Ein Ansatz zur Abbildung von Änderungen in der modell-basierten Objektplanung

Matthias Weise

Schriftenreihe des Instituts für Bauinformatik
Herausgeber: Univ.-Prof. Dr.-Ing. R. J. Scherer

ISBN-10: 3-86005-557-7
ISBN-13: 978-3-86005-557-1

Institut für Bauinformatik, TU Dresden

Postanschrift	Besucheranschrift
Technische Universität Dresden 01062 Dresden	Nürnberger Str. 31a 2.OG, Raum Nr. 204 01187 Dresden

Tel.: +49 351/463-32966
Fax: +49 351/463-33975
E-Mail: Raimar.Scherer@tu-dresden.de
WWW: http://cib.bau.tu-dresden.de

Diese Arbeit wurde unter dem Titel

Ein Ansatz zur Abbildung von Änderungen in der modell-basierten Objektplanung

An approach for the representation of changes in model-based design

an der Fakultät Bauingenieurwesen der Technischen Universität Dresden als

DISSERTATION

von Matthias Weise
geboren am 12.09.1974 in Gera

zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.) genehmigt.

Gutachter:

Prof. Dr.-Ing. Raimar J. Scherer, Technische Universität Dresden

Prof. Dr.-Ing. Berthold Firmenich, Bauhaus Universität Weimar

Prof. Dr.-Ing. habil. Klaus Kabitzsch, Technische Universität Dresden

Tag der öffentlichen Verteidigung: 29. November 2006

Vorwort

In einem Praktikum zu Beginn meines Studiums habe ich die ersten Erfahrungen mit dem Problem des Datenaustauschs gesammelt. Damals zeichnete sich bereits der Umstieg von 2D-Zeichnungen auf 3D-Produktmodelle ab. Wie viele andere auch war ich davon überzeugt, dass durch die Abbildung eines Bauwerks im Computer das Problem des Datenaustauschs kurzfristig zu lösen ist. Erst nach und nach wurde mir das Ausmaß und die Vielfalt dieser Problematik bewusst. Inzwischen bin ich davon überzeugt, dass die hierfür erforderliche Datenintegration nur durch eine Vielfalt an Lösungsmethoden, die Bereitstellung von umfangreichem Wissen und nicht zuletzt dem Verständnis über mögliche Grenzen erreicht werden kann. In diesem Bewusstsein möchte ich mit der vorliegenden Arbeit einen Beitrag dazu leisten, uns dem Ziel der Datenintegration weiter anzunähern.

Den Leser möchte ich ermuntern, die dargelegte Sicht und den gewählten Ansatz kritisch zu hinterfragen. Da das Lesen einer solchen Arbeit nicht unwesentlich von der Verwirklichung eigener Ideen abhält, sind für den Einstieg die Kapitel 1-3 und das Kapitel 8 empfohlen. In diesen Kapiteln sind die getroffenen Annahmen, der hieraus abgeleitete Ansatz und schließlich die Bewertung der Ergebnisse beschrieben. Wer sich darüber hinaus für den Bezug zur Praxis interessiert, der findet in Kapitel 7 eine kritische Darstellung über die Anwendung des Ansatzes. Um sich mit den theoretischen Grundlagen des Ansatzes und den entwickelten Methoden auseinanderzusetzen, sind die Kapitel 4 und 5 vorgesehen. Diese beiden Kapitel bilden den Kern der Arbeit und sind weniger für das schnelle Lesen sondern mehr für ein intensives Studium empfohlen. Das fünfte Kapitel mit seinen vier voneinander unabhängigen Teilen kann dabei auch getrennt voneinander gelesen werden. Die gewählte Umsetzung ist schließlich in Kapitel 6 beschrieben und bezieht sich auf die theoretischen Grundlagen der Kapitel 4 und 5. Hierbei wurde eine knappe Darstellung gewählt, die einen Eindruck über die Integration des Ansatzes in eine verteilte Umgebung, die Datenhaltung sowie die algorithmische Umsetzung der entworfenen Methoden vermittelt.

Schließlich möchte ich das Vorwort nutzen, mich bei all jenen Personen zu bedanken, die zum Entstehen dieser Arbeit beigetragen haben. Mein besonderer Dank gilt meinem Betreuer Prof. Raimar J. Scherer, der meine Arbeit über all die Jahre mit regem wissenschaftlichen Disput und den notwendigen Freiheiten unterstützt hat. Prof. Berthold Firmenich und Prof. Klaus Kabitzsch danke ich für die Begutachtung der Arbeit, die anregenden Diskussionen und ihre Hinweise zur Verbesserung der Arbeit. Meinen Kollegen vom Institut danke ich für die freundschaftliche Atmosphäre und den gegenseitigen Austausch über ein breites Spektrum wissenschaftlicher und nicht wissenschaftlicher Themen. Peter Katranuschkov sei hier stellvertretend erwähnt, weil er mein Denken und Handeln durch seine stets kritische Art besonders beeinflusst hat. Nicht zuletzt danke ich meiner Familie, die mich über die Jahre in vielerlei Hinsicht unterstützt und die notwendige Geduld aufgebracht hat.

Ich wünsche viel Erfolg beim Lesen und würde mich freuen, wenn hieraus neue Erkenntnisse und Ideen entstehen.

Dresden, im Dezember 2006
Matthias Weise

Kurzfassung

Die Planung von Bauwerken ist durch paralleles Arbeiten und den Einsatz einer Vielzahl unterschiedlicher Modelle geprägt. Dadurch entstehen voneinander abhängige und redundante Planungsdaten, die nach Änderungen zu aktualisieren sind. Das Aktualisieren der Planungsdaten ist durch die hohe Komplexität jedoch nicht vollständig automatisierbar und erfordert das Einbeziehen der beteiligten Planer. Hierbei besitzt das Erkennen von Änderungen eine hohe Aussagekraft, da durch das iterative, parallele Arbeiten der wechselseitige Datenabgleich dominiert. Solche Änderungsinformationen sind in der Regel aber nicht verfügbar. Statt dessen sind überarbeitete, häufig fehlerhaft aktualisierte Datensätze zu bewerten. Die vorliegende Arbeit thematisiert mit der Verwaltung einer gemeinsamen Datenbasis diese Problemstellung und beschreibt einen Ansatz, der den Übergang zu einer auf Änderungen beruhenden Dokumentation der Planungsschritte ermöglicht.

Die Grundlage des vorgestellten Ansatzes bildet ein objekt-orientiertes Meta-Modell, das die Unabhängigkeit von konkreten Modellen und somit die Wiederverwendbarkeit der entwickelten Methoden erlaubt. Jeder Planungsschritt wird über Änderungen beschrieben und kann nachträglich bewertet werden. Hierfür wird eine auf wenigen Operationen beruhende Änderungssemantik verwendet, die auch Änderungen im Objektgefüge beschreiben kann. Auf diese Weise können geänderte Informationen unabhängig von ihrer Zuordnung zu Objekten dokumentiert werden. Die hiermit gewonnene Flexibilität ist notwendig, um mit der Beschreibungsvielfalt der Modelle, die für gleiche Informationen eine Vielzahl von Darstellungsformen in der Datenstruktur erlauben, sowie den frühen Phasen des Bauwerksentwurfs, die durch starke Änderungen geprägt sind, umgehen zu können. Zugleich wird damit die Grundlage gelegt, die häufig nicht eindeutig identifizierbaren Informationen in aussagefähige Änderungen zu überführen.

Der Übergang von Datensätzen zu Änderungen wird durch 1.) den Umgang mit individuell benötigten Teildatensätzen, die den Verlust von Informationen verhindern, und 2.) ein auf der Datenstruktur arbeitendes Vergleichsverfahren, das auf einem heuristischen Ansatz zur Zuordnung nicht eindeutig identifizierbarer Informationen beruht, ermöglicht. Auf dieser Grundlage werden die überarbeiteten Teildatensätze in Änderungen überführt und anschließend über einen heuristischen Ansatz auf den Gesamtdatensatz übertragen. Ein Planungsschritt wird somit in drei Teilschritte zerlegt, die jeweils durch Änderungen dokumentiert werden. Der entstandene Gesamtdatensatz ist nachträglich durch die Planer oder hierfür geeignete Werkzeuge zu überprüfen, da die erkannten Änderungen einerseits nicht zwingend den tatsächlich durchgeführten Änderungen entsprechen und andererseits nicht immer konfliktfrei auf den Gesamtdatensatz übertragen werden können. Durch die heuristischen Ansätze werden für beide Problemstellungen sinnvolle Lösungen angeboten, die durch die Dokumentation der Teilschritte getrennt voneinander bewertet werden können. Das gleiche Prinzip wird für das Zusammenführen paralleler Planungsschritte angewendet, indem die getrennt voneinander durchgeführten Änderungen nach vordefinierten Regeln vereinigt und dokumentiert werden. Auf dieser Grundlage ist eine nachvollziehbare, flexible und fehlertolerante Dokumentation der Planungsschritte möglich, die eine wichtige Voraussetzung für das kooperative Arbeiten liefert.

Abstract

Building design is characterized by parallel work and the use of a multitude of different domain models. This leads to the generation of interdependent and redundant design information that has to be synchronised and updated after design changes. However, due to the high complexity of the task, matching of the design data cannot be completely automated. It requires reviewing and corrective actions by the involved designers. Identification of the undertaken changes is hereby of highest significance, especially because of the dominating multi-directional data matching necessitated by the iterative, parallel work. However, change information is typically not available. Instead of being aware of changed design data, evaluation of design steps has to be undertaken with reworked, often erroneously updated design data sets. The presented work addresses this problem by specifically focusing on the management of shared design data. It provides an approach that enables a paradigm shift towards change-based documentation and evaluation of the design steps.

The suggested approach is based on an object-oriented meta-model allowing to handle different kinds of domain models and thus providing reusable methods. Each design step is described with the help of change information and can be reviewed afterwards. For that purpose, a minimal set of change types is used, which also allows to describe changes within the object structure. Consequently, changed design information can be documented without enforced allocation to objects. The flexibility achieved in this way is essential for tackling the representational plurality of the models, allowing different data structures for one and the same design information, and for dealing with the early design stages characterized by more radical changes in the data structure. At the same time, it provides the basis to transform frequently not uniquely identifiable design information into meaningful changes.

Switching from design states to changes is achieved by 1.) using individually needed design data subsets, enabling to prevent data loss, and 2.) a generic compare algorithm, working purely on the data structure and applying a heuristic approach for the appropriate allocation of not uniquely identifiably design information. On that basis, reworked design data subsets are transformed to design changes and, using a heuristic approach, are then applied to update the shared design data. Thus, a design step is split into three stages, each of them being documented by change operations. The resulting new model state, being not necessarily consistent with other concurrently modified design data, has to be reviewed by the involved designers or by appropriate tools in order to find misleadingly identified design changes, i.e. unintended compare results, or wrongly updated design data, i.e. not correctly fixed inconsistencies. By applying heuristics model comparison as well as automated updating of the shared design data both offer meaningful solutions, which can be separately evaluated by using the change information. The same principle is used to harmonise concurrently executed design steps by merging and documenting independently created design changes with the help of predefined rules. The suggested approach is an essential basis to realize easily traceable, flexible and fault-tolerant documentation of design changes, which is an important prerequisite for cooperative design work.

Inhalt

1 Einleitung

1.1 Problemstellung

An ein Bauwerk werden vielfältige Anforderungen gestellt, zu denen nicht nur die Standsicherheit und Gebrauchstauglichkeit, sondern auch ökonomische und ästhetische Vorgaben gehören. Um diese Anforderungen mit einem angemessenen Planungsaufwand erfüllen zu können, werden Problemstellungen idealisiert und in handhabbare Modelle abgebildet. Die Planungsaufgabe wird auf diese Weise in beherrschbare Teilprobleme zerlegt, die von Fachplanern eigenständig bearbeitet werden. Die entstehenden Teillösungen werden im Planungsprozess aufeinander abgestimmt und schrittweise zu einer widerspruchsfreien Gesamtlösung zusammengeführt. Der Planungsprozess wird dadurch in Bearbeitungs- und Abstimmungsphasen unterteilt, die einen regelmäßigen Abgleich der Planungsdaten erfordern.

Der Einsatz von Computern ermöglicht eine umfangreiche und sehr genaue Simulation der Bauwerke. Erst dadurch können die steigenden Anforderungen in immer kürzeren Planungszeiten erfüllt werden. Das Prinzip des Planungsprozesses hat sich damit aber nicht wesentlich geändert. Der Planungsprozess ist nach wie vor durch die Verwendung verschiedener Fachmodelle und eine iterative, durch wiederkehrende Abstimmungsphasen gekennzeichnete Bearbeitung des Bauwerksentwurfs geprägt. Mit der Zunahme der verwalteten Informationen und der gleichzeitig stärkeren Parallelisierung der Aufgaben ist ein höherer Aufwand für den Abgleich der Planungsdaten notwendig, der ohne technische Unterstützung nicht mehr beherrschbar ist.

Beim Abgleich der Planungsdaten werden überarbeitete Teillösungen auf andere Fachmodelle und damit auf andere Idealisierungen des Bauwerks übertragen. Durch die Vielfalt und den hohen Abstraktionsgrad der eingesetzten Modelle entstehen komplexe Abhängigkeiten, die mit Hilfe neutraler Datenmodelle vereinfacht werden sollen. Fachspezifische Teillösungen können auf diese Weise in einer gemeinsamen Datenstruktur zusammengeführt werden und erleichtern die Abstimmung der Fachplaner. Dennoch bleiben die Abhängigkeiten der Fachmodelle schwer formalisierbar und führen zu einem unvollständigen und mitunter fehlerhaften Abgleich der Planungsdaten. Der Datenabgleich ist somit nicht zuverlässig automatisierbar und erfordert zusätzliche Kontrollen durch die Planer.

Der heute übliche Planungsprozess gestaltet sich für die beteiligten Planer nur wenig transparent und erschwert die gegenseitige Abstimmung. Wesentliche Ursache ist der Austausch kompletter Planungsstände, der einen hohen Aufwand für die Bewertung eines Planungsschrittes erfordert. Gleichzeitig entstehen Informationsverluste, weil die ausgetauschten Planungsdaten durch fachspezifische Programme in der Regel nicht vollständig interpretiert werden können. Ein aussagekräftiger Bezug zwischen den Planungsständen ist auf dieser Grundlage nur schwer erkennbar und behindert den Abgleich der Planungsdaten. Dadurch ist ein unverhältnismäßig hoher Aufwand notwendig, um unvollständige und voneinander abweichende Planungsstände zu vermeiden. Für das produktmodellbasierte Arbeiten wird daher ein Ansatz benötigt, der die Abstimmung der Fachplaner durch den Übergang zu einem nachvollziehbaren Planungsprozess vereinfacht.

1.2 Arbeitshypothese

In der Forschung wird allgemein anerkannt, dass die bestehende Modellvielfalt nicht durch ein allumfassendes Bauwerksmodell ersetzt werden kann. Die Anwendung unterschiedlicher Modelle, d. h. die idealisierte Abbildung realer Sachverhalte und damit das Ausblenden irrelevanter Bauwerksinformationen, wird auch weiterhin die Bauplanung prägen. Der modellübergreifende Datenabgleich ist somit ein wichtiges Kriterium, um die Planung schneller, qualitativ besser und kostengünstiger ausführen zu können.

Im Rahmen der Arbeit werden objekt-orientierte *Produktdatenmodelle* und die Pflege einer gemeinsamen Datenbasis als Grundlage für einen effizienten Datenabgleich gesehen. Die Einigung auf ein gemeinsames Produktdatenmodell liefert ein Bezugssystem, das nicht nur für den Datenaustausch, sondern auch für das Koordinieren der Zusammenarbeit genutzt werden kann. Ein fehlerfreier Datenabgleich ist auf dieser Grundlage jedoch nicht zu erwarten. Hierfür gibt es mehrere Gründe, die unter anderem auf technische Probleme und ökonomische Zwänge zurückzuführen sind. Der Datenabgleich wird somit auf absehbare Zeit ein unvollständig abgebildeter Vorgang bleiben, der die Kontrolle und korrigierenden Eingriffe der Anwender verlangt. Es wird daher davon ausgegangen, dass

- der Bedarf eines nachvollziehbaren Planungsprozesses besteht und
- das Erkennen veränderter Planungsdaten eine der wichtigsten Informationen in einem durch Iteration geprägten Planungsprozess ist.

Besondere Anforderungen stellen die oft starken Veränderungen der Planungsdaten dar. Verschiedene Ursachen sind hierfür verantwortlich:

- die zeitlich langen, voneinander unabhängig durchgeführten Planungsschritte,
- die kreativen Phasen des Bauwerksentwurfs, die teilweise zu radikalen Planungsänderungen führen, und schließlich
- die technischen Unzulänglichkeiten der Datenintegration.

In dem betrachteten Ansatz bildet die Pflege einer gemeinsamen Datenbasis die Grundlage für den Abgleich der Planungsdaten. Dadurch kann jedoch nicht verhindert werden, dass Planungsschritte zu inkonsistenten Planungsständen oder gegenseitigen Konflikten führen. Wesentliche Ursache ist das parallele Arbeiten, das oft unverzichtbar für das Einhalten von Terminen ist. Die Gefahr von Konflikten wird somit bewusst akzeptiert und stellt ein einkalkuliertes Risiko dar. Aus diesem Grund ist es häufig nicht gewünscht, den Planungsablauf seitens der Datenbasis durch das Kriterium der Datenkonsistenz zu reglementieren.

In der Arbeit wird die Ansicht vertreten, dass das Ziel der Datenintegration nur durch eine Vielzahl überschaubarer Verbesserungen erreicht werden kann. Ein nachvollziehbarer Planungsprozess leistet hierzu einen wichtigen Beitrag. In einer solchen Lösung wird die Möglichkeit gesehen, die technischen Unzulänglichkeiten des Datenabgleichs zu kompensieren und dadurch eine von einzelnen Anwenderprogrammen unabhängigere Datenbasis zu erreichen.

1.3 Zielsetzung

Mit der Arbeit wird das Ziel verfolgt, das kooperative Arbeiten mit einer gemeinsamen Datenbasis zu unterstützen. Auf der Grundlage eines standardisierten (Produkt-)Datenmodells sollen die Planungsschritte nachvollziehbar in einer gemeinsamen Datenbasis beschrieben werden. Ein hierfür geeigneter Ansatz soll folgendes leisten:

1. die Eignung für objekt-orientierte Datenmodelle, um modellspezifische Lösungen zu vermeiden und somit für eine Vielzahl von Datenmodellen anwendbar zu sein,
2. die Unterstützung der unabhängigen, parallelen Bearbeitung, um den Planungsprozess flexibel organisieren zu können, und
3. die Nutzung von Planungsständen, um bestehende Anwenderprogramme für die Bearbeitung der Planungsdaten nutzen zu können.

Auf der Grundlage eines objekt-orientierten Meta-Modells werden als Teilziele 1.) die Abbildung und Unterstützung der Planungsschritte und 2.) der Umgang mit Planungsständen abgeleitet.

Das *erste Teilziel* betrachtet die Forderung nach Transparenz und Nachvollziehbarkeit der Planungsschritte. Hierfür ist es erforderlich,

- einen aussagekräftigen Bezug zwischen zwei Bauwerksentwürfen,
- ein schnelles Erkennen der geänderten Planungsdaten und schließlich
- das Bewältigen der anfallenden Datenmenge zu ermöglichen.

Für die parallele Bearbeitung der Planungsdaten ist es darüber hinaus erforderlich,

- das Abbilden alternativer Bauwerksentwürfe und
- das Auflösen von Widersprüchen und somit das Wiederherstellen einer gemeinsamen Entwurfslösung zu unterstützen.

Das *zweite Teilziel* verfolgt den Umgang mit Planungsständen. Hierfür ist es notwendig, den Übergang von den typischerweise verwendeten Planungsständen in die hier verwendete, auf Änderungen beruhende Sichtweise zu ermöglichen. Dabei ist zu berücksichtigen, dass die Voraussetzungen für ein fehlerfreies Vergleichsergebnis in der Regel nicht gegeben sind. Dieser Nachteil soll durch die Nutzung von Teildatensätzen reduziert werden. Hierfür sind

- die individuell benötigten Teildatensätze zu beschreiben, aus dem Gesamtdatensatz herauszulösen und als Planungsstand bereitzustellen sowie
- der überarbeitete Teildatensatz in den Gesamtdatensatz wieder einzugliedern.

Schließlich wird ein Vergleichsverfahren benötigt, das im Sinne der Planungsdokumentation die Änderungen zwischen zwei Planungsständen ermitteln kann. Hierfür wird ein Vergleichsverfahren benötigt, das

- auf der Grundlage des objekt-orientierten Meta-Modells arbeitet und
- die Planungsdaten auch ohne eindeutigen Identifikator erkennen kann.

Beide Teilziele sind notwendig, um den Ansatz unter praxisnahen Bedingungen verifizieren zu können. Hierfür sollen das IFC-Modell, verfügbare Anwendungen und reale Datensätze genutzt werden.

1.4 Abgrenzung

Mit der vorliegenden Arbeit wird die Verwaltung einer gemeinsamen Datenbasis thematisiert, die den Abgleich der Planungsdaten unterstützt. Hierbei werden Annahmen getroffen, die für ein effizientes Arbeiten mit der gemeinsamen Datenbasis vorausgesetzt werden. Zu diesen Annahmen gehören:

1. Die Existenz eines Produktdatenmodells, das die auszutauschenden Planungsinformationen aufnehmen kann.
2. Die Verfügbarkeit von Programmen, die auf der Grundlage des vereinbarten Produktdatenmodells die Planungsinformationen lesend und schreibend verarbeiten können.
3. Die Koordinierung der Planungsschritte, die den Planungsablauf im Sinne der Projektziele optimiert und die Zusammenarbeit der Planer regelt.

Es wird vorausgesetzt, dass das verwendete Produktdatenmodell eine objekt-orientierte Datenstruktur beschreibt und in der Modellierungssprache EXPRESS (ISO 10303-11) abgebildet werden kann. Für die Modellierung der Datenstruktur bestehen keine Einschränkungen. Darüber hinaus werden keine zusätzlichen Anforderungen wie die Formulierung von Konsistenzbedingungen oder Abhängigkeitsbeziehungen gestellt.

Mit der Wahl eines Produktdatenmodells wird eine Datenstruktur vereinbart, die in der Regel nicht direkt für die Bearbeitung in einem Programm geeignet ist. Die Planungsdaten sind vielmehr in die programmintern genutzten Datenstrukturen zu überführen. Hierbei sind nicht nur strukturelle, sondern auch semantische Unterschiede zu überwinden. Die Qualität der Planungsdaten wird durch diesen Vorgang maßgeblich beeinflusst und besitzt daher eine hohe Bedeutung. Die hierfür erforderlichen Transformationsregeln und deren Umsetzung sind jedoch nicht Thema dieser Untersuchung.

Der Abgleich von Planungsdaten ist nur dann erforderlich, wenn der Bauwerksentwurf geändert wurde. Auf der Ebene der Datenverwaltung werden logisch zusammengehörige Änderungen als Einheit betrachtet und in einem Planungsschritt zusammengefasst. Hierbei ist es unerheblich, wie die Änderungen eines Planungsschrittes entstanden sind. Es wird davon ausgegangen, dass die Abfolge, die Dauer und der Umfang der Planungsschritte sowie der Bezug zu beteiligten Planern durch eine übergeordnete Prozessplanung verwaltet werden.

1.5 Gliederung

Die Arbeit ist in 8 Kapitel und einen Anhang unterteilt. Den Schwerpunkt bilden die Kapitel 3, 4 und 5, die den Lösungsansatz und die hierfür entwickelten Methoden beschreiben. Die Kapitel 6 und 7 sind der Umsetzung und Validierung gewidmet, die die Eignung des Ansatzes hinsichtlich Realisierbarkeit und Anwendbarkeit zeigen.

Das *zweite Kapitel* beschreibt den Stand der Forschung auf dem Gebiet der Datenintegration. Obwohl dieses Thema seit einigen Jahren ein hohes Interesse erfährt, hat sich die Situation in der Praxis noch nicht wesentlich geändert. Vorhandene Integrationsansätze werden daher kritisch hinterfragt und den Problemen der Datenintegration gegenübergestellt. Zusätzlich werden Anforderungen aus der verteilt kooperativen Bearbeitung diskutiert, die die Möglichkeiten der Datenintegration beeinflussen. Hieraus werden Rahmenbedingungen abgeleitet, die in dem Lösungsansatz zu berücksichtigen sind.

Für das Erreichen der vorgegebenen Ziele wird im *dritten Kapitel* ein Lösungsansatz vorgestellt. Es werden die getroffenen Annahmen formuliert, die eng mit der vorgestellten Lö-

sungsidee, den hierfür benötigten Methoden und der skizzierten Anwendung verbunden sind. Die Darstellung ist auf das Konzept des Lösungsansatzes beschränkt und vermittelt zunächst einen Überblick, der in den nachfolgenden Kapiteln detailliert wird.

Im *vierten Kapitel* wird die theoretische Grundlage des Lösungsansatzes vorgestellt. Losgelöst von möglichen Anwendungsszenarien steht die Dokumentation der Planungsschritte im Mittelpunkt. Hierzu gehören die Organisation der Planungsdaten, die einzuhaltenden Konsistenzbedingungen sowie die Beschreibung der Zugriffsmethoden. Aus dieser Formalisierung leiten sich die beschreibbaren Informationen ab, die den beteiligten Fachplanern für weitere Auswertungen zur Verfügung stehen. Gleichzeitig werden die Informationen festgelegt, die aus dem Planungsprozess für die Dokumentation der Planungsschritte gewonnen werden müssen.

Für die Anwendung in langen Transaktionen und die Unterstützung von Planungsständen werden zusätzliche Methoden benötigt, die die Planungsschritte unterstützen und die durchgeführten Änderungen bestimmen können. Diese Methoden werden im *fünften Kapitel* beschrieben. Die hier vorgestellte Funktionalität bietet die Grundlage, um den Lösungsansatz unter praxisnahen Bedingungen testen und anwenden zu können. Sie stellen dadurch eine Ergänzung des Lösungsansatzes dar, die bei gleicher Funktionalität auch durch andere Methoden ersetzbar sind.

Das *sechste Kapitel* beschäftigt sich mit der Umsetzung des Lösungsansatzes in ein Softwaresystem. Hierfür wird eine Softwarearchitektur vorgestellt, die das Einbinden der Anwenderprogramme in eine verteilte Umgebung und die Zusammenarbeit auf der Grundlage einer gemeinsamen Datenbasis ermöglicht. Für diese Umgebung wird die methodische Umsetzung der Datenverwaltung und der ergänzenden Dienste beschrieben, die das Überprüfen der erarbeiteten Konzepte erlauben.

Die Validierung des Lösungsansatzes erfolgt im *siebenten Kapitel* anhand des prototypisch implementierten Softwaresystems. Als Beispiel mit praktischer Bedeutung werden das IFC-Modell und hierfür verfügbare Anwenderprogramme betrachtet. Für diesen Fall werden die erreichbaren Vorteile diskutiert und den damit verbundenen Problemen gegenübergestellt. Zusätzlich wird untersucht, inwieweit die gewonnenen Erkenntnisse auf andere Datenmodelle übertragbar sind.

Das *achte und letzte Kapitel* fasst die Arbeit zusammen und bewertet die erreichten Ergebnisse. Mit dieser Bewertung werden die gewonnenen Erkenntnisse und der Beitrag zum Themenkomplex der Datenintegration diskutiert. Hieraus werden offene Problemstellungen abgeleitet und ein Ausblick auf die weiteren Arbeiten gegeben.

1.6 Hinweise für den Leser

Im Anhang der Arbeit sind zusätzliche Angaben für ein besseres Verständnis der Arbeit zu finden. Neben ergänzenden Spezifikationen, die die Kapitel vervollständigen und zum Nachschlagen genutzt werden können, sind vor allem Hinweise zu den im Text und den Abbildungen verwendeten Konventionen zu finden. Hierbei sei besonders auf die Erklärung der ab Kapitel 3 verwendeten grafischen Darstellung sowie der ab Kapitel 4 genutzten mathematischen Notation hingewiesen.

2 Grundlagen einer kooperativ nutzbaren Datenbasis

Das Arbeiten mit gleichen Planungsdaten stellt ein allgemein anerkanntes Prinzip für eine schnellere, qualitativ verbesserte und kostengünstigere Planung dar. Dieses Kapitel beschäftigt sich mit dem Stand der Forschung und diskutiert verschiedene Lösungsansätze, um das Ziel einer gemeinsamen, konsistenten Datenbasis zu erreichen.

Im *ersten Abschnitt* werden Abbildung und Analyse der Bauwerksentwürfe unter dem Aspekt der Datenintegration betrachtet. Es wird gezeigt, warum die Planung durch verschiedene Modelle geprägt und der Datenabgleich mit den verfügbaren Informationen nicht fehlerfrei realisierbar ist. Der Einsatz standardisierter Produktdatenmodelle erweist sich dennoch als ein geeigneter Ansatz, um den Abgleich der Planungsdaten zu vereinfachen. Aber auch für diesen Ansatz lässt sich zeigen, dass das Problem der Datentransformation ein zentrales Thema der Datenintegration bleibt und der Verlust von Informationen zu erwarten ist.

Der *zweite Abschnitt* betrachtet Kooperationsstrategien, die das Arbeiten auf einer gemeinsamen Datenbasis regeln. Verschiedene Konzepte der Zusammenarbeit werden unter dem Aspekt widerspruchsfreier Planungsdaten diskutiert. Diese Diskussion verdeutlicht, warum der Planungsablauf nur eingeschränkt durch die Organisation der Datenzugriffe reglementiert werden kann. Es wird gezeigt, dass auf die optimistische Kooperationsstrategie nicht verzichtet und die Konsistenz der Planungsdaten nicht nach jedem Planungsschritt erreicht werden kann. Hieraus resultiert die Notwendigkeit, widersprüchliche Planungsdaten erkennen und zu geeigneter Zeit auflösen zu können. Hierfür ist die Anbindung an die Projektsteuerung erforderlich, die auf der Ebene der Planungsschritte realisiert werden kann.

Der *dritte Abschnitt* behandelt die Dokumentation der Planungsschritte, die nicht nur aus rechtlichen Gründen sinnvoll ist. Ein Vergleich mit der Softwareentwicklung und anderen Ingenieurdisziplinen zeigt, dass das Verwalten von Versionen eine inzwischen unverzichtbare Grundlage für das kooperative Arbeiten ist. Gleichzeitig wird deutlich, dass verfügbare Versionsansätze in der Bauplanung aufgrund anderer Voraussetzungen und Anforderungen nicht direkt anwendbar sind. Dies betrifft in erster Linie die Struktur und den Umgang mit Planungsdaten, die das Bestimmen und Erfassen der Änderungen erschweren.

Die technische Umsetzung ist das Thema des *vierten Abschnitts*. Es werden Softwarearchitekturen vorgestellt, die das verteilt-kooperative Arbeiten auf einer gemeinsamen Datenbasis unterstützen. Die Client-Server-Architektur stellt hierbei einen erprobten Ansatz dar, um den Zugriff auf die Planungsdaten zu überwachen. Die Einbindung verfügbarer Anwenderprogramme wird jedoch durch die komplexe Beschreibung der gewünschten Teildatensätze erschwert. Schließlich wird deutlich, dass das Aktualisieren der gemeinsamen Datenbasis auch bei Verwendung eines einheitlichen Produktdatenmodells nur unter idealen Bedingungen zu einem konsistenten Planungsstand führt. Für das Wiederherstellen der Konsistenz ist es daher unverzichtbar, zusätzliches Fachwissen sowie die Entscheidungen der Anwender einzubeziehen.

Den *Abschluss* des Kapitels bildet die Diskussion der wesentlichen Erkenntnisse, die das Arbeiten auf einer gemeinsamen Datenbasis beeinflussen. Hierbei werden der Forschungsbedarf und die vorhandenen Zwänge zusammenfassend verdeutlicht.

2.1 Modellbildung und Analyse von Bauwerksentwürfen

Das Planen eines Bauwerkes ist ohne Kommunikation nicht vorstellbar. Durch die Kommunikation ist es möglich, einen Bauwerksentwurf zwischen dem Architekten und dem Bauherrn auszutauschen und gemeinsam zu verändern. Sie führt aber auch zu Missverständnissen und ist Ursache für Fehlplanungen. Ogden und Richards (1923) haben das Problem der Kommunikation in ihrer Linguistiktheorie verdeutlicht. In ihrem Bedeutungsdreieck (*meaning triangle*) stellen sie den Bezug zwischen einem Begriff (*symbol*) und seiner Bedeutung (*referent, object*) her. Begriff und Bedeutung stehen aber nicht direkt in Beziehung sondern sind über das Kontextwissen (*thought, reference*) miteinander verbunden. Erst über das Kontextwissen erlangt der Begriff eine Bedeutung (siehe Abbildung 2-1, links). So soll mit dem Wissen des Bauherrn aus den Begriffen des Architekten ein einheitliches Verständnis über das geplante Bauwerk entstehen. Ordnet der Bauherr aufgrund seiner Erfahrungen den Begriffen des Architekten eine andere Bedeutung zu, so kommt es zu Missverständnissen.

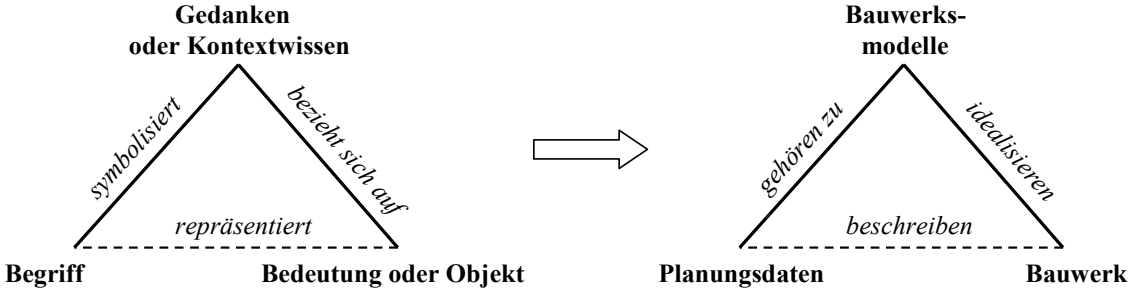


Abbildung 2-1 Bedeutungsdreieck nach Ogden und Richards (1923) und seine Übertragung auf den Bauwerksentwurf

Überträgt man die Theorie von Ogden und Richards auf die Kommunikation über ein geplantes Bauwerk, so kann man vereinfachend die Begriffe mit den Planungsdaten, das Kontextwissen mit den Modellen und das beschriebene Objekt mit dem geplanten Bauwerk gleichsetzen (siehe Abbildung 2-1, rechts). Für die Planungsbeteiligten muss es also möglich sein, aus den verfügbaren Planungsdaten über ihre individuellen bzw. fachbezogenen Modellvorstellungen das gleiche Bauwerk abzuleiten. Dadurch stehen die Modelle im Mittelpunkt, wenn die Verbesserung der Kommunikation erreicht werden soll.

Neben der Kommunikation existieren jedoch noch andere Anforderungen an die verwendeten Bauwerksmodelle. Dies wird deutlich, wenn man das Ziel der Modellbildung betrachtet. Es werden daher Konzepte notwendig, die trotz unterschiedlicher Modellvorstellungen den Austausch über das geplante Bauwerk erlauben. Die Schwierigkeiten der Datenintegration werden verständlich, wenn man die Art und den Umfang des im Computer abgebildeten Modellwissens betrachtet. Trotz verschiedener Integrationsansätze bleibt die Transformation von Planungsdaten, quasi die Übersetzung gleichbedeutender Begriffe, ein bestimmendes Thema, um die Kommunikation eines Bauwerksentwurfs zu ermöglichen.

2.1.1 Die Abbildung der Planungsprobleme in Modelle

Der Modellbegriff ist mit vielfältigen Bedeutungen belegt und bedarf zunächst einer allgemeinen Abgrenzung. Bezogen auf die Informatik sehen Wedekind et al. (1998) den Hauptzweck der Modellbildung darin, „*die aus den Fachwissenschaften stammenden Modelle so umzuschreiben, dass sie mit Hilfe eines Computers dargestellt und bearbeitet werden können*". Damit wird die Frage nach der Modellbildung zunächst auf die Modelle der Fachwissenschaften gelenkt, die nach klassischer Vorstellung vereinfachte Abbilder der Realität sind.

Diese Vorstellung verdeutlicht das komplexitätsreduzierende Prinzip, führt nach Wedekind aber zu einem naiv-realistischen Modellverständnis, da in Wirklichkeit nicht die Realität, sondern immer disziplinspezifische Versuchs- und Beobachtungskontexte abgebildet werden.

Entstehung der Bauwerksmodelle

Modelle sind das Ergebnis eines Entwicklungsprozesses, dessen Ursache das Bestreben nach einem besseren Verständnis über das Verhalten der Bauwerke ist. Im Laufe der Zeit haben neue Erkenntnisse zur Weiterentwicklung von Theorien und schließlich zu einem hohen Spezialisierungsgrad geführt. Aus Versuchen über das Tragverhalten hat sich beispielsweise die Baustatik entwickelt, die unsere Versuchserfahrungen ausreichend genau in handhabbare mathematische Beschreibungen abbildet. Durch diese Entwicklung sind Modelle entstanden, die nur mit speziellem Fachwissen angewendet werden können und zur Entstehung der verschiedenen Fachgebiete geführt haben (Straub 1975). Gleichzeitig wurden *integrierende Modelle* geschaffen, die für eine fachübergreifende Kommunikation verwendet werden. Beispielsweise ist die Technische Zeichnung eine bis heute unverzichtbare Darstellungsform, um einen Bauwerksentwurf für alle Fachplaner gleichermaßen verständlich zu beschreiben (Neufert 1950/2002)[1]. Dies ist möglich, da die Abstraktionsform der Technischen Zeichnung den verschiedenen Fachplanern geläufig ist und sie gelernt haben, aus diesen Angaben den Bezug zu den eigenen Fachmodellen herzustellen.

Die Entwicklung der Modelle ist nicht nur an neu gewonnene Erkenntnisse, sondern auch an die Möglichkeiten der verfügbaren Hilfsmittel gekoppelt. Dieser Zusammenhang ist am Beispiel des Computers erkennbar, der zur Anwendung rechenintensiver Verfahren wie der Finiten-Elemente-Methode (FEM) geführt hat. Eastman (1999) verdeutlicht diesen Zusammenhang an der Entwicklung von CAD-Funktionalitäten, indem er den Bezug zur benötigten Rechenleistung sowie den Eingabe- und Ausgabegeräten herstellt.

Modellvielfalt

Die Modellentwicklung hat zu einer hohen Modellvielfalt und vor allem zu einem deutlichen Zuwachs der Planungsdaten geführt. Der Datenabgleich ist dadurch wesentlich umfangreicher geworden und kann auf klassischem Weg immer weniger bewältigt werden. Aus diesem Bedarf sind Entwicklungsbemühungen entstanden, die nach dem Prinzip der Technischen Zeichnung angemessenen Ersatz schaffen wollen. Dieser Bemühung steht jedoch die Weiterentwicklung der Fachmodelle entgegen. Aus der bisherigen Entwicklung ist erkennbar, dass dies einerseits zu einer weiter zunehmenden Spezialisierung und andererseits zu umfangreicheren Abhängigkeiten zwischen den Fachmodellen führen wird.

2.1.2 Formalisierung der Bauwerksmodelle

Das Ziel der Formalisierung ist die nachvollziehbare Abbildung der Fachmodelle mittels Computersoftware. Aus den Formalisierungsmöglichkeiten, die selbst einem anhaltenden Entwicklungsprozess unterliegen, ergeben sich Grenzen für die Abbildung der Fachmodelle. Diese Grenzen leiten sich zu einem hohen Grad aus dem Aufwand für das Abbilden der Fachmodelle in eine computergerechte Form ab. Weitere Einschränkungen entstehen aus der algorithmischen Umsetzung, die grundsätzlich durch die *Berechenbarkeit* eines Problems (Schö-

[1] Auch in der 37. Auflage (2002) heißt es unverändert: „*Die Sprache des Entwerfers ist die Zeichnung, damit legt er unmißverständlich, international lesbar seine Angaben nieder, rein sachlich in geometrischen Zeichnungen, gewinnend, gar bestechend durch Schaubilder.*“

ning 2001) und die verfügbare Rechenleistung bestimmt werden. Aus diesen Zwängen entsteht die Frage, bis zu welchem Grad die Arbeit des Planers durch die Möglichkeiten des Computers erleichtert werden kann. Gemäß dieser Zielsetzung sind bestimmte Elemente des Fachmodells zu formalisieren. Die Formalisierung wird dabei unter dem Aspekt der Wiederverwendbarkeit betrachtet, um für gleiche Problemklassen anwendbar zu sein.

Anwendung eines Fachmodells

Das Wissen, das ein Fachplaner für die Anwendung seines Fachmodells benötigt, lässt sich in drei Gruppen unterteilen:

1. Der Modellzustand, also die Planungsdaten des Fachmodells. *(Beispiel: Die Länge l des Trägers T_1 beträgt 5 m und seine Belastung q sei 10 KN/m.)*
2. Das Verhalten des Fachmodells, also die Reaktionen auf Änderungen des Modellzustands. *(Beispiel: Das maximale Moment eines Einfeldträgers unter Gleichlast berechnet sich aus $M_{max} = q\ l^2/8$ und ist somit von q und l abhängig.)*
3. Die Verknüpfung mit einem integrierenden Modell, also die Wechselbeziehungen zwischen den Planungsdaten des Fachmodells und den Planungsdaten des integrierenden Modells[1]. *(Beispiel: Die Platte $P_{1\text{-}Statik}$ wird von dem Raum $R_{1\text{-}Architektur}$ belastet und ergibt durch die Nutzung als Büro eine Verkehrslast von 2 KN/m².)*

Die Verknüpfung mit einem integrierenden Modell ist ein wesentliches Element bei der Anwendung eines Fachmodells und darf in einer vollständigen Formalisierung nicht fehlen (Turk 2001). Eir (2004) beschreibt in seiner Arbeit die vielfältigen Beziehungen zwischen den Modellen und stellt einen ganzheitlichen Ansatz vor, der die hieraus ableitbaren Abhängigkeiten formalisiert und für den Planungsprozess nutzt. Solche modellübergreifenden Verknüpfungen bleiben in der Praxis meist unberücksichtigt, da durch die Eigenständigkeit der Fachmodelle keine Einschränkung für die Betrachtung des fachlichen Problems entsteht. Hat beispielsweise der Statiker aus dem Bauwerksentwurf ein statisches System abgeleitet, so kann er allein mit dieser Idealisierung die statische Berechnung durchführen. Die Kommunikation mit anderen Fachplanern wird aus diesem Grund als eigenständiges Problem betrachtet.

Umsetzung der Fachmodelle in eigenständige Programme

Der Modellzustand und das Verhalten des (Fach-)Modells sind dagegen unverzichtbare Informationen, um mit dem Modell arbeiten zu können. Die dafür notwendigen Funktionen, also die Eingabe, Verwaltung und Verarbeitung der Planungsdaten, sind in den Anwenderprogrammen integriert und über Programmiersprachen beschrieben. Die Formalisierung des Fachmodells liegt demzufolge vollständig in der Kontrolle des Programmherstellers, der mit der Wahl einer geeigneten Programmiersprache, einer zweckmäßigen Datenstruktur und der algorithmischen Umsetzung ein wettbewerbsfähiges Produkt schaffen möchte. Das Wissen ist somit implizit im Programmcode enthalten und nicht, wie bei wissensbasierten Systemen angestrebt, explizit in einer Wissensbasis abgelegt (Reimar 1991). Aus einem Fachmodell entstehen dadurch unterschiedliche Formalisierungen, die nicht frei einsehbar und zueinander unverträglich sind. Das abgebildete Fachwissen ist somit nur für den Anwender des Programms nutzbar und schränkt die Kommunikation mit Anwendern anderer Programme ein.

---

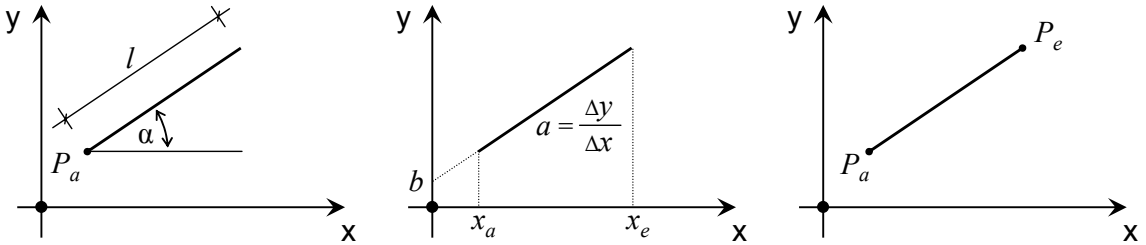
[1] Der Begriff *integrierendes Modell* wird hier stellvertretend verwendet, um die technische Grundlage für den fachübergreifenden Informationsaustausch zu beschreiben.

Kommunikation durch den Austausch von Modellzuständen

Wird die Kommunikation auf den Austausch eines Modellzustandes beschränkt, was derzeit dem Verständnis über den Informationsaustausch zwischen Planern entspricht, dann sind keine Informationen über das Verhalten oder andere Implementierungsdetails notwendig. Vielmehr ist es ausreichend, die Bedeutung der Planungsdaten zu kennen. Beziehen sich die Kommunikationspartner auf das gleiche Fachmodell, so lässt sich die Bedeutung der Planungsdaten aus ihrer Zuordnung zu diesem gemeinsamen Fachmodell ableiten. In der Abbildung 2-2 werden am Beispiel einer *Strecke* die Abhängigkeitsbeziehungen verdeutlicht, um aus den Attributen Anfangspunkt P_a, Winkel α und Streckenlänge l über die im Fachmodell verwendete Streckengleichung $f(x)$ zur Beschreibung der Anfangs- und Endpunkte (P_a und P_e) zu gelangen. Um die Planungsdaten für andere Programme nutzbar zu machen, ist die verwendete Datenstruktur nachvollziehbar zu formalisieren. Es bleibt jedoch den Programmen überlassen, wie die Daten interpretiert und in die eigene Datenstruktur überführt werden.

Möglichkeiten zur Beschreibung einer Strecke

graphische Darstellung



mathematische Darstellung

(P_a, α, l) $\quad f(x) = a*x+b \mid x_a \le x \le x_e$ $\quad (P_a, P_e)$

Funktionale Abhängigkeiten zwischen den Modellen

Modell mit Polarkoordinaten-beschreibung des Endpunktes ⇨ Streckengleichung ⇨ Modell mit Anfangs- und Endpunkt

$a = f(\alpha)$ $\quad x_a = f(P_a)$	$P_a = f(x_a, a, b)$
$b = f(P_a, \alpha)$ $\quad x_e = f(P_a, \alpha, l)$	$P_e = f(x_e, a, b)$

Abbildung 2-2 Darstellungsmöglichkeiten einer Strecke und hieran gekoppelte Abhängigkeiten

Formalisierung der Modelldatenstruktur

Mit der Fokussierung auf den Modellzustand liegt der Schwerpunkt der Formalisierung auf den *Datenmodellen*, die für den Datenaustausch und zunehmend für die Datenverwaltung genutzt werden. Hierbei liegt es nahe, die Datenmodelle aus den Programmen abzuleiten. Dies ist vor allem dann interessant, wenn sie mit Modellierungswerkzeugen konzipiert und formal, beispielsweise über die *Unified Modeling Language* (UML), beschrieben wurden (Rumbaugh, Zuser et al. 2004). Dieser Ansatz wird in der Praxis jedoch kaum verfolgt, da die Ableitung der Datenstruktur nicht ohne Probleme möglich ist. UML und besonders die *Interface Definition Language* (ISO/IEC 14750 1999) haben dagegen eine wesentlich größere Bedeutung für die Definition von Programmschnittstellen, die auf eine programmiersprachliche Integration und somit die Nutzung vorhandener Programmfunktionen ausgelegt ist.

Für die Formalisierung programmunabhängiger Datenmodelle werden Konzepte verwendet, deren Ursprünge in der Wissensrepräsentation und im Datenbankentwurf liegen. Die größte

Bedeutung haben netzartige und schemabasierte Repräsentationsformate erlangt. Hierzu gehören das aus dem Datenbankentwurf stammende *Entity-Relationship Model* (Chen 1976) und der aus der Wissensrepräsentation hervorgegangene *objekt-orientierte Ansatz* (Rumbaugh et al. 1991). Eine Übersicht über hieran angelehnte Repräsentationsformate ist unter anderem in der Arbeit von Hakim (1993) zu finden, die die kontroverse Diskussion über ein angemessenes Format verdeutlicht. In Ingenieuranwendungen ist inzwischen die Beschreibungssprache *EXPRESS* als Standard zur Beschreibung von Datenmodellen akzeptiert und findet breite Unterstützung (ISO 10303-11, 1994).

Die Grundlage von EXPRESS bildet das Entity-Relationship Model, das um objektorientierte Konzepte sowie die Möglichkeit zur Formulierung von Konsistenzbedingungen erweitert wurde. Durch das Bilden von Namensräumen ist zudem die Anlehnung an frameartige Repräsentationsformate erkennbar. Im Vergleich zu objekt-orientierten Programmiersprachen wie C++, Delphi oder JAVA sind die Möglichkeiten zur Beschreibung von Verhalten bei EXPRESS eingeschränkt. Außerdem ist keine *Kapselung* möglich, die ein wesentliches Element des objekt-orientierten Ansatzes darstellt. Andererseits stehen den Programmiersprachen sehr leistungsfähige Vererbungskonzepte und Möglichkeiten für Konsistenzbedingungen gegenüber. Neben der Definition von Regeln (RULES), die beispielsweise die Definition zulässiger Werte oder Wertkombinationen erlauben, können die Relationen zwischen Objekten über ihre Kardinalität und die Definition der Umkehrrelation genauer als in objektorientierten Programmiersprachen beschrieben werden. Damit werden Teile der Forderungen erfüllt, die beispielsweise von Olbrich (1998) formuliert wurden. Andere Forderungen, wie z. B. die Existenzabhängigkeit, komplexe Bindungen, n-äre Assoziation oder attributierte Relationen, werden hingegen nicht erfüllt.

Die Abbildung 2-3 zeigt am Beispiel der Streckenbeschreibung, wie eine Datenstruktur über EXPRESS bzw. EXPRESS-G formalisiert werden kann. In dem gezeigten Beispiel dient die Darstellung mit einem Anfangspunkt P_a, einem Winkel α und einer Streckenlänge l als Grundlage der gewählten Modellierung, die in die zwei Klassen *Strecke* und *Punkt* sowie die notwendigen Attribute, Beziehungen und Konsistenzbedingungen abgebildet wurde.

```
SCHEMA Modell_Strecke;

 ENTITY Strecke;
    Anfangspunkt  : Punkt;
    Winkel        : REAL;
    Laenge        : REAL;
  WHERE
    WR1 : { 0 <= Winkel < 360 };
    WR2 : { Laenge > 0 };
 END_ENTITY;

 ENTITY Punkt;
    X_Koordinate  : REAL;
    Y_Koordinate  : REAL;
 END_ENTITY;

END_SCHEMA;
```

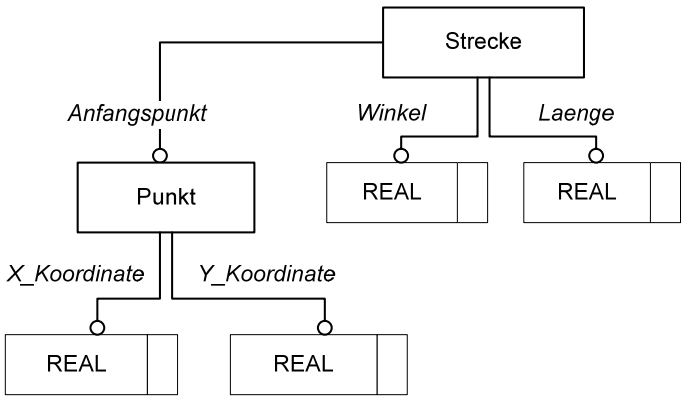


Abbildung 2-3 Formalisierung und grafische Darstellung einer Datenstruktur am Beispiel von EXPRESS und EXPRESS-G

Erhöhung der Ausdrucksstärke

Die Kritiken an EXPRESS zielen auf eine Erhöhung der Ausdruckstärke, um Datenmodelle genauer formalisieren und zusätzliche deskriptive Anmerkungen vermeiden zu können. Solche Vorschläge sind meist durch konkrete Nutzungsinteressen wie beispielsweise die Definition eines bauspezifischen Thesaurus (ISO 12006-3, LexiCon – Woestenenk 2002), die Beschreibung von Bauteilkatalogen (bcXML – van Rees et al. 2002) oder die Strukturierung von Bauprojekten (Neutral Object Tree – Nederveen & Tolman 2001) entstanden. Obwohl sie nutzungsbezogene Datenmodelle darstellen, werden sie durch ihre Allgemeingültigkeit noch als *Meta-Modelle* bezeichnet. Die Grenzen zu *minimalistischen Datenmodellen*, wie beispielsweise von Tarandi (1998) vorgeschlagen, sind jedoch fließend und haben zu Modellarchitekturen geführt, die unterschiedliche Abstraktionsebenen definieren (siehe *7-layer hierarchy* in Katranuschkov (2001)).

Eine parallele Entwicklung stellen Ontologie-Repräsentationsformate wie DAML+OIL (van Harmelen et al. 2001), OWL (W3C 2004) und andere (Gomez-Perez und Corcho 2002) dar, die für die Beschreibung von Inhalten im Internet verwendet werden und durch Anwendung von Schlussfolgerungsmechanismen (inference) eine intelligentere Nutzung der enthaltenen Informationen erlauben (Berners-Lee et al. 2001). Diese Ansätze werden insbesondere für die Definition von *Ingenieurontologien* verwendet, die einen nutzerfreundlichen Zugriff auf Planungsdaten ermöglichen (Katranuschkov et. al 2003, van Rees 2003, Pan & Anumba 2005).

Formalisierung der Datenmodelle als Grundlage für den Datenaustausch

Mit der Formalisierung wird eine Datenstruktur festgelegt, die zusammen mit einer vereinbarten Syntax die Grundlage für den Austausch über einen Modellzustand bildet. Die ISO-Norm 10303-21 (ISO 10303-21, 1994) regelt beispielsweise die Syntax für den dateibasierten Datenaustausch EXPRESS-basierter Modelle. Mit solchen Regelungen sind mitunter Einschränkungen für die austauschbaren Planungsdaten verbunden, die für die Unterstützung der Kooperation nachteilig sind. So ist der dateibasierte Datenaustausch nach ISO-10303-21 nur für die Übertragung vollständiger Planungsdaten geeignet, da die referentielle Integrität, die über dateispezifische Objektschlüssel erreicht wird, nur innerhalb einer Datei gewährleistet ist (siehe Abbildung 2-4). Die Einigung auf eine geeignete Syntax ist somit weniger ein konzeptionelles, sondern vielmehr ein technisches Problem, das den Zugriff auf die Modelldaten regelt.

```
ISO-10303-21;

HEADER;
FILE_DESCRIPTION(('Streckenbeispiel'), '2;1');
FILE_NAME('Strecke.spf','2005-05-10T18:34:00',('Nutzer X'),('Firma Y'),'',
          'Programm Z','Adresse');
FILE_SCHEMA(('MODELL_STRECKE'));
ENDSEC;

DATA;
#1=STRECKE(#3, 35.0, 10.0);
#2=STRECKE(#3, 0.0, 5.0);
#3=PUNKT(1.5, 2.0);
ENDSEC;

END-ISO-10303-21;
```

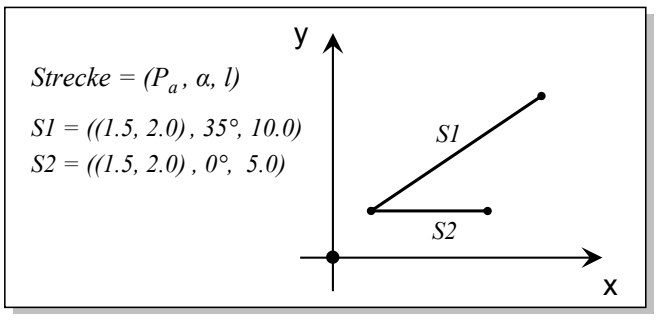


Abbildung 2-4 Beispiel für den dateibasierten Datenaustausch nach ISO-10303 Teil 21

Aus den Problemen der Formalisierung wird deutlich, dass der überwiegende Teil des abgebildeten Modellwissens in den Programmen fest integriert ist. Der Informationsaustausch und die hierfür verwendeten Datenmodelle sind dagegen auf die Beschreibung eines Modellzustandes ausgelegt.

2.1.3 Ansätze der Datenintegration

Ein Bauwerksentwurf wird durch die Gesamtheit der Planungsdaten beschrieben. Aufgrund der Modellvielfalt sind gleiche Informationen in unterschiedlicher Form und anderen fachlichen Zusammenhängen oft mehrfach vorhanden. Aus diesen Redundanzen können Widersprüche entstehen, die mit Hilfe der *Datenintegration* vermieden werden sollen. Verändert der Architekt beispielsweise die Größe eines Raumes, so muss die resultierende Stützweite auch im Tragmodell aktualisiert werden. Die Redundanzen, die zwischen oder auch innerhalb von Modellen auftreten, sind aber nicht immer so offensichtlich wie in diesem Beispiel und führen oft zu komplexen Abhängigkeitsbeziehungen.

Das Ziel der Datenintegration ist die reibungslose Kommunikation über den gemeinsamen Planungsgegenstand, die traditionell über eine gemeinsame Modellvorstellung, wie beispielsweise die Technische Zeichnung, erfolgt. Diese Arbeitsweise wurde zu Beginn der Computernutzung mit neutralen Datenformaten wie dem *Drawing Exchange Format* (DXF, Rudolph et al. 1993) oder IGES (Nagel et al. 1980) auf der Ebene einfacher Zeichnungselemente adaptiert. Die Semantik der Technischen Zeichnung wurde damit aber nicht annähernd in einer für Programme auswertbaren Form erfasst (Koutamanis 1990). Aus der anfänglichen Idee, die Zeichnungselemente mit zusätzlichen Eigenschaften zu verknüpfen, hat sich schließlich das Konzept der heute favorisierten *Produktdatenmodelle* entwickelt, das sich von der zeichnerischen Darstellung als Bezugsebene der Produktinformationen gelöst hat (Reed 1988, Bjork 1995, Eastman 1999).

Die Idee der Produktdatenmodelle

Über die Modellierung von Produktinformationen gibt es eine Vielzahl von Vorschlägen, die unterschiedliche Vorstellungen über eine geeignete Modellstruktur und den Umfang des abzubildenden Wissens erkennen lassen (Luiten et al. 1993, Ekholm 1994, Hannus et al. 1995). Die Diskussion wird zusätzlich erschwert, weil sich noch keine einheitliche Begriffsdefinition durchgesetzt hat und konzeptionell eine klare Abgrenzung zur Datenmodellierung fehlt. Im allgemeinen kann man ein Produktdatenmodell als eine Datenstruktur ansehen, die Informationen über ein konkretes Produkt abbilden kann. Ein Produktdatenmodell wird für einen bestimmten Produkttyp geschaffen und gibt dadurch den Anwendungsbereich vor. So kann es sinnvoll sein, für den Bauwerkstyp Bürogebäude ein anderes Produktdatenmodell als für Industriehallen zu verwenden.

Mit der Definition eines Produktdatenmodells wird das Ziel verfolgt, eine standardisierte und für den Anwendungsbereich möglichst vollständige Abbildung der Produktinformationen zu erreichen. Ein Produktdatenmodell wird dadurch als eine neutrale, allgemein interpretierbare Form angesehen, die für den Austausch und die Verwaltung der Produktinformationen genutzt werden kann (Fischer & Froese 1992). Es wird in der Fachwelt jedoch mehrheitlich bezweifelt, dass in einem Produktdatenmodell alle Produktinformationen, die ja selbst das Ergebnis einer auf Modellen beruhenden Vorstellung über das Produkt sind, in einer einheitlichen und redundanzfreien Datenstruktur abgebildet werden können (Katranuschkov 1995, Wix 1996, Junge & Liebich 1998, Turk 2001, Stouffs & Krishnamurti 2001). Ein Produktdatenmodell stellt daher einen Kompromiss zwischen dem geforderten Informationsgehalt und

einer handhabbaren Datenstruktur dar. Soll ein Produktdatenmodell bewertet werden, so rückt zwangsläufig der vorgesehene Anwendungsbereich in den Mittelpunkt der Betrachtung. Hierbei sind Tendenzen in zwei Richtungen erkennbar. Entweder liegt der Schwerpunkt auf der Integration der fachübergreifend benötigten Bauwerksinformationen oder aber auf der umfassenden Abbildung fachspezifischer Details.

Integration wesentlicher Bauwerksinformationen in einem Modell

Die Integration wesentlicher Bauwerksinformationen in *einem* Modell, das auch als *Building Information Model* (Fehringer 2004) bezeichnet wird, stellt eine Grundlage dar, um den Bauwerksentwurf zwischen verschiedenen Fachplanern auszutauschen. Durch das Zusammenführen der verschiedenen Fachinformationen lassen sich Widersprüche im Bauwerksentwurf wesentlich einfacher erkennen und haben zu einem hohen Forschungsinteresse zu Beginn der 90er Jahre geführt. In Forschungsvorhaben wie beispielsweise COMBINE (Augenbroe 1995), COMBI (Scherer 1995), ATLAS (Tolman & Poyet P. 1995) oder VEGA (Junge et al. 1997) wurden grundlegende Fragen über den Aufbau fachübergreifend nutzbarer Produktdatenmodelle betrachtet. Es wurden Ansätze entwickelt, um die komplexen Bauwerksinformationen in handhabbare Datenstrukturen abzubilden sowie nachträgliche Modellerweiterungen zu ermöglichen (Luiten et al. 1991). Als wichtige Instrumente haben sich die Wiederverwendbarkeit allgemeingültiger Definitionen, wie z.B. die Beschreibung von Einheiten oder der Geometrie beliebiger Körper, sowie die Abstraktion, d.h. der Aufbau einer sich zunehmend spezialisierenden Klassenhierarchie, erwiesen (Bjork 1995, siehe Abbildung 2-5).

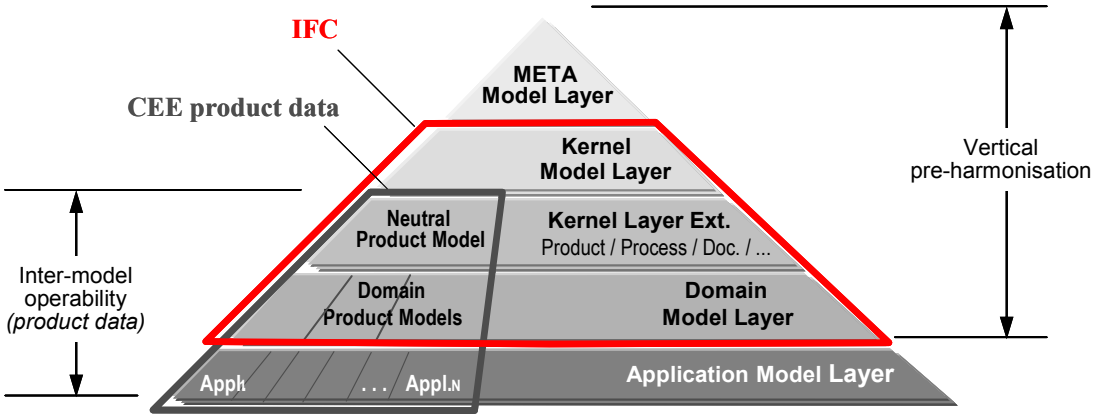


Abbildung 2-5 Integrationsansatz nach Katranuschkov/Scherer und Einordnung des IFC-Modells

Aus der Vielzahl der Bemühungen ist mit den *Industry Foundation Classes* (IFC) inzwischen ein Industriestandard für den Bereich Hochbau hervorgegangen, der den Austausch grundlegender Bauwerksinformationen ermöglicht und von zahlreichen Softwarefirmen unterstützt wird (Wix & Liebich 2003). Die Anwendung konzentriert sich derzeit auf die Architekturplanung und Technische Gebäudeausrüstung und soll schrittweise auf andere Fachbereiche ausgeweitet werden. Trotz dieser Fokussierung, die erst einen Teil der beabsichtigten Anwendungsbereiche berücksichtigt, werden durch verfügbare Implementierungen die Probleme des Ansatzes deutlich. So wird durch die modulare Datenstruktur ein höherer Implementierungsaufwand notwendig, da die Modelldefinition nicht auf fachspezifische Anforderungen optimiert werden kann. Für die programminterne Nutzung müssen die Planungsdaten daher in eine problemorientierte Darstellung überführt werden. Es zeigt sich auch, dass der Gesamtdatensatz für die Bearbeitung in handhabbare Teildatensätze zerlegt werden muss (Steinmann 2005). Dies erzeugt zusätzliche Probleme hinsichtlich der referentiellen Integrität der Pla-

nungsdaten und erfordert eine hohe Qualität der Implementierungen. Bazjanac (2002, 2005) geht auf erste praktische Erfahrungen mit dem IFC-Modell und den verfügbaren Implementierungen ein. Trotz der vorgebrachten Kritiken, die vielerlei Ursachen haben und meist in den Besonderheiten des Bauwesens begründet sind, stellt Bazjanac das zugrunde liegende Konzept nicht in Frage.

Modellverbund

Als Alternative zum obigen Ansatz, der einen hohen Modellierungs- und Implementierungsaufwand erfordert und nicht alle Planungsdaten zweckmäßig integrieren kann, wird die Definition von fach- bzw. aufgabenspezifischen Modellen verfolgt. Die Modelldefinition ist durch das klare Anwendungsziel wesentlich einfacher und kann für die vorgesehene Anwendung optimiert werden. Durch den Verzicht auf ein Integrationsmodell werden die Planungsdaten auf mehrere Modelle verteilt und es entstehen zwangsläufig Redundanzen, die während der Projektbearbeitung widerspruchsfrei bleiben sollen. Der hierfür erforderliche Datenabgleich basiert auf Verknüpfungsregeln, die zusätzlich zwischen den voneinander abhängigen Daten formalisiert werden müssen (siehe Abbildung 2-6). Dadurch wird einerseits die Implementierung der Programme wesentlich vereinfacht. Andererseits ist ein erheblicher Aufwand für die Definition der Verknüpfungsregeln und die entsprechende Datenverwaltung erforderlich.

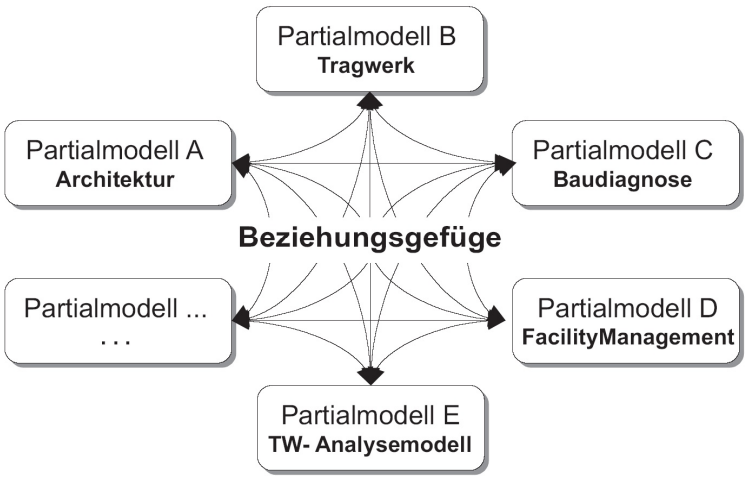


Abbildung 2-6 Verbund von Partialmodellen zu einem logischen Bauwerksmodell (Willenbacher 2001)

Ein Vertreter dieses Ansatzes ist die ISO-Norm 10303 (ISO 10303-1, Froese 1996), die verschiedene Anwendungsprotokolle für die Beschreibung eines Produktes vorsieht[1]. Auch wenn in dieser Norm die Integration durch das Prinzip der Wiederverwendbarkeit stark begünstigt wird und konkrete Integrationsansätze wie beispielsweise von Gonçalves (2004) vorliegen, so ist der Datenaustausch in der Regel auf das zuvor definierte, durch das Nutzungsszenario beschriebene Anwendungsziel beschränkt. Es zeigt sich, dass die Verknüpfung der Modelle ein wesentliches Hindernis für die Datenintegration darstellt und nach wie vor Gegenstand der Forschung ist. Willenbacher (2002) beschäftigt sich in seiner Arbeit mit einer Integrationsplattform, die eine interaktive Verknüpfung der im Lebenszyklus eines Bauwerks verwendeten Modelle ermöglicht. Im Idealfall können so die genutzten Modelle von der Problematik

[1] Für das Bauwesen existieren die Anwendungsprotokolle 225 – *Structural Building Elements Using Explicit Shape Information* (Haas 1998), 230 – *Building Structural Frames: Steelworks* (Watson & Ward 1995) und eine als *Construction Drawing Subset* bezeichnet Untermenge von AP214 (Haas 2000).

der Datenintegration vollständig entkoppelt werden, da die Aktualisierung der Daten durch die Integrationsplattform geleistet wird. In der Praxis haben solche Integrationsansätze aufgrund der sehr komplexen Verknüpfungsregeln aber noch keine praktische Anwendung gefunden (van Nederveen & Tolman 2001) [1].

Zusätzliches Wissen als Voraussetzung der Datenintegration

Durch die Gegenüberstellung der beiden prinzipiellen Integrationsansätze wird deutlich, dass die Datenintegration nur durch das Einbringen zusätzlichen Wissens erreicht werden kann. In Abhängigkeit vom verfolgten Ansatz liegt dieses Wissen zu einem unterschiedlichen Grad in Form von standardisierten Modelldefinitionen und ergänzenden Verknüpfungs- bzw. Transformationsregeln vor. Bei der Bewertung der Integrationsansätze geht es daher weniger um die „Menge“ des benötigten Wissens, sondern vielmehr um die Aufteilung, Beschreibung und Anwendung des Wissens. Gegenwärtig wird in der Praxis der Weg der Standardisierung favorisiert, weil dadurch der Bezug auf ein Modell begrenzt und die Interpretation der Planungsdaten den Anwenderprogrammen überlassen werden kann. Bei dieser Lösung werden wesentliche Teile des Integrationswissens in die Anwenderprogramme verlagert und der gemeinsame Integrationsaufwand auf die Definition einer geeigneten Datenstruktur beschränkt. Auf diese Weise bleibt es den Anwenderprogrammen überlassen, wie ein solcher Datensatz intern verarbeitet wird. Die auftretenden Probleme sind jedoch für alle Anwenderprogramme gleich und umfassen den Zugriff auf die relevanten Informationen, das Überführen in die intern genutzte Daten- bzw. Programmstruktur und schließlich die Aktualisierung des gemeinsamen Datensatzes. Die Qualität der Datenintegration wird dadurch zu einem großen Teil von den Anwenderprogrammen bestimmt.

2.1.4 Datentransformationen

Durch die Vielfalt der genutzten Modelle und die auftretenden Redundanzen werden gleiche Produktinformationen in unterschiedlichen Darstellungsformen beschrieben. Diese Darstellungsformen müssen ineinander überführbar sein, um den Informationsaustausch zwischen den Modellen zu ermöglichen. Die Transformation der Daten ist somit die Voraussetzung für eine widerspruchsfreie Produktbeschreibung und zählt dadurch zu den wichtigsten Aufgaben der Datenintegration.

Probleme der Datentransformation

Das Beispiel der Abbildung 2-2 zeigt drei Möglichkeiten zur Beschreibung einer Strecke. Die zugehörigen Transformationsregeln lassen sich in einfache mathematische Gleichungen fassen. Wird beispielsweise die Länge l einer Strecke verändert, so müssen in den anderen Darstellungsformen die Grenze x_e und der Endpunkt P_e aktualisiert werden. Es ist offensichtlich, dass diese Abhängigkeiten nicht auf eine Abbildungsrichtung beschränkt sind. Die Umkehrrichtung kann aber nicht immer vollständig beschrieben werden und ist im Sinne einer Abbildung nicht mehr eineindeutig. Ist beispielsweise der Querschnitt eines Trägers gegeben, so kann das Trägheitsmoment abgeleitet werden. Umgekehrt ist dies nicht möglich. Folglich tritt ein Informationsverlust auf, der ohne Zusatzinformationen nicht ausgeglichen werden kann.

---

[1] Van Nederveen und Tolman schlussfolgern aus dem Projekt ATLAS: "*As shown in this project, the big practical issue in the view approach is that the relationships between the various view-specific information models (the view conversion) become too complex, leading to too costly implementation and maintenance of conversion software.*"

Neben dem Verlust an Informationen stellt die Komplexität der Transformationsregeln das zweite wesentliche Problem dar. Einerseits ist die Vielzahl der Abhängigkeiten zu erfassen und in Transformationsregeln zu beschreiben. Andererseits können Transformationsregeln sehr komplex werden. Neben einer aufwendigen mathematischen Beschreibung ist hierfür oft zusätzliches Ingenieurwissen und die Kenntnis über getroffene Entwurfsentscheidungen erforderlich. So können aus einer genormten Materialbezeichnung die zugehörigen Materialkennwerte nur mit Hilfe der entsprechenden Norm ermittelt werden. Entwurfsentscheidungen werden demgegenüber relevant, wenn Datentransformationen nur aufgrund bereits bestehender Verknüpfungsbeziehungen möglich sind. Die Nutzungsänderung eines Raumes, die im Architekturmodell durchgeführt wird, kann beispielsweise nur dann auf die resultierende Verkehrslast im mechanischen Modell übertragen werden, wenn der Bezug zur mechanischen Idealisierung des Tragwerksplaners bekannt ist. Dadurch werden neue bzw. bisher nicht abgebildete Produktinformationen geschaffen, die zwangsläufig auch verwaltet werden müssen.

Aus diesen Beispielen werden die prinzipiellen Problemstellungen der Datentransformation deutlich, die je nach Anwendungsszenario eine unterschiedliche Bedeutung besitzen. In der Praxis werden überwiegend dateibasierte Schnittstellen eingesetzt, die vollständige Datensätze zwischen zwei Datenmodellen mit vergleichbarem Informationsgehalt transformieren. Dadurch können die Transformationsregeln auf der Grundlage der beteiligten Datenmodelle vorab und für die betrachtete Richtung mehrheitlich eindeutig beschrieben werden. In diesem Szenario geht es daher primär um die effiziente Formalisierung und Ausführung der Transformation, die sich aus einer Vielzahl von Regeln zur Überwindung der auftretenden Beschreibungs- und Strukturkonflikte zusammensetzt. Hierfür können sogenannte *Mappingsprachen* eingesetzt werden, die zunächst eine Trennung zwischen der Beschreibung und Implementierung der Transformation vornehmen. Zusätzlich werden Sprachkonstrukte angeboten, die die Definition der Transformationsregeln vereinfachen. Gegenüber einer direkten Umsetzung im Programmcode liegen die Vorteile dieses Vorgehens vor allem in der vereinfachten Beschreibung und Wartung der Transformationsregeln, der besseren Aufgabenteilung zwischen den beteiligten Fachingenieuren und Programmierern sowie der Reduzierung des Implementierungsaufwandes.

Mappingsprachen

Katranuschkov (2001) beschäftigt sich in seiner Arbeit ausführlich mit diesem Thema und stellt verschiedene Mappingsprachen wie XP-Rule (Zarli 1995), Operation Mapping (Bijnen 1995), View Mapping Language (Amor 1997), EXPRESS-X (Denno, 1999) und indirekt relevante Ansätze wie ACL/KIF (Genesereth und Fikes 1992) mit ihren jeweiligen Besonderheiten vor. Doch auch mit diesen Formalisierungsansätzen bleibt die Beschreibung einer Datentransformation sehr aufwendig. Dies zeigt das Projekt ST-1 der IAI (Karstila 2001), das die Übertragung von Geometrieinformationen zwischen den Modellen CIMSTEEL (Eastman 2001) und IFC (Wix & Liebich 2003) unter Verwendung von EXPRESS-X zum Ziel hatte (CIS2IFC 2004). Katranuschkov setzt in seiner *Context-Independent Schema Mapping Language* (CSML) daher den Schwerpunkt auf eine Vereinfachung der Transformationsbeschreibung, die durch die Verwendung von Transformationsmustern erreicht wird.

Datentransformationen als Bestandteil der Datenverwaltung

Datentransformationen werden für die beteiligten Datenmodelle in der Regel vordefiniert und sind für alle zugehörigen Datensätze nutzbar. Die Transformation der Daten kann dadurch als ein eigenständiger Dienst realisiert werden, der nach Bedarf ausgeführt werden kann. Ein sol-

cher Ansatz stößt jedoch schnell an Grenzen, da nur eindeutig abbildbare Informationen übertragen werden können. Um den Verlust von Informationen zu vermeiden sowie die gezielte Aktualisierung einzelner Daten zu ermöglichen, werden Lösungen notwendig, die mit der Verwaltung der Daten gekoppelt sind. Eine solche Kopplung wird von verschiedenen Forschungsvorhaben verfolgt und ist unter anderem in EDM-2 (Eastman et al. 1995b, Eastman & Jeng 1999) und den Arbeiten von MacKellar & Peckam (1998), Sriram (2002) und Haymaker et al. (2000, 2005) zu finden. Auf diese Weise entstehen komplexe Datenverwaltungssysteme, die aus der Summe der Planungsdaten die gewünschten Informationen in der benötigten Datenstruktur bereitstellen und weit über die Probleme der Datentransformation hinausgehen. Dieser Ansatz ist auch in relationalen Datenbanken[1] zu finden, die aus einem Tabellensatz verschiedene Sichten (*views*) generieren und für die Bearbeitung bereitstellen können. Ein solches Vorgehen ist jedoch an Bedingungen gekoppelt, die die nutzbaren Sichten und den Zugriff auf die Daten reglementieren. Die Anwendung beschränkt sich dadurch auf vergleichsweise einfache Problemstellungen wie Bestell- oder Reserviersysteme, die zwar stets die Konsistenz der Daten garantieren müssen, dafür aber einen festgelegten Informationsbedarf und hochgradig standardisierbare Abläufe aufweisen. Aus den geschilderten Anwendungsfällen werden die Gegensätze zu der im Bauwesen vorhandenen Datenkomplexität und dem gewohnten Planungsablauf deutlich. Soll dieser Ansatz als ein Vorbild für die Datenintegration im Bauwesen dienen, so sind zwangsläufig tiefgreifende Veränderungen notwendig.

2.2 Kooperationsstrategien

Für die kooperative Planung ist neben der Kommunikation, die den Austausch von Informationen ermöglicht, die Koordination der Abläufe und der daran gekoppelten Informationsflüsse notwendig (Meißner & Rüppel 2003). Diese allgemeingültige Erkenntnis beschreibt das Wesen der Zusammenarbeit und wird unabhängig von der eingesetzten Technologie in jeder Planung praktiziert. Durch die HOAI-Phasen werden beispielsweise die durchzuführenden Arbeiten und deren Abfolge grob beschrieben. Dieses Gerüst wird projektbezogen verfeinert und in konkrete Aufgaben für die beteiligten Planer unterteilt. Dadurch ist es möglich, mit allen Beteiligten das gemeinsame Projektziel zu erreichen.

Mit der computergestützten Kooperation werden kürzere Planungszeiten, eine höhere Planungsqualität und niedrigere Planungskosten angestrebt. Eine geeignete Kooperationsstrategie soll diese widersprüchlichen Ziele miteinander kombinieren. Kürzere Planungszeiten erfordern beispielsweise verstärkt paralleles Arbeiten, das gleichzeitig die Gefahr inkonsistenter Planungsdaten erhöht und einen häufigeren Abgleich der Planungsdaten erfordert. Hieraus wird deutlich, dass Kooperationsstrategien nach unterschiedlichen Kriterien bewertet werden können und eng mit der Problematik der Kommunikation verbunden sind.

Die Koordination wird nachfolgend unter dem Aspekt der Datenverwaltung betrachtet, die paralleles Arbeiten als wesentlichen Bestandteil der kooperativen Planung ansieht. In der Datenbanktheorie ist das damit verbundene Problem als Zugriffskontrolle (*concurrency control*) bekannt. Die hieran gekoppelte Organisation der Datenzugriffe entscheidet über Einschränkungen des Planungsablaufes bzw. zusätzlich notwendige Maßnahmen zur Wahrung bzw. Wiederherstellung der Konsistenz. Letztendlich interessiert, wie die Sicht der Datenverwaltung mit den Prinzipien der Projektsteuerung zusammengeführt werden kann.

---

[1] Eine Verallgemeinerung dieser Problematik stellen *föderierte Datenbanken* dar, die den Zusammenschluss verteilter Datenbanken verfolgen (Motro 1987, Shet & Larson 1990).

2.2.1 Transaktionen

In der Datenbanktheorie werden Datenänderungen über *Transaktionen*[1] ausgeführt, die eine konsistente[2] Zustandstransformation vornehmen. Eine Transaktion muss dafür atomar, konsistent, isoliert und dauerhaft in einem Datenbankmanagementsystem ausführbar sein. Diese Bedingungen werden unter dem Begriff *ACID* zusammengefasst und bilden die Grundlage für konsistente Datenänderungen (Härder & Reuter 1983, Vossen 1994). Eine Transaktion kann dabei aus mehreren Datenbankoperationen bestehen und wird entweder vollständig oder gar nicht ausgeführt. Dadurch verhält sie sich wie eine einzelne (atomare) Datenbankoperation, die ihre interne Struktur gegenüber anderen Transaktionen verbirgt. Transaktionen können deshalb *nicht parallel* ausgeführt werden und erzwingen formal ein sequenzielles Arbeiten. In der Datenbanktheorie wird auf technischer Ebene auch das parallele konditionale Ausführen von Transaktionen betrachtet (*Serialisierbarkeit*). Bei dieser Nebenläufigkeitskontrolle wird geprüft, ob die Datenbankoperationen zueinander in Konflikt stehen. Treten keine Konflikte auf oder sind die Transaktionen zeitlich ausreichend kurz, so kann die Konsistenz der Daten unter quasi parallelem Arbeiten gewährleistet werden (Elmasri & Navathe 2002).

Mehrere Eigenschaften machen die Anwendung *kurzer Transaktionen* für typische Entwurfsaufgaben des Ingenieurwesens ungeeignet. Neben der häufig notwendigen Aktualisierung lokal vorhandener Sichten ist durch die Dauer eines „konsistenten" Entwurfsschrittes ein quasi paralleles Arbeiten nicht möglich. Diese Problematik wird mit dem Begriff der *langen Transaktionen* beschrieben (Herrmann 1991, Katranuschkov 2001) und stellt eine Besonderheit des Ingenieurwesens dar. Eine lange Transaktion ist durch eine Bearbeitungszeit von mehreren Stunden oder Tagen und umfangreiche Änderungen gekennzeichnet. Soll das parallele Arbeiten mit langen Transaktionen unterstützt werden, so sind konservative oder optimistische Kooperationsstrategien zur Vermeidung gegenseitiger Konflikte denkbar.

2.2.2 Konservative Kooperationsstrategie

Das Ziel der konservativen Kooperationsstrategie ist das vorausschauende Verhindern von Konflikten, die bei einer unabhängigen, in lange Transaktionen gekapselten Bearbeitung entstehen können. Hierbei wird der Ansatz verfolgt, die parallele Bearbeitung sich gegenseitig beeinflussender Planungsdaten zu verhindern. Dies geschieht durch *Sperren*, die für die Zeit der Bearbeitung auf aufgabenrelevante Planungsdaten gesetzt werden. Für die Dauer der Transaktion kann dieser Teil der Planungsdaten durch andere Planer weder modifiziert noch gesperrt werden. Das parallele Arbeiten wird somit auf Aufgaben beschränkt, die sich gegenseitig nicht beeinflussen.

Generell ist das Sperren ein sehr wirkungsvolles Mittel, um inkonsistente Planungsdaten zu verhindern. Das Problem dieser Strategie besteht darin, die Sperren angemessen zu setzen. Im Idealfall wird durch das Sperren nur der Teil der Planungsdaten geschützt, der für das Ausführen der Aufgabe notwendig ist. Zu Beginn eines Planungsschrittes kann dieser Teil aber nur selten genau bestimmt werden und ist, sofern bekannt, häufig schwer beschreibbar. Dies

---

[1] „A transaction is a unit of work that is atomic from the view of the enterprise." (Date 1981)

[2] Der Begriff der Konsistenz ist nicht nur auf die Struktur der Daten beschränkt sondern schließt formal auch die Semantik der Daten ein. In komplexen Datenbankschemen sind die hierfür notwendigen Konsistenzbedingungen aber nicht mehr formal beschreibbar. Dies hat unter anderem dazu geführt, dass die Konsistenz der Daten mitunter über den Begriff der Transaktion, der ja eine konsistente Datentransformation beschreibt, definiert wird (Lockemann & Schmidt 1987).

liegt unter anderem an den nur unvollständig erfassten Abhängigkeiten der Daten. Darüber hinaus ist oft eine sehr differenzierte Sicht zu sperren. Soll beispielsweise die Raumaufteilung eines einzelnen Geschosses verändert werden, so sind die raumbegrenzenden Wände zu sperren. Ein konsistenter Entwurfsschritt ist aber nur dann gegeben, wenn auch die Position und die geometrischen Abmessungen der lastabtragenden Stützen beibehalten werden. Diese Stützen brauchen jedoch nicht vollständig gesperrt zu werden, da andere Parameter, z. B. der Bewehrungsgrad und das Material, keinen Einfluss auf die Anordnung der Wände haben. Dies führt entweder zu großzügigen oder sehr aufwendigen Sperren, die beide das parallele Arbeiten behindern und den Planungsablauf nur schwer planen lassen.

Als Alternative wird die sukzessive Vergabe von Objektsperren verfolgt. Dieser Ansatz setzt voraus, dass das dynamische Nachladen und Aktualisieren der Daten unterstützt wird. Für die komplexen Abhängigkeiten zwischen den Datenmodellen der Bauplanung ist diese Voraussetzung aber nur unter idealen Bedingungen erfüllt. Zusätzlich besteht die Gefahr, dass Planungsschritte sich gegenseitig durch das Anfordern gleicher Sperren blockieren.

Mit der konservativen Kooperationsstrategie lassen sich unbeabsichtigte Inkonsistenzen aus paralleler Bearbeitung verhindern. Durch die arbeitsteilige Organisation der Planung, die nach einer Entwurfsänderung einen fachlich konsistenten Planungsstand häufig nur in Zusammenarbeit mit anderen Fachplanern wiederherstellen kann, kann die in der Datenbanktheorie mit einer Transaktion assoziierte Bedingung der Konsistenz a priori jedoch nicht garantiert werden. Dadurch entstehen zwangsläufig temporär inkonsistente Planungsstände, die von der Datenbasis ebenso verwaltet werden müssen.

2.2.3 Optimistische Kooperationsstrategie

Die optimistische Kooperationsstrategie verzichtet auf das Sperren von Planungsdaten und erlaubt dadurch hochgradig paralleles Arbeiten. Bei diesem Ansatz wird vorausgesetzt, dass durch eine sinnvolle Anordnung der Aufgaben Konflikte vermieden werden. Im Gegensatz zur konservativen Kooperationsstrategie werden die Änderungen erst beim Aktualisieren der Datenbasis auf Verträglichkeit mit parallel durchgeführten Transaktionen überprüft. Werden die Verträglichkeitsbedingungen verletzt, so wird die Transaktion verworfen[1]. Für lange Transaktionen, komplexe Datenstrukturen und temporär auftretende Inkonsistenzen ist diese datenbankorientierte Sichtweise nicht praktikabel. Ohne das Überwachen der gegenseitigen Änderungen können dagegen aufgabenrelevante Entwurfsentscheidungen unbemerkt verändert bzw. parallel durchgeführte Planungsleistungen unbeabsichtigt überschrieben werden. Die Integrität der Änderungen ist in diesem Fall nicht gesichert und führt zu einem kaum nachvollziehbaren Planungsprozess.

Um die Vorteile des uneingeschränkten parallelen Arbeitens nutzen zu können, wird der optimistische Kooperationsansatz mit Möglichkeiten zum Erkennen und Auflösen von Konflikten kombiniert. Hierbei werden die parallel entstandenen Planungsstände durch die beteiligten Planer kontrolliert zusammengeführt (Bentley 1998). Dieser als *Merging* bezeichnete Schritt

[1] Die optimistische Kooperationsstrategie wird in der Datenbanktheorie unter dem Aspekt der Transaktion und dem zugrunde liegenden ACID-Prinzip betrachtet. Wird eine Transaktion gestartet und die Datenbasis danach verändert, so können die Voraussetzungen für das Abschließen der Transaktion verletzt werden. In diesem Fall kann die begonnene Transaktion die Datenbasis nicht in einen konsistenten Zustand überführen und wird vollständig verworfen. In der Literatur wird daher auch der Begriff der *optimistischen Sperren* verwendet. Das Ausführen einer Transaktion kann durch dieses Vorgehen wiederholt verhindert werden und führt zu dem sogenannten *starvation problem* (Unland 1994).

erzeugt einvernehmlich einen neuen Planungsstand, indem aufgetretene Konflikte in einem Abstimmungsverfahren beseitigt werden. Durch zusätzlich erforderliche Nutzerinteraktionen wird der weitere Planungsablauf jedoch verzögert und der durch paralleles Arbeiten gewonnene Zeitvorteil relativiert. Ein unmittelbares Auflösen der Konflikte ist aber nicht immer möglich oder zwingend notwendig. Um das Auflösen von Konflikten flexibel und ohne Behinderungen für den weiteren Planungsablauf gestalten zu können, ist die Verwaltung alternativer Planungsstände bzw. entstandener Konflikte notwendig (*managed inconsistencies*). Bei diesem Vorgehen besteht jedoch die Gefahr, dass sich Konflikte in der weiteren Arbeit propagieren und ein größerer Teil der Planung wiederholt werden muss.

Die optimistische Kooperationsstrategie orientiert sich folglich am stärksten an der heute üblichen Organisation der Planung. Diese Art der Zusammenarbeit ist jedoch nur dann sinnvoll, wenn durch die vorausschauende Koordination der Arbeitsabläufe das Entstehen von Konflikten beschränkt werden kann (Hauschild 2003).

2.2.4 Koordination der Planungsschritte

Die vorgestellten Kooperationsstrategien stellen prinzipielle Ansätze für das parallele Arbeiten auf einer gemeinsamen Datenbasis dar. Diese Ansätze beschäftigen sich nicht mit der Organisation der Planung, die für das Erreichen des Planungszieles notwendig ist. Das Erfassen und die Kontrolle des Planungsablaufs sind vielmehr Themen der Prozessplanung, die in der Regel auf einer abstrakten Ebene von „Aufgaben“ durchgeführt wird und dadurch unabhängig von der Datenverwaltung betrachtet werden kann. Eine Aufgabe symbolisiert hierbei eine beliebige Tätigkeit, die zum Erreichen des Planungszieles notwendig ist.

Eine Übersicht über die Themen der Prozessplanung und den Stand der Technik sind unter anderem in Greb & Klauer (2004) und Keller et al. (2004) zu finden. Zu diesen Themen gehören unter anderem die Beschreibung und Optimierung des Planungsablaufes, die Überwachung des Planungsfortschritts sowie die Unterstützung der unternehmensübergreifenden Zusammenarbeit. Auch wenn sich hieraus sehr unterschiedliche Problemstellungen ergeben, so kann die Umsetzung in der Regel auf die Prinzipien der im Baubetriebswesen eingesetzten Ablaufplanung zurückgeführt werden (Schwarze 2001). Der Planungsablauf wird dabei als eine Menge von Aufgaben betrachtet, die aufgrund der Aufgabenabhängigkeiten eine interne Ordnung besitzt. Mathematisch lässt sich diese Struktur als gerichteter Graph beschreiben, wobei ein Knoten des Graphen je nach Ansatz eine Aufgabe (Vorgang) oder einen Zustand darstellt. Neben der Beschreibung der Ordnungsstruktur werden Aufgaben und Abhängigkeiten um weitere Angaben ergänzt, die beispielsweise das Erfassen von Terminen, die Verknüpfung von Aufgaben und Personen oder die Formulierung zusätzlicher Ablaufbedingungen ermöglichen. Auf diese Weise lässt sich der Planungsablauf beliebig detailliert auf der Ebene von Aufgaben beschreiben.

Kopplung zwischen Datenverwaltung und Prozessplanung

Die meist unabhängig von der Produktdatenverwaltung betrachtete Prozessplanung legt über Aufgaben die durchzuführenden Tätigkeiten und benötigten Daten fest. Solche Angaben können je nach Ansatz unterschiedlich genau erfasst werden und bieten auf diese Weise die Möglichkeit, die Produktdatenverwaltung mit der Prozessplanung zu verbinden. Bereits auf der Basis der Aufgabenabhängigkeiten sowie einer einfachen Verknüpfung zwischen der Aufgabe und den erzeugten Daten kann die Weitergabe der Informationen, beispielsweise auf der Ebene von Dokumenten, organisiert werden (Rüppel 1998). Das Erfassen dokumentorientierter Arbeitsabläufe ist das Ziel klassischer *Workflow-Systeme*, die häufig wiederkehrende Vorgän-

ge mit standardisierbaren Informationsflüssen unterstützen (Vossen & Becker 1996). Sollen aufgabenrelevante Planungsdaten genauer erfasst werden, um z. B. den selektiven Zugriff auf eine zentrale Datenbasis zu unterstützen, so sind zusätzliche Angaben notwendig. Hierbei wird eine möglichst effiziente Lösung angestrebt, um den Mehraufwand bei der Ablaufplanung zu begrenzen. Dies wird typischerweise über Zugriffsrechte erreicht, die den Bearbeitern einmalig zugeordnet werden und dadurch den Informationsbedarf ihrer Aufgaben implizit festlegen (Yum & Drogemuller 1997, Wasserfuhr & Scherer 1999). Eine differenziertere, aufgabenspezifische Beschreibung wird hingegen nur dann verfolgt, wenn der höhere Aufwand über vordefinierte *Referenzprozesse* kompensiert werden kann. Dies setzt die Standardisierung häufig wiederkehrender Teilabläufe voraus und ist ein aktuelles Thema der Forschung (Björk 1999, Katranuschkov et al. 2004, Scheer et al. 2004). Über Referenzprozesse soll der Aufwand der Prozessplanung verringert werden. Ein vergleichbarer Ansatz wird im Bereich der Datenverwaltung mit der Definition von Anwendungsszenarien verfolgt, die für ein konkretes Produktdatenmodell den Informationsbedarf über Teildatensätze (*views*) festlegen und dadurch den Umgang mit komplexen, fachübergreifenden Produktdatenmodellen vereinfachen (Steinmann 2005). Hierfür werden Notationen wie IDEF (Mayer et al. 1992) oder UML (Eriksson & Penker 2000) verwendet. Die damit beschriebenen Austauschszenarien sind meist auf sehr allgemeine Abläufe beschränkt, die auf der Ebene konkreter Aufgaben noch zu unspezifisch sind (Berg Von Linde 2001). Ein Beispiel für eine solche Definition ist in der Abbildung 2-7 dargestellt, die den prinzipiellen Ablauf der Tragwerksplanung und die hierfür benötigten Informationen beschreibt.

