeit, ein Objekt zur Laufzeit strukturell zu erweitern. Dies bewirkt eine dynamische Erweiterbarkeit des Datenschemas hinsichtlich der Objektstruktur, wie sie in Anforderung II. beschrieben ist.

Mit der Informationsübermittlung auf Ebene von Attributen als kleinster Einheit können nun ohne Kenntnis einer statischen oder dynamischen Objekthierarchie alle gewünschten Informationen auch über bislang unbekannte Objekte ausgetauscht werden, ohne daß eine vollständige Aktualisierung des Gesamtdatenbestandes nötig ist. Attribute werden dabei über ein eindeutiges Merkmal identifiziert, das im folgenden "Domäne" genannt wird. Die Domäneninformation umfaßt den Datentyp und die Semantik eines Attributes. Da ein Modul alle für seine Bearbeitungslogik relevanten Attribute per Definition kennt, kann es alle anderen (auch zur Laufzeit neu hinzukommende) Attribute ignorieren. Damit ist das Modell des Fehldatensystems als Ganzes erweiterbar, indem neue Module weitere Attribute (separat oder in Kombination mit bestehenden Attributen) verarbeiten.

Ohne Beeinträchtigung der Allgemeinheit wird im folgenden davon ausgegangen, daß die Domäne eines Attributes dem Attributnamen entspricht. Diese Lösung ist bei begrenztem Modellumfang besonders praktikabel, da dieser entsprechend assioziativ gewählt werden kann. Bei komplexen Modellen ist dies wegen der Möglichkeit der Fehldeutungen von Attributnamen oder der Notwendigkeit zur Umbenennung bei Modellerweiterung nicht optimal.

Notwendigkeit eines Datenwörterbuchs

Immer existiert implizit oder explizit ein "Datenwörterbuch" der Domänen. Die Informationen dieses Datenwörterbuchs sind bei der Definition und Implementierung eines Moduls erforderlich, um Domänen korrekt und eindeutig festzulegen. Wenn diese Informationen zur Laufzeit nicht vorhanden sind, wird im folgenden von einem impliziten Datenwörterbuch gesprochen, andernfalls von einem expliziten. In beiden Fällen ermöglicht die eindeutige Semantik des Attributnamens den korrekten Datenaustausch und erfüllt somit Anforderung XVI.

Semantik von Objekten

Die alleinige Betrachtung der Attributsemantik ist jedoch nicht ausreichend. Semantisch identische Angaben können in unterschiedlichen Objekten unterschiedlich genutzt werden. So können etwa "Kosten", die einem Währungsbetrag entsprechen, pro Stück oder pro Mengeneinheit gemeint sein. Dies ließe sich grundsätzlich über die Definition weiterer Domänen realisieren, würde dann aber deren Anzahl stark aufblähen und die Verständlichkeit gleichzeitig senken. Die bessere Alternative ist daher, bei der Verarbeitung von Attributen den "Objekttyp" zu berücksichtigen, der in üblichen Programmiersprachen durch die Klasse des Objektes festgelegt ist. Wenn wegen der dynamischen Strukturveränderlichkeit auf die Zuordnung von Objekten zu den im objektorientierten Paradigma üblichen Klassen verzichtet werden muß, ist ein Grundprinzip der Objektorientierung der Dynamik zum Opfer gefallen. Damit fehlt die zur Verarbeitung notwendige Typin-

formationen, denn ein eindeutig gewählter Objektbezeichner allein läßt im Allgemeinfall keinen Rückschluß auf den Objekttyp zu. In diesem Fall notwendig ist somit eine explizite Identifikation eines Objekttyps als Ersatz für die Klassenzuordnung, um die korrekte Verarbeitung unbekannter Objekte zu ermöglichen.

Obwohl technisch damit die Basis der Objektorientierung fast vollständig verlassen wurde, ist das objektorientierte Paragdigma weiterhin die beste Metapher für die Arbeit mit den Daten des Fehldatenmodells, da Objekte als virtuelle Repräsentanten realer Elemente im Baukontext intuitiv einsetzbar sind sowie die Zuordnung von Attributwerten zu Objekten und von Objekten zu Objekttypen bzw. Klassen dem klassischen objektorientierten Denken entspricht.

4.5 Modellierung verteilter Kommunikation

4.5.1 Zuordnung des Verhaltens

Zu Objekten gehören neben Daten auch die verhaltensbestimmenden Methoden. Der attributbasierte Datenhaltungsansatz wirft durch seine Zerlegung eines Objektes in eigenständig transferierbare Attribute die Frage auf, an welcher Stelle diese Verhaltensinformationen gespeichert werden.

Allgemein sei das Verhalten eine Menge von Regeln, die aus (Eingabe-)Attributwerten jeweils einen (Ausgabe-)Attributwert ermittelt, im mathematischen Sinne also eine Abbildung von endlichen vielen Eingabevariablen auf eine Ausgabevariable. Die Einschränkung auf eine Ausgabevariable beeinträchtigt die Allgemeinheit nicht, da eine Abbildung mit zwei Ausgabevariablen problemlos in zwei Abbildungen mit je einer Ausgabevariable zerlegt werden kann, die jeweils eine Teilmenge der ursprünglichen Eingabevariablenmenge verwendet (Bild 4.4: Regeln mit Ein- und Ausgabeattributen). Solche Verhaltensregeln erlauben trotz der Datengranularität auf Attributebene die Anwendung des Objektparadigmas, da die Summe der Regeln, die Attribute eines Objektes als Ausgabeattribut verwenden, das Objektverhalten bilden. Für den Fall einer automatisch (das heißt nicht-interaktiv) abzuarbeitenden Regel erfüllt dies insbesondere Anforderung IX.

Attributzuordnung der Verhaltensregeln

Existieren nun durch die vorgesehene verteilte Datenhaltung sich überschneidende Attribut-Teilmengen als Repräsentationen eines Objektes, so existieren drei Möglichkeiten der Zuordnung der Verhaltensregeln zu Attributen:

1. Zuordnung zu allen Attributen: Dabei würde das gesamte Objektverhalten allen Teilinstanzen des Objektes mitgegeben.
2. Zuordnung zu den Eingabeattributen: Jedes Attribut kann Eingabewert mehrerer Regeln sein, jede Regel kann mehrere Eingabewerte verwenden.

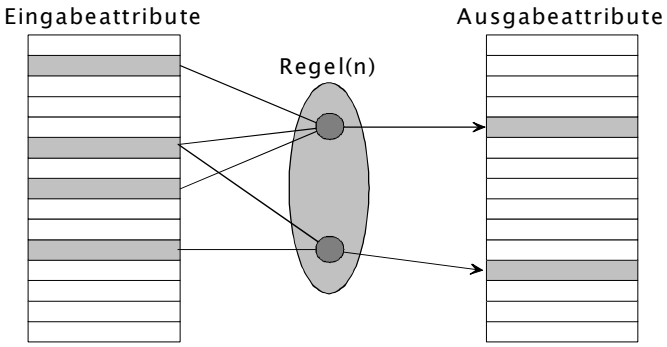

Bild 4.4: Regeln mit Ein- und Ausgabeattributen

Ableitung zweier Attributwerte aus Eingabewerten mittels einer Regel. Diese kann als zwei Regeln mit je einem Ausgabeattribut dargestellt werden.

3. Zuordnung zum Ausgabeattribut: Attribute sind meist eindeutig bestimmbar, allerdings möglicherweise auf verschiedenen Wegen.

Der erste Fall erzeugt potentiell ein hohes Maß an Redundanz, da Regeln auch dann in einer Teilinstanz vorliegen, wenn keines der Ein- oder Ausgabeattribute lokal vorliegt, mithin die Regel lokal nicht verarbeitet werden kann. Auch der zweite Fall erzeugt Redundanz, da jede Regel mehrere Eingabeattribute haben darf. Auch im dritten Fall ist die Zuweisung mehrerer Regeln zum selben Attribut nicht ausgeschlossen, falls mehrere Regeln die Ermittlung desselben Ausgabeattributs zum Ziel haben. Solche "konkurrierenden Regeln" können zweckmäßig sein, etwa um – im Vorgriff auf Abschnitt 4.5.4 ("Datenqualität", Seite 53) – die Wertermittlung für geringeren Stufen der Datenqualität von weniger Eingabeattributen abhängig zu machen.

Jede der skizzierten Zuordnungsmöglichkeiten zu den Daten hat spezifische Nachteile, jedoch ist die Zuordnung zum Ausgabeattribut sinnvoll, da sie wenig Redundanz erzeugt und intuitiv verständlich ist.

4.5.2 Modulstruktur

Nachdem Daten und Verhalten im angestrebten System verteilt sein werden, scheidet eine monolithische Applikation zur Verarbeitung der Objekte als unzweckmäßig aus; vielmehr sollte sich die Verteilung der Verarbeitungsfunktionen an der Verteilung der Datenobjekte orientieren. Damit ist die Modulstruktur Teil der Erfüllung von Anforderung XV.

Die Möglichkeit, verschiedene Attribute desselben Objektes in verschiedenen Applikationen vorzuhalten und zu bearbeiten, erzwingt die im vorangegangenen Abschnitt vorgenommene Zuordnung des Verhaltens zu Attributen (anstelle des Gesamtobjektes) und damit die Aufteilung und Verteilung des Objektverhaltens[12]. Die Implementierung des Objektverhaltens wird hiernach jeweils dort erfolgen, wo das entsprechende Ausgabeattribut ermittelt wird. Da selten eine Applikation ge-

nau ein Ausgabeattribut bearbeiten wird, entsteht eine Bündelung nach funktionalen Aspekten: Inhaltlich ähnliches bzw. zusammenhängendes Attributverhalten wird gebündelt. Diese nach gebündelten Verarbeitungsfunktionen erstellten Applikationen seien Module genannt.

Funktionale Abgrenzung der Module

Es bleibt zu entscheiden, an welchen Kriterien sich die funktionale Abgrenzung der Module gegeneinander orientiert. Software wächst häufig auf Basis einer zunächst begrenzten Anwendung durch Erweiterung der Funktionalität. Dies mag für Einzelapplikationen fallweise sinnvoll sein. Ein Modul, dessen Funktionalität erweitert werden soll, läßt sich jedoch frühzeitig in mehrere teilen, ohne die gesamte Systemstruktur zu durchbrechen. Unter Aspekten der Softwarepflege und des Einarbeitungsaufwandes für Benutzerinnen und Benutzer sollen im vorliegenden System die Module durch die inhaltliche Abgrenzung der Tätigkeiten bei konventioneller Aufgabenverteilung bestimmt sein. Jede Sicht auf inhaltlich zusammenhängende Datenuntermengen erhält so ein eigenes, funktional vergleichsweise kleines Modul. Ein beteiligter Mensch, der bestimmte Rollen im Projekt übernimmt, sollte somit ein oder mehrere Module benutzen. Zwei Menschen mit disjunkten Rollenmengen sollten kein Modul gemeinsam benötigen[13].

Interaktivität

Module lassen sich nach dem Anteil der Benutzerinteraktion unterscheiden. Module, die etwa Daten oder Kommunikationsvorgänge mitschreiben und so Datensicherungen oder Auswertung zur Fehleranalyse oder Dokumention erzeugen, sind nicht-interaktiv. Ebenso sind nicht-interaktive Simulationsmodule sinnvoll. Assistierende Funktionen werden in teilinteraktive Module münden, während allein von Menschen erstellte Daten in vollinteraktiven Editiermodulen erfaßt werden. Teil- und vollinteraktive Module benötigen also eine Benutzungsschnittstelle für die Datenpflege selbst, nichtinteraktive Module hingegen nur für die Administration.

Die skizzierte Modulstruktur ermöglicht speziell bei funktional kleinen Modulen leicht den Austausch verschiedener Module, etwa um im Forschungsbetrieb verschiedene Stufen der Automatisierung eines Vorgangs gegeneinander zu bewerten.

4.5.3 Automatische Konfiguration

Da es durch die Dynamik des Modells nicht nötig ist, den strukturellen Maximalumfang der Daten a priori zu kennen, müssen auch nicht alle Datenerzeuger im Vorhinein bekannt sein. Damit können zur Laufzeit Module dergestalt zu- und abgeschaltet werden, daß nur eine unaufwendige Bekanntgabe der Nutz- und Hilfs-

12. Dies durchbricht das Paradigma der Objektorientierung, Daten und Verhalten zu kapseln (also als einen geschlossenen Block, die sogenannte Klasse, zu betrachten), sofern ein Planungsobjekt oder Bauteil einem Objekt der Implementierung entsprechen soll. Für eine voll objektorientierte Implementierung wäre danach ein einzelnes Attribut und sein Verhalten ein Objekt. Dennoch bleibt diese Arbeit aus Gründen der Anschaulichkeit beim Objektbegriff in der bisherigen Verwendung.

13. Diese Aussage muß eingeschränkt werden, da im Augenblick nur von "Anwendungsmodulen" die Rede ist. Das Implementierungskapitel beschreibt darüber hinaus "Systemmodule", die unabhängig von Rollen eingesetzt werden.

daten nach der Definition in Abschnitt 4.4.1 ("Datenverteilung", Seite 43) für das jeweilige Modul erfolgen muß, um kollaborieren zu können. Diese Daten werden zum Zeitpunkt der Programmierung eines Moduls festgelegt, da zu diesem Zeitpunkt das Objektverhalten, das dieses Modul verarbeitet, definiert wird.

Um administrative Vorgänge für den Benutzer oder die Benutzerin auf das Minimum zu beschränken, sollte das Fehldatensystem sich so weit wie möglich automatisch konfigurieren. Dies entspricht der Anforderung XII. Hierzu ist es insbesondere nötig, daß Softwarekomponenten nach dem Start möglichst automatisch mit bereits laufenden Softwarekomponenten kommunizieren. Dies vermeidet sowohl Aufwand als auch Fehlerquellen manueller Konfiguration. Unter der Annahme, daß die Kompetenz einer Person und ihre Rolle im Planungs- und Bauprozeß mit der Funktionalität der von ihr verwendeten Module deckungsgleich ist, kann die Metaebene zur Teammodellierung so völlig entfallen[14]:

- Datenstrukur und -bedarf werden im Modul codiert.
- Die Kompetenzen und Interessen einer Person ergibt sich aus der Auswahl an Modulen, die diese Person nutzt

Es bleibt das Problem, daß bei einer nicht explizit geregelten Zusammenarbeit Beteiligte und mit ihnen Applikationen, die deren Bearbeitungsschritte in das System überführen, sich zu beliebigen Zeitpunkten zum System addieren oder aus ihm entfernen. Diese Beliebigkeit in der Verfügbarkeit von Komponenten, aber auch die möglichen (und zumindest bei mit Wählleitungen verbundenen Systemen nicht seltenen) Hardwareausfälle, die das System begrenzt tolerieren sollte, erfordern eine Absicherung gegen Datenverlust und Blockade der Weiterarbeit.

Anfrage von Informationen

Die beliebige Verfügbarkeit oder Nichtverfügbarkeit von Komponenten kann zu Blockaden der Arbeit führen, wenn für alle von Menschen oder Modulen vorgesehen nächsten Arbeitsschritten Basisinformationen zur Verarbeitung fehlen. Diese können zum Teil überwunden werden, wenn nicht nur “fertige” Informationen, sondern auch die Anfrage von Informationen in einer Datenbank abgelegt wird. Jedes Modul kennt damit nur die Attribute, die es lesen (auswerten) und die es schreiben (erzeugen) kann. Sind Informationen zu Objekten gefordert, die Modul M1 wegen nicht vollzählig lokal vorhandener Eingabeattribute nicht erstellen kann, werden die fehlenden Eingabeattribute-Werte als Datenanforderung gespeichert, um den sie erzeugenden (und nicht näher bekannten) Modulen M2, M3,... diesen Informationsbedarf mitzuteilen. Jedes Modul muß daher die Datenanfragen bezüglich seiner Ausgabeattribute regelmäßig durchsuchen und (ggf. interaktiv) Informationen gewinnen, um die Attribute mit Daten zu belegen. Ein passendes Modul M2 wird also irgendwann die Daten erzeugen, so daß M1 anschließend mit erweiterten Daten weiterarbeiten kann (Bild 4.5: Datenanfragen und Modullaufzeit).

Vorgabewerte

Mit dieser Modellierung von Informationsanforderungen ist jedoch eine Blockade der Beteiligten nicht auszuschließen:

14. Gleichwohl ist es möglich, das Fehldatensystem um eine solche Kooperationsebene zu erweitern, da Kooperations- und Planungsebene unabhängig voneinander betrachtet werden können. Hierzu wären die Informationen der Kooperationsebene in Objekten zu codieren und in geeigneten Modulen zu bearbeiten.

- Wenn ein Modul, das ausschließliche Quelle bestimmter benötigter Informationen ist, entweder über einen sehr langen Zeitraum nicht in das System eingeschaltet wird, können keine Informationsanfragen gestellt werden.
- Wenn ein ebensolches Modul existiert, aber Anfragen nicht bearbeiten kann (weil diese manuelle Eingriffe erfordern, aber die Benutzerin oder der Benutzer die entsprechenden Informationen nicht liefert bzw. liefern kann), können Informationsanfragen nicht bearbeitet werden.

In beiden Fällen stockt die weitere Bearbeitung im Modul, das die Datenanfrage ausgelöst hat. Hierin liegt eine systeminhärente Grenze, da das skizzierte, sehr unbürokratische Verfahren zum Einbinden von Modulen und zur Dynamisierung des Datenmodells keine Antwortgarantie erlaubt. Dies entspricht jedoch den Stokkungen, wie sie im konventionellen Ablauf ebenso entstehen können; aus diesem Grund sind die Parallelen zwischen Bild 4.3: Datenanforderung und Erfüllung und Bild 4.5: Datenanfragen und Modullaufzeit in der Konzeption begründet. Dieser Punkt wird im Kapitel 6 ("Validierung", Seite 83) wieder aufgegriffen.

Mit solchen Datenanfragen ist es insbesondere möglich, auf vorgegebene Ziele hin zu planen. Wenn die Planerin beispielsweise in einem sehr frühen Stadium ein Gesamtbudget festlegt und die tatsächlichen Gesamtkosten abfragt, kann das von ihr verwendete Kostenmodul alle Informationen anfordern, die es zur Kostenbestimmung braucht, so daß die anderen Module ihrerseits entsprechende Daten anfordern werden, die letztlich in der Aufforderung zur Eingabe relevanter Informationen an andere Beteiligte umgesetzt werden. Die ständig aktuelle Datenaggregation erfolgt dann allein aufgrund der Regeln zur Ermittlung von Ausgabewerten, hier der Projekt-Gesamtkosten. Damit wird die Planungskontrolle direkt in den Planungsvorgang einbezogen, statt auf eine mehr oder weniger fertige Planung als Revision aufzusetzen. Dies kann Fehlerkosten verringern, indem Fehler früher auffallen.

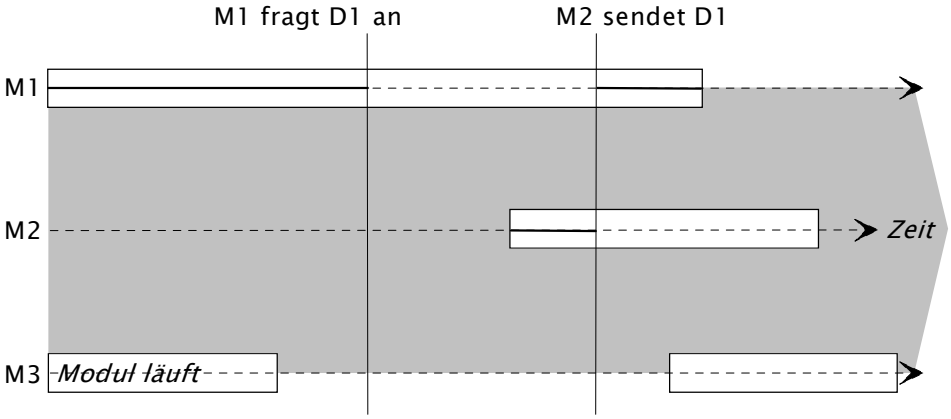

Bild 4.5: Datenanfragen und Modullaufzeit

Datenanfragen werden gesendet, sobald ein diese Daten erzeugendes Modul zur Verfügung steht. Angefragte Datenwerte werden gesendet, sobald diese erzeugt wurden.

4.5.4 Datenqualität

Der Mangel, daß sich Stockungen in der Bearbeitung systemimmanent nicht ausschließen lassen, läßt sich mit dem Einsatz von Vorgabewerten [Kohler 1998] lindern: Stünden Verfahren zur Ermittlung von Vorgabewerten für jedes Attribut in jedem Modul zur Verfügung, so könnten alle Berechnungen ohne Zeitverzögerung angenähert werden. Dies ist ein Beitrag zur Erfüllung der Anforderung XV., vor allem aber die direkte Umsetzung der Anforderung XI. Sofern eine genaue Ermittlung eines Datenwertes nicht möglich ist, wird daher ein vermuteter Wert bereitgestellt, der zunächst als Grundlage der Berechnung dient. Genauere Wertermittlungen ersetzen diesen Vorgabewert, sobald sie zur Verfügung stehen. Dies erhöht die Parallelität maßgeblich.