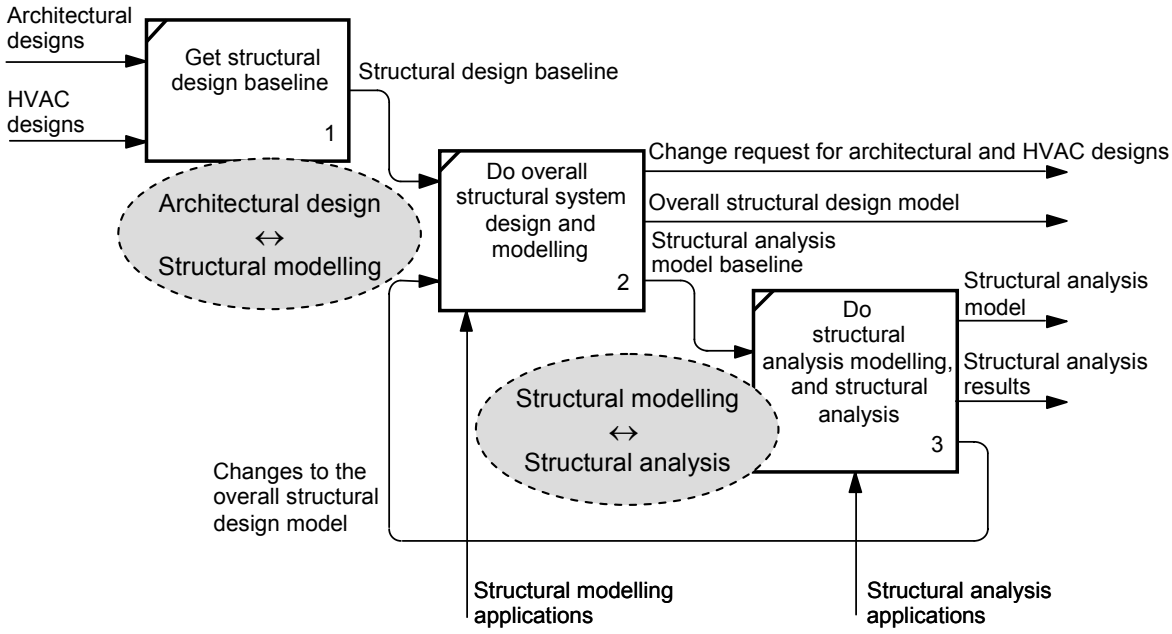


Abbildung 2-7 Auszug aus dem Austauschszenario für das IFC-Tragwerksmodell als IDEF-0 (Karstila 2004)

Die Notwendigkeit für die Verknüpfung der Produktdatenverwaltung mit der Prozessplanung wird in der Forschung allgemein akzeptiert und erfährt ein hohes Interesse[1] (Luiten et al. 1993, Froese et al. 1997, Haymaker 2004, König et al. 2004, Meißner 2005). Dies zeigen auch

[1] Die Verknüpfung von Produkt und Prozess wird auch in anderen Industriezweigen verfolgt, beispielsweise der Softwareentwicklung oder dem Maschinenbau. In der Informatik wird daher auch versucht, eine gemeinsame Grundlage für das Ingenieurwesen zu finden (vgl. Krapp et al. 1998).

Ansätze im Bereich wissensbasierter Systeme, die beispielsweise den Anwender bei Entwurfsaufgaben durch eine Konsistenzprüfung und den Vorschlag sinnvoller Entwurfsschritte unterstützen (Scherer & Eisfeld 2005, Kraft & Wilhelms 2005). Solche Systeme beruhen auf umfangreichen Wissensbasen und sind auf abgrenzbare Entwurfsaufgaben beschränkt. Eine umfassende Kopplung zwischen den Produktdaten und den darauf arbeitenden Prozessen ist derzeit noch nicht in Sicht. Dies liegt unter anderem daran, dass keine allgemein akzeptierten Modelle, Methoden und Arbeitsabläufe existieren (Wix & Liebich 2000). Auch wenn eine stärkere Integration in Teilbereichen bereits erfolgreich angewendet werden kann, so ist die bisher erreichte Kopplung insgesamt eher schwach.

2.3 Erfassen der Planungsschritte

Für das Erfassen der Planungsschritte gibt es mehrere Gründe. Neben der Dokumentation der erbrachten Planungsleistungen, die aus rechtlichen Gründen notwendig ist, können die Änderungen zwischen den verschiedenen Planungsständen ermittelt werden. Dadurch ist es möglich, zu alten Entwurfsständen oder Teillösungen zurückzukehren und getroffene Entwurfsentscheidungen nachzuvollziehen. Nicht zuletzt wird durch das Protokollieren der Planungsschritte die Grundlage geschaffen, das parallele Arbeiten mit einer gemeinsamen Datenbasis zu unterstützen.

In der Planungspraxis ist das Erfassen von Planungsständen meist auf die Sicherung von Dokumenten beschränkt. Dieser dokumentenbasierte Ansatz ist unabhängig von der zugrunde liegenden Datenstruktur und somit für alle Dokumenttypen gleichermaßen geeignet. Auf dieser Ebene ist eine effiziente Zugriffskontrolle jedoch nicht möglich und hat daher zu Ansätzen geführt, die statt dessen auf den Daten der Dokumente beruhen. Die Basis der entwickelten Versionsmodelle bildet zumeist die Datenstruktur, die von der Semantik der Daten abstrahiert und dadurch für verschiedene Datenmodelle anwendbar ist. Die höchste Akzeptanz hat hierbei das objekt-orientierte Modellierungsparadigma erlangt, das von verschiedenen Versionsmodellen genutzt wird.

Durch die derzeit geringe Verbreitung von Produktdatenmodellen haben solche Versionsansätze im Bauwesen noch keine Bedeutung erlangt. Die Entwicklung wird hauptsächlich durch die Softwareentwicklung und andere Ingenieurdisziplinen bestimmt. In diesen Anwendungsgebieten wird das Prinzip der Versionsverwaltung bereits erfolgreich eingesetzt und ist ein wesentliches Element der kooperativen Arbeit. Auch wenn die hierfür entwickelten Versionsmodelle viele Gemeinsamkeiten besitzen, so sind die Anforderungen, Voraussetzungen und verwendeten Werkzeuge dennoch sehr verschieden. Für eine Anwendung im Bauwesen interessieren daher zunächst die Terminologie und prinzipiellen Versionsansätze, die eine Bewertung vorhandener Versionsmodelle erlauben. Mit dieser Übersicht wird auf die Besonderheiten der Softwareentwicklung und anderer Ingenieurdisziplinen eingegangen.

Ein wesentlicher Aspekt beim Verwalten von Versionen ist der Umgang mit Änderungen. Dadurch ist eine platzsparende Speicherung von Planungsständen möglich. Die Änderungen sind aber nicht immer unmittelbar verfügbar und müssen daher oft nachträglich aus Dokumenten ermittelt werden. Hierfür werden Vergleichsverfahren eingesetzt, die die Änderungen bestimmen. Da im Bauwesen der Austausch von Dokumenten vorherrscht, sind die Auswertung und der Vergleich von Dokumenten ein wesentliches Element bei der Anwendung einer Versionsverwaltung.

2.3.1 Versionsansätze für objekt-orientierte Datenmodelle

Das Interesse an einer Versionsverwaltung hat verschiedene Gründe. Aus den damit verbundenen Zielen sind problemspezifische Versionsansätze entstanden, die auf ähnlichen Prinzipien beruhen. In der Forschung werden daher Bemühungen unternommen, die entstandenen Versionsansätze auf eine gemeinsame Grundlage und Terminologie zurückzuführen. Im Rahmen dieser Konsolidierung sind Vorschläge hervorgegangen, die verschiedenen Versionsansätze in einem universell einsetzbaren Versionsmodell zusammenzuführen (Katz 1990, Sciore 1994, Zeller & Snelting 1997). Solche Vorschläge sind nicht unumstritten, da die Anforderungen und verfügbaren Werkzeuge mitunter sehr verschieden sind (Westfechtel & Conradi 1998).

Ziele der Versionsansätze

Bei der Versionsverwaltung werden drei Ziele unterschieden (Sciore 1994):

1. das Erfassen verschiedener Zustände als eine Funktion über die Zeit (*historical database*),
2. die Konfiguration alternativ nutzbarer Komponenten zu einem konsistenten Produkt (*configuration management*) und schließlich
3. die Unterstützung des Entwurfsprozesses (*evolution of product design*, *concurrency control*).

Diese Ziele schließen sich gegenseitig nicht aus. Für die Unterstützung der kooperativen Arbeit werden die Konfiguration und die Unterstützung des Entwurfsprozesses häufig miteinander kombiniert. Zusätzlich werden *externe* und *interne* Versionen unterschieden (Schroeder 1995). Eine externe Version ist außerhalb der Datenverwaltung sichtbar und kann von einem Anwender modifiziert werden (*user level*). Eine interne Version ist demgegenüber ein Hilfsmittel der Datenverwaltung, die die Speicherung von Zwischenzuständen ermöglicht (*system level*). Der Schwerpunkt der betrachteten Versionsansätze liegt auf externen Versionen, die ein dauerhaftes Speichern der Versionen verfolgen.

Aufteilung eines Produktes in Objekte

In den betrachteten Versionsansätzen werden Informationen zu Objekten zusammengefasst. Ein Objekt repräsentiert einen identifizierbaren Teil des Produktes und kann beispielsweise eine bestimmte Baugruppe, Softwarekomponente oder Klassendefinition darstellen. Die Aufteilung in Objekte hängt von der gegebenen Problemstellung ab und kann z. B. mit Dateien oder Datenbankentitäten assoziiert werden. Mit dieser Aufteilung wird das Produkt in einzelne Teile zerlegt, die zueinander in Beziehung stehen. Aus diesen Beziehungen können horizontale und hierarchische Abhängigkeiten entstehen, die für die Bewertung der Konsistenz und die Aktualisierung der Daten (*change propagation*) wichtig werden.

Version, Revision und Varianten

Von einem Objekt können mehrere Zustände vorliegen. Diese Zustände werden als *Objektversionen* bezeichnet. Die Objektversionen eines Objektes stehen zueinander in Beziehung und können entweder eine Revision oder eine Variante beschreiben. Eine Objektversion wird als *Revision* bezeichnet, wenn sie eine Objektversion ersetzt. Demgegenüber liegt eine *Variante* (oder auch *Alternative*) vor, wenn die beiden betrachteten Objektversionen alternative Objektzustände beschreiben.

Erfassen von Zuständen und Änderungen

Objektversionen können in dem Versionsmodell als Zustand (*state-based*) oder als Änderung (*change-based*) beschrieben werden. Der Zustand eines Objektes kann vollständig oder als Differenz (*delta*) relativ zu einer anderen Objektversion gespeichert werden. Das Speichern von Differenzen ermöglicht eine platzsparende Verwaltung der Objektversionen und kann über verschiedene Verfahren realisiert werden. Gegenüber der zustandsbasierten Beschreibung kann eine neue Objektversion auch als Änderungsanweisung ausgedrückt werden. Hierfür werden Modifikationsoperationen verwendet, die auf eine Objektversion in einer gegebenen Reihenfolge anzuwenden sind. Bei diesem Ansatz wird beispielsweise die geometrische Lage eines Objektes nicht explizit über seine neuen Koordinaten beschrieben, sondern über eine Verschiebeoperation und einen Vektor ($\Delta x, \Delta y, \Delta z$).

Organisation zulässiger Konfigurationen

Mit einem Versionsmodell werden verschiedene Zustände oder Konfigurationen eines Produktes erfasst. Diese Konfigurationen müssen aus dem Versionsmodell generierbar sein. Hierbei werden zwei Ansätze unterschieden; 1) die implizite Beschreibung über den Zustand der Objektversionen (*intensional versioning*) und 2) die explizite Benennung der gültigen Objektversionen über ihre eindeutige ID (*extensional versioning*). Bei der impliziten Beschreibung ergibt sich das Problem, aus den möglichen Kombinationen der vorliegenden Objektversionen eine konsistente Konfiguration abzuleiten. Demgegenüber werden bei der expliziten Benennung nur solche Konfigurationen in einer Objektmenge erfasst, die zuvor in das Versionsmodell als gültige Konfiguration eingestellt bzw. benannt wurden.

Kombination zu einem Versionsmodell

Nach den vorgestellten Prinzipien kann ein Versionsmodell charakterisiert werden. Das Erfassen der Objektversionen und die Organisation der Konfigurationen sind hierbei orthogonale Kriterien, die sich gegenseitig nicht ausschließen. Ein Versionsmodell kann daher verschiedene Prinzipien miteinander kombinieren. Je nach Anwendungsgebiet besitzen bestimmte Aspekte ein höheres Gewicht und prägen dadurch die verwendeten Versionsmodelle.

2.3.2 Besonderheiten der Versionsansätze in der Softwareentwicklung

In der Softwareindustrie ist die Versionsverwaltung inzwischen eine unverzichtbare Grundlage für das Entwickeln und Pflegen von Softwareprodukten. Entsprechend sind ausgereifte Werkzeuge verfügbar, die das Verwalten von Versionen ermöglichen und deren Einsatz im Bauwesen erstrebenswert machen (Firmenich et al. 2005). Aus organisatorischen und technischen Gründen können die Lösungen der Softwareindustrie nicht ohne weiteres übernommen werden. Daher werden technische Besonderheiten der Softwareentwicklung betrachtet, die bei der Bewertung der hierfür geschaffenen Werkzeuge berücksichtigt werden müssen.

Software Configuration Management (SCM)

Das Verwalten von Versionen wird in der Softwareindustrie mit dem Begriff *Software Configuration Management* (SCM) verbunden. Aus diesem Begriff wird deutlich, dass die Konfigurationsunterstützung ein wesentlicher Aspekt bei der Softwareentwicklung ist. Den Ursprung dieser Entwicklung bildet die bedingte Übersetzung (*conditional compilation*), die das Umwandeln der Quelltexte in ausführbare Programmteile und das Zusammenführen zu einer lauffähigen Software steuert. In Abhängigkeit vom vorgesehenen Betriebssystem, dem benötigten Funktionsumfang und der betrachteten Programmversion werden unterschiedliche Pro-

grammteile ausgewählt und zu einem konsistenten Produkt zusammengeführt. Dieser Prozess wird traditionell über das Setzen bestimmter Attribute gesteuert, das von hieraus hervorgegangenen Versionsansätzen übernommen wurde (Zeller & Snelting 1997). Voraussetzung für diese Art der Produktselektion ist die Möglichkeit der Konsistenzkontrolle, die eine gültige von einer ungültigen Konfiguration unterscheiden kann.

Neben der Konfigurationsunterstützung ist der Aspekt der Kooperation wichtig. Im Gegensatz zur bedingten Übersetzung steht hierbei das Organisieren von Revisionen im Vordergrund, die traditionell in gerichteten azyklischen Graphen verwaltet werden. Die Ursprünge liegen in den Arbeiten von Rochkind (1975, *Source Code Control System - SCCS*) und Tichy (1985, *Revision Control System - RCS*), deren graphbasierter Ansatz in den heute weit verbreiteten Versionswerkzeugen wie CVS (Cederqvist 2005) oder Subversion (Collins-Sussman et al. 2004) übernommen wurde. Dieser Ansatz wird häufig mit einer platzsparende Speicherung der Objektversionen über Deltas kombiniert.

Diese Anforderungen haben in der Softwareindustrie zu Ansätzen geführt, die sowohl das explizite als auch das implizite Beschreiben von Konfigurationen unterstützen. Eine Übersicht über verschiedene Ansätze und hieraus hervorgegangene Versionsmodelle des SCM ist in der Veröffentlichung von Conradi & Westfechtel (1998) zu finden. Aktuelle Informationen und Links zu verfügbaren Werkzeugen des SCM sind beispielsweise auf der Web-Seite von Appleton (2005) zu finden.

Software als Produkt

Die Softwareindustrie erzeugt im Gegensatz zu anderen Ingenieurdisziplinen kein materielles Produkt. Sie produziert vielmehr ein Programm, das im Computer verbleibt und keinen Übergang in die „Realität“ kennt. Das Problem der Modellbildung, also die Idealisierungen real existierender Objekte, ist daher nicht bzw. nicht in dieser Form vorhanden. Redundante, in unterschiedlichen Sichten mehrfach vorkommende Informationen stellen somit ein untergeordnetes Problem dar und führen zu weniger Querverbindungen in der verwalteten Datenbasis. Die Quelltextdateien und Programmressourcen werden mit Entwicklerwerkzeugen in ausführbare Module übersetzt und zu einem lauffähigen Programm zusammengefügt. In diesem automatisierten Prozess sind die gegenseitigen Abhängigkeiten der Objekte (Bindungen) bekannt und können für die Bewertung der Konsistenz genutzt werden. Hierfür werden integrierte Entwicklungsumgebungen verwendet, die den Entwicklungsprozess vollständig unterstützen.

Textdateien als Grundlage der Versionsverwaltung

Ein wesentliches Merkmal der Softwareentwicklung ist die Verwendung von Textdateien. Diese (Quell-)Textdateien können mit beliebigen Texteditoren gelesen und modifiziert werden. Hierbei findet keine Umwandlung der Daten in andere Darstellungsformen statt. Die Veränderung einer Quelltextdatei entspricht somit 1 : 1 der vom Programmierer vorgenommenen Modifikation (Westfechtel & Conradi 1998). Um verschiedene Programmiersprachen und andere Textinhalte zu unterstützen, werden diese Textdateien häufig als eine beliebige Anordnung von Zeichen interpretiert. Mit dieser Verallgemeinerung gehen für die Softwareentwicklung wichtige Informationen wie z. B. semantische Abhängigkeiten der Dateien verloren. Andererseits wird mit diesem Ansatz eine gewisse Allgemeingültigkeit erreicht, die die hierfür entwickelten Methoden auch für das Bauwesen interessant werden lassen.

Nutzung verfügbarer Werkzeuge im Ingenieurwesen

Die Nutzung der in der Softwareentwicklung eingesetzten Werkzeuge ist mit den Voraussetzungen des Ingenieurwesens nicht unmittelbar möglich. Das Problem liegt in der Granularität und Strukturierung der im Bauwesen verwendeten Dokumente, die nur bedingt für die Verwaltung mit textbasierten Versionsansätzen geeignet sind. Firmenich et al. (2005) haben die Möglichkeit untersucht, CVS-Werkzeuge auch für das Ingenieurwesen anzuwenden. Ihre Untersuchung beruht auf dem Ansatz, Dokumente in ihre Struktur aufzubrechen. Mit dem Wissen über die Struktur der Daten werden aus einem Planungsstand mehrere XML-Dateien erzeugt, die jeweils einem Datenobjekt entsprechen. Bei dieser Serialisierung wird ein Übergang in Textdateien vollzogen, bei dem ein Datenobjekt in eine Reihenfolge von Zeichen transformiert wird. Unter der Voraussetzung, dass jedes Objekt eindeutig identifizierbar und die Abbildung in eine Textdatei eindeutig durchführbar ist, kommen Firmenich et al. (2005) zu dem Schluss, dass diese Werkzeuge auch für die kooperative Arbeit im Bauwesen einsetzbar sind. Aufbauend auf diesen robusten und erprobten Versionswerkzeugen sind lediglich die unvermeidbaren, problemspezifischen Anpassungen notwendig, die beispielsweise das Anzeigen von Änderungen, den Prozess der Konfliktbeseitigung oder den Umgang mit Teildatensätzen unterstützen.

2.3.3 Versionsansätze des Ingenieurwesens

Das Ingenieurwesen ist durch die Verwendung komplexer Datenmodelle geprägt, die unterschiedliche Sichten auf ein Produkt beschreiben. Katz (1990) charakterisiert allgemein: *"The most important aspects of CAD data are their evolutionary, multirepresentational, and complex structure."* Diese Einschätzung teilen auch Westfechtel und Conradi (1998), die in der Komplexität und den verschiedenen Sichten einen wesentlichen Unterschied zur (textbasierten) Softwareentwicklung sehen. Als Begründung werden Standardisierungsbemühungen wie STEP genannt, die zu umfangreichen Modelldefinitionen geführt haben und die Auswertung der Datenstrukturen ermöglichen. Die Datenstrukturen beruhen überwiegend auf dem objektorientierten Konzept, das die Planungsdaten explizit in Objekte bzw. Entitäten und die gegenseitigen Beziehungen unterteilt. Das Produkt besteht auf dieser Ebene aus einer Menge von Objekten, die für eine zweckmäßige Verwaltung häufig weiter strukturiert wird. Hierfür werden zwei Beziehungstypen verwendet, die einerseits das Beschreiben von Objekthierarchien (*Aggregation*) und andererseits das Erfassen von Abhängigkeiten und Äquivalenzrelationen erlauben. Auf diese Weise können Objekte zu Komponenten zusammengefasst und der Bezug zwischen „gleichbedeutenden“ bzw. voneinander abhängigen Objekten erfasst werden.

Organisation der Objektversionen

Die Strukturierung der Produktdaten ist die Grundlage für die Organisation und die Verwaltung der Versionen. Ein Objekt wird hierbei als ein Element der Produktbeschreibung betrachtet, das in verschiedenen Versionen vorliegen kann. Das Zusammenfügen der Versionen zu einer Produktbeschreibung geschieht durch Auswahl der erforderlichen Objekte und der gewünschten Objektversion. Dieses Prinzip wird in verschiedenen Versionsansätzen auch auf Objekthierarchien übertragen, indem eine Komponente, die sich aus mehreren Objekten zusammensetzt, und die gewünschte Version der Komponente gewählt werden können. In der Abbildung 2-8 ist die Unterscheidung in Objekte und Objektversionen dargestellt, die in dieser oder ähnlicher Form in den meisten Ansätzen zu finden ist. Durch diese Unterscheidung werden eine weitestgehend statische, versionsunabhängige Ebene und eine dynamische versionsabhängige Ebene geschaffen. Auf dieser Basis werden statische und dynamische Referen-

zen eingeführt, die entweder an eine Objektversion gebunden oder über ein Objekt und zusätzliche Auswahlkriterien zur Laufzeit aufzulösen sind.

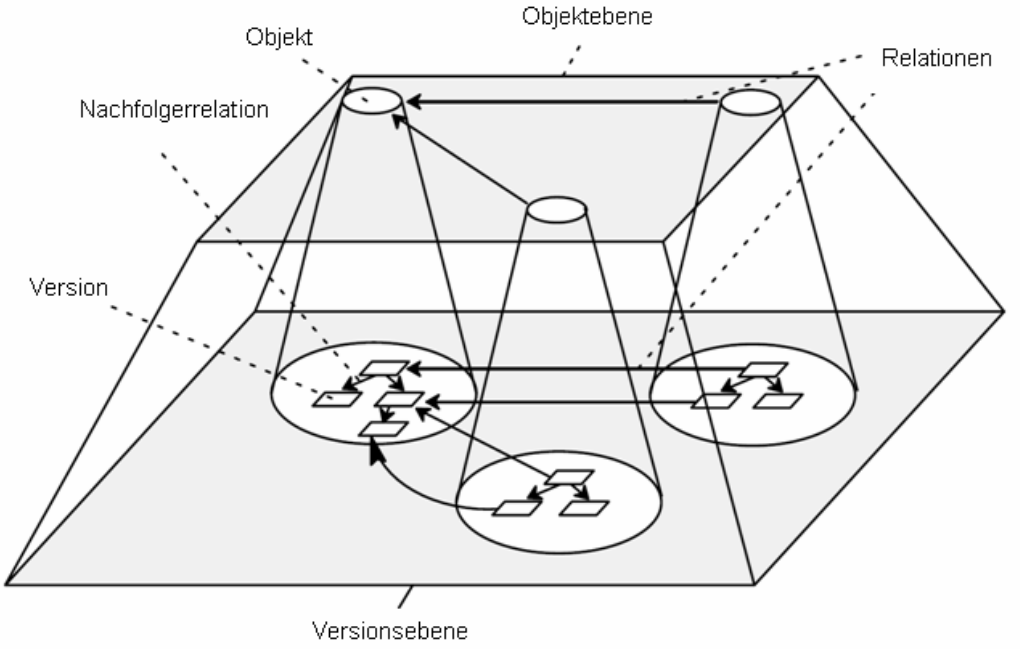


Abbildung 2-8 Grundverständnis über eine objekt-orientierte Versionsverwaltung (Westfechtel 1999)

Für die Organisation der Objektversionen wird eine sogenannte Versionshistorie angelegt, die über Vorgänger-Nachfolger-Beziehung die Entstehung der Objektversionen beschreibt. Hieraus entsteht ein gerichteter, azyklischer Versionsgraph, der zwischen Varianten (Verzweigung des Graphen) und Revisionen (Objektsequenz) unterscheidet. Dieses Verständnis ist weitestgehend identisch mit Versionsansätzen der Softwareentwicklung. Im Vergleich zur Softwareindustrie sind die Daten des Ingenieurwesens aber durch deutlich mehr und vor allem komplexere Abhängigkeiten geprägt. Die Ursache liegt insbesondere in den verschiedenen Sichten auf das Produkt, die schließlich zu Redundanzen führen. Hierdurch entsteht das Problem, Widersprüche in der Datenbasis zu erkennen (*change notification*) und im Idealfall zu beseitigen (*change propagation*). Diese Problemstellung ist eng verwandt mit der Aktualisierung abhängiger Daten (*derived objects*), die im Vergleich zur Softwareindustrie jedoch nicht vollständig formalisiert ist und dadurch nicht automatisch durchgeführt werden kann.

Abhängigkeiten zwischen den Daten

Die Voraussetzung für das Propagieren von Änderungen ist das Wissen über die Abhängigkeiten der Daten. Wird eine Änderung vorgenommen, so ergibt sich unter Umständen die Notwendigkeit, andere Daten zu aktualisieren oder zu ändern. Im Idealfall sind diese Abhängigkeiten funktional beschrieben und können automatisch durchgeführt werden. Hieraus entstehen eigenständige Probleme[1], die beispielsweise in der Arbeit von Hanff (2003) diskutiert werden. Häufig lassen sich die Abhängigkeiten aber nicht funktional beschreiben und werden statt dessen über Bindungsrelationen nur formal erfasst. Dadurch sind bei einer Änderung die zu aktualisierenden Daten bekannt und können in der Datenverwaltung entsprechend markiert werden. Das Erfassen der hierfür erforderlichen Bindungsrelationen kann unterschiedlich detail-

[1] Wesentliche Problemstellungen sind das Propagieren der Änderungen, der Umgang mit Zyklen oder das gezielte Aktualisieren einer bestimmten Teilmenge.

liert erfolgen. Werden die Abhängigkeiten auf der Ebene der Attribute erfasst, so lassen sich die abhängigen Daten relativ genau bestimmen. Werden die Abhängigkeiten dagegen auf der Ebene der Objekte oder Komponenten erfasst, so entsteht häufig das Problem, dass Daten als inkonsistent erkannt werden, die eigentlich nicht zu aktualisieren sind. Dieser Fehler kann sich über die indirekt abhängigen Daten weiter fortpflanzen und schließlich zu einem unbrauchbaren Ergebnis führen (siehe *proliferation problem*, Katz 1990, Conradi & Westfechtel 1998).

Das Beschreiben der Bindungen ist eng verwandt mit dem Problem der Datentransformation. Das Thema der Datentransformation verdeutlicht die Komplexität, die beim Erfassen und Beschreiben modellübergreifender Abhängigkeiten auftritt (siehe Abschnitt 2.1.4). Abhängigkeiten existieren aber auch innerhalb der Modelle. Diese modellinternen Abhängigkeiten sind in den Anwenderprogrammen beschrieben und für die Datenverwaltung in der Regel nicht einsehbar. Ein wesentliches Problem für das Aktualisieren der Datenbasis ist somit die Verfügbarkeit der Bindungen.

Durch die unvollständig erfassten Abhängigkeiten ist das Erkennen einer konsistenten Konfiguration nicht ohne weiteres möglich. Dies ist ein Grund, warum im Ingenieurwesen die explizite Auswahl der Objektversionen favorisiert wird. Ein weiterer Grund liegt in der kombinatorischen Vielfalt, die durch die Vielzahl der Objekte die implizite Auswahl über Eigenschaften nicht sinnvoll erscheinen lässt.

Anwendung im Bauwesen

Die Hauptanwendungsgebiete der Versionsverwaltung bilden neben der Softwareentwicklung vor allem die Chemieindustrie, die Elektrotechnik und der Maschinenbau. Diese Industriezweige sind mit dem Bauwesen in Bezug auf die Arbeitsorganisation und den Grad der Datenintegration nicht unmittelbar vergleichbar. Der Unterschied drückt sich sehr deutlich durch das Verhältnis zwischen Produkt- und Entwicklungskosten aus, das im Bauwesen völlig konträr zu den erwähnten Industriezweigen ist. Dass die Versionsverwaltung auch im Bauwesen sinnvoll eingesetzt werden kann, zeigt die Arbeit von Firmenich (2002). Er stellt einen Ansatz vor, der auf der Grundlage der *Feature Logic* (Smolka 1992) und der Arbeit von Zeller (1997) die Organisation und das Arbeiten mit einer strukturierten Objektversionsmenge ermöglicht. Das entwickelte Versionsmodell beruht konzeptionell auf einem gerichteten, azyklischen Graphen, der zwischen Objekten, Objektversionen (Revisionen und Varianten), Versionsbeziehungen und Bindungen unterscheidet. Für dieses Versionsmodell werden Operationen definiert, die das Bearbeiten einer beliebigen[1] Objektmenge in einem privaten Arbeitsbereich (*workspace*) erlauben. Hierbei wird unter anderem das Nachladen und selektive Austragen der zu bearbeitenden Objektmenge im Arbeitsbereich unterstützt. Ein wichtiger Aspekt der Arbeit stellt die Konsistenz der Planungsdaten dar. Hierfür werden Bindungen genutzt, die für Objektversionen die gegenseitigen Abhängigkeiten definieren und dadurch die zu aktualisierenden Objekte aus dem entstehenden Bindungsgraphen erkennen lassen. Die Strukturierung der Objektversionen erfolgt über Attribute der Objekte (*features*), die den Versionsgraphen und die Bindungen der Objekte abbilden. Der Zugriff auf die Daten und Konfigurationen erfolgt über *Feature Terme*. Mit diesem Ansatz wird die kooperative Bearbeitung gemeinsam verwendeter Planungsdaten unterstützt. Hierbei wird vorausgesetzt, dass die verwendeten An-

[1] Für die Bearbeitung in einem privaten Arbeitsbereich wird die mögliche Auswahl auf logisch zusammengehörige Objekte beschränkt. Dadurch wird verhindert, dass ein Objekt in unterschiedlichen Versionen oder mit unverträglichen Versionen anderer Objekte geladen wird. Der Arbeitsbereich enthält somit eine mögliche Konfiguration der Objektversionen und verhält sich daher wie ein unversioniertes System.

wenderprogramme mit der zugrunde liegenden Datenstruktur ohne Strukturkonflikte arbeiten können. Technische Unzulänglichkeiten der Datenintegration werden hierbei nicht thematisiert.

2.3.4 Erkennen und Verwalten von Änderungen

Mit der Verwaltung unterschiedlicher Entwurfsversionen ist ein Anwachsen der Datenmenge verbunden. In Abhängigkeit von der Datenstruktur und der Anzahl der Versionsstände kann ein Zuwachs an Daten entstehen, der nicht mehr handhabbar ist. Ein wesentliches Ziel ist daher das effiziente Speichern der Versionsstände. Hierfür werden änderungsbasierte Ansätze verfolgt, die anstatt kompletter Planungsstände die wesentlich kleineren Änderungsmengen speichern. Firmenich (2004) schlägt einen Ansatz vor, eine Änderungsmenge durch Operationen auszudrücken und sieht darin unter anderem eine Alternative zur verlustbehafteten Datentransformation. Nach dieser Vorstellung werden keine Planungsstände sondern Modifikationsoperationen ausgetauscht, die auf die anderen Modelle anzuwenden sind. Die Eignung wurde am Beispiel der 3D-Modellierung mit dem ACIS-Modellierkern[1] und den zur Verfügung stehenden Operationen gezeigt. Weitergehende Erfahrungen mit Produktdatenmodellen sind bisher nicht bekannt. Im Gegensatz zur Softwareentwicklung haben solche Ansätze im Ingenieurwesen nur wenig Interesse erfahren (Westfechtel & Conradi 1998). Dies liegt unter anderem daran, das der Zugriff auf die Modifikationsoperationen nicht immer gewährleistet ist. Dieser Ansatz wird daher primär mit integrierten Programmsystemen assoziiert.

Das Speichern von Differenzen ist demgegenüber nicht von der Verfügbarkeit der Modifikationsoperationen abhängig. Statt dessen werden die vorgenommenen Änderungen nachträglich berechnet. Hierfür werden Vergleichsverfahren eingesetzt, die auf der Grundlage der im Versionsmodell verfügbaren Differenzoperationen die Änderungen zwischen den beiden Planungsständen beschreiben. Hierfür existieren im allgemeinen beliebig viele Differenzmengen. Durch das Vergleichsverfahren wird von den theoretisch möglichen Differenzmengen eine Lösung ausgewählt. Unter dem Aspekt der Speicherplatzoptimierung wird hierbei das Minimum an Differenzoperationen angestrebt. Eine solche Differenzmenge entspricht nicht zwingend den tatsächlich durchgeführten Modifikationen und besitzt somit eine eingeschränkte Aussagekraft über den Planungsschritt.

Differenzoperationen und Deltaansätze

Die Vergleichsverfahren werden überwiegend auf drei Editieroperationen zurückgeführt. Hierzu gehören das Erzeugen (*create/insert*), das Löschen (*delete/remove*) und das Ändern (*change/update*). Diese Basisoperationen werden in einigen Ansätzen um das Verschieben (*move*) erweitert (Apostolico & Galil 1997). Durch diese allgemeingültigen Editieroperationen lassen sich unabhängig von der zugrunde liegenden Datenstruktur die Änderungen beschreiben. Als Operanden können dabei sowohl einzelne Zeichen und Zeichenketten als auch Attribute, Objekte und Bäume dienen. Die Unterscheidung der Operanden wird relevant, wenn die Struktur der zu vergleichenden Daten betrachtet wird.

Für das Beschreiben der Deltas wird bezüglich der Versionshistorie zwischen Vorwärts- und Rückwärtsdeltas unterschieden. Beim Einsatz von Vorwärtsdeltas wird eine Entwurfsversion über eine in der Versionshistorie zurückliegende Entwurfsversion beschrieben. Werden Rückwärtsdeltas verwendet, so wird eine nachfolgende Entwurfsversion als Ausgangsbasis

[1] http://www.spatial.com

zur Berechnung der gesuchten Entwurfsversion verwendet (siehe Abbildung 2-9). Das Vorwärtsdeltaverfahren hat mit dem Anwachsen der Versionshistorie das Problem, dass aktuelle Entwurfsversionen über zunehmend mehr Differenzoperationen zu regenerieren sind. Aus diesem Grund wird meist kein reines Vorwärtsdeltaverfahren verfolgt, sondern mit der Speicherung kompletter Planungsstände kombiniert. Demgegenüber werden beim Rückwärtsdeltaverfahren die aktuellen Entwurfsversionen immer vollständig gespeichert und müssen nicht über Deltas berechnet werden. Liegen alternative Entwurfsversionen vor, so sind beim Rückwärtsdeltaverfahren zwangsläufig mehrere Entwurfsversionen vollständig zu verwalten. Zusätzlich besteht das Problem, dass beim Einstellen einer neuen Entwurfsversion der vorhandene Versionsgraph verändert werden muss. Eine zu aktualisierende Version wird hierbei nämlich in Deltas umgewandelt und stört für diesen Zeitraum den Zugriff auf die Datenbasis. Um die Nachteile beider Ansätze zu verringern, werden sogenannte Mischverfahren eingesetzt, die Vorwärts- und Rückwärtsdeltaverfahren miteinander kombinieren.

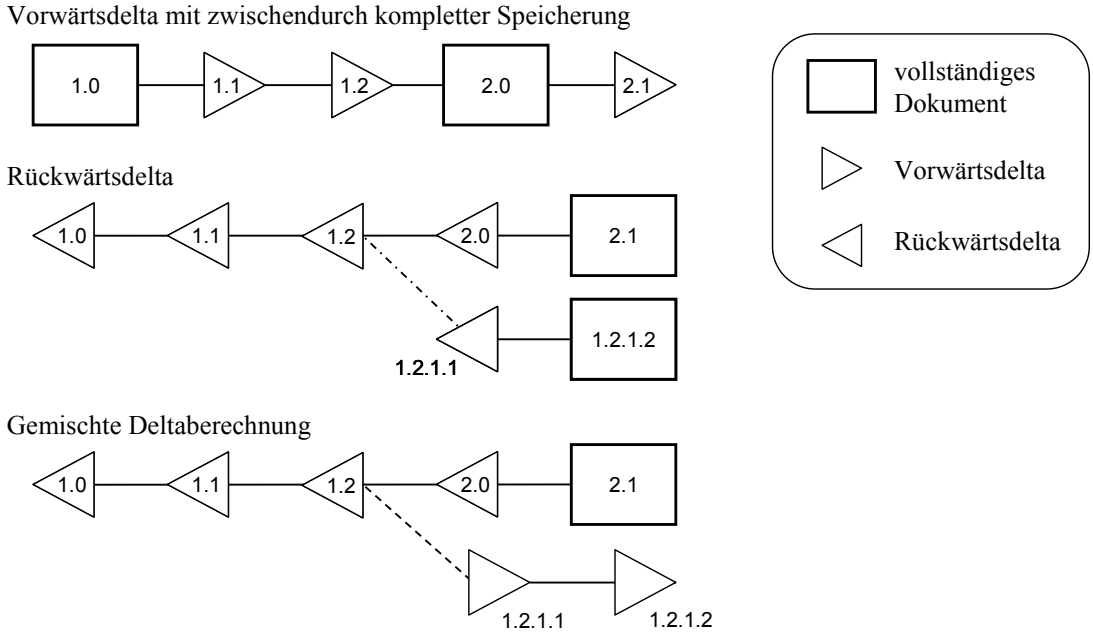


Abbildung 2-9 Deltaverfahren nach Schroeder (1995)

Vergleichsverfahren

In der Softwareentwicklung werden häufig textbasierte Vergleichsverfahren eingesetzt, die für Quelltextdateien meist ausreichend gute Ergebnisse liefern. Hierfür stehen leistungsfähige Vergleichsalgorithmen bereit, die auf Problemstellungen wie das Editieren von Zeichenketten (*string to string correction problem*, Wagner & Fischer 1974) und die Suche nach der längsten gemeinsamen Zeichenfolge (*longest common subsequence*, Apostolico & Galil 1997) zurückgeführt werden. Diese Ansätze beruhen auf der Auswertung von Text-Nachbarschaften und liefern für strukturierte Daten oft keine sinnvollen Ergebnisse. Ist die Datenstruktur bekannt und können die darin enthaltenen Elemente eindeutig identifiziert werden, so werden andere Ansätze bevorzugt. Durch die Identifizierbarkeit der Elemente werden die Planungsdaten in zuordenbare Einheiten aufgeteilt, die gezielt miteinander verglichen werden. Für eine objekt-orientierte Datenstruktur ist im Idealfall jedes Objekt und somit auch jedes Attribut[1]

[1] In einem objekt-orientierten System ist diese Forderung per Definition erfüllt, da jedes Attribut über seinen Namen und das zugeordnete Objekt eindeutig identifizierbar ist.

eindeutig identifizierbar, wodurch ein sehr effizienter Vergleich ermöglicht wird. Für die Datenmodelle der Praxis ist eine eindeutige Zuordnung der Objekte häufig aber nicht gegeben. Dadurch entstehen Attribut- bzw. Objektmengen, die nur über ihre Struktur miteinander vergleichbar sind.

In der Informatik werden strukturbasierte Vergleichsverfahren beispielsweise für den Vergleich von Syntaxbäumen benötigt (*tree editing problem*). Bei dieser Problemstellung wird zwischen geordneten und ungeordneten Bäumen unterschieden, die sich in ihrer Komplexität und den möglichen Lösungsansätzen unterscheiden. Für geordnete Bäume (*labeled ordered trees*) wird unter anderem versucht, die Baumstruktur durch das Umwandeln in eine Zeichenkette über textbasierte Vergleichsverfahren zu behandeln. Dieser Ansatz ist aber nicht immer erfolgreich und für ungeordnete Bäume (*labeled unordered trees*) nicht geeignet. Die für ungeordnete Bäume vorliegende Problemstellung der minimalen Editieroperationen ist np-vollständig[1] (Spinner 1987, Apostolico & Galil 1997) und lässt sich daher nur über Vereinfachungen algorithmisch behandeln. Für die Verringerung der Komplexität werden die möglichen Änderungen, also die Zuordnung der Teilbäume, eingeschränkt. Auf diese Weise lassen sich Lösungen für das Editierproblem finden, die meist sehr nah an dem gesuchten Minimum liegen.

Durch den Erfolg von XML hat der Vergleich von Baumstrukturen in den letzten Jahren ein gestiegenes Interesse erfahren. Hieraus sind neue Verfahren und Werkzeuge entstanden, die sich vor allem mit dem Problem der geordneten Bäume beschäftigen (Cobéna et al. 2002a). Ein wichtiges Thema ist die Effizienz bezüglich Rechenzeit und Speicherverbrauch. Cobéna et al. (2002b) stellen beispielsweise ein Verfahren vor, das vorhandene Identifikatoren berücksichtigt und als Ausgangspunkt für die Zuordnung verbleibender Knoten nutzt. Für objekt-orientierte Datenstrukturen sind solche Verfahren anwendbar, wenn eine Transformation in die Problemklasse der geordneten Bäume durchführbar ist. Die Anwendbarkeit lässt sich auf ungeordnete Baumstrukturen erweitern, wenn der verwendete Vergleichsalgorithmus als Editieroperation das Verschieben (*move*) bzw. ungeordnete Bäume unterstützt.

Erwartete Differenzen

Für den Vergleich zweier Datensätze sind die zu erwartenden Differenzen ein wesentliches Kriterium, das über die Anwendbarkeit eines Vergleichsalgorithmus und die Effizienz der Deltabeschreibung entscheidet. Ein Planungsschritt ist im Allgemeinfall wie folgt charakterisiert:

- ein über mehrere Stunden oder Tage dauernder Vorgang (*lange Transaktion*),
- die Anpassung und Verfeinerung von Entwurfslösungen (*top-down Entwurf*) sowie
- die Transformation von Planungsdaten zwischen unterschiedlichen Darstellungsformen.

Diese Eigenschaften erzeugen häufig Änderungen im Objektgefüge, die mit den klassischen Editieroperationen durch das Erzeugen und Löschen von Objekten beschrieben werden müssen. Die Entstehungsgeschichte der geplanten (Realwelt-)Objekte geht dadurch verloren und führt darüber hinaus zu umfangreicheren Differenzmengen. In diesen Fällen ist es nur eingeschränkt möglich, einen Entwurfsschritt nachvollziehbar zu beschreiben.

[1] Algorithmen für diese Problemklasse lassen sich in ihrer Komplexität nicht polynomial beschränken und sind daher ineffizient.

2.4 Umsetzung der Datenverwaltung in ein Softwaresystem

Technisch sind die Voraussetzungen vorhanden, um räumlich verteilt arbeitende Planer in einem Netzwerk zusammenzuschließen und den Austausch von Informationen zu unterstützen. Auf dieser Grundlage ist ein Abgleich der Planungsinformationen ohne physischen Kontakt und ohne zeitliche Verzögerungen möglich, so dass widersprüchliche Entwurfsentscheidungen unmittelbar erkannt und behoben werden können.

Mit dem Themenkomplex der netzwerkgerechten Zusammenarbeit beschäftigt sich das Forschungsgebiet *Computer Supported Cooperative Work*[1] (CSCW) (Borghoff & Schlichter 1998, Hauschild 2003). In den Lösungsansätzen des CSCW werden unterschiedliche Anwendungsszenarien und Techniken integriert, die sowohl die *synchrone* als auch die *asynchrone* Kooperation[2] unterstützen. Im Rahmen der vorliegenden Arbeit wird ein Teil dieses Themenkomplexes betrachtet. Der Schwerpunkt liegt auf den Problemen der Datenverwaltung, die bei der asynchronen Kooperation über lange Transaktionen entstehen. Die Änderung der Daten erfolgt bei diesem Szenario in einem eigenen Arbeitsbereich und ist für die anderen Planer erst nach Abschluss der Transaktion zugänglich. Zusätzlich wird davon ausgegangen, dass für die Bearbeitung fachspezifische Datenmodelle verwendet werden. Die Transformation der Daten ist demzufolge ein wesentlicher Aspekt.

Für das beschriebene Szenario interessiert, wie die verteilte Zusammenarbeit über ein Computernetzwerk realisiert werden kann. Hierzu gehören Fragen über den Ort, mögliche Kopien sowie den Austausch bzw. das Synchronisieren der Planungsdaten.

2.4.1 Verteilte Architekturen

Über eine verteilte Architektur wird der Zugriff auf gemeinsame Planungsinformationen erreicht. In der Literatur wird hierfür der Begriff *Information sharing* geprägt, der nach klassischer Vorstellung mit dem *Client-Server-Modell* assoziiert wird. Bei diesem Ansatz wird zwischen den drei Komponenten Präsentation, Ausführung (Programmlogik) und Datenhaltung unterschieden. Je nach Zuordnung der Komponenten zu den Clients bzw. dem Server wird dieser Ansatz weiter unterteilt (Borghoff & Schlichter 1998). In der Tabelle 1 sind vier mögliche Fälle gezeigt, wobei die Datenhaltung stets zentral beim Server liegt. Demgegenüber kann die Datenhaltung auch dezentral bei den „Clients“ liegen, wodurch Planungsdaten repliziert werden und zwangsläufig das Propagieren von Änderungen erforderlich ist. Die Softwarearchitektur wird nicht zuletzt durch den gewählten Ansatz der Datenintegration beeinflusst, der entweder eine zentrale oder dezentrale Lösung begünstigt (siehe Abschnitt 2.1.3). Schließlich sind Mischformen möglich, die beispielsweise den Verbund oder die hierarchische Organisation von Servern vorsehen.

Zentrale Datenverwaltung - Produktmodellserver

Von den in Tabelle 1 gezeigten Fällen sind für die Zusammenarbeit über lange Transaktionen nicht alle Möglichkeiten relevant. Für dieses Szenario liegen die Präsentation und Ausführung in der Regel beim Client (*Fat-Client*). Die benötigten Planungsdaten werden zu Beginn der

[1] In der deutschen Übersetzung wird dieses Forschungsgebiet als *Rechnergestützte Gruppenarbeit* bezeichnet.

[2] Im Planungsprozess wechseln sich die Phasen der synchronen und asynchronen Kooperation im Normalfall ab. Die synchrone Kooperation beschreibt dabei die zeitgleich stattfindende, aufeinander abgestimmte Zusammenarbeit. Demgegenüber beschreibt die asynchrone Kooperation eine zeitlich voneinander unabhängige Phase der Zusammenarbeit (Hauschild 2003).

Transaktion in den Arbeitsbereich des Clients geladen (*check-out*), dort unabhängig bearbeitet und am Ende der Transaktion auf den zentralen Server zurückgespielt (*check-in*). Die Vorteile liegen vor allem in der unabhängigen Bearbeitung, die unempfindlich gegenüber zeitweiligen Netzwerkstörungen und der Verfügbarkeit des Servers ist. Der Nachteil liegt in der Replikation der Planungsdaten, die zu einem höheren Aufwand für das Abgleichen parallel ablaufender Planungsschritte führt.

Tabelle 1 Client-Server-Konzepte nach Borghoff & Schlichter (1998)

Client(s)					
Präsentation Ausführung	Präsentation	Präsentation	Präsentation Ausführung	Präsentation Ausführung (mit lokaler Datenbank)	Präsentation Ausführung Datenbank
Server					
  Datenbank	Präsentation Ausführung Datenbank	Ausführung Datenbank	Ausführung Datenbank	 Ausführung Datenbank	 Datenbank
Fall 1	Fall 2		Fall 3		Fall 4

Die Entwicklung zentraler Datenserver geht einher mit der Definition von Produktdatenmodellen. Erste Ansätze sind in Forschungsarbeiten wie COMBINE (Lockley et al. 1995), COMBI (Scherer 1995) und ToCEE (Katranuschkov & Hyvärinen 1998) zu finden, die sich sowohl mit der Datenmodellierung als auch der netzwerkgerechten Datenverwaltung beschäftigt haben. In dieser Zeit wurde der Begriff des *Produktmodellservers* geprägt, der heute als Synonym für die zentrale Verwaltung von Produktdaten verwendet wird. Mit den hierbei gewonnenen Erfahrungen hat sich die Erkenntnis durchgesetzt, dass lange Transaktionen eine wichtige Rolle bei der kooperativen Bearbeitung einnehmen. Darüber hinaus wurde deutlich, dass ein Produktmodellserver nicht auf die Funktionalität verfügbarer Datenbanken beschränkt werden kann.

Der Forschungsschwerpunkt im Bereich der zentralen Produktdatenverwaltung liegt in der Erleichterung der dezentrale Bearbeitung mit eigenständigen Anwenderprogrammen. Wesentliche Aspekte sind der Zugriff auf die benötigten Planungsdaten, das Unterstützen langer Transaktionen und schließlich die Konsistenz der Planungsdaten. Aus diesen Anforderungen sind Ansätze hervorgegangen, die die Programmlogik der Datenintegration im Produktmodellserver zusammenführen. Hierzu gehören unter anderem das Unterstützen von Datentransformationen, das Bilden von Modellsichten (*views*), die Nebenläufigkeitskontrolle, das Propagieren von Änderungen sowie das Überwachen von Konsistenzbedingungen. In den vorangegangenen Abschnitten sind die damit verbundenen Probleme geschildert, die eine Umsetzung des hierfür erforderlichen Funktionsumfangs erschweren (Amor & Faraj 2001). In der Praxis werden statt dessen Ansätze verfolgt, die die benötigte Programmlogik zu einem unterschiedlichen Grad an die Clients delegieren. Die Umsetzung des Produktmodellservers wird somit durch höhere Anforderungen an die Clients und durch Einschränkungen der Funktionalität vereinfacht.

Unabhängig von den Problemen der Datenintegration bildet das Client-Server-Modell die Grundlage für zahlreiche praxisrelevante Anwendungen. Hierzu gehören Dokumentmanagementsysteme, internetbasierte Bauportale und kommerziell verfügbare Produktmodellserver (EDMserver™, Eurostep ModelServer, BSPro COM-Server)[1]. Die Entwicklung und Anwendung dieser Produktmodellserver wird vor allem durch standardisierte Datenmodelle wie IFC (Wix & Liebich 2003), CIS/2 (http://www.cis2.org/) oder STEP-CDS (http://www.step-cds.de/) und die hierfür verfügbaren Anwenderprogramme begünstigt. Sie beruhen auf Lösungen, die in ähnlicher Form auch in anderen Industriezweigen eingesetzt werden und in diesen Bereichen ihren Ursprung bzw. das Hauptanwendungsgebiet haben. Die hierbei verwendeten Ansätze gehen jedoch von einem hohen Grad der Datenintegration und einer qualitativ sehr guten Unterstützung des verwendeten Produktdatenmodells aus.

Dezentrale Datenverwaltung

Neben dem Client-Server-Modell werden auch dezentrale Ansätze verfolgt, wobei die Informationen nach dem Peer-to-Peer-Prinzip[2] (*P2P*) vollständig dezentral verwaltet werden. Mit diesem Ansatz wird vor allem eine sinnvolle Aufteilung der Informationen im Netz sowie die Vereinfachung einzelner Komponenten bzw. die Modularisierung des Gesamtsystems verfolgt. Diesen Vorteilen stehen die hohe Komplexität aus dem Zusammenspiel der (verteilten) Komponenten und der aufwendige Abgleich der Planungsdaten gegenüber.

Um die steigende Komplexität zu beherrschen, wird in der Informatik das Konzept der Aufgabenteilung verfolgt. Hierbei werden technische Aspekte, beispielsweise die Verteilung der Daten, von der problemspezifischen Programmlogik getrennt (*separation of concerns*). Im Idealfall sind ausschließlich fachspezifische Probleme zu betrachten, die als Bausteine in eine Umgebung mit allgemeingültigen Basisfunktionen eingebettet werden können. Beispielsweise werden mit der CORBA-Architektur verteilte Anwendungen realisiert, die nicht direkt mit den technischen Problemen rechnerübergreifender Objektzugriffe konfrontiert werden. Mit Technologien wie P2P, *Grid-Computing* und vor allem *Agentensystemen* werden durch die Informatik sehr leistungsfähige Ansätze für das verteilt kooperative Arbeiten bereitgestellt, die neue bzw. ergänzende Konzepte der Kooperation erlauben. Die Anwendungsmöglichkeiten dieser Technologien und die hierfür erforderlichen Erweiterungen sind derzeit ein Thema der Forschung, deren Entwicklung durch die Aussicht auf intelligente, flexible, leicht erweiterbare sowie nutzerzentrierte Lösungen motiviert wird.

Mit der Idee der dezentralen Datenhaltung wird vor allem das Konzept des Modellverbunds verfolgt, das einen hohen Anteil der Integrationslogik in die Organisation des Rechnerverbundes verlagert. Dadurch werden Anwenderprogramme entlastet und Lösungen gefördert, die das notwendige Integrationswissen der verteilten Umgebung bereitstellen. Dies setzt eine allgemein akzeptierte Integrationsumgebung voraus, die derzeit nicht existiert. Zusätzlich wird eine Mindestmenge an Integrationswissen benötigt, die den Zusammenschluss der dezentralen Daten ermöglicht. Abgesehen von der Verwaltung und dem Austausch von Doku-

[1] EDMserver™: http://www.epmtech.jotne.com/
EuroStep ModelServer: http://www.eurostep.com/prodserv/ems/ems.html
BSPro COM-Server: http://www.bspro.net/

[2] „In einem Peer-to-Peer-Netz sind alle Computer gleichberechtigt und können sowohl Dienste in Anspruch nehmen als auch Dienste zur Verfügung stellen“ (http://de.wikipedia.org/wiki/Peer-to-Peer). Im Sinne des Client-Server-Modells ist jeder Peer somit zugleich Client und Server.

menten haben dezentrale Ansätze für die Datenintegration daher noch keine praktische Bedeutung erlangt.

2.4.2 Zugriff auf die Planungsdaten

Für die Bearbeitung in langen Transaktionen wird in der Regel eine lokale Kopie der benötigten Planungsdaten angelegt. Dadurch kann unabhängig von Netzwerkschwankungen gearbeitet werden. Durch die Strategie, die benötigten Planungsdaten mit möglichst wenigen Abfragen in den lokalen Arbeitsbereich zu übertragen, kann die Netzwerkbelastung zusätzlich reduziert werden (Katranuschkov 2001).

Mit der Anwendung des Client-Server-Modells werden Planungsdaten zentral verwaltet. Über die netzwerkfähige Schnittstelle des Servers (*Application Programming Interface - API*) wird dem Client der Zugriff auf die Planungsdaten ermöglicht. Diese Schnittstelle erlaubt die Kommunikation zwischen Client und Server, die über die Art und Leistungsfähigkeit der Interaktion entscheidet. Für den Zugriff auf die Planungsdaten kann zwischen drei Stufen unterschieden werden:

1. Generische Abfragen auf der Ebene des Modellierungsparadigmas. Hierzu gehört beispielsweise das *Standard Data Access Interface* (SDAI, ISO 10303-22 IS 1998), das eine Abfrage und Manipulation einzelner Objekte bzw. Attribute erlaubt.
2. Der Einsatz deklarativer Abfragesprachen, die ein Zusammenstellen der gewünschten Daten über die Kombination verschiedener Auswahlkriterien ermöglichen. Für relationale Datenbanken hat sich beispielsweise die *Structured Query Language* (SQL) etabliert. Im Bereich der Produktdatenverwaltung werden unter anderem EXPRESS-X (Denno 1999) und die *Partial Model Query Language* (Adachi Y. 2002) verwendet.
3. Der Zugriff über datenmodellspezifische Abfragen, die vordefinierte, an das verwaltete Datenmodell angepasste Funktionen bereitstellen. Eine solche problembezogene API wird z. B. in dem Projekt BLIS-SABLE[1] genutzt. Dadurch wird ein vereinfachter, fachspezifischer Umgang mit komplexen Datenmodellen erreicht.

Für die Anwendung langer Transaktionen bieten deklarative Abfragesprachen einen wesentlichen Vorteil, da die gesuchten Daten mit einer einzelnen Abfrage ohne Verzicht auf Flexibilität beschrieben werden können. Der Aspekt der Flexibilität besitzt durch die Heterogenität der verwendeten Anwenderprogramme eine hohe Bedeutung, da jeder Client, auch wenn er eine bekannte fachliche Problemstellung abdeckt, einen individuellen Informationsbedarf besitzt.

Abfrage der benötigten Planungsdaten

Für die Auswahl der benötigten Planungsdaten kann zwischen statischen und dynamischen Kriterien unterschieden werden. Mit den statischen Auswahlkriterien wird festgelegt, welche Objektklassen für ein bestimmtes Problem relevant sind. Dagegen wird mit den dynamischen Auswahlkriterien bestimmt, welche Planungsdaten für eine konkrete Aufgabe benötigt werden. Diese Zweiteilung ist vor allem für fachübergreifende Datenmodelle sinnvoll, die von fachspezifischen Clients meist nicht vollständig interpretiert werden können. Eine solche Unterteilung wurde in dem Projekt COMBINE 2 (Lockley & Augenbroe 2000) verfolgt, das auf der Grundlage eines integrierten Datenmodells (IDM) zunächst fachspezifische Sichten definiert und diese über zusätzliche Kriterien einschränkt. Ein ähnlicher Ansatz ist für die Nut-

[1] *SABLE – Simple Access to the Building Lifecycle Exchange,* http://www.blis-project.org/~sable/

zung des IFC-Modells vorgesehen, wobei für ausgewählte Austauschszenarien eine statische Teilmenge des Gesamtmodells beschrieben wird (Steinmann 2005). Durch die Wahl einer Problemklasse bzw. einer bestimmten Modellsicht sind für die Auswahl der benötigten Planungsdaten in der Regel nur wenige zusätzliche Nutzerangaben erforderlich. Auf diese Weise ist es möglich, den Informationsbedarf über wenige Auswahlkriterien sehr genau zu beschreiben. Ein solches Vorgehen ist mit dem Datenbankprinzip der *Prepared Statements* vergleichbar, das durch die Belegung der freigehaltenen Parameter eine Auswahl der Daten vornimmt. Dadurch wird in erster Linie die Beschleunigung der Abfrageverarbeitung verfolgt, die für eine spezielle Problemstellung entworfen ist und somit weniger auf die Vereinfachung oder Flexibilisierung der Abfrage abzielt.

Durch die Komplexität der Datenmodelle ist ein Vordefinieren von Abfragen unerlässlich. Eine solche Abfrage muss auf den Informationsbedarf einer Planungsaufgabe abgestimmt und flexibel einsetzbar sein. Diese Anforderung wird durch Abfragesprachen wie EXPRESS-X und PMQL nur teilweise erfüllt. Die Auswahl der Planungsdaten erfordert die Unterstützung eines speziell abgestimmten Clients, der die notwendigen Angaben des Nutzers minimiert und hieraus eine entsprechende *Partialmodellabfrage* generiert. Hierfür ist zusätzliche Programmlogik erforderlich, die nur teilweise durch vordefinierte Abfragebausteine ersetzt werden kann.

Anbindung der Anwenderprogramme

Die Rückgabe der Planungsdaten kann in unterschiedlichen Datenformaten erfolgen. Die ausgewählten Planungsdaten werden in einem entsprechenden Datenstrom an den Client übertragene, in Datenobjekte zurückgewandelt und anschließend lokal bearbeitet. Alternativ ist die Speicherung als Datei möglich, die auf dem Rechner des Clients einem Anwenderprogramm zugänglich gemacht werden kann. Eine solche Lösung ist vor allem deshalb interessant, weil (1) bereits vorhandene, dateibasierte Anwenderprogramme genutzt werden können und (2) noch keine allgemein akzeptierte API für die Client-Server-Kommunikation existiert. Nach diesem Prinzip sind Lösungen entstanden, die über einen Internet-Browser die Interaktion mit dem Server realisieren.

2.4.3 Abgleich der Planungsdaten

Nach einem Bearbeitungs- oder Planungsschritt ist die gemeinsame Datenbasis zu aktualisieren. Hierbei überlagern sich die Probleme der Kommunikation und Koordination. Die damit verbundenen Schwierigkeiten sind in den vorangegangenen Abschnitten beschrieben. Hieraus wird deutlich, dass das Aktualisieren der gemeinsamen Datenbasis nur durch einschränkende Vorgaben, zusätzliche Nutzerinteraktionen und begleitende Hilfestellungen handhabbar ist.

Übertragen der Änderungen in die gemeinsame Datenbasis

Das Aktualisieren der gemeinsamen Datenbasis setzt formal voraus, dass die zu übertragenden Änderungen interpretiert und gemäß ihrer Bedeutung in die Datenbasis integriert werden können. Das hierfür erforderliche Wissen ist im Allgemeinfall sehr hoch und steht der Datenverwaltung nicht oder nur unvollständig zur Verfügung. Aus diesem Grund ist es notwendig, das Integrationswissen auf ein handhabbares Maß zu reduzieren.

Mit der Verwendung standardisierter Datenmodelle wird das erforderliche Integrationswissen reduziert, indem die Datenverwaltung von der Datentransformation entlastet wird. Dieser Ansatz ist jedoch nur dann ohne die Interpretation der Daten erfolgreich, wenn 1) die Identifizierbarkeit der Daten und 2) eine normalisierte Darstellung der Planungsinformationen garan-

tiert werden kann. In praxisrelevanten Datenmodellen wie IFC und CIS/2 sind sowohl die Identifizierbarkeit als auch die normalisierte Darstellung nicht erfüllt. Die Datenverwaltung ist somit auf zusätzliches Fachwissen oder eine entsprechende Funktionalität der interagierenden Clients angewiesen. Das Aktualisieren einer gemeinsamen Datenbasis ist daher vor allem in herstellerspezifischen Lösungen zu finden, die eine gute Abstimmung zwischen den Datenmodellen, der Datenverwaltung und den verwendeten Anwenderprogrammen realisiert haben.

Änderungskontrolle und Benachrichtigungsdienste

Durch das Aktualisieren der Datenbasis können Widersprüche zu vorangegangenen Planungsleistungen entstehen. Hieraus entsteht der Wunsch, über wichtige Änderungen und entstandene Inkonsistenzen informiert zu werden. Jeder Planer hat hierbei einen unterschiedlichen Informationsbedarf, der sich aus seiner Aufgabe und Zuständigkeit ergibt.

Das Erkennen von Inkonsistenzen setzt voraus, dass der Datenverwaltung die einzuhaltenden Konsistenzbedingungen bekannt sind. In Datenmodellen wie IFC und CIS/2 sind die hierfür erforderlichen Konsistenzbedingungen nur teilweise formalisiert und durch die Vielfalt möglicher Widersprüche auch nur schwer beherrschbar. Folglich ist nur ein Teil der Inkonsistenzen automatisch erkennbar und erzwingt das Überwachen der Änderungen durch die verantwortlichen Planer.

Das Überwachen der Planungsdaten ist ein Thema der Änderungskontrolle. Um den Aufwand für die einzelnen Planer möglichst gering zu halten, werden Benachrichtigungsdienste genutzt, die über wichtige Änderungen informieren sollen. Das Einrichten eines Benachrichtigungsdienstes ist vergleichbar mit der Definition von Abhängigkeiten und weist ähnliche Probleme auf. Dadurch droht die Gefahr der Überorganisation bzw. einer Überflutung mit Benachrichtigungen, die beide ein effizientes Arbeiten verhindern.

Sperrmechanismen und Konfliktlösung

Paralleles Arbeiten kann zu Widersprüchen und gegenseitigen Behinderungen führen. Das Ziel der Datenverwaltung liegt zunächst darin, das ungewollte Überschreiben parallel durchgeführter Planungsleistungen zu verhindern. Hierfür ist das Überwachen und Synchronisieren der Datenzugriffe notwendig, das vor allem über Sperrmechanismen realisiert wird. Im einfachsten Fall werden pessimistische Verfahren verfolgt, die die verwendeten Planungsdaten für den Zeitraum der Bearbeitung exklusiv vor dem Verändern durch andere Planer schützen. Dieses einfache, robuste aber auch sehr restriktive Verfahren wird bevorzugt, weil keine divergierenden Planungsstände entstehen und viele Planungskonflikte vermieden werden können. Abweichend vom pessimistischen Kooperationsansatz wird von Bentley (1998) ein optimistisches Verfahren beschrieben, das beim Aktualisieren der Datenbasis über *Einbringsperren* den Abgleich mit den zwischenzeitlich geänderten Planungsdaten erzwingt. Die Datenbasis wird dadurch auf einen aktuellen Planungsstand beschränkt und verhindert unwissentlich vorhandene Konflikte. Das sofortige Zusammenführen ist aber nicht immer gewünscht und kann den Planungsprozess behindern. In diesen Fällen ist die Verwaltung alternativer Planungsstände notwendig, die ein späteres Zusammenführen zu einem gemeinsamen Planungsstand ermöglicht. Aufgrund des höheren Verwaltungsaufwandes sind solche Ansätze derzeit noch Gegenstand der Forschung.

2.5 Diskussion

In diesem Kapitel wurden Probleme diskutiert, die mit der Verwaltung einer gemeinsam nutzbaren Datenbasis verbunden sind. Hierbei überlagern sich verschiedene Aspekte der kooperativen Planung, die selbst eigenständige Themen der Forschung sind und die Möglichkeiten der Datenverwaltung beeinflussen. Zu den bestimmenden Themen gehören die Kommunikation über den Bauwerksentwurf, die Koordination der Zusammenarbeit sowie die Umsetzung in verteilten Softwaresystemen.

Kommunikation

Betrachtet man die Probleme der Kommunikation, so wird deutlich, dass trotz der Integrationsbemühungen eine heterogene Modellwelt erhalten bleiben wird. Das Wissen der Modelle ist überwiegend in Anwenderprogrammen gekapselt und somit für die Datenintegration nicht unmittelbar nutzbar. Für den Austausch der Planungsinformationen werden neutrale Datenmodelle verwendet, die auf die Beschreibung eines Planungsstandes beschränkt sind. Die gemeinsame Grundlage bilden ähnliche Modellierungsphilosophien, die im Idealfall auf das gleiche Meta-Modell zurückzuführen sind. Über diese Abstraktionsebene wird ein einheitlicher Zugriff auf die unterschiedlichen Datenmodelle sowie die Schaffung modellunabhängiger Dienste ermöglicht. Die Interpretation der Modelldaten wird durch die Standardisierung der Datenmodelle erreicht. Dadurch wird der Datenabgleich möglich, der die verschiedenen Modelle miteinander synchronisiert. Für die Vereinfachung des Datenabgleichs wird die Harmonisierung der Datenmodelle verfolgt, die zu unterschiedlichen Integrationsansätzen auf der Ebene der Modelldefinition geführt hat. Trotz dieser Vereinfachungen bleiben die notwendigen Datentransformationen komplex und sind in der Regel nicht vollständig formalisierbar. Der Informationsaustausch ist daher durch Informationsverluste und strukturelle Veränderungen der Daten gekennzeichnet.

Koordination

Die Koordination der Zusammenarbeit ist die Grundlage für das Erreichen der Planungsziele und entscheidet über die Zugriffe auf die gemeinsamen Planungsdaten. Für akzeptable Planungszeiten ist ein flexibles, paralleles Arbeiten unerlässlich, das den Anforderungen einer konsistenten Datenbasis jedoch entgegensteht. Das Überwachen der Datenzugriffe kann helfen, diese gegensätzlichen Ziele miteinander zu verbinden. Eine Bearbeitung über kurze Transaktionen oder die Verwendung von Sperrmechanismen sind jedoch kaum geeignet, um die typischen Aufgaben des Ingenieurwesens zu unterstützen. Hierfür werden lange Transaktionen bevorzugt, die eine unabhängige Bearbeitung der Planungsdaten erlauben. Divergierende Planungsstände lassen sich hierbei nicht vermeiden und erfordern zusätzliche Abstimmungsprozesse, die die entstandenen Widersprüche beseitigen. Die gemeinsame Datenbasis muss diesen Abstimmungsprozess zwangsläufig unterstützen.

Durch paralleles Arbeiten ist das Verwalten mehrerer Planungsstände erforderlich. Dies zeigen die Vergleiche mit der Softwareentwicklung und dem Maschinenbau. Verfügbare Ansätze lassen sich aber nicht einfach auf das Bauingenieurwesen übertragen, da unterschiedliche Anforderungen und Voraussetzungen bestehen. Die Gemeinsamkeit der beschriebenen Versionsansätze ist ihre enge Bindung an das objekt-orientierte Konzept. Mit diesen Ansätzen kann die Dokumentation der Planungsschritte aber nur unter optimalen Bedingungen geleistet werden. Durch die Probleme der Datenintegration gehen wichtige Informationen verloren, die gemeinsam mit den umfangreichen Entwurfsänderungen eine besondere Anforderung an die

Dokumentation der Planungsschritte stellt. Das Eingliedern der veränderten Planungsstände ist vor diesem Hintergrund eine Besonderheit des Bauingenieurwesens.

Softwaresysteme für verteiltes kooperatives Arbeiten

Bei der Umsetzung in Softwaresystemen wird zwischen zentralen und dezentralen Ansätzen der Datenverwaltung unterschieden. Dezentrale Ansätze haben wegen ihrer höheren Komplexität in der Planungspraxis noch keine Bedeutung erlangt. Die Client-Server-Architektur ist dagegen weitestgehend akzeptiert und vergleichsweise einfach strukturiert. Diese Architektur ist weniger flexibel, besitzt jedoch klar festgelegte Informationsflüsse. Durch die zentrale Datenbasis wird nicht nur der Zugriff, sondern auch die Verwaltung der Planungsdaten erleichtert. Die Änderungen von Planungsdaten können dadurch einfacher überwacht werden. Unabhängig von dieser relativ restriktiven Software-Architektur ist der Zugriff auf die Daten nach individuellen Anforderungen frei variierbar. Er reicht von einzelnen Objekten bis hin zu kompletten Planungsständen. Hierbei können auch Standardwerkzeuge eingesetzt werden, die beispielsweise über Web-Portale den dateibasierten Datenaustausch unterstützen können. Auf diese Weise ist auf Anwenderseite ein hohes Maß an Flexibilität gegeben. Die Anforderungen an die Clients sind überschaubar und mit niedrigen Einstiegshürden verbunden. Im Gegensatz dazu können im Server alle Funktionen der Datenverwaltung zusammengeführt werden.

3 Konzeption einer kooperativ nutzbaren Datenbasis

Die Verwaltung gemeinsamer Planungsdaten wird mit vielfältigen, mitunter widersprüchlichen Problemstellungen konfrontiert. Um die gewünschte Funktionalität zu erreichen, werden Vereinbarungen über Datenmodelle, Softwarearchitekturen und Arbeitsabläufe getroffen. Solche Vereinbarungen sind mit Einschränkungen verbunden und in der hochgradig heterogenen Bauindustrie schwer durchsetzbar. In diesem Kapitel wird ein Ansatz vorgestellt, der auf der Dokumentation von Planungsänderungen beruht und durch eine fehlertolerante und flexibel einsetzbare Datenverwaltung diese Einschränkungen verringern kann.

Im *ersten Abschnitt* werden grundlegende Annahmen des Lösungsansatzes diskutiert. Hierzu gehören: 1.) die Akzeptanz der Modellvielfalt, die die Nutzung eines allgemeingültigen Meta-Modells erfordert, 2.) die Unterstützung einer optimistischen Kooperationsstrategie, die paralleles Arbeiten in langen Transaktionen ermöglicht, 3.) die Notwendigkeit der Versionsverwaltung, die auch Änderungen am Objektgefüge dokumentieren muss, sowie 4.) die mögliche Umsetzung in einer Client-Server-Architektur, die das verteilte Arbeiten unterstützt.

Der *zweite Abschnitt* stellt den entwickelten Lösungsansatz vor und diskutiert die Idee der Änderungsdokumentation. Hierfür werden die Operationen der Änderungssemantik und deren Abbildung in das erarbeitete Versionsmodell vorgestellt. Darüber hinaus werden die Verwendung von Teildatensätzen und das Prinzip zur nachträglichen Bestimmung von Änderungen beschrieben, um den für die betrachtete Anwendung wichtigen Umgang mit vollständigen Planungsständen zu unterstützen.

Der *dritte Abschnitt* geht auf die Anwendung des Lösungsansatzes ein. Hierzu gehören: 1.) die Nutzung der Versionshistorie, die den Umgang mit Teildatensätzen ermöglicht, 2.) die Unterstützung der parallelen Bearbeitung, die auf das Sperren von Planungsdaten verzichtet, sowie 3.) die Abbildung von Änderungen im Objektgefüge, die die gewünschte Flexibilität garantiert.

3.1 Anforderungen und Annahmen

Im Kapitel 2 wurden Problemstellungen diskutiert, die für die Verwaltung einer gemeinsamen Datenbasis relevant sind. Neben diesen vorwiegend technischen Aspekten existieren für das Bauwesen weitere Besonderheiten, die Einfluss auf die Akzeptanz und Anwendbarkeit neuer Technologien haben. In der Literatur werden unter anderem folgende Eigenschaften genannt (Hauschild 2001, Hannus et al. 1995):

- Unikatcharakter der Bauwerke,
- Hohe relative Entwurfskosten pro Bauwerk,
- Strukturierung der Bauindustrie,
- Hoher Abstraktionsgrad der Dokumente bzw. Modelle.

Hauschild nennt weitere Eigenschaften, die in ihrer Gesamtheit deutlich machen, dass die Voraussetzungen der Bauindustrie für die Nutzung einer gemeinsamen Datenbasis nicht mit anderen Industrien vergleichbar sind. Es wird deutlich, dass die Datenintegration im Bauwesen nur schrittweise verbessert werden kann. Diese Erkenntnis und allgemein anerkannte Grundlagen der Forschung bilden den Ausgangspunkt des vorgestellten Ansatzes.

3.1.1 Grundlage der betrachteten Datenmodelle

Aus der Vielfalt der Modelle entsteht der Bedarf, die Modelldefinitionen auf eine gemeinsame Grundlage zurückzuführen. Erst dadurch können wiederverwendbare Methoden der Datenverwaltung entwickelt werden. Hierfür werden *Meta-Modelle* eingesetzt, die zumeist auf den Konzepten des objekt-orientierten Prinzips beruhen und für die Beschreibung neutraler Datenmodelle eingesetzt werden. Das in dieser Arbeit verwendete Meta-Modell wird daher auf allgemein anerkannte Konzepte zur Beschreibung objekt-orientierter Datenmodelle zurückgeführt. Hierzu gehören:

- die Strukturierung der Informationen in Form von *Objekten*,
- die Unterstützung der Vererbungsbeziehung sowie
- die Verwendung von Assoziationen und verschiedenen Attributtypen.

Das gewählte Meta-Modell lehnt sich an die standardisierte, im Ingenieurwesen weit verbreitete EXPRESS-Beschreibungssprache (ISO 10303-11, 1994) an und übernimmt die dort verwendeten Datentypen sowie die Möglichkeiten zur Definition von Relationen. Auf die Abbildung von Konsistenzbedingungen und Abhängigkeiten wird dagegen verzichtet, weil diese Informationen meist nicht oder nur sehr begrenzt vorhanden sind. Mit der Wahl von EXPRESS wird im Unterschied zu anderen objekt-orientierten Meta-Modellen zusätzlich auf folgende Konzepte verzichtet:

- die Beschreibung von Verhalten,
- die eindeutige Identifizierbarkeit der Objekte,
- die Verwendung besonderer Relationstypen wie Aggregation sowie
- die Kapselung und Verwendung von Zugriffsrechten.

Trotz dieser Einschränkungen ist die Ausdrucksstärke ausreichend, um den Zustand objekt-orientierter Datenmodelle zu erfassen. Die Identifizierbarkeit der Objekte ist für die Dokumentation der Planungsschritte eine wichtige Eigenschaft, ist aufgrund der unvollständig abgebildeten Datentransformationen aber nicht vollständig umsetzbar. Generell stellen der Verlust an Informationen und die Veränderung des Objektgefüges wesentliche Probleme für die Verwaltung der Planungsdaten dar.

Durch das allgemein gehaltene Meta-Modell wird die Unabhängigkeit von konkreten Datenmodellen erreicht. Eine flexible Lösung wird auch bezüglich der Integrationsproblematik verfolgt, indem der Informationsgehalt des zu verwaltenden Datenmodells nicht näher festgelegt wird. Die gemeinsame Datenbasis lässt sich dadurch mit verschiedenen Integrationsansätzen kombinieren. Als Anwendungsbeispiel wird die Nutzung eines zentralen, standardisierten Kernmodells sowie die Ergänzung um Fachmodelle favorisiert. Ein solcher Ansatz erleichtert den Zugriff auf die Planungsdaten und ist mit vorhandenen Integrationsbemühungen kombinierbar.

3.1.2 Betrachtetes Kooperationsmodell

Die rechnergestützte Kooperation lässt sich nicht auf eine Kooperationsstrategie oder auf ein Anwendungsszenario einschränken. Dennoch liegt der Schwerpunkt auf langen Transaktionen, die parallel und ohne das Sperren von Objekten durchgeführt werden können. Dies hat im wesentlichen zwei Gründe. Einerseits entspricht dies der gewohnten Arbeitsweise und

bedeutet für die bestehende Softwarelandschaft die geringsten Umstellungen. Andererseits werden Problemstellungen thematisiert, die auch in anderen Szenarien relevant werden.

Die Arbeit auf einer gemeinsamen Datenbasis ist durch zahlreiche Planungsschritte gekennzeichnet, die im Idealfall durch eine übergeordnete Ablaufplanung koordiniert werden. Ein Planungsschritt lässt sich auf folgende Sequenz zurückführen (siehe Abbildung 3-1):

1. das Lesen der benötigten Planungsdaten von der gemeinsamen Datenbasis,
2. die Bearbeitung der Planungsdaten in einem eigenen Arbeitsbereich und
3. das Zurückschreiben der geänderten Planungsdaten in die gemeinsame Datenbasis

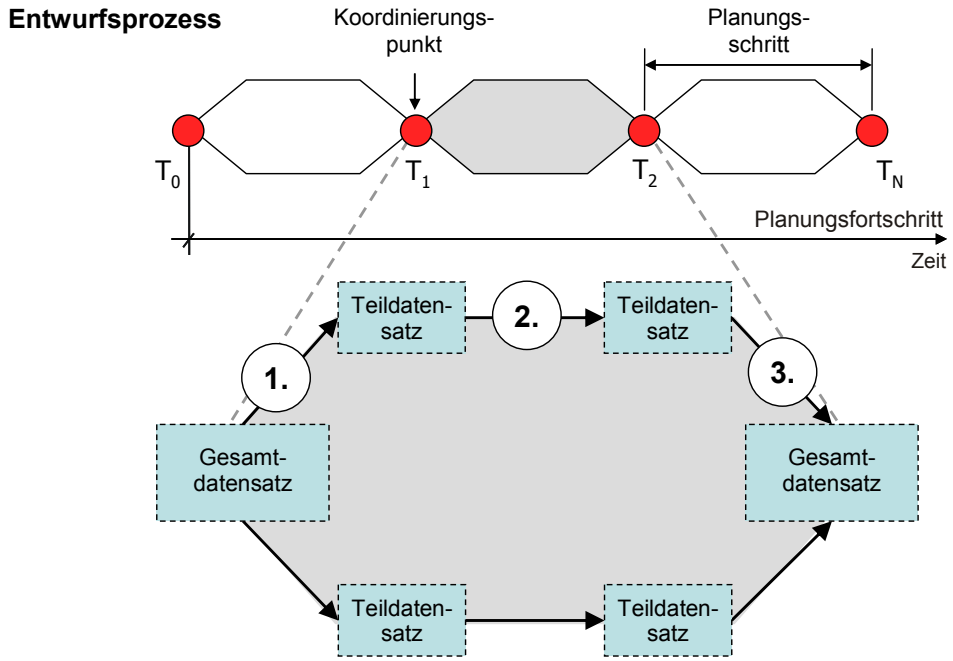


Abbildung 3-1 Planungsschritt und seine Einordnung in den Entwurfsprozess in Anlehnung an das Kooperationsmodell nach Katranuschkov (2001)

In einem Planungsschritt wird meist ein großer Teil der Planungsdaten verwendet, die in einem langen Bearbeitungszeitraum umfangreiche Änderungen erfahren. Aus diesem Ablauf lassen sich folgende Anforderungen ableiten (Weise et al. 2004b):

- die Auswahl der benötigten Planungsdaten,
- die Transformation der gewählten Planungsdaten in eine für die Bearbeitung geeignete Struktur, wobei beide Transformationsrichtungen relevant werden,
- ein Vergleich der Planungsdaten, um die Änderungen zwischen dem alten und dem überarbeiteten Planungsstand zu erkennen und
- das Wiedereingliedern der veränderten Planungsdaten in die gemeinsamen Datenbasis.

Diese Anforderungen charakterisieren einen einzelnen Planungsschritt. Für das parallele Arbeiten ist zusätzlich das Zusammenführen der unabhängig voneinander durchgeführten Änderungen notwendig. Dieser Schritt ist jedoch nicht immer unmittelbar möglich und erfordert die Verwaltung divergierender Planungsstände. Um das Auflösen widersprüchlicher Entwurfsentscheidungen zeitlich flexibel gestalten zu können, soll durch die Datenverwaltung ein wechselseitiges Vorschlag-Zustimmungsverfahren unterstützt werden.

3.1.3 Zwänge der Versionsverwaltung

Das Ziel der Versionsverwaltung ist die Dokumentation der Planungsschritte. Sie ist in vielen arbeitsteilig organisierten Anwendungsgebieten eine unentbehrliche Grundlage zur Unterstützung der kooperativen Arbeit. Firmenich (2001) sieht in der Versionsverwaltung auch einen Vorteil für die Bauplanung und schlägt in seiner Arbeit einen Versionsansatz für die Anwendung in langen Transaktionen vor. Im Unterschied zu seinem Ansatz wird in dieser Arbeit die Forderung nach eindeutiger Identifizierbarkeit der Objekte aufgegeben. Neben unvollständig abgebildeten Datentransformationen sind die in frühen Phasen des Entwurfs oft erheblichen Änderungen ein wesentlicher Grund für diese Entscheidung. Die damit verbundene Verfeinerung des Entwurfs führt zwangsläufig zu Veränderungen im Objektgefüge, die über das Konstrukt der Objektkennung nicht mehr nachvollziehbar sind.

In einer kooperativ nutzbaren Datenbasis sollen die Planungsstände als Grundlage der Planungsdokumentation dienen. Zur einer geeigneten Planungsdokumentation gehören aber auch Beziehungen, die zwischen den Planungsständen die Entwicklung des Entwurfs erkennen lassen. Da diese Beziehungen in der Regel nicht verfügbar sind, muss zusätzlich das Wiederherstellen der hierfür notwendigen Änderungsinformationen aus den voneinander abhängigen Planungsständen betrachtet werden.

An die Planungsdokumentation werden schließlich folgende Anforderungen gestellt:

- Die Beziehung zwischen Planungsständen soll auf der Ebene der Objekte dokumentiert werden.
- Änderungen im Objektgefüge sollen über die Versionsbeziehungen beschreibbar sein.
- Die übergebenen Planungsstände dürfen nicht verändert werden und müssen eindeutig wieder herstellbar sein.
- Es wird eine platzsparende Speicherung der Daten angestrebt.

Die Planungsdokumentation soll darüber hinaus nur mit Datenmodellen kombiniert werden, die auf die Abbildung eines Planungsstandes beschränkt sind und somit nicht selbst eine Versionsverwaltung realisieren[1].

3.1.4 Gewählte Softwarearchitektur

Die Einbettung der Datenverwaltung in eine verteilte Softwarearchitektur gehört nicht unmittelbar zum Thema dieser Arbeit, ist für eine prototypische Umsetzung jedoch nötig. Verallgemeinernd wird davon ausgegangen, dass die vorgeschlagene Datenbasis

- in einem Netzwerk verfügbar ist,
- durch mehrere Anwender gleichzeitig benutzt werden kann sowie
- eine Ergänzung um zusätzliche Dienste erlaubt.

Für dieses Szenario bietet sich die Verwendung einer Client-Server-Architektur an, die eine vergleichsweise einfache Umsetzung ermöglicht und in der Ingenieurpraxis bereits erfolgreich eingesetzt wird (Eastman 1999). Wie in Abbildung 3-2 gezeigt, übernimmt die erweiterbare Datenbasis die Aufgaben des Servers und wird durch Anwenderprogramme als die dezentra-

[1] In praxisrelevanten Datenmodellen werden, wenn überhaupt, meist unkritische Versionsinformationen wie der Zeitpunkt von Änderungen abgebildet, die mit der vorgeschlagenen Versionsverwaltung nicht kollidieren.

len Clients vervollständigt. Eine solche Architektur soll in dieser Arbeit die Grundlage bilden, um die Anwendung eines zentralen, standardisierten Kernmodells zu ermöglichen.

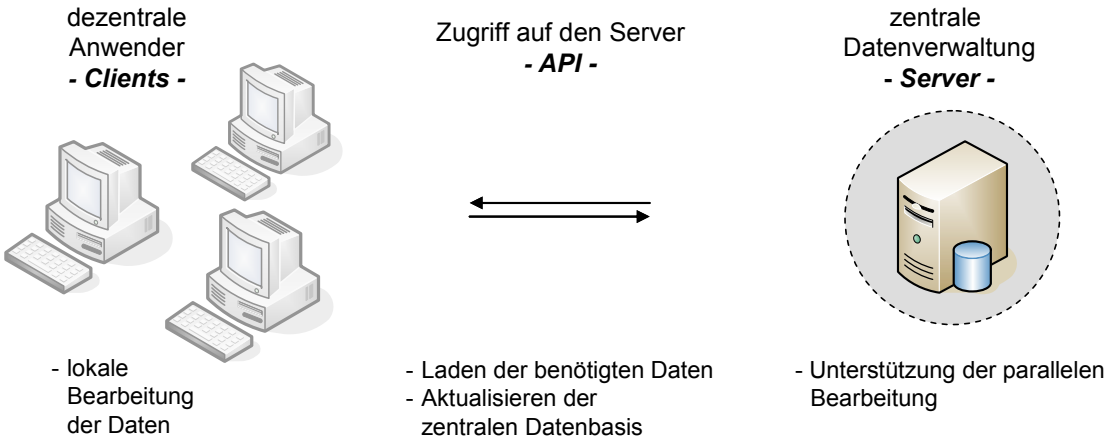


Abbildung 3-2 Prinzipieller Aufbau der Client-Server-Architektur, die für die prototypische Umsetzung genutzt werden soll.

Mit der Wahl einer Client-Server-Architektur wird eine Lösung verfolgt, die eine möglichst geringe Bindung zwischen der Datenbasis und den Anwenderprogrammen erlaubt. Hierbei sollen auch der Datenaustausch über Dateien und die unabhängige Bearbeitung der Planungsdaten unterstützt werden. Mit dieser Entscheidung können Anwenderprogramme verwendet werden, die nur den dateibasierten Import und Export der Planungsdaten erlauben und die Datentransformation intern realisieren. Dies bedeutet, dass 1.) die Datentransformation nicht ausschließlich als Aufgabe der Datenverwaltung betrachtet werden kann und 2.) der Datenaustausch auf die Übertragung von Planungsständen beschränkt ist.

Für die Interaktion zwischen der Datenbasis und den Anwenderprogrammen ist eine Programmschnittstelle (API) notwendig, die das gewählte Kooperationsszenario unterstützt. Eine solche Programmschnittstelle muss in der Lage sein,

- die benötigten Planungsdaten für die Bearbeitung zu laden,
- die Nutzung verschiedener Dienste, wie beispielsweise den Datenvergleich, zu ermöglichen sowie
- das Eingliedern und Zusammenführen der veränderten Planungsdaten zu unterstützen.

Für die Umsetzung können verschiedene Technologien und Spezifikationen genutzt werden, die in ihrem Funktionsumfang vergleichbar sind. Ein allgemein akzeptierter Standard existiert derzeit jedoch nicht, so dass im Rahmen dieser Arbeit eine eigenständige, auf die erforderliche Funktionalität abgestimmte Lösung favorisiert wird.

3.2 Überblick über den erarbeiteten Lösungsansatz

Der vorgestellte Lösungsansatz beruht auf der Erkenntnis, dass für das Arbeiten auf einer gemeinsamen Datenbasis das ***Erkennen der vorgenommenen Änderungen*** eine der wichtigsten Informationen in einer durch Iterationen geprägten Planungspraxis ist. Die vorgenommenen Änderungen besitzen oft die größte Aussagekraft, um die Auswirkungen auf die eigenen Planungsleistungen einschätzen zu können. Sie stehen daher im Mittelpunkt der entwickelten Methoden.

Aus dieser Erkenntnis resultiert die Idee, die gemeinsame Datenbasis auf die Dokumentation der Änderungen zurückzuführen. Damit werden folgende Vorteile verbunden:

- Die vorgenommenen Änderungen eines Planungsschrittes stehen unmittelbar zur Verfügung und müssen auf der Basis zweier Planungsstände nicht neu berechnet werden.
- Entstandene Informationsverluste können durch das Zurücknehmen von Änderungen ausgeglichen werden.
- Ein auf Änderungen beruhendes Verfahren verringert den Speicherplatzbedarf.

Es wird erwartet, dass der höhere Aufwand zum Wiederherstellen kompletter Planungsstände für die vorgesehene Anwendung kein Problem darstellt, da die Interaktionen mit der gemeinsamen Datenbasis nicht als zeitkritisch zu bewerten sind und vergleichsweise selten erfolgen.

Drei Fragestellungen werden für die Umsetzung der auf Differenzmengen beruhenden Idee sowie für dessen Anwendung relevant.

1. Die adäquate Erfassung von Änderungen im Objektgefüge.
2. Die Ermittlung der Änderungen zwischen zwei Planungsständen.
3. Die Unterscheidung zwischen gelöschten und verlorengegangenen Planungsdaten.

Für die Beschreibung des Lösungsansatzes interessiert zunächst, welche Informationen bei der Dokumentation eines Planungsschrittes erfasst werden. Diese Informationen werden in eine Differenzmenge überführt, die in dem vorgeschlagenen Versionsansatz abgebildet wird. Schließlich ist zu klären, wie die Änderungen eines Planungsschrittes aus Planungsständen ermittelt werden können.

3.2.1 Dokumentation eines Planungsschrittes

Zu Beginn eines Planungsschrittes erfolgt die Auswahl eines Planungsstandes, auf dessen Grundlage die benötigten Planungsdaten in einen privaten Arbeitsbereich geladen werden. Die ausgewählten Planungsdaten werden hierbei als Datenobjekte mit all ihren Attributen in den privaten Arbeitsbereich übertragen und unabhängig von der gemeinsamen Datenbasis modifiziert. Die vorgenommenen Änderungen erzeugen beim Zurückschreiben in die gemeinsame Datenbasis einen neuen, unveränderbaren Planungsstand und stehen anschließend für weitere Planungsschritte zur Verfügung. Gemäß dem Prinzip einer deltabasierten Versionsverwaltung werden hierbei nur die Änderungen des Planungsschrittes gespeichert. Der Planungsschritt wird somit in Bezug zum ursprünglich geladenen Planungsstand beschrieben. Die ableitbaren Änderungsinformationen werden schließlich durch die der Delta- bzw. Differenzmenge zugrunde liegenden Änderungssemantik bestimmt.

Verwendete Änderungssemantik

Die Änderungen werden im Rahmen des vorgestellten Ansatzes auf der Grundlage der Datenstruktur erfasst, wobei grundsätzlich zwischen dem

1. *Erzeugen* (create),
2. *Verändern* (change) und
3. *Löschen* (delete)

von Datenobjekten und Attributen unterschieden wird. Um auf der Basis der Änderungen ein Datenobjekt mit allen gültigen Attributen wiederherstellen zu können, wird zusätzlich der Bezug zwischen den *veränderten* Datenobjekten verwaltet. Als *Veränderung* wird aber nicht nur die Modifikation der Attribute sondern auch das

4. *Teilen* (split),
5. *Zusammenlegen* (unify) sowie die
6. *Evolution*[1] (type change)

von Datenobjekten und somit Änderungen im Objektgefüge unterstützt.

Im Vergleich zu operationsbasierten Verfahren (vgl. Firmenich 2004, Schroeder 1995) wird das *Verändern* von Datenobjekten nicht auf die verwendeten Modifikationsoperationen zurückgeführt und demzufolge keine weitere Differenzierung der Änderungssemantik verfolgt. Das in diesem Ansatz genutzte ergebnisorientierte Verfahren ist somit für die Bewertung der durchgeführten Änderungen weniger aussagekräftig, ist dafür aber nicht von der Bereitstellung der Modifikationsoperationen durch Anwenderprogramme abhängig. Das Prinzip des verfolgten Ansatzes ist in Abbildung 3-3 dargestellt. Der Planungsschritt des Planers n_A wird hierbei mit Hilfe der vorgestellten Änderungssemantik erfasst.

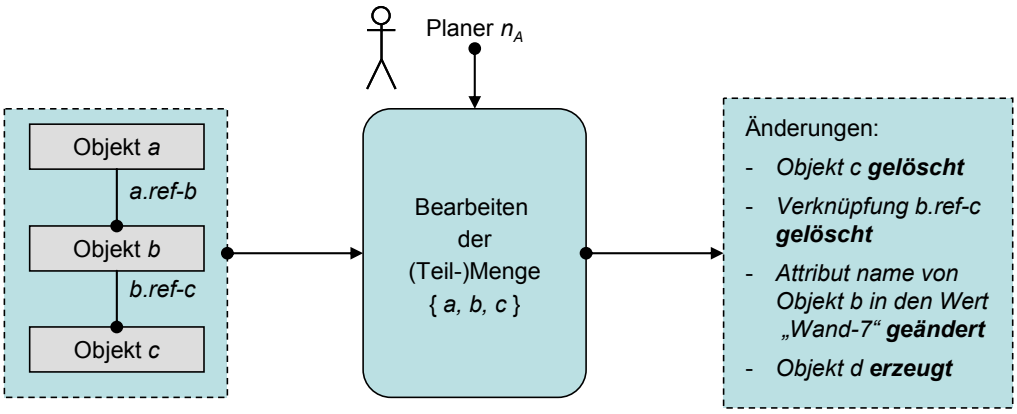


Abbildung 3-3 Darstellung eines Planungsschrittes durch das Erfassen der durchgeführten Änderungen

3.2.2 Verwaltung von Änderungen über Deltas

Die Änderungen eines Planungsschrittes werden in dem vorgeschlagenen Versionsansatz als Differenzmenge gespeichert, die auf der Vorgängerversion und damit auf *Vorwärtsdeltas* beruht. Im Vergleich zum Rückwärtsdeltaverfahren kann ein Umordnen der Differenzmengen und somit die zeitweilige Behinderung der Datenzugriffe vermieden werden. Der mit zuneh-

[1] Mit dem Begriff *Evolution* ist die Veränderung des Objekttyps (type change) gemeint.

mender Anzahl von Versionen ansteigende Wiederherstellungsaufwand wird demgegenüber für eine überschaubare Anzahl von Planungsschritten als unkritisch eingeschätzt.

In dem vorgeschlagenen Versionsansatz wird die Differenzmenge eines neuen Planungsstandes konzeptionell auf folgende Elemente zurückgeführt:

1. (Delta-)Objekte,
2. Versionsbeziehungen sowie
3. eine Modellmenge, die alle aktuellen (Delta-)Objekte des Planungsstandes enthält.

(Delta-)Objekte

Ein (Delta-)Objekt besitzt zunächst alle Eigenschaften, um ein Datenobjekt gemäß dem genutzten Meta-Modell vollständig zu beschreiben. Die verwendeten Objekte definieren aber nur die Attribute, die sich gegenüber einem oder mehreren ihrer unmittelbaren Vorgänger verändert haben. Um das Löschen von Attributen zu ermöglichen, wird zusätzlich ein neuer Attributtyp eingeführt, der für das Markieren gelöschter Attribute verwendet wird. Die *Änderungen an einem Objekt* werden somit auf der Ebene von Attributen in einem neuen Objekt erfasst.

Versionsbeziehungen

Über die Versionsbeziehung wird ein nicht näher definierter logischer Bezug zwischen Objekten unterschiedlicher Planungsstände erfasst. In dem vorgeschlagenen Versionsmodell wird diese Beziehung als eine Vorgänger-Nachfolger-Verknüpfung interpretiert, die damit die Basis der gespeicherten Änderungen definiert. Für die Verknüpfung von Objekten wird zunächst nur gefordert, dass der entstehende Objektgraph zyklenfrei bleibt und nur durch das Hinzufügen von Nachfolgern modifiziert wird. Über Versionsbeziehungen sind dadurch auch *Änderungen im Objektgefüge* beschreibbar.

Planungsstände

Aus der Menge aller Objekte stellen nur bestimmte Objektkombinationen einen zulässigen Planungsstand dar. Eine solche Objektkombination wird über eine Modellmenge definiert, die alle aktuellen Objekte eines Planungsstandes enthält. Um die referenzielle Integrität zwischen geänderten und unveränderten Objekten zu wahren und gleichzeitig das Entstehen weiterer, aus der Aktualisierung von Referenzen resultierender, vermeintlich geänderter Objekte zu vermeiden, wird in dem vorgeschlagenen Ansatz bei Wahrung der Eindeutigkeit auf das Aktualisieren von Referenzen verzichtet (vgl. *version proliferation,* Conradi & Westfechtel 1998). Auf der Grundlage zweier Modellmengen können somit die *geänderten Objekte* zwischen den beiden zugehörigen Planungsständen identifiziert werden.

Die Änderungen aus dem Beispiel der Abbildung 3-3 können mit dem vorgestellten Versionsansatz wie in Abbildung 3-4 gezeigt ausgedrückt werden. In dieser Darstellung wird aus dem unveränderten Objekt *a* auch der Verzicht auf die Aktualisierung von Referenzen erkennbar. Für hochgradig vernetzte Datenmodelle können dadurch die abzuspeichernden Differenzen deutlich verringert und in ihrer Aussagekraft erhöht werden. Darüber hinaus ist erkennbar, dass die Versionsbeziehung nur die Verknüpfung der Objekte beschreibt und nicht für die Speicherung der Differenzen genutzt wird.

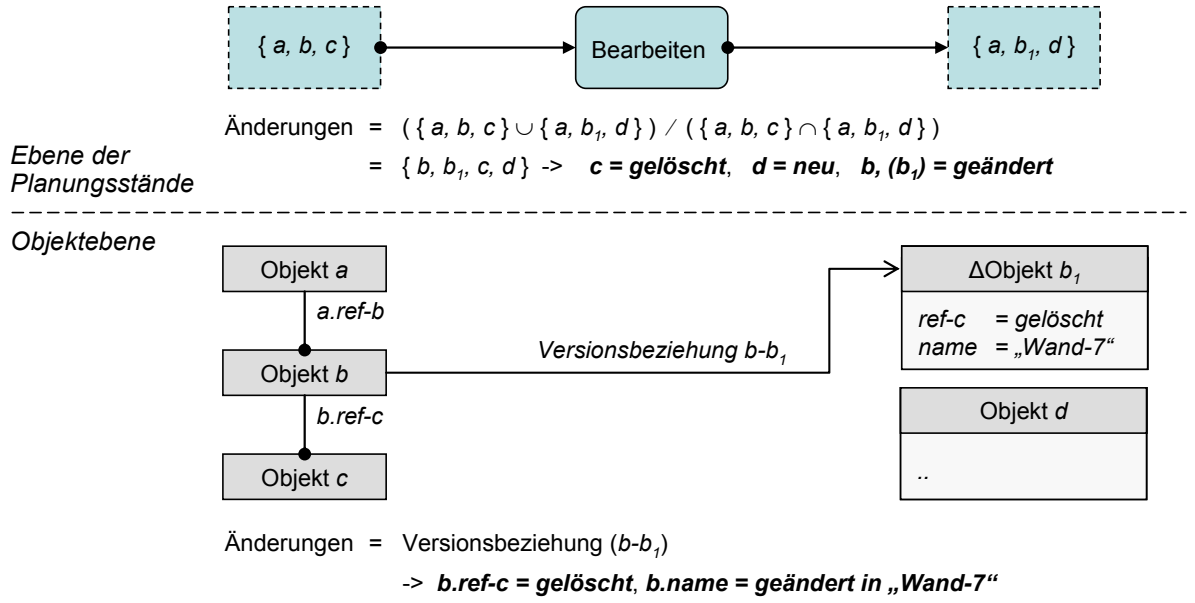


Abbildung 3-4 Übertragen der Änderungen in das Versionsmodell (siehe Abbildung 3-3)

Eine ausführliche Beschreibung des verwendeten deltabasierten Versionsmodells ist in Kapitel 4 zu finden, das auch auf die Vereinfachung der Versionsbeziehung und den damit verbundenen Verzicht auf die pfadabhängige Verwaltung der Deltas eingeht.

3.2.3 Methoden zur Aufbereitung der Planungsdaten

In der Diskussion wurde bisher vorausgesetzt, dass die durchgeführten Änderungen bekannt sind. Für die Anwendung des Ansatzes ist jedoch davon auszugehen, dass anstatt der benötigten Änderungen ein neuer Planungsstand vorliegt. Ein solcher Planungsstand ist potenziell durch Informationsverluste gekennzeichnet, die sich unter anderem durch das Fehlen bzw. das Verändern von Objekt-Identifikatoren äußern. Für das Ermitteln der Änderungen entsteht eine eigenständige Problematik, die in der vorliegenden Arbeit durch die Kombination modellunabhängig einsetzbarer Methoden betrachtet wird. Hierfür werden

1. ein Verfahren zur Beschreibung der von einem Anwenderprogramm handhabbaren Teilmenge des Datenmodells und
2. ein auf der Datenstruktur basierender Algorithmus zum Auffinden der Änderungen eingesetzt.

Ein Planungsschritt wird somit auf die Nutzung eines individuell benötigten Teildatensatzes zurückgeführt, der nach Abschluss der Bearbeitung wieder in den Gesamtdatenbestand integriert werden muss. Durch die Verwendung von Teildatensätzen wird gleichzeitig die Forderung berücksichtigt, das auszutauschende Datenvolumen zu verringern bzw. die Planungsdaten auf aufgabenrelevante Informationen zu beschränken.

Der beschriebene Ablauf ist in Abbildung 3-5 gezeigt und ist durch die Auswahl der benötigten Planungsdaten, die unabhängige Bearbeitung in einem Anwenderprogramm sowie die Re-Integration in den Gesamtdatenbestand gekennzeichnet. Diese drei Teilschritte werden durch den betroffenen Planer n_A gesteuert, stehen jedoch nicht vollständig unter Kontrolle der Datenbasis. Die Ergebnisse der Teilschritte können dafür aber im Versionsmodell abgebildet werden und stehen für somit weitere Auswertungen zur Verfügung.

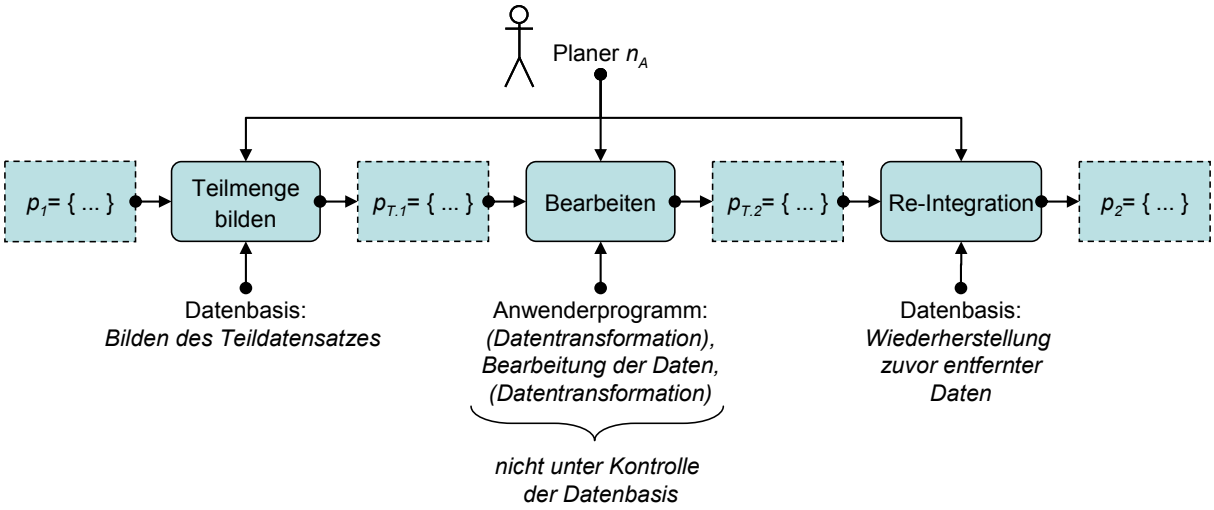


Abbildung 3-5 Abbilden eines Planungsschrittes durch das Zerlegen in drei Teilschritte

Beschreiben von Teildatensätzen

Für die Beschreibung der Teildatensätze wird die Auswahl der benötigten Planungsdaten mit der Definition der vom Anwenderprogramm verarbeitbaren Planungsinformationen kombiniert. Hierfür wird das im Rahmen der Arbeit entwickelte *Generalized-Model-Subset-Definition-Schema* verwendet (GMSD, Weise et al. 2003b). Im Mittelpunkt stehen 1.) eine einfache, mit möglichst wenigen Angaben durchführbare Auswahl der benötigten Planungsdaten und 2.) die Wiederverwendung für eine Bewertung geänderter Teildatensätze. Der Aspekt der Wiederverwendung ist für die Re-Integration geänderter Planungsdaten wichtig, um überarbeitete Planungsdaten filtern bzw. bereinigen zu können.

Das Problem der Teildatensatzbeschreibung wird in Kapitel 5.1 ausführlich vorgestellt. Das GMSD-Schema wird mengentheoretisch formalisiert und auf das in Kapitel 4 beschriebene Versionsmodell angewendet. Auf das Bereinigen überarbeiteter Planungsdaten wird schließlich in Kapitel 7 näher eingegangen.

Vergleich von Planungsständen

Für das Ermitteln der Änderungen wird ein generisches Vergleichsverfahren verwendet, das auf Basis der Datenstruktur arbeitet und das Informationspotenzial des zugrunde liegenden Meta-Modells nutzt. Ein sicheres Erkennen der durchgeführten Änderungen ist aufgrund fehlender oder nur teilweise vorhandener Schlüsselattribute, die für das eindeutige Identifizieren der Objekte und somit das Erzeugen der Versionsbeziehung erforderlich sind, jedoch nicht möglich. Der vorgeschlagene Vergleichsalgorithmus ist daher darauf ausgelegt, alle *Differenzen* zwischen den betrachteten Planungsständen unter der Maßgabe eines vertretbaren Berechnungsaufwands zu erkennen. Der Planungsstand wird dadurch nicht verfälscht. Die ermittelten Differenzen entsprechen aber nicht zwingend den tatsächlich durchgeführten Änderungen. Dies ist die unvermeidbare Konsequenz, die mit dem Versuch verbunden ist, die durchgeführten Änderungen nachträglich zu bestimmen.

Die Problematik des Modellvergleichs wird gemeinsam mit dem hierfür entwickelten Vergleichsverfahren in Kapitel 5.2 vorgestellt. Hierbei werden auch die Auswirkungen auf den Schritt der Re-Integration betrachtet, der mit dem Problem der Datenkonsistenz konfrontiert wird und bei dem Unzulänglichkeiten eines Planungsschrittes bemerkbar werden.

3.3 Anwendung des Lösungsansatzes

Der vorgestellte Versionsansatz erlaubt die Dokumentation der Planungsschritte, die als Grundlage für die Umsetzung einer gemeinsamen Datenbasis angesehen wird. Für die Anwendung des Versionsansatzes sind folgende Problemstellungen relevant:

1. die Nutzung der Versionshistorie,
2. die Unterstützung der parallelen Bearbeitung sowie
3. die Abbildung von Veränderungen im Objektgefüge.

Mit der Betrachtung dieser Problemstellungen wird nachfolgend ein Überblick über die Möglichkeiten und die vorgesehene Anwendung des Ansatzes gegeben.

3.3.1 Nutzung der Versionshistorie

Mit der Versionsverwaltung wird der Zugriff auf zurückliegende Planungsstände ermöglicht, der z. B. für das Nachvollziehen von Entwurfsentscheidungen oder die Rückkehr zu älteren Entwurfslösungen genutzt werden kann. Damit steigt die Aussagekraft der Datenbasis, die gegenüber einem unversionierten System ein wesentlich flexibleres Arbeiten erlaubt. Neben diesen allgemeingültigen Anwendungsmöglichkeiten wird der Ausgleich von Informationsverlusten verfolgt, der formal bereits durch den Umgang mit Teildatensätzen, der das temporäre Entfernen nicht benötigter Informationen beinhaltet, auftritt. Diese Sichtweise erlaubt eine von der Ursache des Informationsverlustes losgelöste Betrachtung der Problematik und lenkt die Aufmerksamkeit auf den Ablauf eines Planungsschrittes.

Das Wiederherstellen verlorener Informationen beruht auf der Beschreibung der benötigten bzw. handhabbaren Teilmenge. Mit einer solchen Beschreibung wird aus dem Gesamtdatenbestand zunächst ein neuer, der gesuchten Teilmenge entsprechender Planungsstand abgeleitet. Hierfür ist es ausreichend, nur die Objekte als eine neue Objektversion zu definieren, die durch die Teilmengenbildung in ihren Attributen „verändert“ werden. Als das Ergebnis eines Planungsschrittes entsteht aus dieser Teilmenge ein neuer Planungsstand, der gemäß dem Versionsansatz über die vorgenommenen Änderungen beschrieben wird. Um die überarbeitete Teilmenge schließlich wieder in den Gesamtdatenbestand zu integrieren, werden die zuvor entfernten Informationen zu einem neuen Planungsstand vervollständigt. Konzeptionell wird dies über das Zurücknehmen der Änderungen erreicht, die zum Bilden der Teilmenge erforderlich waren.

Die Abbildung 3-6 verdeutlicht den Umgang mit Teilmengen an einem Objekt *b*, das für die Bearbeitung selektiert wurde. Durch diese Auswahl wird in der abgeleiteten Teilmenge die Referenz *ref-e*, die auf das nicht zur Auswahl gehörende Objekt *e* verweist, getrennt. Die hierfür erforderliche Änderung wird durch das Objekt $b_{T.1}$ und die darin als gelöscht markierte Referenz *b.ref-e* beschrieben. Der Planungsschritt, der in dem gezeigten Beispiel das Attribut *name* des Objektes $b_{T.1}$ ändert, erzeugt das geänderte Objekt $b_{T.2}$. Durch das “Zurücknehmen” der in $b_{T.1}$ beschriebenen Änderung wird schließlich der Gesamtdatenbestand aktualisiert, wobei das Objekt b_2 entsteht. Die hierbei entstandenen Planungsstände sind über die Mengen $p_{T.1}$, $p_{T.2}$ und p_2 beschrieben, die die drei manipulierend eingreifenden Schritte dokumentieren. Hierzu gehören die Teilmengenbildung, die bewusst durchgeführten Änderungen der Bearbeitung sowie die Re-Integration der geänderten Teilmenge in den Gesamtdatenbestand. Auch wenn das Zusammenfassen der Teilschritte unter bestimmten Voraussetzungen möglich und unter dem Aspekt der Speicherplatzeinsparung sinnvoll ist, so bietet die vorgeschlagene Zer-

legung für die nachträgliche Bewertung eines Planungsschrittes einen wertvollen Informationsgewinn.

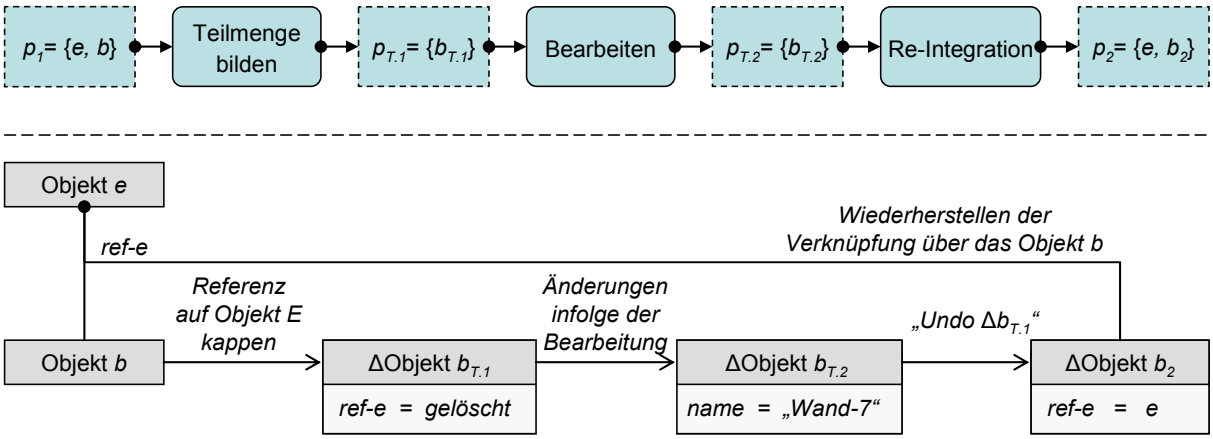


Abbildung 3-6 Wiederherstellen von Informationen durch das „Zurücknehmen“ der Änderungen, die zum Bilden der Teilmenge notwendig waren.

Ergänzend zu den Problemen der Teildatensatzbeschreibung und des Modellvergleichs ist in Kapitel 5.3 der Schritt der Re-Integration formal beschrieben. Hierbei werden auch die Grenzen des vorgeschlagenen generischen Verfahrens diskutiert, die durch weiterführende Ansätze und zusätzliche Nutzerinteraktionen zu überwinden sind.

3.3.2 Paralleles Arbeiten an gleichen Daten

Mit Hilfe der Versionsverwaltung ist ein Arbeiten ohne Sperren von Daten möglich. Dadurch kann eine optimistische Kooperationsstrategie verfolgt werden, die durch paralleles Arbeiten “alternative” Entwurfslösungen erzeugt. Die entstandenen Planungsstände müssen anschließend wieder zusammengeführt werden. Bei diesem Schritt werden im Idealfall die aufgetretenen Konflikte einvernehmlich aufgelöst und aus den *n* divergierenden Planungsständen *ein* gemeinsamer, konsistenter Planungsstand abgeleitet. Folglich werden die als Varianten vorliegenden Objekte wieder zusammengeführt. Hierbei werden neue Objekt(-versionen) erzeugt, die die notwendigen Änderungen gegenüber den zusammengeführten Varianten beschreiben.

Beim Zusammenführen divergierender Planungsstände sind mehrere Planer beteiligt, die koordiniert und durch entsprechende Werkzeuge unterstützt werden müssen. Hierfür existieren verschiedene Ansätze, die einen möglichst effizienten und transparenten Abstimmungsprozess anstreben. Aus Sicht der Datenverwaltung lässt sich die resultierende Problemstellung an dem Beispiel zweier Planer verdeutlichen, die ihre in der Abbildung 3-7 gezeigten Planungsstände zusammenführen sollen. Betrachtet man den Planer n_A, der die Änderungen des Planers n_B bewerten soll, so ergeben sich für jede Änderung folgende Möglichkeiten:

a) das *Übernehmen der Änderung*,

b) das *Ablehnen der Änderung* zugunsten der eigenen Lösung,

c) der *Vorschlag einer neuen Lösung*, die die Änderung ersetzen soll oder

d) der *Verzicht auf eine Entscheidung* zugunsten oder gegen die Änderung.

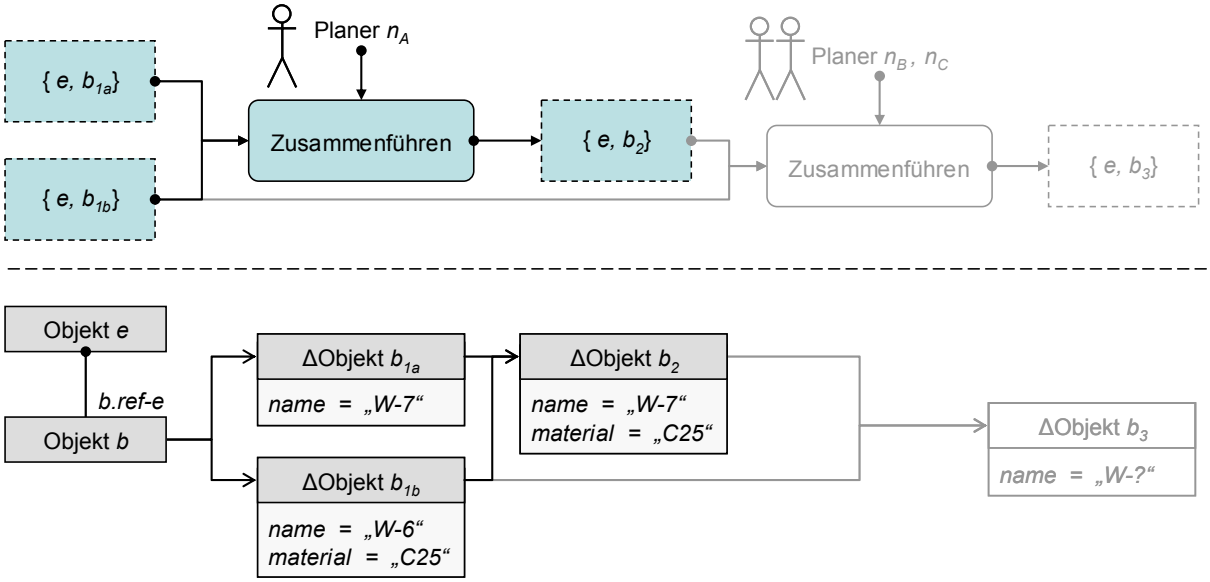


Abbildung 3-7 Zusammenführen divergierender Planungsstände durch Auflösen aller Divergenzen

Wenn der Planer n_A ausschließlich von den Möglichkeiten a), b) und c) Gebrauch macht, so erzeugt er einen neuen Planungsstand, der beide Planungsstände vereint. Sofern dieser Planungsstand nicht bereits einvernehmlich entstanden ist, so muss dieser Vorschlag auch von anderen Planern bewertet werden. Wie in Abbildung 3-7 gezeigt, können dabei weitere Planungsstände entstehen, die eine Alternative zu einem vorangegangenen Vorschlag darstellen.

Können nicht alle Divergenzen „aufgelöst“ werden, so entstehen zwei neue Planungsstände, die sich nur in den verbleibenden Divergenzen unterscheiden. Die Abbildung 3-8 zeigt das Ergebnis eines solchen Schrittes. Zwei alternative Objekte werden hierbei zunächst in einem Teil der Attribute vereint und anschließend, aufgrund der verbleibenden Divergenzen, wieder getrennt.

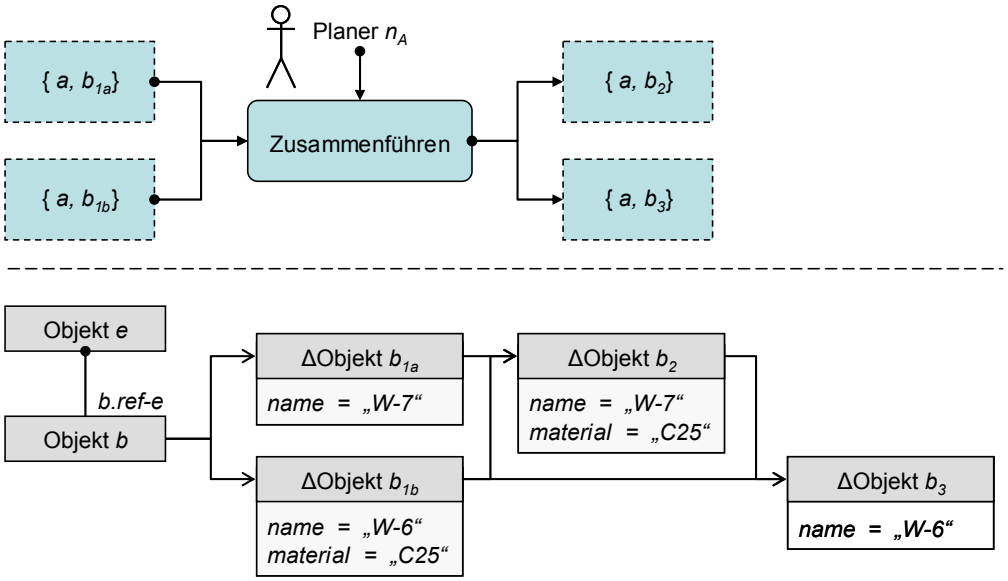


Abbildung 3-8 Zusammenführen divergierender Planungsstände, das durch verbleibende Divergenzen zu zwei Alternativen führt

Auf das Zusammenführen parallel entstandener Planungsstände wird in Kapitel 5.4 näher eingegangen. Im Mittelpunkt steht hierbei ein automatisiertes Verfahren, das die Änderungen im Sinne des Versionsmodells konfliktfrei zu einem neuen Planungsstand zusammenführt.

3.3.3 Veränderungen im Objektgefüge

Der vorgeschlagene Versionsansatz setzt auf einer objekt-orientierten Datenstruktur auf, die die Basis für die Beschreibung eines Planungsstandes darstellt. Durch den Umgang mit "Objekten" wird für den Anwender ein leicht nachvollziehbarer Bezug zu den Entitäten des abgebildeten Fachmodells erreicht. Auf diese Weise wird der Umgang mit den Planungsdaten vereinfacht. Der vorgeschlagene Versionsansatz verfolgt mit der Verwaltung von Änderungen ein ähnliches Ziel, indem logisch voneinander abhängige Objekte über eine Vorgänger-Nachfolger-Beziehung miteinander verknüpft werden. Hierbei wird die strenge Auslegung des Objektkonzeptes vermieden, wobei ein Objekt auf *einen* logischen Nachfolger sowie den gleichen Objekttyp beschränkt wird. Eine solche Forderung stellt für die vorgesehene Anwendung eine nicht gewollte Einschränkung dar, die mit einer geringeren Aussagekraft der Planungsdokumentation verbunden ist.

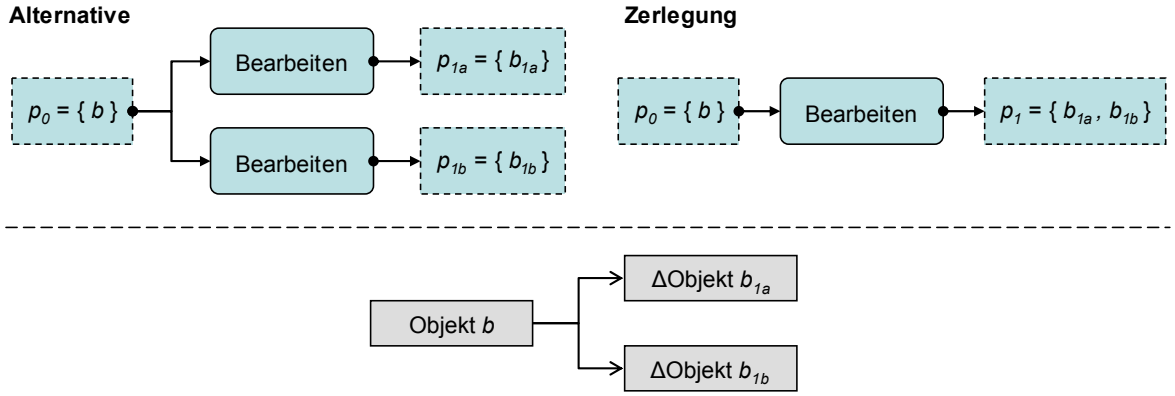


Abbildung 3-9 Zerlegung eines Objektes

Konzeptionell lässt sich sowohl das Teilen als auch das Zusammenlegen über die Kardinalität der Versionsbeziehung ausdrücken, wobei grundsätzlich eine *m:n*-Verknüpfung möglich ist. Über die Versionsbeziehung werden aber auch alternative Objekte beschrieben, die aus paralleler Bearbeitung entstehen und somit nicht das Verfeinern eines Objektes sondern eine Verzweigung des Versionspfades beschreiben. Das Erkennen alternativer Objekte, d. h. die Unterscheidung von einem geteilten Objekt, ist somit nicht direkt aus der Versionsbeziehung möglich, sondern erfordert die Auswertung der Planungsstände und der ihnen zugeordneten Objektmengen.

In der Abbildung 3-9 ist ein Objekt b dargestellt, das über eine *1:2*-Versionsbeziehung die Nachfolger b_{1a} und b_{1b} definiert. Gleichzeitig sind über die Mengen p_0 und p_1 bzw. p_{1a} und p_{1b} mögliche Planungsstände beschrieben, die eine Unterscheidung zwischen der Teilung des Objektes b und der Verzweigung in Alternativen erlauben. Für das Erkennen eines geteilten Objektes ist formal zu prüfen, ob ein Objekt der Menge p_0 mehrere Objekte in der Menge p_1 als Nachfolger besitzt.

Da die Zugehörigkeit eines Objektes zu einer Klasse, also der Objekttyp, unabhängig von seiner Versionsbeziehung vergeben werden kann, ist über die Versionsverwaltung auch der Wechsel des Objekttyps möglich. In der Abbildung 3-10 ist für das Objekt b_1 ein solcher Wechsel dargestellt, der zwangsläufig die nutzbaren Attribute des Vorgängerobjektes b einschränkt. Ein Objekt kann von seinem Vorgänger nämlich nur die Attribute übernehmen, die in den beiden betrachteten Objektklassen identisch definiert sind. Das Stützenobjekt b_1 kann

von dem Wandobjekt *b* demzufolge nur die *Lage* und *Geometrie* übernehmen. Zusätzlich ist ein Wert für das in der Objektklasse *Stütze* definierte Attribut *Bewehrungsgrad* erforderlich.

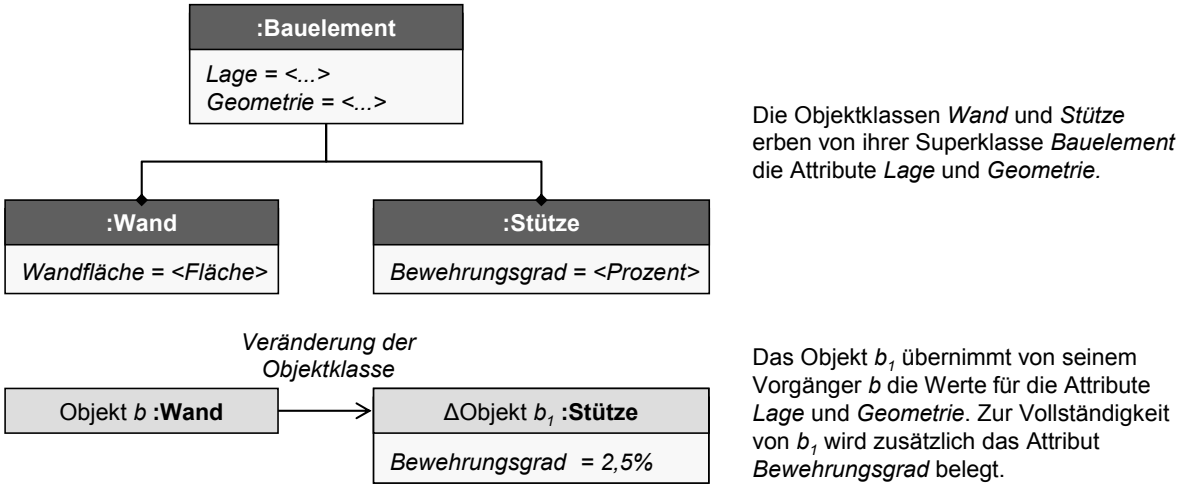


Abbildung 3-10 Änderung des Objekttyps

Aus diesen Beispielen wird der Charakter des Versionsansatzes erkennbar, der das Erfassen der Änderungen in den Vordergrund stellt und dafür das Prinzip der eindeutigen Objektidentifizierung aufgibt. Im Unterschied zu anderen Versionsansätzen wird auch auf die dynamische Konfiguration von Objektversionen verzichtet. Für die Dokumentation der Planungsschritte entsteht dadurch kein Nachteil, weil die zulässigen Objektkonfigurationen, also die entstandenen Planungsstände, ohnehin explizit beschrieben werden. Die gewonnene Flexibilität wird schließlich von dem vorgeschlagenen Vergleichsverfahren verwendet, um die ermittelten Änderungen abbilden zu können.

4 Formalisierung des Versionsmodells

In diesem Kapitel wird das Versionsmodell auf Basis der Mengenlehre formalisiert. Das Konzept der deltabasierten Speicherung wird dadurch konkretisiert. Zum besseren Verständnis der Formalisierung sei auch auf die Beschreibung der verwendete Notation im Anhang der Arbeit verwiesen.

Der *erste Abschnitt* definiert die verwendeten Elemente und Relationen. Damit können nicht nur die Planungsdaten sondern auch die Definition der Datenmodelle erfasst werden. Hierfür wird eine Basismenge definiert, die mit zwei Mengensystemen geordnet wird. In der Basismenge sind *Attribute* enthalten, die einerseits zu *Objekten* und *Planungsständen* und andererseits zu *Merkmalen* und *Objektklassen* zusammengefasst werden. Ein *Attribut* repräsentiert dabei die kleinste unterscheidbare Einheit, die im Sinne des objekt-orientierten Ansatzes mit der Belegung einer Attributdefinition gleichgesetzt werden kann. Außerdem werden drei Relationen definiert, die das Wiederherstellen der Planungsstände und das Erkennen der Änderungen ermöglichen.

Im *zweiten Abschnitt* werden die Konsistenzbedingungen für das Versionsmodell formuliert. Es werden Regeln definiert, die zwischen den Elementen des Versionsmodells zur widerspruchsfreien Verwaltung der Planungsstände einzuhalten sind. Mit diesen Regeln wird festgelegt, welche Attribute in einem Objekt als Delta enthalten sein müssen, welche Objekte zu einem Planungsstand zusammengefasst werden können und welche Objektverknüpfungen beim Anlegen eines Planungsstandes zu aktualisieren sind. Auf diese Weise wird beschrieben, wie die vorgenommenen Änderungen in dem Versionsmodell abzulegen sind.

Abschließend werden im *dritten Abschnitt* grundlegende Operationen auf dem Versionsmodell vorgestellt und mit der Semantik der Änderungen in Verbindung gebracht. Auf der Grundlage dieser Operationen werden der Umgang mit dem Versionsmodell und die auswertbaren Informationen beschrieben. Hierzu gehören das Zusammenführen der Deltas zu einem vollständigen Objekt, das Verfolgen der Objektentstehung, das Ermitteln von Unterschieden und Änderungen, die Umsetzung der Konsistenzbedingungen beim Anlegen eines neuen Planungsstandes und schließlich die Möglichkeit, „alte“ Objekte in neue Planungsstände einzubeziehen.

4.1 Elemente und Relationen des Versionsmodells

Für die Formalisierung des Versionsmodells wird ein Gebilde definiert, das aus einem Mengensystem besteht. Auf den entstehenden Unterstrukturen, die eine Zuordnung von Elementen sowie die Unterscheidung von Elementtypen ermöglicht, werden die zusätzlich benötigten Relationen des Gebildes beschrieben.

4.1.1 Elemente des Versionsmodells

Mit der Definition von Elementmengen wird die Strukturierung der zu verwaltenden Informationen verfolgt. Es werden Elementmengen verwendet, die einen engen Bezug zum objektorientierten Konzept besitzen und durch vergleichbare Begriffe gekennzeichnet sind. In der mengentheoretischen Aufbereitung liegt der Schwerpunkt auf der Strukturierung von Objektversionen, die eine gewisse Abstraktion von den Konzepten des zugrundeliegenden objektorientierten Paradigmas vornimmt.

Menge der Attribute A: Ein Attribut ist die kleinste Informationseinheit eines Objektes und beschreibt den Zustand für eine im Objekt gültige Attributdefinition. Die möglichen Attributdefinitionen werden durch das verwendete Meta-Modell bestimmt und können sowohl aus einfachen als auch aus komplexen, zusammengesetzten Datentypen bestehen. Das Attribut, als eine in diesem Ansatz nicht weiter unterteilte Einheit, stellt somit die kleinstmöglich verwaltbare Differenz einer neuen Objektversion dar. Im Sinne des objekt-orientierten Paradigmas kann ein Attribut auch als eine Kombination aus Attributdefinition und Attributwert angesehen werden.

(4.1)

$$A := \{ x \mid x \text{ ist ein Attribut} \}$$

In der Menge der Attribute werden zusätzlich die beiden Teilmengen A_{NB} und A_{ID} definiert. Die Teilmenge A_{NB} enthält dabei genau die Elemente, die den Zustand *nicht belegt* für die verfügbaren Attributdefinitionen definieren. Es gilt:

(4.2)

$$A_{NB} := \{ x \in A \mid x \text{ beschreibt für eine Attributdefinition den Zustand } \textit{nicht belegt} \}$$
$$\text{mit } A_{NB} \subset A$$

In der Teilmenge A_{ID} sind dagegen genau die Elemente enthalten, die innerhalb des Versionsmodells zur Identifizierung der Objekte verwendet werden. Für die Teilmenge A_{ID} gilt:

(4.3)

$$A_{ID} := \{ x \in A \mid x \text{ ist ein Identifikator} \}$$
$$\text{mit } A_{ID} \subset A \quad \text{und} \quad A_{ID} \cap A_{NB} = \varnothing$$

Aus den nachfolgend gezeigten Beispielen wird deutlich, dass ein Attribut eine Vielzahl von Informationen enthält, die in ihrer Gesamtheit den Attributzustand beschreiben und als gegenseitige Unterscheidungsmerkmale genutzt werden. Wie in der Aussage (4.4) dargestellt, beschreiben zwei Elemente aus der Menge A immer einen voneinander verschiedenen Zustand. Im Rahmen des Versionsmodells wird das Unterscheidungsmerkmal der Attribute nicht näher spezifiziert und stellt daher eine Abstraktion dar, die auf die Aussage (4.4) reduziert wird.

(4.4)

$$x, y \in A: \qquad x \neq y \Leftrightarrow \text{Zustand}(x) \neq \text{Zustand}(y)$$

Beispiel 1 : $x, y, z, w \in A$

Attributbelegung	Attributdefinition
$x = 2.5$	(Gleitkommazahl, Höhe eines Raumes)
$y =$'Büro'	(Zeichenkette, Name eines Raumes)
$z = -1.5, 0.5, 0.0$	(geordnete Menge von 3 Gleitkommazahlen, Lage eines Punktes)
$w = 2.5$	(Gleitkommazahl, Länge eines Balkens)

Attributzustand

$\text{Zustand}(x) \neq \text{Zustand}(w)$, da x und w die Attributbelegung für unterschiedliche Attributdefinitionen beschreiben.

Die Menge der Attribute wird nachfolgend über zwei Mengensysteme in Anlehnung an das objekt-orientierte Paradigma strukturiert. Als Unterscheidungskriterien werden einerseits die Beschreibung von Instanzen (Objekte und Planungsstände) und andererseits die Beschreibung von Typen (Attributdefinitionen und Objektklassen) verwendet.

Menge der Objekte O: Ein Objekt besitzt eine Menge von Attributen, die zur Beschreibung des Objektzustandes sowie zu seiner Identifizierung benötigt werden. Da ein deltabasierter Ansatz verfolgt wird, gehören zu einem Objekt nur die Attribute, die sich gegenüber seinem Vorgänger geändert haben. Jedes Element der Menge O repräsentiert somit ein *Deltaobjekt.* Es wird zusätzlich gefordert, dass jedes Objekt über ein Attribut aus der Menge A_{ID} von anderen Objekten unterscheidbar ist.

(4.5)

$$O := \{ x \subseteq A \mid x \text{ beschreibt ein Objekt} \}$$

$$\text{mit } \underbrace{\forall_{x \in O} \; \exists_{a \in x} \; a \in A_{ID}}_{1)} \wedge \underbrace{\forall_{b \in x, b \neq a} \; b \notin A_{ID}}_{2)} \wedge \underbrace{\forall_{y \in O, y \neq x} \; a \notin y}_{3)}$$

Die für ein Element aus der Menge O formulierte Bedingung drückt aus, dass jedes Objekt des Versionsmodells genau ein Attribut aus der Menge A_{ID} besitzt (Bedingung 1 und 2) und ein solches Attribut ausschließlich von einem Objekt verwendet wird (Bed. 3). Innerhalb des Versionsmodells wird damit die Unterscheidbarkeit der Objekte garantiert, die nicht im Widerspruch zu der in Kapitel 3.1.1 formulierten Annahme bezüglich der Identifizierbarkeit steht. Dies wird auch daraus deutlich, dass jedes Objekt, auch wenn es eine neue Version eines vorhandenen Objektes darstellt, eine eigene „Identität“ besitzt.

Menge der Planungsstände P: Ein Planungsstand beschreibt eine Entwurfsversion und setzt sich aus den hierfür gültigen Objekten zusammen.

(4.6)

$$P := \{ x \subseteq O \mid x \text{ beschreibt einen gültigen Planungsstand} \}$$

Menge der Merkmale M: Ein Merkmal beschreibt eine Attributdefinition und ist damit die kleinste Informationseinheit einer Objektklasse. Für jedes im objekt-orientierten Paradigma unterscheidbare Merkmal existiert in der Menge M ein ihr zugeordnetes Element, das wiederum alle Attribute umfasst, die einen Zustand für die zugehörige Attributdefinition beschreiben.

(4.7)

$$M := \{ x \subseteq A \mid x \text{ definiert ein Merkmal einer Objektklasse} \}$$

$$\text{mit } \forall_{a \in A} \; \exists_{x \in M} \; a \in x \wedge \forall_{y \in M, y \neq x} \; a \notin y$$

In der Menge der Merkmale wird zusätzlich die Teilmenge M_{ID} definiert, die genau die Elemente enthält, die zur Unterscheidung der Objektklassen dienen. Es gilt:

(4.8)

$$M_{ID} := \{ x \in M \mid x \text{ ist ein Merkmal zur Identifizierung} \}$$

$$\text{mit } M_{ID} \subset M \qquad \text{wobei gilt} \qquad \forall_{a \in A, \, a \notin A_{ID}, \, m \in M_{ID}} \; a \notin m$$

Beispiel 2 : $x, y, z \in M$

x = (Gleitkommazahl, Höhe eines Raumes)

y = (ID, Raum)

z = (Gleitkommazahl, Länge eines Balkens)

Aus den beiden Beispielen 1 und 2 wird deutlich, dass ein Merkmal über seine Informationen die Eigenschaften für eine bestimmte Menge von Attributen beschreibt. Besitzt ein Attribut die Eigenschaften eines Merkmals, so ist dieses Attribut ein Element des Merkmals.

Menge der Objektklassen K: Eine Objektklasse definiert die Merkmale für eine Menge von Objekten. Besitzt ein Objekt alle Merkmale einer Klasse, so ist dieses Objekt eine Instanz dieser Klasse. Es wird zusätzlich gefordert, dass jede Objektklasse über ein Merkmal aus der Menge M_{ID} unterscheidbar ist.

(4.9)

$$K := \{ x \subset M \mid x \text{ definiert die Merkmale für eine Klasse von Objekten} \}$$

$$\text{mit } \underbrace{\forall_{x \in K} \; \exists_{a \in x} \; a \in M_{ID}}_{1)} \wedge \underbrace{\forall_{b \in x, b \neq a} \; b \notin M_{ID}}_{2)} \wedge \underbrace{\forall_{y \in K, y \neq x} \; a \notin y}_{3)}$$

Analog zur Beschreibung der Objektmenge wird auch hier eine Bedingung zur Unterscheidbarkeit der Objektklassen formuliert. Jede Objektklasse besitzt genau ein Merkmal aus der Menge M_{ID} (Bed. 1 und 2), wobei ein solches Merkmal ausschließlich von einer Objektklasse verwendet wird (Bed. 3). Ergänzend zu dem Begriff der Objektklasse sind Anmerkungen zur Vererbung sowie der Re-Definition von Merkmalen notwendig. In dem dargestellten Versionsmodell wird zur Vereinfachung auf die Abbildung der Vererbung verzichtet. Folglich muss jede Objektklasse alle ihr zugeordneten Merkmale enthalten, wobei ein Merkmal zu mehreren Objektklassen gehören kann.

Das Gebilde, das durch die eingeführten Mengen entsteht, ist in Abbildung 4-1 beispielhaft dargestellt. Die Basismenge der Attribute wird auf diese Weise geordnet und bildet einerseits die Planungsdaten und andererseits die Definition der Datenmodelle ab.

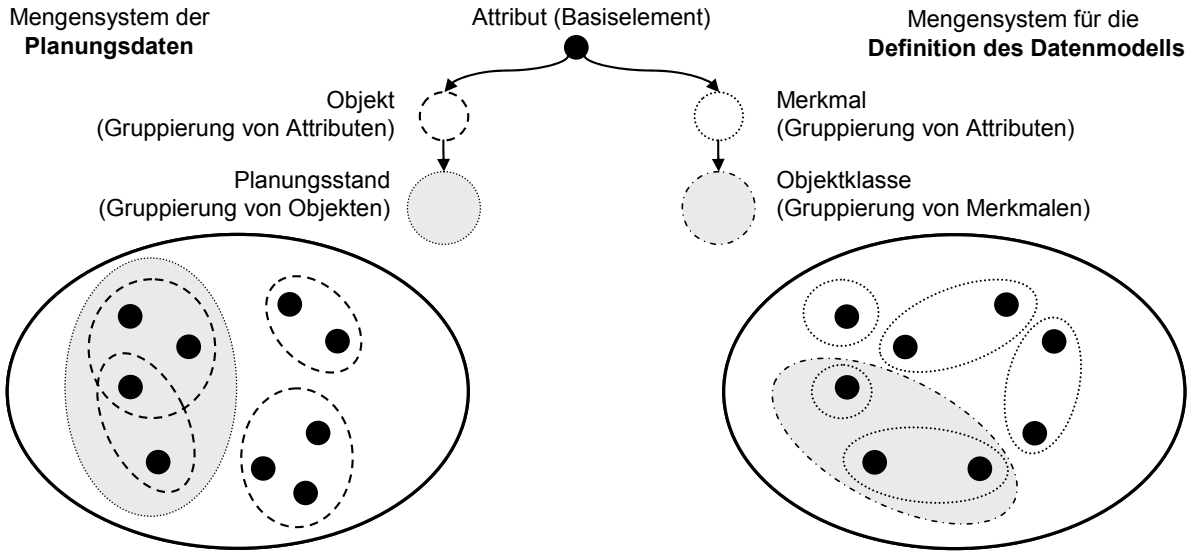


Abbildung 4-1 Ordnung der Basismenge A durch zwei Mengensysteme

4.1.2 Relationen

Auf den oben definierten Elementmengen werden Relationen definiert, um das Versionsmodell weiter zu strukturieren. Jede Relation stellt eine Menge geordneter Paare dar und lässt sich dadurch selbst mengentheoretisch in der Formulierung der Konsistenzbedingungen verwenden.

Versionsrelation R_{VN}: Eine Versionsrelation definiert eine Vorgänger-Nachfolger-Beziehung zwischen zwei Objekten und stellt somit die Verbindung zwischen zwei Objektversionen her. Über die Versionsbeziehung können die Vorgänger eines Objektes ermittelt werden, die zur Wiederherstellung eines vollständigen Objektstandes benötigt werden. Die Versionsrelation ist antireflexiv und asymmetrisch. Zusätzlich wird gefordert, dass der durch R_{VN} erzeugte Graph azyklisch ist.

(4.10)

$$R_{VN} := \{(x,y) \in O \times O \mid R_{VN}(x,y) \text{ ist eine Vorgänger - Nachfolger - Beziehung}\}$$
$$\text{mit } \bigvee_{x \in O} (\neg R_{VN}(x,x)) \quad \text{und} \quad \bigvee_{x,y \in O} (R_{VN}(x,y) \Rightarrow \neg R_{VN}(y,x))$$

Für die Beschreibung der Zyklenfreiheit werden die Begriffe der Verbindung sowie der Transitiven Hülle verwendet. Die Definition der beiden Begriffe lässt sich nach Pahl & Damrath (2002) wie folgt auf die Versionsrelation übertragen:

(4.11)

Verbindung :

$$V_{R_{VN}} := \left\{ (x_1, x_2, ..., x_n) \mid i \in \{1, ..., n-1\} \wedge \bigvee_i ((x_i, x_{i+1}) \in R_{VN}) \right\}$$
$$V_{R_{VN}}(a,b) :\Leftrightarrow \bigvee_{V_{R_{VN}}} (x_1 = a \wedge x_n = b)$$

(4.12)

Transitive Hülle :

$$\langle R_{VN} \rangle_t := \{ (a,b) \in O^2 \mid V_{R_{VN}}(a,b) \}$$

Die Zyklenfreiheit des durch R_{VN} aufgespannten Graphen ergibt sich aus der Forderung, dass die Bedingung der Asymmetrie ebenso für die Transitive Hülle von R_{VN} erfüllt sein soll.

(4.13)

$$\bigvee_{x,y \in O} (\langle R_{VN} \rangle_t (x,y) \Rightarrow \neg \langle R_{VN} \rangle_t (y,x))$$

Die Abbildung 4-2 zeigt mit der bereits in Kapitel 3 verwendeten graphischen Darstellung die Anwendung der Versionsrelation sowie die resultierende Verbindung, die sich aus der Transitiven Hülle der Versionsrelation ableiten lässt.

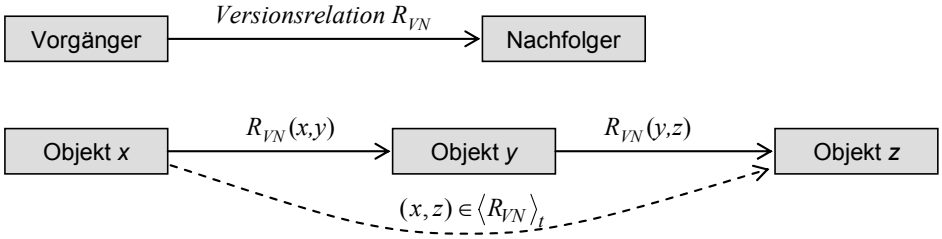


Abbildung 4-2 Graphische Darstellung der Versionsrelation und der resultierenden Verbindung

Referenzrelation R_{AO}: Eine Referenzrelation wird für die Verknüpfung von Objekten verwendet. Da der Zustand eines Objektes dadurch beeinflusst wird, erfolgt die Definition der Referenzrelation über ein Attribut des Objektes.

$$R_{AO} := \{(x,y) \in A \times O \mid R_{AO}(x,y) \text{ beschreibt das über ein Attribut referenzierte Objekt}\} \tag{4.14}$$

In Ergänzung zur Definition der Attribute soll erwähnt werden, dass der Zustand eines Attributes auch über seine Referenzrelationen bestimmt wird. Wird ein Attribut zur Beschreibung von Objektverknüpfungen verwendet, so drückt sich dies in seiner Attributbelegung aus. Neben der Wahrung der Unterscheidbarkeit ist eine solche Attributbelegung auch deshalb notwendig, um eine mögliche Ordnung oder das mehrfache Vorkommen einer Objektverknüpfung beschreiben zu können.

In der Abbildung 4-3 ist beispielhaft die Verknüpfung von zwei Objekten dargestellt, die über das Attribut *r* auf ein referenziertes Objekt verweist. Das Attribut *r* kann jedoch von mehreren Objekten verwendet werden und kann demzufolge für eine beliebige Anzahl von Objekten eine Verknüpfung zu dem referenzierten Objekt herstellen. Da die Referenz über den betrachteten Planungsstand zusätzlich aufgelöst werden muss, kann über das Attribut *r* auch ein Nachfolger des referenzierten Objektes als Verknüpfung ermittelt werden. Die Referenzrelation kann auf diese Weise eine Verknüpfung der Kardinalität m:n beschreiben, die für einen Planungsstand jedoch auf die Verknüpfung des Typs 1:1 beschränkt ist (siehe 4.2.3).

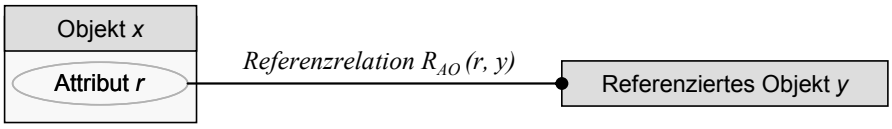


Abbildung 4-3 Graphische Darstellung der Referenzrelation

Instanzrelation R_{OK}: Über eine Instanzrelation wird die Klassenzugehörigkeit eines Objektes beschrieben. Ein Objekt gehört zu genau einer Klasse, über die es die zulässigen Merkmale bezieht (Abbildung 4-4).

$$R_{OK} := \{(x,y) \in O \times K \mid R_{OK}(x,y) \text{ beschreibt die Klassenzugehörigkeit eines Objektes}\} \tag{4.15}$$
$$\text{mit } \forall_{x \in O} \exists_{y \in K} R_{OK}(x,y) \wedge \forall_{z \in K, z \neq y} \neg R_{OK}(x,z)$$

Zu jedem Attribut *a* eines Objektes *o* ist in der über die Instanzrelation R_{OK} zugeordneten Objektklasse *k* folglich ein zugehöriges Merkmal *m* definiert. Gleichzeitig wird gefordert, dass ein Objekt *o* nicht mehrere Attribute eines Merkmals *m* enthält (4.16).

(4.16)

$$\forall_{o\in O}\ \forall_{a\in o}\ \exists_{k\in K}\ \exists_{m\in k}\ R_{OK}(o,k)\wedge m\in k\wedge a\in m\quad\wedge\quad\forall_{b\in o,\,b\neq a}\ b\notin m$$

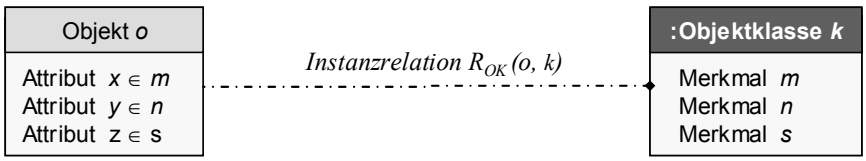


Abbildung 4-4 Graphische Darstellung der Instanzrelation

4.2 Konsistenzbedingungen des Versionsmodells

Für das vorgestellte Gebilde werden Verträglichkeitsregeln benötigt, um im Sinne des in Kapitel 3 vorgestellten Versionsansatzes die als Änderungen abgebildeten Planungsstände widerspruchsfrei verwalten zu können. Bei der Definition der Mengen und Relationen wurden bereits einfache Verträglichkeitsregeln formuliert, die in diesem Abschnitt vervollständigt werden. Dadurch soll die konsistente Abbildung der entstandenen Entwurfsversionen garantiert werden, die aber nicht die Konsistenz der Datenstruktur oder die inhaltliche Konsistenz der Planungsdaten betrachtet. Die Beurteilung der strukturellen und inhaltlichen Konsistenz bleibt anderen Diensten, Anwenderprogrammen und nicht zuletzt den Planern vorbehalten.

4.2.1 Ermitteln der gültigen Attribute eines Objektes

Ein Objekt speichert nur die Attribute, die sich gegenüber seinem Vorgänger verändert haben. Da ein Objekt mehrere Vorgänger haben kann und die Attribute nicht getrennt für jeden Vorgänger verwaltet werden, wird gefordert, dass sich keine Widersprüche bei der Ermittlung der Attribute ergeben. Demzufolge muss garantiert werden, dass unabhängig vom gewählten Vorgänger stets die gleichen gültigen Attribute ermittelt werden.

Über die in (4.17) beschriebene Mengendefinition $M_{ALT}(o)$ werden alle Merkmale eines Objektes *o* zusammengefasst, für die der gültige Zustand über die Vorgängerobjekte ermittelt werden muss.

(4.17)

geg.: $o, q \in O$

$$M(o) := \left\{ m\in M \mid \exists_{k\in K} R_{OK}(o,k)\wedge m\in k \right\}$$

$$M_{NEU}(o) := \left\{ m\in M(o) \mid \exists_{a\in A} a\in m\wedge a\in o \right\}$$

$$M_{ALT}(o) := M(o) - M_{NEU}(o)$$

Die Menge $A_{GÜLTIG}(o)$ enthält alle gültigen Attribute des Objektes *o* und ist rekursiv über die Vorgängerobjekte beschrieben. Die gültigen Attribute der Vorgängerobjekte werden in der Definition (4.18) über die Menge $A_{ALT}(o,q)$ beschrieben und stellen den Anteil dar, der von dem Vorgängerobjekt *q* aus erreichbar ist.

(4.18)

$$A_{GÜLTIG}(o) := \left\{ a \in A \mid a \in o \vee \underbrace{\left(\exists_{m \in M_{ALT}(o)} a \in m \wedge \exists_{q \in O} R_{VN}(q,o) \wedge a \in A_{GÜLTIG}(q) \right)}_{A_{ALT}(o,q) = \text{gültige Attribute der Vorgänger}} \right\}$$

Mit der Bedingung (4.19) wird schließlich eine grundlegende Forderung des deltabasierten Versionsmodells formuliert. Diese Bedingung hat Auswirkungen auf die Abbildung der Änderungen in Deltas und hat unter anderem zur Folge, dass nur ein Vorgänger für die Ermittlung der gültigen Attribute betrachtet werden muss.

(4.19)

$$\forall_{r,s \in O \wedge R_{VN}(r,o) \wedge R_{VN}(s,o)} A_{ALT}(o,r) = A_{ALT}(o,s)$$

Unter der Annahme, dass die in der Abbildung 4-5 gezeigten Objekte alle zur gleichen Objektklasse gehören, sind die gültigen Attribute der einzelnen Objekte aufgelistet. Es wird deutlich, dass nur die fehlenden Attribute eines Objektes über seine Vorgänger ermittelt werden müssen. Wenn für jedes Merkmal der zugehörigen Objektklasse ein gültiges Attribut gefunden wurde, so kann die Suche über die Vorgänger des Objektes abgebrochen werden. Aus dem Beispiel der Abbildung 4-5 wird für das Objekt *a4* auch die Forderung (4.19) deutlich. Für die beiden Merkmale *m* und *n* ist in dem Objekt *a4* ein Attribut definiert, da sonst über die beiden Vorgänger *a2* und *a3* unterschiedliche Objektzustände ermittelt werden. In dem gezeigten Beispiel wird für das Merkmal *m* das Attribut *x2* verwendet, wodurch der Zustand des Vorgängers *a2* übernommen wird. Das Merkmal *n* erhält mit dem Attribut *y4* dagegen eine Belegung, die in keinem der Vorgängerobjekte genutzt wurde. Egal welche Belegung für die Merkmale *m* und *n* gewählt wird, für das Erfüllen der Bedingung (4.19) ist wichtig, dass das Objekt *a4* diese Merkmale „neu“ definiert. Demgegenüber kann für das Merkmal *s* das Attribut *z1* aus dem Objekt *a1* übernommen werden, da in diesem Fall keine Abhängigkeit von dem verwendeten Vorgänger existiert.

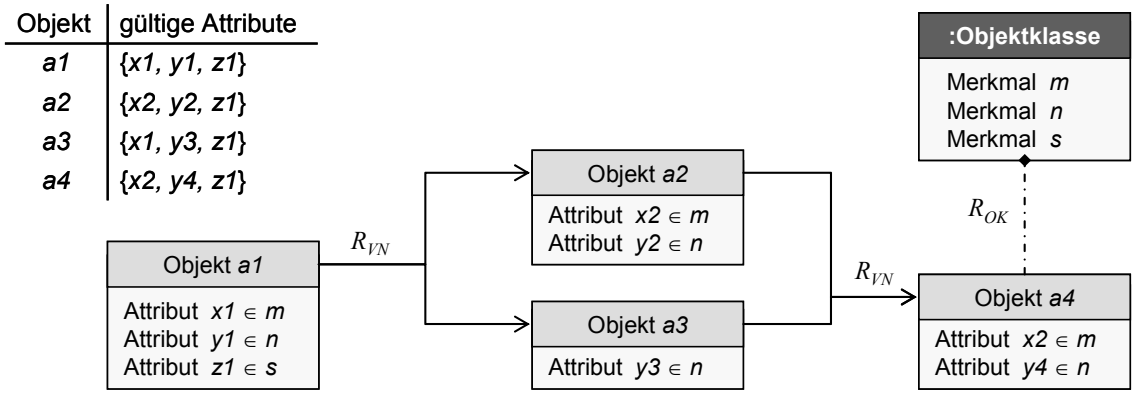


Abbildung 4-5 Ermittlung der gültigen Attribute über die Vorgängerobjekte

4.2.2 Unzulässige Objektkombinationen eines Planungsstandes

Jedes Element der Menge P enthält alle die Objekte, die für die Beschreibung des Planungsstandes notwendig sind. Jeder Planungsstand bildet dabei genau eine Entwurfsversion ab und darf folglich keine Objekte enthalten, die über eine Versionsrelation zueinander in Verbindung stehen. Über die Transitive Hülle der Versionsrelation (4.12) lässt sich diese Bedingung wie folgt definieren:

(4.20)

$$\forall_{p \in P} \; \forall_{x,y \in p} \; \neg \langle R_{VN} \rangle_t (x,y)$$

In der Abbildung 4-6 sind für die gezeigten Objekte und ihre Abhängigkeiten über die Versionsrelation mögliche und unzulässige Planungsstände aufgezeigt, die sich aus der Bedingung (4.20) ableiten lassen.

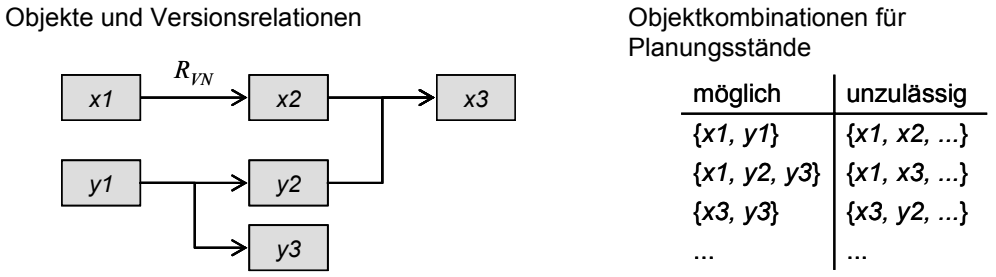


Abbildung 4-6 Unzulässige Objektkombinationen, die aus den Versionsrelationen ableitbar sind

4.2.3 Eindeutigkeit von Referenzen

Über ein Attribut eines Objektes lassen sich Assoziationen zu anderen Objekten definieren. Eine solche Verknüpfung verweist nicht zwangsläufig auf ein Objekt, das in dem betrachteten Planungsstand verwendet wird. Auf diese Weise ist es möglich, die Aktualisierung von Referenzen und somit das Anlegen neuer Objekte deutlich zu verringern. Wird auf ein Objekt verwiesen, das nicht in dem betrachteten Planungsstand enthalten ist und somit eine überarbeitete Objektversion darstellt, so muss über die Versionsrelation das aktuelle, referenzierte Objekt ermittelt werden. Hierfür wird gefordert, dass über die Versionsrelation eindeutig auf das referenzierte Objekt geschlossen werden kann. Sei ein Planungsstand *p* und ein über diesen Planungsstand referenziertes Objekt *o* gegeben, so gilt:

(4.21)

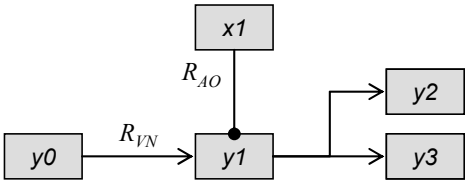
$$\text{geg.:} \quad p \in P,\ o \in O_{REF}(p)$$

$$\text{mit } O_{REF}(p) := \{ x \in O \mid x \text{ wird über ein Objekt des Planungsstandes } p \text{ referenziert} \}$$

$$\exists_{x \in p} \; (x = o \vee \langle R_{VN} \rangle_t (o,x)) \wedge \forall_{y \in p, x \neq y} \; \neg (y = o \vee \langle R_{VN} \rangle_t (o,y))$$

Analog zur Abbildung 4-6 zeigt die Abbildung 4-7 mögliche und unzulässige Planungsstände, die sich aus nicht auflösbaren Referenzen ergeben und sich aus der Bedingung (4.21) ableiten lassen.

Referenziertes Objekt *y1* und seine Einbettung in den Versionsgraphen



Objektkombinationen für Planungsstände

möglich	unzulässig
{x1, y1}	{x1, y2, y3}
{x1, y2}	{x1}
{x1, y3}	{x1, y0}
...	...

Abbildung 4-7 Unzulässige Objektkombinationen, die aus nicht auflösbaren Referenzen entstehen

4.3 Operationen auf dem Versionsmodell

In den vorangegangenen Abschnitten wurden das Versionsmodell und die einzuhaltenden Konsistenzbedingungen formuliert. Der folgende Abschnitt beschäftigt sich mit den Operationen auf dem Versionsmodell, die es erlauben, auf die im Versionsmodell abgelegten Informationen zuzugreifen sowie neue Planungsstände anzulegen.

4.3.1 Wiederherstellen eines Objektes

Soll auf ein Objekt mit all seinen gültigen Attributen zurückgegriffen werden, so muss aus den vorliegenden Änderungen das vollständige Objekt über seine Vorgänger wiederhergestellt werden. Über die Menge der gültigen Attribute (4.18) sowie das Auflösen von Referenzen (4.21) wurden bereits wichtige Grundlagen definiert, die auch für das Wiederherstellen eines Objektes nutzbar sind. Da ein Objekt zu mehreren Planungsständen gehören kann und das Auflösen der Referenzen ohne den Planungsstand nicht möglich ist, wird für das Wiederherstellen eines Objektes zusätzlich der entsprechende Planungsstand benötigt.

$$\text{geg.:}\quad o \in O,\ p \in P \qquad \text{wobei gefordert wird:}\quad o \in p$$

Aus den gültigen Attributen eines Objektes lässt sich die Menge der referenzierten Objekte über die Referenzrelationen bestimmen.

$$O_{REF}(o) := \left\{ x \in O \mid \exists_{a \in A_{GÜLTIG}(o)} R_{AO}(a,x) \right\} \tag{4.22}$$

Über den gegebenen Planungsstand p kann anschließend die Menge der tatsächlich verknüpften Objekte gebildet werden.

$$O_{REF\_AKTUELL}(o,p) := \left\{ x \in O \mid x \in p \wedge \exists_{y \in O_{REF}(o)} x = y \vee (y,x) \in \langle R_{VN} \rangle_t \right\} \tag{4.23}$$

Soll ein vollständiges Objekt mit allen gültigen und zugleich aktualisierten Attributen gebildet werden, so sind die Attribute zu ersetzen, die mindestens ein Objekt referenzieren, das nicht in der Menge der tatsächlich verknüpften Objekte enthalten ist. Sie enthalten alte Referenzen und bilden die Menge A_{Alt}. Diese Attribute müssen durch neue Attribute mit aktualisierten Referenzen ersetzt werden.

(4.24)

$$A_{ALT}(o,p) := \left\{ x \in A \mid x \in A_{GÜLTIG}(o) \wedge \underset{y \in O_{REF\_ALT}(o,p)}{\exists} R_{AO}(x,y) \right\}$$

mit $O_{REF\_ALT}(o,p) := \underbrace{O_{REF}(o) - O_{REF\_AKTUELL}(o,p)}_{\text{Über das Objekt o referenzierte Objekte, die nicht zum Planungsstand p gehören.}}$

Beim Ersetzen der Attribute sind die aktualisierten Referenzrelationen auf die neuen Attribute zu übertragen, wobei ausschließlich die Objekte referenziert werden sollen, die auch zu dem betrachteten Planungsstand gehören. Für ein Attribut a_{NEU}, das ein Attribut a_{ALT} aus der zu aktualisierenden Attributmenge ersetzt, müssen folglich die in (4.25) gezeigten Referenzrelationen vorhanden sein.

(4.25)

$$\text{geg.:} \quad a_{ALT} \in A_{ALT}(o,p), \quad a_{ALT} \Rightarrow a_{NEU} \in A$$

$$\underset{x \in O \wedge \in R_{AO}(a_{ALT},x)}{\forall} \; \underset{y \in O_{REF\_AKTUELL}(o,p)}{\exists} R_{AO}(a_{NEU},y) \wedge \left(x = y \vee \langle R_{VN} \rangle_t (x,y) \right)$$

Über das Attribut a_{NEU} dürfen keine weiteren Referenzrelationen definiert sein und es muss gewährleistet werden, dass über das neue Attribut genau die Informationen beschrieben sind, die auch über das zu ersetzende Attribut und den betrachteten Planungsstand abgeleitet werden können. Da die hierfür notwendigen zusätzlichen Angaben innerhalb des Attributes gekapselt sind und in dem vorgestellten Gebilde nicht beschrieben werden, kann für den Zustand der ersetzenden Attribute keine vollständige Bedingung formuliert werden.

In Ergänzung zu der in (4.21) formulierten Bedingung lässt sich die dort verwendete Menge $O_{REF}(p)$ mit der Menge aus der Definition (4.22) wie nachfolgend gezeigt definieren.

(4.26)

$$O_{REF}(p) := \left\{ x \in O \mid \underset{o \in p}{\exists} x \in O_{REF}(o) \right\}$$

In der Abbildung 4-8 ist für den Planungsstand *p* das Wiederherstellen des Objektes *o* gezeigt. In diesem Beispiel wird das Ermitteln der gültigen Attribute sowie das Ersetzen der Referenz *ref-c*, die für den betrachteten Planungsstand *p* nicht mehr aktuell ist, verdeutlicht.

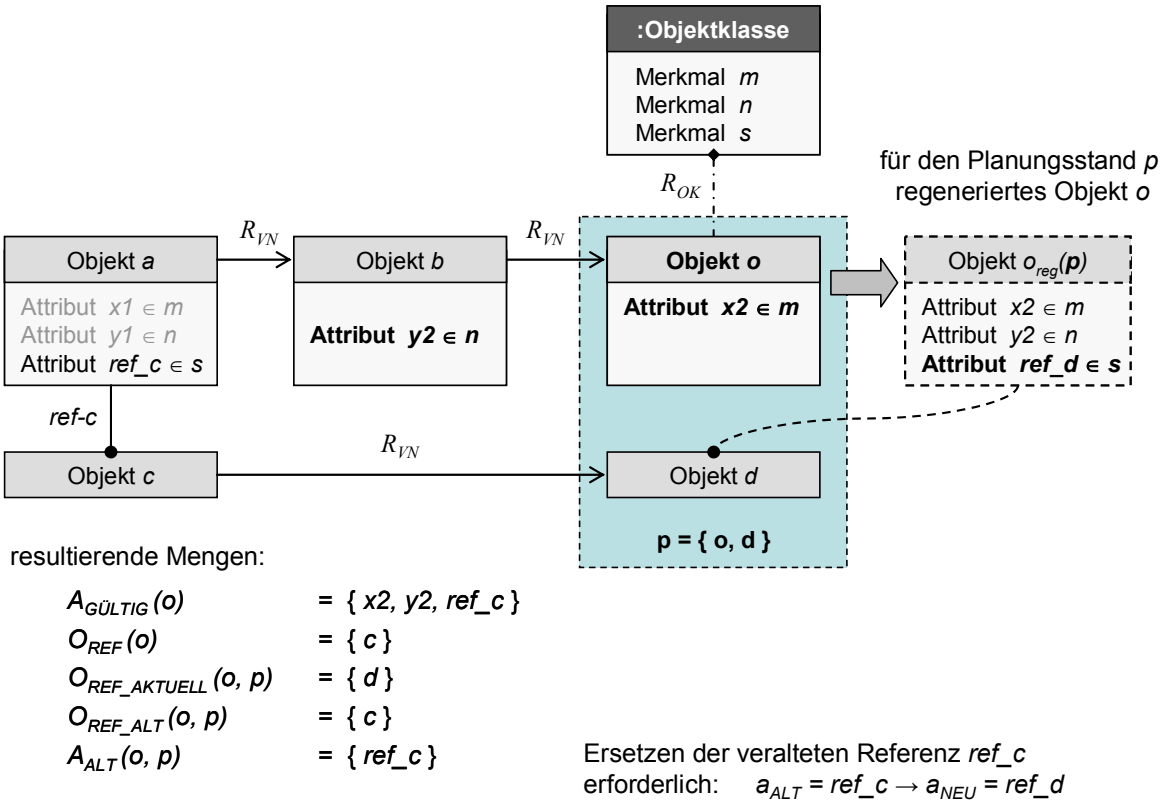


Abbildung 4-8 Wiederherstellen eines Objektes

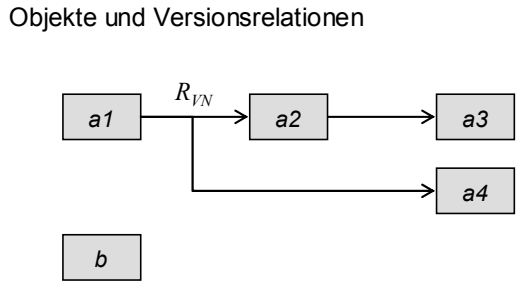
4.3.2 Versionspfade zwischen Objekten

Durch das Verwalten von Objektversionen kann die Entstehungsgeschichte eines Objektes nachvollzogen werden. Sind zwei Objekte gegeben, die einen gemeinsamen Vorgänger bzw. Nachfolger besitzen, so können über die Versionsrelation die Pfade zwischen den beiden Objekten bestimmt werden.

Für zwei gegebene Objekte, für die die Versionspfade bestimmt werden sollen, können folgende Situationen eintreten:

1. Beide Objekte stehen in einer Vorgänger-Nachfolger-Beziehung und sind folglich über die Transitive Hülle der Versionsrelation miteinander verbunden.
2. Beide Objekte beschreiben einen parallelen Entwicklungsstand und besitzen einen gemeinsamen Vorgänger bzw. Nachfolger.
3. Beide Objekte stehen in keinerlei über die Versionsrelation abgebildeten Beziehung.

Diese drei möglichen Fälle sind in der Abbildung 4-9 veranschaulicht. Für zwei gegebene Objekte x und y können diese Fälle, wie nachfolgend gezeigt, durch das Auswerten der Versionsrelation erkannt werden.



Bestimmen der Versionspfade für verschiedene Objektkombinationen

gegebene Objekte	Beziehung zwischen den Objekten
(*a1, a3*)	Fall 1
(*a2, a4*)	Fall 2
(*a1, b*)	Fall 3

Abbildung 4-9 Mögliche Versionsbeziehungen zwischen Objekten

(4.27)

$$\text{geg.:} \quad x, y \in O \qquad \text{mit} \quad x \neq y$$

Fall 1) Zwischen den Objekten *x* und *y* existiert eine Verbindung über R_{VN} .

$$\langle R_{VN} \rangle_t (x, y) \vee \langle R_{VN} \rangle_t (y, x)$$

Ist die Bedingung für den Fall 1 nicht erfüllt, so müssen die gemeinsamen Vorgänger (Fall 2a) bzw. Nachfolger (Fall 2b) ermittelt werden. Die Suche soll dabei auf den ersten gemeinsamen Vorgänger bzw. Nachfolger beschränkt werden. Trotz dieser Beschränkung ist es möglich, dass es mehr als einen ersten gemeinsamen Vorgänger bzw. Nachfolger gibt. In (4.28) ist gezeigt, wie sich für den Fall 2a die Menge der ersten gemeinsamen Vorgänger O_{1V} ermitteln lässt. Auf ähnliche Weise lässt sich auch die Menge der ersten gemeinsamen Nachfolger ermitteln und damit der Fall 2b lösen.

(4.28)

Fall 2a) Beide Objekte besitzen einen gemeinsamen Vorgänger.

$$O_V(x, y) := \left\{ z \in O \mid \langle R_{VN} \rangle_t (z, x) \wedge \langle R_{VN} \rangle_t (z, y) \right\}$$

$$O_{nV}(x, y) := \left\{ z \in O_V(x, y) \mid \exists_{w \in O_V(x, y)} \langle R_{VN} \rangle_t (z, w) \right\}$$

$$O_{1V}(x, y) := O_V(x, y) - O_{nV}(x, y)$$

Sind sowohl die Menge der gemeinsamen Vorgänger als auch die Menge der gemeinsamen Nachfolger leer, so stehen beide Objekte in keinerlei über die Versionsrelation ableitbaren Beziehung.

(4.29)

Fall 3) Beide Objekte besitzen keine Beziehung über die Versionsrelation.

es gilt $O_{1V}(x, y) = O_{1N}(x, y) = \varnothing$

Aus der Menge der ersten gemeinsamen Vorgänger lassen sich schließlich alle die Elemente ermitteln, die in dem Versionspfad der beiden betrachteten Objekte vorkommen. Die Ordnung dieser Objekte bzw. die möglichen Wege von den Vorgängerobjekten zu den betrachteten Objekten ergeben sich schließlich aus den Versionsrelationen. Die Menge der Objekte $O_{VERB,}$ die in dem Versionspfad über einen gemeinsamen Vorgänger der Objekte *x* und *y* beteiligt sind, lassen sich schließlich wie in (4.30) dargestellt ermitteln.

(4.30)

$$O_{VERB}(x,y) := \left\{ o \in O \mid \exists_{q \in O_{IV}(x,y)} \langle R_{VN} \rangle_t (q,o) \wedge \left(\langle R_{VN} \rangle_t (o,x) \vee \langle R_{VN} \rangle_t (o,y) \right) \right\}$$

wenn gilt $O_{IV}(x,y) \neq \varnothing$

In vergleichbarer Weise lässt sich eine solche Menge auch für die gemeinsamen Nachfolger oder für den beschrieben Fall 1) ermitteln.

4.3.3 Unterschiede und Änderungen zwischen Objekten

Für die Bewertung zweier Objektzustände wird zwischen einem *Unterschied* und einer *Änderung* differenziert. Dies beruht auf den beiden möglichen Vergleichsansätzen, die einerseits den Objektzustand und andererseits die Objektentstehung bewerten. Im Gegensatz zu *Unterschieden*, die aus dem Zustand der gültigen Attribute abgeleitet werden, lassen *Änderungen* nur auf unveränderte Attribute schließen. Das Ermitteln von *Änderungen* ist daher nur dann sinnvoll, wenn die zu vergleichenden Objekte eine gemeinsame Entstehungsgeschichte besitzen und dementsprechend über die Versionsrelation zueinander in Beziehung stehen.

Ermitteln der Unterschiede

Das Bestimmen der *Unterschiede* kann auf den Vergleich der gültigen Attribute zurückgeführt werden. Aufgrund der Bedingung (4.4) lassen sich die voneinander verschiedenen Attribute schließlich wie in (4.31) dargestellt mengentheoretisch beschreiben.

(4.31)

$$\text{geg.:} \quad x, y \in O \qquad \text{mit } x \neq y$$

$$A_{DIFF}(x,y) := \left(A_{GÜLTIG}(x) \cup A_{GÜLTIG}(y)\right) - \left(A_{GÜLTIG}(x) \cap A_{GÜLTIG}(y)\right)$$

Für das Bestimmen der *Unterschiede* ist es dabei unerheblich, ob beide Objekte zur gleichen Objektklasse gehören oder ob eine Beziehung über die Versionsrelation besteht. Zusätzlich wird deutlich, dass ein Vergleich der Attribute ohne die Angabe eines Planungsstandes durchgeführt werden kann. Die Bedingung der Gleichheit ist demzufolge unabhängig vom tatsächlich verknüpften Objekt definiert. Diese Vereinfachung ist möglich, weil bei der Veränderung eines verknüpften Objektes keine Aktualisierung der Verknüpfung erforderlich ist.

Bereits auf der Grundlage der beiden gegebenen Objekte können Aussagen über die Menge A_{DIFF} abgeleitet werden. Zu dieser Menge gehören beispielsweise die ID-Attribute, die zur Identifizierung der betrachteten Objekte verwendet werden und sich zwangsläufig unterscheiden. Gehören die betrachteten Objekte zu verschiedenen Objektklassen, so sind auch die Attribute enthalten, die auf unterschiedliche Attributdefinitionen zurückzuführen sind.

Ermitteln der Änderungen

Für die in Abschnitt 4.3.2 beschriebenen Fälle 1 und 2 können neben den Unterschieden auch die geänderten Attribute bestimmt werden. Seien beispielsweise das erste gemeinsame Vorgängerobjekt (4.28) sowie die an der Verbindung beteiligten Objekte (4.30) bekannt, so lassen sich aus den darin dokumentierten Änderungen die (un-)veränderten Attribute ableiten. Dieser Ansatz kann für die Umsetzung in einem Softwaresystem Vorteile bieten, da auf das Ermitteln aller gültigen Attribute verzichtet und der „Vergleich“ der Attribute reduziert wird.

Sei *v* zunächst ein unmittelbarer Vorgänger des Objektes *n*, so sind aufgrund der Bedingung (4.19) die tatsächlich durchgeführten Änderungen eine Teilmenge der in *n* enthaltenen Attribute. Besitzt das Objekt *n* mit dem Objekt *v* dagegen nur einen unmittelbaren Vorgänger, so ist gewährleistet, dass die Attribute von *n* die tatsächlich durchgeführten Änderungen gegenüber *v* beschreiben. Diese grundlegende Eigenschaft des Versionsmodells ist in (4.32) dargestellt.

(4.32)

$$\text{geg.:}\quad v,n \in O \qquad \text{mit } (v,n) \in R_{VN}$$

$$A_{DELTA}(v,n) := \{\, a \in n \mid a \notin v \,\} \qquad \Rightarrow \qquad A_{DELTA}(v,n) \subseteq n$$

Besitzt *n* nur einen Vorgänger, so kann der Vergleich $a \notin v$ entfallen und es gilt :
$A_{DELTA}(v,n) = n$

In Anlehnung an die Ermittlung der gültigen Attribute eines Objektes (4.18) kann die Beschreibung (4.32) erweitert werden, um die Änderungen eines Objektes *n* gegenüber einem beliebigen Vorgänger *v* zu ermitteln. Neben den Objekten *v* und *n* sei zusätzlich eine Objektmenge O_{VN} bekannt, die genau eine Verbindung zwischen *v* und *n* beschreibt.

(4.33)

$$\text{geg.:}\quad v,n \in O,\ O_{VN}(v,n) \quad \text{mit } (v,n) \in \langle R_{VN} \rangle_t, \quad v,n \in O_{VN}(v,n)$$

und

$$\bigvee_{x \in O_{VN}(v,n) \wedge x \neq v \wedge x \neq n} \left(\underbrace{\bigvee_{y,z \in O_{VN}(v,n)} R_{VN}(y,x) \wedge R_{VN}(x,z)}_{\text{Jedes Objekt } x \text{ besitzt in der Menge } O_{VN} \text{ einen Vorgänger und einen Nachfolger.}} \quad \wedge \quad \underbrace{\bigvee_{r,s \in O_{VN}(v,n) \wedge r \neq y \wedge s \neq z} \neg R_{VN}(r,x) \wedge \neg R_{VN}(x,s)}_{\text{In der Menge } O_{VN} \text{ gibt es keine weiteren Objekte, die Vorgänger oder Nachfolger von } x \text{ sind.}} \right)$$

$$A_{DELTA}(v,n) := \left\{ a \in A \mid (a \in n \wedge a \notin v) \vee \left(\bigvee_{m \in M_{ALT}(n)} a \in m \ \wedge \bigvee_{q \in O_{VN}(v,n)} R_{VN}(q,n) \wedge a \in A_{DELTA}(v,q) \right) \right\}$$

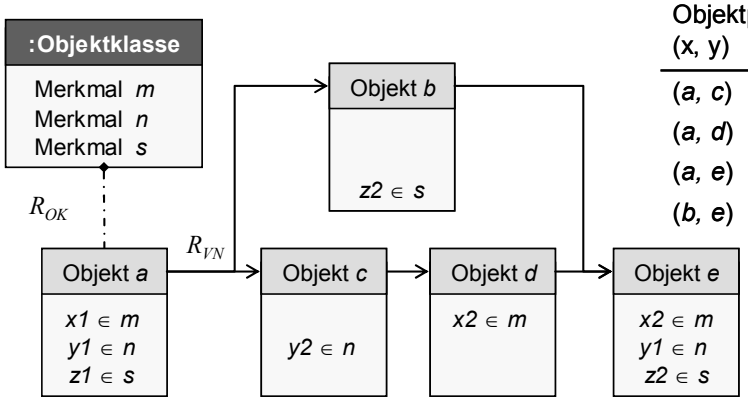
Zusammenhang zwischen Unterschied und Änderung

Dieser Ansatz zeigt, dass das Ermitteln von Änderungen durch ein Aufsammeln der geänderten Attribute über eine der möglichen Verbindungen erfolgen kann. Im Unterschied zu den über (4.31) beschriebenen Attributen werden hier jedoch diejenigen Attribute ermittelt, die im Laufe der betrachteten Versionsgeschichte geändert wurden. In der Menge A_{DELTA} sind folglich auch solche Attribute enthalten, die durch erneute Änderungen wieder rückgängig gemacht wurden. Eine Änderung beschreibt dementsprechend nicht zwingend einen Unterschied und lässt sich, sofern die Klassenzugehörigkeit der betrachteten Objekte identisch ist, in folgender Teilmengenrelation ausdrücken.

(4.34)

$$\text{es gilt} \quad A_{DIFF}(v,n) \subseteq A_{DELTA}(v,n) \quad \text{wenn} \quad \underset{k \in K}{\exists} R_{OK}(v,k) \wedge R_{OK}(n,k)$$

Dieser Zusammenhang ist in der Abbildung 4-10 dargestellt, die *Unterschiede* und *Änderungen* an einem Beispiel gegenüberstellt.



Objektpaar (x, y)	Unterschiede A_{DIFF}(x,y)	Änderungen A_{DELTA}(x,y)
(a, c)	{y2}	{y2}
(a, d)	{x2, y2}	{x2, y2}
(a, e)	{x2, z2}	{x2, y1, z2}
(b, e)	{x2}	{x2, y1}

Abbildung 4-10 Unterschiede und Änderungen zwischen Objekten

Über (4.33) wird zunächst der Fall 1 aus dem Abschnitt 4.3.2 betrachtet. Der Fall 2 lässt sich in Anlehnung an die Bestimmung der Unterschiede schließlich, wie in (4.35) gezeigt, bestimmen. Über die Schnittmenge werden dabei genau die Attribute entfernt, die unabhängig voneinander in den gleichen Zustand geändert wurden.

(4.35)

$$\text{geg.:} \quad x, y \in O, \;\; v \in O_{1V}(x,y)$$

$$A_{DELTA}(x,y) \;\; := \left(A_{DELTA}(v,x) \cup A_{DELTA}(v,y)\right) - \left(A_{DELTA}(v,x) \cap A_{DELTA}(v,y)\right)$$

4.3.4 Änderungen zwischen Planungsständen

Ebenso wie Objekte können komplette Planungsstände miteinander verglichen werden. Bei diesem Vergleich interessieren die Änderungen auf Objektebene, wobei drei disjunkte Objektmengen gebildet werden können. Über die Zuordnung zu einer der Mengen werden schließlich 1.) die unveränderten, 2.) die veränderten sowie 3.) die neuen oder gelöschten Objekte unterschieden. Im Sinne der in Abschnitt 4.3.3 verwendeten Begriffe werden hierbei Änderungen betrachtet, da der in (4.36) beschriebene Vergleich auf der Grundlage der verwendeten Objekte durchgeführt wird.

(4.36)

$$\text{geg.:} \quad p, q \in P \qquad \text{mit } p \neq q$$

$$O_{UNVERÄNDERT}(p,q) \qquad := \{ x \in O \mid x \in p \wedge x \in q \}$$

$$O_{VERÄNDERT}(p,q) \qquad := \left\{ x \in O \mid \underset{z \in O}{\exists} \left(x \in p \wedge z \in q \vee x \in q \wedge z \in p \right) \wedge \left(\langle R_{VN} \rangle_t (x,z) \vee \langle R_{VN} \rangle_t (z,x) \right) \right\}$$

$$O_{NICHT\_ENTHALTEN}(p,q) \qquad := (p \cup q) - \left(O_{UNVERÄNDERT}(p,q) \cup O_{VERÄNDERT}(p,q)\right)$$

Bezogen auf einen Planungsstand kann zusätzlich zwischen den neuen und den gelöschten Objekten unterschieden werden. In einem Planungsstand p sind gegenüber einem Planungsstand q beispielsweise genau die Objekte „neu“, die in den beiden Mengen p und $O_{NICHT\_ENTHALTEN}(p, q)$ enthalten sind.

4.3.5 Anlegen eines Planungsstandes

Um eine neue Entwurfslösung in dem Versionsmodell abzubilden, ist das Anlegen eines Planungsstandes notwendig. Hierbei sind die Konsistenzbedingungen des Versionsmodells einzuhalten. Die Problematik entsteht aus der Aktualisierung einer vorhandenen Entwurfslösung, da im Versionsmodell nur Änderungen verwaltet werden. Die veränderten Objekte werden hierfür durch Nachfolger ersetzt, die die geänderten Attribute als Delta-Werte enthalten. Sind die Änderungen eines zu aktualisierenden Planungsstandes gegeben, so werden für das Anlegen eines Planungsstandes die zu ersetzenden Objekte mit den zu aktualisierenden Attributen gesucht.

Für die weitere Bewertung seien die Änderungen als neue Objekte mit ihren geänderten Attributen sowie den Versionsrelationen zu den zu ersetzenden Vorgängern gegeben. Zusätzlich sei bekannt, welche Objekte in dem neuen Planungsstand ersatzlos zu entfernen sind. Um die unveränderten Objekte zu bestimmen, geht die Beschreibung (4.37) davon aus, dass ein überarbeitetes Objekt stets ein Nachfolger eines im Planungsstand p enthaltenen Objektes ist. Zusätzlich wird ausgeschlossen, dass in der Menge O_{NEU} ein Vorgänger für ein Objekt des Planungsstandes p enthalten ist.

(4.37)

geg.: $p \in P$ (p ist der zu aktualisierende Planungsstand)

$$O_{NEU} := \{ x \in O \mid x \text{ ist ein neues oder überarbeitetes Objekt} \}$$
$$O_{ENTFERNT} := \{ x \in O \mid x \text{ wird ersatzlos entfernt} \}$$
$$\text{wobei gilt} \quad \forall_{x \in O_{ENTFERNT}} \; \forall_{y \in O_{NEU}} \; \neg R_{VN}(x, y) \wedge y \notin p$$

$$O_{UNVERÄNDERT} := \left\{ x \in p \mid x \notin O_{ENTFERNT} \wedge \forall_{y \in O_{NEU}} \neg R_{VN}(x, y) \right\}$$

Unter diesen Voraussetzungen lassen sich drei Problemstellungen identifizieren, die aus

1. dem Zusammenlegen von Objekten,
2. dem Teilen von Objekten sowie
3. dem Ändern der Objektklasse, also der Objektevolution, resultieren.

Fall 1): Beim Zusammenlegen von Objekten besitzt ein Objekt mehrere direkte Vorgänger. Für diesen Fall müssen bestimmte Attribute aktualisiert werden, um die in (4.19) formulierte Bedingung zu erfüllen. In dieser Bedingung wird gefordert, dass über jeden direkten Vorgänger die gleichen Attribute zur Vervollständigung des Objektes ermittelt werden können. Existieren für ein Objekt mehrere direkte Vorgänger, so müssen folglich die zu aktualisierenden Attribute bestimmt werden.

Fall 2): Beim Teilen eines Objektes werden mehrere direkte Nachfolger erzeugt, die alle Element des gleichen Planungsstandes sind. Für diesen Fall müssen mindestens die Objekte aktualisiert werden, die eine Referenz auf das geteilte Objekt definieren und in dem neuen Planungsstand nicht entfernt werden sollen. Die Notwendigkeit für eine Aktualisierung ergibt sich aus der Bedingung (4.21), die die Eindeutigkeit von Verknüpfungen fordert. Folglich werden alle Objekte und ihre Attribute gesucht, die zur Wahrung eindeutiger Verknüpfungen aktualisiert werden müssen.

Fall 3): Beim Ändern der Objektklasse hat ein Objekt im Vergleich zu seinem Vorgänger die Zugehörigkeit zu einer Objektklasse geändert. Mit einer anderen Objektklasse können andere Merkmale definiert sein, die bei einer Vervollständigung des Objektes nicht über den Vorgänger bezogen werden können. Ein Sonderfall stellt das Anlegen eines neuen Objektes dar, da hierbei keinerlei Attribute über einen Vorgänger bezogen werden können. Für beide Problemstellungen sind also die Merkmale zu bestimmen, die beim Ermitteln der gültigen Attribute eine vollständige Belegung gemäß der Definition in der zugehörigen Objektklasse garantieren.

Aus den in (4.37) gegebenen Mengen können die drei unterschiedenen Fälle abgeleitet werden. Dabei ist es möglich, dass sich die Fälle gegenseitig überlagern und dementsprechend ein Objekt nach mehreren Kriterien zu bewerten ist. Bei einer solchen Überlagerung sind die zu aktualisierenden Objekte und Attribute durch Superposition der Einzelergebnisse zu bestimmen.

Für das Zusammenlegen von Objekten seien das überarbeitete Objekt und dessen direkte Vorgänger bekannt (4.38). In der Menge $A_{ALT}(o, q)$, die in (4.18) definiert ist, sind alle gültigen Attribute für das Objekt o enthalten, die über den Vorgänger q und dessen Vorgänger ermittelt werden können. Nach dieser Definition werden genau die Attribute aufgesammelt, die zur vollständigen Beschreibung von o fehlen. Wird diese Menge für jeden Vorgänger von o ermittelt, so lassen sich aus der Schnittmenge alle gleichen Attribute bestimmen. Folglich sind auch die Attribute bekannt, die über verschiedene Vorgänger einen unterschiedlichen Zustand definieren und somit in dem Objekt o zusätzlich zu aktualisieren sind. In Ergänzung zu (4.17) kann schließlich für die Menge $M_{NEU}(o)$ eine alternative Definition formuliert werden, um die Bedingung (4.19) zu erfüllen. Die in (4.38) formulierte Bedingung kann zusätzlich sowohl für neue Objekte als auch für die Änderung der Klassenzugehörigkeit angewendet werden. Dadurch erübrigt sich eine getrennte Betrachtung beider Fälle.

(4.38)

$$\text{geg.:}\quad o \in O_{NEU}, \qquad O_{VORGÄNGER}(o) := \{\, x \in O \mid R_{VN}(x,o) \,\}$$

$$A_{ALT}(o) := \left\{ a \in A \mid \forall_{x \in O_{VORGÄNGER}(o)} \; a \in A_{ALT}(o,x) \right\}$$

$$M_{ALT}(o) := \left\{ m \in M \mid \exists_{a \in A_{ALT}(o)} \; a \in m \right\}$$

$$M_{NEU}(o) := M(o) - M_{ALT}(o)$$

Im Gegensatz zum Zusammenlegen kann das Teilen von Objekten die Konsistenz anderer Objekte verletzen. Diese Objekte sind aus dem zu aktualisierenden Planungsstand über die Auswertung der Bedingung (4.21) zu ermitteln. Wird diese Bedingung für ein Objekt und ein hierfür gültiges Attribut verletzt, so ist die hierüber definierte Verknüpfung in dem neuen Planungsstand zu aktualisieren. Für einen zu aktualisierenden Planungsstand *p* sind in (4.39) die zu aktualisierenden Objekte und Attribute beschrieben, die durch geteilte Objekte andernfalls ein Auflösen der Referenzen verletzen.

(4.39)

$$\text{geg.:}\quad O_{GETEILT} := \left\{ x \in p \mid \underset{y,z \in O_{NEU}}{\exists} \; y \neq z \wedge R_{VN}(x,y) \wedge R_{VN}(x,z) \right\}$$

Das Attribut *a* eines Objektes *o* ist auf Grund eines geteilten Objektes zu aktualisieren,

wenn:

$$\underset{x \in O_{GETEILT}}{\exists} \; a \in A_{GÜLTIG}(o) \wedge R_{AO}(a,x) \wedge o \notin O_{ENTFERNT}$$

Aus der Bedingung (4.39) wird deutlich, dass ein Überprüfen der Konsistenz durch eine Bewertung aller Objekte eines Planungsstandes sehr aufwendig werden kann. Die Auswertung der Bedingung (4.39) ist jedoch nur dann erforderlich, wenn der gegebene und neu anzulegende Planungsstand nicht zwingend konform zu der zugrunde liegenden Datenstruktur ist. Dies ist beispielsweise dann der Fall, wenn auf einer Untermenge des Planungsstandes gearbeitet wird und dadurch die Integrität der Referenzen nur für den betrachteten Teildatensatz gewährleistet werden kann. Unter dieser Voraussetzung ist es jedoch ausreichend, nur die Objekte nach der Bedingung (4.39) zu überprüfen, die nicht zu dem betrachteten Teildatensatz gehören.