Dieses Verfahren suggeriert zunächst eine zweistufige Hierarchie aus Vorgabewerten und genauen Werten. Es sollte jedoch auf eine Hierarchie von mehreren Qualitätsstufen erweitert werden, um etwa individuell am Bau gemesse Werte in der Hierarchie oberhalb von im System berechneten und als genau angenommenen Werten zu plazieren, wobei diese Stufe nicht zwangsläufig erreicht werden muß, wenn die berechneten Werte exakt genug eingehalten werden können, wie dies etwa bei CNC-Fertigung der Fall ist. Die Auswahl dieser Qualitätsstufen ist zunächst frei, wenn auch eine zu große Anzahl an Qualitätsstufen nicht sinnvoll erscheint. Folgende Hierarchien von Werten wären, je nach Einsatzgesichtspunkt, denkbar:

- vorgegebener < abgeleiteter < vorgeschlagener < definitiver Wert. Diese Wertehierarchie ist am am Entscheidungsfindungsprozeß orientiert.
- Vorgabe < Mutmaßung < Schätzung < Überschlagsrechnung < Berechnung < Messung. Diese recht feine Unterteilung erlaubt die Unterscheidung und Bewertung vieler Verfahren zu Datenermittlung.
- Vorgabe < Schätzung < Berechnung < Messung. Diese verkürzte Wertehierarchie sollte für die ersten Einsätze des Fehldatensystems ausreichen.

Für Berechnungen muß nun gelten: Werden Datenwerte aus verschiedenen Eingabewerten berechnet, so ist die Qualitätsstufe des Ergebnisses gleich der Stufe des qualitativ niedrigstwertigen Eingabewertes, höchstens jedoch die Stufe für berechnete Werte.

Datenanfragen können nun gezielt Daten einer höheren Qualitätsstufe anfordern. Hierbei wird die minimal benötigte Qualitätsstufe angegeben. Eine Beantwortung erfolgt nur, wenn ein Modul minimal das geforderte Qualitätsniveau erfüllt. Die Anforderung unter Angabe einer minimalen Qualitätsstufe kann somit beim Einsatz einer Hierarchie, die wie die beiden oben angegebenen manuell eingegebene oder überprüfte Werte auf oberster Stufe vorsieht, auch zur manuellen Überprüfung berechneter Ergebnisse dienen: Eine Berechnung wird etwa durch die Anforderung einer höheren Qualitätsstufe in Frage gestellt, wobei es der bearbeitenden Person obliegt, die Methode der Überprüfung zu wählen oder den Wert aus Kenntnis der Sachlage heraus zu bestätigen.

Konfliktlösung

Durch die freie Kombination von Modulen kann der Fall eintreten, daß mehrere Module Werte zum gleichen Attribut liefern, wobei diese auf gleicher oder unter-

schiedliche Qualitätsstufe ermittelt sein können. Hierfür ist folgende Konfliktlösungsstrategie naheliegend:

- Die zuerst verfügbaren Daten werden auf jeden Fall angenommen.
- Die später verfügbaren Daten werden bei niedrigerer Qualitätsstufe ignoriert, bei gleicher oder hörerer Stufe überschreiben sie den bisherigen Wert.

Im Falle gleicher Datenqualität ist das Überschreiben deshalb nötig, um Werteänderungen, wie sie etwa durch Änderungen im Planungsverlauf entstehen, durch das System zu propagieren. Sollte im Verlauf der Anwendung der Fall eintreten, daß "schlechtere" Daten dabei "bessere" überschreiben, die zum Beispiel durch ein anderes Modul erzeugt wurden, ist dies ein Indiz für den Bedarf einer zusätzlichen Qualitätsstufe.

Simulation in Frühphasen

Da in einem solchen Modell zu jedem Planungszeitpunkt idealerweise ein quasi komplettes Gebäude (zum Teil aus Vorgabewerten) zur Verfügung steht, kann zu jedem Zeitpunkt eine Simulation verschiedener Aspekte zur Bewertung der Planung hinsichtlich Kosten, Energie, Dauerhaftigkeit etc. vorgenommen werden. Weiterhin ist jederzeit eine Nutzungssimulation möglich, die auch die Ansteuerung von MSR-Systemen (Gebäudesteuerung) und Facility Management-Komponenten erlaubt. Damit ist die Grundlage für eine qualifiziertere Bewertung des Gebäudes in frühen Planungsphasen einerseits und für einen fließenden Übergang aus der Vorplanung in die Nutzung und Umnutzungsplanung andererseits gelegt. Zu beachten ist natürlich, daß die Aussagekraft der Simulationen (je nach Verfahren) massiv nachlassen kann, wenn zuviele Datenabweichungen durch Schätzungen und Annahmen in die Simulation eingehen.

4.5.5 Abbildung fehlender Information

Nicht zwangsläufig müssen oder können zu allen benötigten Daten Vorgabewerte existieren - plakativ betrachtet hieße dies, daß jede Planung auf Basis eines Standardhauses beginnen und durch Veränderung individualisiert würde. Dieses (durchaus interessante) Konzept kann im Einzelfall jedoch hinderlich sein.

Daher muß der Fall komplett fehlender Daten im System erlaubt sein. Dieser ist in das Konzept der Qualitätsstufen intuitiv einzubetten, indem "Fehldaten" als unterste Qualitätsstufe vorgesehen wird. Jede Datenanforderung fordert daher mindestens die nächsthöhere Qualitätsstufe an. Daher ist weder bei Wertübermittlung noch bei Datenanfragen diese Qualitätsstufe sinnvoll, kann aber bei der Beantwortung einer Anfrage fehlende (bzw. nicht kurzfristig zu ermittelnde) Daten ausweisen.

Für die prototypische Implementierung ergibt sich damit insgesamt folgende Skala von Qualitätsstufen:

Fehldaten < Vorgabe < Schätzung < Berechnung < Messung

4.6 Datenprotokoll

Dieser Abschnitt faßt die Modellierungsbetrachtungen in einem sehr einfachen und unidirektionalen Protokoll zusammen, das von jeder Implementierung eines Moduls im Fehldatenmodell zur Kommunikation verwendet wird. Dieses ist im ISO-OSI-Schichtenmodell auf der Anwendungsebene angesiedelt und berücksichtigt daher nicht Session- und Transportaspekte, die im folgenden Kapitel behandelt werden.

Für den Datenaustausch reichen zwei Primitve, mit denen bekannte und benötigte Informationen übermittelt werden. Diese werden in Abschnitt 5.2 ("Technische Protokolle", Seite 59) auf die technischen Ebenen erweitert.

DatenWert (Objekt, Attribut, Wert, Qualität)

Die Parameter dieses Primitivs identifizieren zunächst mit Objekt und Attribut eindeutig, welcher Datenwert übermittelt wird. Zusätzlich zum Wert selbst wird die Qualitätsstufe identifiziert, die dieser Wert hat.

Bei paralleler Arbeit kann theoretisch der Fall eintreten, daß zwei Module unterschiedliche Werte desselben Attributs in gleicher Qualitätsstufe gleichzeitig erzeugen und an das jeweils andere Modul senden. Mit dem Primitiv "DatenWert" allein läßt sich dies nur auflösen, wenn über das absendende Modul eine eindeutige Priorisierung der konkurrierende Werte möglich ist und die resultierende Nichtakzeptanz auf der Benutzungsschnittstelle umgesetzt werden kann. Um die Komplexität des Protokolls nicht aufzublähen, bleibt dieser Sonderfall unberücksichtigt.

DatenAnfrage (Objekt, Attribut, Qualität)

Dieses Primitiv bestimmt mit Objekt und Attribut, welcher Wert benötigt wird. Die Qualität bestimmt die mindestens erforderliche Qualitätsstufe.

Bei der Übermittlung von Anfragen kann theoretisch eine Deadlock-Situation entstehen, wenn Attribut A1 aus Attribut A2 (und ggf. anderen) berechnet wird, A2 aus A3 usw. und An wiederum A1 benötigt. Da die Regeln und ihre Abhängigkeiten zur Entwurfszeit festgelegt werden, müssen diese potentiellen Deadlocks zur Entwurfszeit verhindert werden. Hierzu ist eine entsprechende Dokumentation der Attributabhängigkeiten, vorzugsweise in Verbindung mit dem Datenwörterbuch, nötig.

4.7 Erfüllung der Anforderungen

Folgende am Anfang des Kapitels aufgelisteten Anforderungen sind mit der skizzierten Fehldaten-Modellierung erfüllbar:

I. *Das Modell muß die Erstellung und Verwaltung räumlich gebundener, aber auch nicht-räumlicher Information anbieten.* Trotz der gewählten feineren Datengranularität orientiert sich die Modellierung am objektorientierten Paradigma, das die Abbildung realer und abstrakter Objekte erlaubt.

II. *Das Modell muß dynamisch erweiterbar sein.* Die Wahl der Datengranularität auf Attributebene ermöglicht die dynamische Erweiterbarkeit.

VI. *Um Implementierungsfreiheiten offenzuhalten, müssen prinzipiell zu jedem Zeitpunkt alle Daten zugreifbar sein.* Dies wird durch die verteilte Systemstruktur unterstützt.

XI. *Das Modell sollte die Speicherung und Verarbeitung unterschiedlicher Datenqualitäten vorsehen.* Dies wird mit einem mehrstufigen Datenqualitätssystem direkt umgesetzt.

XII. *Das Modell muß erlauben, daß die Struktur der Beteiligten mit geringem Aufwand rekonfiguriert wird.* Modelldynamik und Entfall der expliziten Kooperationsmodellierung sind die Voraussetzungen für die Erfüllung dieser Anforderung.

XV. *Das Modell muß eine effiziente Kommunikation theoretisch beliebiger Partner an beliebigen Orten erlauben.* Dies ist die umfangreichste Anforderung. Ihre Erfüllung benötigt die gewählte verteilte Datenspeicherung mit passender Modulstruktur und die explizite Anfrage von Fehldaten. Auch die Definition verschiedener Datenqualitäten ist hilfreich.

XVI. *Das Modell muß sicherstellen, daß der Datenaustausch nicht zu wesentlichen Informationsverlusten führt.* Das zum Zeitpunkt der Modulspezifikation nötige Datenwörterbuch definiert die Attributsemantik und damit die Austauschspezifika.

Da die Fehldaten-Modellierung im Wesentlichen eine Festlegung zum Datenaustausch ist, sind die Anforderung III. bezüglich der persistenten Speicherung sowie die Anforderungen XIII. und XIV. bezüglich Entscheidungsdokumentation nicht allein durch die Modelldefinition erfüllbar. Sie erfordern die Spezifikation geeigneter Module.

5 Implementierung

Da das Fehldatensystem keine Metaebene zur Kooperationsunterstützung enthalten soll, muß die Implementierung (in der Reihenfolge der Prioritäten)

1. durch eine entsprechend angepaßte Systemstruktur möglichst wenige Verwaltungsinformationen benötigen,
2. die notwendigen Informationen automatisch oder assistiert generieren, und
3. im Falle notwendiger manueller Eingriffe die Informationen im Arbeitszusammenhang bearbeiten und diesen Zusammenhang erkennen lassen.

5.1 Mehrschichtenorganisation

Umfangreiche Software wird üblicherweise mehrschichtig aufgebaut. Diese Schichtenorganisation erlaubt insbesondere, das Datenprotokoll direkt entsprechend der Spezifikation in Abschnitt 4.6 ("Datenprotokoll", Seite 55) umzusetzen. Technischere Ebenen der Implementierung werden in tiefere Schichten verpackt[15]. Dabei soll die Strukturierung analog zum ISO-OSI-Referenzmodell [Tanenbaum 1992] erfolgen.

Die unteren drei Schichten (Bitübertragung, Sicherung, Vermittlung) können unverändert aus gängigen Netzwerken übernommen werden. Das Fehldatensystem greift in die Transportschicht ein und bestimmt die Sitzungssteuerung, Darstellung und Anwendungskommunikation (Bild 5.1: Zuordnung des Fehldatensystems zum ISO-OSI-Schichtenmodell.). Auf diese Weise können neue Module erstellt werden, ohne in technische Aspekte einzugreifen. Insbesondere kann damit eine Programmierumgebung erzeugt werden, in der Fachleute der Anwendungsgebiete Module erstellen oder verändern, ohne allzu tiefgehende Programmier- oder Netzwerkkenntnisse besitzen zu müssen.

Die Mehrschichtenorganisation steht dabei in keinem direkten Zusammenhang mit den verschiedenen Ebenen, die Abschnitt 2.3 ("Gebäudelebenszyklus", Seite 12) einführt; eine Implementierung jeder der drei dort genannten Ebenen würde alle Schichten des ISO-OSI-Referenzmodells benötigen.

15. Werden diese Schichten mit wohldokumentierten Schnittstellen versehen, ist die Trennung in verschiedene Threads oder Applikation möglich. Diese Trennung führt zu einer Multi-Tier-Applikation, die vielerlei technische Vorteile bieten kann [Hartwich 2000]. Diese Option sollte nicht in Vergessenheit geraten, bleibt im folgenden aber unberücksichtigt.

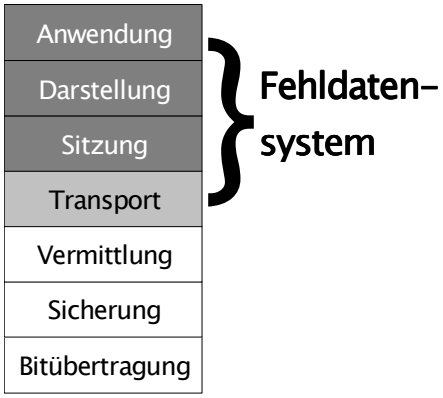

Bild 5.1: Zuordnung des Fehldatensystems zum ISO-OSI-Schichtenmodell.

5.1.1 Anwendungsschicht

In dieser Schicht erfolgt die Visualisierung der Daten sowie die (interaktive oder nicht-interaktive) Ermittlung von Werten. Die Funktionalität dieser Schicht ist somit modulspezifisch.

5.1.2 Darstellungsschicht

Diese Schicht konvertiert und speichert Daten und stellt diese für die Anwendungsschicht bereit. Im Fehldatensystem leistet die Darstellungsschicht die Umsetzung der Daten aus den Datenstrukturen der Anwendungsschicht in ein einheitliches und Fehldatensystem-spezifisches Datenformat. Dieses Datenformat kann wegen der Granularität auf Attributebene und der Dynamik der Anwendungsobjekte diese nicht direkt auf Implementierungsobjekte umsetzen, da die Struktur von Objekttypen in üblichen Programmierumgebungen vorab festgelegt werden muß. Daher wird zur Implementierung auf Listen von einfachen Strukturen zurückgegriffen, die für das Fehldatensystem Attributname[16], Wert und Qualitätsstufe umfassen.

Weiterhin verwaltet die Darstellungsschicht eine Liste aller Ein- und Ausgabeattribute, für die Informationen weiterzuleiten sind. Diese Informationen werden von der Sitzungsschicht bezogen.

Die Darstellungsschicht übernimmt somit die lokale (nicht-persistente) Datenhaltung und initiiert im Bedarfsfall, der sich aus dem betroffenen Attribut ergibt, die Weiterleitung an die Sitzungsschicht.

16. Der Attributname entspricht in der prototypischen Implementierung der Domäne des Attributes; siehe auch Abschnitt 4.4.2 ("Austausch einzelner Attributsdaten", Seite 44)

5.1.3 Sitzungsschicht

Die Sitzungsschicht ist im Fehldatensystem für die Koordination der Module zuständig. Protokollprimitive, die der Regelung dienen, welche Daten ein Modul austauscht oder mit welchen anderen Modulen Daten ausgetauscht werden, gehören zu dieser Schicht. Datenmeldungen werden (ohne Modulinformationen) an die Darstellungsschicht weitergeleitet bzw. von dieser empfangen. Hierzu konsolidiert die Sitzungsschicht die Ein- und Ausgabeattributlisten aller Module zu je einer.

Die Sitzungsschicht läßt sich somit als die Kapselung aller "anderen" Module im Fehldatensystem verstehen.

5.1.4 Tiefere Schichten

Die Transportschicht ist die hardwarenächste, die direkt im Fehldatensystem implementiert wird. Sie kann standardisierte Arten der Datenübertragung nutzen, um die in den abstrakteren Schichten codierten Daten zu versenden. Entsprechend können auch die Vermittlungs- Sicherungs- und Bitübertragungsschicht als unterste Schichten aus Standardbibliotheken übernommen werden, so etwa IP (Internet Protocol) für die Vermittlungsschicht.

5.2 Technische Protokolle

Auf Basis der genannten Schichtenzuordnung können nun die untergeordneten Protokolle spezifiziert werden.

Hierfür werden Primitive zur Kommunikation spezifiziert, die Parameter erhalten können. Die Primitive, ihre Parameter und ein Protokoll (Voraussetzungen zum Senden der Primitive und für gültige Antworten) werden im folgenden, getrennt nach Funktion der Primitive, erläutert.

Bei allen Primitiven wird vorausgesetzt, daß das Empfängermodul den Absender identifizieren kann. Ist dies aufgrund der gewählten Codierung oder Übermittlungstechnik nicht möglich, müssen zusätzliche Parameter zur Identifikation mitgesendet werden.

5.2.1 Anwendungsschicht an Darstellungsschicht

Zunächst wird das Datenprotokoll aus Abschnitt 4.6 ("Datenprotokoll", Seite 55) konkretisiert, das in der Darstellungsschicht Informationen der Anwendungs-

schicht empfängt. Dieses Protokoll wird "AnwendungAnDarstellung" genannt, wenn sich die folgenden Abschnitte oder die Implementierung hierauf beziehen.

DatenWert (Objekt, Attribut, Wert, Qualität)
DatenAnfrage (Objekt, Attribut, Qualität)

Diese Primitive können aus dem abstrakten Datenprotokoll direkt übernommen werden. Ohne Beschränkung der Allgemeinheit sollen alle Parameter Zeichenketten sein, da dies für die zunächst prototypische Anwendung besonders anschaulich und für die Fehlersuche vorteilhaft ist. Der Qualitätsparameter kann konsistent und plausibel dazu benutzt werden, mit der untersten Qualitätsstufe "Fehldaten" die Löschung eines Wertes, einer Anfrage oder einer Regel zu veranlassen – siehe auch Abschnitt 4.5.5 ("Abbildung fehlender Information", Seite 54).

EigeneAttributeFestlegen(Eingabeattributliste, Ausgabeattributliste)

Weiterhin bestimmt die Anwendungsschicht, welche Werte aus welchen anderen Werten ermittelt werden können. Die sich daraus ergebenden Eingabe- und Ausgabeattribute müssen an andere Module weitergegeben und deshalb bis zur Transportschicht durchgereicht werden, um entsprechende Daten und Anfragen zu empfangen.

5.2.2 Darstellungsschicht an Sitzungsschicht

Jedes neu gestartete Modul muß zunächst Kontakt zu einem anderen Modul erhalten, um sich in ein bestehendes Netz aus Fehldatenmodulen einzuhängen, und bedient sich dazu des noch folgenden Transportprotokolls. In Abhängigkeit von der technischen Implementierung kann dies vollautomatisch, nach Vorkonfiguration oder interaktiv erfolgen. Im Fall der interaktiven Kontaktaufnahme benötigt die Sitzungsschicht eine Benutzungsschnittstelle, die das Editieren der Kommunikationsparameter und das manuelle Auslösen einer Kontaktaufnahme erlaubt.

Jedes Modul muß eine Liste von ihm bekannten Modulen verwalten, mit denen es direkt Kontakt aufnehmen kann. Diese Liste kann auf sehr verschiedene Weise entstehen:

- Durch interaktive Eingabe/Konfiguration, gegebenenfalls mit lokaler Speicherung. Dies erfordert allerdings Kenntnisse der Benutzer/-innen über die Topografie des Fehldatensystems. Damit ersteht letztlich die Metaebene zur Koordination wieder auf, die dieses System vermeiden soll.

Folglich sollte diese Information besser in den Modulen selbst gespeichert werden. Dafür gibt es wiederum zwei Varianten:

- Einzelne Knoten fungieren als Router, so daß für nicht-routende Module die Kommunikation auf ein einziges Routingmodul beschränkt bleibt. Mit dem Nachteil, daß wegen der Routingnotwendigkeit nicht mehr alle Module willkürlich kombiniert werden können, erkauft man den Vorteil manueller Eingriffsmöglichkeiten in die Kommunikation, die ansonsten aufwendig (weil

in jedem Modul einzeln) umzusetzen wären. Dies führt ebenfalls zur Diskussion über den Sinn der Metaebene zurück.

- Jeder Knoten ermittelt die im Fehldatensystem verfügbaren Informationen selbsttätig, etwa durch Abfrage anderer Module. Dies muß im Kommunikationsprotokoll der Module entsprechend vorgesehen werden.

Da die letztgenannte Lösung der geplanten Arbeitsweise am besten entspricht und zudem die Routingmöglichkeit nicht ausschließt, wie Abschnitt 5.5.2 ("Systemmodul Routing", Seite 78) zeigt, soll das Sitzungsprotokoll ein Primitiv ModulStarten enthalten.

Ablauf der Modul-An- und Abmeldung

Jedes Modul wird nach dem Start zunächst versuchen, Verbindung mit bestehenden Modulen aufzubauen, um den Bestand vorhandener Daten zu empfangen. Hierzu ist zunächst die Kontaktaufnahme mit einem beliebigen Modul nötig, was die Sitzungsschicht auf Anforderung über das Primitiv ModulStarten betreibt. Das empfangende Modul wird mit der Liste der ihm bekannten Module antworten, so daß in der Sitzungsschicht des neuen Moduls eine Liste bekannter Module aufgebaut werden kann.

Im folgenden wird von den "Primitiven", die Funktionsaufrufe innerhalb eines Moduls sind, die "Botschaft" unterschieden, die eine Bytefolge zur Kommunikation zwischen Modulen bezeichnet.

ModulStarten(Eingabeattributliste, Ausgabeattributliste)

Das Primitiv ModulStarten kontaktiert mit oder ohne Benutzerinteraktion mindestens ein anderes Modul, sofern ein solches verfügbar ist, indem es diesen Modulen eine Botschaft der Art ModulHinzufügen (siehe Abschnitt 5.2.4 ("Botschaften zwischen Modulen", Seite 62)) sendet, die eine Antwort anfordert. Es reicht dazu die Attributlisten weiter, die es aus der Anwendungsschicht empfangen hat. Über die Antworten auf dieses Primitiv kann eine Liste bekannter Module aufgebaut werden.