4.3.6 Zurückgreifen auf alte Objekte

In der Betrachtung des Abschnittes 4.3.5 wird das Zurückgreifen auf alte Objekte, also auf Objekte, die in der Entwicklungsgeschichte vor Objekten des zu aktualisierenden Planungsstandes liegen, ausgeschlossen. Wird die Verwendung solcher Objekte erlaubt, so wird die auf *Chronologie* beruhende Ordnung zwischen den Objekten zweier Planungsstände verletzt. In einem solchen Fall besitzt ein Planungsstand gegenüber einem anderen Planungsstand sowohl Nachfolger als auch Vorgänger. Auch wenn eine solche Situation den geforderten Konsistenzbedingungen des Versionsmodells nicht grundsätzlich widerspricht, so wird die Interpretation der Planungsstände erschwert. Um diese Fälle zu vermeiden, soll das Zurückgreifen auf eine alte Objektversion durch das Ableiten eines neuen Objektes erfolgen. Als Vorgänger sollen sowohl das alte als auch das zu ersetzende Objekte definiert werden. Für zwei beliebige Planungsstände *p* und *q* soll folglich folgende Bedingung erfüllt sein:

(4.40)

$$\text{geg.:}\quad p,q \in P$$

$$\underset{a \in p \wedge b \in q}{\exists} \; \langle R_{VN} \rangle_t (a,b) \quad \Leftrightarrow \quad \underset{c \in p \wedge d \in q}{\forall} \; \neg \langle R_{VN} \rangle_t (d,c)$$

In der Abbildung 4-11 ist der vorgeschlagene Ansatz für das Zurückgreifen auf alte Objekte gezeigt. Das Objekt b_1 kann in dem Planungsstand q aufgrund der Bedingung (4.40) nicht direkt verwendet werden. Um dennoch das Zurückgreifen auf das Objekt b_1 sowie das Ersetzen von b_2 zu dokumentieren, wird ein neues Objekt b_3 erzeugt, das als Vorgänger sowohl b_1 als auch b_2 besitzt.

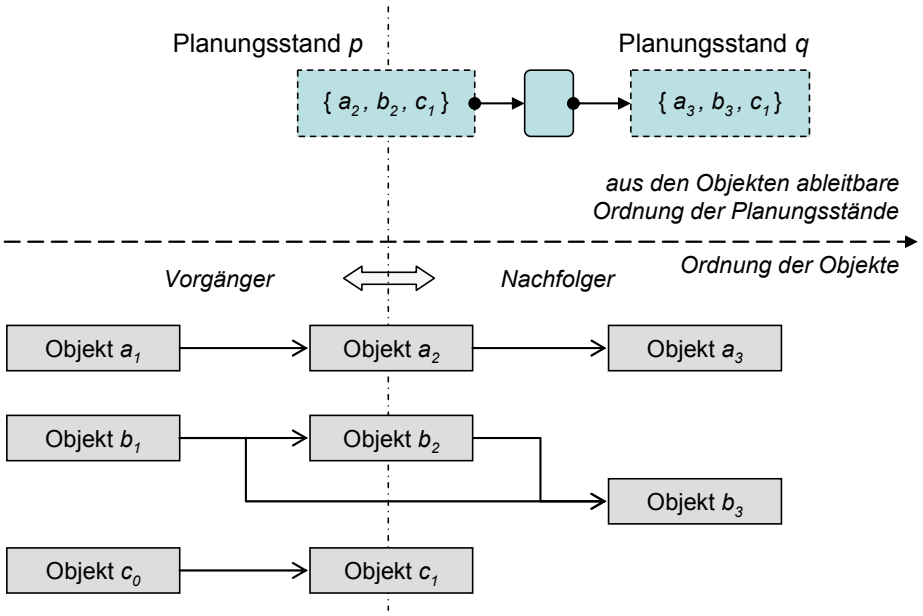


Abbildung 4-11 Zurückgreifen auf alte Objekte unter Einhaltung der Bedingung (4.40)

5 Methoden zur Unterstützung des Kooperationsmodells

In diesem Kapitel werden Methoden vorgestellt, die das Versionsmodell ergänzen und das kooperative Arbeiten in langen Transaktionen unter Verwendung von Teildatensätzen unterstützen. Hierfür werden Methoden benötigt, die 1.) das Bilden von Teildatensätzen, 2.) den Vergleich von Planungsständen, 3.) die Re-Integration der Änderungen und schließlich 4.) das Zusammenführen divergierender Planungsstände unterstützen.

Der *erste Abschnitt* beschreibt das GMSD-Schema, das für die Beschreibung von Teildatensätzen entwickelt wurde. Als wesentliches Merkmal wird das Konzept der vordefinierbaren Filter vorgestellt. Ein solcher Filter wird mit der Auswahl von Objekten kombiniert und dadurch zu einer Teildatensatzbeschreibung vervollständigt. Auf dieser Grundlage ist es möglich, komplexe Teildatensätze mit wenigen Nutzerinteraktionen und einem hohen Maß an Individualisierbarkeit zu beschreiben. Für die Anwendung des GMSD-Schemas wird gezeigt, wie ein solcher Filter formalisiert und auf das Versionsmodell angewendet wird. Hierbei wird ein neuer Planungsstand erzeugt, der den Teildatensatz als Änderung gegenüber dem Gesamtdatensatz beschreibt.

Im *zweiten Abschnitt* wird ein generisches Vergleichsverfahren vorgestellt, das den Vergleich zweier Planungsstände ermöglicht. Als zentrale Problemstellung erweist sich das Identifizieren der Objekte, das meist ohne eindeutigen Identifikator über den Objektzustand durchgeführt werden muss. Die hieraus resultierende Komplexität wird durch ein iteratives Verfahren reduziert, das ausgehend von identifizierbaren Objekten eine Bewertung der verknüpften Objekte vornimmt. Hierfür werden drei Verknüpfungstypen unterschieden, aus denen Annahmen über weitere Objektpaare abgeleitet werden. Auf diese Weise werden akzeptable Laufzeiten erreicht, die gleichzeitig zu einer hohen Erkennungsrate bei einer tolerierbaren Fehlerquote führen.

Im *dritten Abschnitt* wird die Re-Integration der gefundenen Änderungen diskutiert. Bei diesem Schritt wird das Bilden des Teildatensatzes konzeptionell zurückgenommen, indem die Änderungen, die den Teildatensatz beschreiben, durch weitere Änderungen unwirksam gemacht werden. Diese Änderungen sind im Versionsmodell dokumentiert und müssen in Einklang mit den Änderungen des Planungsschrittes gebracht werden. Im Mittelpunkt der Re-Integration steht daher die Konsistenz der Planungsdaten, die aus verschiedenen Gründen verletzt sein kann. Das Erkennen inkonsistenter Planungsdaten und vor allem die Unterscheidung ihrer Ursache ist für einen generischen Ansatz jedoch nicht vollständig möglich. Das Problem der Datenkonsistenz und ihrer Auswirkungen auf den aktualisierten Gesamtdatensatz wird daher differenziert betrachtet.

Der Schritt der Re-Integration bildet die Grundlage, um im *vierten Abschnitt* die Problematik des Zusammenführens zu diskutieren. Im Unterschied zur Re-Integration sind beim Zusammenführen parallel entstandene Entwurfslösungen miteinander zu vereinen. Hierbei ist zusätzlich mit Konflikten zu rechnen, die durch die betroffenen Planungsbeteiligten einvernehmlich aufzulösen sind. Für die Unterstützung der asynchronen Kooperation ist vor allem das Vorschlag-Zustimmungsverfahren interessant, um wechselseitig einen Vorschlag über einen zusammengeführten Planungsstand unterbreiten zu können. Ein solcher Planungsstand kann durch ein formales Verfahren generiert werden, das nur die Änderungen übernimmt, die im Sinne des Versionsmodells miteinander vereinbar sind.

5.1 Formalisierung von Teildatensätzen

Die Nutzung von Teildatensätzen hat verschiedene Gründe. So wird beispielsweise eine Verringerung der zu übertragenden und zu verarbeitenden Planungsdaten oder das gezielte Ausblenden nicht gewünschter Informationen verfolgt. Im Rahmen dieser Arbeit werden Teildatensätze in erster Linie zur Beschreibung der von einem Anwenderprogramm fehlerfrei verarbeitbaren Informationen verwendet.

Ein Teildatensatz stellt im Sinne des Versionsmodells eine Menge von Elementen dar, die die gewünschten Informationen beschreiben. Für die Auswahl eines Teildatensatzes sind folglich alle Objekte und Attribute zu benennen, die als Teilmenge des betrachteten Planungsstandes die gewünschten Informationen enthalten. Eine solche Auswahl soll möglichst einfach und mit wenigen Nutzerinteraktionen durchführbar sein. Zusätzlich wird gefordert, dass die Menge der benötigten Daten in einer einzelnen „Anfrage“ vollständig beschrieben werden kann.

Speziell für diese Anforderungen wurde mit dem ***G**eneralised-**M**odel-**S**ubset-**D**efinition*-Schema (GMSD-Schema) eine Spezifikation entwickelt, die eine Auswahl der benötigten Objekte und Attribute mit wenigen Nutzerinteraktionen ermöglicht (Weise et al. 2003). Wesentliches Merkmal der Teildatensatzbeschreibung ist die Kombination einer zur Laufzeit auswählbaren Objektmenge mit einem auf Klassenebene vordefinierbaren Filter.

5.1.1 Das Generalised-Model-Subset-Definition-Schema

Das GMSD-Schema basiert auf der Verwendung von Objektmengen. Diese Objektmengen werden durch verschiedene Auswahlkriterien und mengentheoretische Verknüpfungen aus einer gegebenen Grundmenge bestimmt. Eine auf diese Weise ermittelte Objektmenge beschreibt entweder den gesuchten Teildatensatz oder eine weiter auszuwertende Grundmenge. Das hierfür notwendige Abbruchkriterium ist in der Teildatensatzbeschreibung formuliert, die die anzuwendenden Auswahlkriterien und mengentheoretischen Verknüpfungen als eine Folge von Anweisungen definiert. Am Ende dieser Anweisungsfolge steht der gewünschte Teildatensatz schließlich als Objektmenge zur Verfügung (siehe Abbildung 5-1).

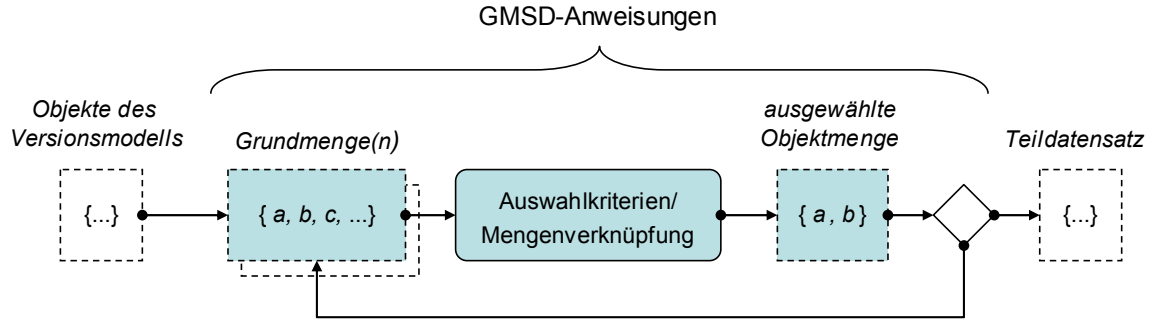


Abbildung 5-1 Prinzip des Generalised-Model-Subset-Definition-Schemas

Eine Grundmenge von Objekten wird zu Beginn durch die Auswahl eines Planungsstandes bestimmt. Auf diese Weise wird der Bezug zwischen einem ausgewählten Objekt und einem Planungsstand hergestellt, der für das Ermitteln verknüpfter Objekte notwendig ist (siehe Abschnitt 4.4.1). Ein ausgewähltes Objekt wird aus diesem Grund gemeinsam mit einem Planungsstand betrachtet und bildet mit diesem für die weitere Auswertung ein Paar. Bei der Verknüpfung von Grundmengen, beispielsweise beim Bilden einer Schnittmenge, wird stets ein solches Paar bewertet. Zwei Elemente sind demzufolge nur dann identisch, wenn sie sowohl das gleiche Objekt als auch den gleichen Planungsstand beschreiben.

Für die Bewertung einer gegebenen Grundmenge wird in dem GMSD-Schema zwischen einer *Selektion* und einem *Filter* unterschieden. Im Fall der Selektion werden die Eigenschaften eines Objektes betrachtet, die für die Auswahl der Objekte genutzt werden. Zu den auswertbaren Eigenschaften gehören beispielsweise der Zustand der Attribute und die Verknüpfung mit anderen Objekten. Dadurch wird eine gezielte Auswahl der Objekte möglich. Im Fall des Filters werden die Objekte dagegen nach ihrer Zugehörigkeit zu einer Objektklasse und ihrer Verknüpfung mit anderen Objekten bewertet. Im Vergleich zur Selektion wird ein Filter neben der Auswahl von Objekten zusätzlich für das Festlegen der benötigten Attribute verwendet.

Das GMSD-Schema liegt als ein objekt-orientiertes Datenmodell vor, das für die Beschreibung von Teildatensätzen genutzt wird. Die vollständige Spezifikation des GMSD-Schemas ist im Anhang zu finden. Als Auszug werden in der Abbildung 5-2 die nachfolgend diskutierten Objektklassen in EXPRESS-G-Notation gezeigt. Ein Teildatensatz wird nach diesem Schema über ein Objekt der Objektklasse *PartialModelQuery* definiert, das eine ausgewählte Objektmenge (*SelectedObjectSets*) wahlweise mit einem zusätzlichen Filter (*DefinesFeatureSubset*) kombinieren kann. Für die Auswahl einer Objektmenge wird der SELECT-Datentyp *QuerySetSelect* verwendet, der eine Objektmenge über eine von vier Möglichkeiten beschreiben kann. Hierzu gehören:

- die Benennung eines Planungsstandes (*ModelID*),
- die Verknüpfung von Objektmengen (*SetOperation*) [1],
- die Objektselektion (*Select*) und schließlich
- die Möglichkeit zur Verwendung anderer Teildatensätze (*PartialModelQuery*).

Ein Teildatensatz lässt sich somit aus einer beliebigen Kombination von Objektmengen zusammenstellen. Bei einer solchen Kombination ist lediglich auszuschließen, dass eine Objektmenge rekursiv verwendet wird. Eine Objektmenge bzw. ein Teildatensatz darf folglich nicht über sich selbst definiert werden, da hierdurch ein unbestimmter Zustand entsteht.

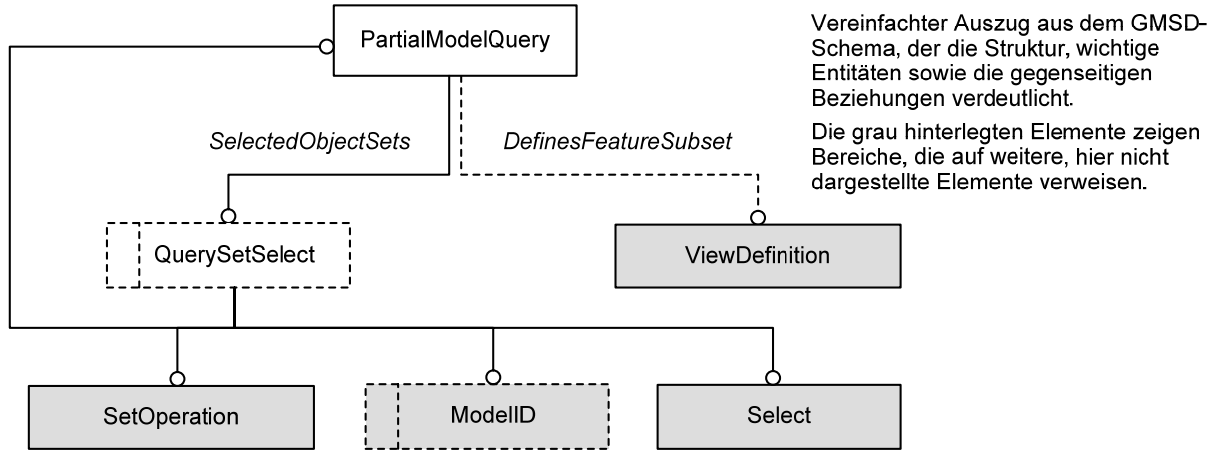


Abbildung 5-2 Vereinfachter Auszug aus dem GMSD-Schema in EXPRESS-G-Notation

Die Objektselektion wird auf einer Objektmenge (Grundmenge) ausgeführt, die durch ein *QuerySetSelect* definiert wird. Dadurch können alle Möglichkeiten zur Definition einer Ob-

[1] Vereinigung, Durchschnitt und Differenz

jektmenge genutzt werden. Zusätzlich zur Grundmenge werden die hierauf anzuwendenden Auswahlkriterien definiert. Über diese Auswahlkriterien ist es möglich, Objekte über den Zustand ihrer Attribute, die Objektidentifizierung, die Zugehörigkeit zu einer Objektklasse sowie die Verknüpfung zu oder mit anderen Objekten auszuwählen. Auf diese Weise lassen sich beispielsweise alle Objekte einer Objektklasse „Wand“ bestimmen, die eine Höhe von 2,5 m haben.

Auf eine ausführliche Betrachtung zur Objektselektion sowie zur Verknüpfung von Objektmengen wird mit dem Hinweis auf den Anhang verzichtet. Hieraus entstehen keine konzeptionellen Fragen. Demgegenüber stellt die Definition und vor allem die Auswertung eines Filters eine Besonderheit des GMSD-Schemas dar, die in der nachfolgenden Beschreibung ausführlich betrachtet wird.

5.1.2 Das Konzept der Filter

Ein Filter wird über ein Objekt der Objektklasse *ViewDefinition* definiert und dient der Auswahl der benötigten Daten aus einer zuvor selektierten Objektmenge. Jedes Objekt der selektierten Objektmenge wird über diesen Filter bewertet, der die Objekte durch ihre Klassenzugehörigkeit unterscheidet. Auf diese Weise können Objekte einer bestimmten Objektklasse aus der Selektionsmenge entfernt werden. Neben der Auswahl von Objekten werden mit dem Filter die gewünschten Attribute festgelegt. Für alle Objekte einer Objektklasse wird demnach beschrieben, welche Attribute zu entfernen bzw. beizubehalten sind. Eine Sonderstellung nehmen hierbei Attribute ein, die eine Verknüpfung mit anderen Objekten definieren. Verknüpfte Objekte werden gesondert in die Filterauswertung einbezogen. Durch diese gesonderte Auswertung kann ein Objekt der ursprünglichen Selektionsmenge nachträglich hinzugefügt und eine alternative Filterdefinition angewendet werden. Auf diese Weise ist es möglich, neben der Klassenzugehörigkeit auch die Verwendung der Objekte zu berücksichtigen.

Ein Filter stellt ein zentrales Element des GMSD-Schemas dar. Das Filterkonzept ist aus zwei Gründen für die Beschreibung der Teildatensätze interessant. Einerseits stellt ein Filter einen eigenständigen Teil einer Abfrage dar und kann dadurch vollständig vordefiniert werden. Andererseits ist eine sehr differenzierte Auswahl möglich, die besonders bei stark vernetzten Datenmodellen bereits aus einer geringen Anzahl selektierter Objekte ableitbar ist. Für das IFC-Modell ist es dadurch beispielsweise möglich, mit einem einzelnen Stockwerkobjekt (*IfcBuildingStorey*) alle zugehörigen Objekte auszuwählen.

Die Abbildung 5-3 zeigt an einem Beispiel das Prinzip eines Filters. In diesem Beispiel wird auf die gegebene Grundmenge mit den Objekten *a*, *b* und *c* ein Filter angewendet, der ausschließlich Objekte der Objektklasse *Raum* bewertet. Dadurch werden die Objekte *b* und *c*, die zur Objektklasse *Wand* gehören, aus der Objektmenge entfernt. Die Objekte der Objektklasse *Raum* sollen dagegen mit einem Teil ihrer Attribute erhalten bleiben. Zu diesen Attributen gehören die Begrenzung des Raumes sowie die Angabe der Raumfläche. Da die Begrenzung eines Raumes über die umhüllenden Bauelementen beschrieben wird, stellt ein *Raum* über das Attribut *Begrenzung* eine Verknüpfung zu anderen Objekten her. Für diese Objekte wird in dem Beispiel ein eigener Filter definiert, der nun die verknüpften Objekte der Objektklasse *Wand* bewertet. Über diesen Filter werden schließlich das Wand-Objekt *b* und das zugehörige Fenster-Objekt *d* mit den hierfür gewünschten Attributen ausgewählt. Um den Teildatensatz verwalten und in den weiteren Ablauf integrieren zu können, werden Kopien der ausgewählten Objekte angelegt, die die gewünschten Daten enthalten.

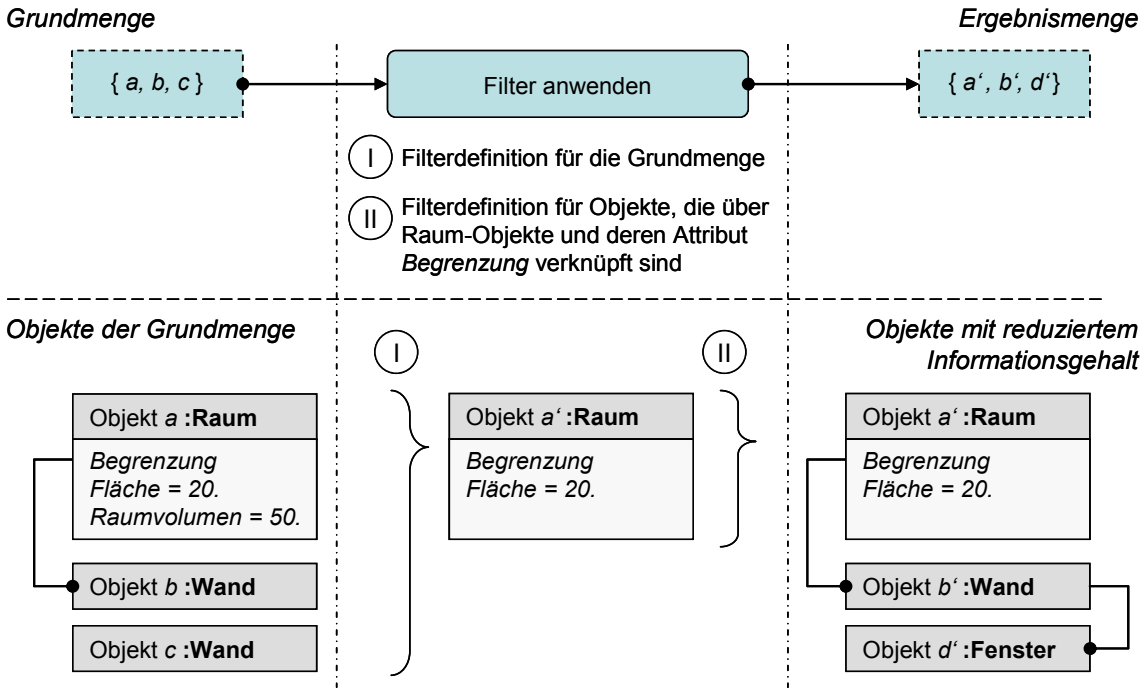


Abbildung 5-3 Beispiel für das Anwenden eines Filters auf eine gegebenen Objektmenge

In dem Beispiel der Abbildung 5-3 werden auf das Wand-Objekt *b* zwei Filterdefinitionen angewendet. Das Wand-Objekt wird dadurch aus der Objektmenge einerseits entfernt und über die Verknüpfung mit dem Raum-Objekt *a* andererseits wieder hinzugefügt. Für eine Bewertung des Objektes *b* liegen also zwei gegensätzliche Filterdefinitionen vor, deren Gültigkeit scheinbar durch die Abfolge der Anweisungen geregelt wird. Die Anweisungsabfolge ist als Entscheidungskriterium aber nicht immer eindeutig. Zu einem Teildatensatz werden daher alle Objekte und Attribute gezählt, die von mindestens einem der anzuwendenden Filter ausgewählt wurden.

5.1.3 Formalisierung der Filterdefinition

Für die Formalisierung wird analog zum Versionsmodell ein Mengensystem definiert, das auf Ebene der Attribute, der Objekte sowie der selektierten Objektmenge einen entsprechenden Filter definiert. Auf diesem Mengensystem werden schließlich die einzuhaltenden Konsistenzbedingungen formuliert, um einen gültigen Filter zu beschreiben.

In der Formalisierung gilt die Regelung, dass ein Filter nur dann zu definieren ist, wenn ein Element aus der betrachteten Elementmenge über einen Filter nicht zu entfernen ist. Ist für ein Element kein Filter definiert, so wird dieses Element aus der Elementmenge vollständig herausgefiltert.

Menge der Attributfilter AF: Ein Attributfilter legt die Verwendung und weitere Auswertung für Attribute einer Attributdefinition fest. Ist für ein Attribut ein Attributfilter vorhanden, so wird dieses Attribut in den Teildatensatz aufgenommen. Zusätzlich wird über den Attributfilter festgelegt, wie die über das Attribut verknüpften Objekte zu bewerten sind.

$$AF := \{ x \mid x \text{ ist ein Attributfilter} \} \tag{5.1}$$

Menge der Objektfilter OF: Ein Objektfilter besteht aus einer Menge von Attributfiltern, die für Objekte einer Objektklasse die Verwendung der Attribute definieren. Ist für ein Objekt ein Objektfilter vorhanden, so wird das Objekt mit den gewünschten Attributen in den Teildatensatz aufgenommen. Es wird gefordert, dass in einem Objektfilter für jedes Attribut maximal ein Attributfilter enthalten ist, um für den Objektfilter die Verwendung der Attribute eindeutig festzulegen.

$$OF := \{ x \subset AF \mid x \text{ beschreibt einen Objektfilter} \} \tag{5.2}$$

Menge der Modellfilter PF: Ein Modellfilter besteht aus einer Menge von Objektfiltern, die für eine beliebig selektierte Objektmenge die Verwendung der Objekte festlegen. In einem Modellfilter ist für jede Objektklasse maximal ein Objektfilter enthalten, um für den Modellfilter die Verwendung der Objekte eindeutig zu festzulegen.

$$PF := \{ x \subset OF \mid x \text{ beschreibt einen Modellfilter} \} \tag{5.3}$$

Auf dem vorgestellten Mengensystem und dem in Kapitel 4 beschriebenen Versionsmodell werden zusätzliche Relationen definiert, die für die Formulierung eines Filters benötigt werden.

Merkmalfilterrelation R_{AFM}: Eine Merkmalfilterrelation wird zwischen einem Attributfilter und einem Merkmal definiert und legt fest, für welches Merkmal ein Attributfilter definiert ist. Über diese Beziehung wird festgelegt, dass auf Attribute dieses Merkmals der zugeordnete Attributfilter angewendet werden kann.

Es wird gefordert, dass jeder Attributfilter über die Merkmalfilterrelation genau einem Merkmal zugeordnet wird. Bei der Definition von Attributfiltern nehmen ID-Merkmale eine Sonderstellung ein, da ein ID-Attribut die eindeutige Unterscheidung der Objekte innerhalb des Versionsmodells garantiert und für jedes Objekt zusätzlich zu den im Datenmodell definierten Attributen existiert. In diesem Sinne handelt es sich nicht um eine definierte Eigenschaft der Objekte, die über einen Filter entfernt werden kann.

$$R_{AFM} := \{ (x,y) \in AF \times M \mid R_{AFM}(x,y) \text{ weist dem Merkmal } y \text{ einen Attributfilter } x \text{ zu} \} \tag{5.4}$$
$$\text{mit} \quad \forall_{x \in AF} \; \exists_{y \in M \wedge y \notin M_{ID}} R_{AFM}(x,y) \wedge \forall_{z \in M \wedge z \neq y} \neg R_{AFM}(x,z)$$

Ein Attributfilter wird nach der obigen Bedingung genau einem Merkmal zugeordnet. Umgekehrt ist es für ein Merkmal möglich, mehrere Attributfilter zuzuordnen. Dies ist notwendig, um die mit einem Attribut verknüpften Objekte über verschiedene Filterdefinitionen auswerten zu können.

Klassenfilterrelation R_{OFK}: Eine Klassenfilterrelation wird zwischen einem Objektfilter und einer Objektklasse definiert und legt fest, 1.) welche Attributfilter in einem Objektfilter enthalten sein dürfen und 2.) auf welche Objekte ein Objektfilter angewendet werden kann. Es wird gefordert, dass jeder Objektfilter über die Klassenfilterrelation genau einer Objektklasse zugeordnet wird.

(5.5)

$$R_{OFK} := \{(x,y) \in OF \times K \mid R_{OFK}(x,y) \text{ weist der Objektklasse } y \text{ einen Objektfilter } x \text{ zu}\}$$
$$\text{mit} \quad \forall_{x \in OF} \exists_{y \in K} R_{OFK}(x,y) \wedge \forall_{z \in K \wedge z \neq y} \neg R_{OFK}(x,z)$$

Aus der Zuordnung des Objektfilters zu einer Objektklasse lassen sich zusätzlich die zulässigen Attributfilter ableiten (siehe Definition 4.9).

(5.6)

$$\text{es gilt} \quad \forall_{x \in OF} \forall_{y \in x} \exists_{k \in K} R_{OFK}(x,k) \wedge \exists_{m \in k} R_{AFM}(y,m)$$

Modellfilterrelation $\mathbf{R_{AFPF}}$: Eine Modellfilterrelation definiert eine Beziehung zwischen einem Attributfilter und einem Modellfilter. Über diese Beziehung wird festgelegt, welcher Modellfilter auf verknüpfte Objekte anzuwenden ist. Der Modellfilter wird aber nur auf Objekte angewendet, die bereits zur Menge der ausgewählten Objekte gehören.

(5.7)

$$R_{AFPF} := \left\{ (x,y) \in AF \times PF \mid R_{AFPF}(x,y) \quad \begin{array}{l} \text{weist dem Attributfilter } x \text{ einen} \\ \text{Modellfilter } y \text{ zu} \end{array} \right\}$$

Selektionsrelation $\mathbf{R_S}$: Eine Selektionsrelation definiert wie die Modellfilterrelation eine Beziehung zwischen einem Attributfilter und einem Modellfilter. Im Gegensatz zur Modellfilterrelation wird über die Selektionsrelation ein Modellfilter definiert, der auf alle verknüpften Objekte angewendet wird und dadurch die Menge der ausgewählten Objekte erweitern kann.

(5.8)

$$R_S := \left\{ (x,y) \in AF \times PF \mid R_S(x,y) \quad \begin{array}{l} \text{weist einem Attributfilter } x \text{ einen Modellfilter } y \\ \text{zur Erweiterung der Selektionsmenge zu} \end{array} \right\}$$

Für die Definition eines Attributfilters soll über die Bedingung (5.9) sichergestellt werden, dass für verknüpfte Objekte immer genau ein Modellfilter definiert ist, der entweder aus einer Selektionsrelation oder einer Modellfilterrelation resultiert.

(5.9)

$$\forall_{x \in AF} \exists_{y \in PF} \underbrace{\left(R_{AFPF}(x,y) \wedge \neg R_S(x,y)\right) \vee \left(R_S(x,y) \wedge \neg R_{AFPF}(x,y)\right)}_{\begin{array}{l}\text{jeder Attributfilter } x \text{ soll entweder über eine Modellfilterrelation } R_{AFPF} \text{ oder} \\ \text{eine Selektionsreltion } R_S \text{ mit einem Modellfilter } y \text{ verbunden sein}\end{array}} \wedge$$
$$\underbrace{\forall_{z \in PF \wedge z \neq y} \neg\left(R_{AFPF}(x,z) \vee R_S(x,z)\right)}_{\begin{array}{l}\text{neben dem Modellfilter } y \text{ soll kein weiterer Modellfilter } z \\ \text{zu dem Attributfilter } x \text{ zugeordnet sein}\end{array}}$$

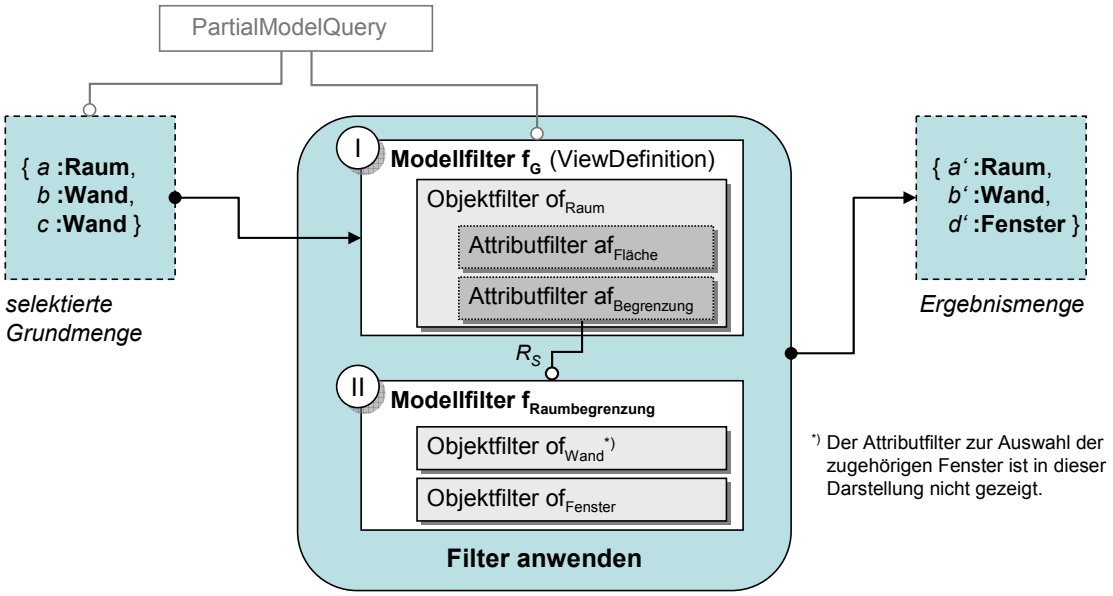
5.1.4 Anwenden eines Filters auf eine Objektmenge

In diesem Abschnitt wird die Auswertung eines Filters beschrieben. Als Grundlage dient das in Kapitel 4 vorgestellte Versionsmodell. Wird ein Filter auf eine Objektmenge angewendet, so kann zwischen der Auswahl der benötigten Objekte und dem Bestimmen der gewünschten Attribute unterschieden werden. Da ein Objekt unter Umständen durch mehrere Filter zu bewerten ist, werden in einem ersten Schritt neben der Auswahl der benötigten Objekte alle zugehörigen Filter ermittelt. In einem zweiten Schritt werden die Filter schließlich auf die ausgewählten Objekte angewendet. Die Aufteilung in zwei Schritte ist notwendig, da zur endgültigen Bewertung der Attribute die Menge der ausgewählten Objekte bekannt sein muss. Erst dadurch ist möglich, innerhalb des Teildatensatzes die Integrität der Verknüpfungen zu garantieren.

5.1.4.1 Beispiel

In dem Beispiel der Abbildung 5-3 wird die Objektmenge $\{a, b, c\}$ durch einen Filter ausgewertet. Für dieses Beispiel sind die beiden skizzierten Schritte in der Abbildung 5-4 dargestellt. Auf die gegebene Grundmenge $\{a, b, c\}$ wird der Modellfilter f_G angewendet, der einen Objektfilter für Raum-Objekte definiert. Aus der Grundmenge werden dadurch ausschließlich Raum-Objekte betrachtet, die über die beiden Attributfilter $af_{Fläche}$ und $af_{Begrenzung}$ weiter ausgewertet werden. Im ersten Schritt wird zunächst die Definition für das Attribut *Begrenzung* betrachtet, da hierüber eine Verknüpfung zu Objekten beschrieben ist. Durch diese Verknüpfung sind die Auswahl weiterer Objekte und die Zuweisung von Modellfiltern möglich. In dem gezeigten Beispiel wird dem Attributfilter $af_{Begrenzung}$ über eine Selektionsrelation R_S der Modellfilter $f_{Raumbegrenzung}$ zugewiesen. Nach der Definition (5.8) ist dieser Attributfilter auf alle verknüpften Objekte anzuwenden und ermöglicht dadurch die Auswahl von Objekten. Durch den Attributfilter $af_{Begrenzung}$ wird über das Raum-Objekt *a* zunächst das Wand-Objekt *b* ausgewählt. Über das Wand-Objekt *b* und den anzuwendenden Modellfilter $f_{Raumbegrenzung}$ wird schließlich das Fenster-Objekt *d* dem Teildatensatz hinzufügt. Die Menge der ausgewählten Objekte wird dadurch schrittweise erweitert. Hierbei werden aber nicht nur Attributfilter berücksichtigt, die über eine Selektionsrelation die Auswahl von Objekten ermöglichen. Vielmehr werden auch Attributfilter einbezogen, die über eine Modellfilterrelation den bereits ausgewählten Objekten einen anderen Modellfilter zuordnen.

Nachdem alle Möglichkeiten einer Objektauswahl ausgewertet wurden, wird der Teildatensatz im zweiten Schritt zusammengestellt. Als Ergebnis des ersten Schrittes stehen alle auszuwählenden Objekte mit den hierauf anzuwendenden Objektfiltern fest. Durch diese Objektfilter wird die Auswahl der gewünschten Attribute bestimmt. Um die Integrität des Teildatensatz zu gewährleisten, werden nur Verknüpfungen auf Objekte zugelassen, die zum Teildatensatz gehören. Für das Raum-Objekt *a* bedeutet dies, dass über das Attribut *Begrenzung* nur das Wand-Objekt *b* verknüpft wird und alle anderen Verknüpfungen, beispielsweise zu Decken-Objekten, entfernt werden. Sind für ein Objekt unterschiedliche Filter definiert, so wird der Informationsbedarf jedes einzelnen Filters erfüllt. Ein Attribut wird also nur dann aus dem Teildatensatz entfernt, wenn die Verwendung des Attributs in keinem der zugeordneten Filter gefordert ist. Wird ein Objekt durch das Anwenden der Filter in seinem Informationsgehalt reduziert, so wird ein neues Objekt mit dem geforderten Informationsgehalt angelegt. Diese Objekte beschreiben schließlich den gesuchten Teildatensatz, der dem Anwender zur weiteren Bearbeitung zur Verfügung steht. In dem Beispiel der Abbildung 5-4 wurden hierfür die drei neuen Objekte *a'*, *b'* und *d'* erzeugt und in einem neuen Planungsstand zusammengefasst.



Auswerten des Filters:

Schritt 1: Auswahl der Objekte und Zuweisung zu den anzuwendenden Filtern
{ *a* :Raum } → f_G
{ *b* :Wand, *d* :Fenster } → $f_{Raumbegrenzung}$

Schritt 2: Anwenden der Filter zur Bewertung der Attribute
→ Erzeugen von Objektkopien mit angepasstem Informationsgehalt
{ *a'* :Raum, *b'* :Wand, *d'* :Fenster }

Abbildung 5-4 Zweistufige Auswertung eines Filters

5.1.4.2 Formalisierung

Für die weiteren Betrachtungen sei eine Grundmenge G gegeben, die aus Objekt-Planungsstand-Paaren besteht und durch einen gegebenen Modellfilter f_G ausgewertet werden soll.

(5.10)

$$\text{geg.:} \quad G := \{(o,p) \in O \times P\} \quad \text{Grundmenge mit Objekt - Planungsstand - Paaren}$$
$$f_G \in PF \quad \text{Modellfilter, der auf die Grundmenge anzuwenden ist}$$

Schritt 1 – Auswahl der Objekte und Filterzuordnung

Im ersten Schritt werden die Auswahl der Objekte und deren Zuordnung zu den anzuwendenden Filtern vorgenommen. Für diesen Schritt wird eine 5-stellige Relation definiert, die für

1. das ausgewählte Objekt,
2. den hierauf anzuwendenden Modellfilter,
3. das für die Filterzuordnung verantwortliche Ausgangsobjekt,
4. den auf das Ausgangsobjekt anzuwendenden Modellfilter sowie
5. den betrachteten Planungsstand

gültig ist und als $R_{FILTERZUWEISUNG}$ bezeichnet wird.

(5.11)

$$R_{FILTERZUWEISUNG} := \left\{ \begin{array}{l} (x_1, x_2, x_3, x_4, x_5) \in O \times PF \times O \times PF \times P \quad | \\ \quad \text{Auf das Objekt } x_1 \text{ ist der Filter } x_2 \text{ anzuwenden, wenn} \\ \quad \text{das Objekt } x_3 \text{ durch den Filter } x_4 \text{ ausgewertet und} \\ \quad \text{der Planungsstand } x_5 \text{ betrachtet wird.} \end{array} \right\}$$

Für die Definition der Relation $R_{FILTERZUWEISUNG}$ wird zunächst die Problematik der Versionsverwaltung ausgeblendet. Mit dieser Vereinfachung kann die 5-stellige Relation um den betrachteten Planungsstand reduziert werden, der beim Vorhandensein mehrerer Versionen für das Auflösen der Referenzen notwendig ist. Zusätzlich wird angenommen, dass die Menge der ausgewählten Objekte $O_{AUSWAHL}$ bekannt ist. Durch das Ausblenden der Versionsproblematik sind in dieser Objektmenge keine Objekt-Planungsstand-Paare sondern nur Objekte enthalten. Mit diesen Voraussetzungen lässt sich die Relation $R_{FILTERZUWEISUNG}$ durch die Definition (5.12) rekursiv beschreiben. Die rekursive Beschreibung ist notwendig, da ausgehend von dem Objekt x_3 das ausgewählte Objekt x_1 über eine unbekannte Anzahl von Verknüpfungen erreicht wird.

(5.12)

$$R_{FILTERZUWEISUNG}(x_1, x_2, x_3, x_4) \quad :\Leftrightarrow$$

$$\exists_{y \in K \wedge w \in OF} \quad \underbrace{R_{OK}(x_3, y) \wedge w \in x_4 \wedge R_{OFK}(w, y)}_{\text{I) Für das Objekt } x_3 \text{ muss in dem Modellfilter } x_4 \text{ ein Objektfilter } w \text{ definiert sein.}} \quad \wedge$$

$$\left(\begin{array}{l} \underbrace{(x_1 = x_3 \wedge x_2 = x_4)}_{\text{II) Kriterium für das Zuweisen eines Filters}} \quad \vee \\ \\ \underbrace{\left(\begin{array}{l} \exists_{a \in x_3 \wedge b \in AF \wedge c \in M} \quad b \in w \wedge c \in y \wedge a \in c \wedge R_{AFM}(b,c) \quad \wedge \\ \left(\underbrace{\left(\exists_{e \in O \wedge f \in PF} \quad R_{AO}(a,e) \wedge R_S(b,f) \wedge R_{FILTERWEISUNG.}(x_1, x_2, e, f) \right)}_{\text{IV) Bewertung aller verknüpften Objekte und Erweiterung der Menge } O_{AUSWAHL}} \quad \vee \right. \\ \left. \underbrace{\left(\exists_{e \in O_{AUSWAHL} \wedge f \in PF} \quad R_{AO}(a,e) \wedge R_{AFPF}(b,f) \wedge R_{FILTERWEISUNG}(x_1, x_2, e, f) \right)}_{\text{V) Bewertung der verknüpften Objekte, die aus der Menge } O_{AUSWAHL} \text{ stammen}} \right) \end{array} \right)}_{\text{III) Auf ein über das Objekt } x_3 \text{ verknüpftes Objekt } e \text{ ist der Modellfilter } f \text{ anzuwenden, über den die Auswahl des Objektes } x_1 \text{ und der anzuwendende Filter } x_2 \text{ definiert sind.}} \end{array} \right)$$

Mit dem Ausdruck (5.12) wird überprüft, ob die Relation $R_{FILTERZUWEISUNG}$ für die Elemente x_1, x_2, x_3 und x_4 erfüllt ist. Durch die Bedingung I) wird zunächst kontrolliert, ob für den Modellfilter x_4 ein Objektfilter für das Objekt x_3 definiert ist. Ist dies nicht der Fall, so ist die Relation $R_{FILTERZUWEISUNG}$ nicht erfüllt, da über das Ausgangsobjekt x_3 und den Modellfilter x_4

keinerlei Zuweisungen zwischen Objekten und Modellfiltern abgeleitet werden können. Ist die Bedingung I) erfüllt, so ist für das Objekt $x_1 = x_3$ und den Modellfilter $x_2 = x_4$ auch das Kriterium der Filterzuweisung erfüllt. Dementsprechend existiert eine Relation $R_{FILTERZUWEISUNG}$, wenn die beiden Bedingungen I) und II) gelten. Ist die Bedingung II) nicht erfüllt, so sind über die Bedingung III) die mit dem Ausgangsobjekt x_3 verknüpften Objekte zu überprüfen. Für ein Attribut a des Objektes x_3 soll hierbei ein Attributfilter b existieren, der für das über a verknüpfte Objekt e einen Modellfilter f definiert. Für die weitere Bewertung ist entscheidend, über welchen Typ der Modellfilter f mit dem Attributfilter b in Beziehung steht. Hierfür kann die Modellfilterrelation R_{AFPF} oder die Selektionsrelation R_S verwendet werden, die im Fall der Modellfilterrelation die auswählbaren Objekte e auf Elemente der Menge $O_{AUSWAHL}$ beschränkt. Demgegenüber kann über die Selektionsrelation die Menge $O_{AUSWAHL}$ um Objekte aus der Menge O erweitert werden.

Aus dieser Darstellung wird deutlich, dass die Menge der ausgewählten Objekte $O_{AUSWAHL}$ über die Selektionsrelation R_S erweitert werden kann. Da die auszuwählenden Objekte zu Beginn nicht bekannt sind, muss die Menge $O_{AUSWAHL}$ ausgehend von der gegebenen Grundmenge G sowie den hierauf anzuwendenden Modellfilter f_G erst ermittelt werden (5.10). Hierfür kann die Relation $R_{FILTERZUWEISUNG}$ genutzt werden, indem die Menge $O_{AUSWAHL}$ zu Beginn mit der Grundmenge G initialisiert und iterativ erweitert wird. Wird die Menge der ausgewählten Objekte $O_{AUSWAHL}$ in einem Iterationsschritt nicht erweitert, so sind alle ausgewählten Objekte gefunden, und die Zuweisung zu Modellfiltern ist abgeschlossen. Die Iteration ist an dieser Stelle beendet. Es sei erwähnt, dass die Iteration in jedem Fall terminiert. Im ungünstigsten Fall ist jedes Objekt durch alle Modellfilter zu bewerten.

(5.13)

$$O_{AUSWAHL,0} = G \qquad (G \text{ enthält für diese Betrachtung nur Objekte und keine Objekt - Planungsstand - Paare})$$

$$O_{AUSWAHL,i+1} := \left\{ x_1 \in O \mid O_{AUSWAHL} = O_{AUSWAHL,i} \wedge \exists_{x_2 \in PF} \exists_{x_3 \in G} R_{FILTERZUWEISUNG}(x_1, x_2, x_3, f_G) \right\}$$

Abbruch, wenn $O_{AUSWAHL,i+1} = O_{AUSWAHL,i}$

Die vorangegangenen Definitionen sind für ein System ohne Versionsverwaltung gültig. Soll die Relation $R_{FILTERZUWEISUNG}$ auf das vorgeschlagene Versionsmodell erweitert werden, so muss zusätzlich berücksichtigt werden, dass

1. eine Abfrage für Objekte unterschiedlicher Planungsstände erfolgen kann,
2. die betrachteten Objekte nur Delta-Werte enthalten sowie
3. die verknüpften Objekte durch das Auflösen der Referenzen zu ermitteln sind.

Für diesen allgemeinen Fall besteht die Grundmenge G aus Objekt-Planungsstand-Paaren (5.10) und ist mit dem Filter f_G über eine 5-stellige Relation $R_{FILTERZUWEISUNG}$ (5.11) auszuwerten. In der Definition (5.12) muss hierfür die Bedingung III) erweitert werden. Das Attribut a und das verknüpfte Objekt e sind durch die nachfolgend in (5.14) aufgeführten Ersetzungen zu modifizieren.

(5.14)

III) Betrachten aller gültigen Attribute

$$a \in x_3 \quad \Rightarrow \quad a \in A_{GÜLTIG}(x_3)$$

IV) Auflösen der Referenzen

$$\exists_{e \in O} R_{AO}(a,e) \quad \Rightarrow \quad \exists_{e \in x_5 \wedge h \in O} R_{AO}(a,h) \wedge \left(e = h \vee \langle R_{VN} \rangle_t (h,e)\right)$$

V) Betrachten von Objekt - Planungsstand - Paaren

$$\exists_{e \in O_{AUSWAHL}} R_{AO}(a,e) \Rightarrow \exists_{e \in x_5 \wedge h \in O} (e,x_5) \in O_{AUSWAHL} \wedge R_{AO}(a,h) \wedge \left(e = h \vee \langle R_{VN} \rangle_t (h,e)\right)$$

$$\text{mit } O_{AUSWAHL,i+1} := \left\{ (x_1,x_5) \in O \times P \mid O_{AUSWAHL} = O_{AUSWAHL,i} \wedge \exists_{x_2 \in PF} \exists_{x_3 \in G} R_{FILTERZUWEISUNG}(x_1,x_2,x_3,f_G,x_5) \right\}$$

Über die Menge der ausgewählten Objekte $O_{AUSWAHL}$ lässt sich schließlich die Zuordnung zwischen den ausgewählten Objekt-Planungsstand-Paaren und den anzuwendenden Filtern ableiten, die in der Menge S zusammengefasst wird und das Ergebnis des ersten Schrittes darstellt (5.15).

(5.15)

$$S(G,f_G) := \left\{ (x_1,x_2,x_5) \in O \times PF \times P \mid (x_3,x_5) \in O_{AUSWAHL} \wedge R_{FILTERWEISUNG}(x_1,x_2,x_3,f_G,x_5) \right\}$$

Schritt 2 – Zusammenstellen der Teildatensätze

Für die ausgewählten Objekte werden im zweiten Schritt die benötigten Attribute bestimmt. Es wird gefordert, dass der zu bildende Teildatensatz schlüssig ist. Dies bedeutet, dass nur Objekte referenziert werden dürfen, die auch im Teildatensatz vorhanden sind. Die ausgewählten Objekte werden bei diesem Schritt gemäß der zugewiesenen Filter in ihrem Informationsgehalt reduziert. Da ein Objekt in dem Versionsmodell nicht verändert werden darf, werden neue Versionen der Objekte angelegt. Entsprechend dem Versionskonzept werden in diesen Objektversionen nur die hierfür vorzunehmenden Änderungen beschrieben. Die erzeugten Objektversionen werden in einem neuen, den Teildatensatz beschreibenden Planungsstand integriert. Jeder der in der Menge S betrachteten Planungsstände erzeugt dabei einen eigenen Teildatensatz.

Aus der Menge der ausgewählten Objekte S lässt sich für den Planungsstand x_5 der gewünschte Teildatensatz wie folgt bestimmen.

(5.16)

$$P_{TEILDATENSATZ}(S,x_5) := \left\{ o \in O \mid \exists_{x_1 \in O} \exists_{x_2 \in PF} (x_1,x_2,x_5) \in S \wedge \begin{array}{l} o = x_1, \text{ wenn kein Attribut von } x_1 \text{ zu ändern ist} \\ o = Version(x_1), \text{ wenn } x_1 \text{ geändert werden muss} \end{array} \right\}$$

Ein ausgewähltes Objekt kann als Original in den Teildatensatz integriert werden, wenn es in keinem seiner Attribute verändert werden muss. Diese Bedingung ist erfüllt, wenn zusätzlich zu allen belegten Attributen auch die hierüber verknüpften Objekte ausgewählt wurden. Wird die Bedingung (5.17) von einem ausgewählten Objekt nicht erfüllt, so muss ein Nachfolger des Objektes angelegt werden. In diesem Nachfolger werden alle Attribute aktualisiert, die aufgrund der Filterdefinition entweder zu entfernen sind[1] oder nur einen Teil der Verknüpfungen definieren.

(5.17)

$$\text{geg.:}\quad S,\quad x_5 \in P,\ x_1 \in O \ \text{ mit } \underset{x_2 \in PF}{\exists} (x_1, x_2, x_5) \in S$$

1) Für alle belegten Attribute gibt es mindestens eine Filterdefinition.

$$\underbrace{\underset{a \in A_{GÜLTIG}(x_1) \wedge a \notin A_{NB}}{\forall}}_{\text{Für alle belegten Attribute des Objektes } x_1 \text{ wird untersucht, ob ...}} \quad \underbrace{\underset{x_2 \in PF}{\exists} (x_1, x_2, x_5) \in S}_{\text{... mindestens ein Modellfilter } x_2 \text{ existiert, für den ...}} \quad \wedge$$

$$\underbrace{\underset{y \in K \wedge w \in OF}{\exists} R_{OK}(x_1, y) \wedge w \in x_2 \wedge R_{OFK}(w, y)}_{\text{... ein Objektfilter } w \text{ definiert ist, der ...}} \quad \wedge$$

$$\underbrace{\underset{b \in AF \wedge c \in M}{\exists} b \in w \wedge c \in y \wedge a \in c \wedge R_{AFM}(b, c)}_{\text{... ein Attributfilter } b \text{ für das Attribut } a \text{ enthält.}}$$

2) Es wurden alle verknüpften Objekte ausgewählt.

$$\underbrace{\underset{e \in O_{REF\_AKTUELL}(x_1, x_5)}{\forall}}_{\text{Für alle über das Objekt } x_1 \text{ verknüpften und für den betrachteten Planungsstand } x_5 \text{ gültigen Objekte } e \text{ ...}} \quad \underbrace{\underset{f \in PF}{\exists} (e, f, x_5) \in S}_{\text{... existiert ein Modellfilter } f \text{ in der Menge } S.}$$

Nach der Auswertung eines Filters steht ein neuer Planungsstand zur Verfügung, der die Konsistenzbedingungen des Versionsmodells erfüllt. In diesem Planungsstand ist der Schritt der Informationsreduzierung als eine Änderung gegenüber dem betrachteten Planungsstand dokumentiert. Aus der Definition (5.16) wird zusätzlich deutlich, dass jeder der betrachteten Planungsstände einen eigenen Teildatensatz und damit einen neuen Planungsstand erzeugt.

[1] Entfernte Attribute werden im Nachfolger als *‚nicht belegte'* Attribute gekennzeichnet.

5.2 Vergleich von Planungsständen

Der Vergleich von Planungsdaten ist notwendig, wenn die Änderungen eines Planungsschrittes unbekannt sind und nur nachträglich aus dem entstandenen Planungsstand ermittelt werden können. Für das Anlegen eines neuen Planungsstandes werden die zu ermittelnden Änderungen in einer Form benötigt, die mit dem Versionsmodell verträglich ist (*editing operations* oder *delta*). Die gesuchten Änderungen müssen daher auf das *Erzeugen*, *Ändern* und *Löschen* von Elementen[1] zurückgeführt werden. Der Vergleichsalgorithmus muss außerdem in der Lage sein, auf der Grundlage des verwendeten Meta-Modells zu arbeiten.

5.2.1 Problemstellung

Die Ausgangsituation für den betrachteten Vergleichsalgorithmus stellen zwei Planungsstände p_A und p_B dar, die miteinander zu vergleichen sind (siehe Abbildung 5-5). Unter diesen Voraussetzungen sind die Änderungen gesucht, die auf den Planungsstand p_A anzuwenden sind, um den Planungsstand p_B zu beschreiben. Da die Änderungen in dem vorgestellten Versionsmodell auf der Grundlage von Objekten beschrieben werden, wird für jedes (Delta-)Objekt der Bezug zu einem Vorgängerobjekt benötigt. Es wird daher davon ausgegangen, dass die Objekte des Planungsstandes p_B ihren Vorgängern im Planungsstand p_A zugeordnet werden können.

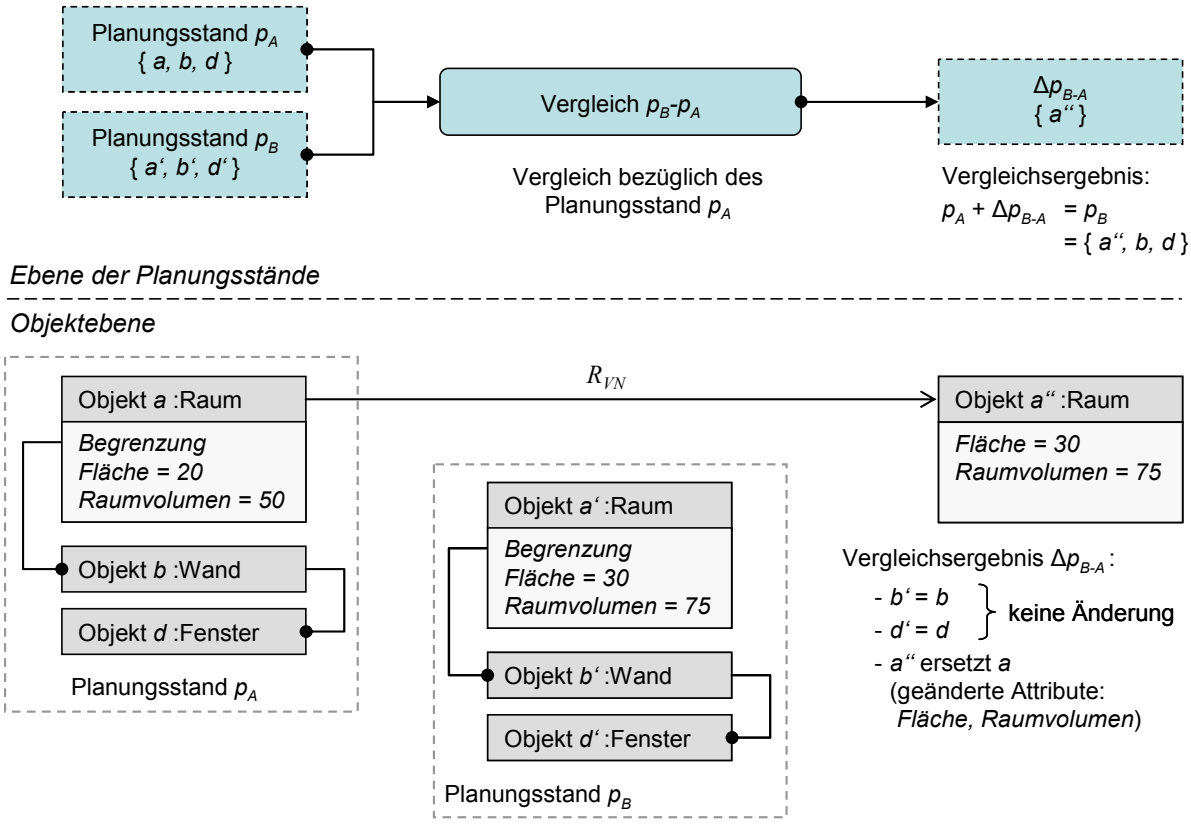


Abbildung 5-5 Prinzip des Vergleichs zweier Planungsstände

Im Sinne des objekt-orientierten Paradigmas wird die Zuordnung von Objekten über die Forderung nach Identifizierbarkeit gelöst. Jedes Objekt besitzt eine eindeutige und unveränderliche Kennung, die unabhängig vom Objektzustand seine Unterscheidbarkeit garantiert. Für die

[1] Objekte und Attribute

Anwendung in der Entwurfspraxis und die Verwaltung von Entwurfsversionen ist die Forderung nach Identifizierbarkeit jedoch mit folgenden Problemen verbunden:

1. Nicht jedes Objekt besitzt die Möglichkeit, eine zur Identifikation nutzbare Objektkennung zu beschreiben[1].
2. Änderungen im Objektgefüge, die beispielsweise durch größere Modifikationen oder Probleme der Datentransformation[2] entstehen, können nicht erfasst werden.
3. Das Verändern eines Objektes erzeugt im Versionsmodell eine neue Objektversion. Für dieses Objekt ist ebenso die Forderung der Identifizierbarkeit zu erfüllen. Jedes Objekt besitzt dadurch eine andere Kennung, die ohne weitere Vereinbarungen nicht für die Zuordnung der Objekte genutzt werden kann[3].

Der gesuchte Vergleichsalgorithmus soll unter diesen Voraussetzungen die Änderungen zwischen zwei Planungsständen ermitteln. Die Änderungen müssen dabei 1.) für die Abbildung in dem Versionsmodell geeignet sein und 2.) einen der beiden Planungsstände ohne Informationsverlust beschreiben (*correctness*, Cobena et al. 2002a). Zusätzlich wird gefordert, dass die gesuchten Änderungen unter vertretbarem Berechnungsaufwand bestimmt werden können.

Die Problemstellung eines generischen, d. h. auf dem Meta-Modell beruhenden Vergleichsalgorithmus lässt sich auf das Bestimmen von Objektpaaren, also das Bilden von Vorgänger-Nachfolger-Beziehungen, zurückführen. Durch teilweise fehlende Objektkennungen ist eine eindeutige Zuordnung der Objekte nicht immer möglich. Daher wird ein Ansatz benötigt, der unter Auswertung der verfügbaren Informationen eine sinnvolle Annäherung an die korrekte Zuordnung der Objekte erreichen kann.

Problem der Objektzuordnung

Eine Objektzuordnung kann bei fehlender Objektkennung nicht auf Korrektheit geprüft werden. Es wird daher ein Kriterium gesucht, das die Güte einer gefundenen Objektzuordnung bewerten kann. Für vergleichbare Problemstellungen wird in der Literatur als Bewertungskriterium beispielsweise der minimale Modifikationsaufwand gewählt (*minimum edit script*, Apostolico & Galil 1997). Hieraus resultiert ein np-vollständiges Optimierungsproblem (Spinner 1989, Zhang et al. 1992). Der damit verbundene Aufwand garantiert jedoch nicht, dass eine „korrekte“ Objektzuordnung gefunden wird. Dies zeigt, dass auch komplexe Verfahren nicht in der Lage sind, die tatsächlich durchgeführten Änderungen zweifelsfrei zu bestimmen. Dadurch rücken die Auswirkungen einer fehlerhaften Objektzuordnung sowie der algorithmische Aufwand in den Mittelpunkt der Untersuchungen.

---

[1] Die Unterscheidung von Objekten wird meist programmintern geregelt. Für den Datenaustausch haben diese Lösungen oft keine Bedeutung. Stattdessen werden eigenständige Konzepte verfolgt, die beispielsweise ein Attribut zur Objektkennung vorsehen. Die Forderung der Identifizierbarkeit wird dadurch nicht selten verletzt.

[2] Diese Problematik ist besonders bei Datenmodellen zu beobachten, die eine große Beschreibungsvielfalt besitzen. Inhaltlich gleichbedeutende Informationen können dadurch auf verschiedene Arten im Datenmodell beschrieben werden.

[3] Dieser Konflikt tritt beim Übergang in die Versionsverwaltung auf. Innerhalb der Versionsansätze wird das Problem der Objektzuordnung durch das Einführen einer Versionsrelation gelöst. Sieht man von Transaktionskonzepten ab, so ist der Übergang in die Versionsverwaltung undefiniert.

Fehlerhafte Objektzuordnungen

Durch fehlerhaft zugeordnete Objekte werden Änderungen ermittelt, die in dieser Form nicht durchgeführt wurden. Dadurch wird eine Bewertung des Planungsschrittes erschwert. In der Abbildung 5-6 ist als Beispiel ein Fenster-Objekt *d* dargestellt, das mit einem Fenstertyp-Objekt *e* verknüpft ist. Wird der Fenstertyp während eines Planungsschrittes geändert, so ist es ein Unterschied, ob die Änderung durch das Ersetzen eines Objektes (Variante 1) oder das Ändern von Parametern (Variante 2) ausgedrückt wird. Auch wenn der resultierende Unterschied für das Beispiel unerheblich erscheint, so kann eine fehlerhafte Objektzuordnung zu unerwünschten Nebeneffekten führen. Wird beispielsweise der Vorgänger für das Fenstertyp-Objekt *f* nicht erkannt (Variante 1), so wird für alle mit *f* verknüpften Fenster-Objekte fälschlicherweise eine Änderung des *Typ*-Attributs ermittelt.

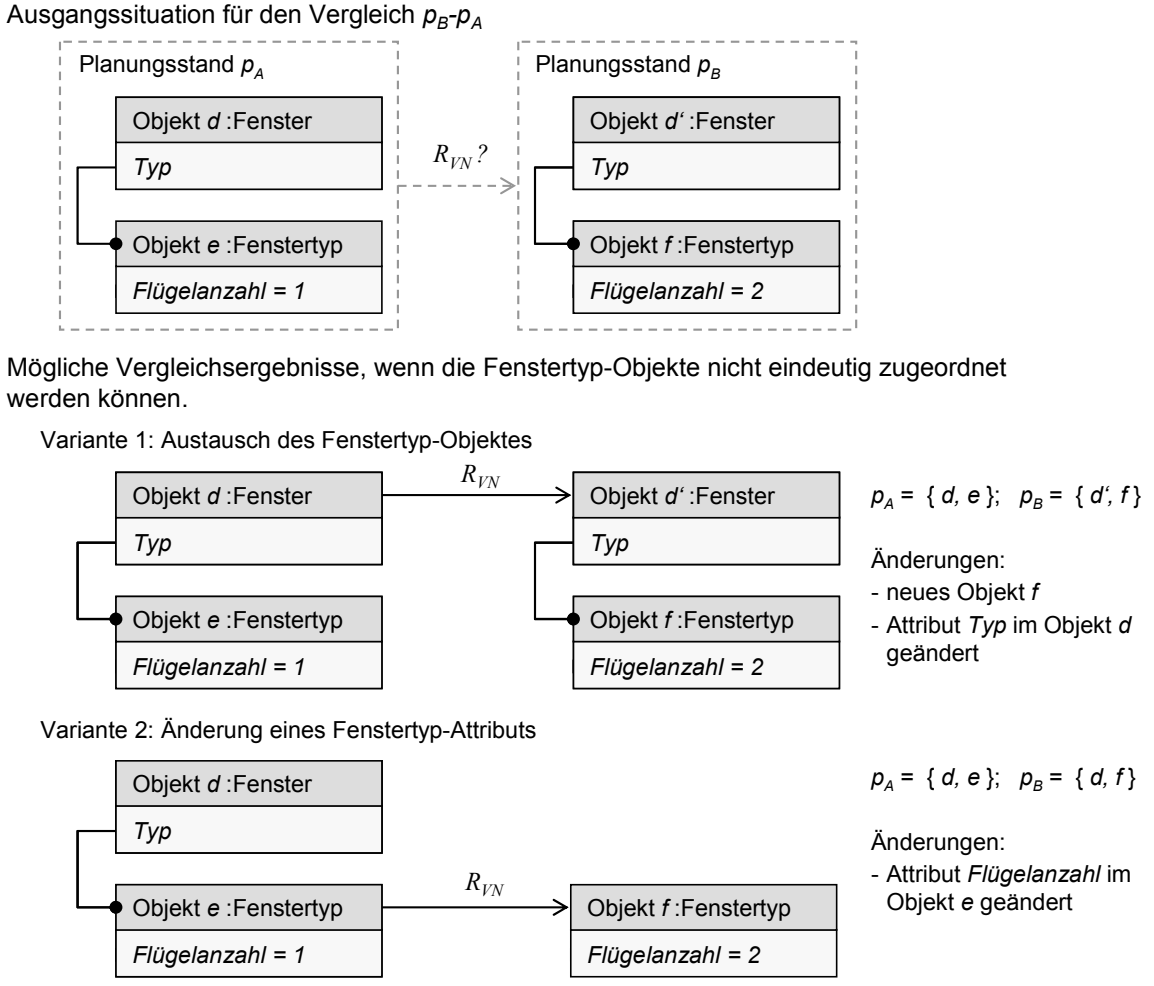


Abbildung 5-6 Fehlerhaft zugeordnete Objekte und ihre Auswirkung auf das Vergleichsergebnis

Durch fehlerhaft zugeordnete Objekte entsteht nicht unmittelbar ein Datenverlust. Probleme entstehen bei der Re-Integration geänderter Teildatensätze, die schließlich zu einem Datenverlust bzw. zu zusätzlichen Nutzerinteraktionen führen können. Unter diesen Voraussetzungen ist durch das vorgeschlagene Verfahren kein automatischer Ausgleich verlorener Informationen zu erwarten. Es wird jedoch ein Ansatz verfolgt, der Hilfestellungen zum Ausgleich von Informationsverlusten anbietet.

Der Umfang der zu erwartenden Fehler ist von vielen Faktoren abhängig und lässt sich nur grob abschätzen. Zu diesen Faktoren zählen beispielsweise die Struktur des Datenmodells, der

Anteil der eindeutig identifizierbaren Objekte, der Umfang der durchgeführten Änderungen und nicht zuletzt die Qualität der verwendeten Anwenderprogramme. Eine qualitative Bewertung des Vergleichsalgorithmus ist folglich nur für ein konkretes Anwenderszenario möglich. Daher ist es wichtig, die Qualität eines Vergleichsergebnisses einschätzen und mögliche Fehler erkennen zu können.

5.2.2 Lösungsansatz

Die Zuordnung der Objekte ist die Voraussetzung, um Änderungen ermitteln zu können. Ist das Bestimmen der Objektpaare nicht über eine eindeutige Kennung möglich, so muss der Zustand der potenziell veränderten Objekte bewertet werden. Die Komplexität der Problemstellung entsteht durch 1.) die hohe Anzahl der theoretisch möglichen Objektpaare und 2.) die Verknüpfungen mit anderen Objekten, die weit verzweigte Objektnetze und Baumstrukturen entstehen lassen.

Begrenzung des Suchraums

Eine ähnliche Problemstellung liegt bei der Objektidentifizierung und speziell der Duplikaterkennung vor. Hierbei wird die Unterscheidung in gleiche und ungleiche Objekte betrachtet (Neiling 2004). Obwohl veränderte Objekte dadurch nicht zugeordnet werden können, kann von diesen Verfahren ein Ansatz übernommen werden, der die Beschränkung von Vergleichspaaren über Vorauswahlmethoden erreicht. Auf diese Weise kann der Suchraum deutlich eingeschränkt werden.

Die Einschränkung des Suchraumes wird in dem vorgeschlagenen Vergleichsverfahren durch eine Strategie erreicht, die ausgehend von eindeutig zuordenbaren Objekten eine Bewertung der verknüpften Objekte vornimmt (Cobena et al. 2002b). Der Ansatz beruht auf der Annahme, dass Objektpaare mit einer hohen Wahrscheinlichkeit über die Auswertung der Verknüpfungen gefunden werden können. Der Zustand der Objekte stellt hierbei ein zweitrangiges Entscheidungskriterium dar. Dem Objektnetz wird damit die Fähigkeit unterstellt, falsche Objektpaare durch die Gewichtung benachbarter Objekte erkennen zu können. Für die Bewertung benachbarter Objekte wird neben der Tiefe der Nachbarschaft auch die Art der Verknüpfung berücksichtigt. Dadurch ist es möglich, für jedes Objektpaar zusätzlich eine Fehlerwahrscheinlichkeit abzuleiten. Eine gefundene Änderung kann auf diese Weise mit der Angabe über ihre Zuverlässigkeit kombiniert werden.

Iterative Zuordnung der Objekte

Um einen vertretbaren Berechnungsaufwand zu erreichen, wird eine iterative Zuordnung der Objekte vorgenommen. Ausgehend von vorhandenen Objektpaaren werden nur direkt benachbarte Objekte betrachtet. Existieren für die Zuordnung der Objekte mehrere Möglichkeiten, so wird der Objektzustand in die Entscheidung einbezogen. Hierfür wird aus den Attributwerten der Objekte eine Prüfsumme gebildet, die den Objektzustand auf einen Vergleichswert reduziert und somit ein Indiz für die Gleichheit von Objekten darstellt. Die Anzahl der Objektvergleiche kann dadurch weiter reduziert werden. Für diesen Vorteil wird auf ein sicheres Kriterium zur Erkennung gleicher Objektzustände verzichtet. Schließlich ist es durch den iterativen Ansatz möglich, zwischen einer hohen Qualität des Vergleichsergebnisses und einer maximalen Zuordnung von Objektpaaren zu wählen. Dieser Aspekt ist für die Re-Integration geänderter Teildatensätze interessant.

5.2.3 Auffinden von Objektpaaren

Die Abbildung 5-7 verdeutlicht an einem Beispiel das iterative Vorgehen, das zum Auffinden von Objektpaaren verwendet wird. Über Objektschlüssel werden zunächst die Menge der gesicherten Objektpaare GP sowie die Menge der nicht zuordenbaren Objekte UO erzeugt. Anschließend werden die Objektpaare aus der Menge GP miteinander verglichen. In dem Beispiel der Abbildung 5-7 wird das Objektpaar (*d, d'*) aus der Menge GP ausgewertet. Beide Objekte verweisen über das Attribut *Typ* auf ein Objekt aus der Menge UO, hier auf die Fenstertyp-Objekte *e* und *f*. Es wird nun mit einer gewissen Wahrscheinlichkeit angenommen, dass die Objekte *e* und *f* ebenfalls ein Objektpaar bilden. Beide Objekte werden dadurch zur Menge der vermuteten Objektpaare VP hinzugefügt. Die Trefferwahrscheinlichkeit, die mit einem vermuteten Objektpaar assoziiert wird, wird aus der Art des Zustandekommens abgeleitet. Hierbei wird zwischen den Verknüpfungstypen der 1.) Einfachreferenz sowie der 2.) geordneten und 3.) ungeordneten Mengenreferenz unterschieden. Diese Verknüpfungstypen stehen in den meisten Meta-Modellen zur Modellierung der Datenstrukturen zur Verfügung.

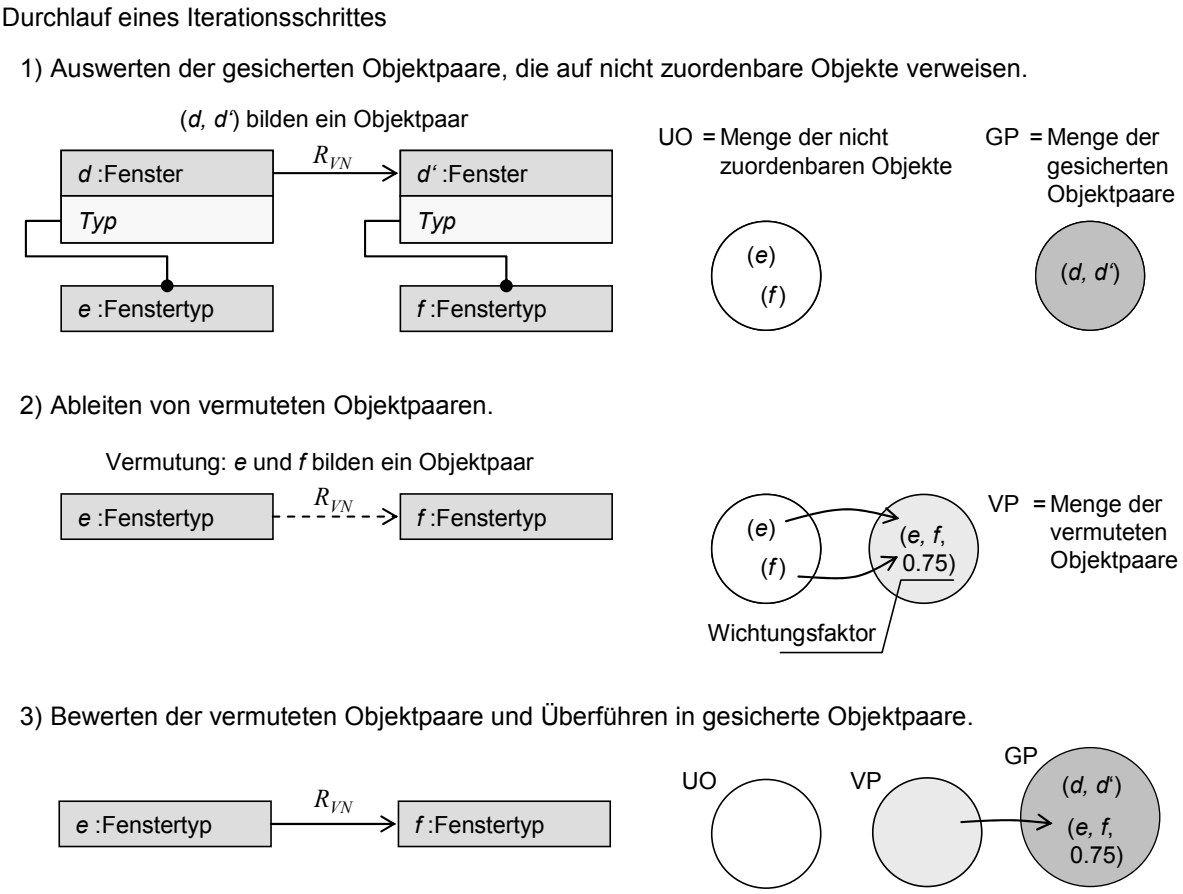


Abbildung 5-7 Prinzip der Objektzuordnung am Beispiel eines Iterationsschrittes

Nach dem Vergleich aller Objektpaare aus der Menge GP werden aus der Menge VP die Objektpaare mit der höchsten Trefferwahrscheinlichkeit in die Menge GP überführt und aus der Menge UO ausgetragen. Anschließend wird ein erneuter Durchlauf über die neuen Objektpaare der Menge GP durchgeführt. Bei den nachfolgenden Iterationsschritten wird in die Trefferwahrscheinlichkeit der gefundenen Objektpaare zusätzlich die Trefferwahrscheinlichkeit des herstellenden Objektpaares einbezogen. Aus diesem Wert kann schließlich die Zuverlässigkeit der gefundenen Objektpaare abgeschätzt werden.

Terminierung der Iteration

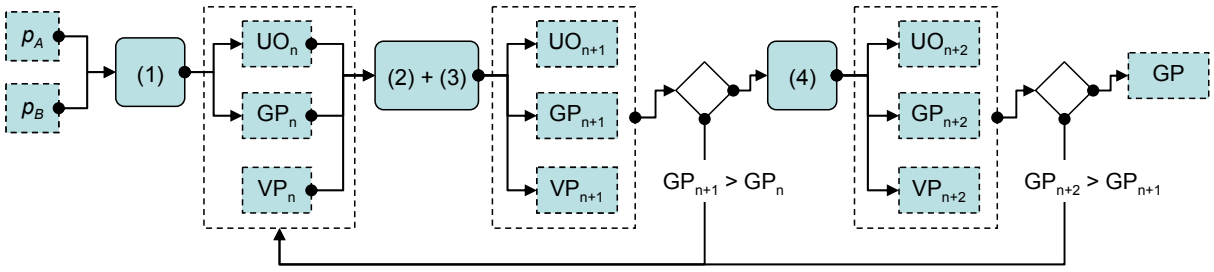
Wenn mindestens eine Objektbeziehung gegeben und *n* die Anzahl der Objekte ist, so lässt sich zeigen, dass dieser Prozess spätestens nach *n-1* Schritten terminiert. Ein vorzeitiger Abbruch ist durch das Begrenzen der zulässigen Trefferwahrscheinlichkeit realisierbar. Bei Beendigung des Algorithmus ist die Menge der nicht zuordenbaren Objekte UO unter Umständen nicht leer. Aus dieser Menge können ggf. weitere Objektpaare durch das Auswerten der Attributwerte gewonnen werden. Werden dabei weitere Objektpaare gefunden, so wird ein weiterer Iterationszyklus durchlaufen.

5.2.4 Teilprobleme der Objektzuordnung

Der beispielhaft beschriebene Ablauf des Vergleichsverfahrens ist in der Abbildung 5-8 dargestellt und lässt sich in vier unabhängige Teilprobleme zerlegen. Zu diesen Teilproblemen gehören:

1. das Initialisieren der Menge GP über das Auswerten verfügbarer Objektschlüssel,
2. die Zuordnungsregeln, die auf die verschiedenen Verknüpfungstypen anzuwenden sind,
3. das Einbeziehen des Objektzustandes als zusätzliche Entscheidungsunterstützung und
4. der Umgang mit Objekten, die nicht über das Auswerten von Verknüpfungen zugeordnet werden können.

In Ergänzung zu dem skizzierten Prinzip der Objektzuordnung wird nachfolgend detailliert auf diese vier Problemstellungen eingegangen.



(1) Initialisieren der Mengen UO und GP
(2) + (3) Bewerten der vorhandenen Objektpaare
(4) Bewerten der verbleibenden Objekte aus UO

Abbildung 5-8 Prinzipieller Ablauf des Vergleichsverfahrens

(1) Paarbildung über vorhandene Objektschlüssel

Der erste und nur einmalig durchzuführende Schritt ist die Zuordnung der Objektpaare über sogenannte Schlüssel. Für das generische Vergleichsverfahren ist hierfür die Information über die Identifizierbarkeit der zu vergleichenden Objekte notwendig. Diese Information stellt eine wichtige, modellabhängige Ergänzung dar, die das Vergleichsergebnis und die Vergleichsgeschwindigkeit entscheidend verbessern kann. Die Schlüssel sind für die Zuordnung von Objekten das einzig sichere Entscheidungskriterium und erhalten dadurch Vorrang vor allen anderen Zuordnungsregeln. Für das Identifizieren der Objekte sind verschiedene Schlüsselkonzepte bekannt (Neiling 2004).

Der *natürliche Schlüssel* wird aus einer Auswahl von Attributen gebildet. Üblicherweise werden hierfür Merkmale genutzt, die auch von Anwendern zur Identifikation verwendet werden.

Zu nennen sind z. B. die Bauteilbezeichnung über Typkürzel, Rasterposition und Stockwerk. Demgegenüber wird der *Surrogat-Schlüssel* oder auch *Objekt-ID* ausschließlich zur Identifikation in einer Datenbank verwendet und enthält keinerlei Angaben über den Zustand des Objektes. Eine Spezialform stellt das Konzept in der ISO-STEP-Norm dar, die in Part 41 (2nd Edition) ein Identifikationsobjekt definiert. Dieses Objekt enthält ein Schlüsselattribut und ist mit dem zu identifizierenden Objekt verknüpft. Eine Mischform ist der *klassifikatorische Schlüssel*, der sich aus einem klassifizierenden Teil (über Taxonomie/Klassenstruktur) und einem Surrogat zusammensetzt. All diese Formen kommen in Modellen des Bauwesens zur Anwendung. Soll ein bestimmtes Datenmodell verwendet werden, so ist das hierbei verwendete Konzept der Objektidentifikation zu unterstützen.

(2) Zuordnungsregeln für die Verknüpfungstypen

Die nachfolgenden Abbildungen zeigen an einfachen Beispielen die Auswertung der unterschiedenen Verknüpfungstypen. Diesen Verknüpfungstypen sind feste Regeln für das Bilden von Objektpaaren zugeordnet.

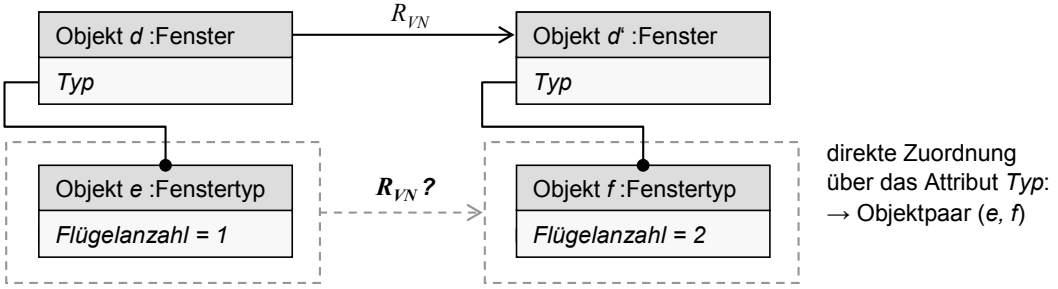


Abbildung 5-9 Zuordnen von Objekten über Einfachreferenzen

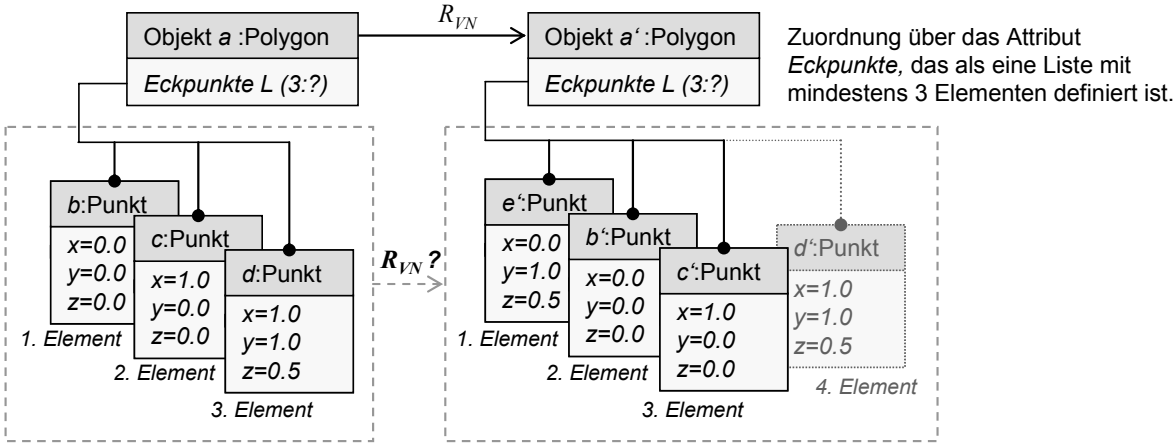
Die Einfachreferenz beschreibt eine Verknüpfung zu genau einem Objekt. Dadurch ist eine direkte Zuordnung der Objekte ohne Berücksichtigung des Objektzustandes möglich. In der Abbildung 5-9 wird über das *Typ*-Attribut der Objekte *d* und *d'* jeweils auf ein Fenstertyp-Objekt verwiesen. Mit der Annahme, dass die Zuordnung der Objekte über das gleiche Attribut zu suchen ist, ist nur eine Kombination für die beiden Objekte *e* und *f* möglich. Demgegenüber ist bei Mengenreferenzen eine direkte Zuordnung der Objekte nicht möglich. Je nach Anzahl der enthaltenen Referenzen sind einschließlich Wiederholungen unterschiedlich viele Kombinationen einer Paarbildung denkbar. Mit der Festlegung, dass nur eineindeutige Vorgänger-Nachfolger-Beziehungen zugelassen werden, wird die Anzahl der möglichen Kombinationen deutlich reduziert. Für die Auswertung einer Verknüpfung wird demzufolge ausgeschlossen, dass mehrere Objektpaare von einem Objekt gebildet werden können. Trotz dieser Vereinfachung verbleiben mehrere Möglichkeiten[1], um die Objekte der beiden Mengenreferenzen einander zuzuordnen. Für das Bilden von Objektpaaren sind folglich zusätzliche Kriterien und Vereinbarungen notwendig, die eine möglichst sinnvolle Zuordnung erlauben.

Besitzen die betrachteten Mengenreferenzen eine Ordnungsstruktur, so ist diese Information für das Bilden der Objektpaare nutzbar. Verwertbar ist entweder die absolute Position oder die Reihenfolge. Im einfachsten Fall erfolgt die Zuordnung über die Position (Abbildung

[1] Wenn n und k die Anzahl der verknüpften Objekte sind und $k < n$ gilt, dann verbleiben

$V_n^k = \frac{n!}{(n-k)!}$ Möglichkeiten für das Bilden von Objektpaaren.

5-10, Variante 1). Dies erscheint insbesondere dann sinnvoll, wenn die beiden Mengenreferenzen auf die gleiche Anzahl von Objekten verweisen. Ist dies nicht der Fall, so ist eine Zuordnung unter Beibehaltung der Reihenfolge zweckmäßig (Abbildung 5-10, Variante 2). Hierbei ist ein möglicher Versatz zwischen den Positionen der beiden Mengenreferenzen gesucht, der über den Objektzustand bestimmt wird. Für die Bewertung des Objektzustandes wird die erwähnte Prüfsumme verwendet. Nach dem Prinzip zur Bestimmung der längsten gemeinsamen Zeichenfolgen (*largest common subsequence,* Wagner & Fischer 1974) wird auf Basis der ermittelten Prüfsummen die Kombination mit der höchsten Übereinstimmung gewählt. Weitere Hinweise auf die Zuordnung der Objekte können unter Umständen aus den verbleibenden Objekten abgeleitet werden. Hierzu gehören das mehrfache Vorhandensein gleicher Objekte oder unterschiedliche Klassenzugehörigkeiten. Das Berücksichtigen dieser Kriterien kann unter Umständen eine andere Reihenfolge erzwingen.



Variante 1 - Zuordnung unter Beibehaltung der Position:

Pos.	Obj. *a*	Obj. *a‘*	Paare
1	*b*	*e‘*	(*b*, *e‘*)
2	*c*	*b‘*	(*c*, *b‘*)
3	*d*	*c‘*	(*d*, *c‘*)
4	-	*d‘*	-

Variante 2 - Zuordnung unter Beibehaltung der Reihenfolge:

Pos.	Prüfs. *a*	Prüfs. *a‘*	Objektpaare bei Versatz 0	Objektpaare bei Versatz 1
1	0	1.5	(*b*, *e‘*)	-
2	1	0	(*c*, *b‘*)	(*b*, *b‘*)
3	2.5	1	(*d*, *c‘*)	(*c*, *c‘*)
4	-	2.5	-	(*d*, *d‘*)
übereinstimmende Prüfsummen:			0	3

Abbildung 5-10 Zuordnen von Objekten über geordnete Mengenreferenzen

Ist keine Ordnungsstruktur vorhanden, so ist die Zuordnung der Objekte nur auf Basis des Objektzustandes entscheidbar (Abbildung 5-11). Sofern durch die Prüfsumme der Objekte keine Mehrdeutigkeiten entstehen, ist die Zuordnung über den Objektzustand möglich. Zusätzlich sind bestimmte Situationen denkbar, die das Auffinden von Objektpaaren vereinfachen. Analog zur geordneten Mengenreferenz können auch hier das mehrfache Vorhandensein gleicher Objekte sowie unterschiedliche Klassenzugehörigkeiten für eine Zuordnung der Objekte ausgenutzt werden. Ist in einem solchen Sonderfall nur eine Objektkombination möglich, so kann nach dem Prinzip der Einfachreferenzen eine direkte Zuordnung erfolgen. Dies tritt beispielsweise ein, wenn in beiden Mengen jeweils nur ein Objekt nicht zugeordnet werden kann.

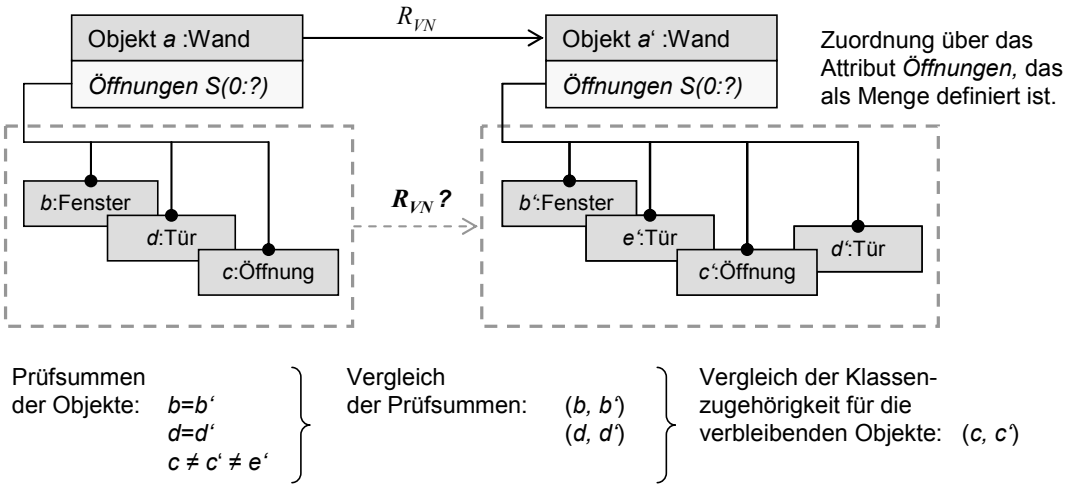


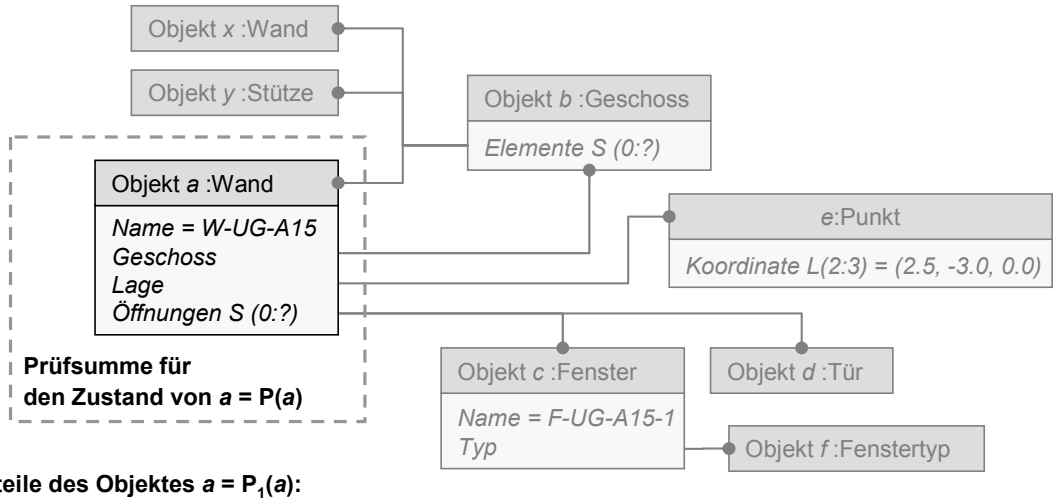
Abbildung 5-11 Zuordnen von Objekten über ungeordnete Mengenreferenzen

Jedem der vermuteten Objektpaare wird ein Wahrscheinlichkeitswert zugeordnet. Dieser Wert beschreibt das Zutreffen der Annahme und leitet sich aus den verschiedenen Verknüpfungstypen ab. Der Einfachreferenz wird hierbei die höchste und der ungeordneten Mengenreferenz die niedrigste Wahrscheinlichkeit beigemessen. Zusätzlich wird die Tiefe der Nachbarschaft berücksichtigt, die ausgehend von einem gesicherten Objektpaar zur Annahme des jeweiligen Objektpaares geführt hat. Die Wahrscheinlichkeiten der drei Verknüpfungstypen werden zur Vereinfachung derart abgestuft, dass eine zusätzliche Nachbarschaft mit einem niedriger bewerteten Verknüpfungstyp gleichgesetzt wird.

(3) Einbeziehen des Objektzustandes

Für die Zuordnung von Objekten ist die Prüfsumme ein wichtiges Entscheidungskriterium. Der Zustand eines Objektes wird hierbei in einem ganzzahligen Wert zusammengefasst. Eine gleiche Prüfsumme ist ein Indiz für gleiche Objektzustände. Unter der Annahme, dass keine Veränderungen für die betrachteten Objekte vorliegen, ist ein gleicher Objektzustand ein Hinweis auf gleiche Objekte. Der Vorteil der Prüfsumme liegt in der einmaligen Ermittlung des Objektzustandes. Andernfalls wäre der Objektzustand für jedes mögliche Objektpaar erneut zu bestimmen. Bezogen auf die Größe der Vergleichsmenge skaliert der Ansatz somit linear.