ModulBeenden

Jedes Modul versendet, ausgelöst durch dieses Primitiv, an jeden Eintrag seiner Modulliste eine Botschaft der Art ModulLöschen, bevor es sich beendet; siehe Abschnitt 5.2.4 ("Botschaften zwischen Modulen", Seite 62). Das empfangende Modul entfernt das zu löschende daraufhin aus seiner Modulliste.

Übermittlung von Datenwerten und Anfragen

Weiterhin sind Primitive nötig, die Datenwerte und -anfragen übermitteln. Die Sitzungsschicht ermittelt anhand des betroffenen Attributs in Abhängigkeit vom Botschaftstyp (Wert oder Anfrage), an welche Module die Datenbotschaft gesendet wird:

DatenWertSenden (Attribut, Datenbotschaft)

Dieses Primitiv stellt einen fertig aufbereiteten Wert zum Versand zur Verfügung und benennt, welches Attribut dies betrifft. In der Datenbotschaft ist neben Objekt, Attribut, Wert und Qualität auch der Botschaftstyp (hier: Wertübermittlung) codiert, so daß diese später nicht mehr modifiziert werden muß.

Datenwert-Botschaften werden an alle Module gesendet, die das Attribut als Eingabeattribut benötigen.

DatenAnfrageSenden (Attribut, Datenbotschaft)

Dieses Primitiv stellt analog eine aufbereitete Anfrage zum Versand zur Verfügung und benennt das betreffende Attribut. Datenanfrage-Botschaften gehen an alle Module, die das Attribut als Ausgabeattribut gemeldet haben.

5.2.3 Sitzungsschicht an Transportschicht

BotschaftSenden (Modul, Botschaft)

Die Transportschicht hat keine semantische Kenntnis der übermittelten Nachrichten. Sie versendet beliebige Botschaften an ein Modul, das von der Sitzungsschicht beim Aufruf explizit genannt wird.

Die Datenprimitive der höheren Ebene, DatenWertSenden und DatenAnfrageSenden, fügen hierzu bereits eine fertig codierte Botschaft bei. Die Modulprimitive ModulStarten und ModulBeenden werden hingegen von der Sitzungsschicht in Botschaften umgesetzt.

5.2.4 Botschaften zwischen Modulen

In der Transportschicht sind alle Informationen in Botschaften verpackt, die an ein anderes Modul gesendet werden können. Die Anführungszeichen in der folgenden Aufstellung verdeutlichen, daß es sich dabei um codierte Nachrichten, nicht um Methodenaufrufe handelt.

Modulbezogene Botschaften

Lediglich vier Botschaften in zwei Kategorien sind nötig. Zunächst erfolgt die Sitzungsverwaltung mittels zweier Botschaften zur Steuerung der lokalen Modulliste:

"ModulHinzufügen (Modul, Eingabeattributliste, Ausgabeattributliste, Sendeflag)"

Diese Botschaft beschreibt, daß das empfangende Modul das in der Botschaft bezeichnete (das nicht zwangsläufig dem Absender entspricht) mit den genannten Ein- und Ausgabeattributen in seine lokale Modulliste eintragen soll. Wenn das Sendeflag gesetzt ist, wird zusätzlich diese Botschaft (jedoch mit gelöschtem Sendeflag) an alle Module der lokalen Modulliste weitergesendet sowie an das bezeichnete Modul die lokale Modulliste gesendet.

Sofern das bezeichnete Modul in der lokalen Modulliste bereits existiert, wird mit dieser Botschaft der Eintrag überschrieben. Auf diese Weise ist eine Aktualisierung der Angaben möglich, wenn ein Modul seine Interessenslage ändert.

Durch den Versand der kompletten eigenen Modulliste an ein anderes Modul bei gesetztem Sendeflag in allen Botschaften können leicht zwei ge-

trennte Fehldatensystem miteinander verschmolzen werden.

"ModulLöschen (Modul)"
: Das empfangende Modul löscht den Eintrag des in der Botschaft bezeichneten Moduls aus seiner lokalen Modulliste. Weitere Kommunikation mit dem bezeichneten Modul wird durch das empfangende nicht ausgelöst.

Datenbezogene Botschaften

Die zweite Gruppe von Botschaften beschäftigt sich mit dem Senden von Datenwerten und -anfragen:

"DatenWert (Objekt, Attribut, Wert, Qualität)"
: Diese Botschaft teilt dem empfangenden Modul ein Datenelement mit. Dieses ist durch Objekt und Attribut eindeutig bestimmt. Der Wert dieser Objekt-Attribut-Kombination und seine Qualitätsstufe vervollständigen die Botschaft.

"DatenAnfrage (Objekt, Attribut, Qualität)"
: Die DatenAnfrage ist das Analogon zur Botschaft DatenWert. Sie bezeichnet mit Objekt und Attribut, welcher Wert gesucht wird. Die Qualität bezeichnet hier die minimal geforderte Qualitätsstufe.

5.2.5 Transportschicht an Sitzungsschicht

BotschaftEmpfangen (Botschaft)
: Auch beim Empfang einer Botschaft nimmt die Transportschicht keine inhaltliche Bewertung vor. Informationen über den Sender der Botschaft setzt das Fehldatensytem nicht voraus, so daß diese ggf. in der Botschaft codiert sein müssen.

Die modulbezogenen Botschaften ModulHinzufügen und ModulLöschen werden innerhalb der Sitzungsschicht verarbeitet. Bei ModulHinzufügen ist insbesondere das Sendeflag von Wichtigkeit: Ist dieses gesetzt, leitet die Sitzungsschicht diese Botschaft (mit gelöschtem Sendeflag) an alle ihm bekannten Module weiter und antwortet dem Absender mit einer Liste aller ihm bekannten Module und deren Eigenschaften.

Sich aus den Änderungen der Modulstruktur ergebende Änderungen der Ein- und Ausgabeattribute fremder Module werden in der Sitzungsschicht konsolidiert und an die Darstellungsschicht weitergereicht.

5.2.6 Sitzungsschicht an Darstellungsschicht

AlleWerteSenden(Attribut)
: Nachdem die Sitzungsschicht "ModulHinzufügen" empfangen hat, setzt sie das hinzugefügte Modul über selbsterzeugte Datenwerte in Kenntnis. Dazu werden die Attribute des hinzugefügten Moduls mit den eigenen vergli-

chen. Alle Werte zu eigenen Ausgabeattributen, die zu den Ein- oder Ausgabeattributen des hinzugefügten Moduls gehören, werden mittels dieses Befehls von der Darstellungsschicht an das jeweilige Modul gesendet.

AlleAnfragenSenden(Attribut)
: Analog zu AlleWerteSenden behandelt dieser Befehl alle eigenen Anfragen zu Ausgabeattributen des hinzugefügten Moduls.

FremdeAttributeFestlegen(Eingabeattributliste, Ausgabeattributliste)
: Hiermit wird die Darstellungsschicht informiert, welche Attribute von (beliebigen anderen) Modulen als Eingabe- bzw. Ausgabeattribute benötigt werden, so daß entsprechende Werte bzw. Anfragen an die Sitzungsschicht weitergegeben werden müssen.

DatenBotschaftEmpfangen(Datenbotschaft)
: Datenbotschaften werden erst in der Darstellungsschicht analysiert. Alle Botschaften, die die Sitzungsschicht nicht als Modulbotschaften identifizieren konnte, werden daher als Datenbotschaft weitergeleitet.

5.2.7 Darstellungsschicht an Anwendungsschicht

DatenWertErmittelt (Objekt, Attribut, Wert, Qualität)
: Jeder neue Datenwert wird an die Anwendungsschicht weitergeleitet, damit diese ggf. die Bildschirmdarstellung anpassen oder weitere Werte berechnen kann (die dann wieder an die Darstellungsschicht gesendet werden müßten, um sie zu speichern und ggf. weiterzuleiten).

DatenWertAngefragt (Objekt, Attribut, Qualität)
: Auch Datenanfragen werden ausnahmslos an die Anwendungsschicht weitergeleitet. Diese kann damit die Ermittlung von Datenwerten priorisieren, ggf. weitere Anfragen an Fremdmodule auslösen oder an der Benutzungsoberfläche die angefragten Daten markieren.

5.3 Chronologischer Ablauf

Dieser Abschnitt zeigt eine kurze Beispielsitzung mit vier Modulen zur Erläuterung der Protokolle. Hierbei folgt die Notation der Konvention

<Modul>.<Protokollname>.<Primitivname>(<Parameterliste>)

Die Parameterliste ist hierbei optional. Sollte einer der Parameter wiederum eine Liste sein, wird diese in eckige Klammern gefaßt. Die Parameter werden wie folgt bezeichnet:

M_i Module, hier M_1 bis M_4

Pi Personen, die die Module bedienen. Px bedient hierbei genau Mx.

A_i Attribute. Der Index ist modulübergreibend: A_3 kann beispielweise Ausgabeattribut von M_1 und Eingabeattribut von M_2 sein.

Oi Objekte, wiederum mit modulübergreifendem Index. Welche Attribute konkret zu welchem Objekt gehören, ist für das Beispiel irrelevant, so daß davon ausgegangen wird, daß alle Objekte alle Attribute enthalten[17].

W_i Werte. Unterschiedliche Indizes bezeichnen unterschiedliche Werte, ohne daß mit der Indexzahl weitere Semantik verbunden wäre.

Q_i Qualitätsstufen, hier Q_1 (niedrigste) bis Q_5 (höchste)

Zur besseren Nachvollziehbarkeit der Beschreibung werden in diesem Beispiel zwei Annahmen getroffen, die die Allgemeingültigkeit nicht einschränken:

1. Jedes Modul Mx wird von genau einer Person Px bedient, und jede Person bedient nur ein Modul.
2. Alle Attributwerte werden nur aus Attributwerten desselben Objektes oder durch Interaktion ermittelt.

5.3.1 Starten des ersten Moduls

Beim ersten zu startenden Modul M1 werden zunächst die Regeln an die Darstellungsschicht übergeben:

```
M1.AnwendungAnDarstellung.EigeneAttributeFestlegen ([A1, A2, A4], [A3, A5])
```

Nachdem nun die Parameter der Regeln dieses Moduls der Darstellungsschicht bekannt sind, kann diese weitere Module suchen:

```
M1.DarstellungAnSitzung.ModulStarten([A1, A2, A4], [A3, A5])
```

Hierauf erfolgt keine Antwort, da das Modul das erste und bislang einzige laufende des Fehldatensystems ist. Die Sitzungsschicht ist damit über die Eigenschaften des eigenen Moduls informiert und kann diese weitergeben.

5.3.2 Ermittlung von Daten

Nehmen wir an, das laufende Modul M1 ermittle nun Daten. Diese Daten können durch Interaktion, Berechnung, Abfrage von verbundenen Geräten o.ä. erstellt werden. Insbesondere müssen die Eingabeattribute hierzu jedoch vorhanden sein. Nehmen wir zur Vereinfachung an, die folgenden Werte seien im Modul lokal gespeichert gewesen und würden nun wiederhergestellt. Sie werden den angemeldeten Ausgabeattributen A3 und A5 in den jeweiligen Objekten zugeordnet:

17. Die Betrachtungsweise von Daten als Objekt bei gleichzeitiger Nutzung einzelner Attribute als Daten(austausch)einheit führt Abschnitt 4.4.2 ("Austausch einzelner Attributsdaten", Seite 44) ein.

M1.DarstellungAnAnwendung.DatenWertErmittelt (O1, A1, W1, Q3)

M1.DarstellungAnAnwendung.DatenWertErmittelt (O1, A2, W2, Q4)
M1.DarstellungAnAnwendung.DatenWertErmittelt (O1, A3, W3, Q3)
M1.DarstellungAnAnwendung.DatenWertErmittelt (O1, A4, W4, Q4)
M1.DarstellungAnAnwendung.DatenWertErmittelt (O1, A5, W5, Q3)
M1.DarstellungAnAnwendung.DatenWertErmittelt (O2, A1, W6, Q4)
M1.DarstellungAnAnwendung.DatenWertErmittelt (O2, A2, W7, Q4)
M1.DarstellungAnAnwendung.DatenWertErmittelt (O2, A3, W8, Q4)

Beachtenswert ist hier, daß nur O1.A3, O2.A3 und O1.A5 Ausgabeattribute von M1 sind. Da jedoch die Interpretation und Speicherung auf der Darstellungsschicht erfolgt, sendet diese alle Daten zur Anzeige und weiteren Verarbeitung an die Anwendungsschicht. Außerdem bleibt die Qualitätsstufe von O1.A3 und O1.A5 unter der dem Modul maximal möglichen Stufe zurück, offenbar weil O1.A1 nur die Qualitätsstufe Q3 erreicht.

5.3.3 Starten und Verbinden des zweiten Moduls

Der Startvorgang des Moduls M2 beginnt zunächst analog zu dem des ersten:

M2.AnwendungAnDarstellung.EigeneAttributeFestlegen ([A5], [A1, A2, A6])
M2.DarstellungAnSitzung.ModulStarten([A5], [A1, A2, A6])

Dieses Modul ist also in der Lage, A1 und A2 bei höchster Qualitätsstufe (bei der vorgesehenen Skala entsprechend durch Messung) und ohne Voraussetzung anderer Werte als Eingabeparameter zu ermitten. Das Ausgabeattribut A5 des Moduls M1 ist hier Eingabeattribut zur Ermittlung von A6.

Auf "ModulStarten" hin nimmt nun M2 Kontakt mit M1 auf. M1 empfängt diese Botschaft, meldet die veränderte Interessenslage seiner Darstellungsschicht und antwortet aufgrund des gesetzten Sendeflags mit der Liste aller ihm bekannten Module, nämlich nur sich selbst. M2 schließlich verarbeitet die Dateninteressen von M1 analog (Bild 5.2: Anmeldung eines zweiten Moduls im Fehldatensystem):

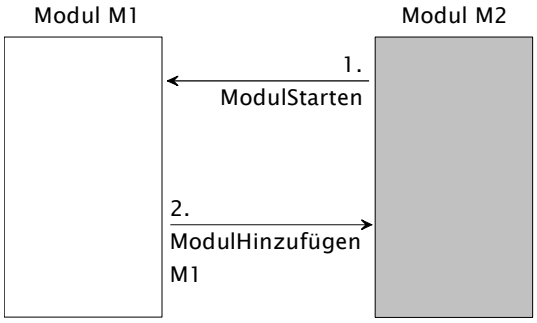

Bild 5.2: Anmeldung eines zweiten Moduls im Fehldatensystem

M2.SitzungAnTransport.BotschaftSenden (M1, "ModulHinzufügen (M2, [A5], [A1, A2, A6], true)")
M1.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M2, [A5], [A1, A2, A6], true)")
M1.SitzungAnDarstellung.FremdeAttributeFestlegen([A5], [A1,A2,A6])
M1.SitzungAnTransport.BotschaftSenden (M2, "ModulHinzufügen (M1, [A1, A2, A4], [A3, A5], false)")
M2.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M1, [A1, A2, A4], [A3, A5], false)")
M2.SitzungAnDarstellung.FremdeAttributeFestlegen([A1, A2, A4], [A3, A5])

Damit ist beiden Modulen alle erforderliche Information bekannt, um zusammenzuarbeiten.

5.3.4 Kommunikation von Datenänderungen

Die Darstellungsebene von M1 prüft nun die vorhandenen Daten auf Nutzen und sendet O1.A5 mit Hilfe der Sitzungsschicht an M2:

M1.DarstellungAnSitzung.DatenWertSenden (A5, "DatenWert (O1, A5, W5, Q3)")
M1.SitzungAnTransport.BotschaftSenden (M2, "DatenWert (O1, A5, W5, Q3)")
M2.TransportAnSitzung.BotschaftEmpfangen ("DatenWert (O1, A5, W5, Q3)")
M2.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenWert (O1, A5, W5, Q3)")
M2.DarstellungAnAnwendung.DatenWertErmittelt (O1, A5, W5, Q3)

Modul M2 habe nun einen Wert für O1.A1 ermittelt, der den gespeicherten Wert in M1 überschreibt, da er eine höhere Qualitätsstufe hat:

M2.AnwendungAnDarstellung.DatenWert(O1, A1, W9, Q5)
M2.DarstellungAnSitzung.DatenWertSenden (A1, "DatenWert (O1, A1, W9, Q5)")
M2.SitzungAnTransport.BotschaftSenden (M1, "DatenWert (O1, A1, W9, Q5)")
M1.TransportAnSitzung.BotschaftEmpfangen ("DatenWert (O1, A1, W9, Q5)")
M1.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenWert (O1, A1, W9, Q5)")
M1.DarstellungAnAnwendung.DatenWertErmittelt (O1, A1, W9, Q5)

Dies führt zu einer Neuberechnung in M1, zunächst von A3. Dieses erreicht die höhere Qualitätsstufe Q4 (jedoch nicht Q5, weil sein anderer Eingabeparameter immernoch Q4 ist). Da sich kein Modul für A3 interessiert, bleibt diese Änderung in der Darstellungsschicht ohne weitere Auswirkungen:

M1.AnwendungAnDarstellung.DatenWert(O1, A3, W10, Q4)

Auch A5 wird neu berechnet und ändert seine Qualitätsstufe, wird jedoch daraufhin an M2 gesendet:

M1.AnwendungAnDarstellung.DatenWert(O1, A5, W11, Q4)
M1.DarstellungAnSitzung.DatenWertSenden (A5, "DatenWert (O1, A5, W11, Q4)")
M1.SitzungAnTransport.BotschaftSenden (M2, "DatenWert (O1, A5, W11, Q4)")

M2 kann diesen Wert nun verarbeiten und hieraus A6 ermitteln:

M2.TransportAnSitzung.BotschaftEmpfangen ("DatenWert (O1, A5, W11, Q4)")
M2.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenWert (O1, A5, W11, Q4)")
M2.DarstellungAnAnwendung.DatenWertErmittelt (O1, A5, W11, Q4)
M2.AnwendungAnDarstellung.DatenWert(O1, A6, W12, Q4)

5.3.5 Kommunikation von Anfragen

Angenommen, die Werte zu A6 sei für den weiteren Planungsverlauf zu diesem Zeitpunkt nötig. Der Benutzer oder die Benutzerin des Moduls M2 signalisiert dies interaktiv, so daß das Modul für O2, wo der nötige Eingabewert A5 noch fehlt, eine entsprechende Anfrage mit Angabe der Mindestqualitätsstufe stellen kann:

M2.AnwendungAnDarstellung.DatenAnfrage(O2, A5, Q3)
M2.DarstellungAnSitzung.DatenAnfrageSenden(A5, "DatenAnfrage(O2, A5, Q3)")
M2.SitzungAnTransport.BotschaftSenden(M1, "DatenAnfrage(O2, A5, Q3)")

M1 empfängt diese Anfrage, kann sie aber ein Ermangelung eines Wertes für A4 nicht beantworten. Es stellt deshalb seinerseits eine Anfrage, die jedoch in der Sitzungsschicht hängenbleibt, weil derzeit keine Module existieren, die A4 ermitteln:

M1.TransportAnSitzung.BotschaftEmpfangen ("DatenAnfrage(O2, A5, Q3)")
M1.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenAnfrage(O2, A5, Q3)")
M1.DarstellungAnAnwendung.DatenWertAngefragt (O2, A5, Q3)
M1.AnwendungAnDarstellung.DatenAnfrage(O2, A4, Q3)
M1.DarstellungAnSitzung.DatenAnfrageSenden(A4, "DatenAnfrage(O2, A4, Q3)")

5.3.6 Verbinden zweier Fehldatennetze

Zwei getrennt laufende Fehldatensysteme können jederzeit verbunden werden, sobald dies technisch und inhaltlich sinnvoll ist.

Im vorliegenden Fall kann die Person P1 an Modul M1 potentiell feststellen, daß der Datenwert A4 die weitere Verarbeitung hemmt. Er oder sie kann nun auf üblichem Weg per Telefon, Email oder persönlichem Kontakt veranlassen, daß dieser Wert ermittelt wird. Dies ist bereits eine signifikante Verkürzung der Kommunikation, da bei korrekter Definition der Module normalerweise zunächst die Person P2 die Person P1 kontaktiert hätte, um (möglicherweise mit Verzögerung) herauszufinden, daß P1 diese Daten derzeit ebenfalls nicht liefern kann. In dieser unbürokratischen Propagierung von Anfragen liegt der möglicherweise größte Vorzug des Fehldatensystems.

Realistisch könnte etwa in einem anderen Büro ein Team mit zwei Modulen M3 und M4 arbeiten, wobei folgende sehr einfache Datenregeln gelten sollen:

- M3 ermittelt A4 aus A7.
- M4 ermittelt A7 ohne weitere Eingabewerte.
- M4 interessiert sich für A6 (das von M2 ermittelt wird).

Beide Module seien bereits gestartet und miteinander verbunden. Der kompliziertestmögliche Fall tritt nun ein, wenn M1 und M3 miteinander verbunden werden, da A6 zwischen zwei Modulen ausgetauscht werden muß, die zunächst nichts voneinander wissen.

P1 erhält nun von P3 die nötige Information zur Verbindung zwischen M1 und M3 und löst diese manuell auf der Sitzungsebene aus. Infolge dessen sendet M1 alle Modulinformationen über das eigene Fehldatensystem an M3 (Bild 5.3: Verbinden zweier Fehldatennetze mit je zwei Modulen):

M1.SitzungAnTransport.BotschaftSenden (M3, "ModulHinzufügen (M1, [A1, A2, A4], [A3, A5], true)")
M1.SitzungAnTransport.BotschaftSenden (M3, "ModulHinzufügen (M2, [A5], [A1, A2, A6], true)")
M3.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M1, [A1, A2, A4], [A3, A5], true)")
M3.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M2, [A5], [A1, A2, A6], true)")

Die Botschaften werden beim Empfänger in der Sitzungsschicht analysiert und sogleich umgesetzt, was in diesem Ablauf nicht ersichtlich ist. Da in den Botschaften

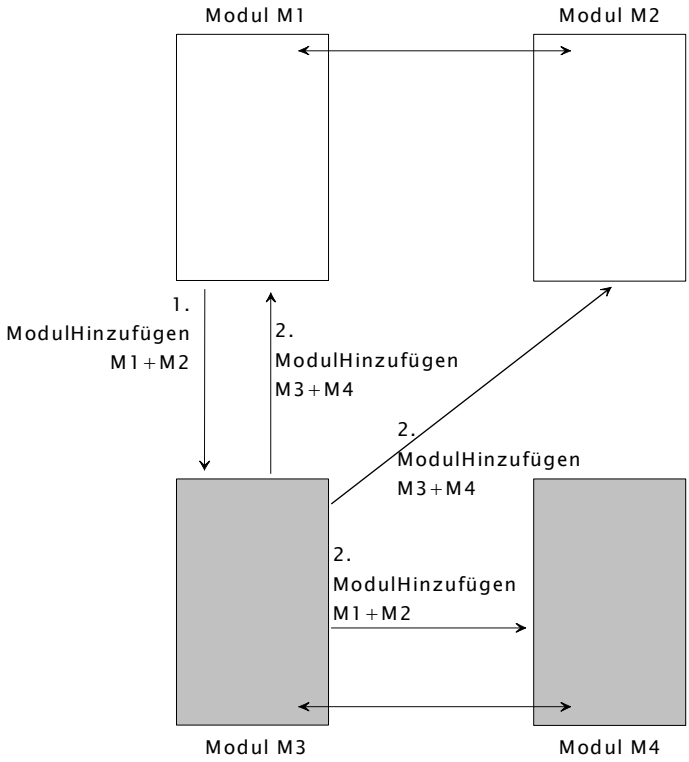

Bild 5.3: Verbinden zweier Fehldatennetze mit je zwei Modulen

über M1 und M2 das Sendeflag gesetzt war, beantwortet M3 beide Anfragen mit allen Daten seines eigenen Fehldatensystems und sendet beide Botschaften an alle Module seiner lokalen Modulliste (nämlich M4) weiter:

M3.SitzungAnTransport.BotschaftSenden (M1, "ModulHinzufügen (M3, [A7], [A4], false)")
M3.SitzungAnTransport.BotschaftSenden (M1, "ModulHinzufügen (M4, [A6], [A7], false)")
M3.SitzungAnTransport.BotschaftSenden (M2, "ModulHinzufügen (M3, [A7], [A4], false)")
M3.SitzungAnTransport.BotschaftSenden (M2, "ModulHinzufügen (M4, [A6], [A7], false)")
M3.SitzungAnTransport.BotschaftSenden (M4, "ModulHinzufügen (M1, [A1, A2, A4], [A3, A5], false)")
M3.SitzungAnTransport.BotschaftSenden (M4, "ModulHinzufügen (M2, [A5], [A1, A2, A6], false)")

Bei den empfangenden Modulen M1, M2 und M4 werden diese Botschaften aufgelöst und in die Modulliste aufgenommen:

M1.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M3, [A7], [A4], false)")
M1.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M4, [A6], [A7], false)")
M2.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M3, [A7], [A4], false)")
M2.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M4, [A6], [A7], false)")
M4.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M3, [A7], [A4], false)")
M4.TransportAnSitzung.BotschaftEmpfangen ("ModulHinzufügen (M4, [A6], [A7], false)")

Damit sind beide Netze verschmolzen; alle vier Module sind nun über alle anderen informiert. Anschließend folgt der Austausch aller relevanten Daten und Anfragen. Bezüglich der Daten betrifft dies in diesem Fall nur A6, das von M4 verarbeitet wird:

M2.DarstellungAnSitzung.DatenWertSenden (A6, "DatenWert (O1, A6, W12, Q4)")
M2.SitzungAnTransport.BotschaftSenden (M4, "DatenWert (O1, A6, W12, Q4)")
M4.TransportAnSitzung.BotschaftEmpfangen ("DatenWert (O1, A6, W12, Q4)")
M4.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenWert (O1, A6, W12, Q4)")
M4.DarstellungAnAnwendung.DatenWertErmittelt (O1, A6, W12, Q5)

Insbesondere kann M1 nun die Anfrage bezüglich A4 bearbeiten, indem es sie weitersendet. M3 löst diese in eine Anfrage an M4 bezüglich A7 auf:

M1.SitzungAnTransport.BotschaftSenden(M3, "DatenAnfrage(O2, A4, Q3)")
M3.TransportAnSitzung.BotschaftEmpfangen ("DatenAnfrage(O2, A4, Q3)")
M3.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenAnfrage(O2, A4, Q3)")
M3.DarstellungAnAnwendung.DatenWertAngefragt (O2, A4, Q3)
M3.AnwendungAnDarstellung.DatenAnfrage(O2, A7, Q3)
M3.DarstellungAnSitzung.DatenAnfrageSenden(A7, "DatenAnfrage(O2, A7, Q3)")
M3.SitzungAnTransport.BotschaftSenden(M4, "DatenAnfrage(O2, A7, Q3)")
M4.TransportAnSitzung.BotschaftEmpfangen ("DatenAnfrage(O2, A7, Q3)")
M4.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenAnfrage(O2, A7, Q3)")
M4.DarstellungAnAnwendung.DatenWertAngefragt (O2, A7, Q3)

M4 ermittelt den noch nicht vorhandenen Wert A7, so daß M3 A4 senden kann. Hieraus ermittelt M1 A5 und sendet diesen Wert an M2:

M4.AnwendungAnDarstellung.DatenWert(O2, A7, W13, Q3)

M4.DarstellungAnSitzung.DatenWertSenden (A7, "DatenWert (O2, A7, W13, Q3)")
M4.SitzungAnTransport.BotschaftSenden (M3, "DatenWert (O2, A7, W13, Q3)")
M3.TransportAnSitzung.BotschaftEmpfangen ("DatenWert (O2, A7, W13, Q3)")
M3.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenWert (O2, A7, W13, Q3)")
M3.DarstellungAnAnwendung.DatenWertErmittelt (O2, A7, W13, Q3)
M3.AnwendungAnDarstellung.DatenWert (O2, A4, W14, Q3)
M3.DarstellungAnSitzung.DatenWertSenden (A4, "DatenWert (O2, A4, W14, Q3)")
M3.SitzungAnTransport.BotschaftSenden (M1, "DatenWert (O2, A4, W14, Q3)")
M1.TransportAnSitzung.BotschaftEmpfangen ("DatenWert (O2, A4, W14, Q3)")
M1.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenWert (O2, A4, W14, Q3)")
M1.DarstellungAnAnwendung.DatenWertErmittelt (O2, A4, W14, Q3)
M1.AnwendungAnDarstellung.DatenWert (O2, A5, W15, Q3)
M1.DarstellungAnSitzung.DatenWertSenden (A5, "DatenWert (O2, A5, W15, Q3)")
M1.SitzungAnTransport.BotschaftSenden (M2, "DatenWert (O2, A5, W15, Q3)")
M2.TransportAnSitzung.BotschaftEmpfangen ("DatenWert (O2, A5, W15, Q3)")
M2.SitzungAnDarstellung.DatenBotschaftEmpfangen ("DatenWert (O2, A5, W15, Q3)")
M2.DarstellungAnAnwendung.DatenWertErmittelt (O2, A5, W15, Q3)

5.3.7 Abmelden eines Moduls

Wenn ein Modul, hier M1, aus diesem zusammengeschlossenen Fehldatensystem beendet wird, wird nur der jeweilige Eintrag des endenden Moduls gelöscht. Die restlichen Module bleiben verbunden:

M1.DarstellungAnSitzung.ModulBeenden
M1.SitzungAnTransport.BotschaftSenden(M2, "ModulLöschen(M1)")
M1.SitzungAnTransport.BotschaftSenden(M3, "ModulLöschen(M1)")
M1.SitzungAnTransport.BotschaftSenden(M4, "ModulLöschen(M1)")
M2.TransportAnSitzung.BotschaftEmpfangen(M4, "ModulLöschen(M1)")
M3.TransportAnSitzung.BotschaftEmpfangen(M4, "ModulLöschen(M1)")
M4.TransportAnSitzung.BotschaftEmpfangen(M4, "ModulLöschen(M1)")

Auch hier finden wieder Analyse und sofortige Umsetzung innerhalb der Sitzungsschicht der empfangenden Module statt.

5.4 Technische Umsetzung

Bis zu diesem Moment ist die getroffene Spezifikation unabhängig von Hardware- und Softwareumgebung. Die folgenden zwei Abschnitte legen nun die technischeren Ebene des OSI-Schichtenmodells sowie die Programmierumgebung (und damit die verfügbaren Hardware-/Betriebssystem-Plattformen) fest.

5.4.1 Transportverfahren

Rechnerkomponenten und Netzwerkverbindungen sind bekanntlich ausfallgefährdet. Aufgrund der vielen beteiligten Hardwarekomponenten ist dieser Umstand bei verteilten Anwendungen wie dem Fehldatensystem von besonderer Bedeutung.

Um für den Fall einer unterbrochenen oder abgebrochenen Kommunikation ungültige Systemzustände zu vermeiden, können einerseits Fallback-Strategien bzw. explizite, eventuell interaktiv durchzuführende Maßnahmen zur Rettung des Bearbeitungszustandes im Rahmen einer Transaktionssteuerung getroffen werden. Andererseits kann das System so aufgebaut werden, daß es nicht auf kontinuierlich bestehende Verbindungen aufsetzt. Für das Fehldatensystem sind zunächst beide Alternativen möglich.

Ausfallbehandlung mittels Transaktionen

Abgesicherte Verbindungen erlauben, zuverlässig den Ausfall von Komponenten festzustellen. In Verbindung mit Transaktionen können so auch bei ausgefallenen Komponenten konsistente Systemzustände garantiert werden. Dieser Aufbau einer “sicheren” Verbindung auf Anwendungsebene resultiert in der Suggestion, es handle sich um ein geschlossenes und einfaches System. Treten die unvermeidlichen Ausfälle auf, werden Benutzerin oder Benutzer mit Meldungen wie *“No Route to Host”, “Network Timeout”* und ähnlichem auf eine technische Ebene der Fehlerbeschreibung gezwungen, die in der Regel in hilflosen Hinweisen wie “Benachrichtigen Sie Ihren Systemadministrator” endet, da typische Anwender/-innen diese Meldungen nicht interpretieren, geschweige denn die Ursache beheben können. Jedes Zurücksetzen auf den letzten konsistenten Zustand erfordert aber einen Hinweis an Benutzerin oder Benutzer, wenn dabei (und das ist der übliche Fall) einzelne bearbeitete Daten verlorengehen. Damit ist die Transaktionsstrategie nur dann gerechtfertigt, wenn (etwa in einem lokalen Netz) hohe Ausfallsicherheit angenommen werden kann, mithin die Vereinfachung durch die abgesicherte Anwendungsebene stärker zu gewichten ist als die Nachteile der expliziten Ausfallbehandlung.

Genau dies soll das Fehldatensystem nach Möglichkeit vermeiden. Auch wenn dies keinesfalls zwingend erforderlich ist, soll daher ein Transportverfahren gewählt werden, das keine dauerhaft beständige Verbindung voraussetzt. Botschaften werden so innerhalb des Fehldatensystems verschickt, ohne auf Rückmeldung zu warten: Das sendende Modul kümmert sich nicht darum, ob die Nachricht ankommt. Dies hat den Vorteil, daß das sendende Modul sich nicht mit Erfolg oder Mißerfolg der Aussendung befassen muß. Solche Transportmechanismen existieren bereits und heißen "Datagramm".

Wenn man ein übliches IP-Netzwerk als gegeben annimmt, da es das am weitesten verfügbare plattformübergreifende System ist, so verzichtet man für den Einsatz von Datagrammen auf TCP als Protokoll und setzt stattdessen UDP als Datagramm-Transportprotokoll ein. Das Fehldatensystem soll also auf UDP/IP aufbauen[18].

Vorgehen bei der UDP-Kommunikation

Bei der Kommunikation über UDP werden Sender und Empfänger über IP-Adresse (respektive Rechnername, wenn Nameserver zur Verfügung stehen) sowie einen Port identifiziert. Da ein Modul eindeutig adressierbar sein muß, ist die einfachste Möglichkeit zur Modulidentifikation die Kombination aus IP-Adresse und Port.

Daraus ergibt sich eine recht einfache allgemeingültige Möglichkeit, ein sich selbst konfigurierendes Fehldatensystem zu gestalten: Da auf jedem Rechner mehrere Module laufen können, wählt man einen Standardport, an den das neu gestartete Modul zunächst sendet, und einen Testport, auf dem das neue Modul zunächst empfängt. Wenn auf dem Standardport kein Modul ansprechbar ist, erhält das Modul den Standardport. Ansonsten ist bereits Anschluß an das Fehldatensystem gefunden. Der Testport wird nun wieder freigegeben. Lediglich, wenn die Module eines Rechners zum ersten Mal mit einem Fehldatensystem auf anderen Computern verbunden werden sollen, ist einmalig ein manueller Eingriff zur Angabe der IP-Adresse nötig[19].

Dies alles könnte direkt und elegant mit der Botschaft "ModulHinzufügen" geschehen, wenn nicht damit im gesamten Fehldatensystem die Testportadresse als die des Moduls verbreitet würde. Dies läßt sich entweder durch das (unelegante) Ab- und wieder Anmelden beheben, oder aber durch zusätzliche Testbotschaften[20]. Der Ablauf beim Starten eines Moduls ist damit wie folgt:

- Neues Modul sendet Botschaft "ModulTest" mit seiner eigenen Modul-ID an Standard-Port auf lokalem Rechner.
- Solange keine Antwort kommt, arbeitet das Modul ohne Anschluß an andere auf dem Testport.
- Eine eventuelle Antwortbotschaft "ModulPort (Portnummer)" eines laufenden Moduls enthält einen freien Port, den das antwortende Modul aus seiner lokalen Modulliste ermitteln kann. Diesen weist sich das Modul zu und meldet sich beim Standardport-Modul ordnungsgemäß mit "ModulHinzufügen" an.
- Nach einer Timeout-Zeit (z.B. sind in der Regel 10 Sekunden mehr als ausreichend, da die Abfrage auf dem lokalen Computer erfolgt) weist sich das Modul den Standardport zu, falls es zu diesem Zeitpunkt immernoch den Testport benutzt.

18. Dieses Vorgehen ist zweifellos experimentell und kann, wie bereits geschildert, durch eine übliche TCP/IP-Verbindung über Ports oder durch Named Pipe oder viele andere Mechanismen des Datenaustauschs ersetzt werden. Da der Datagramm-Transport in besonderer Weise dem Fehlen jeglicher Prämissen zur Verfügbarkeit im Fehldatensystems entspricht, soll dieses Experiment dennoch hier begonnen werden, um in dieser Arbeit Erkenntnisse über die besonderen Eigenschaften dieser Kombination aus Datagramm-Kommunikation und weitgehend selbstkonfigurierendem verteilten System zu gewinnen.

19. Natürlich läßt sich mit mehr Aufwand auch dieser manuelle Eingriff in einem Großteil der Fälle noch vermeiden, etwa durch Hinterlegung einer IP-Adresse und eines Ports, die immer zuerst abgefragt werden sollen. Dieses Modul würde dann als "Telefonbuch" dienen. Da aber der Aufbau des Fehldatensystems zur Illustration des Konzeptes so einfach wie möglich sein soll, wird auf diesen Ausbau verzichtet.

20. Manche UDP-Komponenten stellen eine Abfrage zur Verfügung, ob ein bestimmter Port von UDP belegt ist. Da diese jedoch nicht immer implementiert ist, wird auf ihre Verwendung hier verzichtet.

- Nachdem der Testport durch einen anderen ersetzt wurde, kann interaktiv der Kontakt zu anderen Modulen aufgebaut werden.

5.4.2 Programmierumgebung

Nachdem Struktur und Transportverfahren festliegen, kann die Auswahl der Programmierumgebung weitgehend pragmatischen Erwägungen folgen. Unmittelbare Voraussetzung ist lediglich die Unterstützung des Transportverfahrens. Wählt man die für größere, interaktive Anwendungen übliche Familie der objektorientierten Programmiersprachen, bleibt immernoch die Wahl zwischen Sprachen wie

- Smalltalk oder Java, auch Microsofts C#, als "reinrassigen" objektorientierten Sprachen
- Objective-C als syntaktisch saubere Erweiterung des prozeduralen C mit Elementen von Smalltalk
- C++, Delphi/Kylix von Borland oder VisualBasic von Microsoft als Derivate prozeduraler Sprachen.

Diese Liste erhebt natürlich keinen Anspruch auf Vollständigkeit. Wesentliche Faktoren bei der Auswahl sollten die Entwicklungsumgebung, die Unterstützung der notwendigen Funktionalität durch Bibliotheken sein. Für das Fehldatensystem von besonderer Bedeutung sind Kommunikationskomponenten für die unteren OSI-Schichten, die etwa blockierend (Hauptprogramm wartet) oder nichtblockierend (Hauptprogramm läuft während der Kommunikation weiter) ausgelegt sein können und damit maßgeblich die Implementierung beeinflussen.

Der Autor gesteht freimütig, aus der obigen Aufzählung aufgrund der modernen Sprachkonzepte Java oder C# zu präferieren, obwohl er wegen seiner persönlichen Programmiererfahrung Delphi zur Implementierung genutzt hat. Durch die Nutzung der CLX-Bibliotheken ist dabei eine Sourcecode-Kompatibilität zu Kylix gegeben, so daß das Fehldatensystem ohne Portierung unter Windows und Linux läuft. Auf Basis der Protokolle sollte auch eine Portierung in andere Programmiersprachen keine große Hürde darstellen.

5.4.3 Benutzungsschnittstelle

Mit der Modellierung von Datenanfragen und der Zusatzinformation der Qualitätsstufe zu einem Datenwert kommen Metadaten zum Wert hinzu, die für die Datenbearbeitung durch einen Menschen relevant sind. Damit stellt sich die Frage, wie diese Informationen auf der Benutzungsschnittstelle repräsentiert werden.

Modale und nichtmodale Dialoge

Anfragen eines Systems an Benutzerinnen und Benutzer werden häufig durch Dialogfenster visualisiert, die "modal" oder "nichtmodal" sein können. Modale Dialoge blockieren die weitere Bearbeitung innerhalb der Anwendung, während nichtmodale Dialoge vom Menschen zur späteren Bearbeitung zurückgestellt werden können. Ist von einem System eine größere Anzahl von Anfragen zu erwarten, wird die

Visualisierung mit Dialogen jedoch schnell unpraktikabel, da sie im modalen Fall eine ständige Blockade der regulären Arbeit, im nichtmodalen Fall immernoch eine häufige Arbeitsunterbrechung durch das explizite Zurückstellen der Dialoge erfordert. Selbst wenn nichtmodale Dialoge defensiv im Hintergrund erscheinen würden (was ihrem Sinn, Aufmerksamkeit zu erringen, zuwider liefe), wären sie ungeeignet, wenn pro regulärem Bearbeitungsvorgang in einem typischen Szenario durchschnittlich auch nur mehr als eine Anfrage erwartet würde, da die Bertrachtung jedes einzelnen Dialogs einen Fenster- und damit Kontextwechsel erfordert.

Dabei sollte ein "Bearbeitungsvorgang" die Arbeitsschritte innerhalb eines Kontextes umfassen, also etwa die Anpassung eines Bauteils an veränderte Bedingungen oder die Veränderung eines Geschosses hinsichtlich neuer Anforderungen. Ein solcher Bearbeitungsvorgang kann leicht viele Bauteile verändern und damit eine große Anzahl von Anfragen auslösen. Innerhalb der Bearbeitungszeit kann daher durch eine andere Person leicht eine Anfragenmenge entstehen, die Dialoge unpraktikabel werden läßt.

Es muß daher eine Lösung gesucht werden, die den Menschen nicht aus dem aktuellen Arbeitskontext herauszwingt. Hierbei ist zwischen zwei Aspekten zu unterscheiden, die unterschiedliche Darstellungsarten erfordern:

- Informationen zur Qualitätsstufe sind potentiell zu jedem Zeitpunkt interessant. Sie müssen jederzeit unaufwendig abrufbar sein, aber sind immer nur für das aktuell bearbeitete Element relevant.
- Informationen zu Anfragen sind naturgemäß nur von Belang, sofern Anfragen vorliegen. Es kann sinnvoll davon ausgegangen werden, daß keine Anfrage existiert, sofern die Applikation nichts Gegenteiliges anzeigt. Allerdings ist wichtig, daß Anfragen applikationsweit leicht zu identifizieren sind, da deren Vorhandensein die Bearbeitungspriorität des Menschen beeinflussen kann und soll.