Dennoch kann das Bestimmen der Prüfsumme durch das Verfolgen aller Referenzen sehr aufwendig werden. In dem vorgeschlagenen Verfahren wird daher ein zweistufiger Ansatz verfolgt, der zunächst eine Zuordnung ohne die Berücksichtigung verknüpfter Objekte versucht. Erst wenn auch hier Mehrdeutigkeiten entstehen, wird eine Prüfsumme verwendet, die die Attributwerte der verknüpften Objekte einbezieht. Für diesen Fall ist ein Abbruchkriterium erforderlich, da ein Referenzbaum grundsätzlich abzählbar unendlich sein kann. Da aber nur endlich viele Teilbäume existieren, kann bei einer wiederholten Betrachtung eines Teilbaums abgebrochen werden. Auf diese Weise werden in der Prüfsumme alle verknüpften Objekte berücksichtigt. Durch die Notwendigkeit eines Abbruchkriteriums, das sich für jedes Objekt unterscheidet, ist die Prüfsumme eines Objekt über seine Verknüpfungen immer vollständig zu bilden (Abbildung 5-12).



Anteile des Objektes $a = P_1(a)$:

aus Klassenname = f ("Wand")

aus Attributwerten = f ("Name", "W-UG-A15")

Anteile aus den verknüpften Objekten = $P_2(a)$:

= f ("Geschoss", *b*, Abbruchkriterium *a*) = f ("Geschoss") * ($P_1(b)$ + f ("Elemente", (*x, y, a*), Abbruchkr. *a*))

= f ("Lage", *e*, Abbruchkr. *a*) = f ("Lage") * ($P_1(e)$ + f ("Koordinate", ((1, 2.5), (2, -3.0), (3, 0.0)), Abbruchkr. *a*))

= f ("Öffnungen", (*c, d*), Abbruchkr. *a*) = f ("Öffnungen") * ($P_1(c)$ + f ("Typ ", *f*, Abbruchkr. *a*) + $P_1(d)$)

Prüfsumme $P(a) = P_1(a) + P_2(a)$

Abbildung 5-12 Berechnung einer Prüfsumme und die hierfür verwendeten Informationen

Die Prüfsumme wird aus den verfügbaren Informationen eines Objektes gebildet. Falls kein Schlüssel[1] vorhanden ist, werden neben den Attributwerten auch der Klassenname, die Attributnamen sowie die Position in einer geordneten Menge berücksichtigt.

(4) Umgang mit verbleibenden Objekten

Können nicht alle Objektpaare über Verknüpfungen gefunden werden, so ist auch hier eine Zuordnung über Prüfsummen anzustreben. Die dabei auftretende Problemstellung ist mit der Zuordnung über ungeordnete Mengenreferenzen vergleichbar, stellt durch die Beschaffenheit der betrachteten Objektmengen jedoch eine Verallgemeinerung der Problematik dar. Im Unterschied zur ungeordneten Mengenreferenz liegen zunächst keinerlei Informationen über die Position der betrachteten Objekte innerhalb des Objektnetzes vor. Zusätzlich ist mit größeren Objektmengen zu rechnen, die die Wahrscheinlichkeit für ein mehrfaches Vorhandensein gleicher Objektzustände erhöhen. Soll unter diesen Bedingungen eine Zuordnung von Objekten erfolgen, so ist eine zufällige Auswahl notwendig. Durch die gegenseitigen Verknüpfungen entstehen Abhängigkeiten, die bei einer zufälligen Zuordnung zu berücksichtigen sind. Bereits durch eine Objektzuordnung können die Möglichkeiten einer zufälligen Zuordnung eingeschränkt werden. Wurde bei ungeordneten Mengenreferenzen auf ein Überprüfen der Verknüpfungen verzichtet, so ist dieses Vorgehen für den allgemeinen Fall nicht zweckmäßig.

Für die Berücksichtigung gegenseitiger Verknüpfungen sind zunächst die Objekte der verbliebenen Menge UO zu bewerten. Aus dieser Menge sind alle Objekte zu ermitteln, die nicht

[1] Ein vorhandener Objektschlüssel ersetzt das Bilden einer Prüfsumme, da nicht der Objektzustand, sondern das Objekt selbst charakterisiert werden soll.

von anderen Objekten der Menge UO referenziert werden. Ein Problem entsteht auch dann, wenn durch das Fehlen einer *inversen* Beziehung die Prüfsumme nicht alle Verknüpfungen berücksichtigen kann. Für alle Objekte, die selbst nicht referenziert werden, wird eine Zuordnung über die Prüfsumme vorgenommen. Treten mehrere Möglichkeiten für das Bilden von Objektpaaren auf, so wird eine zufällige Zuordnung notwendig. Eine zufällige Zuordnung ist jedoch nur für Objekte möglich, die voneinander unabhängig sind, d. h. nicht über Verknüpfungen direkt oder indirekt miteinander verbunden sind. In dem Beispiel der Abbildung 5-13 treffen diese Voraussetzungen für die Objekte *c* und *d* bzw. *c'* und *d'* zu. Aufgrund gleicher Prüfsummen können diese Objekte wie gezeigt beliebig zugeordnet werden.

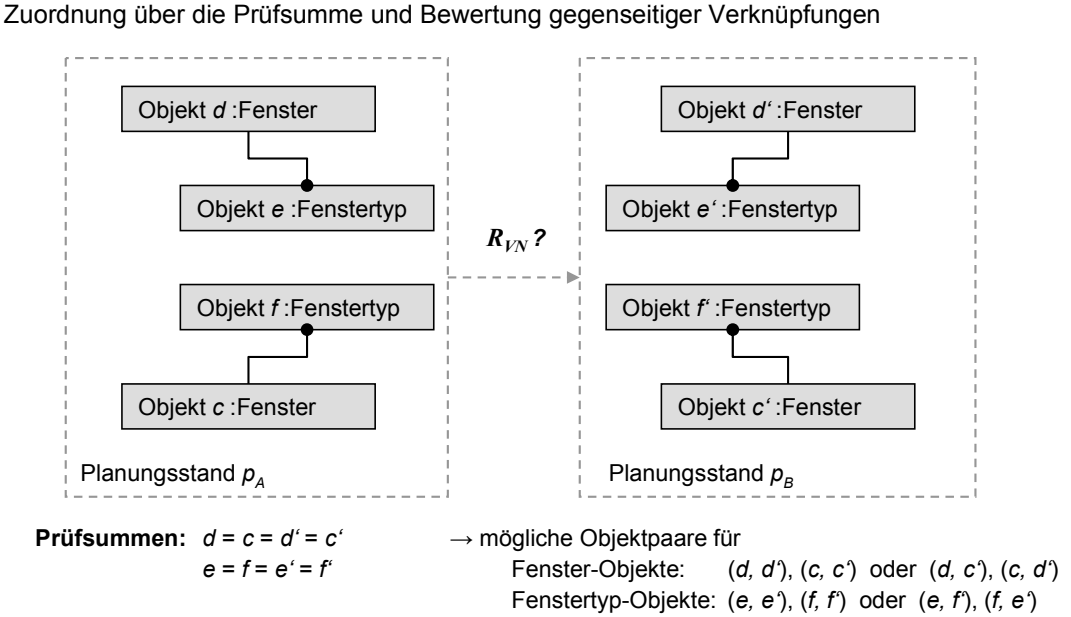


Abbildung 5-13 Zuordnen von Objekten über die Prüfsumme und Verknüpfungen

5.2.5 Überführen der vermuteten Objektpaare in gesicherte Objektpaare

Nach einem Iterationsschritt liegen in der Menge VP vermutete Objektpaare vor. Für die weitere Auswertung sind diese Objektpaare in die Menge der gesicherten Objektpaare GP zu übertragen. Hierbei werden nur Objektpaare berücksichtigt, die in der Menge VP die höchste Zutreffenswahrscheinlichkeit besitzen. Alle anderen Objektpaare verbleiben in der Menge VP und werden nach dem darauf folgenden Iterationsschritt erneut bewertet.

In dem geschilderten Verfahren kann es vorkommen, dass ein Objekt in mehreren Objektpaaren vorhanden ist. Es können also Vorgänger-Nachfolger-Beziehungen mit der Kardinalität *m:n* entstehen. Diese Multiplizität ist unter Umständen nicht gewünscht. Beispielsweise ist eine Beschränkung auf *1:1*-Beziehungen sinnvoll, wenn für das verwaltete Datenmodell die Kardinalität *m:n* ausgeschlossen werden kann oder Probleme für die weitere Verwendung der Daten entstehen. Soll die Paarbildung auf *1:1*-Beziehungen beschränkt werden, so wird für das Überführen der Objektpaare ein zusätzliches Auswahlkriterium benötigt. Hierfür kann z. B. die Häufigkeit des Zustandekommens verwendet werden. Durch solche Vereinbarungen wird das Vergleichsergebnis entscheidend mitbestimmt. Das Überführen der Objektpaare besitzt dadurch eine hohe Bedeutung.

5.2.6 Vergleich der Objektpaare

Nach dem Auffinden der Objektpaare erfolgt der Objektvergleich. Erst dadurch werden die gesuchten Unterschiede ermittelt. Für alle gefundenen Objektpaare wird bei diesem Schritt die Gleichheit korrespondierender Attribute überprüft. Die Gleichheit von Attributen ist gegeben, wenn beide Attribute den gleichen Attributzustand beschreiben. Für die verschiedenen Datentypen[1] sind die erforderlichen Kriterien der Gleichheit allgemein festgelegt. Diese Kriterien können aber auch individuell beeinflusst werden. So ist es für Zeichenketten unter Umständen sinnvoll, auf die Unterscheidung zwischen Groß- und Kleinschreibung zu verzichten. Für gebrochene Zahlen ist die Berücksichtigung von Toleranzen zweckmäßig, um Differenzen aus ungewollten Abweichungen nicht als Änderung auszuweisen.

Etwas aufwendiger ist das Überprüfen von Verknüpfungen. Zwei Verknüpfungen werden als gleich bewertet, wenn zwischen den beiden verknüpften Objekten eine Vorgänger-Nachfolger-Beziehung besteht. Zusätzlich muss nach den Konsistenzbedingungen des Versionsmodells garantiert sein, dass über die Vorgänger-Nachfolger-Beziehung und die betrachteten Planungsstände eindeutig auf den Nachfolger geschlossen werden kann. Für zwei Verknüpfungen ist die Gleichheit also nur dann erfüllt, wenn die verknüpften Objekte über die Kardinalität *1:1* oder *m:1* miteinander verbunden sind (siehe 4.3.3).

Um die Änderungen eines Objektes gegenüber seinem Vorgänger zu beschreiben, sind im Sinne des Versionsmodells alle unveränderten Attribute aus dem Objekt zu entfernen (siehe 4.3.1). Stimmt ein Objekt in allen Attributen mit seinem Vorgänger überein, so kann es durch den Vorgänger ersetzt werden. Das Ersetzen eines Objektes durch seinen Vorgänger ist aber nur dann zulässig, wenn die Konsistenzbedingungen des Versionsmodells dadurch nicht verletzt werden (siehe 4.3.2).

[1] Neben einfachen Datentypen, wie beispielsweise Zeichenketten oder den ganze und gebrochene Zahlen, ist damit auch deren Verwendung in geordneten und ungeordneten Mengen gemeint.

5.3 Re-Integration bearbeiteter Teildatensätze

Der Schritt der Re-Integration ist notwendig, wenn eine neue Entwurfsversion durch das Bearbeiten eines Teildatensatzes entstanden ist. Ist der Teildatensatz unverändert oder wurde mit einem vollständigen Datensatz[1] gearbeitet, so ist praktisch keine Re-Integration erforderlich. Für die nachfolgend betrachtete Problemstellung der Re-Integration wird vorausgesetzt, dass der überarbeitete Teildatensatz bereits in das Versionsmodell integriert ist und die vorgenommenen Änderungen bezüglich des verwendeten Teildatensatzes bekannt sind.

Durch das Verwenden eines Teildatensatzes ist ein „Informationsverlust“ entstanden. Dieser Informationsverlust ist beim Schritt der Re-Integration auszugleichen. Das Wiederherstellen entfernter Informationen kann durch die Umkehr der Teildatensatzbildung erreicht werden. Unter idealen Bedingungen wird dadurch ein vollständiger und konsistenter Datensatz erzeugt. In der Praxis sind mit diesem Verfahren aber eine Reihe von Problemen verbunden. Das formale Wiederherstellen von Information kann nämlich zu 1.) Widersprüchen mit dem Versionsmodell, 2.) fehlerhaft wiederhergestellten Informationen sowie 3.) Inkonsistenzen im Gesamtentwurf führen.

5.3.1 Probleme der Re-Integration

Liegen Änderungen im Objektgefüge vor, so können durch das formale Wiederherstellen von Informationen die Konsistenzbedingungen des Versionsmodells verletzt werden. Diese Art von Inkonsistenz ist beim Schritt der Re-Integration unmittelbar aufzulösen. Andernfalls entstehen Widersprüche im Versionsmodell, die die Datenbasis unbrauchbar machen. Da die Konsistenzbedingungen für das Versionsmodell vollständig formalisiert sind, kann diese Art von Problemen automatisch erkannt, und, wenn kein anderer Lösungsansatz verfolgt wird, nach vordefinierten Regeln formal aufgelöst werden. Im Gegensatz dazu sind fehlerhaft wiederhergestellte Informationen eine Folge falsch zugeordneter Objekte und führen zu ungewollten Inkonsistenzen im Gesamtentwurf. Diese Art der Inkonsistenz kann auch durch Änderungen der Daten verursacht werden, wobei es für die Datenverwaltung kein Unterschied ist, ob die Änderungen durch die Überarbeitung des Entwurfs oder durch technische Abläufe, wie beispielsweise die Datentransformation, entstanden sind. Die Ursache der Inkonsistenz rückt dadurch zunächst in den Hintergrund. Vielmehr stellt das Erkennen der Inkonsistenzen das vorrangige Problem dar. Hierfür wird Fachwissen benötigt, das nicht oder nur sehr begrenzt der Datenverwaltung zur Verfügung steht. Der Umgang mit diesem Problem erfordert daher die Unterstützung des Anwenders. Dies ist auch deshalb notwendig, weil das Eingliedern von Änderungen häufig mit zusätzlichen Entwurfsentscheidungen bzw. einer Anpassung anderer, während des Planungsschrittes nicht unmittelbar in Betracht gezogener Entwurfslösungen verbunden ist.

5.3.2 Ablauf der Re-Integration

Das Prinzip der Re-Integration ist in der Abbildung 5-14 am Beispiel des geänderten Wand-Objektes *a''* gezeigt. Dieses Objekt wurde vor dem Planungsschritt beim Bilden des Teildatensatzes bewusst um Informationen reduziert. Die in dem Wand-Objekt *a'* entfernten Attribute *Name* und *Lage* bzw. die aufgelösten Verknüpfungen zu *e* und *d* werden bei der Re-

[1] Wird ein vollständiger Datensatz verwendet, so wird der alte Planungsstand durch die neue Entwurfsversion vollständig ersetzt.

Integration des geänderten Objektes *a''* wiederhergestellt. Voraussetzung dafür ist, dass die bewusst entfernten Informationen über das Ausgangsobjekt *a* sowie das davon abgeleitete Objekt *a'* dokumentiert sind. Die verlorenen Informationen werden auf dieser Grundlage in einem von *a''* abgeleiteten Objekt *a'''* ergänzt. In dem gezeigten Beispiel werden das Attribut *Name* und *Lage* sowie die Verknüpfungen zu den Objekten *e* und *d* dem Objekt *a'''* hinzugefügt. Sind bei diesem Verfahren die betrachteten Objektpaare auf die Beziehung des Typs *1:1* beschränkt, so entstehen keine Konflikte mit dem Versionsmodell. In diesem Fall kann ein Vorschlag für den Planungsstand p_2 automatisch erstellt werden.

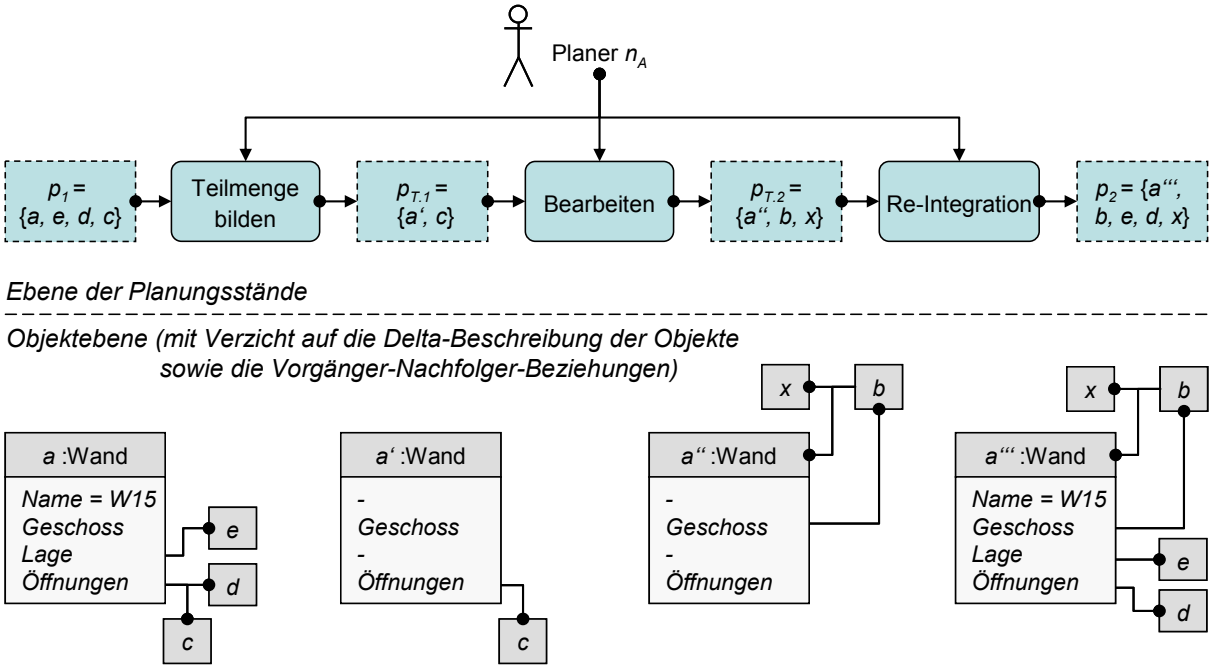


Abbildung 5-14 Re-Integration eines geänderten Objektes

Bestimmen der entfernten Informationen

Für den Schritt der Re-Integration werden zunächst alle Objekte und Attribute gesucht, die beim Erzeugen des Teildatensatzes entfernt wurden. Sind der Ausgangsdatensatz p (p_1) sowie der davon abgeleitete Teildatensatz q ($p_{T.1}$) bekannt, so können die entfernten Informationen wie in (5.18) gezeigt durch die Menge $A_{ERSETZT}(o)$, also diejenigen Attribute, die beim Bilden des Teildatensatzes durch ein geändertes Objekt o ersetzt wurden, ermittelt werden.

(5.18)

geg.: $p, q \in P$ mit p = Ausgangsdatensatz,
q = von p abgeleiteter Teildatensatz

Objekte, die aus q entfernt wurden :

$$O_{ENTFERNT}(p,q) := \left\{ x \in p \mid x \notin q \wedge \neg \exists_{y \in q} R_{VN}(x,y) \right\}$$

Objekte, die beim Bilden des Teildatensatzes um Informationen reduziert wurden :

$$O_{GEÄNDERT}(p,q) := \left\{ x \in q \mid x \notin p \right\}$$

Attribute, die in einem Objekt entfernt oder modifiziert wurden :

$$o \in O_{GEÄNDERT}(p,q)$$
$$A_{GEÄNDERT}(o) := \{a \in A \mid a \in o \wedge a \notin A_{ID}\}$$

Folgende Attribute wurden somit durch das Objekt *o* ersetzt :

$$A_{ERSETZT}(o) := \left\{ \begin{array}{l} b \in A \mid \underset{v \in O}{\exists} R_{VN}(v,o) \wedge b \in A_{GÜLTIG}(v) \quad \wedge \\ \qquad \underset{a \in A_{GEÄNDERT}(o)}{\exists} \; \underset{m \in M}{\exists} \; a \in m \wedge b \in m \end{array} \right\}$$

Integrität der wiederherzustellenden Informationen

Beim Wiederherstellen entfernter Informationen darf die Integrität der Planungsdaten nicht verletzt werden. Im Rahmen des Versionsmodells ist das Überprüfen der Integrität im wesentlichen auf die Bewertung der Verknüpfungen beschränkt. Hierfür sind die in Kapitel 4 definierten Konsistenzbedingungen zu überprüfen. Nur wenn diese Bedingungen eingehalten werden, ist ein widerspruchsfreies Verwalten der Planungsdaten gewährleistet.

Voraussetzung für das Wiederherstellen einer Verknüpfung ist, dass das verknüpfte Objekt bzw. ein davon abgeleiteter Nachfolger in dem wiederhergestellten Datensatz vorhanden ist. Zusätzlich sind die wiederherzustellenden Verknüpfungen mit den durchgeführten Änderungen auf Verträglichkeit zu überprüfen. Unter der Annahme, dass alle entfernten Objekte beim Schritt der Re-Integration in den zu vervollständigenden Datensatz aufgenommen werden, kann eine Verknüpfung unter Einhaltung der Bedingung (5.19) wiederhergestellt werden. Nach dieser Bedingung ist es zunächst ausreichend, nur die entfernten Verknüpfungen aus der Menge $A_{ENTFERNT}$ zu betrachten.

(5.19)

geg.: $p,q,s \in P$ mit $s =$ überarbeiteter Teildatensatz, der als Änderung gegenüber *q* beschrieben ist

zu überprüfende Attribute :

$$A_{ENTFERNT}(p,q) := \left\{ \begin{array}{l} a \in A \mid \underset{o \in O}{\exists} \; \left(o \in O_{GEÄNDERT}(p,q) \wedge a \in A_{ERSETZT}(o)\right) \; \vee \\ \qquad \left(o \in O_{ENTFERNT}(p,q) \wedge a \in A_{GÜLTIG}(o)\right) \end{array} \right\}$$

eine Verknüpfung $R_{AO}(a, o)$ mit $a \in A_{ENTFERNT}(p,q)$ ist wiederherstellbar, wenn

$$:\Leftrightarrow o \in O_{ENTFERNT}(p,q) \vee o \in s \vee \left(\underset{x \in s}{\exists} \langle R_{VN} \rangle_t (o,x) \wedge \underset{y \in s \wedge y \neq x}{\forall} \neg \langle R_{VN} \rangle_t (o,y) \right)$$

Erzeugen eines neuen Planungsstandes

Das Wiederherstellen entfernter Informationen erfordert das Anlegen eines neuen Planungsstandes. Im Vergleich zum überarbeiteten Teildatensatz werden hierbei die bei der Teildatensatzbildung entfernten Informationen ergänzt. Diese Informationen lassen sich in 1.) entfernte Objekte, 2.) entfernte Attribute sowie 3.) entfernte Verknüpfungen unterteilen.

Ein entferntes Objekt kann im Idealfall unverändert in einem wiederhergestellten Planungsstand verwendet werden. Dies ist möglich, wenn alle Attribute nach der Bedingung (5.19) auch in dem aktualisierten Planungsstand gültig sind. Ist die Gültigkeitsbedingung für eines der Attribute verletzt, so muss ein Nachfolger des Objektes angelegt werden. Hierbei sind alle inkonsistenten Attribute zu aktualisieren. Beim Aktualisieren eines Attributs werden alle ungültigen Verknüpfungen entfernt, die entsprechend durch ein neues bzw. *nicht belegtes* Attribut zu beschreiben sind.

Wurde ein Objekt um Attribute reduziert (siehe Bedingung (5.20)), so ist diese Änderung durch einen Nachfolger dokumentiert. Für das Wiederherstellen der Attribute ist ein weiteres Objekt mit den hierfür erforderlichen Versionsbeziehungen anzulegen. Hierüber wird der Schritt der Re-Integration dokumentiert (siehe Abbildung 5-14). Ist ein entferntes Attribut nach der Bedingung (5.19) weiterhin gültig, so kann dieses Attribut unverändert dem hierfür vorgesehenen Objekt hinzugefügt werden. Wird diese Bedingung nicht erfüllt, so muss das Attribut wie zuvor beschrieben aktualisiert werden. Da über die Integritätsbedingung nur Verknüpfungen betrachtet werden, kann ein Attribut immer dann unverändert in das „wiederhergestellte“ Objekt integriert werden, wenn das Attribut keine Verknüpfung mit anderen Objekten definiert.

(5.20)

Ein Attribut a wurde entfernt, wenn

$$:\Leftrightarrow \quad \exists_{o \in O_{GEÄNDERT}(p,q)} \quad a \in o \wedge a \in A_{NB}$$

Werden von einem Attribut nur einzelne Verknüpfungen entfernt, so sind ein Nachfolger des betrachteten Objektes und ein neues Attribut vorhanden (siehe Attribut *Öffnung* in der Abbildung 5-14). Der Schritt der Re-Integration ist auch hier durch ein neues Objekt und ein aktualisiertes Attribut zu dokumentieren. Hiefür sind zunächst die entfernten Verknüpfungen zu ermitteln und auf Gültigkeit zu überprüfen. Ist mindestens eine der entfernten Verknüpfungen gültig, so ist das Attribut zu aktualisieren.

(5.21)

Durch das Attribut a wurde eine Verknüpfung entfernt, wenn

$$:\Leftrightarrow \quad \exists_{o \in O_{GEÄNDERT}(p,q)} \quad a \in A_{GEÄNDERT}(o) \wedge a \notin A_{NB}$$

Das Attribut a hat ein Attribut b ersetzt, wenn

$$:\Leftrightarrow \quad \exists_{v \in O \wedge m \in M} \quad R_{VN}(v,o) \wedge b \in A_{GÜLTIG}(v) \wedge a \in m \wedge b \in m$$

Aus einem Attribut b sind Verknüpfungen zu folgenden Objekten entfernt worden :

$$O_{ENTFERNT}(a,b) := \{ o \in O \mid R_{AO}(b,o) \wedge \neg R_{AO}(a,o) \}$$

Auf dieser Grundlage entstehen Objekte, die den formal wiederhergestellten Planungsstand beschreiben. Bei dieser Betrachtung wurde zunächst der Idealfall angenommen, nämlich dass das Wiederherstellen entfernter Informationen keine Konflikte im Versionsmodell erzeugt. Dies trifft jedoch nur dann zu, wenn die Vorgänger-Nachfolger-Beziehungen auf den Typ *1:1* beschränkt bleiben. Da beim Erzeugen von Teildatensätzen keine Veränderungen im Objekt-

gefüge vorgenommen werden, ist die Forderung nach einer eineindeutigen Abbildung zwischen den Objektmengen der beiden betrachteten Planungsstände stets erfüllt. Widersprüche mit den Konsistenzbedingungen des Versionsmodells leiten sich damit nur aus dem Planungsschritt ab. Setzt man voraus, dass der geänderte Teildatensatz die Konsistenzbedingungen des Versionsmodells nicht verletzt, so entstehen Inkonsistenzen ausschließlich aus den Veränderungen des Objektgefüges. Hierzu gehören das Teilen, das Zusammenlegen sowie das Verändern der Klassenzugehörigkeit.

Teilen von Objekten

Ein geteiltes Objekt liegt vor, wenn zwischen den Objekten der betrachteten Teildatensätze (*q* und *s*, siehe (5.19)) eine Vorgänger-Nachfolger-Beziehung des Typs *1:n* existiert. Für diesen Fall ist zu prüfen, ob Verknüpfungen auf das geteilte Objekt wiederherzustellen sind. Eine Verknüpfung auf ein geteiltes Objekt verletzt die Eindeutigkeit von Referenzen (Bedingung 4.21) und ist folglich zu aktualisieren. Eine Inkonsistenz durch das Teilen von Objekten liegt demnach vor, wenn aus der Menge der entfernten Attribute $A_{ENTFERNT}$ eine Verknüpfung zu einem geteilten Objekt definiert ist (5.22).

(5.22)

Inkonsistenz durch das Teilen von Objekten :

$$\exists_{o \in q \wedge x,y \in s \wedge x \neq y} \; R_{VN}(o,x) \wedge R_{VN}(o,y) \quad \wedge \quad \exists_{a \in A_{ENTFERNT}(p,q) \wedge z \in O} \; R_{AO}(a, z) \wedge \left(z = o \vee \left\langle R_{VN} \right\rangle_t (z,o) \right)$$

Für das Beseitigen des unbestimmten Zustandes kommt eine der folgenden Möglichkeiten in Betracht:

- das Aufheben der Verknüpfung
- das Aktualisieren der Verknüpfung durch die Auswahl eines der beiden Objekte
- das Erweitern der Verknüpfung auf beide Objekte (sofern die Verknüpfung über eine Mengenreferenz definiert wird)

Erst durch die Auswahl einer dieser Möglichkeiten wird die Verwaltung der wieder eingegliederten Daten in dem Versionsmodell möglich.

Zusammenlegen von Objekten

Das Zusammenlegen von Objekten liegt vor, wenn zwischen den Objekten der betrachteten Teildatensätze eine Vorgänger-Nachfolger-Beziehung des Typs *m:1* existiert. Sind entfernte Informationen für ein zusammengelegtes Objekt wiederherzustellen, so stehen mehrere Objekte für die Ergänzung der gesuchten Informationen zur Verfügung. Eine Inkonsistenz durch das Zusammenlegen von Objekten liegt vor, wenn für einen der Vorgänger das Wiederherstellen von Attributen notwendig wird (5.23).

(5.23)

Inkonsistenz durch das Zusammenlegen von Objekten :

$$\exists_{x,y \in q \wedge x \neq y \wedge o \in s} \; R_{VN}(x,o) \wedge R_{VN}(y,o) \wedge \left(x \in O_{GEÄNDERT}(p,q) \vee \; y \in O_{GEÄNDERT}(p,q) \right)$$

Für die Auswahl der zu übernehmenden Informationen sind verschiedene Möglichkeiten denkbar:

- der Verzicht auf das Wiederherstellen von Informationen
- die Auswahl eines Objektes, von dem eine bestimmte Information wiederhergestellt werden soll
- das Verwenden der Informationen aller Vorgängerobjekte (sofern mehrere Informationen in dem neuen Objekt aufgenommen werden können)

Die Auswahl einer dieser Möglichkeiten kann beispielsweise vom Zustand der Vorgängerobjekte abhängig gemacht werden. Hierfür sind verschiedene Kriterien vorstellbar. Darüber hinaus kann es sinnvoll sein, die Auswahl für jedes Attribut getrennt zu entscheiden. Unabhängig von der verfolgten Strategie und der getroffenen Wahl ist mit dem Zusammenlegen von Objekten meist ein Verzicht auf Informationen verbunden. Diese Entscheidung ist bewusst zu treffen und daher nur eingeschränkt automatisierbar.

Ändern der Klassenzugehörigkeit

Das Ändern der Klassenzugehörigkeit kann zu einer Kombination der beiden zuvor genannten Problemstellungen führen. Durch eine geänderte Klassendefinition ist es einerseits möglich, dass die zuvor entfernten Informationen in dem neuen Objekt nicht mehr aufgenommen werden können. Andererseits können Verknüpfungen mit Objekten ihre Gültigkeit verlieren. Im Gegensatz zu den zuvor genannten Problemstellungen steht als Lösungsmöglichkeit aber nur das Entfernen der Informationen bzw. der Verknüpfungen zur Verfügung. Für die geänderte Klassenzugehörigkeit ist dadurch ein formaler Weg zur Auflösung der Inkonsistenzen bestimmt. Dieser Weg führt aber nicht zwangsläufig zu einer sinnvollen Lösung[1].

Auf der Grundlage des Versionsmodells ist erkennbar, ob wiederherzustellende Attribute in einem in seiner Klassenzugehörigkeit geänderten Objekt aufgenommen werden können (5.24). Das Erkennen inkonsistenter Verknüpfungen ist demgegenüber nicht möglich, da die hierfür erforderlichen Informationen in dem verwendeten Versionsmodell nicht beschreibbar sind. Diese Problematik markiert den Übergang zu Inkonsistenzen, die nur mit Fachwissen erkannt werden können.

(5.24)

Inkonsistenz durch das Ändern der Klassenzugehörigkeit :

$$\exists_{x \in q \wedge o \in s} R_{VN}(x,o) \wedge x \in O_{GEÄNDERT}(p,q) \wedge \exists_{a \in x \wedge m \in M \wedge k \in K} a \in m \wedge R_{OK}(o,k) \wedge m \notin k$$

Durch die beschriebenen Ansätze kann die Konsistenz des Versionsmodells eingehalten werden. Die Konsistenz des Gesamtentwurfs wird dadurch jedoch nicht garantiert. Durch automatisch getroffene Entscheidungen kann die Konsistenz des Gesamtentwurfs auch negativ beeinflusst werden. Hieraus entstandene Inkonsistenzen sind beim Schritt der Konsistenzsicherung zusätzlich zu berücksichtigen. Dieser Schritt ist im übrigen auch dann notwendig, wenn ausschließlich Versionsbeziehungen des Typs *1:1* vorhanden sind.

[1] Die beschriebenen Ansätze betrachten die Inkonsistenzen separat und beschränken sich auf den Ort der Inkonsistenz. Sie sind darauf ausgerichtet, die Konsistenzbedingungen des Versionsmodells zu erfüllen. Für das Beseitigen von Inkonsistenzen ergeben sich aber bedeutend mehr Möglichkeiten.

5.3.3 Auswirkung fehlerhafter Objektpaare

Durch den in Kapitel 5.2 beschriebenen Vergleichsalgorithmus kann der Verlust der Objektkennung ausgeglichen werden. Über die Objektkennung wird die Zuordnung der Objekte und damit das Erkennen der Änderungen möglich. Dies bildet die Grundlage der vorgeschlagenen Versionsverwaltung. Das Ausgleichen verlorengegangener Objektkennungen ist aber nicht immer fehlerfrei möglich und kann dadurch zu falschen Vorgänger-Nachfolger-Beziehungen führen. In der Folge werden Änderungen falsch erkannt und erschweren die Bewertung eines Entwurfsschrittes. Falsch zugeordnete Objekte wirken sich aber auch auf die Re-Integration geänderter Teildatensätze aus. Werden Informationen von einem falsch zugeordneten Vorgänger für das Wiederherstellen genutzt, so entstehen ungewollte Inkonsistenzen im Entwurf.

Der Vergleichsalgorithmus beruht auf der Annahme, dass in den betrachteten, stark vernetzten Objektstrukturen zumindest ein Teil der Objekte identifizierbar bleibt. Identifizierbare Objekte besitzen in der Regel eine höhere Bedeutung als nicht identifizierbare Objekte. Eine Datenstruktur bzw. die individuelle Sichtweise eines Teildatensatzes trifft somit eine Aussage über wichtige Objekte. Da identifizierbare Objekte den Ausgangspunkt des Vergleichsalgorithmus darstellen, wird das Vergleichsergebnis im Sinne der beigemessenen Objektbedeutung beeinflusst (siehe Abschnitt 5.2.3). Diese Abhängigkeit ist in der Abbildung 5-15 dargestellt.

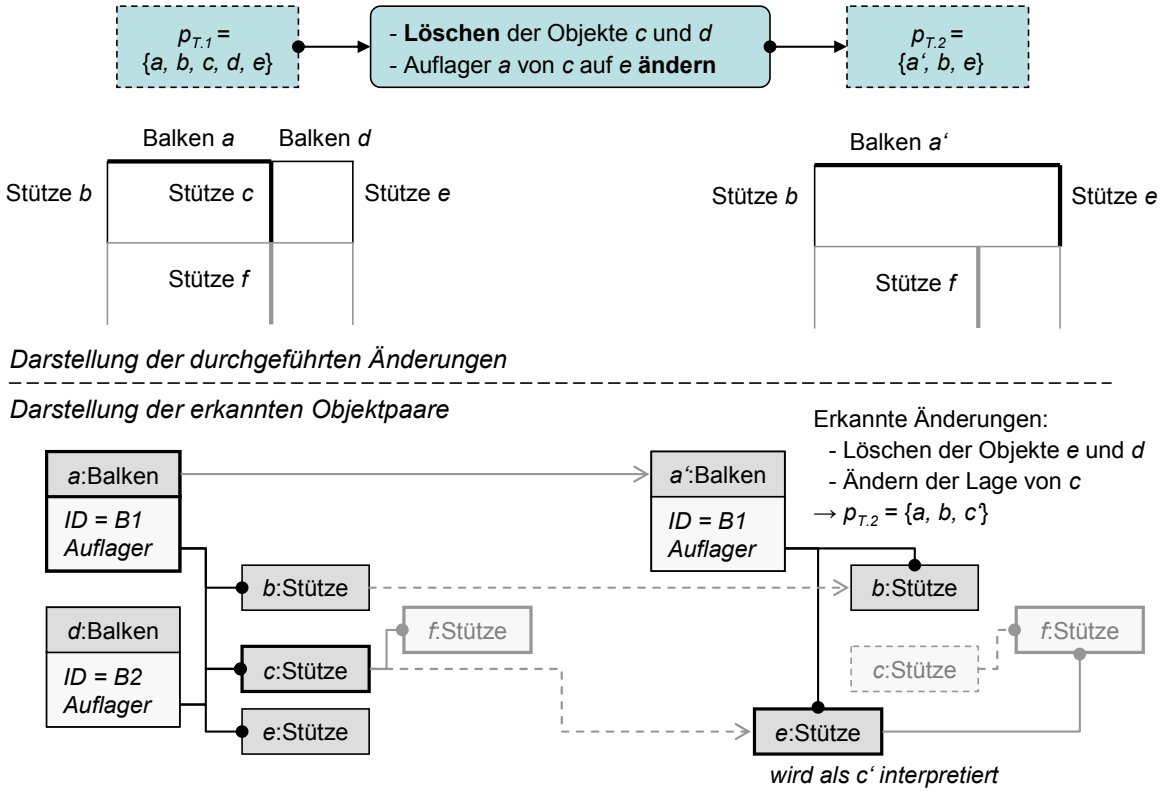


Abbildung 5-15 Auswirkungen fehlerhafter Objektpaare auf den Schritt der Re-Integration

An dem Beispiel der Abbildung 5-15 wird der Zusammenhang zwischen dem verwendeten Teildatensatz und dem zu erwartenden Vergleichsergebnis verdeutlicht. In dem dargestellten Teildatensatz $p_{T.1}$ sind die Balken-Objekte *a* und *d* sowie die Stützen-Objekte *b*, *c* und *e* enthalten. Durch einen Planungsschritt werden die Objekte *c* und *d* gelöscht. Zusätzlich wird der Balken *a* verlängert und seine rechte Auflagerung von der Stütze *c* auf die Stütze *e* geändert. Diese Änderungen sollen durch den Vergleich der beiden Planungsstände $p_{T.1}$ und $p_{T.2}$ nachträglich gefunden werden. Da in dem gezeigten Beispiel nur die Balken-Objekte über das

ID-Attribut identifizierbar sind, müssen die Stützen-Objekte *b* und *e* über die Auflagerung des Balkens *a'* zugeordnet werden. Unter diesen Voraussetzungen erzeugt der Vergleichsalgorithmus eine fehlerhafte Zuordnung zwischen den Stützen *c* (von $p_{T.1}$) und *e* (von $p_{T.2}$). Bei der anschließenden Re-Integration des geänderten Teildatensatzes $p_{T.2}$ ist die Auflagerung für die Stützen *b* und *e* wiederherzustellen. Durch den Vergleich wurden die Stützen *e* und *c* aber miteinander verwechselt. Dadurch wird der Stütze *e* fälschlicherweise die Stütze *f* als Auflagerung zugewiesen. Folglich entsteht eine Inkonsistenz im Gesamtentwurf, die aufgrund der vorgenommenen Änderungen nicht zu erwarten war.

Das Beispiel zeigt, dass eine fehlerhafte Zuordnung von Objekten zu Inkonsistenzen im Gesamtentwurf führt. Gleichzeitig wird deutlich, dass diese Inkonsistenzen in engem Zusammenhang mit dem verwendeten Teildatensatz stehen und bereits auf der Basis der ermittelten Änderungen erkennbar sind. Für das Beheben der entstandenen Inkonsistenzen sind daher zwei Strategien möglich:

- ein nachträgliches Erkennen und Korrigieren der Inkonsistenzen oder
- das Vermeiden der Inkonsistenzen, indem vor dem Schritt der Re-Integration eine Korrektur fehlerhaft zugeordneter Objekte erfolgt.

In beiden Fällen sind die gefundenen Änderungen zu bewerten. Diese Änderungen müssen mit den tatsächlich vorgenommenen Änderungen übereinstimmen und dürfen nicht zu ungewollten Inkonsistenzen im Gesamtentwurf[1] führen. Eine solche Bewertung ist aber nur durch den Anwender möglich, da nur ihm die tatsächlich durchgeführten Änderungen bekannt sind und nur er zwischen ungewollten und „beabsichtigten" Inkonsistenzen unterscheiden kann.

5.3.4 Strategien zum Ausgleich von Informationsverlusten

Durch die Probleme der Datenintegration ist ein Planungsschritt häufig mit Informationsverlusten verbunden. Mit der Dokumentation der verwendeten Planungsdaten, dem nachträglichen Vergleich von Planungsständen und schließlich der Re-Integration wird ein Ansatz zum Ausgleich der Informationsverluste verfolgt. Für diesen Ansatz beschreibt der Abschnitt 5.3.2 ein formales Vorgehen zum automatisierten Wiederherstellen verlorener Informationen. Hierdurch können Inkonsistenzen entstehen, die mit den in Abschnitt 5.3.3 beschriebenen Verfahren zu beseitigen sind. Der damit verbundene Aufwand steht aber nicht immer in einem nutzbringenden Verhältnis zu den ausgeglichenen Informationsverlusten. In solchen Fällen kann es sinnvoll sein, auf das Wiederherstellen kritischer Informationen zu verzichten und statt dessen den Anwender durch das Anzeigen der verlorenen Informationen beim manuellen Wiederherstellen zu unterstützen.

Vermeiden von Inkonsistenzen durch Akzeptanz der Informationsverluste

Das Auftreten ungewollter Inkonsistenzen kann nur dann ausgeschlossen werden, wenn die Vorgänger-Nachfolger-Beziehung ausschließlich für eindeutig identifizierbare Objekte hergestellt wird. Alle anderen Objekte verbleiben ohne Vorgänger und stellen demnach neue Objekte dar. Ein geändertes, aber nicht eindeutig identifizierbares Objekt bleibt somit auch ohne Nachfolger und gilt in dem neuen Planungsstand als gelöscht. Für solche Objekte ist das Wiederherstellen zuvor entfernter Informationen nicht mehr möglich. In der Abbildung 5-16 ist

[1] Ein fehlerhaftes Vergleichsergebnis kann nur zu ungewollten Inkonsistenzen im Gesamtentwurf führen. Die Konsistenz des Teildatensatz wird dadurch nicht beeinflusst.

als Alternative zur Abbildung 5-15 der Verzicht auf das Zuordnen der Stützen-Objekte gezeigt. Durch fehlende Vorgänger für die Stützen-Objekte *b'* und e' werden Änderungen erkannt, die im Vergleich zur fehlerhaften Objektzuordnung der Abbildung 5-15 eine Verschlechterung darstellen. Zusätzlich gehen die Verknüpfungen zu den Stützen *g* und *h* verloren, die für die Objekte *b'* und *e'* anschließend erneut zu definieren sind.

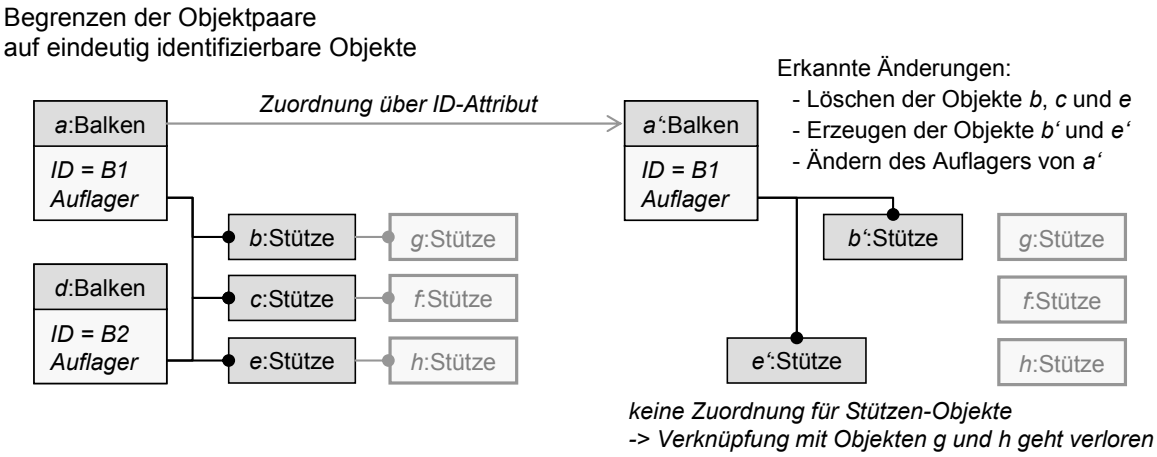


Abbildung 5-16 Verzicht auf die Zuordnung der Stützen als Alternative zur Abbildung 5-15

Kompromiss zwischen möglichen Inkonsistenzen und kalkulierbarem Informationsverlust

Das Verhindern ungewollter Inkonsistenzen ist mit dem Verzicht auf das Wiederherstellen von Informationen verbunden. In Abhängigkeit von der vorliegenden Situation ist folglich ein Kompromiss gesucht, der für den Anwender einen möglichst geringen Zusatzaufwand bei der Re-Integration geänderter Teildatensätze bedeutet. Für nicht (eindeutig) identifizierbare Objekte muss im Einzelfall zwischen einem kalkulierbaren Informationsverlust und möglichen Inkonsistenzen abgewogen werden.

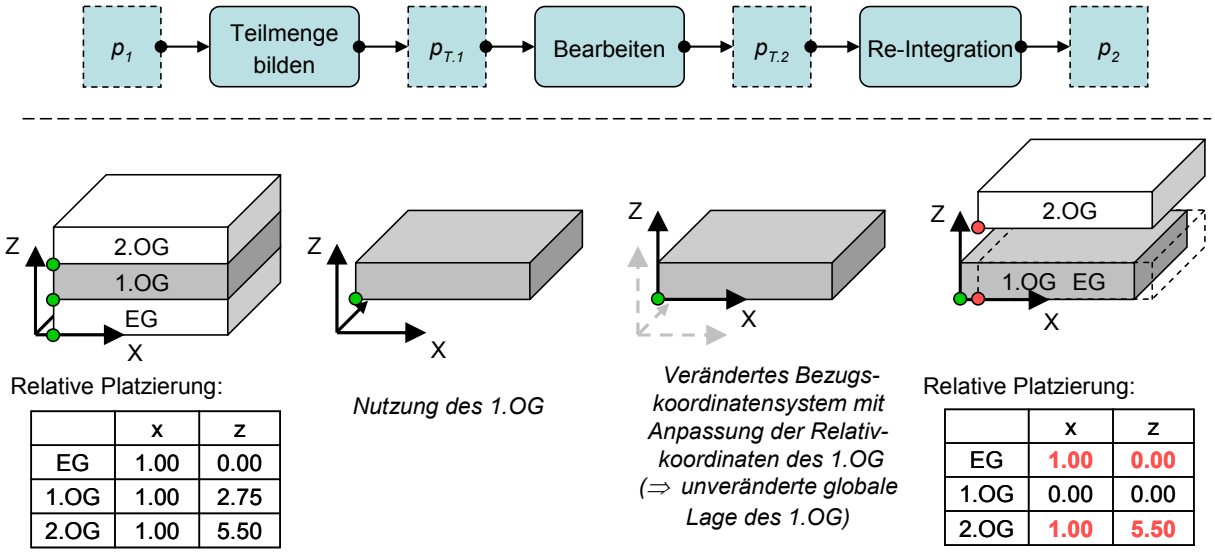
Um in der Summe einen möglichst geringen Zusatzaufwand durch das Beheben von Inkonsistenzen und das Wiederherstellen verlorener Informationen zu erreichen, kommt ein Abwägen verschiedener Kriterien in Betracht. Als Kriterien können unter anderem die Zutreffenswahrscheinlichkeit der Objektzuordnung, die Klassenzugehörigkeit sowie die Art der wiederherzustellenden Informationen verwendet werden. Durch solche Kriterien ist es außerdem vermeidbar, das Wiederherstellen von Information über das Vorhandensein der Vorgänger-Nachfolger-Beziehung steuern zu müssen. Der Verzicht auf wiederherzustellende Informationen setzt demzufolge nicht mehr den Verzicht auf Versions- und Änderungsinformationen voraus. Auf diese Weise ist es z. B. möglich, trotz vorhandener Objektzuordnung die in Abbildung 5-15 gezeigten Stützen-Objekte von der Re-Integration auszuschließen. In diesem Zusammenhang ist die Zutreffenswahrscheinlichkeit der Objektzuordnung interessant. Diese Hilfsgröße des Vergleichsalgorithmus kann als ein einfaches und vor allem generisches Entscheidungskriterium genutzt werden. Um diese Information bei der Re-Integration einbeziehen zu können, muss die Zutreffenswahrscheinlichkeit einer Versionsrelation dauerhaft in dem Versionsmodell verfügbar sein.

5.3.5 Verbleibende fachliche Widersprüche im Gesamtentwurf

Die vorangegangenen Abschnitte beschäftigen sich mit dem formalen Schritt der Re-Integration und den Problemen, die aus fehlerhaften Vergleichsergebnissen entstehen. Auch wenn der Schritt der Re-Integration im Sinne der zuvor betrachteten Problemstellungen feh-

lerfrei durchgeführt werden kann, so können die vorgenommenen Änderungen dennoch zu Inkonsistenzen im Gesamtentwurf führen. Hierzu gehören bewusst einkalkulierte fachliche Widersprüche, die aus der Überarbeitung des Entwurfs entstehen und in Zusammenarbeit mit anderen Planern in späteren Planungsschritten aufgelöst werden. Eine weitere Ursache sind technische Abläufe, die nicht auf bewusst durchgeführte fachliche Änderungen zurückzuführen sind. Besitzt eine solche, für den Anwender nicht wissentlich vorhandene Änderung Einfluss auf wiederherzustellende Objekte, so können Inkonsistenzen im Gesamtentwurf entstehen, die durch die vorgenommenen fachlichen Veränderungen nicht zu erwarten waren.

In der Abbildung 5-17 ist ein Beispiel für eine aus technischen Abläufen entstandene Inkonsistenz gezeigt. Die Ursache ist die Verschiebung eines mehrfach genutzten lokalen Bezugskoordinatensystems. Während innerhalb des bearbeiteten Teildatensatzes das relativ zu diesem Bezugskoordinatensystem platzierte erste Obergeschoss entsprechend angepasst wurde, beziehen sich die beiden anderen Geschosse weiterhin auf die Position des alten Bezugskoordinatensystems. Durch den formalen Schritt der Re-Integration wird die Lage der beiden Geschosse implizit verändert.



	x	z
EG	1.00	0.00
1.OG	1.00	2.75
2.OG	1.00	5.50

	x	z
EG	1.00	0.00
1.OG	0.00	0.00
2.OG	1.00	5.50

Abbildung 5-17 Beispiel für Widersprüche, die aus fachlich unbegründeten Änderungen entstehen.

Die in Abbildung 5-17 gezeigte Verschiebung des Bezugskoordinatensystems ist nicht zwingend eine Entscheidung des Anwenders. Eine Ursache können beispielsweise die Datentransformationen beim Import und Export sein. Analog zum Problem fehlerhafter Vergleichsergebnisse entstehen hierdurch ungewollte Inkonsistenzen, die durch den Anwender beim Schritt der Re-Integration aufzulösen sind. Durch eine Bewertung der geänderten Daten sind auch hier die unbewusst verursachten Inkonsistenzen für den Anwender meist leicht erkennbar und können entsprechend behoben werden. Für das gezeigte Beispiel ist der Widerspruch zu den wiederherzustellenden Planungsdaten aus der Lageänderung des Bezugskoordinatensystems sowie der Anpassung der Relativkoordinaten des 1. OG ersichtlich. Da bereits das Erkennen von Inkonsistenzen meist nicht vollständig automatisierbar ist, ist beim Schritt der Re-Integration eine aktive Unterstützung durch den Anwender wünschenswert. Eine solche Unterstützung ist demnach auch dann notwendig, wenn ein fehlerfreies Vergleichsergebnis garantiert werden kann. Im Idealfall wird durch den Schritt der Re-Integration ein neuer Planungsstand bereitgestellt, der nur bewusst einkalkulierte fachliche Widersprüche enthält.

5.4 Zusammenführen divergierender Planungsstände

Das Zusammenführen divergierender Planungsstände ist notwendig, um Änderungen aus parallel durchgeführten Planungsschritten in einem neuen Planungsstand zu vereinen (*Verschmelzen von Deltamengen*). Konzeptionell ist dieser Schritt mit der Problematik der Re-Integration vergleichbar, weil auch hier die Änderungen eines Planungsschrittes auf einen anderen Planungsstand zu übertragen sind. Das damit verbundene Problem möglicher Inkonsistenzen wird beim Zusammenführen jedoch um den Aspekt gegensätzlicher Entwurfsentscheidungen erweitert. Trotz dieser konzeptionellen Gemeinsamkeiten ergeben sich für die Abbildung im Versionsmodell deutliche Unterschiede. Im Vergleich zur Re-Integration sind beim Zusammenführen parallel entstandene Objektversionen zu vereinen. Hieraus leiten sich Versionsbeziehungen des Typs *m:1* ab, wobei *m* die Anzahl der zusammenzuführenden Planungsstände ist. Die nachfolgend betrachtete Methode setzt daher voraus, dass die zusammenzuführenden Planungsstände einen gemeinsamen Ursprung besitzen und die zu vereinenden Objekte über eine Versionsrelation zueinander in Beziehung stehen.

Im Rahmen dieser Arbeit wird das Zusammenführen als ein unverzichtbarer Schritt im Planungsprozess betrachtet. Dieser Schritt ist durch das Versionsmodell zu unterstützen und zu dokumentieren. Der Schwerpunkt liegt hierbei auf Ansätzen, die das Zusammenführen divergierender Planungsstände mit den Möglichkeiten einer generischen Datenverwaltung unterstützen können.

5.4.1 Formales Zusammenführen divergierender Objektversionen

Mit dem formalen Zusammenführen wird die Vereinigung parallel durchgeführter Änderungen verfolgt. Auf der Ebene des Versionsmodells ist dieser Schritt unter Wahrung der in Kapitel 4 definierten Konsistenzbedingungen zu dokumentieren. In dem nachfolgend betrachteten Szenario werden ausgehend von einem Planungsstand p_X die Änderungen eines parallel veränderten Planungsstandes p_Y übernommen (siehe Abbildung 5-18). Die beiden Planungsstände p_X und p_Y besitzen im Planungsstand p_A einen gemeinsamen Ursprung. Durch das formale Zusammenführen wird ein neuer Planungsstand p_Z erzeugt, der ein Nachfolger der Planungsstände p_X und p_Y ist.

Durch die voneinander unabhängig vorgenommenen Änderungen können verschiedene Szenarien für das Zusammenführen der beiden Planungsstände p_X und p_Y entstehen. Grundsätzlich wird ein Objekt in einem Planungsschritt entweder geändert[1], gelöscht oder unverändert beibehalten. Für die beiden Planungsstände p_X und p_Y leiten sich hieraus verschiedene Kombinationen ab, die in der Abbildung 5-19 beispielhaft dargestellt sind. Für jede dieser Kombinationen sind in der gezeigten Tabelle die Auswirkungen auf das formale Zusammenführen beschrieben. Einen Sonderfall stellt das gleichzeitige Verändern eines Objektes dar, das zusätzlich eine Bewertung auf Attributebene erfordert. Wird ein Konflikt erkannt, so ist das formale Zusammenführen nicht ohne eine Entscheidung bezüglich der verursachenden Änderungen durchführbar. Stehen zwei Änderungen in Konflikt, so gibt es prinzipiell zwei Möglichkeiten. Entweder wird nur eine oder keine der Änderungen in den gemeinsamen Planungsstand übernommen. Neben den geschilderten Kombinationen können in einem Planungsschritt auch neue Objekte erzeugt worden sein. Im Sinne der Abbildung 5-19 können

[1] Unter einer Änderung wird dabei auch das Teilen, Zusammenlegen und das Ändern der Klassenzugehörigkeit verstanden.

neue Objekte formal, also nicht zwingend (fachlich) konsistent, in den neuen Planungsstand aufgenommen werden.

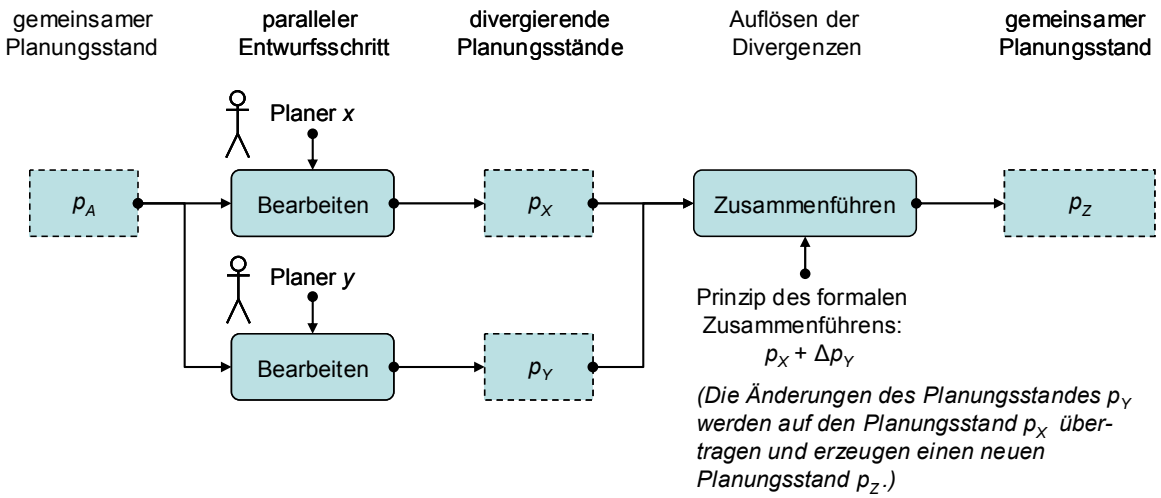


Abbildung 5-18 Formales Zusammenführen divergierender Planungsstände

Für das formale Zusammenführen der Änderungen gilt zunächst die pauschale Annahme, dass die in einem der Planungsschritte unverändert beibehaltenen Daten durch die in einem anderen Planungsschritt vorgenommenen Änderungen ersetzt werden. Aus dieser Annahme und unter Vernachlässigung möglicher Konflikte lassen sich aus den Änderungen der beiden Planungsstände p_X und p_Y nachfolgende Bedingungen für den gemeinsamen Planungsstand p_Z ableiten. Gemäß dem Beispiel aus der Abbildung 5-18 werden hierfür die Beziehungen zwischen den vier betrachteten Planungsständen p_a, p_x, p_y und p_z formuliert.

(5.25)

geg.:	$p_a,$	gemeinsamer Ursprung der geänderten Planungsstände
	$p_x, p_y \in P$	geänderte Planungsstände
ges.:	$p_z \in P$	Planungsstand, der die Änderungen zusammenführt

Für die in Abbildung 5-19 aufgeführten möglichen Kombinationen werden die Auswirkungen auf den neuen Planungsstand p_z betrachtet, die auf der Grundlage des in Kapitel 4 vorgestellten Versionsmodells beschrieben sind.

Unveränderte Objekte, die dementsprechend in p_a, p_x und p_y enthalten sind, sollen auch in dem Planungsstand p_z enthalten sein. Folglich soll gelten:

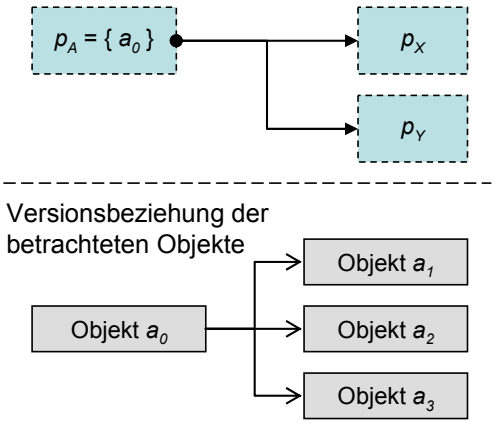
(5.26)

$$\forall_{o \in p_a \cap p_x \cap p_y} \quad o \in p_z$$

Gelöschte Objekte, die aus den beiden Planungsständen p_x und p_y entfernt wurden, sollen in dem Planungsstand p_z nicht enthalten sein. Mit der Definition (4.36) kann folgende Bedingung für die Elemente der Menge p_z formuliert werden:

(5.27)

$$\forall_{o \in O_{NICHT\_ENTHALTEN}(p_a, p_x) \cap O_{NICHT\_ENTHALTEN}(p_a, p_y)} \quad o \notin p_z$$



Kombination neuer Planungsstände und ihre lokalen Auswirkungen auf das Zusammenführen:

$p_X =$ / $p_Y =$	$\{ a_0 \}$	$\{ a_1 \}$	$\{ \}$	$\{ a_1, a_2 \}$
$\{ a_0 \}$	+	o	o	o
$\{ a_2 \}$	o	?	-	x
$\{ \}$	o	-	+	-
$\{ a_3 \}$	o	?	-	-

Zeichenerklärung:

+	Zusammenführen möglich, es sind keine Inkonsistenzen zu erwarten
o	formales Zusammenführen möglich, Inkonsistenzen können aber nicht ausgeschlossen werden
?	formales Zusammenführen muss auf Attributebene auf mögliche Konflikte überprüft werden
-	formales Zusammenführen nicht ohne Konflikt möglich
x	unzulässige Kombination

Abbildung 5-19 Parallele Nutzung eines Objektes und mögliche Kombinationen der Änderungen

Neu erzeugte Objekte, die dementsprechend keinen Vorgänger in dem Planungsstand p_a besitzen, sollen in dem Planungsstand p_z ebenso enthalten sein.

(5.28)

$$\forall_{o \in (O_{NICHT\_ENTHALTEN}(p_a, p_x) \cup O_{NICHT\_ENTHALTEN}(p_a, p_y)) - p_a} \quad o \in p_z$$

Ein Objekt, das in einem der beiden Planungsstände gelöscht und dem anderen Planungsstand nicht verändert wurde, soll als gelöscht gelten und dementsprechend in p_z nicht enthalten sein.

(5.29)

$$\forall_{o \in \left(O_{NICHT\_ENTHALTEN}(p_a, p_x) \cap O_{UNVERÄNDERT}(p_a, p_y)\right) \cup \left(O_{UNVERÄNDERT}(p_a, p_x) \cap O_{NICHT\_ENTHALTEN}(p_a, p_y)\right)} \quad o \notin p_z$$

Ein geändertes Objekt, das nur in einem der beiden Planungsstände verändert wurde, soll mit seinen Änderungen übernommen werden und somit auch im Planungsstand p_z enthalten sein.

(5.30)

$$\forall_{o \in \left(O_{VERÄNDERT}(p_a, p_x) \cap O_{UNVERÄNDERT}(p_a, p_y)\right) \cup \left(O_{UNVERÄNDERT}(p_a, p_x) \cap O_{VERÄNDERT}(p_a, p_y)\right)} \quad \exists_{n \in p_x \cup p_y} \langle R_{VN} \rangle_t (o, n) \wedge n \in p_z$$

Wurde ein Objekt in beiden Planungsständen verändert, so ist das Paar veränderter Objekte in dem Planungsstand p_z durch einen gemeinsamen Nachfolger zu ersetzen. Dieser Nachfolger vereint die divergierenden Objektversionen, indem alle geänderten Attribute zusammengeführt werden.

(5.31)

$$\forall_{o \in O_{VERÄNDERT}(p_a,p_x) \cap O_{VERÄNDERT}(p_a,p_y)} \exists_{n \in p_x \wedge m \in p_y} \langle R_{VN} \rangle_t(o,n) \wedge \langle R_{VN} \rangle_t(o,m) \exists_{s \in O} \langle R_{VN} \rangle_t(n,s) \wedge \langle R_{VN} \rangle_t(m,s) \wedge s \in p_z$$

Für den konfliktfreien Fall soll für s gelten : $\forall_{a \in A_{DIFF}(m,n) - A_{ID} - A_{GÜLTIG}(o)} a \in s$

Durch diese Bedingungen ist das formale Zusammenführen divergierender Planungsstände definiert. Sie sind jedoch nur dann anwendbar, wenn hieraus keine Konflikte entstehen. Kann eine Änderung aufgrund eines Konfliktes nicht in den gemeinsamen Planungsstand übernommen werden, so können weitere Konflikte aus den hiervon abhängigen Änderungen entstehen. Ein einzelner Konflikt kann somit größere Auswirkungen auf den Gesamtdatenbestand besitzen, die beim formalen Zusammenführen insgesamt zu beseitigen sind. Die Änderungen des Planungsstandes p_y werden daher nur übernommen, wenn sie nicht in Konflikt mit den Änderungen des Planungsstandes p_x stehen (siehe Abbildung 5-18). Ein Konflikt wird auf diese Weise pauschal zugunsten der Änderungen des Planungsstandes p_x beseitigt. Im Sinne eines Vorschlag-Zustimmungszyklus stellt der so entstandene neue Planungsstand p_z einen Vorschlag dar, der die Zustimmung durch die betroffenen Planer oder den Vorschlag eines alternativen Planungsstandes erfordert.

5.4.2 Erkennen von Konflikten

Ein Konflikt entsteht immer dann, wenn die Änderungen des einen Planungsstandes mit den Änderungen des anderen Planungsstandes nicht vereinbar sind. Für das Erkennen von Konflikten ist es folglich ausreichend, nur die Änderungen der beiden Planungsschritte zu bewerten. Ein Konflikt im Sinne dieser Definition tritt also nur dann auf, wenn Änderungen nicht formal zusammengeführt werden können. Grundsätzlich lässt sich hierbei zwischen Konflikten auf Attributebene, Objektebene und Objektstrukturebene unterscheiden.

Konflikte auf Attributebene

Ein Konflikt auf Attributebene tritt auf, wenn widersprüchliche Änderungen an einem Attribut vorgenommen wurden und somit zwei alternative Attributzustände vorliegen. Folglich sind die veränderten Attribute eines sowohl in p_x als auch in p_y geänderten Objektes o zu überprüfen. Sind für ein Merkmal unterschiedliche Attributwerte vorhanden, so liegt für dieses Objekt ein Konflikt auf Attributebene vor. Mit den Planungsständen aus der Definition (5.25) und den nachfolgend definierten Objekten lässt sich ein solcher Konflikt durch die Auswertung der Bedingung (5.33) bestimmen.

(5.32)

$$\text{geg.:} \quad o \in p_a$$
$$n \in p_x \wedge\ m \in p_y \qquad \text{mit } \langle R_{VN} \rangle_t (o,n) \wedge \langle R_{VN} \rangle_t (o,m)$$

(5.33)

Ein Konflikt durch widersprüchliche Attributwerte existiert, wenn

$$:\Leftrightarrow \underset{a,b \in A_{DIFF}(m,n) - A_{ID} - A_{GÜLTIG}(o)}{\exists} \quad \underset{w \in M}{\exists}\ a,b \in w \wedge\ a \neq b$$

Konflikte auf Objektebene

Ein Konflikt auf Objektebene tritt auf, wenn die Änderungen an einem Objekt zu einem widersprüchlichen Objektgefüge führen. Wird ein Objekt *o* in den beiden Planungsständen p_x und p_y geändert, so leiten sich mit den in (5.25) und (5.32) gegebenen Planungsständen und Objekten in den nachfolgend betrachteten Situationen Konflikte ab.

Für das in beiden Planungsständen veränderte Objekt *o* existiert auf Objektebene ein Konflikt, wenn in einem der beiden Planungsstände das Objekt geteilt wurde. Der Konflikt begründet sich durch die Tatsache, dass formal nicht entscheidbar ist, wie Änderungen auf geteilte Objekte zu übertragen bzw. aufzuteilen sind.

(5.34)

Ein Konflikt durch geteilte Objekte existiert, wenn

$$:\Leftrightarrow \underset{r \in p_x \cup p_y}{\exists} \quad r \neq n \wedge r \neq m \wedge \langle R_{VN} \rangle_t (o,r)$$

Ein Konflikt auf Objektebene tritt auf, wenn ein Objekt *o* in einem der beiden Planungsschritte gelöscht und zugleich in dem anderen Planungsschritt verändert wurde. Beide Änderungen stehen im Konflikt, weil sie nicht gemeinsam in einen neuen Planungsstand übertragbar sind.

(5.35)

Ein Konflikt durch gleichzeitiges Löschen und Ändern existiert, wenn

$$:\Leftrightarrow \underset{o \in p_a}{\exists} \begin{pmatrix} o \in \left(O_{NICHT\_ENTHALTEN}(p_a,p_x) \cap O_{VERÄNDERT}(p_a,p_y)\right) \cup \\ \left(O_{VERÄNDERT}(p_a,p_x) \cap O_{NICHT\_ENTHALTEN}(p_a,p_y)\right) \end{pmatrix}$$

Wurde die Klassenzugehörigkeit eines Objektes *o* in den beiden Planungsschritten so geändert, dass ein Widerspruch bezüglich der Klassenzugehörigkeit entsteht, so entsteht auch hieraus ein Konflikt auf Objektebene.

(5.36)

Ein Konflikt durch geänderte Klassenzugehörigkeit existiert, wenn

$$:\Leftrightarrow \underset{j,k,l \in K}{\exists} \quad R_{OK}(o,j) \wedge R_{OK}(n,k) \wedge R_{OK}(m,l) \wedge k \neq l \wedge k \neq j \wedge j \neq l$$

Ein Konflikt auf Objektebene entsteht auch dann, wenn in einem der beiden Planungsschritte ein Attribut des Objektes *o* verändert wurde, das durch eine Änderung der Klassenzugehörigkeit innerhalb des anderen Planungsschrittes kein gültiges Merkmal für das Objekt *o* ist.

(5.37)

Ein Konflikt durch das Ändern nutzbarer Merkmale existiert, wenn

$$:\Leftrightarrow \underset{a\in A_{DIFF}(m,n)-A_{GÜLTIG}(o)}{\exists} \underset{w\in M}{\exists} \underset{k\in K}{\exists} a\in w \wedge \left(R_{OK}(n,k)\vee R_{OK}(m,k)\right)\wedge w\notin k$$

Im Sinne der Definition entsteht ein Konflikt auch dann, wenn ein in beiden Planungsschritten überarbeitetes Objekt in einem der Planungsstände durch das Zusammenlegen mit einem Objekt aus p_a modifiziert wurde. Ein solcher Fall führt zwar nicht zwingend zu widersprüchlichen Änderungen, ist jedoch mit der in Abschnitt 5.4.1 formulierten Bedingung (5.30) nicht vereinbar[1]. Zusätzlich können Mehrdeutigkeiten aus der Bedingung (5.31) entstehen, die verschiedene Lösungen für ein vereintes Objekt erlauben. Durch die für das Zusammenlegen zunächst verallgemeinernd formulierte Konfliktbedingung (5.38) werden folglich nicht nur widersprüchliche Änderungen erfasst, sondern auch solche, die mit den in Abschnitt 5.4.1 definierten Bedingungen des formalen Zusammenführens nicht vereinbar sind.

(5.38)

Ein Konflikt durch das Zusammenlegen von Objekte existiert, wenn

$$:\Leftrightarrow \underset{r\in p_a}{\exists} r\neq o \wedge \left(\langle R_{VN}\rangle_t(r,n)\vee\langle R_{VN}\rangle_t(r,m)\right)$$

Eine Ergänzung der Bedingungen (5.30) und (5.31) erlaubt eine Präzisierung der Konfliktbedingung. Hierfür sind zunächst die Mehrdeutigkeiten aus der Bedingung (5.31) durch die Einschränkung (5.39) zu vermeiden. Durch diese Bedingung werden Verzweigungen für die in p_z zusammengeführten Objekte vermieden, die für das formale Zusammenführen grundsätzlich auszuschließen sind. Für das in Abbildung 5-20 gezeigte Beispiel wird auf diese Weise das Zusammenlegen der Objekte *o* und *q*, das eine Änderung des Planungsschrittes p_x ist, auch für das Übertragen der durch den Planungsschritt p_y durchgeführten Änderungen erzwungen.

(5.39)

Für die in p_z zusammengeführten Objekte soll gelten :

$$\underset{s,t\in p_z-(p_x\cup p_y)}{\forall} \underset{r\in(p_x\cup p_y)-p_z}{\forall} \langle R_{VN}\rangle_t(r,s)\vee\neg\langle R_{VN}\rangle_t(r,t)$$

[1] Wurden beispielsweise die Objekte *o* und *q* in dem Planungsschritt p_x zu einem Objekt *n* zusammengelegt und sind zusätzlich Änderungen aus dem Planungsstand p_y für das Objekt *o* beim Zusammenführen zu übertragen, so muss aufgrund der Bedingung (5.31) von dem Objekt *n* ein neues Objekt *s* abgeleitet werden. Für das in p_y unveränderte Objekt *q* erfüllt jedoch nur das Objekt *n* die Bedingung (5.30), die für den gezeigten Fall zwangsläufig im Widerspruch zu der Bedingung (5.31) steht.

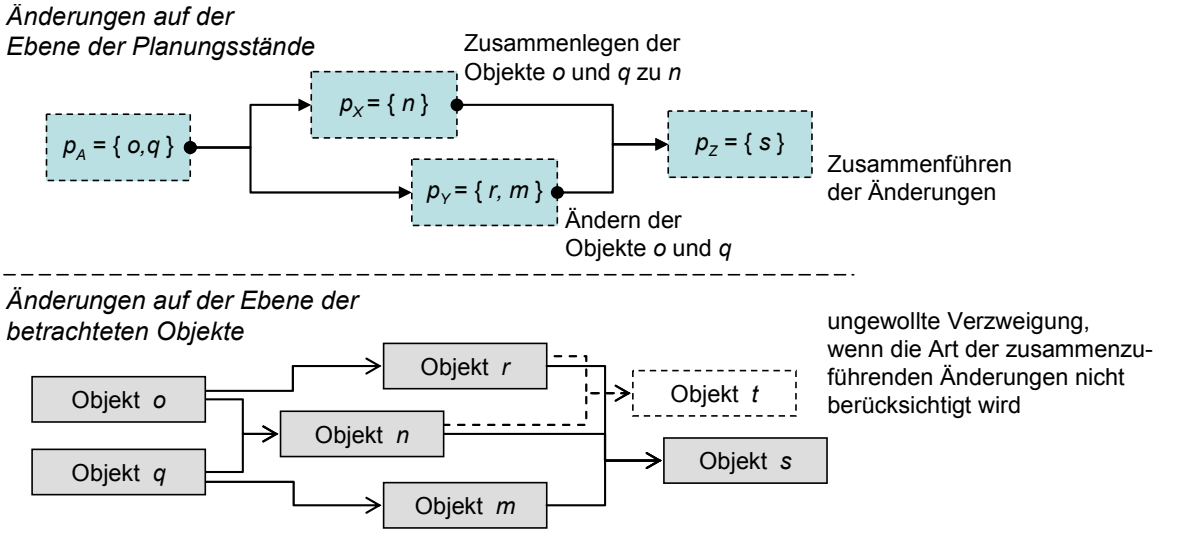


Abbildung 5-20 Gefahr einer ungewollten Verzweigung, die aus der Bedingung (5.31) entsteht.

Zusätzlich ist eine Ergänzung notwendig, um ein ungewolltes Zusammenlegen von Objekten auszuschließen. Da das Zusammenlegen nur in einem der Planungsschritte durchgeführt worden sein kann, müssen über mindestens ein Vorgängerobjekt *r* eines zusammengeführten Objektes *s* die gleichen Vorgängerobjekte in p_a erreichbar sein (5.40). Wird das Zusammenführen der geänderten Objekte unabhängig von den Vorgängerobjekten betrachtet, so kann die in Abbildung 5-21 dargestellte Situation nicht ausgeschlossen werden, in der die Objekte *o* und *q* beim Schritt des Zusammenführens in dem Objekt *s* zusammengelegt werden.

(5.40)

Für die in p_z zusammengeführten Objekte soll zusätzlich gelten :

$$\forall_{s \in p_z - (p_x \cup p_y)} \quad \exists_{r \in p_x \cup p_y} \quad \langle R_{VN} \rangle_t (r,s) \quad \forall_{o \in p_a \wedge \langle R_{VN} \rangle_t (o,s)} \quad \langle R_{VN} \rangle_t (o,r)$$

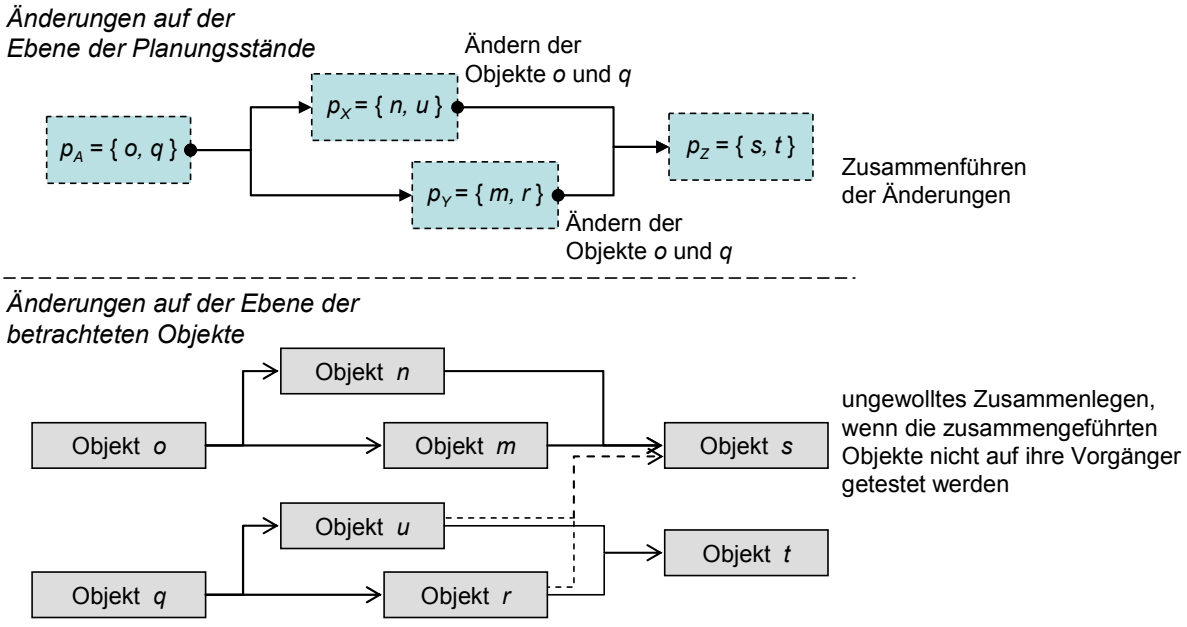


Abbildung 5-21 Gefahr des ungewollten Zusammenführens, die aus der Bedingung (5.31) entsteht.

Um die Vereinbarkeit zwischen den Bedingungen (5.30) und (5.31) zu gewährleisten, wird eine Ausnahme für die Bedingung (5.30) definiert. Diese Ausnahme gilt für den Fall, dass eines der beiden zusammengelegten Objekte *o* und *q* in dem parallel durchgeführten Planungsstand verändert wurde. Für diesen Fall wird die Bedingung derart verändert, dass ein Nachfolger des zusammengelegten Objektes *u* zu verwenden ist (5.41).

(5.41)

$$\text{geg.:}\quad o \in \begin{array}{l}\left(O_{VERÄNDERT}(p_a,p_x) \cap O_{UNVERÄNDERT}(p_a,p_y)\right) \cup \\ \left(O_{UNVERÄNDERT}(p_a,p_x) \cap O_{VERÄNDERT}(p_a,p_y)\right)\end{array}$$

$$q \in \left(O_{VERÄNDERT}(p_a,p_x) \cap O_{VERÄNDERT}(p_a,p_y)\right) \quad \exists_{u \in p_x \cup p_y} \quad \langle R_{VN}\rangle_t(o,u) \wedge \langle R_{VN}\rangle_t(q,u)$$

$$\Leftrightarrow \quad \exists_{v \in p_x \cup p_y} \langle R_{VN}\rangle_t(q,v) \quad \exists_{s \in O} \langle R_{VN}\rangle_t(u,s) \wedge \langle R_{VN}\rangle_t(v,s) \wedge s \in p_z$$

In der Abbildung 5-22 ist ein Beispiel dargestellt, das eine Anwendung der Bedingung (5.41) erfordert. Ergänzend sei erwähnt, dass die betrachteten Objekte *u* und *v* nur zu unterschiedlichen Planungsständen gehören können, da andernfalls ein Konflikt aus der Bedingung (5.34) resultiert.

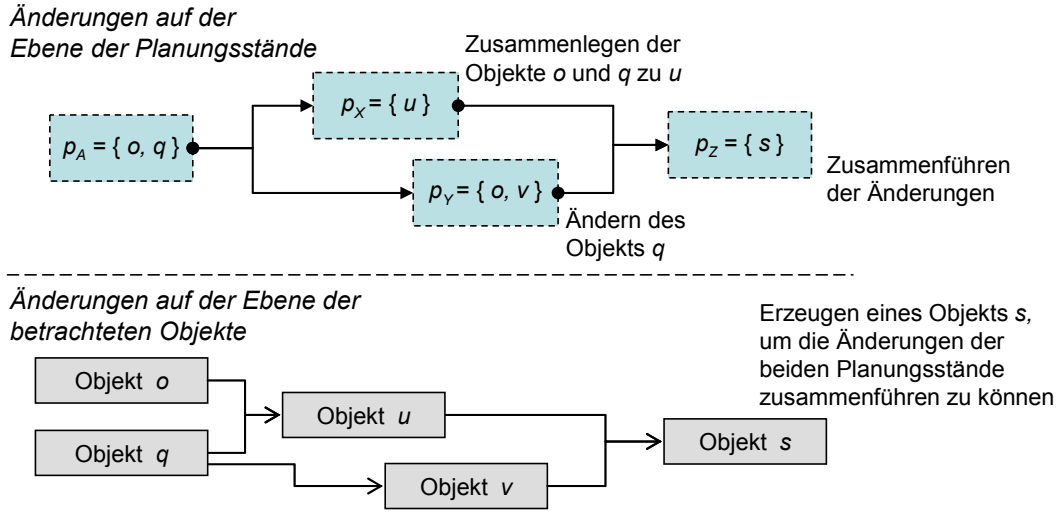


Abbildung 5-22 Situation, die eine Ausnahme von der Bedingung (5.30) erforderlich macht.

Mit dieser Präzisierung kann der Konflikt durch das Zusammenlegen auf ‚reale' Widersprüche beschränkt werden, die nur dann entstehen, wenn ein Objekt *o* in den beiden betrachteten Planungsschritten p_x und p_y mit unterschiedlichen Objekten aus p_a zusammengelegt wurde. Dieser Fall ist in der Abbildung 5-23 dargestellt.

(5.42)

Ein Konflikt durch das Zusammenlegen von Objekte existiert, wenn

$$:\Leftrightarrow \exists_{r,q \in p_a} \left(\begin{array}{l} r \neq o \wedge q \neq o \wedge r \neq q \wedge \langle R_{VN}\rangle_t(r,n) \wedge \langle R_{VN}\rangle_t(q,m) \wedge \\ \qquad \left(\neg\langle R_{VN}\rangle_t(q,n) \vee \neg\langle R_{VN}\rangle_t(r,m)\right) \end{array} \right)$$

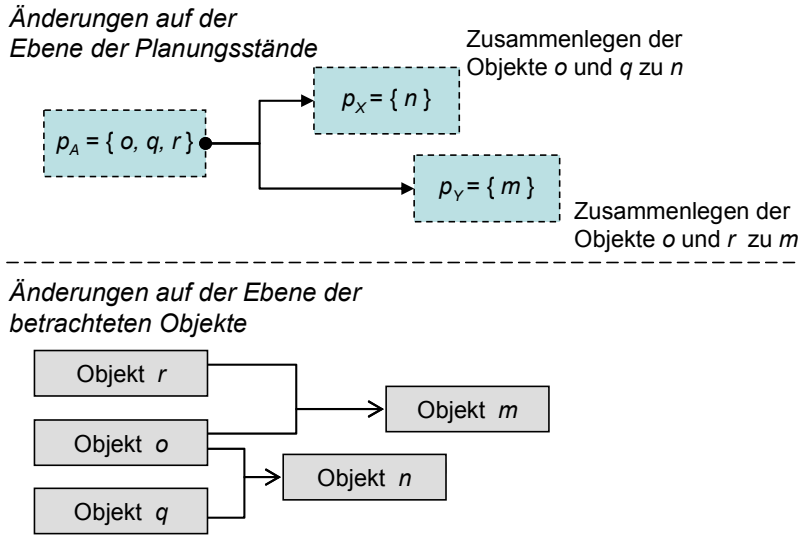


Abbildung 5-23 Konflikt durch widersprüchliches Zusammenlegen

Konflikte auf Objektstrukturebene

Ein Konflikt auf Objektstrukturebene tritt auf, wenn die Integrität der Objektverknüpfungen verletzt wird. Um Konflikte dieser Art zu ermitteln, sind die Verknüpfungen zu geänderten Objekten zu bewerten. Als Konfliktursachen sind 1.) das Erzeugen neuer Verknüpfungen sowie 2.) das Beseitigen anderer Konflikte zu unterscheiden.

Das Erzeugen von Verknüpfungen führt zu einem Konflikt, wenn das verknüpfte Objekt durch den parallel durchgeführten Planungsschritt gelöscht wurde. Ein Objektstrukturkonflikt, der aus gelöschten Objekten entsteht, kann mit den in (5.25) gegebenen Planungsständen durch die Bedingung (5.43) erkannt werden. Da formal nur die neu erzeugten Verknüpfungen zu überprüfen sind, können alle unveränderten Objekte vernachlässigt werden ($q \notin p_a$).

(5.43)

Ein Objektstrukturkonflikt durch das Löschen von Objekten existiert, wenn

$$:\Leftrightarrow \exists_{o \in \left(\begin{matrix} O_{NICHT\_ENTHALTEN}(p_a, p_x) \cup \\ O_{NICHT\_ENTHALTEN}(p_a, p_y) \end{matrix} \right) \cap p_a} \quad \exists_{q \in (p_x \cup p_y) - p_a} \quad \exists_{a \in A_{GÜLTIG}(q)} \quad R_{AO}(a,o)$$

Eine neu erzeugte Verknüpfung führt ebenso zu einem Konflikt, wenn das verknüpfte Objekt durch den parallel durchgeführten Planungsstand geteilt wurde. Ein Objektstrukturkonflikt, der aus geteilten Objekten entsteht, kann mit den Definitionen (5.25) und (5.32) durch die Bedingung (5.44) erkannt werden.

(5.44)

Ein Objektstrukturkonflikt durch das Teilen von Objekten existiert, wenn

$$:\Leftrightarrow \exists_{o \in \begin{matrix} O_{VERÄNDERT}(p_a, p_x) \cup \\ O_{VERÄNDERT}(p_a, p_y) \end{matrix}} \quad \exists_{\begin{matrix} u,v \in p_x \vee \\ u,v \in p_y \end{matrix}} \quad u \neq v \wedge \langle R_{VN} \rangle_t (o,u) \wedge \langle R_{VN} \rangle_t (o,v)$$

$$\exists_{q \in (p_x \cup p_y) - p_a} \quad \exists_{a \in A_{GÜLTIG}(q)} \quad R_{AO}(a,o)$$

In allen anderen Fällen sind die Konflikte auf Objektstrukturebene auf das Beseitigen anderer Konflikte zurückzuführen. Wenn nicht alle Änderungen in einen gemeinsamen Planungsstand übertragbar sind, so erzeugen die hiervon abhängigen Änderungen nun einen Widerspruch. Da die Abhängigkeiten der Änderungen formal nicht bekannt sind, müssen alle Änderungen des betroffenen Planungsschrittes erneut auf mögliche Konflikte untersucht werden.

5.4.3 Abbilden von Vorschlag-Zustimmungszyklen

Das formale Zusammenführen divergierender Planungsstände erzeugt einen aus der Sicht des Versionsmodells konfliktfreien Planungsstand, der nicht zwingend alle parallel vorgenommenen Änderungen vereint und darüber hinaus potenziell inkonsistent ist. Ein formal zusammengeführter Planungsstand ist daher durch die betroffenen Anwender zu überprüfen und entsprechend zu korrigieren. Dies kann beispielsweise über ein wechselseitiges Vorschlag-Zustimmungsverfahren realisiert werden, das bei Findung eines gemeinsam akzeptierten Planungsstandes beendet wird. Jeder Anwender hat hierbei die Möglichkeit, die in einem Vorschlag übernommenen Änderungen zu bewerten und gegebenenfalls einen Alternativvorschlag zu unterbreiten. Der prinzipielle Ablauf des Vorschlag-Zustimmungsverfahrens ist in der Abbildung 5-24 dargestellt, der die beiden unmittelbar betroffenen Planer *x* und *y* in den Abstimmungsprozess einbezieht.

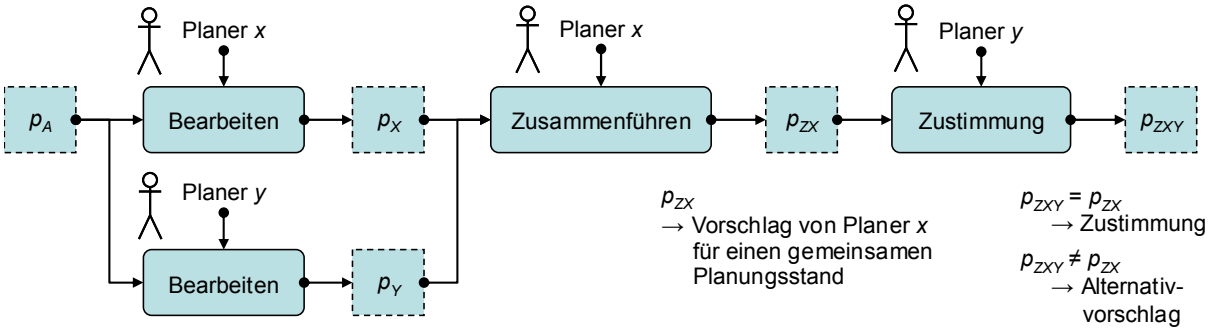


Abbildung 5-24 Vorschlag-Zustimmungsverfahren, das zum Erreichen eines gemeinsam akzeptierten Planungsstandes genutzt werden kann.

Für die Bewertung eines Vorschlags stehen alle Informationen aus dem abgebildeten Planungsprozess zur Verfügung. Hieraus ist beispielsweise erkennbar, welche der eigenen Änderungen in einen Vorschlag aufgenommen und welche abgelehnt wurden. Ebenso sind die Änderungen des parallel durchgeführten Planungsschrittes und die hieraus resultierenden Konflikte erkennbar. Auf dieser Grundlage ist eine gezielte Bewertung eines Vorschlages möglich, der formal als akzeptiert gelten kann, wenn ein Vorschlag ohne Änderungen übernommen wurde.

6 Technische Umsetzung

Das folgende Kapitel beschäftigt sich mit der Umsetzung des Lösungsansatzes in ein Softwaresystem. Für die mengentheoretische Formalisierung wird eine Implementierung vorgestellt, die mit der Programmiersprache JAVA prototypisch für das Meta-Modell EXPRESS (ISO-10303-11) umgesetzt wurde. Dadurch wird die Verwaltung verfügbarer Datenmodelle möglich, die eine praxisbezogene Validierung des Ansatzes erlauben.

Der *erste Abschnitt* vermittelt einen Überblick über die Softwareumsetzung. Im Mittelpunkt steht die Einbindung der Datenverwaltung in eine Client-Server-Architektur. Neben dem Datenserver wird ein generischer Client beschrieben, der den Zugriff auf die zentral verwalteten Planungsdaten ermöglicht. Die Bearbeitung der Planungsdaten geschieht mit verfügbaren Anwenderprogrammen, die über den Austausch von Dateien in die Architektur integriert werden. Die erarbeiteten Methoden sind im Datenserver als eigenständige Plug-ins realisiert und stellen der Datenverwaltung eine spezifische Funktion zur Verfügung. Der Aufruf der Funktionen wird schließlich durch das Durchlaufen einer langen Transaktion gesteuert.

Dieser Überblick wird im *zweiten Abschnitt* durch die Umsetzung des vorgeschlagenen Versionsmodells ergänzt. Es wird eine JAVA-Klassenstruktur vorgestellt, die die Eigenschaften des beschriebenen Versionsmodells für das Meta-Modell EXPRESS implementiert. Hierzu gehören die Darstellung der Klassen und Methoden, die zur Abbildung der objekt-orientierten Datenstrukturen, der Planungsstände und ihrer Änderungen (Deltas) sowie der beschriebenen Operationen umgesetzt wurden. Diese Klassenstruktur bildet den Kern der Datenverwaltung und wird unter anderem durch die Plug-ins für die Implementierung der erarbeiteten Methoden genutzt.

Im *dritten Abschnitt* wird schließlich die Integration der in Kapitel 5 beschriebenen Methoden und Verfahren diskutiert. Es wird gezeigt, wie diese Erweiterungen der Versionsverwaltung als eigenständige Dienste implementiert sind. Hierzu gehören das Teildatensatzverfahren, der Vergleich von Planungsständen, die Re-Integration geänderter Planungsdaten sowie das Zusammenführen divergierender Planungsstände. Diese Dienste arbeiten auf den im Versionsmodell abgebildeten Daten und zeigen durch die Darstellung der prinzipiellen Abläufe die gewählte algorithmische Umsetzung.

6.1 Überblick über die gewählte Softwareumsetzung

Mit der Softwareumsetzung wird das Ziel verfolgt, den vorgestellten Ansatz an praxisnahen Beispielen zu validieren. Aus diesem Grund wurde eine Architektur gewählt, die die Einbindung vorhandener CAD-Anwendungen und die Nutzung der hieraus generierten Daten ermöglicht. Eine solche Architektur gibt den Rahmen für die benötigten Funktionalitäten vor, die nachfolgend unterschiedlich detailliert vorgestellt werden. Der Schwerpunkt der prototypischen Umsetzung liegt jedoch auf den Methoden, die zur Aufbereitung und Verwaltung der Daten notwendig sind.

6.1.1 Gesamtarchitektur

Es wurde eine *Client-Server-Architektur* gewählt, die die Aufgabe der Datenverwaltung in einem zentralen *Server* übernimmt und eine relativ einfache Umsetzung einer verteilten Umgebung erlaubt. Der Zugriff auf den Server erfolgt über das Internet und ist über eine netz-

werkfähige Schnittstelle (API) realisiert. Für einen solchen Zugriff sind *Clients* notwendig, die die Nutzung verfügbarer CAD-Anwendungen ermöglichen. Der prinzipielle Aufbau der gewählten Architektur ist in der Abbildung 6-1 dargestellt, die auf oberster Ebene zwischen der Anwender- und Serverseite unterscheidet. Zwischen diesen beiden Seiten findet über die erwähnte Schnittstelle ein Informationsaustausch statt, wobei der Anwender eine aktive, anfragende und der Server eine passive, antwortende Rolle einnimmt.

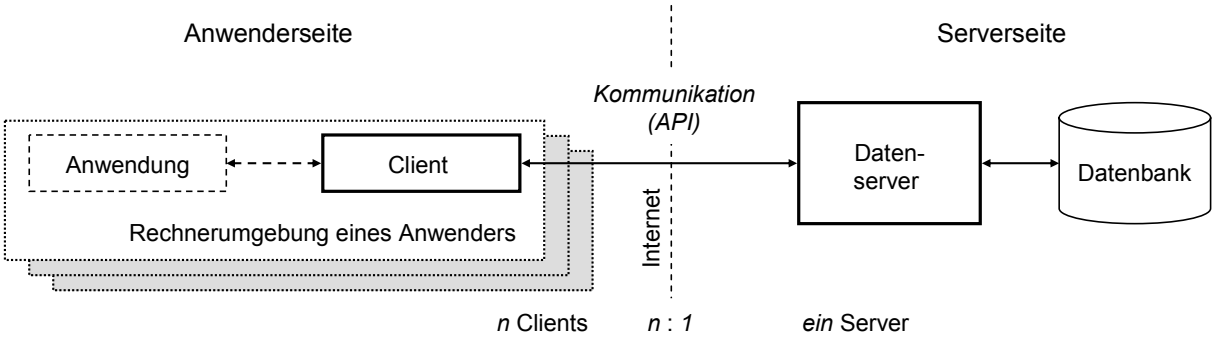


Abbildung 6-1 Einordnung der betrachteten Werkzeuge in eine Client-Server-Architektur

Auf der Anwenderseite befinden sich Werkzeuge, die ausschließlich auf dem Rechner des Anwenders ausgeführt werden. Hierzu gehören eine CAD-Anwendung, die für die Bearbeitung der Daten genutzt wird, sowie ein zugehöriger Client, der den Zugriff auf die zentral verwalteten Daten ermöglicht. Der betrachtete Client übernimmt in diesem Szenario die Aufgabe, den Datenaustausch zwischen einer vorhandenen CAD-Anwendung und dem Datenserver zu realisieren. Im Rahmen der Umsetzung stellt der Client ein Hilfsmittel dar, um praxisnahe CAD-Daten bei der Validierung berücksichtigen zu können.

Auf der Serverseite wird zwischen dem Datenserver und einer dahinter liegenden Datenbank unterschieden. Der Datenserver vereint hierbei alle Funktionen, die über eine dauerhafte Speicherung der Daten hinausgehen. Hierzu gehören die Interaktion mit den Clients, die in Kapitel 5 beschriebenen Dienste sowie die Anbindung an eine (relationale) Datenbank. Die im Kontext dieser Arbeit betrachteten Funktionalitäten sind somit vollständig in dem Datenserver integriert, der auch den Schwerpunkt der prototypischen Umsetzung darstellt.

6.1.2 Aufbau des Datenservers

Die wesentlichen Bestandteile des Datenservers sind in der Abbildung 6-2 dargestellt. Im Zentrum befindet sich die *Datenverwaltung*, die den Zugriff auf die Daten überwacht und das Zusammenspiel der einzelnen Komponenten koordiniert. Zu den Komponenten gehören die *Zugriffskontrolle*, die *Kommunikation*, der Bereich der *Plug-ins* sowie die Menge der im Arbeitsspeicher vorhandenen *flüchtigen Daten*. Für die prototypische Umsetzung sind in erster Linie die Plug-ins sowie die im Arbeitsspeicher abgelegten Daten interessant, da hier die vorgeschlagenen Methoden integriert bzw. die benötigten Daten abgelegt sind.

Ein Plug-in stellt eine wohl definierte Funktion bereit, die von der übergeordneten *Datenverwaltung* aufgerufen wird und eine Auswertung oder Erweiterung der Daten vornimmt. Die Ursache eines Plug-in-Aufrufs ist die Anfrage eines *Clients*, die von der *Kommunikation* entgegen genommen und an die *Datenverwaltung* weitergeleitet wird. Nach dem Überprüfen der Zugriffsrechte kann die gestellte Anfrage durch den Aufruf eines geeigneten Plug-ins unter der Kennung des anfragenden Anwenders ausgeführt werden. Die *Datenverwaltung* übernimmt hierbei verschiedene Aufgaben. Hierzu gehören die Zugriffe auf die Planungsstände,

Datenobjekte und Klassendefinitionen, die über verschiedene Suchkriterien abgefragt werden können. Zusätzlich werden die im Speicher vorhandenen Datenobjekte verwaltet, die bei Bedarf aus der Datenbank geladen bzw. in die Datenbank abgelegt werden. Ein Plug-in kann somit auf Datenobjekten arbeiten, die im Arbeitsspeicher des Datenservers bereitgestellt werden. Das Nachladen benötigter Datenobjekte geschieht im Hintergrund und muss von den Plug-ins nicht unmittelbar berücksichtigt werden. Ein Datenobjekt besitzt hierfür die Möglichkeit, weitere Datenobjekte dynamisch von der Datenverwaltung anzufordern.

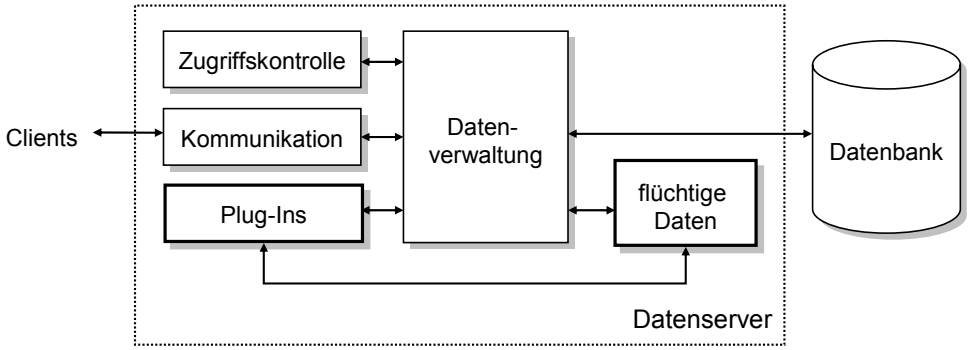


Abbildung 6-2 Wesentliche Bestandteile des Datenservers und ihre gegenseitigen Beziehungen

Als Plug-ins sind neben den in Kapitel 5 beschriebenen Diensten auch wichtige Basisdienste wie das Einlesen einer EXPRESS-Definition und das Lesen und Schreiben von STEP Dateien (ISO 10303-21) realisiert. Die Datenverwaltung ist dadurch auf organisatorische Aufgaben beschränkt, die unter anderem das Verwalten der Plug-ins und der vorhandenen Planungsstände umfasst.

6.1.3 Zugriff auf die Daten

Für den Zugriff auf die Daten sind technische, organisatorische und funktionale Aspekte interessant. Diese Aspekte bestimmen die Interaktion zwischen den Clients und dem Server.

Technisch steht der Datenaustausch im Mittelpunkt, der über die netzwerkfähige Schnittstelle realisiert ist. Die Grundlage bildet das Internetprotokoll TCP/IP, das für die Übertragung der Anfragen und auszutauschenden Datenobjekte verwendet wird. Die zu übertragenden Informationen werden in einen Datenstrom überführt, über das Netzwerk verschickt und auf der Gegenseite wieder in Anfragen und Datenobjekte zurückverwandelt. Auf diese Weise können auch Dateien übertragen werden, wodurch der dateibasierte Datenaustausch mit verfügbaren CAD-Anwendungen möglich wird. Die hierfür notwendigen Funktionalitäten sind serverseitig in dem Bereich der Kommunikation und anwenderseitig in den verwendeten Clients implementiert (siehe Abbildung 6-3).

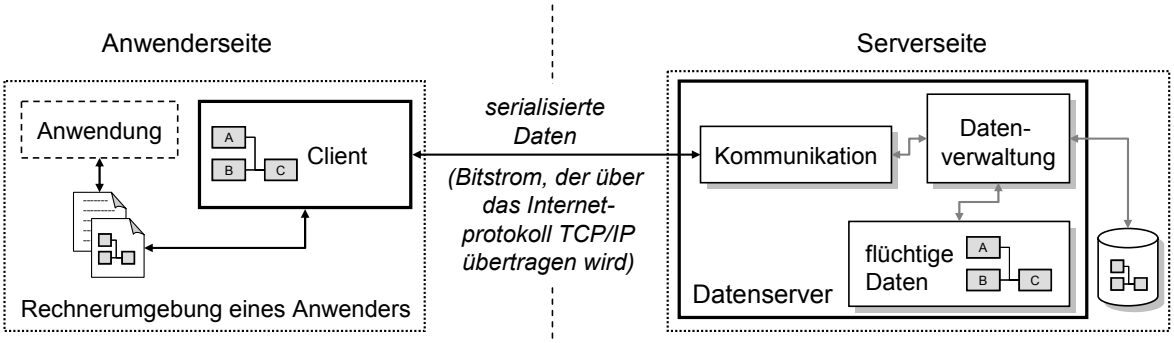


Abbildung 6-3 Datenaustausch zwischen Anwender- und Serverseite

Organisatorisch werden die Anfragen eines Anwenders in lange Transaktionen eingebettet, die konzeptionell nach den in Abschnitt 3.1.2 beschriebenen Schritten ablaufen. Um den Mehrbenutzerbetrieb zu ermöglichen, sind weitere, für die Verwaltung der Anfragen wichtige Schritte notwendig. Hierzu gehören das An- und Abmelden, das Authentifizieren der Anwender sowie das Eröffnen, Fortführen und Abschließen von Transaktionen. Aus diesen Schritten lässt sich die in Abbildung 6-4 dargestellte Sequenz ableiten, die für einen typischen Planungsschritt zu durchlaufen ist. Diese Sequenz ist auch für das Zusammenführen divergierender Planungsstände gültig, die im Vergleich zu einem Planungsschritt jedoch eine andere Ausprägung für das Laden und Speichern der Daten besitzt. Der Fortschritt innerhalb dieser Sequenz bestimmt schließlich die serverseitig nutzbaren Funktionen.

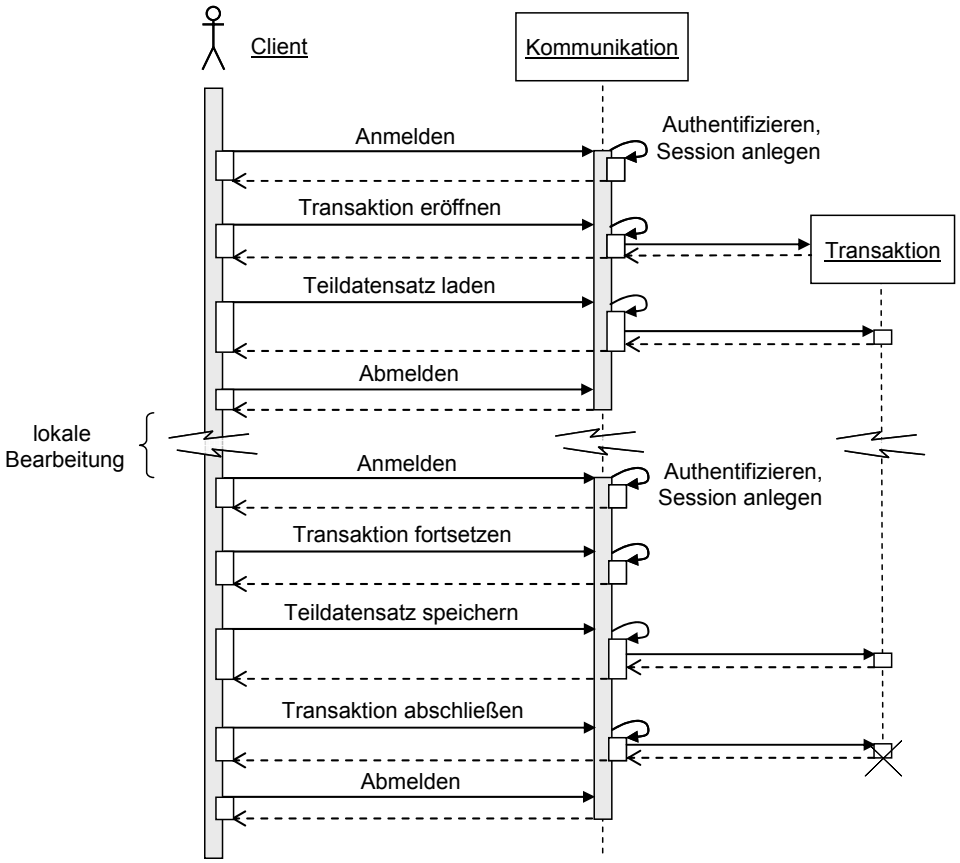


Abbildung 6-4 Typischer Ablauf eines Planungsschrittes als UML – Sequenzdiagramm

Das Ausführen einer Transaktion wird durch das Bereitstellen serverseitiger Funktionen unterstützt. Der Zugriff auf diese Funktionen erfolgt durch Anfragen, die in der verwendeten API spezifiziert sind. In dieser API sind die verfügbaren Funktionen, ihre Funktionsaufrufe sowie die zu erwartenden Rückgabewerte formal beschrieben. Die API ist als ein objektorientiertes Modell implementiert, das verschiedene Anfrage- und Rückgabeobjekte definiert. Eine konkrete Anfrage wird hierbei durch den Typ der verwendeten Anfrageobjekte sowie ihrer Attributbelegung bestimmt. Durch diesen Ansatz ist es möglich, die Kommunikationsobjekte wie Datenobjekte zu behandeln und somit die gleichen Mechanismen für den Datenaustausch zu verwenden. Die Grundlage bildet das Schema *Communication*, das ebenso in EXPRESS definiert ist.

6.2 Umsetzung des Versionsmodells

Das in Kapitel 4 beschriebene Versionsmodell wurde in eine JAVA-Klassenstruktur überführt, die zur Abbildung von EXPRESS-Datenstrukturen und den zugehörigen Daten geeignet ist. Der mengentheoretischen Formalisierung des Versionsmodells stehen hierbei die in EXPRESS nutzbaren Konzepte zur Definition einer objekt-orientierten Datenstruktur gegenüber. In der gewählten Umsetzung wurden beide Spezifikationen miteinander kombiniert, um die Anwendbarkeit des vorgeschlagenen Versionsmodells zu zeigen.

Das objekt-orientierte Konzept wird in dem vorgeschlagenen Versionsmodell auf die Elemente *Objekt*, *Attribut*, *Objektklasse* und *Merkmal* sowie die *Referenz-* und *Instanzrelation* reduziert (siehe Abschnitt 4.1). Hierdurch wird ein Abstraktionsgrad erreicht, der für die Betrachtung der Versionsverwaltung ausreichend ist. Für das Abbilden einer EXPRESS-Datenstruktur ist eine weitere Differenzierung notwendig, um zwischen den verschiedenen Attributtypen und Objektverknüpfungen unterscheiden zu können. Die nachfolgende Beschreibung orientiert sich an der Abstraktion des Versionsmodells und verzichtet zugunsten einer besseren Lesbarkeit auf die vollständige Darstellung der umgesetzten JAVA-Klassenstruktur. Neben der Abbildung der objekt-orientierten Datenstruktur ist vor allem die Umsetzung der Versionsverwaltung interessant, die sich mit der Abbildung der *Planungsstände* und *Versionsrelationen* beschäftigt. Um Inkonsistenzen in der Versionsverwaltung zu vermeiden, sind darüber hinaus Mechanismen zur Einhaltung der im Versionsmodell formulierten Konsistenzbedingungen notwendig. Auf dieser Grundlage werden abschließend die Zugriffsmethoden betrachtet, die eine Interaktion mit den verwalteten Daten erlauben.

6.2.1 JAVA-Klassen zur Beschreibung der EXPRESS-Datenstruktur

In der Abbildung 6-5 ist ein Auszug der implementierten JAVA-Klassenstruktur zur Verwaltung EXPRESS-basierter Daten dargestellt[1]. In dieser Struktur werden die zwei Wurzelklassen *ExpElement* und *RelationInfo* definiert, die einerseits für die Beschreibung von Attributen, Objekten und Objektklassen und andererseits für die Beschreibung von Merkmalen genutzt werden.

Objekte und Attribute

Eine Instanz der Klasse *ExpElement* stellt eine referenzierbare Informationseinheit dar, die entweder ein einfacher Datentyp (*ExpSimpleType*), eine beliebig strukturierte Menge von Informationseinheiten (*ExpAggregation*), ein gelöschtes Attribut (*DeletedAttribute*) oder ein eigener Datentyp (*ExpNamedDataType*) sein kann. Ein eigener Datentyp besitzt einen frei wählbaren „Namen“ und unterteilt sich weiter in die Klassen *ExpEntity* und *ExpType*. Mit Ausnahme von *ExpEntity* stellen diese Informationseinheiten im Sinne des Versionsmodells Attribute dar und dienen der Beschreibung von Attributzuständen. Der Typ *ExpEntity* wird dagegen für die Definition von Objekten genutzt und strukturiert die Informationseinheiten zu eindeutig identifizierbaren Datenobjekten. Die Attribute eines solchen Datenobjektes werden über die beiden Listen *attributes* und *relationInfo* organisiert, die einerseits die Verknüpfung

[1] Ein objekt-orientiertes Datenmodell wird somit in einer objekt-orientierten Programmiersprache „nachgebildet“. Dies ist notwendig, da nicht alle Konzepte der EXPRESS-Datenstruktur sowie des vorgeschlagenen Versionsmodells in JAVA unterstützt werden. Der Einsatz einer objekt-orientierten Programmiersprache bietet dennoch viele Vorteile, führt aber zu begrifflichen Überschneidungen. Für eine bessere Unterscheidung werden daher die Begriffe *Instanz* und *Klasse* für die Beschreibung der Programmumsetzung verwendet, die in dem Versionsmodell mit *(Daten-)Objekt* und *Objektklasse* bezeichnet sind.

mit der entsprechenden Informationseinheit und andererseits einen Verweis auf das dadurch beschriebene Merkmal definieren. Die Zuordnung erfolgt über die Position in der Liste und wird durch die *ExpEntity*-Instanz überwacht. Eine Informationseinheit, die in der Liste *attributes* die Position *n* besitzt, beschreibt folglich den Attributzustand für ein Merkmal, das sich in der Liste *relationInfo* ebenso an der Position *n* befindet. Im Sinne des Versionsmodells übernimmt die Position der Liste somit die Aufgabe, die Zugehörigkeit eines Attributes zu einem Merkmal zu beschreiben.

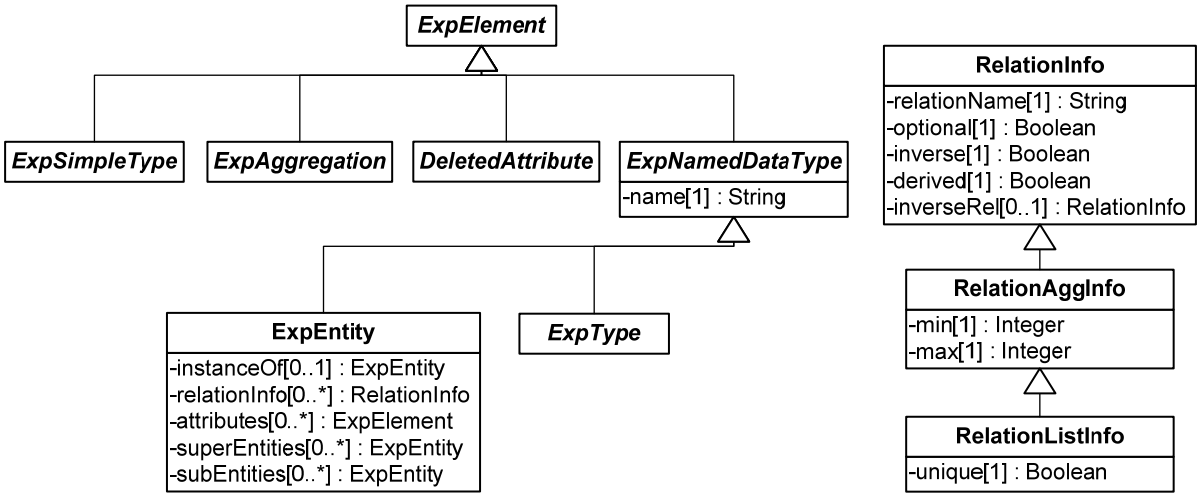


Abbildung 6-5 UML-Darstellung der JAVA-Klassenstruktur und wichtiger Attribute, die zur Abbildung von EXPRESS Datenstrukturen und zugehörigen Daten umgesetzt wurde.

Merkmale

Eine Instanz der Klasse *RelationInfo* oder einer davon abgeleiteten Klasse beschreibt im Sinne des Versionsmodells ein Merkmal. Sie wird von einer Instanz der Klasse *ExpEntity* verwendet, um einem Attribut das hierdurch beschriebene Merkmal zuzuordnen. Neben einem eindeutigen Namen (*relationName*) werden verschiedene Konsistenzbedingungen erfasst, die in der abgebildeten EXPRESS-Datenstruktur formuliert sind und für Konsistenzprüfungen, den Vergleich von Planungsständen sowie den Datenaustausch nach ISO 10303-21 genutzt werden. Hierzu gehören die Wahrheitswerte *optional*, *inverse*, *derived* sowie der Verweis zu einem hiermit verbundenen, in umgekehrter Richtung definierten Merkmal (*inverseRel*). Wird ein Merkmal über eine Menge von Informationseinheiten beschrieben, so sind zusätzlich Angaben über die zulässige Kardinalität (*min*, *max* in der Klasse *RelationAggInfo*) sowie die Möglichkeit über das mehrfache Vorhandensein derselben Informationseinheiten (*unique* in der Klasse *RelationListInfo*) notwendig. Dagegen sind in dieser Klasse keine Informationen über den verwendeten Attributtyp enthalten. Der Attributtype wird statt dessen über die Objektklasse festgelegt.

Objektklassen

Eine Objektklasse wird ebenso wie ein Objekt durch eine Instanz der Klasse *ExpEntity* dargestellt. Auch wenn konzeptionell zwischen einer Objektklasse und einem Objekt unterschieden wird, so bestehen viele Gemeinsamkeiten, die ein Zusammenlegen begünstigen. Die Unterscheidung zwischen einem Objekt und einer Objektklasse wird durch das in *ExpEntity* definierte Attribut *instanceOf* vorgenommen. Für Objekte wird über dieses Attribut die zugehörige Objektklasse beschrieben. Das Attribut *instanceOf* entspricht somit der Instanzrelation des

Versionsmodells und bleibt für die Beschreibung einer Objektklasse unbelegt. Entsprechend dem vorgestellten Versionsmodell wird ein Objekt auf die Zugehörigkeit zu genau einer Objektklassen beschränkt, die damit auf die Abbildung von *OR*-Vererbungsbeziehungen festgelegt ist. Die Merkmale einer Objektklasse werden schließlich auf die gleiche Weise wie die Zuordnung der Attribute zu Objekten beschrieben. Anstatt einer konkreten Attributbelegung wird über die Liste *attributes* aber der Attributtyp beschrieben, der ebenso über Instanzen der von *ExpElement* abgeleiteten Klassen beschrieben wird. Neben Klassen, die die Beschreibung von Attributtypen und Attributzuständen vereinen, stehen hierfür eigenständige Klassen bereit, die beispielsweise für die Beschreibung von Aufzählungen und verschiedenen *Select*-Datentypen benötigt werden.

Vererbung und Versionsrelationen

Die Klasse *ExpEntity* verfügt mit den Attributen *superEntities* und *subEntities* zusätzlich über die Möglichkeit, die Vererbung zwischen den Objektklassen zu erfassen. Ist eine Objektklasse von anderen (Basis-)Objektklassen abgeleitet, so wird diese Beziehung über die Liste *superEntities* abgebildet. Diese Basis-Objektklassen besitzen gleichzeitig eine Beziehung zu den davon abgeleiteten Objektklassen, die über das Attribut *subEntities* beschrieben werden. Die Beziehung zu einer Basis-Objektklasse erzeugt folglich eine Umkehrbeziehung, die als Information zwar redundant, für einen schnellen Zugriff auf abgeleitete Objektklassen aber notwendig ist. Während für Objektklassen auf diese Weise die Vererbungsbeziehung abgebildet wird, werden diese Attribute für Objekte zur Beschreibung der Versionsrelation genutzt. Das Attribut *superEntities* wird hierbei für die Beschreibung der Vorgänger-Objekte und das Attribut *subEntities* für die Beschreibung der Nachfolger-Objekte genutzt.

In der Abbildung 6-6 ist die Anwendung der vorgestellten Klassenstruktur an einem Beispiel gezeigt. In diesem Beispiel werden zwei Objekte einer Objektklasse definiert, die über eine Versionsrelation miteinander verbunden sind. Durch die Instanz *Objektklasse1* der Klasse *ExpEntity* wird eine Objektklasse definiert, die einen beliebigen Punkt in einer Ebene beschreiben kann. Diese Objektklasse besitzt den Namen *2D_Punkt* und ist durch ein Merkmal für die X- und die Y-Koordinate gekennzeichnet. Beide Merkmale besitzen den Typ *ExpReal*[1] und sind über die Instanzen *Merkmal1* und *Merkmal2* der Klasse *RelationInfo* beschrieben. Von der Objektklasse *2D_Punkt* existieren zusätzlich zwei Objekte, die ebenso über eine Instanz der Klasse *ExpEntity* beschrieben und als *Objekt1* und *Objekt2* bezeichnet sind. Beide Objekte definieren über das Attribut *name* einen eindeutigen Identifikator, der durch den Datenserver vergeben und für die Verwaltung der Objekte verwendet wird. Zwischen diesen Objekten und der Objektklasse *2D_Punkt* besteht eine Instanzrelation, die über das Attribut *instanceOf* definiert ist. Über diese Beziehung werden die zulässigen Merkmale der beiden Objekte festgelegt, die schließlich mit Werten belegt werden können. In dem Beispiel wird für das Objekt *Objekt1* die X-Koordinate zu -1.5 und die Y-Koordinate zu 5.0 definiert. Für das Objekt *Objekt2*, das über das Attribut *superEntities* als Nachfolger des Objektes *Objekt1* definiert ist, wird die Y-Koordinate auf den Wert 3.0 geändert und besitzt dadurch die Koordinaten X=-1.5, Y=3.0.

[1] *ExpReal* ist eine Subklasse von *ExpSimpleType* und dient der Beschreibung eines Attributtyps (Merkmal). Zu der Objektklasse *ExpReal* existiert zusätzlich die Subklasse *ExpRealValue*, die zur Beschreibung der Attributbelegung verwendet wird.

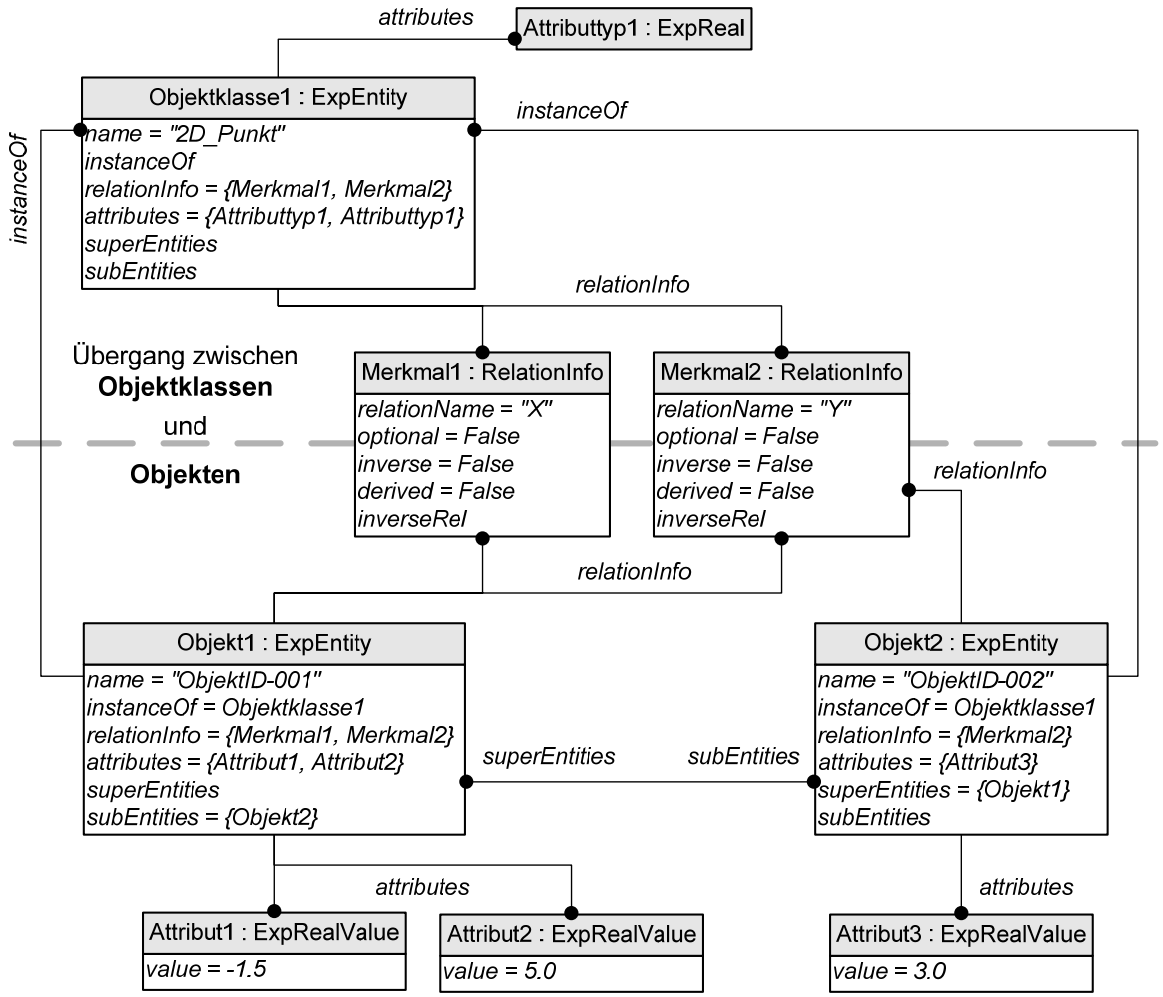


Abbildung 6-6 Anwendung der JAVA-Klassenstruktur an einem Beispiel

Die Elemente des formalisierten Versionsmodells und ihre Entsprechung in der umgesetzten JAVA-Klassenstruktur sind abschließend in der nachfolgenden Tabelle zusammengefasst.

Tabelle 1 Gegenüberstellung des Versionsmodells mit seiner Umsetzung in eine JAVA-Klassenstruktur

Versionsmodell	**Abbildung in JAVA-Klassen**
Objekt $o \in O$	ExpEntity
Objektklasse $k \in K$	ExpEntity
Instanzrelation R_{OK}	Verknüpfung über ExpEntity.instanceOf
Attribut $a \in A$	*ExpElement* und Subklassen
Merkmal $m \in M$	*ExpElement* + *RelationInfo* und deren Subklassen
$o \subset A$	ExpEntity.attributes und ExpEntity.name (für a_{ID})
$m \subset A$	Synchronisierung der Listen ExpEntity.attributes und ExpEntity.relationInfo
$k \subset M$	ExpEntity.attributes und ExpEntity.relationInfo

Referenzrelation R_{AO}	ExpEntity als Subklasse von ExpElement
Versionsrelation R_{VN}	ExpEntity.superEntities (Vorgänger) ExpEntity.subEntities (Nachfolger)
Planungsstand P	ExpDataModel (siehe Abbildung 6-7)

6.2.2 Verwaltung der Planungsstände

Ein Planungsstand wird nach der Definition (4.6) durch eine Menge von Objekten beschrieben. Durch diese Gruppierung wird gemeinsam mit der Versionsbeziehung eine Ordnung der Objekte erreicht, die für den Zugriff auf die verwalteten Daten notwendig ist. Zwischen den Planungsständen besteht ebenfalls eine Ordnung, die aber nicht durch das Versionsmodell, sondern durch den Planungsablauf bzw. den Planungsschritt dokumentiert wird. Dennoch wird in der gewählten Umsetzung eine Beziehung zwischen den Planungsständen abgebildet. Durch das Verwenden einer Vorgängerbeziehung ist es analog zur Objektbeschreibung möglich, nur die veränderten Objekte zu erfassen. Eine Objektmenge zur Beschreibung eines Planungsstandes wird dadurch deutlich reduziert. In der Umsetzung ist folglich zwischen der internen Abbildung der Daten und den nach außen bereitgestellten Funktionen zu unterscheiden. Diese Unterscheidung besteht aber nicht nur formal, sondern ist auch praktisch durch die Trennung in eine Objektklasse, die die Daten eines Planungsstandes abbildet, und eine JAVA-Klasse, die auf diese Daten zugreift und die benötigten Funktionen bereitstellt, umgesetzt. Durch diesen Ansatz ist es möglich, die Planungsstände und die verwalteten Produktdaten auf dieselbe Grundlage zu stellen. Für die Speicherung und Übertragung der Daten können somit die gleichen Mechanismen genutzt werden.

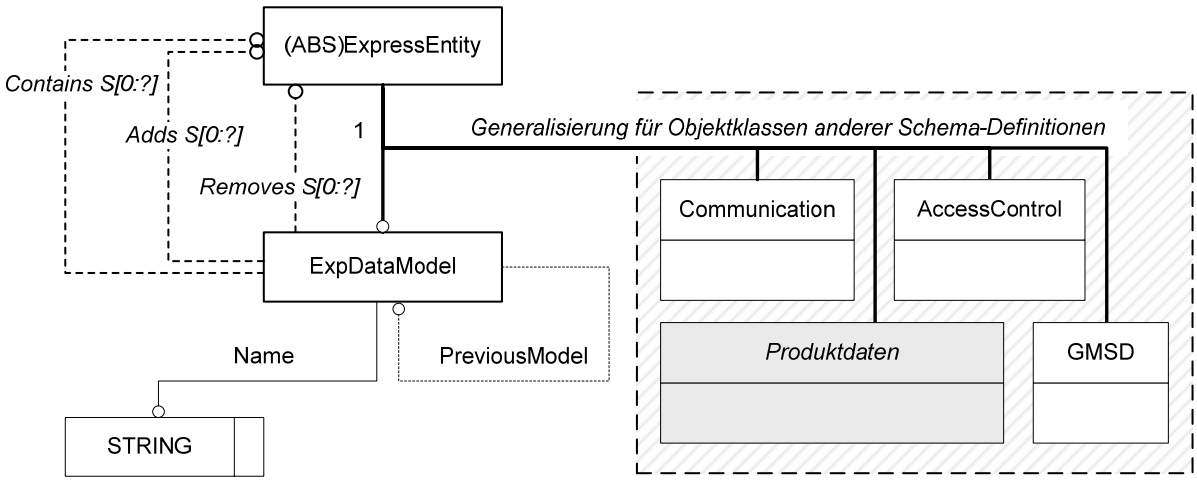


Abbildung 6-7 EXPRESS-G Darstellung des *ExpServerMgmt* Schemas und dessen Bezug zu anderen Schema-Definitionen

Objektklasse zur Beschreibung eines Planungsstandes

Für die Abbildung der Planungsstände wird das Meta-Schema *ExpServerMgmt* verwendet, das die beiden Objektklassen *ExpressEntity* und *ExpDataModel* definiert. Die abstrakte Objektklasse *ExpressEntity* stellt innerhalb des Datenservers die Wurzelklasse für alle anderen Objektklassen dar. Durch diese Wurzelklasse ist die Definition einer Verknüpfung möglich, die alle im Datenserver verwalteten Objekte erreichen kann. Eine solche Verknüpfung wird von der Objektklasse *ExpDataModel* für die Gruppierung von Objekten genutzt, um einen Planungsstand selbst als ein *Objekt* erfassen zu können. Die Bedeutung der Wurzelklasse *Ex-*

pressEntity ist aber nicht nur auf die Abbildung von Planungsständen beschränkt, sondern wird auch in anderen Bereichen genutzt, beispielsweise für die Zugriffskontrolle und die Kommunikation. In der Abbildung 6-7 ist die Definition der beiden Objektklassen sowie die Generalisierungsbeziehung für die Objektklassen anderer Schema-Definitionen dargestellt. Durch diese Generalisierung sind die Produktdaten erreichbar, die zu einem Planungsstand über die Objektklasse *ExpDataModel* gruppiert werden. Hierfür werden die Attribute *Contains*, *Adds* und *Removes* verwendet, die bezugnehmend auf einen über *PreviousModel* angegebenen Planungsstand die zugehörigen Objekte beschreiben. Zusätzlich ist ein Attribut *Name* definiert, das die eindeutige Identifizierung eines Planungsstandes ermöglicht.

JAVA-Klasse für den Zugriff auf einen Planungsstand

Für die Verwaltung eines Planungsstandes wird innerhalb des Datenservers eine JAVA-Klasse bereitgestellt, die den Zugriff auf die Daten und das Anlegen eines neuen Planungsstandes übernimmt. Die in der Abbildung 6-7 dargestellte Objektklasse *ExpDataModel* wird somit durch eine JAVA-Klasse implementiert. Diese hat ebenfalls den Namen *ExpDataModel* und besitzt als Attribut den verwalteten Planungsstand, also eine Instanz der Klasse *ExpEntity*.

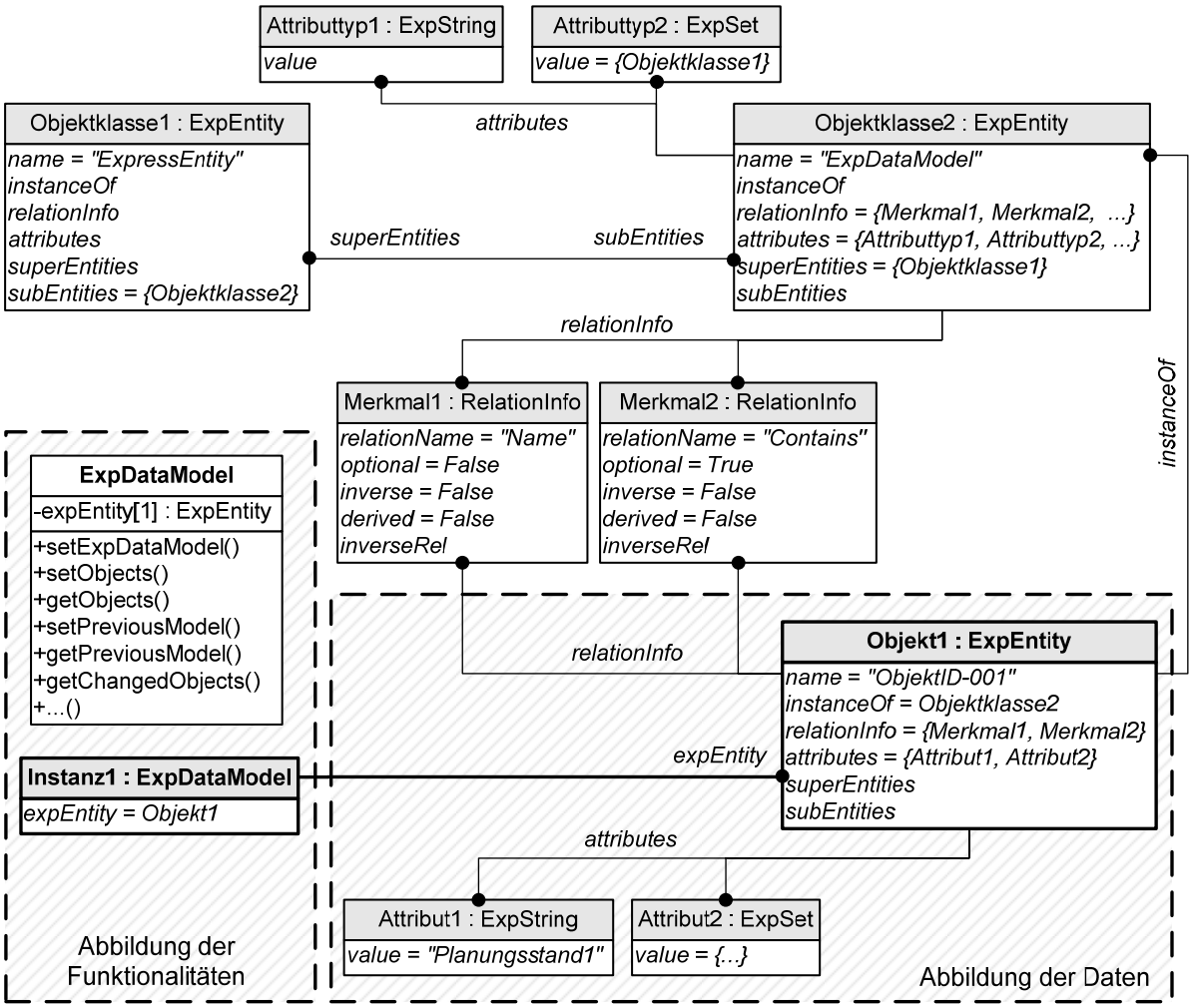


Abbildung 6-8 Verwalten eines Planungsstandes.

Dieser Zusammenhang ist in der Abbildung 6-8 am Beispiel der Instanzen *Instanz1* und *Objekt1* dargestellt. Beide Instanzen sind über das Attribut *Instanz1.expEntity* miteinander ver-

bunden. Über diese Verbindung ist der Zugriff auf den verwalteten Planungsstand (*Objekt1*) möglich. Hierfür werden die auszugsweise abgebildeten Zugriffsmethoden der JAVA-Klasse *ExpDataModel* verwendet. Zu diesen Zugriffsmethoden gehören das Erzeugen eines neuen Planungsstandes, das Zurückgeben der zugehörigen Objekte sowie das Ermitteln von Änderungen zwischen Planungsständen. Darüber hinaus können die in Abschnitt 4.2 beschriebenen Konsistenzbedingungen für einen gültigen Planungsstand überprüft werden.

6.2.3 Methoden für den Datenzugriff

Die Klasse *ExpEntity* wird für die Beschreibung von Objekten und Objektklassen verwendet. Für den Zugriff auf die hierüber beschriebenen Daten werden Methoden bereitgestellt, die den Umgang mit dem zugrunde liegenden Versionsmodell vereinfachen. Durch diese Methoden ist es möglich, mit den Objekten wie in einem unversionierten System zu arbeiten und zusätzlich die Informationen der Versionsverwaltung zu nutzen.

Das vorgestellte Versionsmodell wird für die Abbildung der Planungshistorie eingesetzt. Aus diesem Grund sind die Planungsdaten der dokumentierten Planungsstände nicht veränderbar. Dennoch werden Methoden bereitgestellt, die das Modifizieren der Objekte bzw. Objektklassen ermöglichen. Dies ist notwendig, um neue Objekte im Sinne des Versionsansatzes aufbereiten und in das vorhandene Objektnetz integrieren zu können. Ein Objekt ist aber nur solange veränderbar, solange es nicht in das Objektnetz integriert und für andere Anwender freigegeben wurde. Aus diesen Gründen wird bei den nachfolgend betrachteten Methoden zwischen dem lesenden und dem schreibenden Datenzugriff unterschieden.

6.2.3.1 Lesender Datenzugriff

Für den lesenden Zugriff auf eine Instanz der Klasse *ExpEntity* sind folgende Angaben interessant:

- die Unterscheidung zwischen Objekt und Objektklasse,
- der Objektzustand bzw. die Definition der Objektklasse,
- die eindeutige Objektkennung bzw. der Name der Objektklasse sowie
- die Vorgänger und Nachfolger eines Objektes bzw. die Super- und Subklassen einer Objektklasse.

Die Unterscheidung zwischen Objekt und Objektklasse leitet sich aus der Belegung des Attributs *instanceOf* ab. Die Objektkennung bzw. der Klassenname ist durch das Attribut *name* beschrieben. Schließlich sind über die Attribute *superEntities* und *subEntities* die Vorgänger und Nachfolger bzw. die Super- und Subklassen ermittelbar. Die hierfür bereitgestellten Methoden sind dementsprechend auf einfache „Get“-Funktionen beschränkt, die unter Wahrung der Zugriffsrechte den Zustand der genannten Attribute zurückgeben. Im Vergleich hierzu sind die Methoden für den Zugriff auf den Objektzustand komplexer und erfordern das Auswerten der Vorgänger sowie das Einbeziehen des betrachteten Planungsstandes (siehe Abschnitt 4.3.1). Dies ist notwendig, da ein Objekt bezüglich seiner Vorgänger nur die „geänderten“ Attribute in der Liste *attributes* enthält und eine Verknüpfung nicht immer auf die gesuchte Objektversion verweist. Um nicht nur ein gültiges, sondern auch ein richtig verknüpftes (aktuelles) Attribut ermitteln zu können, ist neben dem gesuchten Merkmal folglich auch der betrachtete Planungsstand anzugeben.

Zugriff auf Attribute eines Objektes

Wird eine Anfrage nach einem Attribut gestellt, so wird in einem ersten Schritt über das gesuchte Merkmal zunächst das gültige Attribut ermittelt. Ist das Attribut in dem betrachteten Objekt nicht enthalten, so wird die Anfrage an ein Vorgängerobjekt weitergeleitet. Nach der Definition (4.19) des Versionsmodells ist es hierbei egal, welches der unmittelbaren Vorgängerobjekte verwendet wird. Ein gültiges Attribut wird auf diese Weise spätestens dann ermittelt, wenn kein weiterer Vorgänger vorhanden ist. Konnte kein Attribut gefunden werden, so ist das gesuchte Merkmal nicht belegt. Wird über ein gültiges Attribut eine Verknüpfung zu anderen Objekten beschrieben, so wird in einem zweiten Schritt die Aktualität der Verknüpfung überprüft. Ist ein verknüpftes Objekt nicht in dem gegebenen Planungsstand enthalten, so wird das referenzierte Objekt über dessen Nachfolger ermittelt. Nach der Bedingung (4.20) gehört genau ein Nachfolger zu dem betrachteten Planungsstand, der durch das Traversieren des Versionsbaumes gefunden werden muss. Die Suche wird folglich abgebrochen, wenn das referenzierte Objekt gefunden ist.

Methodenaufruf: ***Objekt10.getAttribut(Merkmal1, Objekt-Planungsstand1);***

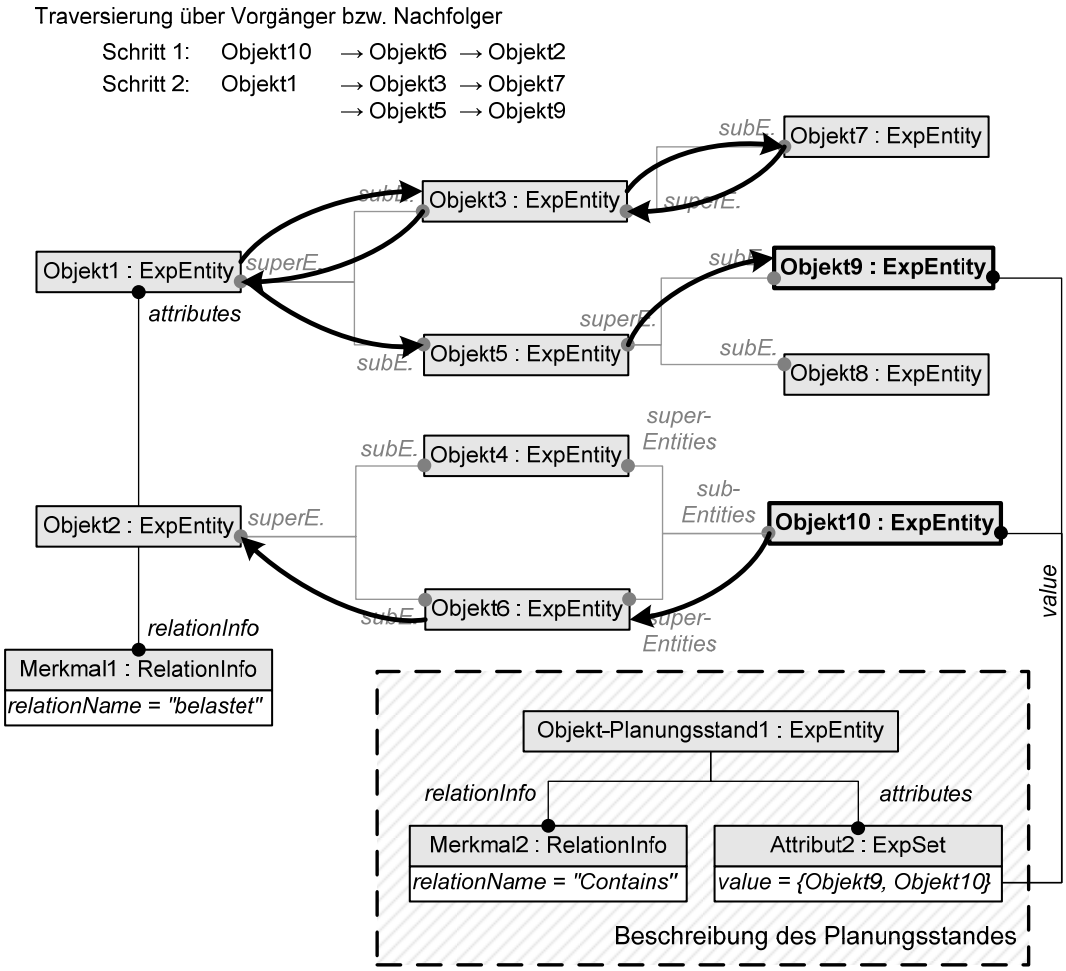


Abbildung 6-9 Ermittlung eines aktuellen Attributs durch Traversierung des Versionsbaumes

In der Abbildung 6-9 ist ein Beispiel für eine Attributanfrage und die hierfür notwendige Traversierung gezeigt. Für das Objekt *Objekt10* und den Planungsstand *Objekt-Planungsstand1* wird die Belegung für die Verknüpfung *Merkmal1* ermittelt. Im ersten Schritt wird das gültige Attribut bestimmt, das in dem Vorgänger *Objekt2* definiert ist. Über dieses Attribut wird auf

das Objekt *Objekt1* verwiesen, das jedoch nicht zu dem gegebenen Planungsstand gehört. Um das tatsächlich referenzierte Objekt zu ermitteln, werden die Nachfolger von *Objekt1* über eine geeignete Suchstrategie überprüft. In dem gezeigten Beispiel werden hierfür die Objekte *Objekt3*, *Objekt7* und *Objekt5* durchlaufen, bevor mit *Objekt9* das in dem gegebenen Planungsstand enthaltene und somit referenzierte Objekt gefunden wird.

Unterscheidung in aktuelle, gültige und geänderte Attribute

Für den Zugriff auf die Attribute stehen mehrere Methoden zur Verfügung, wodurch insbesondere die gesuchten Attribute unterschieden werden. Die Unterscheidung der Attribute ist eine Folge der Versionsverwaltung, die *aktuelle*, *gültige* und *geänderte* Attribute kennt. Diese Unterscheidung leitet sich aus den zuvor beschriebenen Schritten ab. Wie gezeigt erfordert das Ermitteln eines *aktuellen* Attributs die Angabe des betrachteten Planungsstandes sowie das Durchlaufen beider Schritte. Fehlt die Angabe des Planungsstandes, so wird nach dem ersten Schritt abgebrochen und das ermittelte *gültige* Attribut zurückgegeben. Neben der Betrachtung eines konkreten Merkmals sind auch pauschale, auf den vollständigen Objektzustand bezogene Anfragen denkbar. Hierbei werden nicht einzelne Attribute sondern alle Attribute eines Objektes in einer Menge von Merkmal-Attribut-Paaren zurückgeben. Für pauschale Anfragen ergibt sich zusätzlich die Möglichkeit, nur *geänderte* Attribute zu ermitteln. In diesem Fall sind Attribute gesucht, die in der Liste *attributes* des betrachteten Objektes enthalten sind. Sofern mehrere Vorgänger existieren, stellt diese Liste eine Obermenge der vorgenommenen Änderungen dar. Sind in diesem Fall die Änderungen zu einem der Vorgänger gesucht, so sind aus dieser Obermenge die unveränderten Attribute zu entfernen. Hierfür werden alle gültigen Attribute des gegebenen Vorgängers mit den Delta-Attributen des betrachteten Objektes verglichen und bei Wertegleichheit aus der Ergebnismenge entfernt.

6.2.3.2 Schreibender Datenzugriff

Durch den schreibenden Zugriff ist es möglich, eine Instanz der Klasse *ExpEntity* zu verändern. Der schreibende Zugriff ist jedoch auf die Änderung der Objektattribute (*attributes* und *relationInfo*) und Vorgänger (*superEntities*) beschränkt. Alle anderen Informationen werden entweder beim Erzeugen der Instanz festgelegt (*name* und *instanceOf*) oder durch interne Abläufe aktualisiert (*subEntities* sowie die Zulässigkeit schreibender Zugriffe über den zusätzlich verwalteten Objektstatus). Die Aktualisierung von *subEntities* (Nachfolger) wird durch die Änderung der Umkehrrelation, die über *superEntities* beschrieben ist, erzwungen. Dagegen ist die Änderung des Objektstatus an eine Konsistenzprüfung gekoppelt, die von einem Planungsstand vorgenommen wird. Solange kein solcher Statuswechsel durchgeführt wurde, ist der schreibende Zugriff, also eine Änderung der Objektattribute und Vorgängerbeziehungen, erlaubt.

Im Gegensatz zur Klasse *ExpEntity* werden für alle anderen von *ExpElement* abgeleiteten Klassen keine modifizierenden Methoden bereitgestellt. Dies ist notwendig, da diese „Attribute“ den Datenzugriff nicht überwachen und daher nur einmalig, nämlich beim Erzeugen, mit Werten belegt werden können. Die Änderung eines Objektattributs erfolgt daher nicht durch die Änderung des Attributzustandes sondern durch das Anlegen eines neuen Attributs, das dem betrachteten Objekt zusätzlich zugewiesen werden muss und dadurch das alte Attribut ersetzt.

6.3 Ergänzende Dienste für die Versionsverwaltung

Die gewählte Serverarchitektur bietet die Möglichkeit, die vorgestellten Basisfunktionalitäten durch spezialisierte Dienste zu erweitern. Diese Dienste werden als Plug-ins realisiert, die in dem Datenserver über einen eindeutigen Namen registriert werden. Jedes Plug-in implementiert das Interface *ExpServerPluginInterface*, das jeweils eine Methode für das Starten und das kontrollierte Beenden des Dienstes deklariert. Ein Plug-in wird als ein JAVA-Thread gestartet und kann mit den Zugriffsrechten des aufrufenden Anwenders selbstständig und parallel zu anderen Diensten mit den verwalteten Objekten arbeiten. Auf diese Weise sind unter anderem Basisdienste implementiert, die für das Verarbeiten von EXPRESS-Datenstrukturen sowie das Einlesen und Schreiben des STEP-Dateiformats benötigt werden. Als Plug-ins sind aber auch die in Kapitel 5 vorgestellten Dienste realisiert, die

- das Erzeugen von Teildatensätzen,
- den Vergleich von Planungsständen,
- die Re-Integration von Änderungen sowie
- das Zusammenführen divergierender Planungsstände

ermöglichen und nachfolgend näher betrachtet werden.

6.3.1 Berechnung von Teildatensätzen

Für die Beschreibung eines Teildatensatzes wird das in Abschnitt 5.1 vorgestellte GMSD-Schema verwendet, das als objekt-orientierte Struktur in EXPRESS abgebildet ist (siehe Anhang). Eine Teildatensatzbeschreibung stellt somit eine Belegung des GMSD-Schemas dar und wird in eine Teildatensatzabfrage, die ebenfalls in EXPRESS definiert ist, eingebettet und über die in Abschnitt 6.1.3 beschriebenen Mechanismen des Informationsaustausches zum Datenserver übertragen. Auf dem Datenserver liegt die auszuwertende Teildatensatzbeschreibung dementsprechend als Instanzen der in Abschnitt 6.2.1 beschriebenen Klassen vor, die von einem als Plug-in implementierten Dienst ausgewertet werden kann. Dem Plug-in wird hierfür eine Instanz der Klasse *ExpEntity* übergeben, die ein Objekt der Objektklasse *PartialModelQuery* repräsentiert und die Wurzel[1] der Teildatensatzbeschreibung darstellt.

Ein Objekt der Objektklasse *PartialModelQuery* unterteilt die Teildatensatzbeschreibung in zwei Schritte: 1.) die Objektselektion und 2.) die Anwendung eines optionalen Objektfilters. Beide Schritte werden sequenziell durchlaufen und beruhen auf der Auswertung und Weitergabe von Objektmengen. Jede Objektmenge hat ihren Ursprung in der Wahl eines Planungsstandes. Ein solcher Planungsstand wird für die Auflösung von Verknüpfungen benötigt und beschreibt zugleich eine Menge zulässiger Objekte. Eine Objektmenge ist somit stets an einen Planungsstand gebunden und bildet in dieser Kombination die Grundlage für die Auswertung der Selektions- und Filteranweisungen.

Objektselektion

Für die Auswahl einer Objektmenge (*QuerySetSelect*) existieren vier Möglichkeiten, die mit Ausnahme des Planungsstandes (*ModelID*) auf der Auswertung anderer Objektmengen beru-

[1] Eine Teildatensatzbeschreibung kann aus mehreren Objekten der Objektklasse *PartialModelQuery* bestehen, die in einer Baumstruktur angeordnet sind. Den Ausgangspunkt einer Teildatensatzbeschreibung bildet das Wurzelobjekt, dem alle anderen *PartialModelQuery*-Objekte untergeordnet sind.

hen. Dadurch entstehen Abhängigkeiten zwischen den Selektionsanweisungen einer Anfrage. In der Umsetzung führen diese Abhängigkeiten zu einem rekursiven Abstieg, der erst mit der Wahl eines Planungsstandes beendet wird. Ist ein Planungsstand ermittelt, so wird über den eindeutigen Namen des Planungsstandes mit Hilfe der Datenverwaltung die zugehörige Instanz von *ExpDataModel* ermittelt und gemeinsam mit der zugehörigen Objektmenge den hiervon abhängigen Selektionsanweisungen zur Verfügung gestellt. Je nach Selektionsanweisung wird anschließend eine hierfür geeignete Methode aufgerufen, die auf den gegebenen Objekten und unter Verwendung des zugehörigen Planungsstandes die geforderte Auswahl vornimmt. Das Ergebnis der Auswahl wird wiederum als eine Objektmenge zurückgeben. Durch diese Schachtelung von Selektionsanweisungen entsteht für jeden betrachteten Planungsstand eine Menge ausgewählter Objekte. Diese Objektmengen stellen schließlich das Ergebnis der Objektselektion dar.

Der beschriebene Ablauf ist beispielhaft in der Abbildung 6-10 dargestellt. Ausgehend von einem *PartialModelQuery*-Objekt wird der rekursive Abstieg bis zur Auswahl eines Planungsstandes und der anschließende Aufstieg mit Auswertung der Selektionsanweisungen gezeigt. In dem Beispiel wird über die Mengenoperation *S1:SetOperation* die Differenzmenge zwischen den Objekten des Planungsstandes 1 (*FirstOperand*) und den Objekten der Objektklasse „Wand“ (*SecondOperand*), die zuvor über die Anweisung *S2:Select* bzw. *S3:SelectInstancesOf* aus den Objekten des Planungsstandes 1 ermittelt wurden, gebildet.

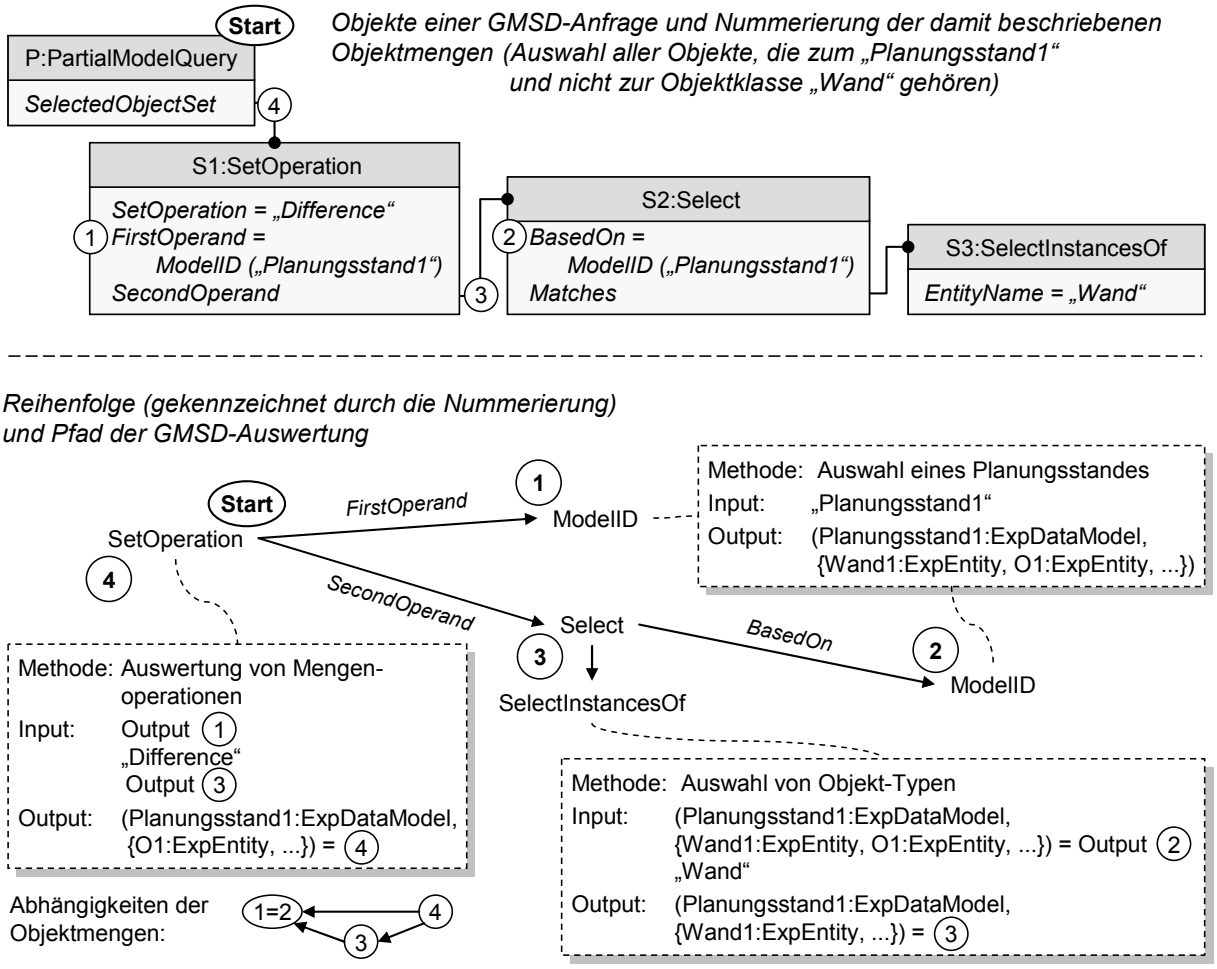


Abbildung 6-10 Auswertung einer Objektselektion, die durch rekursiven Abstieg die gegenseitigen Abhängigkeiten auflöst.

Anwenden der Filterdefinition

Das Ergebnis der Objektselektion wird in einem zweiten Schritt durch eine Filteranweisung ausgewertet. Auf der Grundlage der im ersten Schritt ausgewählten Objekte werden zunächst die tatsächlich gesuchten Objekte sowie die hierauf anzuwendenden Objektfilter bestimmt. Die Zuweisung zwischen Objekt und Objektfilter geschieht durch die Kombination zweier Kriterien: 1.) der Zugehörigkeit zu einer Objektklasse und 2.) der Erreichbarkeit über Verknüpfungen (siehe Definitionen 5.7 und 5.8). Durch diese Kombination können einem Objekt mehrere Objektfilter zugewiesen werden, die in ihrer Gesamtheit den geforderten Informationsbedarf beschreiben. Über das Kriterium der Erreichbarkeit ist neben der Zuweisung von Objektfiltern auch die Auswahl weiterer Objekte möglich. Diese Auswahl wird durch das Auswerten bereits zugeordneter Objektfilter bestimmt. Für neu zugewiesene Objektfilter bzw. zusätzlich ausgewählte Objekte ist das Kriterium der Erreichbarkeit folglich erneut zu überprüfen und führt zwangsläufig zu einem iterativen Verfahren.

In der gewählten Umsetzung wird das Reduzieren der notwendigen Iterationsschritte bzw. der zu wiederholenden Operationen verfolgt. Da die Auswahl weiterer Objekte, also das Erweitern der Objektmenge $O_{AUSWAHL}$, die Filterzuweisung nach der Definition (5.12) direkt beeinflusst, wird vorrangig die Erweiterung der ausgewählten Objektmenge, also die Auswertung der Selektionsrelation, verfolgt. Erst wenn keine weiteren Objekte über die Selektionsrelation auswählbar sind, wird die Filterzuweisung auch für die Modellfilterrelation durchgeführt. Durch diese Filterzuweisung erhalten Objekte unter Umständen neue Objektfilter, die sich in ihrer Definition von den bereits zugewiesenen und ausgewerteten Objektfiltern unterscheiden. Ein weiterer Iterationsschritt ist erforderlich, wenn hierüber neue Objekte ausgewählt werden.

Der iterative Ablauf zum Ermitteln der gesuchten Objekte sowie das Zuweisen der anzuwendenden Objektfilter ist in der Abbildung 6-11 gezeigt. Anknüpfend an das Beispiel der Abbildung 6-10 wird den ausgewählten Objekten, die zusammen mit dem zugehörigen Planungsstand als Objektmenge gegeben sind, zunächst ein Objektfilter zugewiesen. Die Zuweisung geschieht über das Kriterium der Klassenzugehörigkeit und wird aus der gegebenen Filterdefinition *V1:ViewDefinition* ermittelt. Diese Initialisierung ist notwendig, um in den nachfolgenden Schritten zunächst die Selektions- und daran anschließend die Modellfilterrelationen auswerten zu können. Schließlich werden durch das Initialisieren auch die nicht unterstützten Objekte aus der betrachteten Objektmenge entfernt.

Nach dem Initialisieren werden ausgehend von einem Objekt und einem zugeordneten Objektfilter im Sinne der Definition (5.13) rekursiv weitere Objekte ausgewählt und mit Objektfiltern verknüpft. Der rekursive Aufruf wird beendet, sobald ein Objekt mit einem Objektfilter wiederholt verknüpft werden soll und folglich in dieser Kombination bereits ausgewertet wurde. Nach Auswertung der Selektions- und Modellfilterrelationen entscheidet sich, ob die beiden letzten Schritte erneut durchgeführt werden müssen. Sind keine weiteren Iterationsschritte erforderlich, so können die zugewiesenen Objektfilter angewendet und hieraus der geforderte Teildatensatz erzeugt werden. Die ausgewählten Objekte werden bei diesem letzten Schritt entweder unverändert in die Ergebnismenge übernommen oder in ihrem Informationsgehalt reduziert. Ein Reduzieren des Informationsgehaltes erzwingt das Erzeugen eines Nachfolgeobjektes, das die entfernten Informationen als Änderungen abbildet. Ein Attribut wird hierfür entweder vollständig entfernt (Ersetzen durch *DeletedAttribute*) oder im Informationsgehalt reduziert (Ersetzen durch ein anderes Attribut).

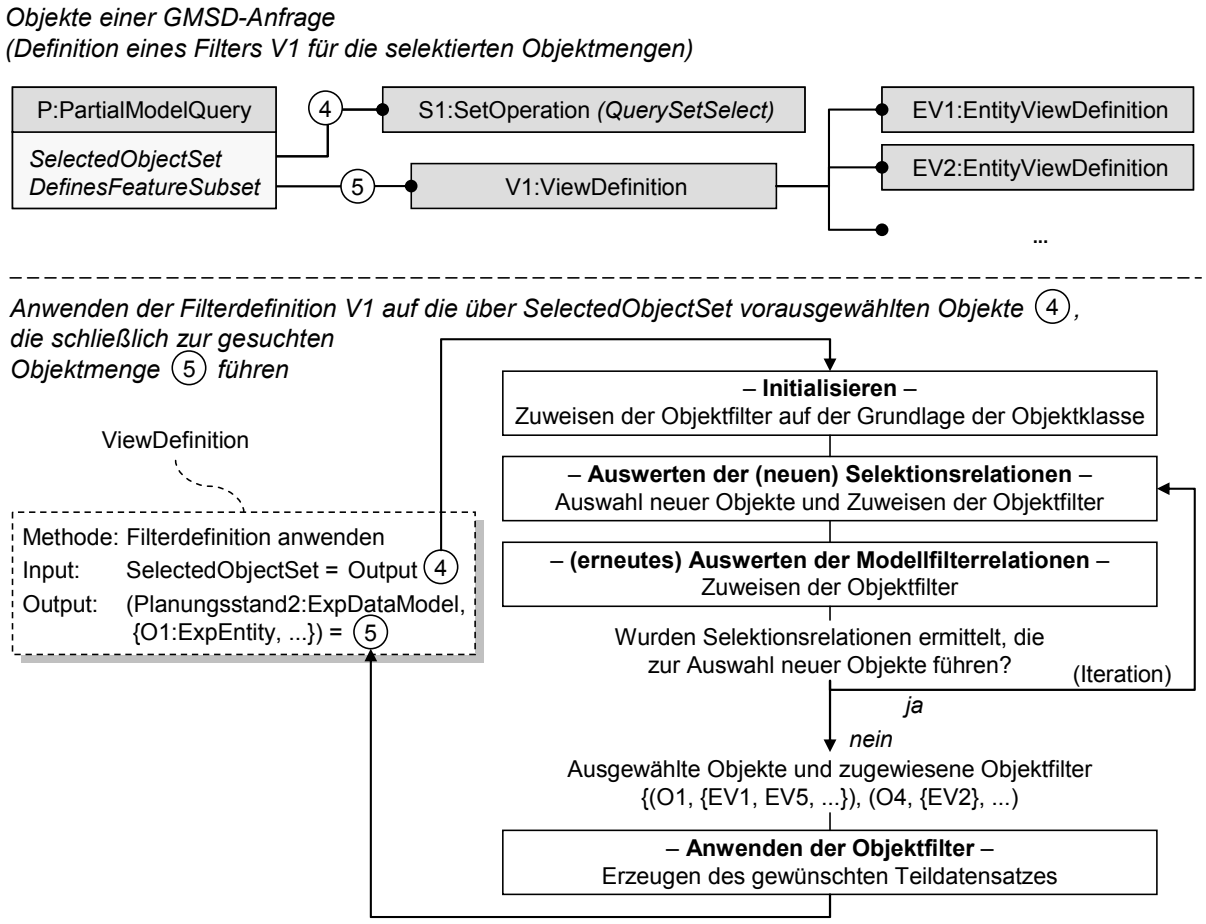


Abbildung 6-11 Ablauf der Filterauswertung, die auf die vorselektierte Objektmenge angewendet wird.

6.3.2 Vergleich von Planungsständen

Mit dem umgesetzten Vergleichsverfahren wird das Ziel verfolgt, einen überarbeiteten, als vollständigen (Teil-)Datensatz vorliegenden Planungsstand in das Versionsmodell zu integrieren. Die ermittelten Änderungen werden demzufolge gemäß dem Versionsansatz durch Delta-Objekte beschrieben. Hieran gekoppelt sind Vorgänger-Nachfolger-Beziehungen, die den Bezug zwischen den geänderten Objekten und den Delta-Objekten herstellen.

Das vorgeschlagene Vergleichsverfahren und der hierfür vorgesehene Ablauf sind in Abschnitt 5.2 beschrieben. Konzeptionell werden zwei Schritte unterschieden: 1.) das Auffinden und 2.) der Vergleich von Objektpaaren. Beide Schritte werden durch den implementierten Algorithmus jedoch nicht strikt voneinander getrennt, sondern miteinander kombiniert. Dadurch wird ein wiederholtes Abfragen von Attributen reduziert, das in Verbindung mit dem betrachteten Versionsmodell die Laufzeit des Modellvergleichs beeinflusst (siehe Abbildung 6-9). Die Laufzeit und das Verhalten des vorgeschlagenen Vergleichsverfahrens sind darüber hinaus von äußeren, nicht beeinflussbaren Faktoren abhängig (siehe Abschnitt 5.2.1). Deshalb wurde die Möglichkeit geschaffen, das Vergleichsverfahren hinsichtlich der erwarteten Qualität, Laufzeit und Erkennungsrate anpassen zu können. Hierfür werden Steuergrößen verwendet, die einerseits die nutzbaren Kriterien für das Auffinden der Objektpaare festgelegen und andererseits die Häufigkeit der durchzuführenden Iterationen beeinflussen. Da die Güte des Vergleichsergebnisses auf verschiedenen, voneinander unabhängigen heuristischen Annah-

men beruht, sind die hierfür verwendeten Programmteile getrennt voneinander implementiert und können bei Bedarf individuell angepasst werden.

Die zu vergleichenden Planungsstände werden als „Vorgänger“ und „Nachfolger“ über ihren eindeutigen Namen gegeben. Mit dem Ziel, den „Nachfolger“ in das Versionsmodell zu integrieren und die Änderungen gegenüber dem „Vorgänger“ als Delta abzuspeichern, müssen die zum „Nachfolger“ gehörenden Objekte den zwangsläufig schreibenden Datenzugriff erlauben. Andernfalls ist es nicht möglich, die Objekte des „Nachfolgers“ in Delta-Objekte umzuwandeln und mit den geänderten Objekten zu verbinden.

6.3.2.1 Auffinden der Objektpaare

Für das Auffinden der Objektpaare wird ein iteratives Verfahren genutzt, das ausgehend von gefundenen Objektpaaren weitere Objektpaare sucht (siehe Abbildung 6-12). Dementsprechend ist eine anfängliche Initialisierung notwendig, die, sofern vorhanden, über eindeutige Objektschlüssel realisiert wird. Die Zuordnung über Objektschlüssel ist von dem verwalteten Datenmodell abhängig und erfordert eine hierauf abgestimmte Methode. Im Rahmen der prototypischen Umsetzung wird die Zuordnung über einen *Surrogat*-Schlüssel, der in den Objekten als Attribut abgebildet wird, unterstützt. Für diese datenmodellspezifische Festlegung wird neben den bereits erwähnten Steuergrößen die Angabe des Schlüsselattributs benötigt, die sich aus dem Attributnamen und der Objektklasse zusammensetzt.

Zuordnung über korrespondierende Verknüpfungen vorhandener Objektpaare

Existieren Objekte, die kein oder kein belegtes Schlüsselattribut besitzen, so werden die Verknüpfungen bereits zugeordneter Objekte ausgewertet. Für jeden der drei Verknüpfungstypen wird hierfür eine eigene Methode bereitgestellt. Eine solche Methode implementiert die in Abschnitt 5.2.4 beschriebenen Zuordnungsregeln und leitet aus der Menge der nicht zuordenbaren Objekte UO vermutete Objektpaare ab. Die hierbei gefundenen Objektpaare werden anschließend in die Menge der vermuteten Objektpaare VP eingetragen und je nach Verknüpfungstyp mit einem „Wichtungswert“ kombiniert. Die Menge der vermuteten Objektpaare VP wird dadurch in drei Teilmengen zerlegt. Zusätzlich wird vermerkt, welche Verknüpfung zu einem vermuteten Objektpaar geführt hat. Mit dieser Angabe ist es in den nachfolgenden Schritten möglich, die vermuteten Objektpaare besser zu bewerten und wiederholte Attributvergleiche zu vermeiden.

Für die Auswertung der Verknüpfungen kann über eine Steuergröße zusätzlich festgelegt werden, welche Verknüpfungstypen für das Ermitteln vermuteter Objektpaare berücksichtigt werden sollen. Hierdurch ist es möglich, die Qualität des Vergleichsergebnisses zu beeinflussen, indem beispielsweise „unsichere“ Arten der Zuordnung von der Auswertung ausgeschlossen werden[1]. Neben einem generellen Verzicht auf bestimmte Verknüpfungstypen kann das Vergleichsergebnis auch dadurch beeinflusst werden, indem erst dann „unsichere“ Verknüpfungstypen berücksichtigt werden, wenn über bevorzugt behandelte Verknüpfungstypen keine weiteren Objektpaare gefunden werden können. Auf diese Weise ist es beispielsweise möglich, erst alle Einfachreferenzen zu berücksichtigen, bevor geordnete und schließlich un-

[1] Für das Festlegen der auswertbaren Verknüpfungstypen werden folgende vier Fällen unterschieden: 1.) keine Zuordnung über Verknüpfungen (Zuordnung nur über Objektschlüssel und somit keine falschen Objektpaare), 2.) Einfachreferenzen, 3.) Einfachreferenzen und geordnete Mengenreferenzen und 4.) Auswertung aller drei Verknüpfungstypen.

geordnete Mengenreferenzen ausgewertet werden. Dieser Eingriff kommt der Veränderung des Wichtungskonzeptes gleich, das beim Überführen der vermuteten in gesicherte Objektpaare relevant wird.

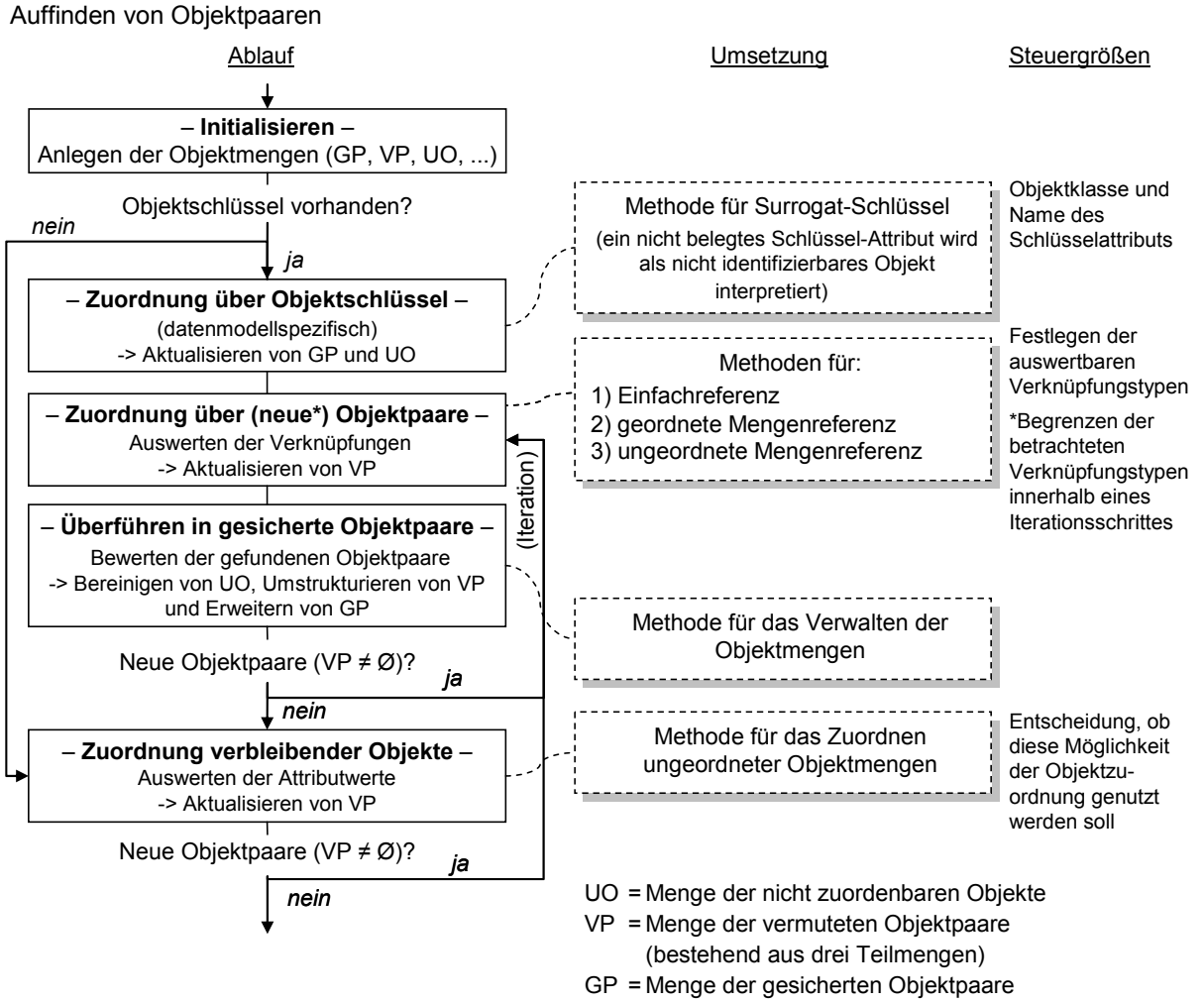


Abbildung 6-12 Ablauf, Umsetzung und Steuergrößen beim Zuordnen von Objekten

Überführen in gesicherte Objektpaare

Nach Auswertung der Verknüpfungen steht eine aktualisierte Menge mit vermuteten Objektpaaren zur Verfügung. Nach ihrem Zustandekommen setzt sich diese Menge aus drei Teilmengen zusammen. Aus diesen Teilmengen werden vermutete Objektpaare in die Menge der gesicherten Objektpaare überführt und auf diese Weise ein neuer Iterationsschritt vorbereitet. Im Rahmen der prototypischen Umsetzung wird durch die Zugehörigkeit zu einer der Teilmengen entschieden, welche der vermuteten Objektpaare akzeptiert, bis zum nächsten Iterationsschritt „zurückgestellt“ oder verworfen werden. In der Abbildung 6-13 ist das Überführen in gesicherte Objektpaare dargestellt. Durch das Ermitteln der vermuteten Objektpaare werden zunächst die drei Teilmengen der Menge VP belegt. Das Überführen in gesicherte Objektpaare wird anschließend durch das Übertragen von Objektmengen realisiert, das bis zu drei Iterationsschritte benötigen kann. Wird ein vermutetes Objektpaar schließlich in die Menge der gesicherten Objektpaare überführt, so sind weitere Anpassungen an den beteiligten Objekten und Objektmengen notwendig. Hierzu gehören:

- das Entfernen der beteiligten Objekte aus der Menge UO (und auch VP),
- das Herstellen der Vorgänger-Nachfolger-Beziehung,
- das Aufbereiten des Nachfolgers, indem nicht belegte Attribute durch den Typ *DeletedAttribute* beschrieben werden,
- das Verwerfen der Objektpaare, die für die beteiligten Objekte zusätzlich, jedoch mit einer geringeren Wichtung ermittelt wurden sowie
- das Überprüfen des(r) verursachenden Attributs(e), das bei Gleichheit und Einhalten der Konsistenzbedingungen vom Nachfolger[1] des gesicherten Objektpaares entfernt werden kann.

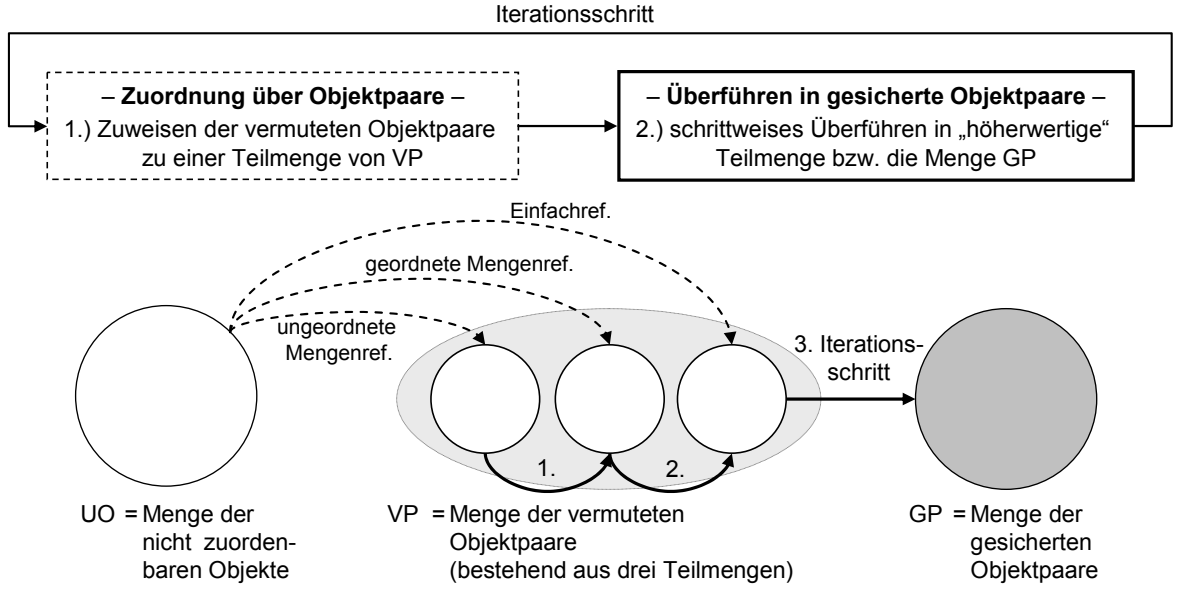


Abbildung 6-13 Überführen der vermuteten in gesicherte Objektpaare

Die Ergänzung durch den Typ *DeletedAttribute* ist durch das Herstellen der Vorgänger-Nachfolger-Beziehung erforderlich. Erst dadurch kann zwischen *nicht belegten* und unveränderten Attributen unterschieden werden. Wird diese Ergänzung nicht vorgenommen, so wird für ein nicht belegtes Attribut das Attribut des zugewiesenen Vorgängers übernommen. Mit den beschriebenen Anpassungen wird nicht zuletzt das Ziel verfolgt, die Vorgänger-Nachfolger-Beziehung für die beteiligten Objekte endgültig festzulegen. Dies wird durch das Entfernen der Objekte aus der Menge UO und dem Verwerfen anderer Objektpaare erreicht. Damit wird die Voraussetzung geschaffen, um das Ermitteln der Objektpaare mit dem Vergleich der Objektpaare zu kombinieren. Hierbei wird das für das Finden des Objektpaares verantwortliche Attribut bewertet. Dies ist aber nur dann möglich, wenn für die betrachteten Objekte die Vorgänger-Nachfolger-Beziehungen nicht mehr verändert werden. Durch dieses

[1] Ein vermutetes Objektpaar wird durch den Vergleich zweier Attribute (Verknüpfungen) ermittelt, die aus dem Vorgänger bzw. Nachfolger eines gesicherten Objektpaares abgeleitet werden. Nach dem vorgestellten Versionsmodell kann die Verknüpfung des Nachfolgers durch die Verknüpfung des Vorgängers ersetzt werden, wenn das verknüpfte Objekt mit der Angabe eines entsprechenden Planungsstandes eindeutig ermittelt werden kann. Wird das vermutete Objektpaar in ein gesichertes Objektpaar umgewandelt, so ist durch die Vorgänger-Nachfolger-Beziehung die Voraussetzung für das Ersetzen der Verknüpfung erfüllt. Bei der Zuordnung über gesicherte Objektpaare können somit auch Attribute ersetzt werden, die unter Einhaltung der Konsistenzbedingungen auf ein gesichertes Objektpaar verweisen bzw. den gleichen Attributzustand besitzen.

Vorgehen wird schließlich der Aufwand für nachfolgende Iterationsschritte verringert und letztendlich die Vergleichsgeschwindigkeit erhöht. Ein Nebeneffekt ist, dass ein Objekt nur dann mit mehreren Objekten über die Vorgänger-Nachfolger-Beziehung verbunden wird (Auffinden von zusammengelegten bzw. geteilten Objekten), wenn die hierfür erforderlichen Vermutungen den gleichen Wichtungswert besitzen.

Zuordnung verbleibender Objekte

Konnte die Menge der gesicherten Objektpaare durch das Überführen der vermuteten Objektpaare erweitert werden, so kann ein neuer Iterationsschritt beginnen. Andernfalls wird überprüft, ob die Menge UO leer ist. Sind keine Elemente in der Menge UO vorhanden, so wird die Zuordnung der Objekte beendet. Ist die Menge UO nicht leer, so wird für die darin verbliebenen Objekte versucht, eine Zuordnung durchzuführen. Die damit verbundene Problemstellung ist mit der Zuordnung ungeordneter Mengenreferenzen vergleichbar. Dennoch wird ein erweitertes Verfahren umgesetzt, das nicht nur den Objektzustand über eine Prüfsumme, sondern auch die gegenseitigen Verknüpfungen berücksichtigt. Nach dem in Abschnitt 5.2.4 beschriebenen Verfahren werden zunächst alle Objekte ermittelt, die nicht von anderen Objekten referenziert werden. Hierfür wird das Objektnetz[1] der nicht zuordenbaren Objekte über die Verknüpfungsbeziehung sortiert. Für beide Planungsstände wird hierbei im Idealfall jeweils eine nicht leere Objektmenge bestimmt. Aus diesen beiden Objektmengen werden anschließend Objektpaare nach dem gleichen Verfahren wie für ungeordnete Mengenreferenzen abgeleitet. Als Zuordnungskriterium wird der Objektzustand verwendet, der nur dann den Zustand der verknüpften Objekte einbezieht, falls Mehrdeutigkeiten eine eindeutige Objektzuordnung verhindern. Treten trotz der Berücksichtigung verknüpfter Objekte Mehrdeutigkeiten auf, so wird eine zufällige Zuordnung gewählt. Die zufällige Zuordnung wird aber auf Objekte begrenzt, die voneinander unabhängig sind.

Wird durch das Sortieren über die Verknüpfungsbeziehung eine leere Objektmenge zurückgegeben, so erfolgt die Zuordnung aufgrund des Objektzustandes und wird für voneinander unabhängige Objektmengen auf *ein* Objektpaar beschränkt. Auf diese Weise wird sichergestellt, dass die zufällige Zuordnung von Objekten nicht im Widerspruch zum Objektnetz steht.

6.3.2.2 Integration der Änderungen in das Versionsmodell

Der Objektvergleich wurde mit dem Auffinden der Objektpaare kombiniert, sodass nicht nur die Vorgänger-Nachfolger-Beziehungen sondern auch die geänderten Attribute bzw. Verknüpfungen ermittelt wurden und als (Delta-)Objekte vorliegen. Neben diesen geänderten Objekten verbleiben in der Menge UO unter Umständen Objekte, die nicht zugeordnet werden konnten. Diese Objekte stellen im Sinne des Vergleichs neue bzw. gelöschte Objekte dar. Für neue Objekte wurde bis jetzt aber noch keine Überprüfung der Attribute vorgenommen. Wird über ein Attribut eines neuen Objektes auf ein unverändertes Objekt verwiesen, so wird die entsprechende Verknüpfung auf das unveränderte Vorgängerobjekt gesetzt. Das Ersetzen der Verknüpfungen ist aber auch hier nur möglich, wenn dadurch die Konsistenzbedingung (4.21) nicht verletzt wird.

Als Vergleichsergebnis liegen auf Objektebene somit die zugeordneten, gelöschten und neuen Objekte vor. Dieses Ergebnis wird durch einen neuen Planungsstand dokumentiert. Der neue Planungsstand wird hierfür in Bezug zu dem verglichenen, im Versionsmodell bereits integ-

[1] Das Objektnetz ist ein gerichteter zyklischer Graph.

rierten und somit unveränderbaren Planungsstand gesetzt. Von den erzeugten (Delta-)Objekten, die unter Umständen keine Attribute enthalten und somit im Vergleich zu ihrem Vorgänger keine Änderungen aufweisen, werden zuvor die tatsächlich benötigten Objekte bestimmt. Alle leeren und somit unveränderten (Delta-)Objekte, die nicht zur Wahrung der Konsistenzbedingungen benötigt[1] werden, werden durch ihre Vorgänger ersetzt und durch Auflösen ihrer Vorgänger-Nachfolger-Beziehung aus dem Versionsmodell entfernt. Von den zugeordneten Objekten bleiben auf diese Weise nur „veränderte“ Objekte übrig. Mit dem Bezug zum verglichenen Planungsstand werden die veränderten Objekte schließlich durch ein entferntes (*Removes*) und ein neues Objekt (*Adds*) ausgedrückt.

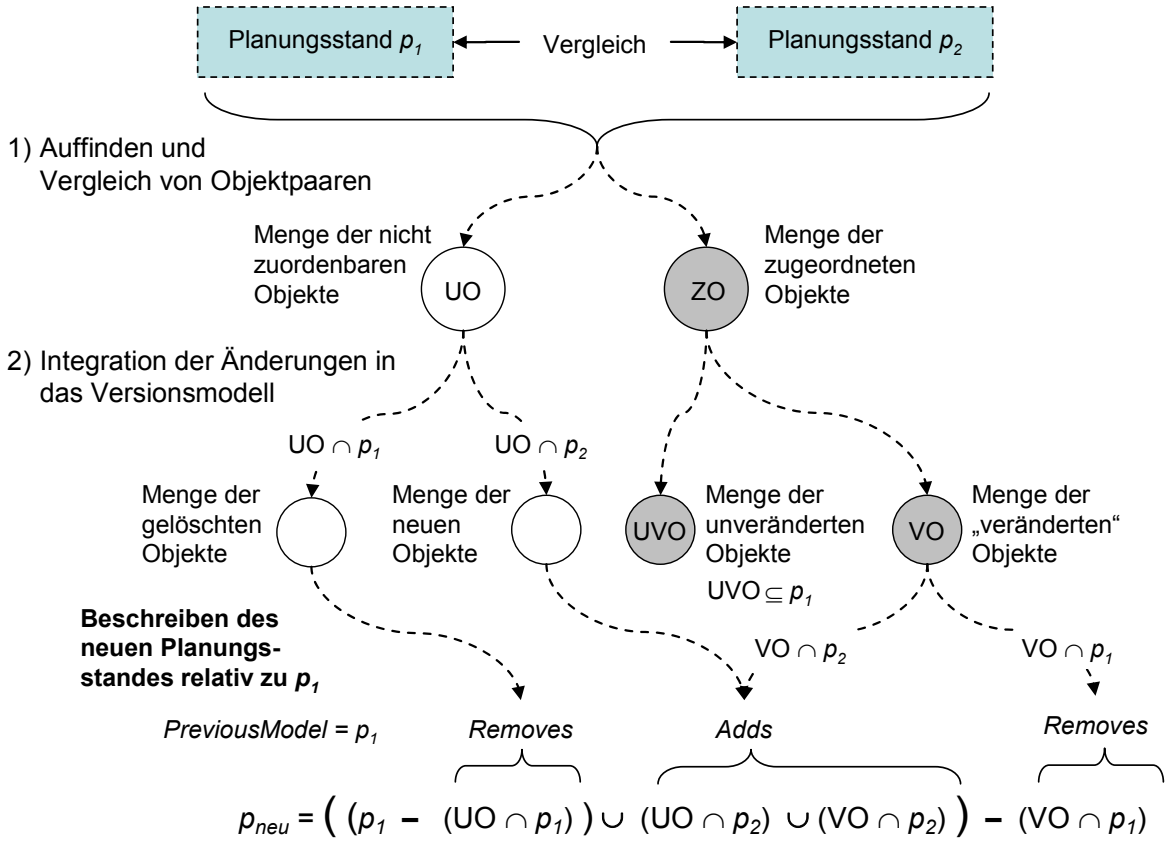


Abbildung 6-14 Aufbereiten der gefundenen Änderungen zu einem neuen Planungsstand

Das Aufbereiten der gefundenen Änderungen und das Abbilden in einen Planungsstand sind in der Abbildung 6-14 gezeigt. In diesem Beispiel werden die beiden Planungsstände p_1 und p_2 miteinander verglichen, wobei p_1 der integrierte, unveränderbare Planungsstand ist. Über das Auffinden und den Vergleich der Objektpaare werden zunächst die beiden Mengen UO und ZO (Menge der zuordenbaren Objekte) erzeugt. Diese Mengen werden, wie zuvor beschrieben, in disjunkte Teilmengen für gelöschte, neue, unveränderte und veränderte Objekte weiter unterteilt. Wird der übergebene Planungsstand p_2 als Änderung gegenüber p_1 ausgedrückt, so können diese Teilmengen direkt für die hinzugefügten (*Adds*) und entfernten (*Removes*) Objekte genutzt werden.

[1] Ein leeres (Delta-)Objekt wird benötigt, sobald es über ein *gültiges* Attribut referenziert wird.

6.3.3 Re-Integration geänderter Teildatensätze

Mit der Re-Integration wird das Ziel verfolgt, einen geänderten Teildatensatz in die gemeinsame Datenbasis zu integrieren. Dieser Schritt einer Transaktion wird mit der Problematik der Datenkonsistenz konfrontiert, die auf unterschiedlichen „Ebenen“ verletzt sein kann. Neben Inkonsistenzen mit dem Versionsmodell können Konflikte auftreten, die aus Widersprüchen mit dem zugrunde liegenden Datenmodell (*database inconsistency*) oder fachlichen Problemen (*design inconsistency*) resultieren. Bei dem in Abschnitt 5.3 beschriebenen formalen Verfahren wird die Konsistenz des Versionsmodells berücksichtigt. Konflikte mit dem Versionsmodell werden hierbei nach vorgegebenen Mustern aufgelöst und erzeugen ohne zusätzliche Anwenderinteraktionen einen neuen Planungsstand.