Die Mensch-Maschine-Schnittstelle eines interaktiven Computersystems mit grafischer Benutzungsoberfläche ist von so hoher Komplexität, das sie nicht sinnvoll innerhalb dieser Arbeit betrachtet werden kann. Der Prototyp des Fehldatensystems setzt daher nach empirischen Versuchen folgende Visualisierung ein:

- Die Qualitätsstufe wird durch "Hints" dargestellt. Diese blenden kleine rahmenlose Fenster neben einem Datenwert ein, wenn der Mauszeiger über dem Wert verharrt, und blenden diese bei Maus- oder Tastaturaktion selbsttätig aus (siehe Bild 5.4: Fehldatenmodul mit Anzeige von Anfragen und Qualitäten.). Alternativ kommt die Einblendung in eine Statuszeile in Frage, die die Überblendung eines Fensterteils vermeidet, aber aufgrund der räumlichen Entfernung zum Datenelement einen Blickwechsel erfordert.
- Das Vorhandensein einer Anfrage wird durch farbige Hinterlegung des Datenwertes visualisiert. Details zur Anfrage werden in den Hint aufgenommen. (Bild 5.4: Fehldatenmodul mit Anzeige von Anfragen und Qualitäten.) Um die Entscheidung der Bearbeitungspriorität durch den Menschen zu unterstützen, können zusätzlich Symbole eingesetzt werden, wenn angefragte Datenwerte nicht aktuell sichtbar sind. Listen- Baum- und Karteiansichten

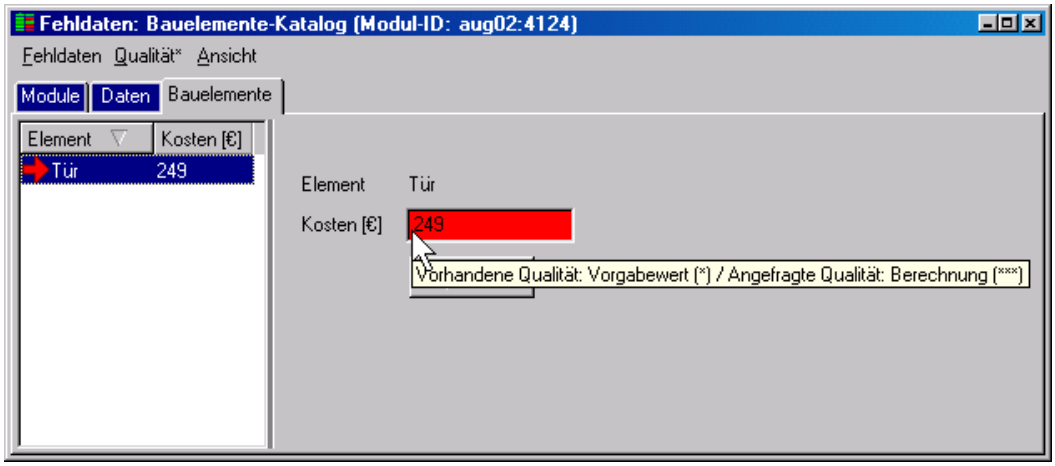

Bild 5.4: Fehldatenmodul mit Anzeige von Anfragen und Qualitäten.

Der (rote) Pfeil und die Hinterlegung des Eingabefeldes kennzeichnen, daß der entsprechende Wert angefragt wurde. Der "Hint" informiert über die vorhandene und angefragte Qualitätsstufe.

verbergen häufig mehrere Datenwerte hinter einer Schaltfäche, die diese Werte einblendet bzw. in den Vordergrund bringt. Diese Schaltflächen sollten ein Symbol erhalten, das Anfragen andeutet, sofern zu den von ihnen verborgenen Datenwerten Anfragen vorliegen.

Der eher intutive Einsatz dieser Mittel hat sich für den Prototyp als praktikabel erwiesen. Speziell bei hinsichtlich Modulanzahl und -vielfalt umfangreicheren Fehldatensystem muß die Qualität dieser Darstellung jedoch tiefergehend untersucht werden.

5.5 Grundlegende Module

Die Implementierung eines realen Fehldatensystems basiert auf einem Template-Modul, das alle beschriebenen Protokollebenen und weitere nötige Funktionalität zur Verfügung stellt, um durch eine entsprechende Benutzungsschnittstelle und einfache Methodenaufrufe System- und Anwendungsmodule abzuleiten.

Dieser Abschnitt beschreibt verschiedene Arten Module, die zum Teil in der prototypischen Implementierung des Fehldatensystems umgesetzt sind. Dabei wird unterschieden nach

- dem Template-Modul, das grundlegende Eigenschaften aller Module implementiert und an die folgenden Modularten vererbt,
- Systemmodulen, die rollenübergreifende Aufgaben im Fehldatensystem erfüllen

- Schnittstellenmodulen, die Verbindungen zu bestehenden Applikationen herstellen, und
- Anwendungsmodulen, die (mindestens eines Teils) einer Rolle einer beteiligten Person im Gebäudelebenszyklus entsprechen.

Anwendungsmodule werden in der Regel von der Person, die im jeweiligen Projekt die dem Modul entsprechende Rolle innehat, interaktiv bedient. Aus Gründen der Haftung und der Akzeptanz eher technisch als praktisch vorstellbar ist der Einsatz nichtinteraktiver Anwendungsmodule; die Grenze zwischen Modulen als reine Eingabe- und Datenverteilungshilfe über assistierende bis zu vollautomatischen System ist fließend und kann projektspezifisch gewählt werden, jedoch müssen Änderungen in der Implementierung der Software durchgeführt werden. Abschnitt 6.1 ("Fehldaten-Prototyp", Seite 83) schildert die Anwendungsmodule des Prototyps.

5.5.1 Template-Modul

Das Template-Modul dient der Vereinfachung der Implementierung, indem es viele der Anforderungen aus Kapitel 3 ("Bauwerkslebenszyklus-Anforderungen", Seite 21) erfüllt, ohne daß diese jeweils neu definiert werden müssen. Die übrigen Anforderungen sollen von Systemmodulen erfüllt werden, so daß sich der Vorgang einer Fehldatensystem-Implementierung auf die Umsetzung der Anwendungsregeln, das heißt der Generierung von Ausgabeattributen aus Eingabeattributen, und der gewünschten Benutzungsschnittstelle beschränkt.

Das Templatemodul implementiert daher zunächst die beschriebenen Protokolle zur Kommunikation und die Benutzungsschnittstelle für den Kommunikationsaufbau, wie sie in den vorangegangenen Abschnitten dieses Kapitels beschrieben sind. Sie bilden die Basis für alle weitergehenden Implementierungseigenschaften.

Entscheidungsdokumentation

Zur Dokumentation von Entscheidungen (Anforderung XIII.) implementiert das Templatemodul die Möglichkeit, eine textuelle Historie zu erzeugen (Bild 5.5: Fehldatenmodul mit lokaler Entscheidungshistorie.). Durch die Protokollierung innerhalb eines Moduls ist eine nahezu beliegig ausführlich Dokumentation des Entscheidungsweges zusätzlich zum Datenzustand, d.h. Entscheidungsergebnis, zur Erfüllung von Anforderung XIV. möglich. Hierbei ist zu berücksichtigen, daß die Historie aufgrund des Datenverteilungskonzeptes jeweils in jedem einzelnen Modul erzeugt werden muß, um die für Entscheidungen wichtige lokale Sicht der Daten zu berücksichtigen.

Die Erklärung eines komplexen Entscheidungsprozesses kann jedoch häufig nur durch gemeinsame Betrachtung mehrerer Logdateien erfolgen. Daher sollte ein separates Werkzeug entwickelt werden, das die Logdateien auswertet. Da dieses Werkzeug nicht auf aktuelle Daten im Fehldatensystem, sondern lediglich auf die Logdateien zugreift, muß es nicht in das Fehldatensystem integriert sein[21].

21. Im Gegenteil böte sich eine Implementierung in einer Skriptsprache wie Perl an, wenn mit geringem Aufwand Anpassungen an individuellen Analysebedarf erfüllt werden sollen.

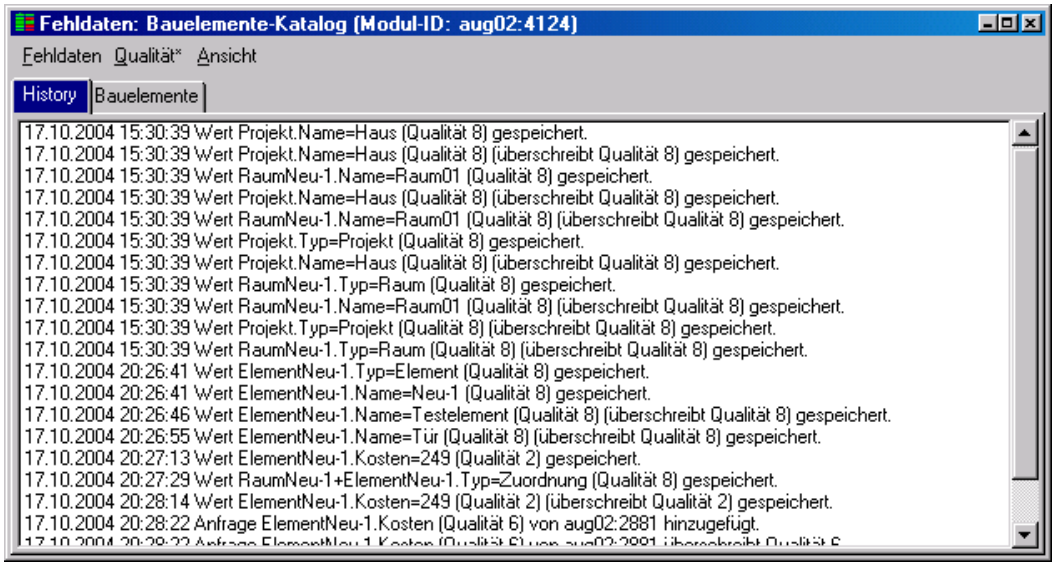

Bild 5.5: Fehldatenmodul mit lokaler Entscheidungshistorie.

Diagnose

Weiterhin stellt das Templatemodul Funktionalität zur Verfügung, um die Diagnose des Modulverhaltens im Fehldatensystem zu erleichern. Diese umfaßt

- eine Modulübersicht mit der Möglichkeit, Ein-/Ausgabeinformationen über Module oder Attribute anzeigen zu lassen oder manuell die Verbindung zu anderen Modulen herzustellen
- eine Datenübersicht zur Anzeige, persistenten Speicherung und Wiederherstellung von Datenwerten und -anfragen
- ein Kommunikationslogbuch, das die Weitergabe von Daten durch die verschiedenen Schichten und zwischen den Modulen protokolliert. (Bild 5.6: Fehldatenmodul mit Kommunikations-Logbuch.)

5.5.2 Systemmodul Routing

Routing zur Laststeuerung

Routingmodule arbeiten als Kommunikationsserver. Sie sammeln alle Datenanforderung topologisch naher Module und kommunizieren diese zu anderen Routingmodulen. Kommunikationsseitig zerlegt dies eine großes Fehldatensystem in mehrere kleine, ist jedoch für die beteiligten Softwaremodule transparent; diese verstehen das gesamte Fehldatensystem außerhalb des eigenen Routingmoduls als ein Modul, was den Kommunikationsaufwand erheblich reduziert oder auf Rechner mit besonders schneller Netzwerkanbindung konzentrieren kann. Faktisch ist die beschriebene Funktionsweise von Routingmodulen vergleichbar mit der einer Middleware.

Auf der Sitzungsebene funktionieren Routingmodule wie folgt: Wenn das erste gestartete Modul als Router fungiert, wird es auf Anfrage anderer Module nur sich selbst zur Kommunikation nennen. Alle nachfolgenden Module kommunizieren entweder ausschließlich mit dem Router oder aber mit den "in der Nähe" verfüg-

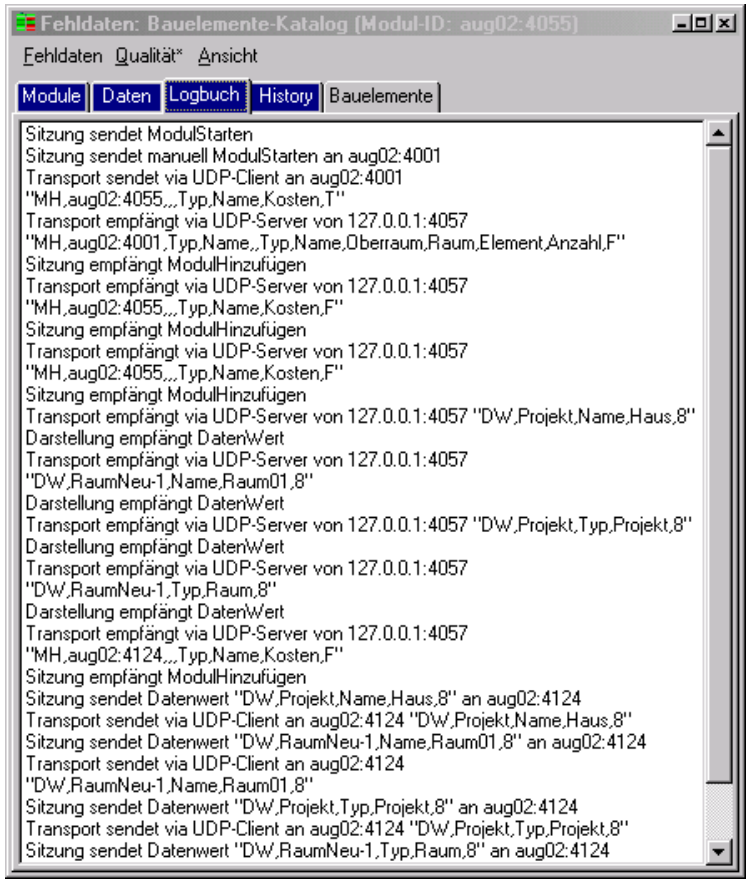

Bild 5.6: Fehldatenmodul mit Kommunikations-Logbuch.

Die blau hinterlegten Schaltflächen öffnen (von links nach rechts) die Modulabfrage, die Datenabfrage, das Kommunikationslog und die Entscheidungshistorie.

baren Modulen und mit dem Router. Routingmodule unterscheiden sich von nichtroutenden Modulen also dadurch, daß sie auf Anfrage nicht die Liste aller ihnen bekannten Module weitergeben, sondern nur sich selbst nennen. Sie interessieren sich im einfachsten Fall pauschal für alle Daten, können dies zur Reduktion des Netzwerkverkehrs bei entsprechender Programmierung aber auch auf die Daten für die ihnen angeschlossenen Module beschränken. Insofern verändern Routingmodule die Anwendung der in dieser Arbeit beschriebenen Protokolle in einer zum Fehldatensystem kompatiblen Weise.

Routingmodule sind transparent einsetzbar: Weder muß ein Fehldatensystem Routingmodule einsetzen noch ändert sich die Arbeit mit dem System hierdurch. Lediglich die Lastverteilung der Kommunikation wird durch Routingmodule beeinflußt.

Routing zur Netztrennung

Beim derzeitigen Systemaufbau können zwei Fehldatensysteme leicht vollständig verschmolzen werden. Sind jedoch zwei Fehldatennetze aus jeweils mehreren Modulen über eine gemeinsam genutzte Einwahlverbindung verschmolzen worden, würde die Trennung der Netze nur möglich sein, wenn alle Module eines der beiden Netze beendet und anschließend neu gestartet werden, da nur dann die nöti-

gen "ModulLöschen"-Botschaften versendet würden - ein in der Praxis unzumutbarer Vorgang.

Hier hilft ein Systemmodul, das die Topologie des Gesamtnetzes kennt, um dieses durch gezielte "ModulLöschen"-Botschaften wieder in zwei Netze zu trennen. Dies erfüllt ein Routingmodul durch die oben beschriebene Protokollmanipulation. Wenn dies vor der physikalischen Trennung der Netze geschieht, reicht ein Modul im Gesamtsystem, um die Trennung von beiden Seiten zu veranlassen. Kann die Netzwerkverbindung jederzeit ausfallen, läßt sie sich durch den Einsatz zweier Routingmodule absichern, die bei Ausfall lediglich beendet werden müssen, um den überflüssigen Datenverkehr zu unterbinden.

5.5.3 Systemmodul Datenspeicherung

Da die Technik der effizienten Datensicherung nicht Kernthema dieser Arbeit und für die Validierung des Fehldatenmodells nicht erforderlich ist, wird die Persistenz nicht weiter vertieft. Für die Testimplementierung wird angenommen, daß das Gesamtsystem eine ausreichende Größe besitzt, um alle Daten zu jedem Zeitpunkt an mindestens zwei Stellen vorzuhalten. Dies beeiträchtigt die Allgemeinheit des Systems nicht, da sich dieser Zustand jederzeit mit zwei Instanzen eines Datenhaltungsmoduls erreichen läßt, das

- nicht interaktiv ist,
- jedes Attribut als Eingabeinformation anfordert,
- auf Anforderung jede bekannte Information sendet,
- und dessen Funktionalität in der persistenten Speicherung der empfangenen Informationen liegt.

Ein Modul zur Datenspeicherung könnte alle Daten und Anfragen im Hauptspeicher (bzw. virtuellen Speicher) vorhalten, jedoch wäre dies bei heute üblichem Speicherausbau und realen Anwendungsszenarien nicht oder nur mit signifikanten Leistungseinbußen umsetzbar, da ein Datenspeicherungsmodul alle (nicht nur eine zur lokalen Bearbeitung relevante Auswahl) Daten bereithalten müßte. Es kann diese Daten besser in einer Datenbank ablegen, so daß jederzeit der aktuelle Stand oder sogar eine Historie der Datenzustände zu erhalten ist. Hierfür bietet sich eine relationale eher als eine objektorientierte oder objektrelationale Datenbank an, da implementierungsseitig das objektorientierte Paradigma wegen der Datengranularität und Schemadynamik aufgelöst ist. Im Falle der Speicherung der kompletten oder selektiven Historie des Ablaufs erlaubt ein solches Modul auch eine konsolidierte Form der Protokollierung von Arbeitsabläufen, wenn Effekte aus Netzwerklaufzeiten und nicht bestehenden Verbindungen vernachlässigbar sind (etwa weil ausschließlich in einem lokalen Netzwerk gearbeitet wird).

Für den Zugriff auf alle Attribute wird ein reservierter Attributbezeichner '*' eingeführt, der analog zum "Joker" in der Kommandoschnittstelle gängigen Betriebssystemen eingesetzt wird. Meldet ein Modul '*' als Eingabeattribut, wird es über alle Attribute benachrichtigt, unabhängig davon, ob diese zur Implementierungs-

zeit des Moduls bereits bekannt waren. Mit '*' als Ausgabeattribut meldet es Interesse an allen Anfragen an.

Jede solche Form eines Datenspeicherungsmoduls erfüllt Anforderung III.

Umsetzung im Prototyp

Bereits im Prototyp des Fehldatensystems ist ein einfaches Persistenzmodul enthalten, das mit den unter Windows als Konfigurationsdateien ("ini-Dateien") üblichen Strukturen arbeitet und diese interaktiv speichern und laden kann.

Zusätzlich enthält die Darstellungsschicht des Prototypen einige kommentierte Zeilen, die einfach die automatische Speicherung beim Beenden des Moduls und das Laden nach Modulstart erlauben würden und damit eine einfache Form lokaler Persistenz auf Modulebene umsetzen können.

Mögliche Erweiterung

Von der Art der persistenten Speicherung hängt der Funktionsumfang dieses Moduls wesentlich ab: Wenn anstelle des jeweils aktuellen Datenwertes der Zeitpunkt der Veränderung mit dem jeweils neuen Wert abgelegt wird, ist - transparent für das Gesamtsystem - ein "Rückspulen" auf jeden Datenstand in der Vergangenheit möglich. Dies ist hilfreich einerseits für die Entscheidungsdokumentation, andererseits können damit Prognosen der Vergangenheit mit später ermittelten, realen Zahlen verglichen werden, was zur Verifikation von Anwendungsmodulen im Fehldatensystem beiträgt.

Während das "Rückspulen" auf der Daten-Zeitachse konzeptionell etwa in [Hovestadt 1994] bereits modelliert ist und das ArchE-Gesamtsystem entscheidend prägte, hätte die Umsetzung im Persistenzmodul des Fehldatensystems den Vorteil, für die beteiligten Module vollkommen transparent, das heißt ohne Eingriffe in die Modullogik umsetzbar zu sein - zur Laufzeit wird den Modulen lediglich ein anderer Datenstand übermittelt, während die Information zur Daten-Zeitachse ausschließlich im Persistenzmodul vorliegt[22].

5.5.4 Schnittstellenmodule

Für viele spezifische Anwendungsfälle im Bauwesen existieren Softwarelösungen, die wegen ihrer Komplexität in Logik oder Benutzungsschnittstelle nicht sinnvoll in kurzer Zeit durch ein originäres Fehldatenmodul zu ersetzen sind. Dazu zählen CAD-Systeme ebenso wie Simulations- oder Virtual-Reality-Umgebungen.

Diese "Wirtsapplikationen" bieten häufig Datenaustausch-Schnittstellen, die für die Anbindung an das Fehldatensystem genutzt werden können. Dann ist "nur" ein Fehldatenmodul nötig, das Daten einerseits mit der Wirtsapplikation, andererseits mit dem Fehldatensystem austauscht. Je nach Art der Datenaustausch-Schnittstelle ergeben sich möglicherweise folgende Einschränkungen:

22. Die Information zum Zeitpunkt auf der Datenachse kann natürlich vom Persistenzmodul als Datenwert bereitgestellt werden, sofern dies für die Anwendung sinnvoll ist.

- Üblicherweise sind Datenaustausch-Schnittstellen nicht für laufende Kommunikation ausgelegt, sondern übermitteln situativ Datenmengen. Dies erfordert häufig, daß die bearbeitende Person den Export bzw. Import explizit auslöst, so daß faktisch eine Art "Offline-Datenbearbeitung" stattfindet [Arnold 2000]. Daraus können die bei Offline-Bearbeitung üblichen Inkonsistenzen entstehen, die über Versionenkonsilidierung gelöst werden müssen.
- Die Übermittlung von Anfragen als Kern des Fehldatensystems ist in keinem bekannten System implementiert.

Wegen dieser Einschränkungen ist ein allgemeingültiges Urteil über Sinn oder Unsinn solcher Implementierung nicht möglich. Bemühungen zu einheitlichen Schnittstellen (etwa von CAx-Systemen [Kilb et al. 1998]) können dies rasch ändern.