Aufbereiten der Objektmengen

Für die Re-Integration sind drei Planungsstände gegeben. Diese Planungsstände beschreiben die ersten beiden Teilschritte eines Planungsschrittes. Hierzu gehören der zugrunde liegende Planungsstand p_1, der verwendete Teildatensatz $p_{T.1}$ und schließlich der geänderte Teildatensatz $p_{T.2}$. Unter diesen Voraussetzungen werden zwischen den Planungsständen p_1 und $p_{T.1}$ zunächst die geänderten[1] und entfernten Objekte ermittelt. Zwischen den Planungsständen $p_{T.1}$ und $p_{T.2}$ werden zusätzlich alle Änderungen bestimmt, die zu einem Konflikt mit dem Versionsmodell führen können. Hierzu gehören Objekte, die entweder gelöscht, geteilt, zusammengelegt oder in ihrer Klassenzugehörigkeit verändert wurden. Als Ausgangspunkt für das Wiederherstellen der entfernten Informationen liegen durch diese Unterteilung die in Abbildung 6-15 gezeigten Objektmengen vor. Auf dieser Basis ist ein schnelles Erkennen von Inkonsistenzen möglich.

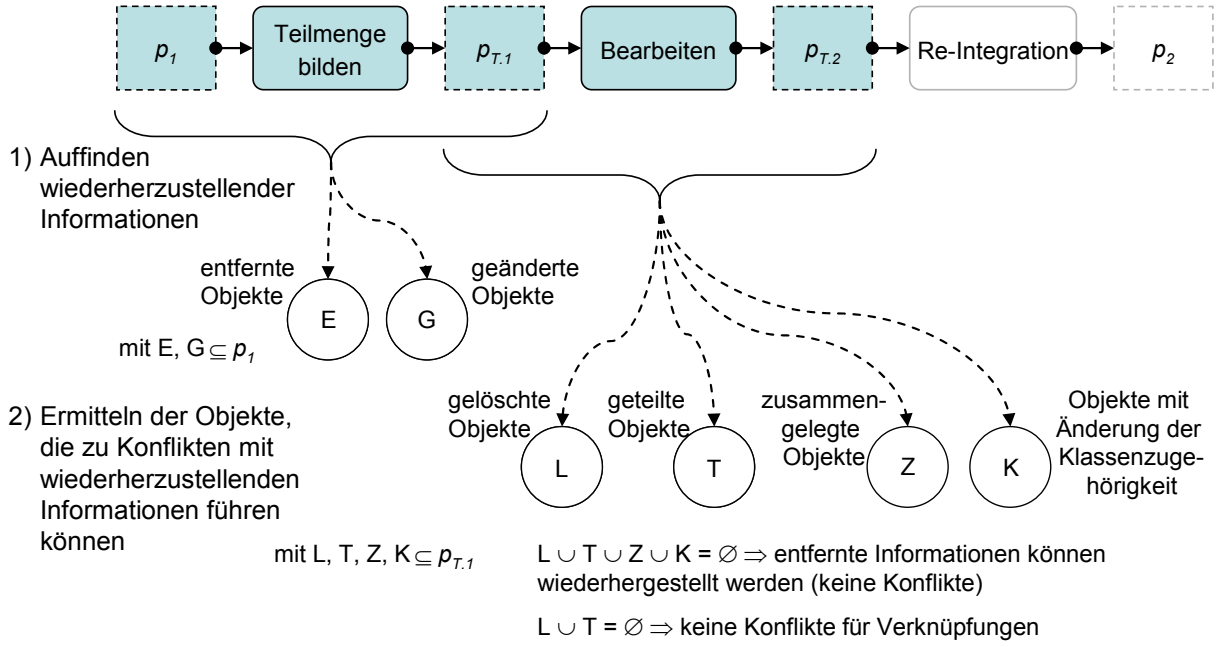


Abbildung 6-15 Aufbereiten der gegebenen Planungsstände zum Wiederherstellen entfernter Informationen

[1] Die Änderung wird in diesem Fall durch das Entfernen von Informationen hervorgerufen.

Erkennen der Inkonsistenzen

Für das Wiederherstellen der Informationen werden die entfernten und geänderten Objekte bewertet. Hierbei wird kontrolliert, ob die in $p_{T.1}$ entfernten Informationen in Konflikt zu dem geänderten Planungsstand $p_{T.2}$ stehen. Wird ein Konflikt erkannt, so ist ein automatisiertes Wiederherstellen des betrachteten Attributs nicht möglich und führt zwangsläufig zum Verlust der hierdurch beschriebenen Information. Die Ursache eines Konfliktes ist entweder

- eine mehrdeutige oder ungültige Verknüpfung, die durch eine Referenz auf ein geteiltes bzw. gelöschtes Objekt entsteht, oder
- eine nicht (eindeutig) wiederherstellbare Information, die aus Änderungen im Objektgefüge resultiert und nur bei veränderten Objekten auftreten kann.

Für das Erkennen einer mehrdeutigen oder ungültigen Verknüpfung werden die Mengen der geteilten und gelöschten Objekte (E, G) bzw. die Verknüpfung mit diesen Objekten überprüft. Sind beide Mengen leer, so sind keine Konflikte mit dem Versionsmodell vorhanden. Ist eine der beiden Mengen nicht leer, so wird für alle wiederherzustellenden Verknüpfungen überprüft, ob ein geteiltes oder gelöschtes Objekt referenziert wird. Eine ähnliche Abhängigkeit ist für das Erkennen nicht wieder herstellbarer Informationen formuliert. Eine solcher Fall tritt ein, wenn ein Objekt aus der Menge G bei der Bearbeitung gelöscht, mit einem anderen Objekt zusammengelegt oder in mehrere Objekte geteilt[1] wurde. Eine Information ist schließlich auch dann nicht wieder herstellbar, wenn ein Objekt die Zugehörigkeit zu einer Objektklasse derart geändert hat, dass ein entferntes Attribut nicht mehr abgebildet werden kann.

Bestimmen der wiederherzustellenden Informationen

Bevor die wiederherzustellenden Informationen auf Inkonsistenzen bezüglich des Versionsmodells überprüft werden, werden die entfernten und geänderten Attribute bestimmt. Die Attribute der geänderten Objekte (Objekte der Menge G) werden mit den Attributen ihrer Nachfolger verglichen, um schließlich die entfernten und nachfolgend zu überprüfenden Verknüpfungen zu ermitteln. Das Attribut eines Objektes wurde geändert, wenn der Nachfolger das Attribut neu belegt. Für diese Belegung gibt es zwei Möglichkeiten, entweder eine Instanz der Klasse *DeletedAttribute*, die ein vollständig entferntes Attribut beschreibt, oder eine Instanz einer von *ExpAggregation* oder *ExpType* abgeleiteten Klasse, die folglich ein Attribut beschreibt, das nur um einen Teil der hierüber beschriebenen Verknüpfungen reduziert wurde.

Wiederherstellen der Informationen

Nach diesem Schritt sind die wiederzustellenden Informationen bestimmt, die sich in

- Objekte,
- Attribute und
- Verknüpfungen

aufteilen. Die kleinste zu überprüfende Einheit ist die Verknüpfung, die zu einem Attribut und somit zu einem Objekt zugeordnet ist. Hieraus ergeben sich folgende Abhängigkeiten:

[1] Für ein zerlegtes Objekt ist ein Wiederherstellen von Informationen durch das Duplizieren der Attribute möglich. In der Umsetzung wird auf diese Vervielfältigung von Informationen jedoch verzichtet.

- Ein *Objekt* kann unverändert in den neuen Planungsstand übernommen werden, wenn alle seine Attribute unverändert übernommen werden können. Andernfalls wird ein Nachfolgeobjekt abgeleitet, das die zu verändernden Attribute neu definiert.
- Ein *Attribut* kann unverändert übernommen werden, wenn alle hierüber beschriebenen Verknüpfungen unverändert übernommen werden können. Andernfalls wird ein neues Attribut erzeugt, das nur Verknüpfungen enthält, die unveränderbar übernommen werden können.
- Eine *Verknüpfung* kann unverändert übernommen werden, wenn dadurch kein Widerspruch mit dem neuen Planungsstand entsteht. Andernfalls wird die betroffene Verknüpfung nicht wiederhergestellt.

Für alle Objekte der Menge E wird folglich das Objektkriterium, für alle vollständig entfernten Attribute das Attributkriterium und schließlich für alle selektiv entfernten Verknüpfungen das Verknüpfungskriterium überprüft. Durch das Anwenden dieser Kriterien leiten sich für die Objekte aus E und dem Planungsstand $p_{T.1}$ unter Umständen neue Objekte ab, die gemeinsam mit den unverändert nutzbaren Objekten den neuen Planungsstand $p_{T.2}$ definieren.

6.3.4 Zusammenführen divergierender Planungsstände

Analog zur Unterstützung der Re-Integration wird beim Zusammenführen divergierender Planungsstände ein automatisiertes Verfahren verfolgt, das die Konflikte mit dem Versionsmodell nach vordefinierten Regeln „auflöst“. Für das Zusammenführen seien die beiden divergierenden Planungsstände p_X und p_Y sowie der gemeinsame Ursprung p_A gegeben. Aus diesen gegebenen Planungsständen wird ein neuer Planungsstand p_Z erzeugt, der auf der Grundlage des Planungsstandes p_X alle konfliktfreien Änderungen von p_Y übernimmt. Da Änderungen in der Regel von anderen Änderungen abhängig sind, werden im Rahmen der Umsetzung die als voneinander abhängig erkennbaren Änderungen entweder vollständig übernommen oder gemeinsam verworfen.

Bilden disjunkter Änderungsmengen

Für den Planungsstand p_Y werden zunächst die Änderungen gegenüber p_A ermittelt, die zu gelöschten, geänderten und neuen Objekten führen. Alle geänderten Objekte werden weiter in geteilte, zusammengelegte und in ihrer Klassenzugehörigkeit geänderte Objekte differenziert. Für die Änderungen werden anschließend die gegenseitigen Abhängigkeiten ermittelt, die zu disjunkten Änderungsmengen führen. Abhängigkeiten zwischen Änderungen entstehen dabei durch:

- das Löschen von Objekten, wenn dadurch das Löschen von Verknüpfungen oder Objekten notwendig wird,
- das Ändern oder Erzeugen von Objekten, wenn dadurch die Änderung von Verknüpfungen notwendig wird, oder
- das Ändern von Attributen, das stets ein „Verändern“ von Objekten bewirkt.

Übernehmen der konfliktfreien Änderungsmengen

Für jede Änderungsmenge wird überprüft, ob eine hierin enthaltene Änderung zu einem Konflikt mit Planungsstand p_X führt und somit das Verwerfen der davon abhängigen Änderungen erzwingt. Tritt kein Konflikt auf, so werden die Änderungen einer betrachteten Änderungs-

menge in den Planungsstand p_Z übertragen. Um einen Konflikt mit dem Planungsstand p_X zu erkennen, werden die Änderungen zwischen den Planungsständen p_X und p_Z nach den in Abschnitt 5.4.2 beschriebenen Kriterien bewertet. Auch hier werden für den Planungsstand p_X die Änderungen gegenüber p_Z in verschiedenen Mengen zusammengefasst, die mit Hilfe einfacher Mengenoperationen zum Erkennen von tatsächlichen und potenziellen Konflikten auf Objekt- und Attributebene genutzt werden. Die Konflikte auf Objektstrukturebene werden dagegen durch die gebildeten Änderungsmengen erkannt, die zusammenhängende Änderungen zusammenfassen.

Der prinzipielle Ablauf für das Zusammenführen ist in der Abbildung 6-16 gezeigt. In den Schritten 1, 2 und 3 werden zunächst die Änderungen wie beschrieben aufbereitet. Hieraus sind im Schritt 4 die übertragbaren Änderungen erkennbar. Das Übertragen der Änderungen erzeugt schließlich einen neuen Planungsstand p_Z, der relativ zum Planungsstand p_X beschrieben wird.

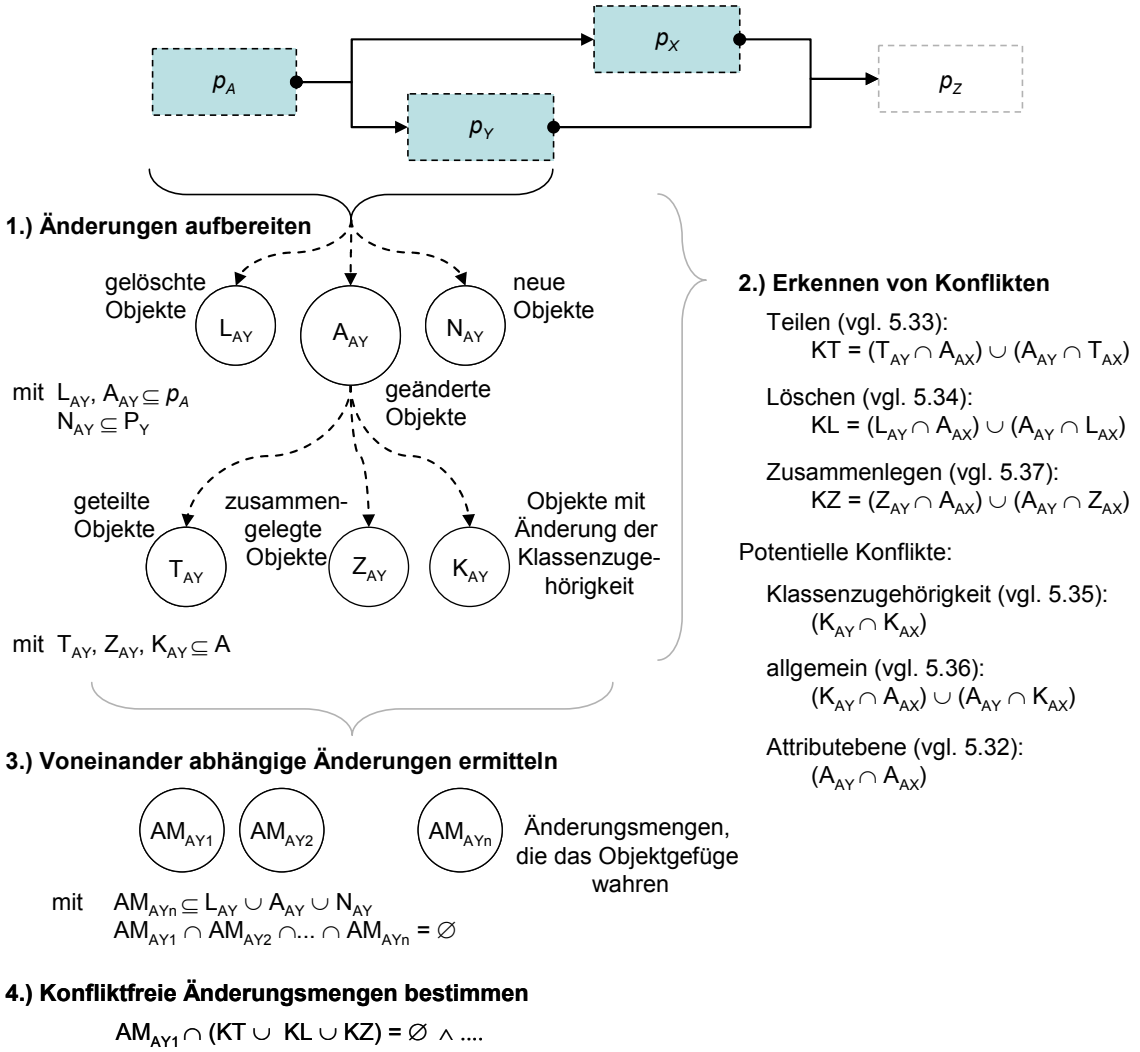


Abbildung 6-16 Aufbereitung der divergierenden Planungsstände für das Zusammenführen

7 Anwendung des entwickelten Ansatzes

Das folgende Kapitel beschreibt die Anwendung des entwickelten Ansatzes. Als Grundlage der Untersuchung wird das IFC-Modell genutzt, das als fachübergreifendes Bauwerksmodell (*Building Information Model*) das Konzept der gemeinsamen Datenbasis begünstigt. Mit der Anwendung des IFC-Modells wird gezeigt, dass die entwickelten generischen Methoden unter praxisnahen Bedingungen vorteilhaft anwendbar sind. Der Praxisbezug wird durch die Nutzung verfügbarer CAD-Anwendungen und realer Datensätze erreicht.

Der *erste Abschnitt* beschreibt das zur Validierung des erarbeiteten Lösungsansatzes verwendete IFC-Anwendungsszenario. Für dieses typische Anwendungsszenario werden die Probleme der Praxis skizziert und dem vorgeschlagenen Lösungsansatz gegenübergestellt. Es wird gezeigt, wie die entwickelten Werkzeuge in Ergänzung zu kommerziellen CAD-Programmen und auf der Grundlage eines dateibasierten Datenaustauschs genutzt werden können. Durch den Umgang mit Teildatensätzen wird schließlich ein Konzept für die Implementierung der sogenannten *IFC-Views* vorgestellt.

Im *zweiten Abschnitt* werden die Anwendung der entwickelten Methoden und die dabei gewonnenen Erkenntnisse diskutiert. Im Mittelpunkt stehen die Definition der benötigten Teildatensätze, die Qualität der Vergleichsergebnisse und schließlich die Brauchbarkeit der Datensätze, die bei der Re-Integration und beim Zusammenführen entstehen. Es wird gezeigt, dass die erarbeiteten Methoden die Anwendung des IFC-Modells erleichtern und unter idealen Voraussetzungen auch ohne korrigierende Eingriffe anwendbar sind. Für die praktische Anwendung ist die Kontrolle der Ergebnisse jedoch unverzichtbar, um mögliche Fehler erkennen und korrigierend eingreifen zu können.

Der *dritte Abschnitt* löst sich von dem Beispiel des IFC-Modells und weitet die Untersuchung auf andere Datenmodelle aus. Die Bewertung der Datenstrukturen zeigt, dass ähnliche Ergebnisse wie beim IFC-Modell zu erwarten sind. Damit werden die Vorteile des generischen Ansatzes deutlich, der auch hier eine geeignete Grundlage für die Verwaltung gemeinsamer Planungsdaten liefert.

7.1 Anwendung des Ansatzes am Beispiel IFC

Für die Validierung des Lösungsansatzes wird ein Szenario verwendet, das sich an der praktischen Nutzung der IFC-Daten orientiert und den Implementierungsempfehlungen der IAI folgt. Bei dieser Empfehlung werden verschiedene Anwendungsszenarien definiert, die den Datenaustausch auf die hierfür relevanten Teile des Gesamtmodells beschränken und dadurch die Implementierung für hochspezialisierte Anwenderprogramme vereinfachen.

7.1.1 Überblick über das IFC-Modell

Mit dem IFC-Modell wird von der IAI[1], einer internationalen Organisation aus Industrie und Forschung, das Ziel verfolgt, den fachübergreifenden Datenaustausch über ein standardisiertes Produktdatenmodell zu realisieren. Die Integration verschiedener Fachbereiche wird über eine Modellstruktur erreicht, die auf einer Erweiterung von Basiskonzepten und der Wieder-

[1] International Alliance for Interoperability - http://www.iai-international.org

verwendung allgemein gültiger Ressourcen beruht. Für diese Modularisierung werden in dem IFC-Modell vier, aufeinander aufbauende Ebenen unterschieden:

1. die wiederverwendbaren Ressourcen, die durch Referenzierung genutzt werden und beispielsweise die Geometrie oder Materialeigenschaften kapseln (*Resource Layer*),
2. abstrakte oder allgemeingültige Basisklassen, die in der Regel durch Vererbung zu nutzbaren Klassen erweitert werden und Eigenschaften wie Identität, Objektgruppierung oder Projektinformationen beschreiben (*Kernel Layer*),
3. fachübergreifend nutzbare Klassen, aus denen Objekte für eine allgemein verständliche Kommunikation über ein Bauwerk erzeugt werden und unter anderem Klassen wie Wand oder Stütze definieren (*Interoperability Layer*) und
4. fachspezifische Klassen, die den Austausch problembezogener Informationen ermöglichen und beispielsweise die gewählte statische oder klimatechnische Modellierung eines Bauwerks beschreiben können (*Domain Layer*).

Neben diesen vier Ebenen wird zusätzlich eine Unterteilung in einzelne Schemata verfolgt, die innerhalb einer Ebene eine Abgrenzung von Ressourcen, Basisfunktionen oder Fachbereichen erlauben. Auf dieser Grundlage, die durch zusätzliche Einschränkungen bezüglich der Spezialisierung von Klassen (Einfachvererbung und Substitutionsprinzip von Liskow) sowie der gegenseitigen Referenzierung (*ladder principle* und Nutzung von Beziehungsobjekten) reglementiert ist, wird schließlich ein modularer Charakter des Gesamtmodells erreicht (Wix & See 1999).

Austausch von Planungsdaten

Die Unterteilung des IFC-Modells in verschiedene Ebenen und Schemata ist weniger für den Datenaustausch sondern vielmehr für die Pflege der Modelldefinition relevant. Der Datenaustausch wird statt dessen auf der Grundlage von Teildatensatzbeschreibungen organisiert. Mit diesen problemspezifischen Modellsichten, die in der IAI als *Views* bezeichnet werden und neben den benötigten Klassen auch die zu unterstützenden Attributdefinitionen angegeben, ist im Vergleich zur Konfiguration verschiedener Schemata eine wesentlich genauere Beschreibung der relevanten Teildatensätze möglich.

Der Umgang mit Teildatensätzen ist derzeit jedoch nur unzureichend formalisiert und kollidiert zusätzlich mit dem dateibasierten Datenaustausch. Ein wesentlicher Kritikpunkt bezieht sich auf die Art der Teildatensatzbeschreibung, die nicht als eine softwaretechnisch auswertbare Spezifikation, sondern als eine von Softwareherstellern zu implementierende Beschreibung vorliegt (Steinmann 2005). Eine auf diese Weise beschriebene Modellsicht wird dadurch zu einer kaum individualisierbaren Lösung, die nicht auf den Bedarf und die Möglichkeiten der einzelnen Anwenderprogramme angepasst werden kann. Ein weiterer Kritikpunkt bezieht sich auf die referentielle Integrität der überarbeiteten Teildatensätze, die in diesem Ansatz nicht geklärt ist. Die Aktualisierung des Gesamtdatensatzes ist aus diesem Grund ein ungelöstes Problem und schränkt infolgedessen die Anwendung des IFC-Modells auf einen einseitig gerichteten Datenaustausch ein.

7.1.2 IFC-Anwendungsszenario

Das betrachtete Anwendungsszenario ist in der Abbildung 7-1 dargestellt und wird für ein CAD-Programm und einen hierfür angepassten Filter, der sich an den *Coordination View* der IAI anlehnt, untersucht. Durch den vordefinierten Filter wird ein Datensatz bereitgestellt, der

den individuellen Fähigkeiten des verwendeten CAD-Programms entspricht und über optionale Objektselektionen weiter eingeschränkt werden kann. Der ausgewählte Datensatz wird anschließend dem CAD-Programm zur Bearbeitung übergeben. Hierfür wird eine IFC-Datei gemäß den STEP Konventionen verwendet (ISO 10303-21), die vom CAD-Programm eingelesen und für die weitere Bearbeitung in die programminterne Datenstruktur umgewandelt wird. Nach der Bearbeitung wird eine neue IFC-Datei erzeugt und dem Server übergeben. Anschließend werden die durchgeführten Änderungen ermittelt und in das Gesamtmodell eingefügt. Zusätzlich ist der neue Planungsstand mit parallel durchgeführten Änderungen abzustimmen und zu einem gemeinsamen Planungsstand zusammenzuführen.

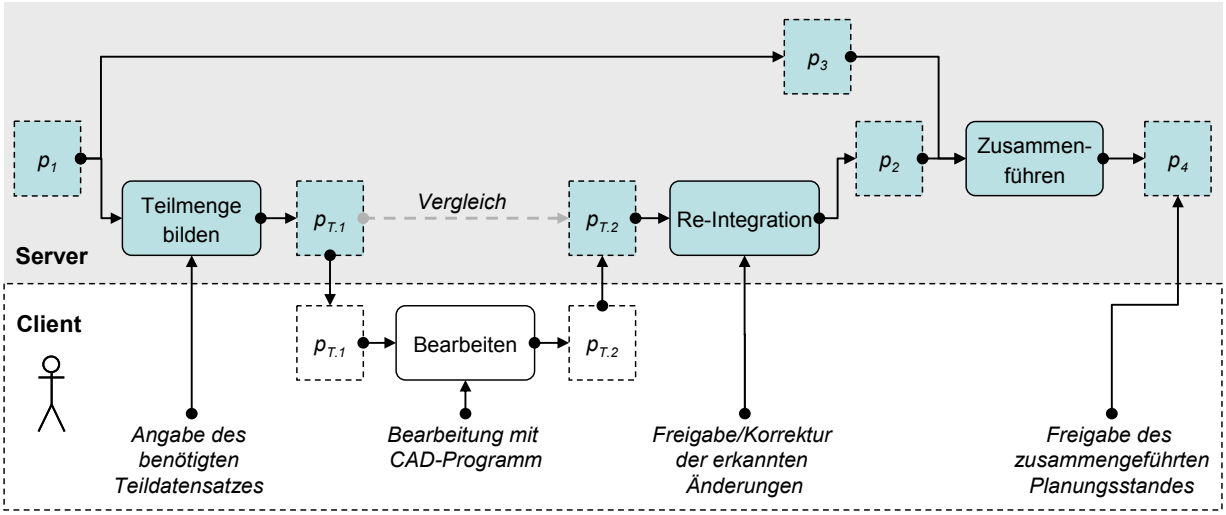


Abbildung 7-1 Prinzipieller Ablauf des Szenarios, Verteilung der verwendeten Daten und Zuordnung der benötigten Funktionalität

Die Interaktion mit dem Server wird über einen Client realisiert, der den Zugriff auf die zentral verwalteten Planungsdaten ermöglicht. Jeder Planungsschritt wird in einer Transaktion gekapselt und verläuft nach dem zuvor beschriebenen Prinzip. Nach der Anmeldung an den Server kann der Anwender entweder eine Transaktion beginnen oder eine laufende Transaktion fortführen. Wird eine Transaktion begonnen, so kann der zu bearbeitende Planungsstand, wie in Abbildung 7-2 dargestellt, durch die Auswahl der entsprechenden Modellversion bestimmt werden.

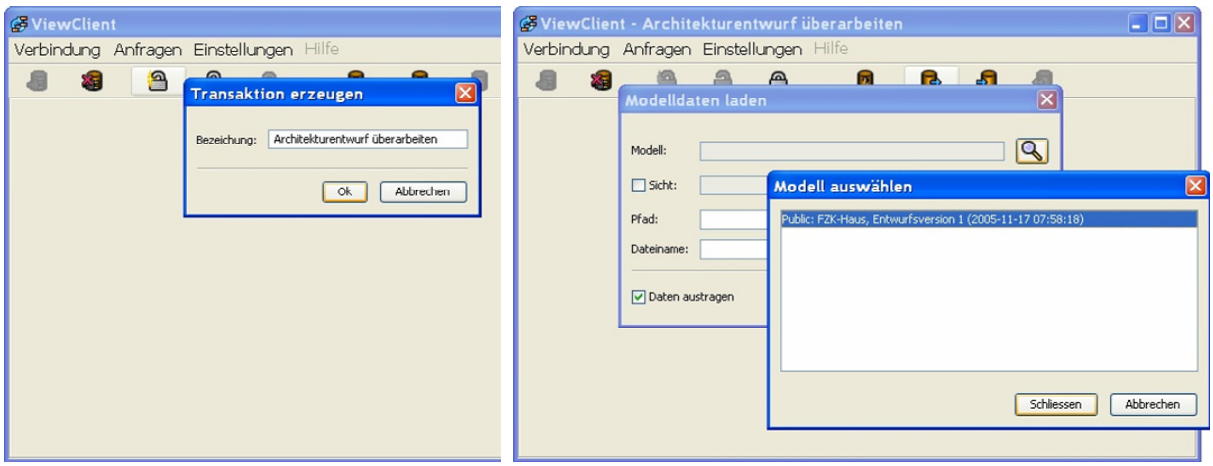


Abbildung 7-2 Transaktion beginnen (links) und zu bearbeitende Modellversion auswählen (rechts)

Für das Laden der gewünschten Modelldaten ist zusätzlich ein zum Anwenderprogramm passender Filter zu wählen, der nach Bedarf mit der Auswahl einzelner Stockwerke kombiniert werden kann (Abbildung 7-3). Der ausgewählte Teildatensatz wird anschließend auf dem Server als „neue“ Modellversion zusammengestellt und dem Client als IFC-Datei übergeben.

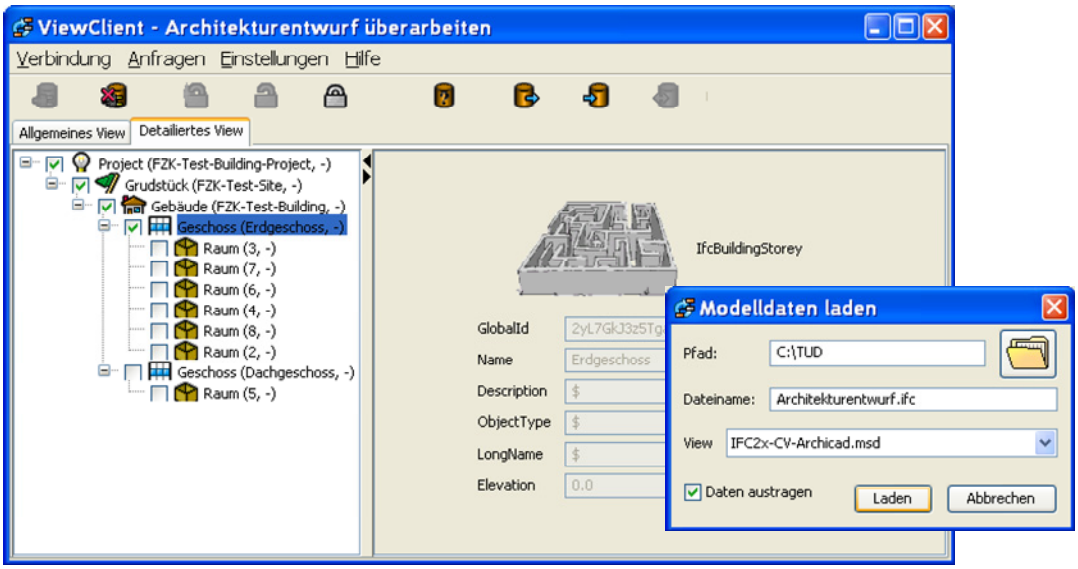


Abbildung 7-3 Auswahl eines Stockwerks zur Einschränkung des gewünschten Teildatensatzes

Nach dem Laden der IFC-Datei kann die Verbindung zum Server geschlossen werden und muss erst für das Übertragen der überarbeiteten IFC-Datei wieder geöffnet werden. Bevor serverseitig ein Gesamtdatenstand erzeugt wird, sind die nachträglich ermittelten Änderungen durch den Anwender freizugeben (Abbildung 7-4). Dieser zusätzliche Schritt ist notwendig, um die Möglichkeit für korrigierende Eingriffe zu schaffen und einen fehlerhaften Gesamtdatenbestand zu vermeiden. Ein ähnliches Vorgehen ist für das Zusammenführen parallel entstandener Änderungen notwendig, das im Rahmen eines automatisierten Vorschlages den Anwender über die übernommenen bzw. zurückgewiesenen Änderungen informieren muss. Es verbleibt das Abschließen der Transaktion, um die entstandenen Modellversionen auch für andere Planer zugänglich zu machen.

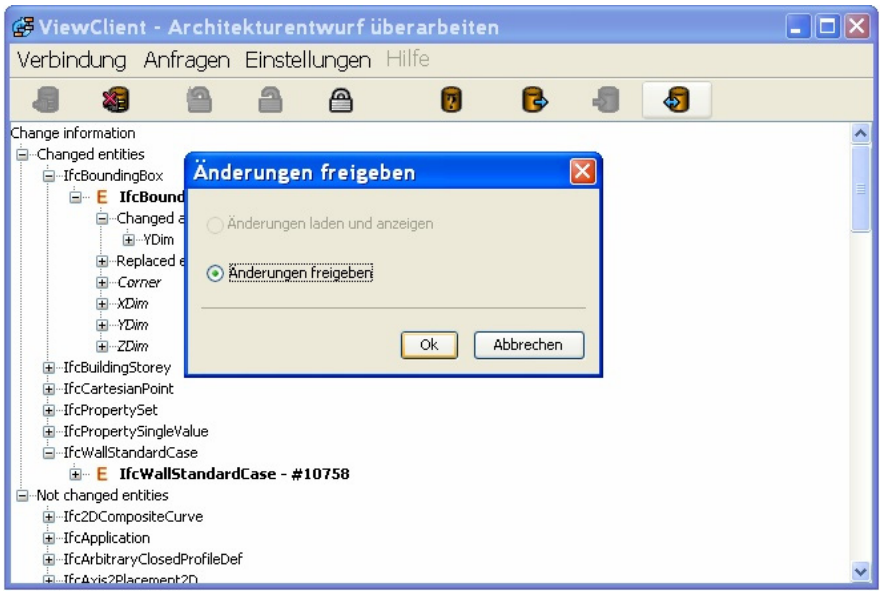


Abbildung 7-4 Anzeige und Freigabe von Änderungen

7.1.3 Konsistenzbedingungen und die Nutzung von Dateien

Das in Abbildung 7-1 dargestellte Szenario beschreibt das Vorgehen, um die Verwendung von Teildatensätzen zu unterstützen. Für die praktische Anwendung ergeben sich jedoch Probleme, die aus dem dateibasierten Datenaustausch und den einzuhaltenden Konsistenzbedingungen resultieren.

Das in Abbildung 7-5 dargestellte, erweiterte Szenario ist notwendig, weil die in IFC definierten Konsistenzbedingungen den dateibasierten Datenaustausch reglementieren und dadurch nicht jeder Teildatensatz nach den STEP-Konventionen austauschbar ist. Als Folge dieser Einschränkung sind dem Teildatensatz ungewünschte Informationen hinzuzufügen. Um nach der Bearbeitung die durchgeführten Änderungen bestimmen zu können, müssen die ungewünschten Informationen aus den SPF konformen Teildatensätzen wieder entfernt werden. Hierfür werden neue Modellversionen erzeugt, die den gewünschten Teildatensätzen entsprechen. Für den anschließenden Vergleich wird darüber hinaus ein Umorganisieren der deltabasierten Speicherung notwendig, um die temporär erzeugten dateikonformen Teildatensätze später aus der Versionshistorie wieder entfernen zu können.

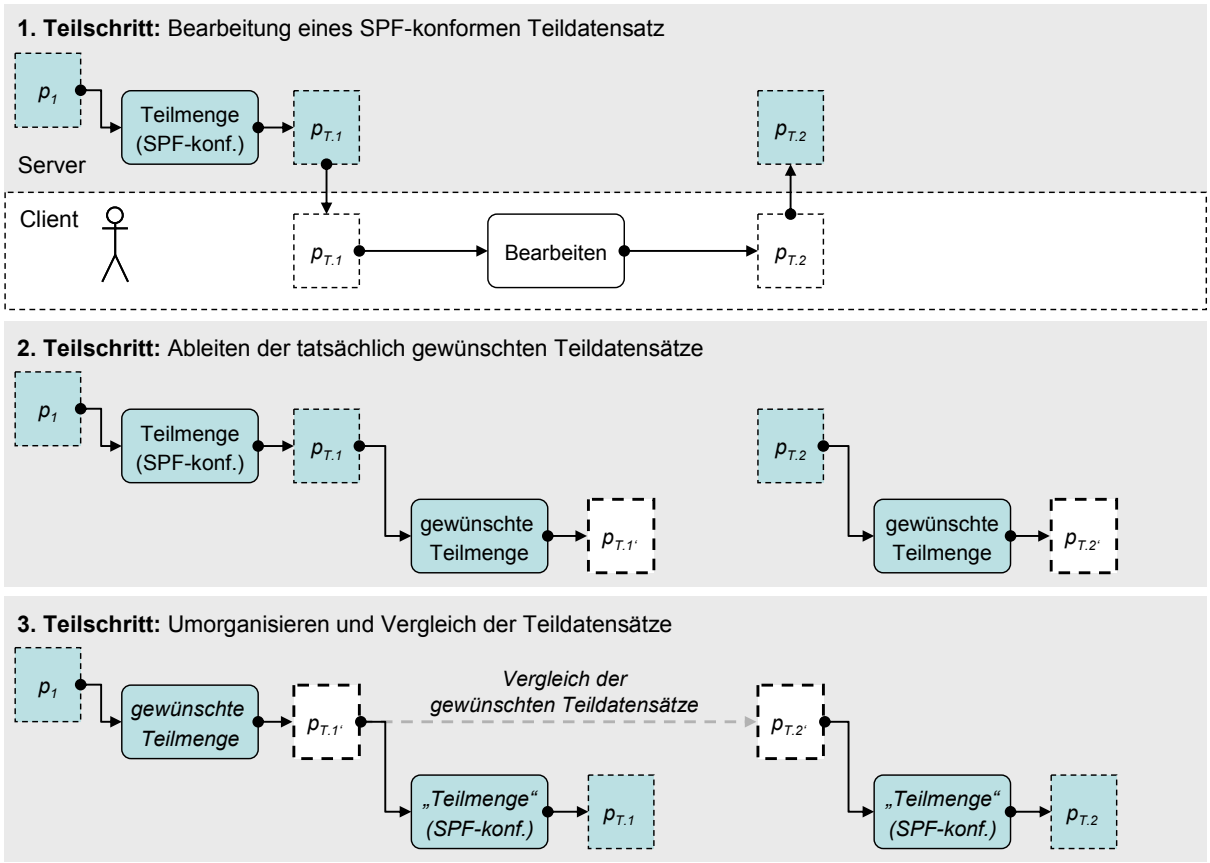


Abbildung 7-5 Prinzipieller Ablauf des erweiterten Szenarios

7.2 Test und Bewertung des IFC-Anwendungsszenarios

Das vorgestellte Szenario gibt den Rahmen für die durchgeführten Untersuchungen vor. Hierbei wurden IFC-Datensätze verschiedener IFC-Versionen[1] und mit einer Größe von bis zu 130000 Instanzen betrachtet. Die Bearbeitung wurde mit den Programmen ArchiCAD, Auto-

[1] Es wurden die IFC-Versionen IFC2x, IFC1.5.1 und IFC2x2 verwendet.

desk ADT und Nemetschek Allplan aber auch gezielt manuell durchgeführt. Mit diesen Beispielen, die auf der Arbeit von Korpowski (2003) aufbauen, wurden die Teilprobleme des Gesamtszenarios untersucht. Eine allgemeingültige Bewertung ist trotz umfangreicher Untersuchungen nur begrenzt ableitbar, da verschiedene Faktoren die Qualität der Ergebnisse beeinflussen. Dieser Einfluss beginnt mit dem verwendeten Datenmodell, setzt sich mit den untersuchten Teildatensätzen und betrachteten Anwenderprogrammen fort und endet mit dem Umfang der beispielhaft durchgeführten Änderungen.

7.2.1 Beschreibung von Teildatensätzen

Mit dem GMSD-Schema kann die Auswahl bestimmter Objekte mit einem vordefinierten Filter kombiniert werden. Ein solcher Filter wird für die Formalisierung der diskutierten *Views* verwendet. Zusätzlich wurde die Möglichkeit untersucht, den Teildatensatz über die Kombination mit einer Objektauswahl weiter einzuschränken.

Vordefinierte Filter zur Beschreibung programmspezifischer Views

Für das IFC-Modell existiert mit dem sogenannten *Coordination View* ein Beispiel, das die Komplexität einer solchen Vereinbarung zeigt. Mit einer View-Definition wird seitens der IAI ein Teildatensatz festgelegt, der von hierfür zertifizierten Anwenderprogrammen unterstützt wird. Untersuchungen haben gezeigt, dass Anwenderprogramme von dieser Vereinbarung häufig abweichen und auf diese Weise einen individuell verarbeitbaren Teildatensatz festlegen (Korpowski 2003). Dadurch entsteht die Notwendigkeit, für jedes Programm einen eigenen Filter zu definieren und zu pflegen. Für die Eignung einer Filterbeschreibung ist neben der Formalisierbarkeit somit auch der notwendige Aufwand für die Definition und Pflege entscheidend. Dieser Aspekt wird einerseits durch das GMSD-Schema über die Möglichkeit der Voreinstellungen und andererseits durch das in Abbildung 7-6 dargestellte Werkzeug zur Filterdefinition berücksichtigt.

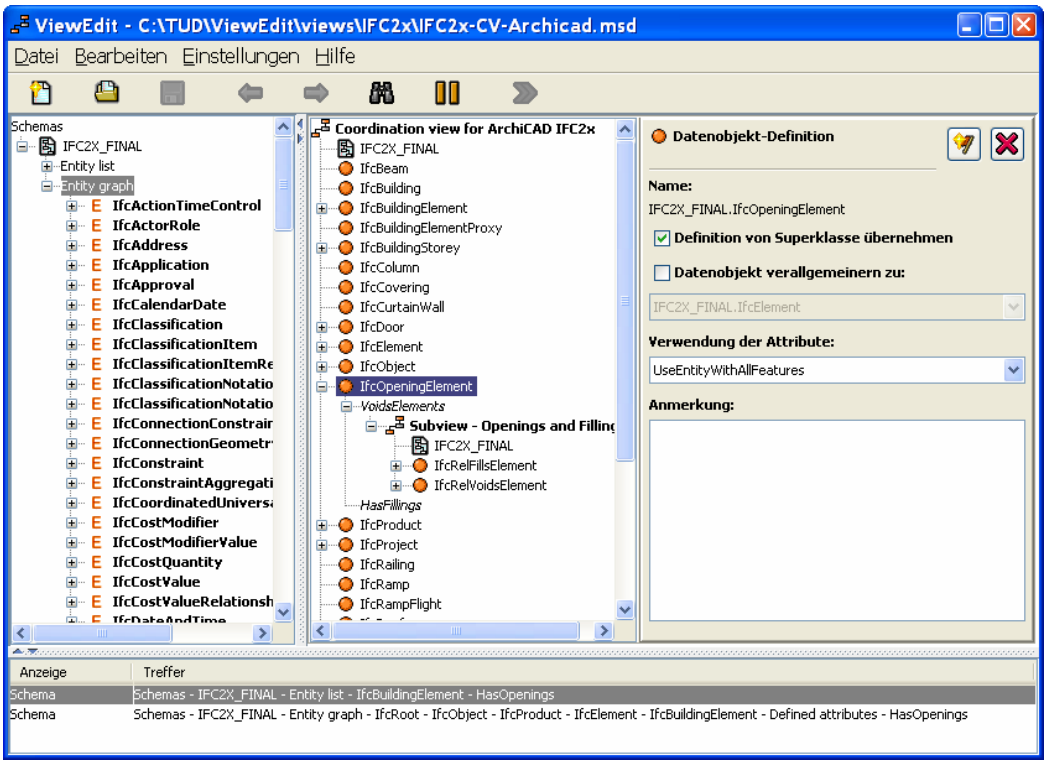


Abbildung 7-6 Unterstützung bei der Definition komplexer Filter durch das Werkzeug *ViewEdit*.

Am Beispiel des *Coordination View* wird deutlich, dass die Auswahl der benötigten Informationen nicht allein über die Objektklasse beschrieben werden kann. Die Geometrie eines Elementes soll beispielsweise nur dann in einem Teildatensatz enthalten sein, wenn das entsprechende Element ebenso zum Teildatensatz gehört. Da die Beschreibung der Geometrie unabhängig vom beschriebenen Elementtyp ist, ist für eine View-Definition auch die Nutzung der Objektklasse entscheidend. Dieser Zusammenhang ist in dem Beispiel der Abbildung 7-7 gezeigt, in dem die Geometriebeschreibung nur für Stützen (*IfcColumn*) und nicht für Räume (*IfcSpace*) in dem Teildatensatz enthalten sein sollen. Diese Funktionalität ist für eine IFC-Filterdefinition wichtig und wird in dem GMSD-Schema über *Subviews* unterstützt.

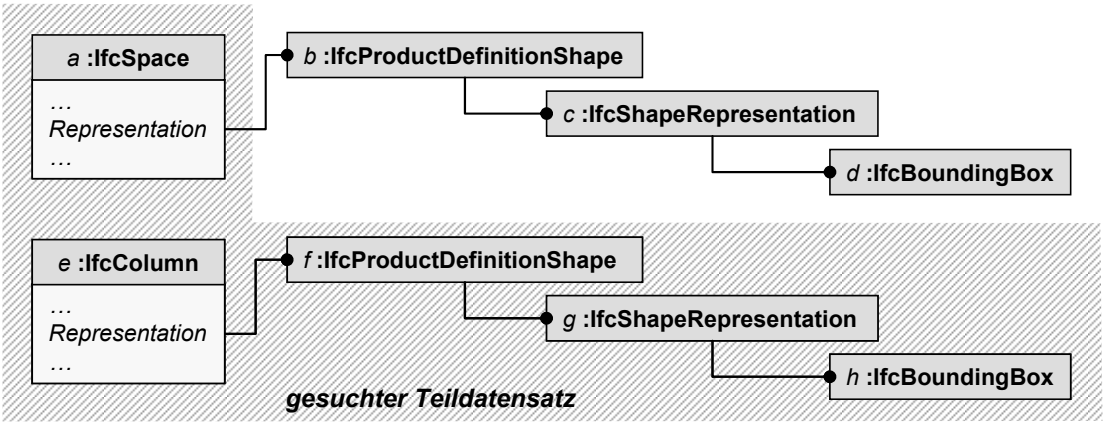


Abbildung 7-7 Unterscheidung der Objekte über die Objektklasse und ihre Verwendung

Bei der Formalisierung des *Coordination View* hat sich gezeigt, dass bereits mit der Filterdefinition die Mehrheit der notwendigen Auswahlkriterien definierbar ist. Gleichzeitig hat sich bestätigt, dass komplexe Teildatensätze relativ einfach definiert und schnell auf individuelle Anforderungen angepasst werden können. Ein vollständiges Vordefinieren des *Coordination View* in Form eines Filters scheitert jedoch aus zwei Gründen. Einerseits wird zwischen dem Lesen und Schreiben der Planungsdaten unterschieden, wodurch die Aufteilung in eine für die Datenabfrage und eine für das Aktualisieren zu verwendende Filterdefinition notwendig wird. Andererseits wird die Attributbelegung in die Auswahl der gesuchten Planungsdaten einbezogen, die über die Kombination mit der Objektselektion beschrieben werden muss. Eine solche Kombination ist für eine vollständige Unterstützung des *Coordination View* aber auch deshalb unvermeidbar, weil die Teildatensätze nicht ausschließlich auf der Grundlage der Datenmodelle beschreibbar sind[1].

Kombination programmspezifischer Filter mit der Auswahl von Objekten

In dem vorangegangenen Abschnitt wurde deutlich, dass zu einer Filterdefinition die attributbasierte Auswahl von Objekten notwendig werden kann. Hieraus entsteht das Problem, die Filterdefinition möglichst effizient mit der Objektauswahl zu kombinieren. Um mit einer solchen Kombination den gewünschten Teildatensatz zu beschreiben, sind Konsistenzbedingungen zwischen der Filterdefinition und der Objektauswahl einzuhalten. Für die Auswahl eines Stockwerks ist es beispielsweise notwendig, ein Stockwerksobjekt zu benennen. Eine nutzer-

[1] Beispielsweise definiert der *Coordination View*, dass nur beim lesenden Zugriff mehrere Stockwerke in dem Teildatensatz enthalten sein dürfen. Für den schreibenden Zugriff ist dementsprechend festzulegen, auf welches Stockwerk der Teildatensatz eingeschränkt werden soll. Eine solche Auswahl ist aber erst zur Laufzeit durch den Anwender möglich.

freundliche Auswahl der Objekte sowie die Einhaltung der Konsistenzbedingungen erfordert schließlich eine auf das Datenmodell abgestimmte Umsetzung, wodurch ein gewisser Grad an Flexibilität und Allgemeingültigkeit aufgegeben werden muss. Die mit dem GMSD-Schema möglichen Abfragestrategien sind in der Abbildung 7-8 dargestellt. Hierbei kann zwischen der alleinigen Anwendung eines Filters (Strategie 1), der Kombination von Objektauswahl und Filter (Strategie 2) und schließlich einer mehrfachen Anwendung von Strategie 1 bzw. 2 unterschieden werden.

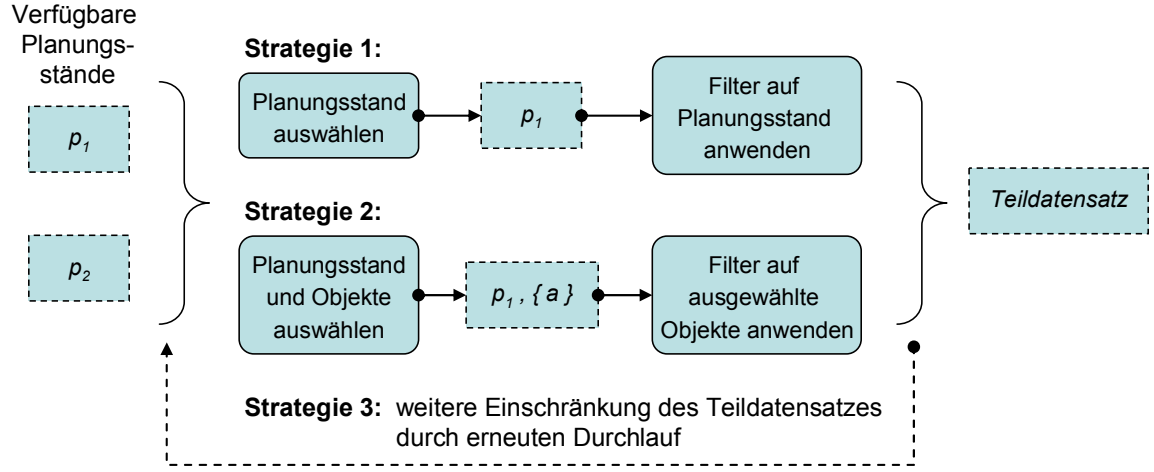


Abbildung 7-8 Mögliche Strategien zur Beschreibung eines Teildatensatzes

Um ausgehend von einem ausgewählten Objekt an alle weiteren Informationen zu gelangen, müssen die hierfür benötigten Objekte über Verknüpfungen erreichbar sein. Kann ein benötigtes Objekt auf diesem Weg nicht erreicht werden, so muss es mit Hilfe der Objektauswahl explizit benannt werden. In dem IFC-Modell spielt dieses Problem eine untergeordnete Rolle, da logisch zusammengehörige Objekte in der Regel über sogenannte „Container"-Klassen[1] gruppiert werden oder über Verknüpfungen zueinander in Beziehung stehen. Eine Ausnahme bildet die Beschreibung der Materialeigenschaften (*IfcMaterialProperties*), die abweichend von der sonst üblichen Verknüpfungsrichtung nicht über das näher definierte Objekt, in diesem Fall das Material-Objekt (*IfcMaterial*), erreichbar ist. Soll der Teildatensatz auf die Materialeigenschaften der benötigten Bauelemente beschränkt werden, so ist folglich eine zusätzliche Objektauswahl notwendig.

7.2.2 Vergleich bearbeiteter IFC-Datensätze

Ein Datensatz kann in identifizierbare und nicht identifizierbare Objekte unterteilt werden. Für IFC-Datensätze ist zu beobachten, dass der Anteil der eindeutig identifizierbaren Objekte im Idealfall bei ca. 20% und für praxisrelevante Problemstellungen zwischen 5% und deutlich unter 1% liegt[2]. Mit dem vorgestellten Vergleichsverfahren wird versucht, zwischen den zu vergleichenden Modellversionen eine möglichst vollständige Zuordnung der zusammengehörigen Objekte zu erreichen. Ausgehend von identifizierbaren Objekten wird durch die Bewertung der Verknüpfungen das Identifizieren weiterer Objekte verfolgt. Auf diese Weise kann

[1] Hierzu gehören beispielsweise *IfcSpatialStructureElement*, *IfcGroup* und alle davon abgeleiteten Klassen.

[2] Im Idealfall sind alle Objekte, die das Attribut *GlobalId* besitzen, über dieses Attribut identifizierbar. In der Praxis ist aber nur Teil dieser Objekte identifizierbar, weil über die View-Definitionen die Identifizierbarkeit der Objekte weiter eingeschränkt wird.

ein Objekt mit einer gewissen Wahrscheinlichkeit, die über die Verknüpfungstiefe und die Verknüpfungsart abschätzbar ist, identifiziert werden. Es kann folglich nicht garantiert werden, dass die erkannten Änderungen mit den tatsächlich durchgeführten Änderungen übereinstimmen. Die ermittelten Änderungen können dadurch an Aussagekraft verlieren und die spätere Re-Integration behindern.

Das Erkennen zugehöriger Objekte kann zu Mehrdeutigkeiten führen, die in dem vorgestellten Vergleichsverfahren durch die Bewertung des Objektzustandes aufgelöst werden. Das Vergleichsverfahren ist neben den veränderbaren Verknüpfungen somit auch vom Attributzustand abhängig. Der Umfang und die Art der Änderungen haben dadurch Einfluss auf das Vergleichsergebnis. Dieser Einfluss wurde durch die Untersuchung unterschiedlich stark veränderter Datensätze abgeschätzt. Als ein Maß wird im Folgenden der Umfang der Änderungen verwendet, der auf drei Ebenen betrachtet werden kann: 1.) die Gesamtheit der vorgenommenen Änderungen, 2.) die Anzahl der Änderungen, die ausgehend von einem eindeutig identifizierbaren Objekt erkannt werden müssen und 3.) die Anzahl der Änderungen, die sich auf die Bewertung einer Verknüpfung und somit die Zuordnung anderer Objekte auswirken. Hieraus wird deutlich, dass der Anteil der eindeutig identifizierbaren Objekte eine wichtige Bezugsgröße darstellt. Bezogen auf diesen Anteil wird nachfolgend zwischen geringfügig und stark veränderten Datensätzen unterschieden. Als grobe Abgrenzung wird ein Datensatz als geringfügig verändert bezeichnet, wenn das Verhältnis zwischen geänderten und eindeutig identifizierbaren Objekten kleiner als 5 bleibt. Ein Datensatz wird demgegenüber als stark verändert bezeichnet, wenn dieses Verhältnis auf über 20 anwächst.

Vergleich geringfügig veränderter Datensätze

Ein geringfügig oder nicht veränderter Datensatz bietet die besten Voraussetzungen, um ein gutes Vergleichsergebnis zu erzielen. Mit diesem Szenario kann für ein gegebenes Datenmodell überprüft werden, welcher Anteil der Objekte korrekt zugeordnet werden kann, welche Verknüpfungsarten hierfür auszuwerten sind und wie empfindlich das Vergleichsergebnis auf gezielte Änderungen reagiert.

Für die untersuchten Datensätze hat sich gezeigt, dass trotz des geringen Anteils eindeutig identifizierbarer Objekte eine zumeist vollständige und fehlerfreie Zuordnung der Objekte möglich ist. Um dieses Ergebnis zu erreichen, werden alle Formen der beschriebenen Zuordnungsregeln benötigt. Hierzu gehören Einfachreferenzen, Mehrfachreferenzen sowie die Auswertung nicht erreichbarer Objekte. Diese Zuordnungsregeln haben einen unterschiedlichen Anteil am Vergleichsergebnis. Entsprechend dem relativ geringen Anteil an Einfachreferenzen hat sich bestätigt, dass über diese Form der Zuordnung, die formal die höchste Zutreffenswahrscheinlichkeit besitzt, nur wenige Objekte erkannt werden können. Den höchsten Anteil am Vergleichsergebnis haben Mengenreferenzen, wobei die ungeordneten Mengenreferenzen gegenüber den geordneten Mengenreferenzen deutlich überwiegen. Für diese Form der Zuordnung ist die Bewertung des Objektzustandes notwendig. Hierbei hat sich gezeigt, dass eine Entscheidung häufig nur dann möglich ist, wenn auch die toleranzbehafteten Attribute (Gleitkommazahlen) in die Bewertung des Objektzustandes einfließen. Diese Art von Attributen ist für das Bilden der Prüfsumme kritisch, weil erlaubte Toleranzen den Wert der Prüfsumme nicht beeinflussen sollen. Die im Rahmen der Arbeit vorgenommene Rundung der Gleitkommazahlen ist nicht unkritisch, hat bei den Untersuchungen aber keine Probleme bereitet.

Ausgehend von identifizierbaren Objekten kann mit Hilfe der Einfach- und Mengenreferenzen die Mehrzahl der Objekte zugeordnet werden. Häufig verbleibt aber ein Rest von Objekten, die auf diese Weise nicht erreicht und somit nicht zugeordnet werden können. Neben neuen bzw. gelöschten Objekten gehören hierzu Objekte, die über ihre Verknüpfungen noch nicht zugeordnet werden konnten. Hierzu gehören mitunter auch verwaiste Objekte, die für den Datensatz keine Bedeutung mehr besitzen. Im Gegensatz zu ungeordneten Mengenreferenzen, bei denen formal die gleiche Problemstellung vorliegt, wird für den verbleibenden Rest von Objekten eine aufwendigere Bewertung des Objektzustandes verfolgt. Dieser Bewertungsansatz, der nicht mehr skalierbar ist, ist jedoch nur für eine begrenzte Anzahl von Objekten nutzbar. Ist eine entsprechend hohe Anzahl von Objekten zuzuordnen, ist eine weitere Unterteilung der Objektmenge, beispielsweise auf Basis der Objektklasse, notwendig.

Vergleich stark veränderter Datensätze

In den untersuchten Datensätzen stehen einem eindeutig identifizierbaren Objekt mitunter 175 nicht identifizierbare Objekte gegenüber. Um alle Objekte zuordnen zu können, sind bis zu 6 Verknüpfungen zu verfolgen (siehe Abbildung 7-9). Diese Verhältnisse lassen eine hohe Empfindlichkeit gegenüber Änderungen erwarten.

Die Auswirkung einer Änderung ist von der Einbettung in dem Objektnetz sowie ihrer Auswertung durch den Vergleichsalgorithmus abhängig. Die Zuordnung eines Objektes wird gestört, wenn aufgrund der Änderung eine Entscheidung zwischen verschiedenen Zuordnungsmöglichkeiten nicht mehr eindeutig ist. Durch diese Störung ist auch die Zuordnung der damit verknüpften Objekte gestört, die zu einer Vervielfachung der Störung führen kann. Je früher die Verknüpfungsfolge unterbrochen wird, umso höher werden die zu erwartenden Störungen sein. Dieser Effekt ist auf das gemittelte Verhältnis von 1:175 begrenzt, solange die Änderung durch ein identifizierbares Objekt gekapselt[1] wird.

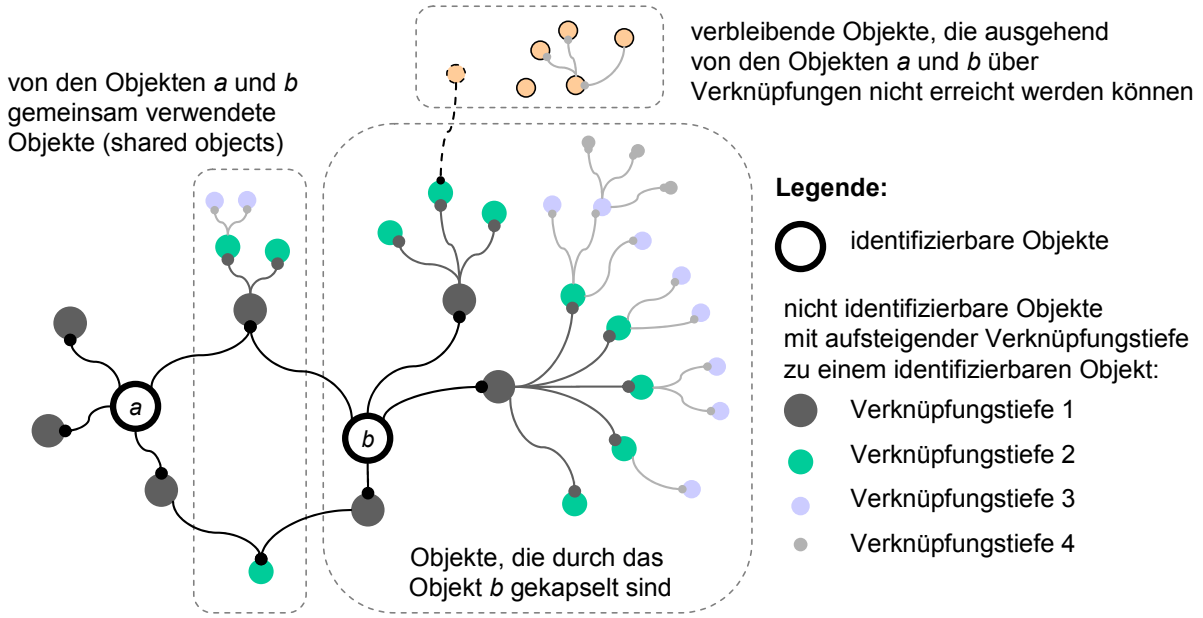


Abbildung 7-9 Darstellung eines über die gegenseitigen Verknüpfungen entstehenden Objektnetzes

[1] Der in dem objekt-orientierten Ansatz geprägte Begriff der Kapselung wird hier mit der exklusiven Verwendung eines Objektes assoziiert. Nicht gekapselte Objekte werden dementsprechend von mehreren (identifizierbaren) Objekten verwendet und in den IFC als *shared objects* bezeichnet.

Aus dieser Betrachtung wird deutlich, dass Änderungen nur unter bestimmten Voraussetzungen zu einer Störung der Objektzuordnung führen. Für geringfügig geänderte Daten werden diese Voraussetzungen nur selten erreicht, weil der hohe Anteil unveränderten Daten den Informationsverlust der vorgenommenen Änderungen gut kompensieren kann. Trat dennoch eine Störung auf, so waren die Auswirkungen auf wenige Objekte begrenzt und haben nicht zu einer unkontrollierten Ausbreitung geführt. Demgegenüber sind für stark veränderte Datensätze die Voraussetzungen für eine Störung der Objektzuordnung deutlich häufiger gegeben. Doch auch in diesem Fall hat sich gezeigt, dass die Auswirkungen einer Störung meist auf wenige Objekte begrenzt sind. Dieser Effekt resultiert aus den eindeutig identifizierbaren Objekten, die eine gewisse Korrekturwirkung besitzen. Der Vergleichsalgorithmus arbeitet unter diesen Voraussetzungen auch bei relativ stark veränderten Datensätzen vergleichsweise robust und erzeugt ein brauchbares Vergleichsergebnis. Infolge einer Störung ist insgesamt zu beobachten, dass deutlich häufiger keine als eine falsche Zuordnung der Objekte stattfindet. Für einen sehr stark veränderten Datensatz und eine geringe Anzahl identifizierbarer Objekte stößt der Vergleichsalgorithmus dennoch an Grenzen, weil unter diesen Voraussetzungen das Vergleichsergebnis durch einen hohen Anteil „neuer“ Objekte einen begrenzten Nutzwert hat.

Veränderungen am Objektgefüge

Eine Veränderung am Objektgefüge tritt ein, wenn die Versionsverwaltung der Objekte auf irgendeine Weise gestört wird. Hierzu gehören das Löschen, Einfügen, Teilen und Zusammenlegen von Objekten sowie die Änderung des Objekttyps. An dieser Stelle entsteht ein Konflikt für identifizierbare Objekte, da durch die Forderung nach Eindeutigkeit das Teilen und Zusammenlegen als Änderung formal ausgeschlossen wird. Diese Einschränkung kann mit dem vorgeschlagenen Vergleichsverfahren nur für Objekte überwunden werden, deren Zuordnung über die Bewertung der Verknüpfungen erfolgt. Um mit diesem Ansatz ein geteiltes bzw. zusammengelegtes Objekt erkennen zu können, müssen mit Hilfe der Verknüpfungen mehrere Möglichkeiten für die Zuordnung eines Objektes ableitbar sein. Das Teilen und Zusammenlegen wird aber nur dann als Änderung akzeptiert, wenn aufgrund gleicher Zutreffenswahrscheinlichkeiten keine Entscheidung zugunsten einer Objektzuordnung möglich ist.

In dem IFC-Modell sind diese Mehrdeutigkeiten bei gemeinsam verwendeten Objekten (*shared objects*) wie *IfcOwnerHistory* und *IfcMaterial* zu beobachten. Für diese Objekte stellt das Teilen und Zusammenlegen nur bedingt eine sinnvolle Änderungsinformation[1] dar. In diesen Fällen ist es häufig vorteilhaft, durch das Einbeziehen zusätzlicher Entscheidungskriterien auf das Teilen bzw. Zusammenlegen zu verzichten. Dies verhindert die beim Teilen notwendige Aktualisierung der Referenzen, die zu einem unverhältnismäßigen Anwachsen der Änderungsmenge führen kann. Auf das Teilen und Zusammenlegen kann häufig auch deshalb verzichtet werden, weil für die betroffenen Objekte beim Wiedereingliedern des bearbeiteten Teildatensatzes keine bzw. nur fragliche Informationen wiederhergestellt werden müssen.

[1] Obwohl das Teilen von *IfcMaterial* und *IfcOwnerHistory* nicht sinnvoll erscheint, so ist diese Änderungsinformation nicht unberechtigt. Wird beispielsweise einem Objekt ein anderes Material zugewiesen, so ist der damit verbundene Übergang zu einem anderen Material auch ohne das geänderte Objekt erkennbar. Ähnlich kann für die Beschreibung der Objektentstehung (*IfcOwnerHistory*) argumentiert werden, die unabhängig von den dadurch beschriebenen Objekten eine verallgemeinerte Bewertung der vorgenommenen Änderungen erlaubt. Dieser „Informationsgewinn“ ist jedoch an die Änderung des Objektgefüges gebunden, der nur selten in einem brauchbaren Verhältnis zu den damit verbundenen Anpassungen steht.

Der generische Vergleichsalgorithmus kann das Teilen und Zusammenlegen von Objekten nur schwer erkennen und stößt bei dieser Problemstellung an Grenzen. Dies liegt vor allem auch daran, dass Objekte, die für diese Form der Änderung besonders interessant sind, meist über ein Schlüsselattribut identifizierbar sind und auf diese Weise nicht als geteilt oder zusammengelegt erkannt werden können. Der Nutzen ist dadurch auf die Dokumentation von Mehrdeutigkeiten begrenzt, die bei einer späteren Bewertung und dem Auflösen der Mehrdeutigkeit helfen kann. In der Tabelle 1 sind die Ergebnisse des Vergleichsverfahrens für das IFC-Modell als verallgemeinernde Abschätzung zusammengefasst. Hieraus wird deutlich, dass in diesem Szenario nur ein Teil der beschreibbaren Änderungsinformationen effektiv genutzt werden kann.

Tabelle 1 Abschätzung der Erkennungs- und Fehlerraten für die verschiedenen Änderungstypen

Änderung	**Erkennungs- und Fehlerrate für**				**Bemerkungen**
	geringfügig veränderte Datensätze		stark veränderte Datensätze		
	E-Rate	F-Rate	E-Rate	F-Rate	
Erzeugen	+	+	+	-	bei stark veränderten Daten werden geänderte Objekte tendenziell als erzeugte bzw. gelöschte Objekte erkannt
Löschen	+	+	+	-	
Ändern (inkl. Änderung der Objektklasse)	++	++	o	+	
Teilen	--	o	--	o	keine Unterstützung durch das IFC-Modell
Zusammenlegen	--	o	--	o	

(++ = sehr hohe Erkennungs- bzw. sehr geringe Fehlerrate
o = ausgeglichenes Verhältnis
-- = sehr geringe Erkennungs- bzw. sehr hohe Fehlerrate)

7.2.3 Re-Integration geänderter Teildatensätze

Der Schritt der Re-Integration hat die Aufgabe, die vorgenommenen Änderungen auf den Gesamtdatensatz zu übertragen. Das hierfür vorgeschlagene formale Verfahren garantiert, dass die ermittelten Änderungen vollständig in dem Gesamtdatensatz integriert und die Konsistenzbedingungen des Versionsmodells eingehalten werden. Durch dieses formale Verfahren wird jedoch nicht gewährleistet, dass die Informationen des ursprünglichen Gesamtdatensatzes (p_1, siehe Abbildung 7-1) in den neuen Gesamtdatensatz (p_2) vollständig übernommen werden können. Zusätzlich besteht die Gefahr, dass die vorgenommenen Änderungen den Gesamtdatensatz implizit verändern und dadurch einen ungewünschten Planungsstand erzeugen. Diese Gefahr resultiert aus den nicht fehlerfrei erkennbaren Änderungen sowie der Tatsache, dass zwischen inhaltlichen und strukturellen Änderungen eine dem generischen Verfahren unbekannte Abhängigkeitsbeziehung besteht.

Verträglichkeit der Änderungen mit dem ursprünglichen Gesamtdatensatz

In dem beschriebenen Ablauf wird die Gefahr unverträglicher Änderungen durch die Bewertung und Freigabe der ermittelten Änderungen reduziert (siehe Abbildung 7-4). Werden kriti-

sche Änderungen erkannt, so müssen diese Änderungen korrigiert werden. Sind nur wenige Fehler bzw. kritische Änderungen aufgetreten, so ist eine manuelle Korrektur denkbar. Andernfalls sind automatisierte Verfahren notwendig, die beispielsweise durch ein „Umformen" des Teildatensatzes und ein wiederholtes Vergleichen bessere Vergleichsergebnisse erzielen können. Das „Umformen" und erneute Vergleichen bietet sich an, wenn inhaltliche und strukturelle Änderungen stark differieren. Bei der Anwendung unterschiedlicher CAD-Programme stellen diese Probleme das größte Hindernis für den Schritt der Re-Integration dar. Als kritisch haben sich vor allem das Verändern von Bezugskoordinatensystemen und die Verwendung anderer Maßeinheiten (*IfcUnit*) erwiesen. In diesen Fällen werden bei der Aktualisierung des Gesamtdatensatzes implizit auch andere Daten verändert. Diese Problemstellung ist nur mit Datentransformationen zu lösen, die die Differenzen zwischen inhaltlichen und strukturellen Änderungen beseitigen können.

Die Gefahr einer unbeabsichtigten impliziten Veränderung kann durch die Korrektur des Vergleichsergebnisses nicht vollständig ausgeschlossen werden. Dieser Fall ist für Objekte des Typs *IfcOwnerHistory* zu beobachten, die beim Schreiben eines IFC-Datensatzes meist vollständig von den CAD-Programmen ersetzt werden. Der Vergleichsalgorithmus erkennt diese Objekte nicht als *neu* sondern als geändert und erzeugt damit ein für den betrachten Teildatensatz plausibles Ergebnis. Aus diesem Vergleichsergebnis entsteht beim Aktualisieren des Gesamtdatensatzes ein Konflikt, wenn mit diesem Objekt die Entstehungsgeschichte auch für andere, nicht im Teildatensatz enthaltene Objekte beschrieben wird. In diesem Fall wird die Entstehungsgeschichte unbeabsichtigt auf unveränderte Objekte übertragen. Das geschilderte Problem ist ein negatives Beispiel[1] für ein häufig sinnvolles und für das Aktualisieren des Gesamtdatensatzes unverzichtbares Vorgehen. Das Erkennen einer unbeabsichtigten impliziten Veränderung ist durch das vorgeschlagene generische Verfahren jedoch nicht eindeutig möglich und kann über die Art der gemeinsamen Nutzung bestenfalls abgeschätzt werden.

Konsistenz der Datenstruktur

Neben der Verträglichkeit von Änderungen, die sich über die Bedeutung der Daten erschließt, ist bei der Aktualisierung des Gesamtdatensatzes die Konsistenz der Datenstruktur zu gewährleisten. Auf dieser Ebene stehen formalisierte Verträglichkeitsbedingungen zur Verfügung, die von generischen Verfahren ausgewertet werden können. In dem IFC-Modell werden solche Verträglichkeitsbedingungen genutzt, um die Vollständigkeit von Informationen, zulässige Wertbereiche und auszuschließende Wertkombinationen zu formalisieren. Für das Aktualisieren des Gesamtdatensatzes sind jedoch nur solche Verträglichkeitsbedingungen zu prüfen, die die Beziehungen zwischen veränderten und unveränderten Daten einschränken. Im Rahmen des vorgestellten Ansatzes werden die Verknüpfungen der Objekte sowie die Vollständigkeit der Informationen bewertet.

Eine beim Bilden des Teildatensatzes entfernte Verknüpfung kann nur dann wieder hergestellt werden, wenn bei der Bearbeitung des Teildatensatzes das verknüpfte Objekt nicht gelöscht wurde. Der damit verbundene Informationsverlust ist ein Hinweis auf fachliche Widersprüche, die sich teilweise über unvollständige Objektbelegungen ausdrücken und somit die Kon-

[1] Der beschriebene Fall ist auf Unzulänglichkeiten der IFC-Implementierungen zurückzuführen und deshalb ein schlechtes Beispiel für die zugrunde liegende Problematik. Wird der verwendete Teildatensatzes nämlich wie vorgesehen auf die nutzbaren Informationen beschränkt, so sind Objekte des Typs *IfcOwnerHistory* nicht im Teildatensatz enthalten und können demzufolge auch nicht verändert werden. Mit dieser Herangehensweise kann für den beschriebenen Fall das Problem ungewollter Veränderungen vermieden werden.

sistenzbedingungen der Datenstruktur verletzten. Werden die Konsistenzbedingungen der Datenstruktur verletzt, so ist der entstandene Konflikt nachträglich aufzulösen. Hierfür kann nur bedingt auf automatisierte Verfahren zurückgegriffen werden, da aus einer unvollständigen Objektbelegung nur eingeschränkt die Existenzberechtigung des betreffenden Objektes ableitbar ist. Ein Entfernen unvollständiger Objekte kann die Konsistenz der Datenstruktur zwar formal erfüllen, ist jedoch nicht immer eine fachlich sinnvolle Reaktion auf den entstandenen Konflikt. Eine fachliche Bewertung ist demzufolge unverzichtbar und erfordert häufig das Einbeziehen anderer Planer. Für die Bewertung des aktualisierten Gesamtdatensatzes ist schließlich entscheidend, ob die hierfür verwendeten Werkzeuge mit strukturell inkonsistenten Planungsdaten umgehen können oder ob formal die Konsistenzbedingungen der Datenstruktur einzuhalten sind.

Konsistenz des Versionsmodells

Strukturell inkonsistente Planungsdaten sind häufig auf ungültige und daraufhin entfernte Verknüpfungen zurückzuführen. Das Entfernen der Verknüpfungen ist notwendig, um die Konsistenzbedingungen des Versionsmodells auch beim aktualisierten Gesamtdatensatz zu wahren. Das Einhalten der Konsistenzbedingungen wird beim Schritt der Re-Integration durch das Löschen ungültiger bzw. mehrdeutiger Verknüpfungen erzwungen und garantiert dadurch eine widerspruchsfreie Versionsverwaltung. Eine Verknüpfung wird dementsprechend gelöscht, wenn das referenzierte Objekt entweder entfernt oder geteilt wurde. Bei den Untersuchungen hat sich bestätigt, dass gelöschte Objekte die Hauptursache für ungültige Verknüpfungen darstellen.

Die Tabelle 2 enthält einen Überblick über die beschriebenen Probleme, die gewählten Lösungsansätze und schließlich die Relevanz für die untersuchten IFC-Beispiele.

Tabelle 2 Übersicht über die Probleme der Re-Integration, ihre Ursachen und deren Lösung

Problem	Lösungsansatz	Relevanz für IFC
Unverträgliche Änderungen	manuell (Freigabe und Korrektur der Änderungen durch den Anwender)	problematisch für gemeinsam verwendete Objekte (kritisch vor allem für Maßeinheiten - *IfcUnit* und Bezugskoordinatensysteme)
Inkonsistenzen bzgl. der Datenstruktur	halbautomatisch (automatisches Erkennen und manuelle Korrektur der Inkonsistenzen)	unvollständige Objekte infolge gelöschter Verknüpfungen und der Nutzung von Teildatensätzen
Inkonsistenzen bzgl. des Versionsmodells	automatisch (Auflösen durch vordefinierte Regeln)	ungültige bzw. mehrdeutige Verknüpfungen vor allem infolge gelöschter Objekte

7.2.4 Zusammenführen von Änderungen

Das Zusammenführen parallel durchgeführter Änderungen ist mit den Problemen der Re-Integration vergleichbar. Im Unterschied zur Re-Integration, bei der die vorgenommenen Änderungen gegenüber einem alten Planungsstand bewertet und vollständig übernommen werden, werden beim Zusammenführen zwei Änderungsmengen betrachtet. Ausgehend von einer der beiden Änderungsmengen werden von der anderen Änderungsmenge nur solche Ände-

rungen übernommen, die keine Konflikte mit dem Versionsmodell verursachen. Setzt man voraus, dass durch den Schritt der Re-Integration die Konsistenz der Datenstruktur sowie die Verträglichkeit der Änderungen gewährleistet werden, so sind die Probleme des Zusammenführens im wesentlichen auf fachliche, durch den Anwender zu bewertende Widersprüche begrenzt.

In Analogie zur Re-Integration sind in der Tabelle 3 die Probleme für das Zusammenführen von Änderungen dargestellt. Bei diesem Verfahren wird ein neuer Planungsstand erzeugt, der die Konsistenzbedingungen des Versionsmodells erfüllt. Hierbei werden auch Abhängigkeiten berücksichtigt, die sich aus dem Zurückweisen einer Änderung ergeben. Dieses Vorgehen ist besonders dann sinnvoll, wenn auch die Konsistenzbedingungen der Datenstruktur einbezogen werden. Die zurückgewiesenen Änderungen werden dadurch nicht individuell, sondern als eine zusammengehörige Einheit betrachtet, die entweder vollständig oder gar nicht übernommen wird. In diesem Fall wird eine konsistente Datenstruktur ohne zusätzliche Nutzerinteraktionen gewährleistet. Darüber hinaus erfüllen die übernommenen Änderungen in der Regel auch das Kriterium der Verträglichkeit. Diese Aussage ist jedoch an die Bedingung gekoppelt, dass beim Schritt der Re-Integration die unverträglichen Änderungen im Sinne des ursprünglichen Planungsstandes aufgelöst wurden. Unter diesen Voraussetzungen verbleibt für den zusammengeführten Planungsstand lediglich die Gefahr, fachliche Widersprüche zu enthalten (siehe Tabelle 3). Damit zeigt sich, dass das Zusammenführen von Änderungen auf die fachliche Bewertung der übernommenen Änderungen beschränkt werden kann. Aus dem zusammengeführten Planungsstand ist schließlich erkennbar, welche Änderungen übernommen und welche Änderungen zurückgewiesen wurden.

Tabelle 3 Abschätzung des Gefährdungspotenzials durch das Zusammenführen von Änderungen

Probleme des zusammengeführten Planungsstandes	**Einschätzung der Gefahr**	**Bemerkung**
Fachliche Widersprüche	hoch	Können nicht automatisch erkannt werden und stellen die größte Gefahr für das Zusammenführen dar.
Unverträgliche Änderungen	gering	Die Gefahr impliziter Änderungen ist gering, wenn unverträgliche Änderungen beim Schritt der Re-Integration im Sinne des ursprünglichen Planungsstandes beseitigt wurden.
Inkonsistenzen bzgl. der Datenstruktur	mittel bzw. nicht vorhanden*	*Kann vermieden werden, wenn die Konsistenzbedingungen der Datenstruktur und die Abhängigkeiten von Änderungen berücksichtigt werden.
Inkonsistenzen bzgl. des Versionsmodells	nicht vorhanden	Wird durch das Verfahren ausgeschlossen.

7.3 Anwendung auf andere Datenmodelle

Der erarbeitete generische Ansatz ist nicht auf das IFC-Modell beschränkt, sondern kann auch auf andere Datenmodelle angewendet werden. Für das Bauwesen ist eine Anwendung unter anderem für CIMSTEEL (CIS/2), STEP-CDS (Construction Drawing Subset), die Produktschnittstelle Stahlbau (PSS), den Objektkatalog Straße (OKSTRA) und andere, vor allem in EXPRESS definierte Datenmodelle interessant. Jedes dieser Datenmodelle deckt in Art und Umfang einen unterschiedlichen Bereich des Bauwesens ab und verfolgt mitunter eine eigene Modellierungsphilosophie. Die hieraus resultierenden Unterschiede haben Einfluss auf die Anwendbarkeit der entwickelten Methoden. Die nachfolgende Bewertung beruht in erster Linie auf einer Untersuchung der Datenstrukturen, die nur teilweise mit konkreten Datensätzen verifiziert werden konnte.

7.3.1 Art und Umfang der abgebildeten Informationen

Die zuvor genannten Datenmodelle sind in der Tabelle 4 mit ihrem Definitionsumfang und den abdeckten Fachbereichen zusammengestellt. Aus dieser Übersicht wird deutlich, dass mit jedem Datenmodell ein bestimmter Schwerpunkt bezüglich der Kooperationsunterstützung verbunden ist. Das OKSTRA-Modell ist ähnlich wie das IFC-Modell auf eine fachübergreifende Kooperation ausgelegt, ist vergleichbar komplex und ebenfalls in verschiedene Schemata aufgeteilt. Mit den Modellen CIS/2 und PSS wird dagegen ein fach- bzw. problemspezifischer Schwerpunkt gesetzt, der eine detailliertere Abbildung von Fachinformationen erlaubt. Bezüglich der Komplexität ist das CIS/2-Modell mit dem IFC-Modell vergleichbar. Das PSS-Modell hat demgegenüber eine verhältnismäßig einfache Struktur, obwohl inhaltlich eine ähnliche Ausrichtung wie bei CIS/2 verfolgt wird. Die einfache Struktur ist aber nicht nur auf einen geringeren Informationsgehalt, sondern auch auf die Modellierung (Verzicht auf das Vererbungskonzept) und die Begrenzung der Darstellungsvielfalt (normierte Abbildung von Informationen) zurückzuführen. Im Vergleich zu diesen Modellen lässt sich der Schwerpunkt von STEP-CDS dagegen nicht auf fachlicher Basis einordnen, weil die Zusammenarbeit auf der Grundlage der fachlich neutralen technischen Zeichnung organisiert wird.

Tabelle 4 Umfang und abgedeckte Fachbereiche verschiedener Datenmodelle

Datenmodell	Anzahl der Klassen und Datentypen	Fachbereich(e)
IFC2X2	623 Klassen, 312 Datentypen	fachübergreifendes Datenmodell für den Bereich Hochbau
OKSTRA	747 Klassen, 61 Datentypen	fachübergreifendes Datenmodell für den Bereich Straßen- und Verkehrstechnik
CIMSTEEL (CIS/2)	731 Klassen, 173 Datentypen	Stahlbau (inkl. Statik und Fertigung)
PSS	46 Klassen, 41 Datentypen	Stahlbau (inkl. Statik und Fertigung)
STEP-CDS	272 Klassen, 97 Datentypen	Zeichnungen (inkl. Sachdaten)

Charakterisierung der Zusammenarbeit

Der Schwerpunkt eines Datenmodells hat Einfluss auf die bei der Zusammenarbeit benötigten Teildatensätze. Es ist zu beobachten, dass in Modelle wie CIS/2, PSS aber auch STEP-CDS die klassenbezogene Unterscheidung der Teildatensätze eine geringe Bedeutung als in IFC oder OKSTRA besitzt. In diesen Modellen beruht die Arbeitsteilung häufig auf „ortsbezogenen" statt fachbezogenen Teildatensätzen, die nicht auf Klassenebene sondern auf Objektebene unterschieden werden. Für die Auswahl der Teildatensätze ist demzufolge die Benennung der benötigten Objekte notwendig, um beispielsweise einen bestimmten Gebäudebereich, ein Konstruktionsdetail oder eine Zeichnung auszuwählen. Aber auch hier ist die Kombination mit einem vordefinierbaren Filter sinnvoll, weil nicht immer alle mit der Objektauswahl verbundenen Informationen gewünscht werden.

Definition der Teildatensätze

Das erarbeitete Verfahren zum Umgang mit Teildatensätzen ist für die genannten Modelle ebenso anwendbar. Wie beim IFC-Modell können auch hier Objektgruppen und gegenseitige Verknüpfungen genutzt werden, um die gewünschten Teildatensätze effizient und flexibel zu beschreiben. Diesem modellunabhängigen Ansatz steht als Alternative eine modellspezifische Lösung des OKSTRA-Modells gegenüber. Im Unterschied zu allen anderen Modellen ist im OKSTRA-Modell die Verwendung von Teildatensätzen explizit vorgesehen und findet sich in der Modelldefinition über symbolische Verweise und die Aufteilung in verschiedene Schemata wieder. Bei diesem Ansatz sind die möglichen Schnittstellen auf ausgewählte, eindeutig identifizierbare Objekte festgelegt. Beim „Herausschneiden" eines Teildatensatzes werden an den Schnittstellen die Verknüpfungen durch symbolische Verweise ersetzt und auf diese Weise ein in sich vollständiger Teildatensatz mit Bezug zum Gesamtmodell erzeugt. Diese Herangehensweise wird durch das vorgestellte, modellunabhängige Verfahren nicht unterstützt, weil der Teildatensatz nicht durch das Entfernen von Informationen sondern durch das Verändern von Attributwerten erzeugt wird. Um dieses modellspezifisch vereinbarte Vorgehen seitens der Datenverwaltung zu unterstützen, sind zwangsläufig entsprechende Erweiterungen unverzichtbar.

7.3.2 Diskussion der Datenstrukturen

Für die Anwendung der generischen Verfahren stellt das Ermitteln der durchgeführten Änderungen eine der schwierigsten und fehleranfälligsten Aufgaben dar. In der Tabelle 5 sind Identifizierungskonzepte der genannten Datenmodelle gegenüberstellt, die zusätzlich den gewünschten Idealfall am Beispiel objekt-orientierter Programmiersprachen charakterisiert. Aus dieser Übersicht wird deutlich, dass in keinem der Datenmodelle die Identifizierbarkeit der Objekte durchgehend gewährleistet wird. Wie beim IFC-Modell besteht zusätzlich das Problem, dass der Inhalt und die Struktur der Daten eine unbekannte Abhängigkeitsbeziehung besitzen. Eine Veränderung der Daten ist also nicht zwingend mit einer inhaltlichen Änderung gleichzusetzen, auch wenn das PSS-Modell durch seine vergleichsweise streng parametrisierte Beschreibung weniger als andere Modelle von diesem Problem betroffen ist. Unabhängig von diesem Problem sind auf Basis der Datenstruktur zunächst die veränderten Daten über das generische Vergleichsverfahren zu bestimmen. Hierfür sind vor allem die von den Datenmodellen verfolgten Identifizierungskonzepte sowie der Grad der Objektvernetzung entscheidend.

Identifizierungskonzepte

In den untersuchten Datenmodellen werden verschiedene Identifizierungskonzepte verfolgt. Unterschiede bestehen hinsichtlich der Beschreibung der Objektkennung und ihrer Gültigkeit innerhalb eines Datensatzes. Für die Beschreibung der Objektkennung wird überwiegend ein gesondertes (unveränderbares) Attribut definiert, dass unabhängig vom Objektzustand die Unterscheidung der Objekte gestattet.

Tabelle 5 Gegenüberstellung der Datenmodelle und ihrer Identifizierungskonzepte

Datenmodell	**Separates ID-Attribut**	**Eindeutigkeit der ID**	**Identifizierbarkeit aller Objekte**
Objekt-orientierte Programmiersprache	ja Objektreferenz/ Speicheradresse	ja	ja
IFC	ja *GlobalId*[1] (über Vererbung)	ja (128 Bit)	nein für alle von *IfcRoot* abgeleitete Objekte verfügbar
STEP Part 41 2nd Edition (Ressource-Schema für andere *Application Protocols*)	ja über separate Klasse *id_attribute. attribut_value*[2]	(ja) – aber nicht als unique definiert	nein für *Application Protocols* unterschiedlich
STEP-CDS	ja klassenspezifisch definiert	(ja) – aber nicht als unique definiert	nein
CIMSTEEL (CIS/2)	ja (zwei Konzepte) 1) siehe Part 41 2) separate Klasse *managed_data_item. instance_id*	(ja) – aber nicht als unique definiert	bedingt über *instance_id* (jedoch nicht obligatorisch)
PSS	ja *id* (klassenspezifisch definiert)	ja	nein
OKSTRA	nein klassenspezifisch definiert	nein (über Typzugehörigkeit und Zeitstempel)	nein

[1] *GlobalId* ab Version 2.0, vorher *ProjectId*.

[2] Mit dieser Lösung können prinzipiell alle Objekte nachträglich einen Identifikator erhalten (hohe Flexibilität). Damit wird aber nicht zwingend für alle möglichen Objekte eine ID vergeben. Es ist anzumerken, dass das Id-Attribute nur über den Klassennamen eindeutig referenziert werden kann, da der Name attribute_value in anderen Klassen ebenfalls verwendet wird.

Für das Identifizieren von Objekten wird neben den eigens definierten ID-Attributen (Surrogat, IFC-Modell) auch die Kombination von Attributen (kombinatorischer Schlüssel, OKSTRA) oder das Einführen eigener, ausschließlich für die Identifizierung verwendeter Objektklassen (STEP Part 41) verfolgt. Das Identifizieren der Objekte ist mitunter auch an die Objektklasse gebunden (klassifikatorische Schlüssel, PSS), auf dessen Basis eine Unterscheidung zwischen mehrfach vorkommenden Schlüsselwerten vorgenommen wird. Aus der in Tabelle 5 zusammengefassten Gegenüberstellung wird deutlich, dass die Datenmodelle der Praxis in dieser Beziehung das objekt-orientierte Paradigma[1] nicht vollständig unterstützen und sehr unterschiedliche Identifizierungskonzepte verfolgen. Hierfür sind modellspezifische Anpassungen des Vergleichsverfahrens notwendig, um das generische Vergleichsverfahren mit gesicherten Objektpaaren sinnvoll initialisieren zu können. Auf dieser Grundlage ist das erarbeitete generische Vergleichsverfahren ebenso anwendbar.

Grad und Informationsgehalt der Objektvernetzung

Mit dem vorgestellten Vergleichsverfahren wird versucht, das Fehlen von Objektschlüsseln durch das Auswerten der Verknüpfungen auszugleichen. Für die Qualität der Vergleichsergebnisse ist es dementsprechend vorteilhaft, wenn eine Vielzahl auswertbarer Verknüpfungen zur Verfügung steht. In den untersuchten Datenmodellen sind die Voraussetzungen für die Zuordnung nicht identifizierbarer Objekte zumeist erfüllt, da diese ausgehend von identifizierbaren Objekten in der Regel über Verknüpfungen erreichbar sind. Dennoch unterscheiden sich die Datenmodelle im Grad und Informationsgehalt ihrer Objektvernetzung, die unter anderem durch die Anzahl der gemeinsam verwendeten Objekte, die Häufigkeit von Umkehrrelationen und nicht zuletzt die Art der Verknüpfungen bestimmt wird. Seitens der Objektvernetzung ist dementsprechend zu erwarten, dass die Qualität der Vergleichsergebnisse je nach Datenmodell variiert. Eine Bewertung der erreichbaren Ergebnisse ist außerdem von den Anwenderprogrammen und der Qualität der erzeugten Daten abhängig.

Auf der Grundlage der vergleichenden Betrachtung mit dem IFC-Modell ist zu erwarten, dass das vorgeschlagene Verfahren für die untersuchten Datenmodelle tendenziell ähnliche Ergebnisse liefern wird. Eine verallgemeinerte qualitative Bewertung ist nur eingeschränkt möglich, da der Einfluss der verschiedenen Faktoren nur bei detaillierteren Untersuchungen eingeschätzt werden kann.

[1] Auf Grund der eingeschränkten Identifizierbarkeit sind nach der Definition von Ullman (1988) die meisten Datenmodelle zumindest in Teilen als '*value oriented*' und daher nicht als '*object oriented*' zu bezeichnen.

8 Abschließende Bemerkungen

In diesem Kapitel wird abschließend der Beitrag zum Themenkomplex der Datenintegration und dessen praktische Bedeutung diskutiert.

Der *erste Abschnitt* fasst die Arbeit zusammen und stellt den Bezug zum Problem der Datenintegration her. Als Grundlage der Datenintegration werden Produktdatenmodelle gesehen, die mit einem neuartigen Ansatz der Datenverwaltung kombiniert werden. Der Ansatz geht davon aus, dass ein Produktdatenmodell im Sinne einer objekt-orientierten Datenbankverwaltung nicht fehlerfrei unterstützt werden kann. Daher wird die Strategie verfolgt, diese Fehler über die Verwendung von Teildatensätzen zu reduzieren bzw. durch eine Versionsverwaltung und das Zurückführen auf Änderungen erkennen zu können. Auf diese Weise ist es möglich, die Qualität und Aussagekraft der gemeinsamen Datenbasis schrittweise zu verbessern.

Den Schwerpunkt des Kapitels bildet der *zweite Abschnitt*, der eine Bewertung der Ergebnisse vornimmt. In der Diskussion der Problemstellung wurde deutlich, dass das Problem der Datenintegration zu einem hohen Maß auf das Fehlen von formalisiertem Fachwissen zurückzuführen ist. Vor diesem Hintergrund wird der Stellenwert des vorgestellten, generischen Lösungsansatzes betrachtet. Es wird gezeigt, welche Voraussetzungen für die Anwendung der entwickelten Methoden erfüllt sein müssen und wo die Grenzen und Probleme des Ansatzes liegen.

Im *dritten Abschnitt* wird abschließend ein Ausblick auf weitere Arbeiten gegeben. Hierbei werden verbleibende Probleme, mögliche Lösungsansätze und die damit verbundenen Erwartungen benannt. Dieser Ausblick ist nicht nur auf den beschriebenen Lösungsansatz beschränkt, sondern betrachtet auch äußere, nicht direkt beeinflussbare Randbedingungen, die aus den Ergebnissen der Arbeit geschlussfolgert werden.

8.1 Zusammenfassung

In der Arbeit wurde ein neuartiger Ansatz für die Verwaltung gemeinsam genutzter Planungsdaten vorgestellt. Mit diesem Ansatz ist es möglich, die Zusammenarbeit auf der Grundlage eines gemeinsam akzeptierten Produktdatenmodells und unter Verwendung langer Transaktionen zu organisieren. Für dieses Szenario wurden Methoden entwickelt, die die Auswahl der benötigten Planungsdaten, das Aktualisieren der gemeinsamen Datenbasis und schließlich das Zusammenführen divergierender Planungsstände unterstützen.

8.1.1 Prinzipien des vorgestellten Ansatzes

Der vorgestellte Ansatz beruht auf den Prinzipien der *Transparenz*, *Flexibilität* und *Fehlertoleranz*. Durch die Kombination dieser Prinzipien werden zwei wesentlichen Problemstellungen der Datenverwaltung berücksichtigt: 1.) der Umgang mit unvollständig und fehlerhaft integrierten Daten und 2.) die Unterstützung der kreativen Phasen des Bauwerksentwurfs.

Transparenz der Planungsschritte

Mit dem Prinzip der Transparenz wird die Kontrolle der gemeinsamen Datenbasis und der durchgeführten Nutzerinteraktionen verfolgt. Die Grundlage für die Verwaltung der Planungsdaten bildet die Dokumentation der Planungsschritte. Ein Planungsschritt wird aber nicht als neuer Planungsstand, sondern über die vorgenommenen Änderungen beschrieben.

Dadurch ist es möglich, die Planungsschritte nachvollziehbar in der gemeinsamen Datenbasis abzulegen und für eine spätere Bewertung zu verwenden. Für diesen Kooperationsansatz wird dokumentiert,

- welche Planungsdaten für die Bearbeitung verwendet,
- welche Änderungen durchgeführt und schließlich
- wie diese Änderungen in die gemeinsame Datenbasis integriert wurden.

Über die vorgenommenen Änderungen, die auf der Ebene der verwalteten Planungsdaten die Überarbeitung des Bauwerksentwurfs beschreiben, sind mögliche Informationsverluste und fehlerhafte Daten wesentlich einfacher als bisher erkennbar.