5.6 Erfüllung der Anforderungen

Das Datenspeicherungsmodul erfüllt folgende Anforderung:

III. Das Modell muß die persistente Datenspeicherung erlauben.

Das Templatemodul erfüllt mit seiner Logdatei-Funktion diese Anforderungen:

XIII. Das Modell muß relevante Entscheidungen dokumentieren, das heißt speichern und abrufbar halten.
XIV. Das Modell sollte die Abhängigkeiten dieser Entscheidungen mit dokumentieren, um die Rückverfolgbarkeit und Konsistenzsicherung zu erleichtern.

Damit ist in Verbindung mit den durch die Fehldaten-Modellierung umgesetzten Anforderungen aus Abschnitt 4.7 ("Erfüllung der Anforderungen", Seite 55) der Kanon der Anforderungen erfüllt und das Fehldatensystem prinzipiell einsetzbar, wobei konkrete Anwendungen durch geeignete Schnittstellenmodule oder neu zu entwickelnde Applikationen abgedeckt werden.

6 Validierung

6.1 Fehldaten-Prototyp

Dieser Abschnitt skizziert die prototypischen Implementierung eines Fehldatensystems. Dieser Prototyp dient im Wesentlichen der Überprüfung der technischen Kommunikation und des Kollaborationsverfahrens anhand eines Szenarios aus der Bauwelt, so daß für potentielle Anwender die Vergleichbarkeit des Fehldatensystems mit bestehenden Systemen gegeben sein soll.

Dabei wurde auf das Raumbuch-Konzept zurückgegriffen, da dieses immer wieder Gegenstand der Forschung am Institut für Industrielle Bauproduktion ist. Der erste Kontakt mit diesem Konzept entstand für den Autor durch [Dingler 1999]. Der vorliegende Prototyp beruht auf dem Anwendungsfall und den Daten aus [Juister 2004]. Darin wird zum einen das Raumbuch auf die Denkmalpflege angewandt, die stärker als andere Szenarien auf der Bestandsanalyse aufsetzt, so daß die Falle der Beschränkung auf eine einzelne Lebensphase vermieden wird. Zum anderen setzt diese Arbeit mit einer Weboberfläche die nach heutigem Stand modernste Benutzungskonzeption ein, die technisch üblich ist, und bietet so einen besonders interessanten Vergleich zum Fehldatensystem.

Die Daten in [Juister 2004] sind in einer relationalen Datenbank erfaßt. Dabei werden folgende Entitäten modelliert:

- Bearbeiter: Diese entfallen konzeptbedingt im Fehldatensystem und werden durch die in Modulen implizit definierten Rollen ersetzt
- Objekt, Geschoß, Raum: Diese festgelegte, aber im Bauwesen übliche Hierarchie zur räumlichen Strukturierung wird mit unterschiedlichen Attributen versehen. Der Fehldaten-Prototyp gibt dies zugunsten einer flexiblen Hierarchie aus Zonen auf, die eine Verallgemeinerung dieser Struktur darstellt und die zugrunde liegenden Daten damit voll abbilden kann[23].
- Abbildungen: Diese für den realen Einsatz, speziell bei der Bestandsbeurteilung, sehr nützliche Zugabe entfällt im Fehldaten-Prototyp, da sie nicht zur Konzeptbeurteilung beiträgt.
- Elementkatalog: Dieser wird in der Arbeit von Juister als gegeben angenommen, im Fehldatenprototyp wird seine Bearbeitung als eigene Rolle verstanden und als Modul repräsentiert.
- Bauteile (Boden, Decke, Wand, Fenster und Tür): Diese Bauteile werden im Fehldaten-Prototyp entsprechend abgebildet.
- Beläge, Oberflächen und Leisten: Sie beschreiben die Bauteilbeschaffenheit in oberster Schicht und werden auf Attribute der Bauteile abgebildet

23. Diese flexible Hierarchie ließe sich selbstverständlich rückwärts auf ein webbasiertes oder konventionell gestaltetes Raumbuch übertragen, hängt also nicht konzeptionell mit dem Fehldatensystem zusammen.

- DIN 277 und Bauzustand-Kategorien: Diese bezeichnen letztlich erlaubte Werte zu bestimmten Attributen, ohne weitere Informationen hinzuzufügen. Auf sie verzichtet der Fehldaten-Prototyp; sie können eingabeseitig (z.B. in Auswahlfelder) hinterlegt sein.

Mit einem Modul "Raumbuch" ist damit die Repräsentation eines entsprechenden Datenbestandes im Fehldatensystem möglich. Die Erweiterung um den Elementkatalog bietet erste Wechselwirkungen zwischen zwei Rollen. Zusätzlich implementiert der Prototyp ein Modul zur Projektkostenermittlung, das auf Elementkatalog und Raumbuch (das die Instanzen der Elemente beschreibt) zurückgreift. Die Kosten wurden dabei fiktiv eingegeben, da entsprechende Daten nicht zur Verfügung standen.

Zweck des Prototyps

Der Prototyp demonstriert damit insbesondere die Implementierung folgender typischer Fälle im Fehldatensystem:

- die Umsetzung einer flexiblen Hierarchie von Containerobjekten, die im Raumbuch eine Zone abbilden und die wiederum Objekte (weitere Zonen und Bauelemente) enthalten können.
- die freie Definition von Attributen zur Laufzeit, die im Fehldatensystem (im Unterschied zu anderen Applikationen) durch die attributive Datengranularität nicht nur beschreibend, sondern in anderen Modulen zur automatischen Weiterverarbeitung genutzt werden können.
- Die Datenanfrage und -auswertung von solchen Attributen, hier der Bauteilkosten.

6.1.1 Anwendungsmodul Bauelemente

Das Bauelemente-Modul implementiert Objekte, die Zonen zugeordnet werden können. Neben Bauelementen können auf dieser Basis auch Mobiliar oder Einbauten wie Treppen erfaßt werden.

Im Prototyp des Fehldatensystems werden Bauelemente benutzt, um sie mit Kosten zu versehen und Zonen des Raumbuchs zuzuordnen. Zur Annäherung an den Raumbuch-Prototyp aus [Juister 2004] werden hierzu Typangaben und Kostengruppen erfaßt, die in einer entsprechen erweiterten Benutzungsschnittstelle etwa zur Filterung der Informationen eingesetzt werden können.

Für die Zustände "leichte Abnutzung" und "größere Abnutzung" werden zu jedem Modul Renovierungskosten erfaßt. Weiterhin stellt das Modul Felder für Einbau- und Rückbaukosten der Elemente bereit (Bild 6.1: Bauelemente-Modul des Prototyps.).

Dieses Modul enthält keine Logik zur Verarbeitung der eingegebenen Daten, sondern stellt diese lediglich anderen Modulen bereit.

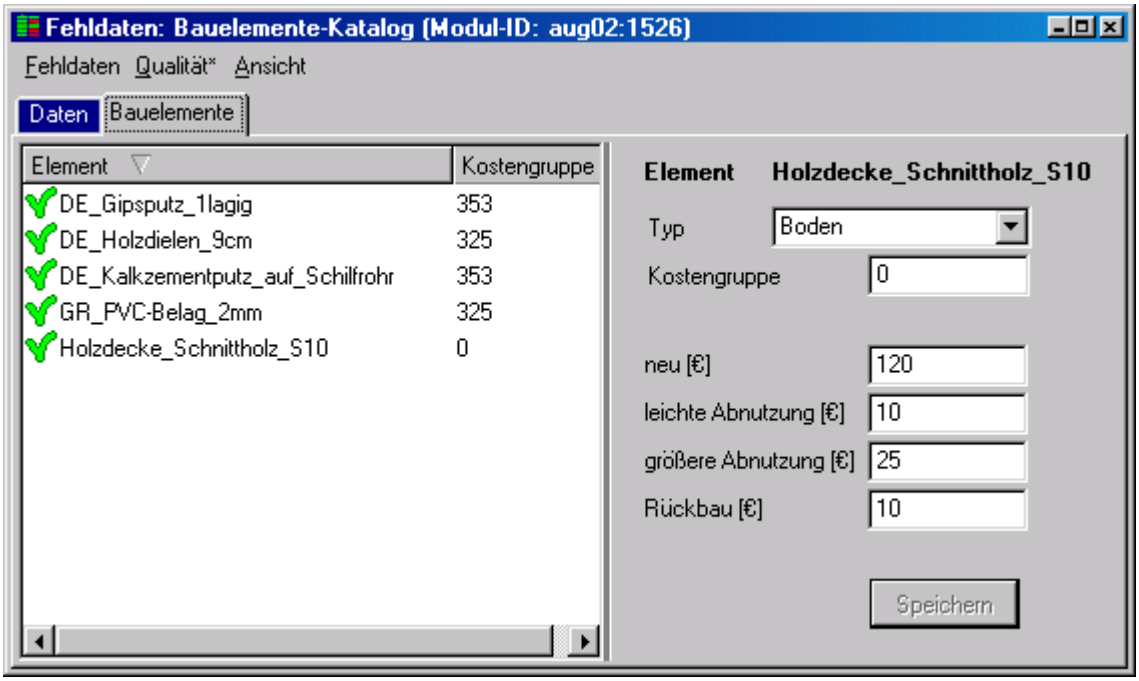

Bild 6.1: Bauelemente-Modul des Prototyps.

6.1.2 Anwendungsmodul Raumbuch

Das Raumbuchmodul erlaubt, ein Bauobjekt in Zonen zu strukturieren. Diese können Gebäudeteilen, Räumen oder einem Raumteil entsprechen und beliebig tief strukturiert (geschachtelt) werden: So kann ein Museumsgebäude etwa zunächst in einen öffentlichen und einen nichtöffentlichen Bereich, diese dann in verschiedene Räume unterteilt werden. Für größere offene Räume, die mehrere Funktionen kombinieren, (wie ein Foyer mit Kassen- und Garderobenzone oder Restaurants) bietet sich eine weitere Unterteilung an. Diesen Zonen lassen sich Bauelemente und weitere Eigenschaften zuordnen.

Im konkreten Anwendungsfall des Prototyps auf das Bauernhaus aus [Juister 2004] bildet das Anwendungsmodul Raumbuch die Struktur aus Objekt, Geschoß und Raum auf eine dreistufige Hierarchie ab, der Bauelemente zugewiesen werden. Diese Bauelemente müssen im Bauelement-Modul angelegt, aber nicht mit allen Informationen (etwa Preisen) erfaßt sein. Auch nachträgliche Umbenennungen der Bauelemente werden korrekt in das Raumbuch übernommen.

Die Bauelemente werden mit Angaben zu Alter und Zustand versehen (Bild 6.2: Raumbuch-Modul des Prototyps.). Im Gegensatz zu den zugrundeliegenden Daten wird nicht der Zustand zum Erfassungszeitpunkt dokumentiert, sondern für die üblichen Zustände "leichte Abnutzung" und "größere Abnutzung" der (geschätzte oder bekannte) vergangene oder voraussichtliche zukünftige Zeitpunkt, ab dem dieser Zustand gilt. Der Zustand "Ende der Lebensdauer" wird dabei durch "Erneuerungsbedarf" ersetzt, was von "Rückbau" unterschieden wird. Der Anfangszeitpunkt für den Zustand "gut" entspricht dabei dem Einbauzeitpunkt.

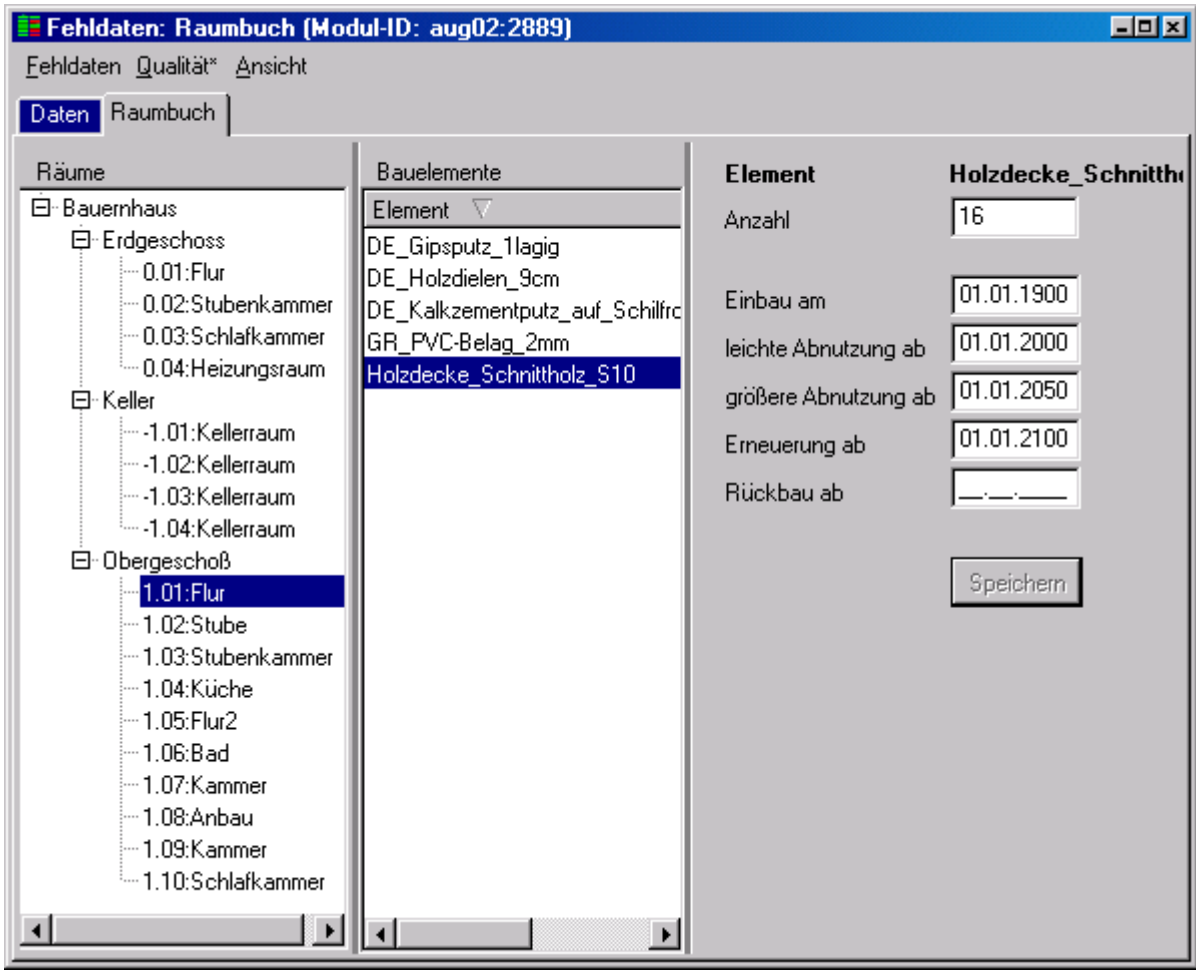

Bild 6.2: Raumbuch-Modul des Prototyps.

Umsetzung des Element-Lebenszyklus

Damit kann der komplette Bauelement-Lebenszyklus dokumentiert werden. Liegt der Einbauzeitpunkt in der Zukunft, ist das entsprechende Element im Rahmen einer Umbaumaßnahme geplant. Ist das Datum für den "Erneuerungsbedarf" gesetzt, soll das Bauelement nach dem Ende seiner Lebensdauer durch ein entsprechendes ersetzt werden. Wenn stattdessen das Datum für "Rückbau" gesetzt ist, soll das Bauelement zum entsprechenden Zeitpunkt aus dem Bestand entfernt werden; ein eventueller Ersatz wird durch ein oder mehrere andere Elemente mit entsprechendem Einbauzeitpunkt repräsentiert.

Daß der Wechsel in einen schlechteren als den heutigen Zustand dabei geschätzt wird, wird dabei über die Datenqualitäten dokumentiert. Nach Renovierungs- oder Umbaumaßnahmen oder in projektspezifisch zu bewertenden Zeitabständen sollten daher die Zeitangaben für die Zustände überprüft und editiert werden.

6.1.3 Anwendungsmodul Projektkosten

Im Projektkostenmodul können Kosten für das Gesamtprojekt oder bestimmte Teile durch Datenaggregation ermittelt werden. Sofern diese noch nicht erfaßt sind, werden sie durch Datenanfrage ermittelt. Dieses Modul demonstriert so das Zusammenwirken der im Prototyp verhandenen Informationen und den Ablauf der Kommunikation.

Auf Basis der Zustandsinformation aus dem Raumbuch und der Kosteninformation aus dem Bauelement-Modul ist die Kalkulation von Projektkosten für die Renovierung oder den Umbau an jedem beliebigen Stichtag unter Berücksichtigung des (faktischen oder prognostizierten) Bestandes und Bestandzustands ermittelt werden (Bild 6.3: Projektkosten-Modul des Prototyps.). Die Art der Einrechnung jedes Elementes ergibt sich logisch aus Istzeitpunkt, Projektstichtag und den Zustandsübergängen der Elemente (Bild 6.4: Mögliche Elementlebensdauern bezüglich Istzeitpunkt und Projektstichtag.):

E1 Liegt der Rückbauzeitpunkt eines Bauelementes in der Vergangenheit, wird das Element nicht mehr für die Projektkosten berücksichtigt.

E2 Liegt der Erstellzeitpunkt eines Elementes in der Vergangenheit und ist der Rückbauzeitpunkt nach dem Projektstichtag oder nicht angegeben, ergibt sich der Zustand des Elementes aus den Daten für die Zustandsübergänge. Der Preis für die Renovierungsmaßnahme ergibt sich aus dem Bauelemen-

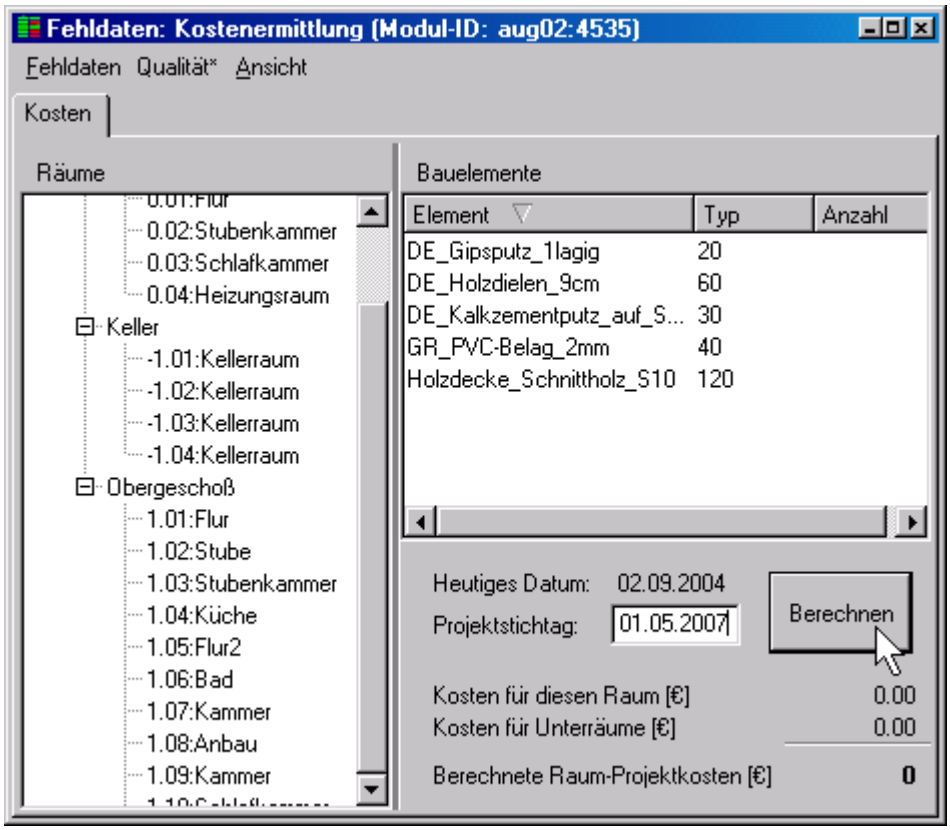

Bild 6.3: Projektkosten-Modul des Prototyps.

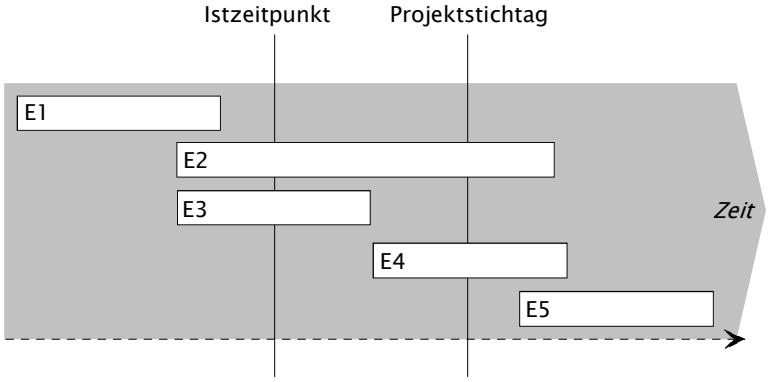

Bild 6.4: Mögliche Elementlebensdauern bezüglich Istzeitpunkt und Projektstichtag.

tekatalog, wobei Elemente, die das Ende der Lebensdauer zum Projektstichtag erreicht haben, mit Rückbau- plus Neubaukosten in die Kalkulation eingehen.

E3 Liegt der Erstellzeitpunkt eines Elementes in der Vergangenheit und der Rückbauzeitpunkt in der Zukunft, aber vor dem Projektstichtag, wird das Element mit den Rückbaukosten eingerechnet.

E4 Liegt der Erstellzeitpunkt in der Zukunft, aber vor dem Projektstichtag, wird das Element mit den Erstellkosten eingerechnet.

E5 Liegt der Erstellzeitpunkt eines Elementes noch nach dem Projektzeitpunkt in der Zukunft, wird das Element nicht berücksichtigt.

Da das System in der Konfiguration des Prototyps nur den jeweils aktuellen Datenwert speichert, wird ein renoviertes Bestandselement im Raumbuch neue Daten für die Zustandsübergänge erhalten. Aussagen über Projektkosten in der Vergangenheit sind so nicht möglich. Hierfür müßte entweder eine Historie vergangener Daten mit Änderungszeitpunkt gespeichert werden oder eine Liste beliebiger Zustandsübergänge verwaltet werden, deren Werte nicht verändert, sondern nur ergänzt werden.

6.2 Einordnung des Prototyps

Auch wenn die Benutzungsschnittstelle und die Attributmenge sicherlich erweitert werden können, erlaubt der Prototyp hinsichtlich des Zusammenspiels der verschiedenen durch Module repräsentierten Rollen ein erstes Urteil. Die Umsetzung der Qualitäten hat die Implementierung in Datenhaltung und Visualisierung wesentlich komplexer gestaltet, erlaubt aber im Zusammenhang mit einfachen Zeitangaben in Raumbuch- und Projektkostenmodul bereits einen ersten Eindruck, wie auch Simulationen ablaufen könnten. Auch haben die nahtlose Integration mit dem Zustand fehlender Daten und die Ersparnis expliziter Kommunikation zur

Anforderung "besserer" Daten einen so hohen funktionellen Nutzen, daß dies gerechtfertigt scheint. Die Mensch-Maschine-Schnittstelle speziell für die Arbeit mit Qualitätsstufen wird allerdings bei hohen Datenvolumina tiefergehende Untersuchungen erfordern.

Dies führte auch zu Beschränkungen der Protokolle, die im praktischen Betrieb hinderlich sein können. Für die tägliche Nutzung mag man eine Mischform vorziehen, die zum Beispiel konkurrierende Wertesetzungen oder Netzwerkunzulänglichkeiten um den Preis einer komplexeren Kommunikation aufdeckt. Hier sind weitere Untersuchungen zur Verteilung und Modelldynamik erforderlich.

Eigenschaften

Die prototypische Implementierung zeigt dabei einige interessante Eigenschaften auf:

- Es ist grundsätzlich nicht nötig, die Personen, mit denen kooperiert wird, zu kennen.
- Ebenso ist es bei generierten Informationen nicht nötig, den Generierungsweg zu kennen oder nachzuvollziehen, sofern man (eine durchaus wichtige Einschränkung) dem Programmalgorithmus vertraut.

In beiden Aspekten beschleunigt das Fehldatensystem die Kooperation durch schnellere Kommunikation der Beteiligten und reduziert die notwendige Meta-Arbeit, das heißt die nur mittelbar zum Ergebnis beitragenden Tatigkeiten wie Kontaktaufnahme und Recherche.

- Falls das Fehldatensystem Informationen nicht beschaffen kann, erfolgen Recherche bzw. Generierung auf konventionellem Weg, bevor die Information im Fehldaten-System erfaßt werden.
- Es findet keine Umlenkung von Kommunikationsprozessen statt, die nicht direkt datenbezogen sind: Strategische Treffen bzw. Entscheidungen werden nicht “virtualisiert”.

Diese Eigenschaften resultieren direkt aus dem Verzicht auf die Einführung einer Metaebene zur Kooperationsmodellierung, da das Fehldatensystem allein Datenwerte und -anfragen kommuniziert und keine Prozeßmodellierung vorsieht.

Arbeit mit Fehldaten ähnelt manueller Arbeit

Üblicherweise wird eine Computerunterstützung einen Prozeß nicht nur im Rechner abbilden, sondern in der Folge auch verändern, wie dies bei Fertigungsprozessen bereits zu beobachten ist. Selbst unter der idealen Annahme, daß psychologische Effekte wie Berührungsängste mit neuen Geräten und technische Probleme nicht auftreten, ist die Arbeitseffizienz und damit der Nutzen eines Systems wesentlich von der Nähe zur bisherigen, geübten und üblichen Arbeitsweise abhängig.

Zusammenfassend weist das System eine Charakteristik auf, die es mit der üblichen Arbeitsweise vergleichbar macht. Unter den analysierten Anwendungsfällen mit dem Fehldatensystem war genau dann eine Abweichung vom direkten Weg zum Ziel nötig, wenn sie auch bei konventioneller Arbeitsweise nötig gewesen wäre: Bei fehlendenden Informationen und nicht vollständig abgedeckten Kompetenzen. In den anderen Fällen ist die Arbeit mit dem Fehldatensystem schneller, da

sie die Kommunikation beschleunigt, ohne Modellierungsarbeiten auf der Metaebene vorauszusetzen. Dies läßt auf eine kurze Einarbeitungszeit und gute Eignung für den praktischen Betrieb hoffen.

6.3 Grenzen

6.3.1 Organisatorische Implikationen

Das Fehldatenmodell bezieht seine Flexibilität aus der Aufteilung komplexer Aufgaben in Teilaufgaben (Rollen). Während bei heute üblicher Software eher mit jeder Überarbeitung weitere Funktionen zugefügt werden, um zusätzliche Kunden zu gewinnen, wird im Fehldatensystem eine funktionale Erweiterung häufiger in Form eines zusätzlichen Moduls sinnvoll umzusetzen sein. Entsprechend der Wirklichkeit kann eine Person mehrere Rollen innehaben und damit mehrere Module benötigen, kaum wird jedoch ein Modul mehrere Rollen übernehmen.

Teil dieses Ansatzes ist, daß einzelne, überschaubare Rollen sich wenig von Projekt zu Projekt ändern. Daher spielte die Parameterisierung der erfaßten Information keine Rolle, wegen der Wechselwirkungen zwischen Eingabefenstern und Daten und der für den Prototyp gewünschten Übersichtlichkeit sind alle datenrelevanten Implikationen direkt in der Software implementiert und nicht etwa durch Optionen steuerbar. Die Abläufe wiederum ergeben sich aus den Daten selbst, in anderen Worten: sie hängen von den erfaßten Daten und ihrer Benennung ab.

Sollte sich, etwa aus der Erweiterung des Blickfeldes bei der Implementierung neuer Module, die Notwendigkeit zu Änderungen an bestehenden Modulen ergeben, erfordert dies eine Implementierungsänderung. Dies ist unangenehm, weil es mehr Fachkenntnis und Zeit verlangt als die Änderung von möglichen "Projektoptionen". Sollte dieser Fall zur Regel häufiger auftreten, böte sich eine Überarbeitung des Templatemoduls anbieten, um die Konfigurierbarkeit zu erleichtern. Recht schnell sind Ein- und Ausgabeattribute parameterisierbar und damit laufzeitkonfigurierbar zu implementieren. Formulargeneratoren in Datenbankapplikationen und Parser wären Vorbilder für die (deutlich komplexere) Umsetzung einer parameterisierten Eingabe und Verarbeitung.

Prinzipiell ist damit der Nachteil der notwendigen Implementierungsänderung zu weiten Teilen vermeidbar. Praktisch wird jedoch der nicht unerhebliche Implementierungsaufwand erst gerechtfertigt, wenn er die zu erwartende Zeit für Implementierungsanpassungen unterschreitet.

6.3.2 Skalierbarkeit

Um das verteilte Konzept der Fehldatenmodellierung in der Implementierung nachzuweisen und in der Implementierung dennoch sichtbar zu halten, basiert diese Arbeit auf einem recht einfachen Kommunikationsprotokoll. Dieses Protokoll kommuniziert mit allen aktiven Modulen des Fehldatensystems direkt, muß also im Extremfall jede Datenändeurng jedem Modul übermitteln. Dies ist in großen Systemen ineffizient, da es eine Kommunikation innerhalb eines vollständigen Netzes von Modulen ausführt: Wenn 20 Module an einem bestimmten Attribut interessiert sind, sendet das erzeugende Modul 20 Nachrichten auf direkt Wege. Dies kann bei langsamer Netzwerkanbindung oder bei langen Wegen im Netzwerk langsamer sein als die klassische Client-Server-Struktur.

Optimierung durch Routing

Hier können Routingmodule, wie sie in Abschnitt 5.5.2 ("Systemmodul Routing", Seite 78) beschrieben sind, helfen: Das erzeugende Modul sendet idealerweise nur eine Botschaft an das Routingmodul, das die 20 Module versorgt, die sich in größerer topologischer Nähe des Routingmoduls befinden.

Optimierung durch Protokollerweiterung

Auch ist die Optimierung der Kommunikation über die Einbeziehung des Absenders in das Protokoll möglich: Während derzeit für jeden Wert nur eine Anfrage gespeichert wird, und ermittelte Werte an alle potentiell interessierten Module gesendet werden, könnten stattdessen Anfragen mit Absender gespeichert werden und Werte nur an diejenigen Module versendet werden, die angefragt haben.

6.3.3 Datensicherheit und -konsistenz

Die Architektur des Internet geht davon aus, daß Rechner bzw. Rechnerverbindungen jederzeit ausfallen können. Desgleichen gilt für das Fehldaten-System. Damit ist es unmöglich, Antwortzeiten oder auch nur die Existenz einer Antwort zu garantieren. Dies ist für eine geschäftliche Kooperation kein wünschenwerter Zustand, andererseits aber bei den heute üblichen auf TCP/IP basierenden Systemen unumgänglich, so daß Ausfälle auf der Applikationsebene bearbeitet werden müssen. Die Systemarchitektur des Fehldatensystems sieht von vorneherein vor, mit nur sporadisch bestehenden Verbindungen zu arbeiten und keine Annahmen über Antwortzeiten zu treffen. Ausfälle führen dadurch nicht zu Fehlerzuständen, werden aber die Weiterverarbeitung von Informationen einschränken, speziell weil an eine Rechnerverbindung zwischen verschiedenen kleinen Unternehmen keine hohen Anforderungen hinsichtlich garantierter Übertragungsleistung oder Stabilität der Verbindungen und der beteiligten Hardware gestellt werden können.

Hier sind Untersuchungen zur Fehlertoleranz nötig. Ein Netzwerkausfall wird bei einem interaktiven Modul durch die bearbeitende Person sicherlich sofort bemerkt. Die Funktionalität der Module sollte dahingehend verbessert werden, daß der solitäre Einsatz sinnvoll möglich ist. Zusätzlich können Konzepte zur Redundanz von nicht-interaktiven Funktionen der Module die Wirkung von Hardware-Ausfällen begrenzen. Ein Verfahren zur Analyse und Bewertung der prozentualen Verfügbarkeit der Maximalfunktion für vorgegebene Netzwerk- und Modulkonfigurationen ist nötig, um auf diesem Gebiet optimale Strategien zu entwickeln.

6.3.4 Verarbeitungspfad, Konkurrenz, Versionen

Da sich das Fehldatensystem bezüglich seiner Verteilung von Modulen streng an das Verteilungskonzept des Internet anlehnt, “erbt” es dessen Eigenschaften. Mithin ist es nicht möglich, Antwortzeiten vorherzubestimmen oder konkrete Verarbeitungspfade a priori zu generieren – jeder Computer und jedes Fehldatenmodul kann sich jederzeit abmelden, so daß ein eben generierter Verarbeitungspfad beim Versuch seiner Anwendung vielleicht nicht mehr existiert. Aus dem letzten Punkt ergibt sich, daß die Verarbeitungsanalyse nur a posteriori möglich ist.

Die fehlende Garantie eines vorbestimmten Verarbeitungspfades wirkt sich einerseit bezüglich der Antwortzeit aus, die im vorangegangenen Abschnitt beschrieben wurde – sie kann sich theoretisch beliebig verlängern. Andererseits können aber verschiedene Verarbeitungswege miteinander konkurrieren, wozu Kommunikationsregeln existieren müssen, nach denen diese Wege zur Laufzeit priorisiert werden und ein abschließendes Verarbeitungsergebnis miteinander aushandeln. Die bestehende Implementierung handelt hier nach der Devise, daß der jüngst empfangene Wert (der obersten empfangenen Qualitätsstufe) gültig ist.

Diese Strategien zur Priorisierung sind existientiell für Funktion und Korrektheit des Fehldaten-Systems und damit eine Achillesferse der Modellierung. Derzeit lassen sich Szenarien finden, in denen zwei getrennte Fehldatensysteme unterschiedliche Werte gleicher Qualitätsstufe für dasselbe Attribut besitzen. Beim Verschmelzen dieser Fehldatensysteme würden durch das wechselseitige Senden der Daten beide Werte durch den jeweils anderen überschrieben, mithin die Werte getauscht. Gewünscht wäre hier die (vorhersagbare) Dominanz eines Wertes anhand einfacher Kriterien – die Alternative der interaktiven Konfliktbewältigung nach dem Muster von Serverreplikationen sind unbefriedigend, so daß dieses Problem immernoch ungelöst ist.

6.3.5 Rückverfolgbarkeit und Zugriffsrechte

Der vollständige Ersatz der Benutzermodellierung durch Rollen, die sich implizit aus den Modulen ergeben, ist ein bewußt radikaler Schnitt. Die Arbeit zeigt, daß zunächst (das heißt in kleinen Projekten) dieses Weniger an zu pflegenden Daten ein Mehr an Nähe zur konventionellen, zielorientierten Arbeit bringt und insgesamt vorteilhaft beurteilt werden kann.

In größeren Projekten ist unter anderem aus Gründen der Haftung und der Qualitätssicherung die Rückverfolgbarkeit nötig, wer welche Bearbeitung durchgeführt hat, da dies zugleich Entscheidungen im Projektverlauf dokumentiert. Dies erfordert jedoch nicht die Wiedereinführung der Personenmodellierung, sondern lediglich die Protokollierung der Entscheidung: Zeitpunkt und IP-Adresse des Rechners reichen aus, um Daten eindeutig einer Person zuzuordnen – wenn die Zugriffsrechte auf Module und Rechner sinnvoll eingesetzt werden und der Zugriff (etwa Login-Zeiten) ebenfalls protokolliert wird. Für diese Funktion erfordert der Fehldatensystem-Prototyp eine Erweiterung um die Speicherung von Zeitpunkt und IP sowie im Vergleich zu anderer Software eine stärkere Zusammenarbeit mit der Sy-

stemadministration. Letzteres läßt sich auf Kosten der konzeptionellen Reinheit natürlich vermeiden, wenn man (ohne Auswirkungen auf die weitere Modellierung) eigene paßwortgeschützte Benutzerkonten einführt.

6.4 Zusammenfassende Bewertung

Diese Arbeit hat als Beitrag zur Grundlagenforschung versucht, eine möglichst geradlinige Umsetzung der grundlegenden Konzepte verteilten Arbeitens zu finden. Im Ergebnis sind die Unterschiede gegenüber herkömmlichen, auch modernen computergestützten Ansätze ablesbar.

Insgesamt bestätigt das vorgelegte Fehldatenmodell die eingangs aufgestellte These, daß ohne explizite Modellierung von Prozeßbeteiligten, Rollen oder Rechten computergestützte kooperative Zusammenarbeit im Bauwesen möglich ist. Mit dem Entfall der Bürokratie einer Kooperationsverwaltung ist Arbeit fast wie an einem Einzelplatzsystem möglich, auch entsprechen die Unwägbarkeiten hinsichtlich Erreichbarkeit und Antwortzeit denen bei manueller Arbeit.

Versuche mit dem Prototyp zeigen so, daß die gewonnene Arbeitsweise in jeder Hinsicht der traditionellen Arbeitsweise mit Tusche und Papier stark entspricht. Diese aus dem verteilten System resultierende Phänomenologie ist das zentrale Ergebnis dieser Arbeit.

Diese Ergebnisse sind ohne eingehende Untersuchungen mit geübten Nutzerinnen und Nutzern aus dem Bauwesen, etwa im Rahmen der Drittmittelforschung, entstanden, die das Bild noch verändern können.

6.5 Ausblick

Erweiterung für praktische Anwendung

Vordringlich für die allgemeine Einsatzfähigkeit ist die Erweiterung um Module, die für den Bauwerkslebenszyklus relevante Information verarbeiten. Die Übernahme von Raumbuch-Daten in eine grafische Planungsumgebung, die Weiterverarbeitung und Aufbereitung von Planungsdaten für das Facility Management, die erleichterte Bauaufnahme durch teilautomatische Generierung von Planungsdaten aus einer sensorisch unterstützten Gebäudeerfassung sind Gebiete, die von großem Interesse für eine Effizienzsteigerung in der Gebäudeproduktion sind.

Diese und ähnliche Funktionen sind wesentlich für jede Gebäudelebenszyklusmodellierung. Während ihrer Implementierung entstehen üblicherweise weitere Erkenntnisse zu Möglichkeiten und Grenzen des zugrundeliegenden Modells, so daß

die Erweiterung des Fehldatensystems um weitere Anwendungsmodule sowie das Generieren von Testfällen für das und mit dem Fehldaten-System die nächsten Schritte bilden sollten.

Bei der Erweiterung werden auch die technischen Aspekte wieder in den Vordergrund rücken, die in Abschnitt 5.4.3 ("Benutzungsschnittstelle", Seite 74) und 6.3.2 ("Skalierbarkeit", Seite 91) beschrieben sind und die menschliche bzw. maschinelle Leistungsfähigkeit des Gesamtsystems entscheidend beeinflussen können.

Dennoch ist die Umsetzung des Fehldatenmodells, die bewußt eine Extremform der kooperativen Arbeit darstellt, nicht als Ersatz für kooperative Systeme anderer Paradigmen gedacht. Aus Sicht des Autors wäre ein Vergleich der Stärken und Schwächen verschiedener Konzepte wünschenswert, wenn mehr Erfahrungen mit dem Fehldatensystem vorliegen, um gegebenenfalls verbesserte Konzepte daraus abzuleiten.

Ein langfristiger Aspekt

Zum theroretischen Verständnis des Problems und damit zur Konzeptverbesserung kann auch ein anderer Ansatz beitragen: Sowohl in der Architektur als auch in der Informationstechnik werden Infrastrukturprobleme behandelt, die aufgrund ihrer Komplexität schwer formalisierbar sind. Auch wenn die Fachleute beider Disziplinen unterschiedliche Sprachen sprechen, so sind doch die Probleme in vielerlei Hinsicht überraschend ähnlich (vergleiche dazu beispielhaft [NN 1990: FiFF]). Ein Gebäude kann man etwa als verteiltes System verstehen, in dem die Räume (Speicher-, Übertragungs-) entsprechen und die Entfernungen den Kommunikationsaufwand abbilden. Die Simulation der Gebäudenutzung durch Menschen etwa hat eine Entsprechung in der Lastverhalten-Simulation für Rechnernetzwerke. Das Problem der Anordnung von Räumen in einem Gebäude hat nicht erst Ähnlichkeit mit der Frage der Stukturierung von Software und ihren Funktionen, seit die Benutzungsoberfächen dreidimensional beziehungsweise in Virtual-Reality-Technik umgesetzt sind. Weitere Beispiele lassen sich finden.

Auch wenn das Bauwesen anders als Informationsinfrastrukturen mit Massenträgkeit und physikalischen Grenzen zu kämpfen hat, gibt es in der Untersuchung der Ähnlichkeiten und Unterschiede vermutlich Lernpotential für beide Seiten – für Gewinne aus der Formalisierung der Problemlösung ebenso wie aus der ganzheitlichen Betrachtung unter Nutzbarkeitsgesichtspunkten. In den Worten von [Rechtin et al. 1997]: *"Today's systems architecting is indeed driven by, and serves, much the same purposes as civil architecture -- to create and build systems too complex to be treated by engeneering analysis alone."*

Auch während der Tätigkeit am ifib und an dieser Arbeit traten diese Parallelen immer wieder zutage – ohne sich jedoch in einer Weise systematisieren zu lassen, die zur Arbeit beigetragen hätte. Hier wäre noch grundlegender anzusetzen: Bei der Suche nach einem formalen Modellierungsverfahren für Infrastrukturen, ihre Kapazität und Nutzung kann die Bedingung, für Bauwesen und Informatik gleichermaßen geeignet zu sein, durch die dann notwendige Abstraktion interessante Aspekte hervorbringen.

Anhang A Kurzanleitung zum Erstellen neuer Module

Dieser Anhang beschreibt die nötigen Schritte, um ein einfaches neues Fehldatenmodul aus dem Templatemodul abzuleiten. Damit soll der Einstieg in die Programmierung erleichtert werden.

Diese Anleitung setzt eine laufende Installation und Grundkenntnisse von Borland Delphi 6 bzw. Borland Kylix sowie die Quelltexte dieser Arbeit voraus.

1. *Kopieren des Projektes*
   Laden Sie das Projekt "FehldatenTemplate.dpr" und speichern Sie diese unter dem gewünschten Projektnamen, etwa "FehldatenPlanung.dpr". Über das Menü "Project / Options..." können Sie Ihrem Modul einen neuen Namen ("Application Settings: Title") geben, der als Fenstertitel verwendet wird.

2. *Kopieren der Anwendungsschicht und des Formulars.*
   Das Projekt enthält die Dateien "SchichtAnwendungTemplate.pas" und "ViewTemplate.pas". Speichern Sie diese Dateien als "SchichtAnwendungPlanung.pas" und "ViewPlanung.pas" (bzw. unter analoger Verwendung des von Ihnen gewählten Projektnamens). Benennen Sie die in diesen Dateien enthaltenen Objekte dieser Dateien mittels des Objektinspektors in "PlanungAnwendungSchicht" und "PlanungForm" um; die Klassennamen werden dabei automatisch angepaßt.

3. *Implementierung der Kommunikation*
   In "PlanungAnwendungSchicht" überschreiben Sie den Methodenaufruf in "Initialisieren", so daß die relevanten Ein- und Ausgabeattribute an die Darstellungsschicht gemeldet werden. Gegebenenfalls erweitern Sie für deren Definition die Datei "Woerterbuch.pas".
   Zusätzlich implementieren Sie hier die nötigen Methoden, die Ihre Benutzungsschnittstelle aufrufen wird, um Daten und Anfragen zu kommunizieren. Diese Methoden senden Datenänderungen und Anfragen über das Fehldatenprotokoll weiter.

4. *Erstellen der Benutzungsschnittstelle*
   In "PlanungForm" können Sie die Karteikarte "Anwendung" zur Plazierung der Elemente Ihrer Benutzungsschnittstelle nutzen. Zu diesen Elementen erzeugen Sie Ereignismethoden ("Events"), die Ihre Methoden aus "PlanungAnwendungSchicht" zur Daten- und Anfragenkommunikation aufrufen. Als Beispiele hierfür können Sie die Quelltexte der Module des Prototyps heranziehen.
   Zusätzlich fügen Sie in "DatenJetztAktualisieren" anstelle des Kommentars Funktionalität ein, die den Bildschirminhalt aktualisiert, nachdem neue Daten und Anfragen vorhanden sind.

5. *Verknüpfung der Anwendungsschicht mit dem Modul*

   Während das geänderte Fenster für die Benutzungsschnittstelle voreingestellt ist und durch die Umbenennung automatisch korrekt verwendet wird, müssen Sie die Anwendungsschicht manuell einbinden.
   Als letzten Schritt ersetzen Sie dazu in der Datei "ViewPlanung" "SchichtAnwendungTemplate" durch "SchichtAnwendungPlanung" sowie "TTemplateAnwwndungSchicht" durch "TPlanungAnwendungSchicht". Die betreffenden zwei Stellen sind mit einem Kommentar "Template ersetzen" versehen.

Anschließend können Sie Ihr Fehldatenmodul compilieren und testen.

Anhang B Datenwörterbuch

Dieser Anhang entspricht der Datei "woerterbuch.pas" aus der Implementierung des Prototypen. Sie umfaßt Konstanten für alle verwendeten Attribute sowie als Kommentare Informationen zu Modulen, die diese Attribute zur Eingabe bzw. Ausgabe verwenden.

```
unit Woerterbuch;
interface

const                                    // EINGABE                    AUSGABE
  // Objekttyp Bauelement
  objElement = 'Element';
  attrTyp = 'Typ';                      // *                         Bauelement
  attrName = 'Name';                    // Kosten,Raumbuch           Bauelement
  attrElementtyp = 'Elementtyp';        // Kosten,Raumbuch           Bauelement
  attrKostengruppe = 'Kostengruppe';    //                           Bauelement
  attrPreisNeu = 'PreisNeu';            // Kosten                    Bauelement
  attrPreisLeicht = 'PreisLeicht';      // Kosten                    Bauelement
  attrPreisGross = 'PreisGross';        // Kosten                    Bauelement
  attrPreisRueckbau = 'PreisRueckbau';  // Kosten                    Bauelement

  // Objekttyp Projekt
  objProjekt = 'Projekt';
  // attrTyp                            // *                        Raumbuch
  // attrName                           // Kosten                   Raumbuch
  attrUnterraumKosten = 'UnterraumKosten';  //                        Kosten

  // Objekttyp Raum
  objRaum = 'Raum';
  // attrTyp                            // *                        Raumbuch
  // attrName                           // Kosten                   Raumbuch
  attrOberraum = 'Oberraum';            // Kosten                   Raumbuch
  attrRaumKosten = 'RaumKosten';        //                           Kosten
  // attrUnterraumKosten                //                           Kosten

  // Objekttyp Bauteil-Raum-Zuordnung
  objZuordnung = 'Zuordnung';
  // attrTyp                            // *                        Raumbuch
  attrRaum = 'Raum';                    // Kosten                   Raumbuch
  attrElement = 'Element';              // Kosten                   Raumbuch
  attrAnzahl = 'Anzahl';                // Kosten                   Raumbuch
  attrDatumEinbau = 'DatumEinbau';      // Kosten                   Raumbuch
  attrDatumLeicht = 'DatumLeicht';      // Kosten                   Raumbuch
  attrDatumGross = 'DatumGross';        // Kosten                   Raumbuch
  attrDatumErneuerung = 'DatumErneuerung';  // Kosten               Raumbuch
  attrDatumRueckbau = 'DatumRueckbau';  // Kosten                   Raumbuch
```

```
(* Objekttyp muß in Objektname kodiert sein, um Eindeutigkeit herzustellten.
   Dies schließt Konflikte durch gleichzeitige Bearbeitung mit gleicher Namensver-
      gabe aber nicht aus.

   Weiterhin sollten alle Ausgabeattribute auch Eingabeattribute sein, um Parallel-
      lauf gleicher Module zu unterstützen.*)

implementation

end.
```

Anhang C Literatur

[Amor et al. 1997]
Amor R.W., Clift M., Scherer R., Katranuschkov P., Turk Z. and Hannus M.: A Framework for Concurrent Engineering - ToCEE.
European Conference on Product Data Technology, PDT Days 1997, CICA, Sophia Antipolis, France, 15-16 April 1997, S. 15-22.
http://wwwcib.bau.tu-dresden.de/sphinx/anonymous?OID=publication.SchererPDTAG97 (11.6.99)

[Archer 1964]
Archer B.: Systematic Methods for Design.
In: Design, 172, 174, 176, 179, 181, 185, 185, 188. 1963-1964

[Arnold 2000]
Arnold F.: MACAO - A Journey into CAx Interoperability and Collaborative Design.
Arbeit im Rahmen des ANICA-Projektes, in: E. Banissi, M. Bannatyne, C. Chen, F. Khosrowshahi, M. Safraz, A. Ursyn (Hrsg.): Proceedings of the 4th IEEE International Conference on Information Visualisation (IV 2000), Collaborative Design Visualisation Session, London, England, 19 -21 July 2000, Published by IEEE Computer Society, pp. 557-562.

[Asimow 1964]
Asimow M.: Introduction to Design.
Englewood Cliffs, N.J. : Prentice-Hall, 1964

[Augenbroe 1995]
Augenbroe G.L.M.: Combine2 - EU DG XII Joule.
TU Delft. Delft 1995. p. 92 Final Report No. Contract JOU2-CT92-0196
siehe auch http://erg.ucd.ie/combine/ (geprüft 17.4.2004)

[Bentley et al. 1997]
Bentley R., Appelt W., Busbach U., Hinrichs E., Kerr D., Sikkel S., Trevor J., Woetzel G.: Basic Support for Cooperative Work on the World Wide Web.
In: International Journal of Human-Computer Studies 46(6): Special issue on Innovative Applications of the World Wide Web, p. 827-846, Academic Press Juni 1997

[Bergmann et al. 2000]
Bergmann, A., Breunig, M., Cremers, A. B., Shumilov, S.: A Component Based, Extensible Software Platform Supporting Interoperability of GIS Applications.
In: A. B. Cremers, K. Greve (Hrsg.): Environmental Information for Planning, Politics and the Public, Proceedings of the 12th International Symposium "Computer Science for Environmental Protection" of Gesellschaft für Informatik (GI), S. 90-102, Bonn, 4.-6. Oktober 2000.

[Beucke 2001]
Beucke, K.: Projektbericht D1 - Strukturierung von Informationen und Prozessen für Revitalisierungsaufgaben.
In: SFB 524 "Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken" - Kolloquium zu den Ergebnissen der 1. Förderperiode. Bauhaus-Universität Weimar; Weimar, 2001

[Bijl 89]
Bijl, A.: Computer discipline and design practice: shaping our future.
Edinburgh University Press, Edinburgh 1989
ISBN 0-85224-644-7, ifib #9097 C 984

[Blodau 1999]
Blodau G.: Erkennung und Behandlung von Kollisionen in der kooperativen Architekturplanung.
Vortrag im Institut für Industrielle Bauproduktion, Universität Karlsruhe, 20. Juli 1999

[Böhms et al. 1994]
Böhms M., Storer G.: *ATLAS: Architecture, Methodology and Tools for Computer-Integrated Large-Scale Engeneering.*
in: Proceedings JSPE-IFIP WG 5.3 Workshop, DIISM'93. Tokyo 1994.
http://www.fcul.research.ec.org/esp-syn/text/7280.html

[Boley 1996]
Boley H.: Austausch und Evolution von Wissen
Vortrag im Rahmen des VEGA-Projektes, Kaiserslautern: Universität 25.11.1996
siehe auch http://www.dfki.uni-kl.de/~vega/Home.html (geprüft 17.4.2004)

[Bundestag 1995]
Deutscher Bundestag (Hrsg.): Handlungsrahmen der Bundesregierung für eine Initiative zum kosten- und flächensparenden Bauen sowie Bericht der Kommission zur Kostensenkung und Verringerung von Vorschriften im Wohnungsbau „Mehr Wohnungen für weniger Geld"
BT-Drucksache 13/2247 vom 29.08.95, Bonn 1995

[Churchman 1967]
Churchman C.: Wicked problems.
In: Management Science Nr. 4, Seite 141-142. 1967

[Dingler 1999]
Dingler F.: Das Raumbuch als Vademecum des Gebäudes - Strukturierung ovn Gebäudeinformationen aus Planung, Ausführung und Nutzung für den Einsatz in DV-gestützten Planungsumgebungen am Beispiel des Raumbuchs.
Diplomarbeit. Karlsruhe: Institut für Industrielle Bauproduktion 1999.

[Donath 2001]
Donath, D.: Projektbericht D2 - Bauplanungsrelevantes digitales Gebäudeaufnahme- und Informationssystem.
In: SFB 524 "Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken" - Kolloquium zu den Ergebnissen der 1. Förderperiode. Bauhaus-Universität Weimar; Weimar, 2001

[Eastman et al. 1997]
Eastman C., Jeng T.S., Chowdbury R., Jacobsen K.: Integration of Design Applications with Building Models.
CAAD futures, München 1997

[Eiermann et al. 1994]
Eiermann O., Heitz S.: RETEx Abschlußbericht.
BMFT-Vorhaben 0329132A, Institut für Industrielle Bauproduktion, Universität Karlsruhe, 1994

[Fink 1998]
Fink D.: Virtuelle Unternehmensstrukturen - Strategische Wettbewerbsvorteile durch Telearbeit und Telekooperation.
Wiesbaden: Gabler 1998

[Fitzpatrick 1998]
Fitzpatrick G.: The Locales Framework - Understanding and Designing for Cooperative Work.
Dissertation Universität Queensland. Queensland (Australien), Dept. of Computer Science and Electrical Engeneering 1998.

[Gehrlein 1999]
Fa. Gehrlein: Datenaustausch mit DXF-Daten.
http://www.gehrlein.de/dxfdaten.htm 22.7.1999

[Graphisoft 1999]
Fa. Graphisoft: ArchiCAD for TeamWork - Produkt-Info.
http://www.archicad.de/de/products/actw.html 21.7.1999

[Haller 1974]
Haller F.: midi- ein offenes system für mehrgeschossige bauten mit integrierter medieninstallation.
Münsingen (Schweiz): USM bausysteme haller, 1974

[Hansen 1974]
Hansen F.: Konstruktionswissenschaft - Grundlagen und Methoden.
ISBN 3-446-11957-4, München/Wien: Hanser 1974.

[Hartwich 2000]
Hartwich C.: N-Tier Enterprise-Applikationen
Berlin: Freie Universität 2000

[Hauschild et al. 2000]
Hauschild T., Hübler R.: Aspekte der verteilten Bauwerksmodellierung in kooperativen Entwurfsumgebungen auf Basis dynamischer Objektstrukturen
Weimar: Bauhaus-Universität 2000

[Henckels 1994]
Henckels D.: Analyse eines konkreten Architekturentwurfes und Validierung der Architektur-Datenbankschnittstelle.
Studienarbeit, Institut für Programmstrukturen und Datenorganisation, Universität Karlsruhe, 1994.

[Henckels 1999]
Henckels D.: Internet-Recherche zum Begriff "Fehldaten".
(Im Quellen-Volltextverzeichnis der CD zu dieser Dissertation enthalten.)
Karlsruhe, 18.5.1999

[Henckels et al. 1997]
Henckels D., Hovestadt V., Mülle J.: Entwicklung einer Datenbank-unterstützten Architektur-Entwurfsumgebung.
Abschlußbericht, Universität Karlsruhe, Institut für Programmstrukturen und Datenorganisation / Institut für Industrielle Bauproduktion 1997

[Hovestadt 1994]
Hovestadt L.: A4 - Digitales Bauen.
Ein Modell für die weitgehende Computerunterstützung von Entwurf, Konstruktion und Betrieb von Gebäuden. Fortschrittsbericht VDI, Reihe 20, Band 120, Düsseldorf 1994

[Hovestadt et al. 1999]
Hovestadt V.; Hovestadt L.: The ARMILLA Project.
In: Automation in Construction (8), S. 325 - 337. Amsterdam: Elsevier 1999

[Hübler et al. 2001]
Hübler, R.; Werner, F.: Projektbericht D3 - Digitales Bauwerksmodell als Grundlage der Datenintegrationsebene für Bestandsinformationen.
In: SFB 524 "Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken" - Kolloquium zu den Ergebnissen der 1. Förderperiode. Bauhaus-Universität Weimar; Weimar, 2001
siehe auch http://www.uni-weimar.de/sfb/sitemap.html (geprüft 17.4.2004)

[IAI 1997]
International Alliance for Interoperability: Industry Foundation Classes, Release 1.0 - End-User Guide and Specifications.
1997

[IAI 2003]
International Alliance for Interoperability: Industry Foundation Classes, Release 2x2 final distribution.
http://www.iai-international.org/iai_international/Technical_Documents/iai_documents.html (geprüft 17.4.2004) 2003

[ISO 1996: STEP]
International Organisation for Standardisation: ISO 10303: Product Data Representation and Exchange.
Parts 11, 22, 41-45, 106, 225, 230, 231. ISO TC184/SC4 1994-96

[Juister 2004]
Juister N.: Raumbuchkonzepte für die Gebäudesanierung und Denkmalpflege
Diplomarbeit, Institut für Industrielle Bauproduktion (ifib), Universität Karlsruhe 2004

[Kilb et al. 1998]
Kilb T.; Arnold F.: Data Management in Distributed CAx Systems.
Arbeit im Rahmen des ANICA-Projektes, in: R. Anderl, D. Trippner (Hrsg.): Proceedings of the ProSTEP Science Days '98: Product Data Technology - Facing the Future, Wuppertal, 17.-18. Juni 1998, S. 94-105. Wuppertal 1998

[Kohler et al. 1996: KOBEK]
Kohler N.; Klingele M.: KOBEK - Methode zur kombinierten Berechnung von Energiebedarf, Umweltbelastungund Baukosten.
Abschlußbericht Universität Karlsruhe (TH). Karlsruhe: Institut für Industrielle Bauproduktion 1996

[Kohler 1998]
Gespräch mit Prof. Niklaus Kohler.
Institut für Industrielle Bauproduktion, Universität Karlsruhe, 22.10.1998.

[Kohler et al. 1997]
Kohler, N. und beteiligte Institute: Integration von Planung und Fertigung im Gebäude-Lebenszyklus.
Konzept zu Einrichtung eines Sonderforschungsbereichs (SFB 1683). Universität Karlsruhe, 1997.

[Kohler et al. 1998]
Kohler N., Forgber U., Müller C.: Zwischenbericht des Projektes RETEx II / INTESOL für das Jahr 1997.
Universität Karlsruhe (TH). Karlsruhe: Institut für Industrielle Bauproduktion 1998

[Kohler 2002]
Kohler, N.: Dynamische Gebäude - Positionspapier.
Universität Karlsruhe (TH). Karlsruhe: Institut für Industrielle Bauproduktion 1998

[Laabs et al. 1997]
Laabs A., Pahl P.J.: Das Beobachterkonzept zum Erhalt der Konsistenz in Datenmodellen.
In: Proceedings Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen - IKM, Bauhaus-Universität Weimar, 26.2. - 1.3. 1997

[Lockemann et al. 1993]
Lockemann P.C.; Krüger G.; Krumm H.: Telekommunikation und Datenhaltung.
ISBN 3-446-17465-6, München/Wien: Hanser, 1993

[Müller 1999]
Müller C.: Der Virtuelle Projektraum - Organisatorisches Rapid Prototyping in einer internetbasierten Telekooperationsplattform für Virtuelle Unternehmen im Bauwesen.
Dissertation. Karlsruhe: Universität Karlsruhe (TH), 1999.

[Müller et al. 1998]
Müller C.; Rodewald R.: INTEGRA - Eine integrierende Groupwareanwendung für ein Architekturbüro. X.
Bauforum Bauinformatik, Weimar 1998.

[NN 1990: FiFF]
Software-Engeneering als Hausbau
in: "FIFF Kommunikation" Ausgabe 4/90, München 1990
abrufbar unter http://www.gksoft.com/a/fun/hausbau.html (geprüft 18.4.2004)

[NN 1996: HOAI]
Verordnung über die Honorare für Leistungen der Architekten und der Ingenieure in der Fassung der fünften Änderungsverordnung,
inkrafttretend 1. Januar 1996. Berlin: Bauverlag 1996

[NN 1998: 4D-CAD]
Collaborative 4D-CAD.
CIFE, University of Stanford,
http://gaudi.stanford.edu/4D-CAD/INTRO-4DCAD 8.6.1998

[Norman 1990]
Norman D.E.: The design of everyday things.
ISBN 0-385-26774-6, New York: Basic Books 1990

[Pfaus 1998]
Pfaus M.: Das virtuelle Ingenieurbüro.
Ingenieurkammer Baden-Württemberg. Stuttgart 1998.

[Rechtin et al. 1997]
Rechtin E., Maier M.W.: The art of systems architecting.
ISBN 0-8493-7836-2, ifib #9988 B1607, Boca Raton: CRC Press, 1997

[Rezgui et al. 1996]
Rezgui Y., Brown A., Cooper G., Aouad G., Kirkham J, Brandon P.: An Integrated Framework for Evolving Construction Models.
University of Salford, Manchester, 1996

[Rittel et al. 1973]
Rittel H., Webber M.: Planning Problems are Wicked Problems.
In: Dilemmas in a general theory of planning. Policy Sciences (4), Amsterdam: Elsevier 1973.

[Rudolph et al. 1999]
Rudolph D., Stürznickel T., Weissenberger L.: DXF intern.
ISBN 3-9805108-1-6, Essen: CR/LF 1999.
siehe auch http://www.crlf.de/Verlag/DXF-intern/DXF-intern.html

[Scheer et al. 1996]
Scheer A., Kocian C.: Kiesel - Das Virtuelle Umwelt Kompetenzzentrum - Theorie und Praxis der virtuellen Unternehmung.
In: Management & Computing, 1996

[Scherer 2000]
Scherer R.: Towards a Concurrent Engeneering Environment - The ToCEE Client-Server System for Concurrent Engeneering.
ESPRIT-Projekt, Short Final Report, Dresden: Technische Universität 2000
siehe auch http://www.cib.bau.tu-dresden.de/tocee

[Schmitt et al. 1996]
Schmitt I., Saake, G.: Flexible Generation of Global Integrated Schemata using GIM.
Institut für Technische Informationssysteme, Otto-von-Guericke-Universität Magdeburg
siehe auch http://gandalf.cs.uni-magdeburg.de/iti_db/veroeffentlichungen/96/FDBS-96.html

[Seiler 1985]
Seiler W.: Technische Modellierungs- und Kommunikationsverfahren für das Konzipieren und Gestalten auf Basis der Modell-Integration.
VDI-Fortschrittsberichte, Reihe 10, Band 49, Düsseldorf: VDI-Verlag 1985

[SIA 1996]
Schweiz. Ingenieur- und Architektenverein: TOP - Teamorientiertes Planen mit dem neuen Leistungsmodell 95 des SIA (LM 95).
ISBN 3-905251-04-3. Bern 1996.

[Stulz 1993]
Stulz R.: Integrale Planung - mehr als ein Schlagwort.
Referat im Seminar "Integrale Planung" der Swissair am 26.3.1993, Intep AG, Zürich 1993

[Sturm 1997]
Sturm R.: Dynamische Regelmengen zur Beschreibung von Entwurfsspielräumen.
Promotion Universität Karlsruhe / Fortschrittsberichte VDI Reihe 10 Nr. 495, ISBN 3-18-349510-4, Düsseldorf: VDI-Verlag 1997

[Suter et al. 1986]
Suter P., Kohler N., Gfeller R., Van Gilst J.: Haustechnik in der integralen Planung.
Impulsprogramm Haustechnik, EDMZ, Bern, 1986

[Tanenbaum 1992]
Tanenbaum A.S.: Computer-Netzwerke.
ISBN 3-86033-142-6, Attenkirchen: Wolfram's Fachverlag 1992.

[Teufel et al. 1995]
Teufel S., Sauter C., Mühlherr T., Bauknecht K.: Computerunterstützung für die Gruppenarbeit.
ISBN 3-89319-878-4, Bonn: Addison-Wesley, 1995

[Türker et al. 1996]
Türker C., Conrad S.: Using Active Mechanisms for Global Integrity Maintenance in Federated Database Systems.
Institut für Technische Informationssysteme, Otto-von-Guericke-Universität Magdeburg, 1996.
siehe auch http://gandalf.cs.uni-magdeburg.de/iti_db/veroeffentlichungen/96/FDBS-96.html

[W3C 2003]
Scalable Vector Graphics SVG 1.2 - 3rd W3C working draft 15. July 2003.
http://www.w3.org/TR/2003/WD-SVG12-20030715/ (geprüft 17.4.2004) 2003
siehe auch http://www.svg.org,

[Willenbacher et al. 1998]
Willenbacher H., Hübler R.: Relationen zwischen Domänenmodellen – Ansatz zur Schaffung einer integrierenden computergestützten Bauplanungsumgebung.
SFB 524 "Werkstoffe und Konstruktionen für die Revitalisierung von Bauwerken", Weimar: Bauhaus-Universität 1998.

[Willenbacher 2002]
Willenbacher H.: Interaktive verknüpfungsbasierte Bauwerksmodellierung.
Dissertation, Weimar: Bauhaus-Universität 2002