Flexibilität der Modelle und Methoden

Mit dem Prinzip der Flexibilität wird die Verallgemeinerung des Ansatzes und der entwickelten Methoden verfolgt. Damit werden vor allem die Vielfalt und Heterogenität der verwendeten Modelle und Anwenderprogramme sowie die datentechnisch kaum reglementierbaren Entwurfsschritte berücksichtigt. Durch das Zurückführen auf allgemein akzeptierte Grundlagen der Datenverwaltung wird ein Maß an Allgemeingültigkeit erreicht, der den Ansatz für verschiedene Szenarien nutzbar werden lässt.

Die entwickelten Methoden und Modelle beruhen auf dem objekt-orientierten Modellierungsparadigma und sind dadurch unabhängig von der Semantik der verwalteten Datenmodelle. Auf diese Weise wird von Modellierungsphilosophien abstrahiert und eine neutrale Basis für die Verwaltung der Planungsdaten geschaffen. Die Dokumentation der Änderungen wird auf sechs Operationen zurückgeführt. Neben den drei Basisoperationen Erzeugen, Verändern und Löschen gehören hierzu das Teilen, Zusammenlegen sowie die Evolution von Objekten. Im Vergleich zu anderen Versionsansätzen wird eine liberalere Auslegung der Objektorientierung erreicht, die den Schwerpunkt auf die Entstehung und Veränderung der in den Objekten enthaltenen Planungsinformationen legt.

Toleranz gegenüber Unzulänglichkeiten der Datenintegration

Mit dem Prinzip der „Fehlertoleranz“ wird ein Konzept verfolgt, das die Probleme der Datenintegration akzeptiert und mit den hieraus resultierenden Unzulänglichkeiten umgehen kann. Damit wird die Tatsache berücksichtigt, dass die Qualität der Datenintegration in erster Linie von äußeren, nicht unmittelbar beeinflussbaren Faktoren bestimmt wird. Ein wesentliches Element des Lösungsansatzes ist daher die Möglichkeit, den negativen Einfluss einzelner Anwenderprogramme zu reduzieren. Der hierfür erforderliche Ausgleich von Informationsverlusten basiert auf der Strategie, den individuellen Informationsbedarf der Anwenderprogramme zu bedienen und Methoden zum Wiederherstellen der nicht handhabbaren Informationen bereitzustellen.

Mit dieser Strategie ist nicht nur der Verlust von Planungsinformationen, sondern auch das Fehlen wichtiger Verwaltungsinformationen auszugleichen. Für das Aktualisieren der Datenbasis entsteht dadurch neben den nicht auszuschließenden strukturellen und semantischen Konflikten eine weitere Ursache für Inkonsistenzen. Der Umgang mit temporär inkonsistenten Planungsständen stellt somit einen weiteren Aspekt der Fehlertoleranz dar, die folglich nicht nur wegen technischer Unzulänglichkeiten der Datenintegration, sondern auch durch „unvollständige“ Änderungen eines Planungsschrittes erforderlich ist.

8.1.2 Praxisbezug des betrachteten Kooperationsmodells

Mit dem gewählten Kooperationsmodell werden der Bezug zu den Problemen der Praxis hergestellt und vorhandene Integrationsbemühungen berücksichtigt. Hierzu gehören: 1.) die zentralisierte Nutzung standardisierter Datenmodelle, 2.) die Unterstützung langer Transaktionen und 3.) das Verfolgen einer optimistischen Kooperationsstrategie. Es wird somit ein Ansatz aufgezeigt, der den verlust- und fehlerbehafteten sequenziellen Datenaustausch durch die Nutzung einer gemeinsamen Datenbasis ingenieurgerecht ablösen kann.

Zentrale Verwaltung eines standardisierten Datenmodells

In dem erarbeiteten Ansatz wird ein allgemein akzeptiertes Datenmodell als Grundlage für den Austausch von Planungsinformationen genutzt. Hierbei wird die indirekte Kommunikation über eine gemeinsame Datenbasis verfolgt, die die zentrale Verwaltung der Planungsdaten ermöglicht und die Komplexität der direkten Kommunikation vermeidet. Die Informationsflüsse sind dadurch bekannt und stellen geringe Anforderungen für die Umsetzung in einem verteilten System.

Lange Transaktionen und die Verwendung von Planungsständen

Für die Ausführung der Planungsschritte werden lange Transaktionen und die Verwendung von Planungsständen unterstützt. Dadurch wird die in der Praxis dominierende unabhängige Bearbeitung der Planungsdaten berücksichtigt. Es können eigenständige Anwenderprogramme verwendet werden, die das gemeinsame Datenmodell über Import- und Exportfunktionen unterstützen. Durch diese moderaten Voraussetzungen können vorhandene Anwenderprogramme ohne Anpassungen in die Bearbeitung einbezogen werden.

Unterstützung der optimistischen Kooperationsstrategie

Mit dem vorgestellten Ansatz wird eine optimistische Kooperationsstrategie verfolgt, die das parallele Arbeiten ohne das Sperren von Planungsdaten unterstützt. Für die Phase der Bearbeitung ist dadurch ein hohes Maß an Flexibilität gegeben, das durch die Dokumentation der Planungsschritte ermöglicht wird. Hierbei entstehen voneinander abweichende Planungsstände, die in einem zusätzlichen Abstimmungsprozess mit Unterstützung der Datenverwaltung wieder zusammengeführt werden können. Durch diese Herangehensweise kann der Bearbeitungsprozess ohne Einschränkungen seitens der Datenbasis durchgeführt werden und kommt damit der gewohnten Arbeitsweise entgegen.

8.1.3 Entwickelte Modelle und Methoden

Im Rahmen der Arbeit wurden Modelle und Methoden entwickelt, die das beschriebene Kooperationsmodell unterstützen. Hierzu gehören: 1.) das der Datenverwaltung zugrunde liegende Versionsmodell, 2.) die Formalisierung der in einem Planungsschritt benötigten bzw. verarbeitbaren Planungsdaten, 3.) ein Verfahren zum Vergleich von Planungsständen und schließlich 4.) die Konzeption für das formale Wiederherstellen und Zusammenführen von Informationen.

Versionsmodell

Das entwickelte Versionsmodell bietet die Grundlage, um die Planungsschritte über die verwendete Änderungssemantik dokumentieren zu können. Im Unterschied zu anderen Versionsansätzen entsteht aus den abzubildenden Änderungsoperationen die Forderung, auch Verände-

rungen am Objektgefüge beschreiben zu können. Diese Flexibilität wird in dem vorgestellten Versionsmodell mit einer platzsparenden Speicherung über *Deltas* kombiniert. Das Versionsmodell beschreibt unter diesen Voraussetzungen die Organisation und den Zugriff auf die gespeicherten Informationen.

Partialmodellunterstützung

Die Bearbeitung in langen Transaktionen erfordert die Auswahl der benötigten Planungsdaten. Mit der hierfür entwickelten Spezifikation, dem *Generalised-Model-Subset-Definition*-Schema (GMSD), wird der dabei auftretende Konflikt zwischen einer möglichst einfachen, aber dennoch präzisen sowie flexiblen Beschreibung der gewünschten Planungsdaten thematisiert. Der Ansatz von GMSD sieht die Verwendung vordefinierbarer Filter vor, die auf Klassenebene beschrieben und in einer Abfrage mit der Auswahl von Objekten kombiniert werden. Bei der Auswertung eines Filters werden aber nicht nur die ausgewählten, sondern auch die referenzierten Objekte berücksichtigt. Dadurch ist es möglich, die erforderlichen Nutzerangaben auf die Auswahl weniger Objekte zu beschränken und gleichzeitig den Informationsbedarf sehr genau zu beschreiben.

Vergleichsverfahren

Beim Austausch von Planungsständen sind die durchgeführten Änderungen nachträglich zu bestimmen. Durch das Reduzieren der ausgetauschten Datensätze auf individuell handhabbare Planungsdaten werden auch Verwaltungsinformationen herausgefiltert, die für das Identifizieren der Planungsdaten und damit das Erkennen der durchgeführten Änderungen notwendig sind. Mit dem entwickelten Vergleichsverfahren ist es möglich, den Verlust an Verwaltungsinformationen durch die Bewertung der Objektbeziehungen zu berücksichtigen. Die wesentlichen Merkmale und zugleich Unterschiede zu anderen strukturbasierten Vergleichsverfahren sind die verwendete Änderungssemantik sowie die hohe Gewichtung der Objektbeziehungen, die einen effizienten Vergleich großer Datenmengen ermöglichen.

Konzept zur formalen Re-Integration

Nach der Beendigung eines Planungsschrittes wird der geänderte Teildatensatz mit den zuvor entfernten Informationen vervollständigt. In dem vorgestellten Verfahren werden konzeptionell die zuvor erkannten Änderungen auf den Gesamtdatensatz übertragen. Bei diesem formalen Ansatz können Widersprüche entstehen, die nicht nur auf technische Abläufe, sondern auch auf bewusst durchgeführte Änderungen zurückzuführen sind. Das Erkennen von Inkonsistenzen sowie die Unterscheidung ihrer Ursache kann von einem generischen Ansatz jedoch nur teilweise geleistet werden. Im Sinne eines nachvollziehbaren Planungsschrittes wird durch die Re-Integration deshalb ein Vorschlag für einen aktualisierten Gesamtdatensatz erstellt, der über vordefinierte Regeln die Konsistenzbedingungen des Versionsmodells erzwingt.

Konzept für das Zusammenführen von Änderungen

Durch unabhängiges, paralleles Arbeiten entstehen voneinander abweichende Planungsstände, die mit dem Ziel eines einheitlichen Planungsstandes wieder zusammengeführt werden sollen. Das hierbei verfolgte Prinzip ist vergleichbar mit der Re-Integration und versucht, die Änderungen eines parallel durchgeführten Planungsschrittes zu übernehmen. Durch die Verallgemeinerung auf n zusammenzuführende Planungsstände entstehen jedoch komplexere Abhängigkeiten, die für eine konsistente Abbildung in dem Versionsmodell zu berücksichtigen sind.

8.2 Bewertung der Ergebnisse

Der vorgestellte Ansatz liefert für die Verwaltung einer gemeinsamen Datenbasis eine datenstrukturbasierte Grundlage, die auf zusätzliches Fachwissen verzichtet und dadurch universell einsetzbar ist. Das erforderliche Fachwissen zur Bewältigung der Datenintegration wird an Anwenderprogramme, zusätzliche Werkzeuge, fachspezifische Erweiterungen der Serverfunktionalität und nicht zuletzt an das verwendete Datenmodell delegiert. Mit diesem Ansatz, der zwangsläufig von äußeren, nicht beeinflussbaren Faktoren abhängig ist, wird die Strategie verfolgt, die Zusammenarbeit entsprechend den gegebenen Randbedingungen verbessern zu können. In der nachfolgenden Bewertung werden die Ergebnisse der Arbeit unter diesem Aspekt betrachtet. Hieraus leiten sich die praktische Bedeutung sowie die minimal zu erfüllenden Anforderungen ab, die für einen nutzbringenden Einsatz der vorgestellten Methoden notwendig sind.

8.2.1 Konzept des Versionsmodells

Für die Bewertung des Versionsmodells sind im wesentlichen die verwendete Änderungssemantik und ihre Abbildung in Deltas zu diskutieren, die die Dokumentation der Planungsschritte mit der gewünschten Flexibilität ermöglichen.

Bewertung der Änderungssemantik

Abweichend von dem objekt-orientierten Prinzip werden neben den Basisoperationen *Erzeugen*, *Löschen* und *Ändern* auch das *Teilen*, *Zusammenlegen* sowie die *Evolution* von Objekten unterstützt. Durch diese Operationen wird die Aussagekraft der dokumentierten Änderungen erhöht. Dies ist vorteilhaft, wenn eine Verfeinerung (*top-down*) oder Verallgemeinerung (*bottom-up*) des Entwurfs erfolgt. Darüber hinaus können Änderungen erfasst werden, die aus ungewollten strukturellen Veränderungen resultieren. Mit der Änderungssemantik ist es demzufolge möglich, ohne die strikte Bindung an Datenobjekte die Änderungen der Planungsinformationen zu dokumentieren. Dadurch wird vermieden, dass Änderungen bei einer Unverträglichkeit mit dem objekt-orientierten Konzept auf das Löschen und Erzeugen von Objekten zurückgeführt werden müssen und die Beziehungen zwischen den darin beschriebenen Informationen verloren gehen.

Im Vergleich zu operationsbasierten Ansätzen ist die Aussagekraft des vorgeschlagenen Versionsmodells deutlich geringer, da die Änderungen auf Grundlage der Datenstruktur und nicht auf Basis der vorgenommenen Modifikationsoperationen dokumentiert werden. Für das betrachtete Kooperationsmodell wird jedoch bewusst auf einen höheren Informationsgehalt verzichtet, um einerseits die Allgemeingültigkeit des Versionsansatzes zu erhalten und andererseits die Abhängigkeit von den beteiligten Anwenderprogrammen, die die Modifikationsoperationen bereitstellen müssen, zu vermeiden. Die mögliche Divergenz zwischen dem Inhalt und der Struktur der Daten wird mit diesem Ansatz demzufolge nicht überwunden und kann den Nutzwert der Änderungsinformationen beeinträchtigen.

Abbildung der Änderungen als Deltas

Bei der Abbildung der Änderungen wird eine platzsparende Speicherung über Deltas verfolgt. Hierbei werden Vorwärtsdeltas verwendet, die Änderungen gegenüber ihren Vorgängerversionen speichern. Dadurch wird das Umorganisieren der Datenbasis vermieden. Unabhängig vom Delta-Prinzip stellt die Konsistenz der Objektreferenzen ein wesentliches Problem einer platzsparenden Speicherung dar, da das Aktualisieren von Verknüpfungen zu einer Vervielfäl-

tigung neuer Objektversionen führt. Diese Vervielfältigung wird durch einen Ansatz vermieden, der indirekte Verknüpfungen ermöglicht und dadurch das Aktualisieren von Referenzen reduziert. Im Vergleich zu anderen Versionsansätzen wird das Auflösen indirekter Verknüpfungen nicht auf die Unterscheidung in Objekte und zugehörige Objektversionen zurückgeführt, sondern über die Auswertung der Versionsbeziehungen erreicht. Dadurch ist es möglich, Änderungen im Objektgefüge darzustellen und mit dem Ziel einer möglichst geringen Delta-Menge zu verbinden.

Durch die Nutzung indirekter Verknüpfungen und die Anwendung des Vorwärtsdeltaprinzips wird mit jeder neuen Version der Aufwand für das Regenerieren des aktuellen Planungsstandes erhöht. Für die Verwaltung mehrerer Planungsschritte können hieraus Wartezeiten entstehen, die nicht akzeptabel sind. Ein gewisser Berechnungsaufwand ist auch für das Ermitteln der Änderungen notwendig, da die gespeicherten Deltas für den Fall zusammengeführter Objektversionen bzw. zusammengelegter Objekte nicht direkt den vorgenommenen Änderungen entsprechen. Für die Dokumentation weniger Planungsschritte ist der damit verbundene Aufwand unkritisch. Soll die Verwaltung der Planungsschritte auf größere Planungsphasen ausgeweitet werden, so sind Ansätze zur Optimierung der Zugriffsgeschwindigkeit unverzichtbar.

8.2.2 Definition und Nutzung von Teildatensätzen

Die Nutzung von Teildatensätzen besitzt für das Kooperationsmodell eine hohe Bedeutung. Eine wesentliche Voraussetzung ist die effiziente Definition der verwendeten Teildatensätze. Mit dem hierfür entwickelten *Generalised-Model-Subset-Definition*-Schema wird der Aufwand zur Beschreibung eines Teildatensatzes begrenzt. Für eine Bewertung ist entscheidend, ob die benötigten Teildatensätze beschreibbar sind und welcher Aufwand für den Anwender entsteht.

Untersuchte Szenarien der Teildatensatzbeschreibung

Für die Anwendung vordefinierbarer Filter wurden zwei Szenarien untersucht:

1. die Kombination mit einem kompletten Planungsstand und
2. die Kombination mit ausgewählten Objekten.

Das erste Szenario beschränkt die Auswahl der Objekte auf die Zugehörigkeit zu einer bestimmten Objektklasse, die zusätzlich über die Bewertung der Verknüpfungen eingeschränkt werden kann. Hiermit lassen sich fach- bzw. problemspezifische Untermengen des Gesamtdatensatzes beschreiben, die fast vollständig vordefiniert werden können. Sofern die Attributbelegung für die Auswahl der benötigten Daten keine Bedeutung besitzt, ist dieser Ansatz ausreichend und damit sehr effizient. Soll die Nutzung eines Objektes dagegen zusätzlich von der Belegung eines Wertes abhängig gemacht werden, so ist vor Anwendung des Filters eine attributbasierte Auswahl der Objekte vorzunehmen. Im Rahmen der Untersuchungen wurde eine solche Auswahl auf wenige Schlüsselobjekte beschränkt, die anschließend durch vordefinierte Filter bewertet wurden. Um auf diese Weise den gewünschten Teildatensatz beschreiben zu können, müssen, ausgehend von den ausgewählten Schlüsselobjekten, alle zusätzlich benötigten Datenobjekte über Referenzen erreichbar sein. Der Teildatensatz enthält dadurch die ausgewählten bzw. gefilterten Objekte sowie den Pfad ihrer gegenseitigen Verknüpfung. Die Effizienz dieses Vorgehens wird durch die Anzahl der manuell auszuwählenden Objekte bestimmt, die von der Strukturierung des Datenmodells sowie dem gesuchten Teildatensatz

abhängig ist. Eine Bewertung der Effizienz ist somit nur in Abhängigkeit vom betrachteten Anwendungsfall möglich.

Anwendung für das IFC-Modell

Die Untersuchung anhand des IFC-Modells und praxisnaher Beispiele hat gezeigt, dass der vorgestellte Ansatz eine effiziente Auswahl der benötigten Teildatensätze erlaubt. Dies ist möglich, weil die Struktur des IFC-Modells die Anwendung des GMSD-Schemas einerseits durch Container-Klassen wie *IfcBuildingStorey* oder *IfcGroup* und andererseits durch eine starke, häufig bidirektionale Verknüpfung der Objekte begünstigt. Auf diese Weise lässt sich auch das von der IAI verfolgte Konzept der Views unterstützen. Die Verwendung vordefinierbarer Filter stößt jedoch an Grenzen, wenn Planungsdaten über generische Klassen wie *IfcProperty* nach dem Prinzip der Schlüssel-Wertpaare übertragen werden und dadurch die Semantik der Daten nicht auf Klassenbasis, sondern über die Wertbelegung des Schlüsselattributs unterschieden wird. Eine Differenzierung auf dieser Ebene definiert die Grenzen der Filterdefinition, die nur in Ergänzung mit der Objektauswahl überwunden werden kann.

Verallgemeinerte Bewertung

Die Stärke des GMSD-Ansatzes liegt in der Bewertung der Objektverknüpfungen, die vor allem bei stark vernetzten Modelldaten vorteilhaft ist. Für die untersuchten Beispiele hat sich die Kombination der Objektselektion mit einem vordefinierbaren Filter als ausreichend und sehr effizient erwiesen. Es bleibt jedoch zu klären, wie die Teildatensatzbeschreibung auf zulässige Objekt-Filter-Kombinationen beschränkt werden kann. Für die Beschreibung wertabhängiger Teildatensätze ist auch die Vordefinierbarkeit von Objektselektionen näher zu untersuchen, um die Grenzen der Filterdefinition überwinden zu können.

8.2.3 Nachträgliches Ermitteln durchgeführter Änderungen

Das nachträgliche Ermitteln der durchgeführten Änderungen ist eine Notwendigkeit, die aus dem Umgang mit Planungsständen resultiert. Mit dem erarbeiteten Vergleichsverfahren kann das Fehlen eindeutiger Objektkennungen kompensiert werden, um die Planungsinformationen identifizieren und in ihrer Entwicklung verfolgen zu können. Damit wird versucht, einen im Planungsprozess aufgetretenen Informationsverlust durch einen generischen Algorithmus auszugleichen. Diese Zielsetzung ist nur bedingt erreichbar und daher als fehlerbehaftet zu akzeptieren. Unter diesen Voraussetzungen ist entscheidend, zu welchem Grad der Informationsverlust ausgeglichen werden kann und welche Fehlerquote dabei zu erwarten ist. Eine Bewertung des Vergleichsalgorithmus ist somit auch hier nur für konkrete Anwendungsszenarien möglich, weil die Qualität der Vergleichsergebnisse durch das zugrunde liegende Datenmodell, den verwendeten Teildatensatz und nicht zuletzt die vorgenommenen Änderungen beeinflusst wird.

Vergleich von IFC-Datensätzen

Die Untersuchungen für das IFC-Modell haben gezeigt, dass der vorgeschlagene Vergleichsalgorithmus auch bei einem sehr geringen Anteil eindeutig identifizierbarer Objekte eine korrekte Zuordnung der anderen Objekte und damit das Erkennen der Änderungen erreichen kann. Für stark veränderte Datensätze wird diese hohe Erkennungsrate jedoch nicht erzielt und enthält statt dessen einen relativ hohen Anteil nicht zugeordneter Objekte. Dieses Verhalten ist für den Schritt der Re-Integration vorteilhaft, weil eine nicht zugeordnete Information besser als eine falsch zugeordnete Information zu bewerten ist. Insgesamt problematisch ist

das Erkennen geteilter und zusammengelegter Objekte, da dies im Widerspruch zur eindeutigen Objektkennung steht. Der Vergleichsalgorithmus kann diese Art von Änderungen demzufolge nur dann erkennen, wenn die Zuordnung der Objekte über die Bewertung der Verknüpfungen erfolgt. Trotz dieser Einschränkung hat sich in den Untersuchungen gezeigt, dass das Teilen bzw. Zusammenlegen unter bestimmten Voraussetzungen als Änderung ermittelt wird und durchaus eine brauchbare Information darstellen kann. Diesem Vorteil steht mitunter jedoch ein unverhältnismäßiges Anwachsen der Änderungsmenge gegenüber, deren Änderungsinformation nicht immer sinnvoll ist. In einem solchen Fall kann der Verzicht auf die zugrunde liegende Teilungsinformation vorteilhaft sein.

Verallgemeinerte Bewertung

Der vorgestellte Vergleichsalgorithmus ist in der Lage, trotz fehlender Objektkennungen eine Zuordnung der Planungsinformationen zu ermöglichen. Das vorgestellte Verfahren arbeitet effizient, wenn ein gewisser Anteil eindeutig identifizierbarer Objekte vorhanden und über die Bewertung der Verknüpfungen eine Zuordnung der verbleibenden Objekte möglich ist. Ein algorithmisch hoher Aufwand ist zu beobachten, wenn eine hohe Anzahl von Objekten trotz hochgradiger Vernetzung über ihre Wertbelegung zuzuordnen ist. Durch den Umgang mit Teildatensätzen, die ebenfalls über Verknüpfungen realisiert werden, wird die Anwendung des Vergleichsverfahrens begünstigt. Dies geschieht, weil einerseits die zu vergleichende Objektmenge begrenzt wird und andererseits die Verknüpfungen zwischen den gesuchten Informationen erhalten bleiben.

8.2.4 Re-Integration und Zusammenführen

Die Re-Integration und das Zusammenführen stehen am Ende eines Entwurfsschrittes und werden mit dem Problem der Datenkonsistenz konfrontiert. Erst mit dieser Funktionalität wird das kooperative Arbeiten mit einer gemeinsamen Datenbasis möglich. In diesen beiden Teilschritten kumulieren die Probleme der Datenintegration, die in dem betrachteten Szenario einerseits aus fehlerhaft ermittelten Änderungen und andererseits aus den Unzulänglichkeiten der Datentransformationen entstehen. Mit den beiden vorgestellten Verfahren wird auf der Basis der Datenstruktur, dem verwendeten Teildatensatz und den ermittelten Änderungen, ein Vorschlag für einen aktualisierten bzw. zusammengeführten Planungsstand abgeleitet. Dieser Vorschlag ist durch die beteiligten Anwender auf Richtigkeit zu überprüfen und gegebenenfalls zu korrigieren. Der Nutzen des generischen Ansatzes wird schließlich durch den erforderlichen Korrekturaufwand bestimmt.

Kriterien für den erforderlichen Korrekturaufwand

Für das Aktualisieren der gemeinsamen Datenbasis sind die Auswirkungen der Änderungen auf den zu aktualisierenden Datenbestand entscheidend. Sind die Auswirkungen lokal begrenzt, so können die hierfür verantwortlichen Änderungen ohne ein implizites Verändern anderer Planungsinformationen übernommen werden. Haben Änderungen globale Auswirkungen, so ist die automatische Integration der Änderung zu vermeiden. In einem solchen Fall können andere Informationen ungewollt verändert werden und erzeugen einen erhöhten Korrekturbedarf.

Ein weiteres Kriterium für das Aktualisieren der gemeinsamen Datenbasis ist die Beschaffenheit der verwendeten Teildatensätze. Hierbei sind Informationen zu ergänzen, die zuvor beim Bilden des Teildatensatzes entfernt wurden. Für ein problemloses Aktualisieren ist es dementsprechend erforderlich, dass der Modellvergleich eine fehlerfreie Zuordnung der Informatio-

nen sicherstellen kann. Diese Forderung sollte zumindest für Objekte erfüllt werden, die bei dem Schritt der Re-Integration zu vervollständigen sind. Es ist daher vorteilhaft, wenn der verwendete Teildatensatz für diese Objekte eine geringe Fehlerquote garantieren kann.

Bewertung für das IFC-Modell

Für das IFC-Modell sind die vorgeschlagenen generischen Verfahren nur unter bestimmten Voraussetzungen effizient. Zu diesen Voraussetzungen gehört, dass Änderungen mit globalen Auswirkungen erkannt werden können. In dem IFC-Modell besteht die Gefahr einer unverträglichen Änderung vor allem durch die globale Definition der verwendeten Maßeinheiten sowie die relative Platzierung geometrischer Elemente. Für das Erkennen kritischer Änderungen werden dementsprechend IFC-spezifische Erweiterungen benötigt, die eine automatisierte Aktualisierung verhindern bzw. korrigierend eingreifen können.

In den IFC sind wichtige Objekte zumeist eindeutig oder zumindest mit einer hohen Wahrscheinlichkeit identifizierbar. Für die untersuchten Teildatensätze werden wichtige Schnittstellen, die für das Herausschneiden des Teildatensatzes bedeutsam sind, mehrheitlich durch solche Objekte definiert. Diese Tatsache ist für den Schritt der Re-Integration vorteilhaft und kann dadurch grob-fehlerhafte Ergebnisse zumeist vermeiden. Darüber hinaus besitzen die IFC einen hohen Differenzierungsgrad der Objektklassen, die die Planungsinformationen auf vergleichsweise viele Objekte verteilt. Dadurch ist es möglich, bereits auf Klassenebene die benötigten Planungsdaten weitgehend zu spezifizieren. Mit dieser Strukturierung ist der Effekt zu beobachten, dass Objekte, die eine hohe Wahrscheinlichkeit einer fehlerhaften Objektzuordnung besitzen, relativ selten zu vervollständigen sind. Dieser Effekt ist insgesamt positiv und reduziert den Verlust an Informationen.

8.2.5 Fazit und Beitrag der Arbeit zu den Problemen der Praxis

Die vorgestellten Modelle und Methoden unterstützen in ihrer Gesamtheit das parallele, unabhängige Bearbeiten von Teildatensätzen und adressieren damit praxisrelevante Probleme. Jeder der diskutierten Teilaspekte wird mit seiner spezifischen Funktionalität für die Verwaltung gemeinsamer Planungsdaten benötigt und wirkt sich auf die Qualität des Endergebnisses aus. Die hierdurch erreichbare Qualität bestimmt schließlich den Wert des vorgeschlagenen Verfahrens, der zusätzlich von äußeren, nicht direkt beeinflussbaren Faktoren abhängig ist. Das Verfahren erhebt dabei nicht den Anspruch, fehlerfrei zu arbeiten und gibt damit ein unter Praxisbedingungen nur schwer erreichbares Ideal auf. Vielmehr wird ein Ansatz aufgezeigt, der die Verwaltung einer gemeinsamen Datenbasis schrittweise verbessern kann. In der vorgestellten Arbeit wurden hierfür allgemeingültige Grundlagen gelegt und anhand des IFC-Modells verifiziert. Für praxisnahe Beispiele konnte die Eignung des Ansatzes und der entwickelten Methoden gezeigt werden. Eine praxistaugliche Nutzung ist auf diesem Stand jedoch noch nicht möglich. Hierfür werden modellspezifische Erweiterungen benötigt, die die Interaktion mit dem Anwender unterstützen und den Umgang mit den dokumentierten Änderungen erlauben. In der Möglichkeit der Erweiterbarkeit wird nicht zuletzt ein wesentlicher Vorteil des Ansatzes gesehen, der zusätzlich eine Modularisierung und somit eine Arbeitsteilung verfolgt. Mit dieser Herangehensweise ist es möglich, die vorgestellten Dienste schrittweise zu verbessern und auf konkrete Problemstellungen anzupassen.

Die Untersuchungen für das IFC-Modell haben verdeutlicht, dass mit einem überschaubaren Aufwand die Grenzen des generischen Ansatzes überwunden werden können. Für die Praxis ist das vorgestellte Verfahren vor allem für das Arbeiten mit Teildatensätzen interessant, weil

der sonst übliche Informationsverlust deutlich reduziert werden kann. Auch wenn das Verfahren den Verlust von Informationen nicht ausschließen kann, so sind über die Dokumentation eines Bearbeitungsschrittes die vorgenommenen Änderungen jederzeit einsehbar und können für die Korrektur des Planungsstandes genutzt werden. Neben der Verringerung der Informationsverluste ergibt sich für den Anwender der Vorteil, dass dieser Prozess nachträglich auf mögliche Fehler kontrolliert werden kann. Diese Eigenschaft ist nicht nur für die Qualität der Planungsdaten wichtig, sondern auch, um das Vertrauen der Anwender zu gewinnen.

8.3 Ausblick

Die Anwendung des vorgestellten Verfahrens verdeutlicht die Grenzen des generischen Ansatzes. In den untersuchten Beispielen ist es daher erforderlich, die verbleibenden Probleme mit Hilfe der beteiligten Anwender manuell zu beseitigen. Eine effiziente Anwendung der erarbeiteten Methoden ist aber nur dann realistisch, wenn die zusätzlich erforderlichen Anwenderinteraktionen einerseits begrenzt und andererseits möglichst nutzerfreundlich gestaltet werden. Hierfür sind Erweiterungen nötig, die auf das verwaltete Datenmodell abgestimmt sind. Außerdem sind verbesserte Randbedingungen wünschenswert, die hinsichtlich verschiedener Aspekte formuliert werden können.

8.3.1 Notwendige Erweiterungen

Als notwendige modellspezifische Erweiterungen werden einerseits die Verbesserung der generischen Verfahren und andererseits ein anwenderfreundlicher Umgang mit den dokumentierten Änderungen gesehen. Beide Strategien sind notwendig, um mit einem vertretbaren Aufwand eine effiziente Anwendung des vorgeschlagenen Verfahrens zu erreichen. Für eine langfristige Anwendung ist zusätzlich zu klären, wie die Verwaltung der Änderungen auf die Dokumentation mehrerer Planungsschritte erweitert werden kann.

Verbesserung der generischen Verfahren

Als kritisch haben sich vor allem die Vergleichsergebnisse erwiesen, die unter praxisnahen Bedingungen nicht immer brauchbar sind. Eine wesentliche Ursache ist die Divergenz zwischen dem Inhalt und der Struktur der Daten, die den formalen Schritt der Re-Integration durch die Gefahr unverträglicher Änderungen behindert. Um die Verträglichkeit der Änderungen zu gewährleisten, ist für kritische Planungsdaten eine Korrektur vorzusehen. Für das IFC-Modell hat sich gezeigt, dass eine solche Korrektur für die global verwendeten Maßeinheiten und die Bezugskoordinatensysteme notwendig ist. Erst durch diesen Schritt kann bei unverträglichen Änderungen ein für das weitere Vorgehen sinnvolles Vergleichsergebnis erzielt werden. Auf der Grundlage des generischen Vergleichsverfahrens können weitere Verbesserungen durch eine Anpassung der Zuordnungsregeln erzielt werden. Beispielsweise ist es denkbar, je nach Objektklasse bestimmte Formen der Änderung auszuschließen und auf diese Weise die Aussagekraft der Änderungen zu erhöhen. Ein ähnliches Vorgehen ist auch für die Re-Integration und das Zusammenführen vorstellbar, indem in kritische, bisher nur auf Basis der Datenstruktur getroffene Entscheidungen die Bedeutung der Daten einbezogen wird. Die Ergebnisse der generischen Verfahren können auf diese Weise mit einem überschaubaren Aufwand mitunter deutlich aufgewertet werden.

Vereinfachung der Nutzerinteraktionen durch spezialisierte Clients

Trotz modellspezifischer Erweiterungen wird das Aktualisieren des Gesamtdatensatzes nicht ohne Nutzerinteraktionen durchführbar sein. Hierfür sind zusätzliche Clients notwendig, die

den Anwender über die erkannten Änderungen, den aktualisierten Gesamtdatenstand und vor allem über den parallel durchgeführten Planungsschritt angemessen informieren können. Mit diesen Clients soll auch die Korrektur der Änderungen vereinfacht werden, indem der Anwender wie gewohnt eine fachspezifische Aufbereitung und Manipulation der Daten nutzen kann. Dadurch sind auch Einschränkungen korrigierbar, die aus dem objekt-orientierten Konzept resultieren und die Dokumentation der geänderten Informationen verfälschen.

Dokumentation mehrerer Planungsschritte

Die Nutzung des vorgestellten Versionsmodells betrachtet vor allem die Unterstützung einzelner Planungsschritte. Für eine vollständige Dokumentation der Planung sind zusätzliche Anstrengungen erforderlich, um einerseits die anfallenden Datenmengen zu bewältigen und andererseits einen ausreichend schnellen Zugriff auf die Planungsdaten zu gewährleisten. Hierfür ist eine Optimierung des Versionsmodells notwendig, die beispielsweise über die Auslagerung älterer Planungsstände oder die Verwaltung vollständiger Zwischenstände realisiert werden kann. Mit einer solchen, eher langfristigen Zielsetzung ist auch die Konsolidierung einzelner Planungsprozesse überlegenswert, die die verwendeten und veränderten Teildatensätze aus der Versionshistorie entfernt. Dies ist vor allem dann sinnvoll, wenn der Gesamtdatensatz wie gewünscht aktualisiert wurde und keine weiteren Probleme aus der Nutzung der Teildatensätze zu erwarten sind.

8.3.2 Verbesserung der äußeren Randbedingungen

Neben der Verbesserung des vorgestellten Lösungsansatzes sind auch Veränderungen an den äußeren Randbedingungen wünschenswert. Aus den Erfahrungen der Arbeit können Vorschläge hinsichtlich der verwendeten Datenmodelle, der genutzten Anwenderprogramme und der verteilten Softwarearchitektur formuliert werden. Nicht weniger wichtig als diese technischen Aspekte ist die Integration der Anwender, die als abschließende Betrachtung den Grundgedanken der Arbeit unterstreicht.

Datenmodelle

Die Standardisierung eines Datenmodells ist nicht ohne Kompromisse möglich. Um möglichst viele der gestellten Anforderungen zu erfüllen, wird eine Vielzahl möglicher Darstellungsformen unterstützt. Die resultierende Beschreibungsvielfalt, die zu komplexen Abhängigkeiten zwischen dem Inhalt und der Struktur der Daten führt, ist für die Anwendung strukturbasierter Methoden nicht ohne Probleme. Aus Sicht der Datenverwaltung ist es daher wünschenswert, die bestehende Beschreibungsvielfalt einzuschränken. Diese Forderung ist vor allem dann sinnvoll, wenn mit der Wahl einer anderen Darstellungsform eine größere Anpassung inhaltlich unveränderter Daten erforderlich ist.

Vor dem Hintergrund der Beschreibungsvielfalt erscheinen die an das objekt-orientierte Paradigma angelehnten Identifizierungskonzepte als zu unflexibel. Diesbezüglich wird ein leistungsfähigerer Ansatz gewünscht, der die Abhängigkeiten zwischen dem Inhalt und der Struktur der Daten berücksichtigen kann. Zusätzlich wird eine Regelung über die „identifizierbaren Einheiten“ der Datenstruktur gewünscht. In Modelldefinitionen wie IFC ist nämlich erkennbar, dass einzelne Objekte als identifizierbare Einheit als zu detailliert angesehen werden. Die logisch zusammengehörigen Informationen werden statt dessen über ein identifizierbares Objekt und seine Objektverknüpfungen beschrieben. Dieser Umstand wird von dem erarbeiteten Vergleichsverfahren genutzt. Bessere Vergleichsergebnisse sind zu erwarten, wenn die Abgrenzung der identifizierbaren Einheiten nachvollziehbar geregelt werden könnte.

Einen weiteren Aspekt der Modelldefinition stellt die Erweiterung der Integritätsbedingungen dar, die eine bessere Entscheidungsgrundlage für die Re-Integration von Teildatensätzen und das Zusammenführen von Änderungen liefern kann. Zu nennen ist vor allem die Formalisierung der Existenzabhängigkeit, um „überflüssige“ Instanzen wie ungenutzte Punkte erkennen zu können. Hierfür ist eine Abgrenzung zu „unvollständigen“ Instanzen nötig, die zu einem späteren Zeitpunkt vervollständig werden sollen.

Anwenderprogramme

Die eingesetzten Anwenderprogramme entscheiden über die Qualität der Planungsdaten. Aus diesem Grund besteht hier ein sehr hohes Optimierungspotenzial. An erster Stelle steht die bessere Unterstützung der Datenmodelle, die vor allem an der Fähigkeit zur Aktualisierung eines Datensatz gemessen werden kann. Für die Arbeit mit Teildatensätzen ist in diesem Zusammenhang wünschenswert, dass bestimmte Objekte, die beispielsweise den Bezug zum Gesamtdatensatz herstellen, als unveränderbar deklariert werden können. Weitergehende Verbesserungen sind durch die Anbindung der Anwenderprogramme an die Datenverwaltung erreichbar. Ideal wäre der Datenabgleich über die Angabe der durchgeführten Änderungen sowie die Unterstützung der notwendigen Anwenderinteraktionen. Durch die Vielzahl der eingesetzten Anwenderprogramme ist diesbezüglich aber auch weiterhin mit einer großen qualitativen Streuung zu rechnen.

Verteilte Softwarearchitektur

Die Umsetzung in eine verteilte Softwarearchitektur bietet eine Vielzahl von Optionen, die den Ansatz leistungsfähiger und flexibler gestalten können. In dem vorgestellten Lösungsansatz wird von einer festen Konfiguration ausgegangen, die eine Zuordnung der benötigten Dienste auf die zentrale Datenverwaltung und die dezentralen Anwenderprogramme vornimmt. Diese Konfiguration stellt nicht immer ein Optimum dar und sollte individuell anpassbar sein. Voraussetzung für eine flexible Konfiguration ist die Identifizierung und Modularisierung der benötigten Dienste, die dann auch von Drittanbietern als Plug-ins oder Web-Services bereitgestellt werden können. Neben den erarbeiteten Methoden, die in diesem Sinne jeweils einen Dienst bereitstellen, gehören hierzu vor allem die Datentransformation sowie die Konsistenzprüfung.

Beteiligte Anwender

Für eine erfolgreiche Zusammenarbeit ist neben der Qualität der technischen Lösung auch die Mitgestaltung durch die beteiligten Anwender entscheidend. Die technischen Möglichkeiten reichen vielfach nicht aus, um die hohen Anforderungen an gemeinsame Planungsdaten zu erfüllen. Aus diesem Grund ist die Bereitschaft der Anwender erforderlich, in die Qualität der gemeinsamen Planungsdaten zu investieren. Aus Sicht des Autors ist es daher nicht dienlich, wenn die Möglichkeiten der Datenintegration mit zu hohen Erwartungen verbunden werden.

A Anhang

A.1 Elemente der graphischen Darstellungen

Zur Veranschaulichung des Lösungsansatzes werden verschiedene Abbildungen verwendet, die zur besseren Lesbarkeit auf einen einheitlichen Satz graphischer Elemente zurückgreifen. Dabei werden vorwiegend allgemein akzeptierte Symbole genutzt, die von anderen Darstellungsformen übernommen wurden. Für die Beschreibung des Lösungsansatzes wird zwischen zwei Sichten unterschieden, die einerseits für die Darstellung der Abläufe auf der Ebene der Planungsstände genutzt werden und andererseits das Versionsmodell auf der Ebene der Objekte und Attribute veranschaulichen. Beide Sichten werden in einer Abbildung häufig gemeinsam genutzt, um den Bezug zwischen beiden Ebenen herzustellen. Die Abgrenzung der beiden Sichten ist durch eine gestrichelte Linie gekennzeichnet.

Für die Beschreibung von Abläufen wird die IDEF-0 Notation verwendet (Mayer et. al 1992), mit deren Hilfe ein Vorgang über vier Parameter charakterisiert werden kann. Hierzu gehören Eingangs- und Ausgangsgrößen, die daran beteiligten Nutzer sowie die benötigten Werkzeuge bzw. Methoden. Mehrere Vorgänge können über ihre Eingangs- bzw. Ausgangsgrößen in Beziehung zueinander gesetzt werden. Dadurch können Abhängigkeiten bzw. die Abfolge der Vorgänge beschrieben werden. Als Eingangs- bzw. Ausgangsgrößen werden in dieser Arbeit ausschließlich Planungsstände, also Objektmenge, genutzt. Planungsstände sind durch ein eigenes Symbol gekennzeichnet und besitzen entweder einen Namen oder enthalten die zugehörigen Objekte, die über ihre Objektnamen in geschweiften Klammern aufgelistet sind.

In der Abbildung A-1 ist beispielhaft ein Vorgang dargestellt, der durch den Nutzer n_A und unter Verwendung bestimmter Werkzeuge und Methoden die beiden Planungsstände p_A und p_B in den Planungsstand p_C zusammenführt. Für den Planungsstand p_B ist zusätzlich ersichtlich, dass hierzu die Objekte a_1, b_1 und c_1 gehören. Über die Bezeichnung der Planungsstände sowie die Beschreibung der zugehörigen Objekte wird schließlich der Bezug zur zweiten, in dieser Abbildung nicht enthalten Sicht, der Sicht auf das Versionsmodell, hergestellt.

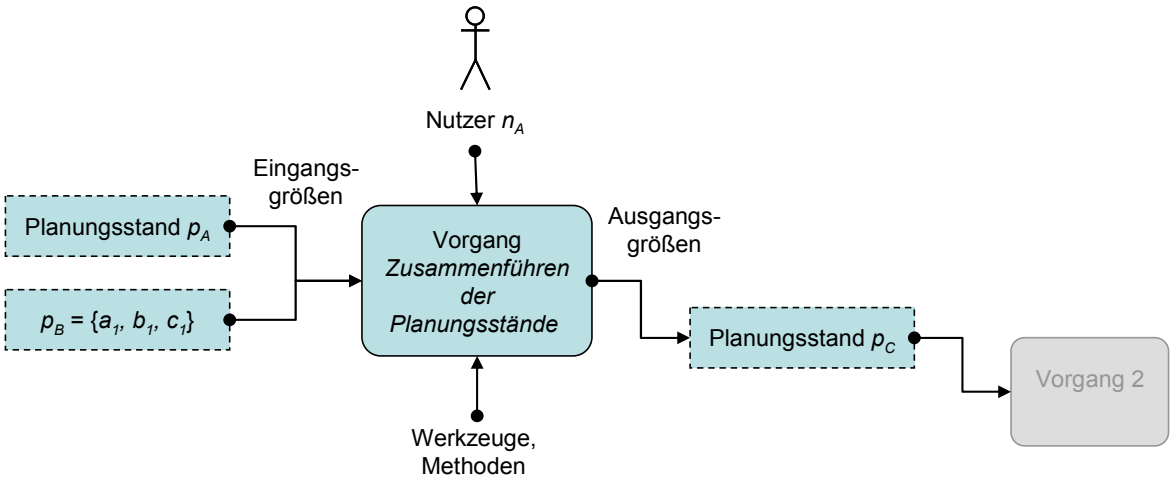


Abbildung A-1 Darstellung von Abläufen auf der Ebene von Planungsständen

Für die Darstellungen des Versionsmodells werden Symbole der UML-Notation verwendet (Zuser et. al 2004). Diese Notation wurde um Aspekte der Versionsverwaltung erweitert. In der Abbildung A-2 sind die verwendeten Symbole beispielhaft gezeigt, wobei grundsätzlich zwischen Objekten und Klassen unterschieden wird. Ein Objekt stellt die Ausprägung einer Klasse dar und steht als Träger der Daten im Mittelpunkt der Betrachtungen. Ein Objekt ist durch einen Objektnamen und wahlweise den Namen der zugehörigen Klasse gekennzeichnet. Zusätzlich kann ein Objekt durch seine Attribute beschrieben werden, die im Rumpf des Objektes mit den Attributnamen und den zugewiesenen Attributbelegungen enthalten sind. Zwischen Objekten können zwei Arten von Beziehungen definiert sein, einerseits die Verknüpfung oder Referenz, die einen nicht näher unterschiedenen Bezug zwischen Objekten des gleichen Planungsstandes[1] beschreibt, und andererseits die Versionsbeziehung, die im Rahmen des Versionsmodells eine Vorgänger-Nachfolger-Beziehung zwischen Objekten unterschiedlicher Planungsstände definiert. Durch die Vorgänger-Nachfolger-Beziehung wird die Weiterentwicklung eines Objektes gekennzeichnet, die durch eine Beschreibung der geänderten Attribute ausreichend definiert ist und dadurch nicht alle Attribute enthalten muss. So ist das Objekt a_1 in der Abbildung A-2 ein Nachfolger des Objektes *a*, der sich von seinem Vorgänger nur durch das Attribut *Höhe* unterscheidet. Im Gegensatz zu Objekten werden Klassen für die Beschreibung von Eigenschaften genutzt, die für eine Menge von Objekten gültig ist. Die Definition von Klassen und den hierauf definierten Strukturen wird im Rahmen dieser Arbeit nicht weiter untersucht und bleibt in den graphischen Darstellungen weitestgehend unberücksichtigt.

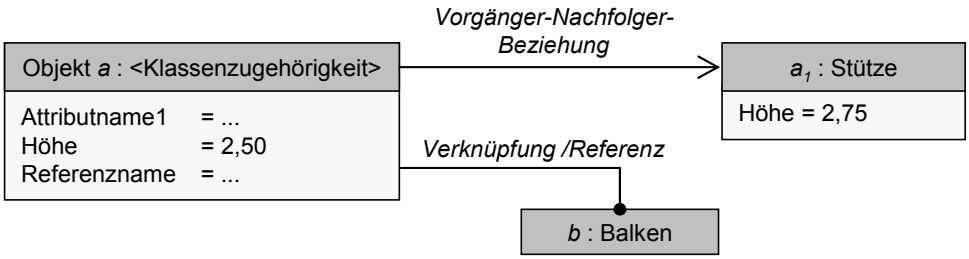


Abbildung A-2 Darstellung von Objekten und den beiden möglichen Verknüpfungstypen

Für die Darstellung von Datenstrukturen wird zusätzlich die EXPRESS-G Notation verwendet. Ausführliche Informationen über die EXPRESS-G Notation sind in (ISO 10303-21) zu finden.

[1] Einschränkend sei angemerkt, dass eine Verknüpfung nicht immer physisch auf ein Objekt des gleichen Planungsstandes verweist. Über das Versionsmodell wird jedoch sichergestellt, dass über diese Verknüpfung auf das eigentlich gesuchte Objekt des gleichen Planungsstandes geschlossen werden kann.

A.2 Verwendete Notationen

Für die Formalisierung des Lösungsansatzes wird ein Satz von Symbolen verwendet, der für die Beschreibung von Mengen und der hierauf definierten Mengenalgebra genutzt wird. Zusätzlich werden Symbole aus der Prädikatenlogik verwendet, um die Eigenschaften der Elemente und Relationen näher zu beschreiben. Folgende Symbole werden für die Formalisierung verwendet (Gellert et. al 1986, Pahl & Damrath 2000):

Name	**Symbol**	**Verwendung**
Negation	$\neg$	$\neg P$
Konjunktion	$\wedge$	$P \wedge Q$
Disjunktion	$\vee$	$P \vee Q$
Äquivalenz	$\Leftrightarrow$	$P \Leftrightarrow Q$
Definition	$:\Leftrightarrow$	$P :\Leftrightarrow Q$
	$:=$	$M := \{a, b, c\}$
Existenzaussage	$\exists$	$\underset{x}{\exists} P(x)$
Allaussage	$\forall$	$\underset{x}{\forall} P(x)$
Zugehörigkeit	$\in, \notin$	$x \in M,\ x \notin M$
Teilmenge	$\subseteq$	$N \subseteq M$
Echte Teilmenge	$\subset$	$N \subset M$
Gleichheit	$=$	$N = M,\ x = y$
Durchschnitt	$\cap$	$N \cap M$
Vereinigung	$\cup$	$N \cup M$
Differenz	$-$	$N - M$
Kreuzprodukt	$\times$	$M \times M$ oder M^2
Relation	R	$R(x, y)$ oder $(x, y) \in R$

Zusätzlich werden folgende Konventionen vereinbart:

- Mengen werden über Großbuchstaben beschrieben.
- Elemente einer Menge werden über Kleinbuchstaben beschrieben.
- Relationen werden über ein *R* gekennzeichnet und durch ihren Index unterscheiden.

Es ist zu beachten, dass in einem Mengensystem diese Konvention nicht durchgehend eingehalten werden kann. Ein Element einer Menge kann in einem Mengensystem nämlich selbst eine Menge sein. Großbuchstaben werden aus diesem Grund nur dann verwendet, wenn Mengen entweder definiert werden oder kein spezielles Element einer anderen Menge repräsentiert wird. Eine Menge kann auch in Abhängigkeit von anderen Elementen beschrieben wer-

den. In diesem Fall werden die beschreibenden Elemente in Klammern zusammengefasst und der Bezeichnung der Menge hinzugefügt. So beschreibt der Ausdruck $O_V(x, y)$ eine Menge, die in Abhängigkeit von den Elementen x und y generiert wird. In der Definition der Menge O_V wird zusätzlich definiert, welche Bedingung die Elemente x und y zu erfüllen haben. Im Normalfall wird diese Bedingung über die Zugehörigkeit zu einer anderen Menge ausgedrückt. Auf diese Weise lässt sich beispielsweise der Durchschnitt zweier Mengen wie folgt in der Menge $O_{DURCHSCHNITT}$ zusammenfassen:

$$\text{geg.:} \quad x, y \in O$$
$$O_{DURCHSCHNITT}(x, y) := \{ z \in O \mid z \in x \wedge z \in y \}$$

A.3 Das GMSD-Schema

Das Konzept des *Generalised-Model-Subset-Definition*-Schemas ist in Kapitel 5 für den Teil der Filterdefinition mengentheoretisch formalisiert. In Ergänzung zu dieser Beschreibung wird nachfolgend die vollständige Spezifikation vorgestellt. Das GMSD-Schema liegt als eine objekt-orientierte Datenstruktur vor, die in EXPRESS beschrieben ist. Für die graphische Darstellung der Entitäten und gegenseitigen Beziehungen wird daher die EXPRESS-G Notation verwendet. Zu jeder Entität wird nachfolgend eine kurze Beschreibung gegeben. Zusätzlich wird auf Unterschiede eingegangen, die zur Vereinfachung der Teildatensatzbeschreibung eingeführt wurden.

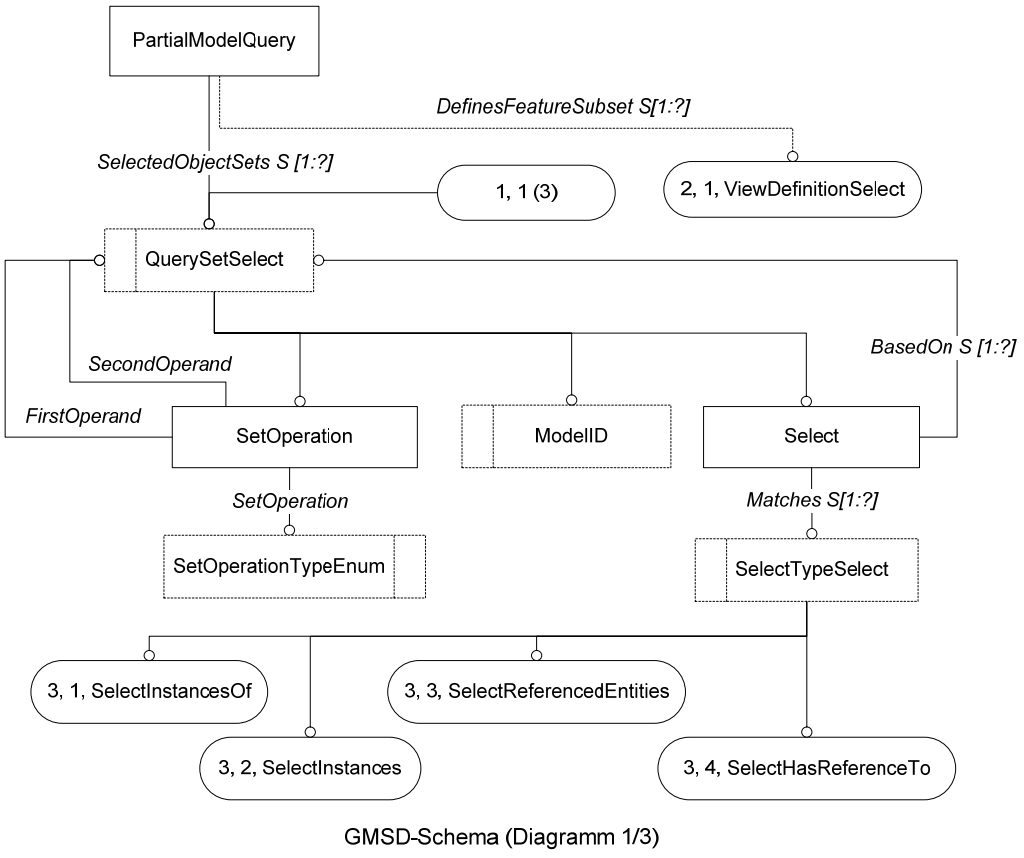
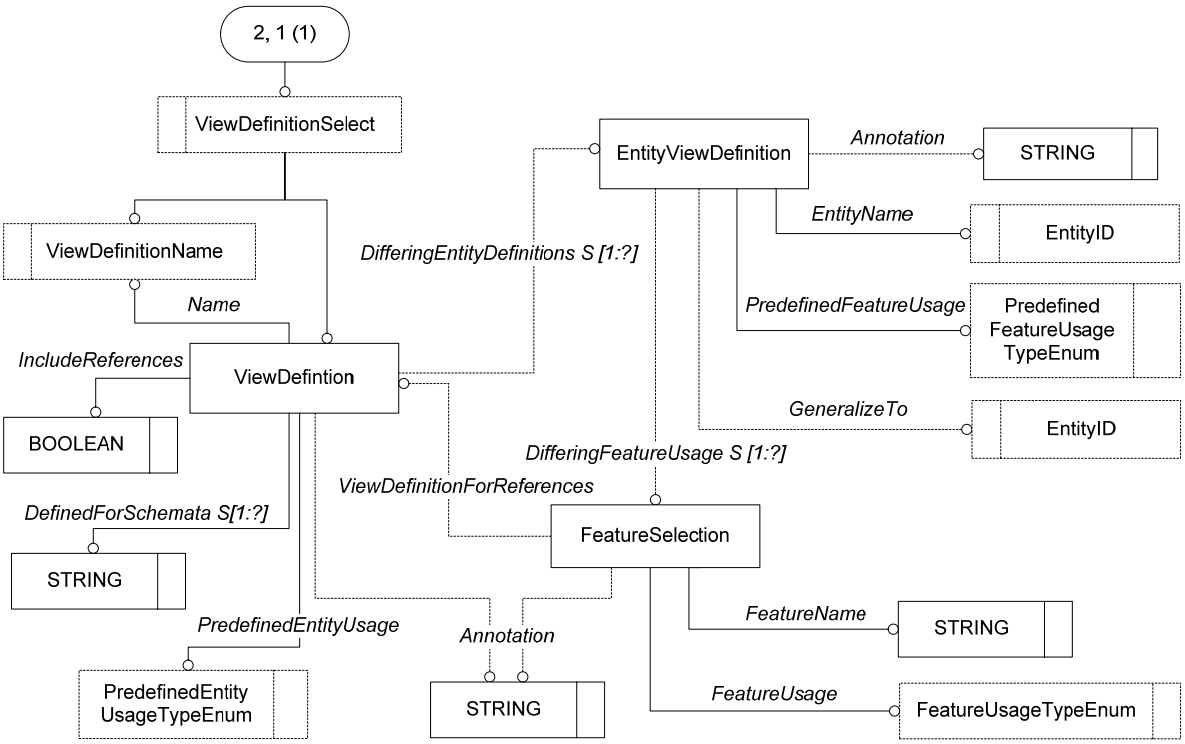
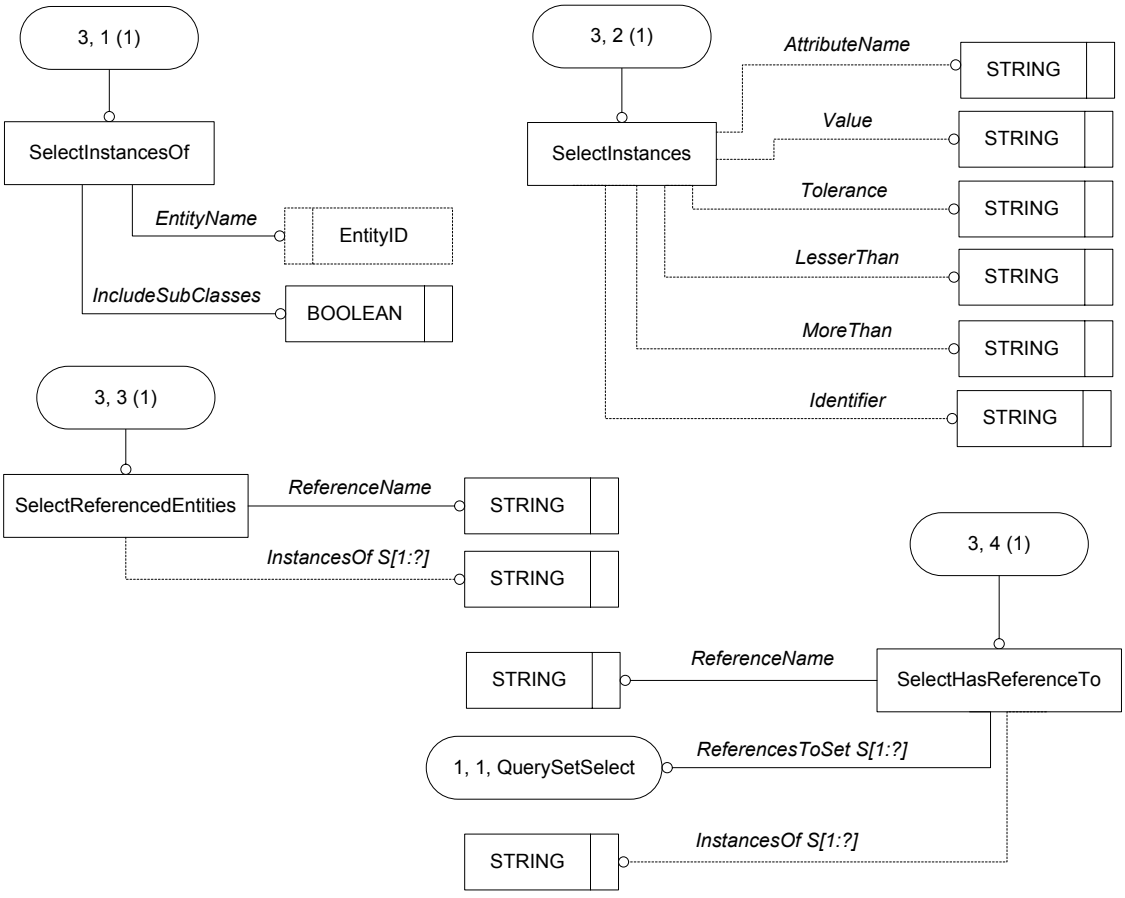


Abbildung A-3 EXPRESS-G Darstellung des GMSD-Schema (Diagramm 1/3)



GMSD-Schema (Diagramm 2/3)



GMSD-Schema (Diagramm 3/3)

Abbildung A-4 EXPRESS-G Darstellung des GMSD-Schema (Diagramme 2/3 und 3/3)

TYPE: ENTITYID

Definition: Die Typdefinition *EntityID* dient der eindeutigen Identifikation von Objektklassen. Für die Bezeichnung einer Objektklasse wird folgende Konvention verwendet: <Name des Schemas>.<Name der Objektklasse>.

EXPRESS Spezifikation:

```
TYPE EntityID = STRING;
END_TYPE;
```

Beispiel: Das nachfolgende Beispiel identifiziert die Objektklasse *IfcRoot* aus dem Schema *IFC2X_FINAL.*

```
IFC2X_FINAL.IfcRoot
```

TYPE: MODELID

Definition: Die Typdefinition *ModelID* wird verwendet, um Planungsstände eindeutig zu identifizieren.

EXPRESS Spezifikation:

```
TYPE ModelID = STRING;
END_TYPE;
```

TYPE: VIEWDEFINITIONNAME

Definition: Die Typdefinition *ViewDefinitionName* wird zur eindeutigen Identifizierung einer View-Definition verwendet. Sofern eine View-Definition über einen Namen definiert ist, so kann in einer GMSD-Anfrage stellvertretend für die vollständige View-Definition auch der Name der View-Definition verwendet werden.

EXPRESS Spezifikation:

```
TYPE ViewDefinitionName = STRING;
END_TYPE;
```

TYPE: SETOPERATIONTYPEENUM

Definition: Die Aufzählung *SetOperationTypeEnum* definiert Mengenoperationen, die von dem Entity *SetOperation* ausgeführt werden können. In der Aufzählung wird zwischen drei Typen unterschieden:

1) **Union** – beschreibt die Vereinigung zweier Objektmengen. Das Ergebnis ist eine Menge von Objekten, die keine Duplikate enthält[1].

[1] Diese Aussage ist für die Verwaltung von Objektmengen sinnvoll, da im Gegensatz zur Mengenlehre das mehrfache Vorhandensein eines Objektes möglich und an verschiedentlich gewünscht ist.

2) **Difference** – wird verwendet, um die als *second operand* gegebene Objektmenge von der als *first operand* gegebene Objektmenge abzuziehen. In der Ergebnismenge sind folglich nur Objekte enthalten, die in der Menge *first operand* und nicht in *second operand* enthalten sind.
3) **Intersection** – wird verwendet, um die Differenzmenge zwischen zwei Objektmengen zu ermitteln. In der Ergebnismenge sind folglich nur Objekte enthalten, die sowohl in der Menge *first operand* als auch in *second operand* enthalten sind.

EXPRESS Spezifikation:

```
TYPE SetOperationTypeEnum = ENUMERATION OF (Union,
                                            Difference,
                                            Intersection);
END_TYPE;
```

TYPE: PREDEFINEDENTITYUSAGETYPEENUM

Definition: Die Enumeration *PredefinedEntityUsageTypeEnum* wird genutzt, um per Voreinstellung die Auswertung für Objektklassen festzulegen, für die in der betrachteten *Viewdefinition* keine eigenständige Definition verfügbar ist. Es wird zwischen zwei möglichen Voreinstellungen unterschieden:

1) **UseAllEntities** – Alle Objekte, die zu einer nicht explizit definierten Objektklasse gehören, werden per Voreinstellung mit all ihren Attributen in die Teilmenge aufgenommen.
2) **RemoveAllEntities** – Alle Objekte, die zu einer nicht explizit definierten Objektklasse gehören, gehören per Voreinstellung nicht zu der gewünschten Teilmenge.

Über diese Voreinstellungen wird eine Vereinfachung der View-Definitionen angestrebt, die zusätzlich zu einer kompakteren GMSD-Abfrage beiträgt.

EXPRESS Spezifikation:

```
TYPE PredefinedEntityUsageTypeEnum =
                            ENUMERATION OF (UseAllEntities,
                                            RemoveAllEntities);
END_TYPE;
```

TYPE: PREDEFINEDUSAGETYPEENUM

Definition: Die Enumeration *PredefinedUsageTypeEnum* wird genutzt, um per Voreinstellung die Auswertung einer Objektklasse und ihrer Attribute festzulegen. Die Voreinstellung ist für Attribute gültig, für die in der betrachteten *EntityViewDefinition* keine eigenständige Definition verfügbar ist. Es wird zwischen den folgenden vier Voreinstellungen unterschieden:

1) **UseEntityWithAllFeatures** – Objekte mit dieser Voreinstellung werden mit allen Attributen in die Teilmenge aufgenommen. Wird über ein Attribut eine Referenz definiert, so können die referenzierten Objekte über die Auswertung der aktuell angewendeten *ViewDefinition* zusätzlich in die Teilmenge aufgenommen werden. Eine Referenz auf

ein anderes Objekt wird jedoch nur dann unterstützt, wenn das referenzierte Objekt über die GMSD-Beschreibung ebenfalls in die Teilmenge aufgenommen wird. Die Voreinstellung entspricht konzeptionell somit der *Selektionsrelation*, die mit der aktuell angewendeten *ViewDefinition* verknüpft ist.

2) **UseEntityAndDownsizeFeatures** – Objekte mit dieser Voreinstellung werden mit allen Attributen in die Teilmenge aufgenommen. Eine Referenz auf ein anderes Objekt wird jedoch nur dann unterstützt, wenn das referenzierte Objekt über die GMSD-Beschreibung ebenfalls in die Teilmenge aufgenommen wird. Eine Erweiterung der Teilmenge über die Aufnahme referenzierter Objekte wird bei dieser Voreinstellung jedoch nicht möglich. Die Voreinstellung entspricht konzeptionell somit der *Modellfilterrelation*, die mit der aktuell angewendeten *ViewDefinition* verknüpft ist.
3) **UseEntityAndRemoveAllFeatures** – Objekte mit dieser Voreinstellung werden ohne Attribute in die Teilmenge aufgenommen.
4) **RemoveEntity** – Objekte mit dieser Voreinstellung werden nicht in die Teilmenge aufgenommen.

Die Voreinstellungen für die Verwendung von Attributen kann durch eine eigenständige Attributdefinition ersetzt wird. Dies ist jedoch nur für die Voreinstellungen *UseEntityWithAll-Features*, *UseEntityAndDownsizeFeatures* und *UseEntityAndRemoveAll-Features* sinnvoll. Im Gegensatz dazu ist für die Voreinstellung *RemoveEntity* eine anderweitige Attributdefinition ohne Bedeutung, da mit dieser Einstellung das entsprechende Objekt nicht zur Teilmenge gehört.

EXPRESS Spezifikation:

```
TYPE PredefinedEntityUsageTypeEnum
              = ENUMERATION OF(UseEntityWithAllFeatures,
                               UseEntityAndDownsizeFeatures,
                               UseEntityAndRemoveFeatures,
                               RemoveEntity);
END_TYPE;
```

TYPE: FEATUREUSAGETYPEENUM

Definition: Die Enumeration *FeatureUsageTypeEnum* wird genutzt, um die Auswertung eines Attributes zu bestimmen. Es wird zwischen den folgenden drei Typen unterschieden:

1) **UseFeature** – Die Attribute des betrachteten Attributtyps sollen in die Teilmenge aufgenommen werden. Wird über das Attribut eine Referenz definiert, so können die referenzierten Objekte über die Auswertung einer festzulegenden *ViewDefinition* zusätzlich in die Teilmenge aufgenommen werden. Eine Referenz auf ein anderes Objekt wird jedoch nur dann unterstützt, wenn das referenzierte Objekt über die GMSD-Beschreibung ebenfalls in die Teilmenge aufgenommen wird. Diese Einstellung entspricht konzeptionell somit der *Selektionsrelation*, die entweder mit der aktuell angewendeten oder einer neu festgelegten *ViewDefinition* verknüpft ist.
2) **DownsizeFeature** – Die Attribute des betrachteten Attributtyps sollen in die Teilmenge aufgenommen werden. Eine Referenz auf ein anderes Objekt wird jedoch nur dann unterstützt, wenn das referenzierte Objekt über die GMSD-Beschreibung ebenfalls in die

Teilmenge aufgenommen wird. Eine Erweiterung der Teilmenge über die Aufnahme referenzierter Objekte wird bei dieser Voreinstellung jedoch nicht möglich. Diese Einstellung entspricht konzeptionell somit der *Modellfilterrelation*, die mit der aktuell angewendeten oder einer neu festgelegten *ViewDefinition* verknüpft ist.

3) **RemoveFeature** – Die Attribute des betrachteten Attributtyps gehören nicht zur gewünschten Teilmenge.

EXPRESS Spezifikation:

```
TYPE FeatureUsageTypeEnum = ENUMERATION OF (UseFeature,
                                            DownsizeFeature,
                                            RemoveFeature);
END_TYPE;
```

TYPE: QUERYSETSELECT

Definition: Die Typdefinition *QuerySetSelect* wird für die Beschreibung einer Menge von Objekten verwendet. Eine solche Objektmenge kann durch eine der folgenden Möglichkeiten definiert werden:

1) **ModelID** – Eine Objektmenge wird über die Auswahl eines Planungsstandes festgelegt.
2) **SetOperation** – Die gesuchte Objektmenge wird durch eine Mengenoperation definiert.
3) **Select** – Die gesuchte Objektmenge wird über eine Objektselektion bestimmt.
4) **PartialModelQuery** – Die gesuchte Objektmenge wird aus einer GMSD-Abfrage ermittelt.

EXPRESS Spezifikation:

```
TYPE QuerySetSelect = SELECT (ModelID,
                              SetOperation,
                              Select,
                              PartialModelQuery);
END_TYPE;
```

TYPE: SELECTTYPESELECT

Definition: Die Typdefinition *SelectTypeSelect* wird für die Definition einer Selektion verwendet, die auf eine Objektmenge anzuwenden ist. Die Selektion kann durch eine der folgenden Möglichkeiten definiert werden:

1) **SelectInstancesOf** – Wird verwendet, um Objekte einer bestimmten Objektklasse zu selektieren.
2) **SelectInstances** – Wird für die Selektion von Objekten verwendet, die ausgewählte Kriterien erfüllen.
3) **SelectReferencedEntities** – Wird für die Selektion von Objekten verwendet, die ausgehend von gegebenen Objekten über ein ausgewähltes Attribut referenziert werden.

4) **SelectHasReferenceTo** – Wird für die Selektion von Objekten verwendet, die eine Referenz auf eine der gegebenen Objekte besitzen.

EXPRESS Spezifikation:

```
TYPE SelectTypeSelect = SELECT (SelectInstancesOf,
                                SelectInstances,
                                SelectReferencedEntities,
                                SelectHasReferenceTo);
END_TYPE;
```

TYPE: VIEWDEFINITIONSELECT

Definition: Die Typdefinition *ViewDefinitionSelect* wird für die Auswahl einer View-Definition verwendet. Für die Auswahl stehen die folgenden Möglichkeiten zur Verfügung:

1) **ViewDefinitionName** – Eine View-Definition wird über ihren eindeutigen Namen ausgewählt.
2) **ViewDefinition** – Eine View-Definition wird über ein Objekt des Typs *ViewDefinition* vollständig definiert.

EXPRESS Spezifikation:

```
TYPE ViewDefinitionSelect = SELECT (ViewDefinitionName,
                                    ViewDefinition);
END_TYPE;
```

ENTITY: PARTIALMODELQUERY

Definition: Die Objektklasse *PartialModelQuery* wird für die Definition eines Teildatensatzes verwendet, der über die Auswahl einer Objektmenge und einem optional anzuwendenden Filter beschrieben wird. Ein Objekt der Objektklasse *PartialModelQuery* ist der Ausgangspunkt einer vollständigen GMSD-Abfrage. Ausgehend von einem solchen Objekt können über die Referenz *SelectedObjectSet* weitere Objekte dieser Objektklasse referenziert werden. Hierdurch wird eine Baumstruktur aufgespannt, die ausgehend von dem Wurzelobjekt auszuwerten ist.

EXPRESS Spezifikation:

```
ENTITY PartialModelQuery;
  SelectedObjectSet      : QuerySetSelect;
  DefinesFeatureSubset   : OPTIONAL ViewDefinitionSelect;
END_ENTITY;
```

Attributbeschreibung:

SelectedObjectSet:	Durch diese Referenz wird eine Verknüpfung zu einem Objekt des Typs *QuerySetSelect* definiert. Über dieses Objekt wird die Auswahl von Objekten beschrieben, die durch einen optionalen Filter weiter auszuwerten sind oder, falls keine Filter definiert ist, die Objekte des gesuchten Teildatensatz beschreiben.
DefinesFeatureSubset:	Über diese optionale Referenz wird ein Filter für die über *SelectedObjectSet* ausgewählten Objekte definiert.

ENTITY: SETOPERATION

Definition: Die Objektklasse *SetOperation* wird für die Beschreibung einer Mengenoperation verwendet, die zwei auszuwählende Objektmengen miteinander verknüpft. Das Ergebnis einer Mengenoperation ist wiederum eine Objektmenge, die ebenso in einer Mengenoperation verwendet kann und dadurch die Schachtelung von Mengenoperationen erlaubt.

EXPRESS Spezifikation:

```
ENTITY SetOperation;
  SetOperation   : SetOperationTypeEnum;
  FirstOperand   : QuerySetSelect;
  SecondOperand  : QuerySetSelect;
END_ENTITY;
```

Attributbeschreibung:

SetOperation:	Definiert den Mengenoperator, der auf die über *FirstOperand* und *SecondOperand* definierten Objektmengen anzuwenden ist. Als Mengenoperation kann zwischen der Vereinigung (*union*), der Differenz (*difference*) und der Schnittmenge (*intersection*) gewählt werden.
FirstOperand:	Definiert eine Objektmenge, die als erster Operand der Mengenoperation verwendet wird. Die Reihenfolge der Operanden ist zu beachten, wenn die Differenzmenge gebildet werden soll.
SecondOperand:	Definiert eine Objektmenge, die als zweiter Operand der Mengenoperation verwendet wird. Die Reihenfolge der Operanden ist zu beachten, wenn die Differenzmenge gebildet werden soll.

ENTITY: SELECT

Definition: Die Objektklasse *Select* wird für die Selektion von Objekten verwendet, die über Objektmengen und hierauf anzuwendende Selekt-Anweisungen beschrieben wird.

EXPRESS Spezifikation:

```
ENTITY Select;
  BasedOn        : SET [1:?] OF QuerySetSelect;
  Matches        : SET [1:?] OF SelectTypeSelect;
END_ENTITY;
```

Attributbeschreibung:

BasedOn:	Definiert die Objektmengen, die durch Selekt-Anweisungen auszuwerten sind.
Matches:	Definiert die Selekt-Anweisungen, die die Auswahl der gesuchten Objekte aus den gegebenen Objektmengen beschreiben.

ENTITY: SELECTINSTANCESOF

Definition: Die Objektklasse *SelectInstancesOf* wird für die Auswahl von Objekten verwendet, die über das Unterscheidungskriterium der Klassezugehörigkeit beschrieben wird.

EXPRESS Spezifikation:

```
ENTITY SelectInstancesOf;
  EntityName        : EntityID;
  IncludeSubClasses : BOOLEAN;
END_ENTITY;
```

Attributdefinition:

EntityName:	Definiert den Namen der Objektklasse, die eine gültige Auswahl beschreibt.
IncludeSubClasses:	Über dieses Flag wird festgelegt, ob auch Objekte einer von *EntityName* abgeleitet Objektklasse das Auswahlkriterium erfüllen (TRUE) oder nicht (FALSE).

ENTITY: SELECTINSTANCES

Definition: Die Objektklasse *SelectInstances* wird für die Auswahl von Objekten verwendet, die auf der Grundlage der Objekteigenschaften oder einem eindeutigen Objektidentifikator beschrieben wird. In dieser Objektklasse sind verschiedene Möglichkeiten zur Objektauswahl zusammengefasst, die über eine Kombination verschiedener Attribute definiert wird. Alle Attribute sind über den Datentyp STRING beschrieben, der bei Bedarf in andere Datentypen zu überführen sind.

EXPRESS Spezifikation:

```
ENTITY SelectInstances;
  AttributeName      : OPTIONAL STRING;
  Value              : OPTIONAL STRING;
  Tolerance          : OPTIONAL STRING;
  LesserThan         : OPTIONAL STRING;
  MoreThan           : OPTIONAL STRING;
  Identifier         : OPTIONAL STRING;
END_ENTITY;
```

Attributbeschreibung:

AttributeName:	Über *AttributeName* wird ein Attribut bestimmt, das zur Auswahl genutzt werden soll.
Value:	Über *Value* wird ein Wert festgelegt, der mit dem über *AttributName* bestimmten Attribut verglichen wird. Sind beide Werte identisch, so wird das zugehörige Objekt in die Selektion aufgenommen.
Tolerance:	Über *Tolerance* wird eine zulässige Abweichung zwischen den zu vergleichenden Werten festgelegt.
LesserThan:	Ein Attribut erfüllt das Auswahlkriterium, wenn der entsprechende Attributwert kleiner als der in *LesserThan* angegebene Wert ist.
MoreThan:	Ein Attribut erfüllt das Auswahlkriterium, wenn der entsprechende Attributwert größer als der in *MoreThan* Angegebene Wert ist.
Identifier:	Das Auswahlkriterium wird erfüllt, wenn der Objekt-Identifikator mit dem Wert von *Identifier* übereinstimmt.

Formale Konsistenzbedingungen:

WR1:	Ein über die Attribute *LesserThan* und *MoreThan* gegebener Wertebereich soll größer als 0 sein.
WR2:	Wird über das Attribut *Value* gegeben, so ist die Angabe eines Wertebereichs nicht zulässig. Wird über die Attribute *LesserThan* und *MoreThan* ein Wertebereich gegeben, so ist die Angabe eines gesuchten Wertes nicht zulässig.
WR3:	Wird über das Attribut *Identifier* ein Objekt-Identifikator gegeben, so ist die Belegung aller anderen Attribute unzulässig.
WR4:	Sind die Attribute *Value, LesserThan* or *MoreThan* gegeben, so muss der Name des dadurch beschriebenen Attributes über *AttributeName* ebenso gegeben sein.

ENTITY: SELECTREFERENCEDENTITIES

Definition: Die Objektklasse *SelectReferencedEntities* wird für die Auswahl von Objekten verwendet, die über das angegebene Attribut referenziert werden. Die Auswahl referenzierter Objekte kann zusätzlich über die Zugehörigkeit zu einer Objektklasse eingeschränkt werden.

EXPRESS Spezifikation:

```
ENTITY SelectReferencedEntities;
  ReferenceName      : STRING;
  InstancesOf        : OPTIONAL SET [1:?] OF EntityID;
END_ENTITY;
```

Attributbeschreibung:

ReferenceName: Über *ReferenceName* wird der Name des Attributes bestimmt, über den die gesuchten Objekten referenziert werden.

InstancesOf: Dieses optionale Attribut wird zur Angabe der Objektklassen verwendet, die als zusätzliches Auswahlkriterium die zulässigen Objektklassen der gesuchten Objekte beschreiben.

ENTITY: SELECTHASREFERENCETO

Definition: Die Objektklasse *SelectHasReferenceTo* wird für die Auswahl von Objekten verwendet, die über das angegebene Attribut eine Referenz zu einem der angegebenen Objekte besitzt. Die Auswahl referenzierter Objekte kann zusätzlich über die Zugehörigkeit zu einer Objektklasse eingeschränkt werden.

EXPRESS Spezifikation:

```
ENTITY SelectHasReferenceTo;
  ReferenceName      : STRING;
  ReferenceToSet     : SET [1:?] OF QuerySetSelect;
  InstancesOf        : OPTIONAL SET [1:?] OF EntityID;
END_ENTITY;
```

Attributbeschreibung:

ReferenceName: Über *ReferenceName* wird der Name des Attributes bestimmt, das von den gesuchten Objekten für die Verknüpfung mit einem der gegebenen Objekte genutzt wird.

ReferenceToSet: Definiert Objekte, die von den gesuchten Objekten über das angegebene Attribut referenziert werden sollen. Wird eines dieser Objekte referenziert, so ist die Auswahlbedingung erfüllt.

InstancesOf: Dieses optionale Attribut wird zur Angabe der Objektklassen verwendet, die als zusätzliches Auswahlkriterium die zulässigen Objektklassen der gesuchten Objekte beschreiben.

ENTITY: VIEWDEFINITION

Definition: Die Objektklasse *ViewDefinition* wird für die Beschreibung eines Filters verwendet und entspricht somit einem *Modellfilter*. Über einen Filter werden Bedingungen definiert, die zur weiteren Auswertung einer gegebenen Objektmenge genutzt werden. Das Ergebnis eines Filters ist ebenso eine Objektmenge, die den gesuchten Informationsbedarf erfüllt.

EXPRESS Spezifikation:

```
ENTITY ViewDefinition;
  ViewDefinitionName         : STRING;
  Annotation                 : OPTIONAL STRING;
  DefinedForSchemata         : SET [1:?] OF STRING;
  PredefinedEntityUsage      : PredefinedEntityUsageTypeEnum;
  DifferingEntityDefinitions : OPTIONAL SET [1:?] OF
                                        EntityViewDefinition;
  IncludeReferences          : BOOLEAN;
  ViewDefinitionRoot         : BOOLEAN;
END_ENTITY;
```

Attributbeschreibung:

ViewDefinitionName:	Definiert den Namen einer Filterdefinition, der zur Identifikation genutzt werden kann. Ist die Definition eines Filters bekannt, so kann stellvertretend für die vollständige Definition der Name der Filterdefinition verwendet werden.
Annotation:	Dieses optionale Attribut steht für die nähere Beschreibung der Filterdefinition zur Verfügung.
DefinedForSchemata:	Definiert die Schemata, für die diese Filterdefinition anwendbar ist. Hierfür wird eine Menge von Schema-Namen verwendet.
PredefinedEntityUsage:	Legt per Voreinstellung die Auswertung von Objekten fest, für deren Objektklasse kein expliziter Objektfilter festgelegt wurde.
DifferingEntity Definitions:	Definiert eine Menge von Objektfiltern, die in Abweichung von der Voreinstellung eine expliziten Objektfilter festlegen.
IncludeReferences:	Dieser Wahrheitswert gibt an, ob alle über diesen Modellfilter zusätzlich ausgewählten Objekte über ihren Wert (*true*) oder als Referenz (*false*) in den Teildatensatz aufzunehmen sind.
ViewDefinitionRoot:	Über diesen Wahrheitswert ist erkennbar, ob der Modellfilter das Wurzelobjekt (*true*) darstellt oder Teil eines anderen Modellfilters (*false*) ist.

ENTITY: ENTITYVIEWDEFINITION

Definition: Die Objektklasse *EntityViewDefinition* wird für die Beschreibung eines *Objektfilters* verwendet. Über einen Objektfilter wird der Teil an Informationen festgelegt, der von Objekten einer bestimmten Objektklasse benötigt wird. Ein Objektfilter wird auf Objekte angewendet und erzeugt Objekte, die den gesuchten Informationsbedarf erfüllen.

EXPRESS Spezifikation:

```
ENTITY EntityViewDefinition;
  EntityName            : EntityID;
  PredefinedUsage       : PredefinedUsageTypeEnum;
  Annotation            : OPTIONAL STRING;
  GeneralizeTo          : OPTIONAL EntityID;
  InheritViewDefinition : BOOLEAN;
  DifferingFeatureUsage : OPTIONAL SET [1:?] OF FeatureSelection;
END_ENTITY;
```

Attributbeschreibung:

EntityName:	Definiert die Objektklasse, für den dieser Objektfilter anwendbar ist.
PredefinedUsage:	Legt per Voreinstellung die Auswertung von Attributen fest, für die kein expliziter Attributfilter festgelegt wurde.
Annotation:	Dieses optionale Attribut steht für die nähere Beschreibung der Filterdefinition zur Verfügung.
GeneralizeTo:	Über dieses optionale Attribut kann festgelegt werden, zu welcher Superklasse (Objektklasse aus der Vererbungshierarchie) das betrachtete Objekt verallgemeinert werden soll.
InheritViewDefinition:	Über diesen Wahrheitswert wird festgelegt, ob die Objektfilterdefinition(en) der Superklassen berücksichtigt werden soll(en) (*true*). Andernfalls (*false*) wird die vollständige Definition aus der Definition des vorliegenden Objektfilters abgeleitet. Sollen die Objektfilter der Superklassen übernommen werden und existieren für den Fall der Mehrfachvererbung widersprüchliche Definitionen, so ist für die betroffenen Attribute eine Neudefinition für der Attributfilter notwendig.
DifferingFeatureUsage:	Optional set of feature usage definitions which define an individual processing for the contained feature types. It is used to define a differing processing then defined by *PredefinedUsage*.

ENTITY: FEATURESELECTION

Definition: Die Objektklasse *FeatureSelection* wird für die Beschreibung eines *Attributfilters* verwendet. Über einen Attributfilter wird festgelegt, ob und in welcher Form die Informationen des betrachteten Attributes benötigt werden. Zusätzlich kann der Attributfilter einen Modellfilter festlegen, der für die Bewertung der referenzierten Objekte angewendet werden soll. Wird kein solcher Modellfilter angegeben, so wird hierfür der aktuell gültige Modellfilter genutzt.

EXPRESS Spezifikation:

```
ENTITY AttributeSelection;
  FeatureName                 : STRING;
  FeatureUsage                : FeatureUsageTypeEnum;
  Annotation                  : OPTIONAL STRING;
  ViewDefinitionForReferences : OPTIONAL ViewDefinitionSelect;
END_ENTITY;
```

Attributbeschreibung:

FeatureName:	Gibt den Namen des Attributes an, für den dieser Attributfilter anwendbar ist.
FeatureUsage:	Legt die Verwendung des über *FeatureName* ausgewählten Attributes fest. Hierfür kann zwischen den Typen *UseFeature, DownsizeFeature* oder *RemoveFeature* gewählt werden.
Annotation:	Dieses optionale Attribut steht für die nähere Beschreibung der Filterdefinition zur Verfügung.
ViewDefinitionFor References:	Optionale Verknüpfung mit einer Modellfilterdefinition, die auf referenzierte Objekte angewendet werden soll. Ist keine Modellfilterdefinition angegeben, so gilt auch hierfür die aktuell angewendete Modellfilterdefinition.

A.4 Abbildung der EXPRESS-Datenstruktur in JAVA-Klassen

Die prototypische Umsetzung wurde für die Verwaltung von EXPRESS-Datenstrukturen (ISO-10303-11) realisiert. Hierfür wurde der *late-binding* Ansatz gewählt, der das Einlesen und Verarbeiten von EXPRESS-Datenstrukturen zur Laufzeit erlaubt. Die Abbildung der verwalteten Daten sowie der zugrunde liegenden Datenstruktur werden über JAVA-Klassen realisiert, die die Modellierungskonzepte der EXPRESS-Spezifikation nachbilden und die benötigte Funktionalitäten bereitstellen. Alle hierfür genutzten Klassen sind in dem Package *cib.expserver.model* enthalten.

Für die Umsetzung wurde eine Klassenstruktur gewählt, die die Abbildung der Daten mit der Abbildung der Datenstruktur kombiniert. Durch diesen Ansatz wird auf eine scharfe Trennung zwischen Objekten und Objektklassen verzichtet. In Ergänzung zu Kapitel 6 ist die gewählten Klassenstruktur in der Abbildung A-5 vollständig dargestellt.

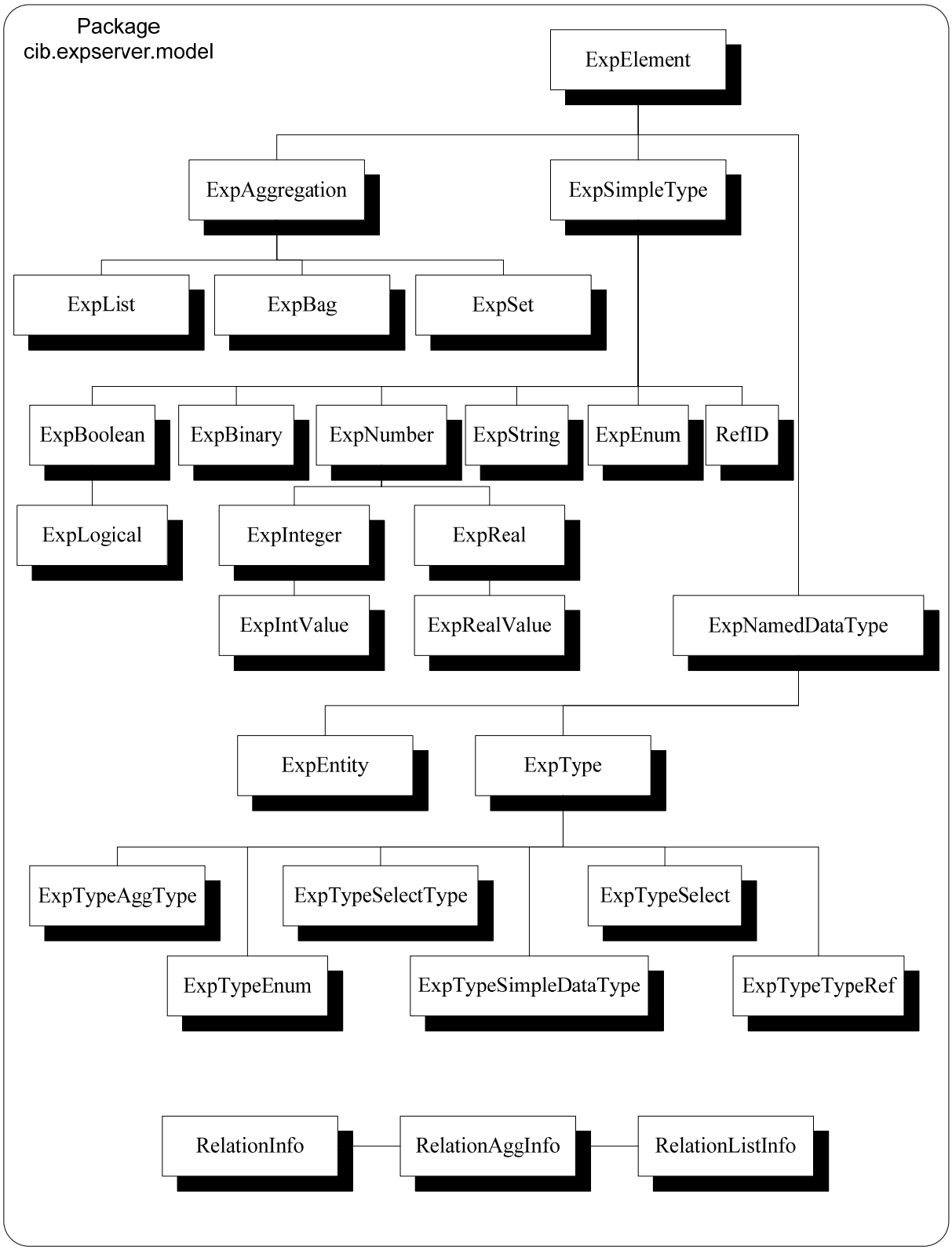


Abbildung A-5 JAVA-Klassenstruktur, die das Versionsmodell implementiert und EXPRESS-basierte Datenmodelle und zugehörige Daten verwalten kann.

Literaturverzeichnis

Adachi Y. (2002): *Overview of Partial Model Query Language*, VTT Building and Transport /SECOM Co. Ltd., Intelligent Systems Lab., VTT Report VTT-TEC-ADA-12.

Amor R. (1997): *A Generalised Framework for the Design and Construction of Integrated Design Systems*, Ph.D. Thesis, University of Auckland, New Zealand.

Amor R. & Faraj I. (2001): *Misconceptions about Integrated Project Databases.* e-journal ITcon, www.itcon.org, Vol. 6.

Apostolico A. & Galil Z. (eds.) (1997): *Pattern Matching Algorithms.* Oxford University Press.

Appleton B. (2005): *Software Configuration Management Links*, http://www.cmcross-roads.com/bradapp/links/scm-links.html

Augenbroe G. (1995): *An Overview of the COMBINE Project.* in: Scherer R. J. (ed.), Product and Process Modelling in the Building Industry, Proc. ECPPM'94, Dresden, Balkema, Rotterdam, The Netherlands.

Bazjanac V. (2002): *Early Lessons from Deployment of IFC Compatible Software*, ECPPM 2002 - eWork and eBusiness in Architecture, Engineering and Construction, Turk Z. & Scherer R.J, (ed.); A.A. Balkema.

Bazjanac V. (2005): *Model Based Cost and Energy Performance Estimation during Schematic Design.* Proceedings of the CIB-W78 Conference 2005 - Distributing Knowledge in Building, Dresden, Germany.

Beetz K., Junge R., Steinmann R. (1998): *The O.P.E.N. Platform.* Proceedings of the Second European Conference on Product and Process Modelling in the Building Industry, Watford, U.K., printed by Clowes Group, Beccles, Suffolk, England.

Bentley K. (1998): *ProjectBank, Eliminating the Chaos of Managing Engineering Project Information*, White Paper, Bentley Systems, Incorporated, October 15, 1998.

Berg Von Linde R. (2001): *Making Process Models Usable*. Royal institute of Technology, Department of Construction Management and Economics, Licentiate thesis, Stockholm, Sweden.

Berners-Lee T., Hendler J. and Lassila O. (2001): *The Semantic Web.* Scientific American, (5).

Beucke K. (1998): *Schlussbericht zum Forschungsvorhaben DBV 188 "CAE-Bauwerksmodelle"*, Fraunhofer IRB Verlag, Stuttgart.

Bijnen A. (1995): *Operation Mapping Or How to Get the Right Data?*, Scherer R. J. (ed.), „Product and Process Modelling in the Building Industry", Proceedings of the ECPPM'94, Dresden, Balkema, Rotterdam, The Netherlands.

Björk B.-C. (1995): *Requirements and information structures for building product data models*, Ph.D. Thesis, Technical Research Centre of Finland, VTT Publications 245.

Björk, B.-C. (1999): *Information Technology in Construction: Domain Definition and Research Issues*, Int. J. of Computer Integrated Design and Construction (CIDAC) 1(1), SETO, London, UK.

Borghoff U. M. & Schlichter J. H. (1998): *Rechnergestützte Gruppenarbeit*, Springer-Verlag, Berlin.

Cederqvist P. (2005): *Version Management with CVS.* for CVS 1.11.20, Free Software Foundation, Inc., http://ximbiot.com/cvs/manual/.

Chen P. (1976): *The entity-relationship model: towards a unified view of data*, ACM Transactions on Database Systems, Vol.1, ACM Press.

CIS2IFC (2004): *Harmonization for CIS/2 and IFC*, http://www.coa.gatech.edu/~aisc/cisifc/

Cobéna G., Abdessalem T., Hinnach Y.assine (2002a): *A comparative study for XML change detection*, Institut National de Recherche en Informatique et en Automatique, Rocquencourt, France.

Cobéna G., Abiteboul S., Marian, A. (2002b): *Detecting changes in XML documents*, Proceedings of 18th International Conference on Data Engineering.

Collins-Sussman B., Fitzpatrick B. W., Pilato C. M. (2004): *Version Control with Subversion*, First Edition June 2004, O'Reilly Media, http://svnbook.red-bean.com/en/1.0/svn-book.pdf.

Conradi R., Westfechtel B. (1998): *Version models for software configuration management*, ACM Computing Surveys, Vol. 30(2), ACM Press.

Date C. J. (1981): *An introduction to database systems*. 3rd ed. Addison Wesley, 1981.

Denno P. /ed/ (1999): *EXPRESS-X Language Reference Manual*, ISO/TC184/SC4/WG11/ N088, http://www.steptools.com/library/express-x/n088.pdf

Eir A. (2004): *Construction Informatics – issues in engineering, computer science, and ontology.* Ph.D. Thesis. Technische Universität Dänemark.

Eastman C., Assal H. & Jeng T.-S. (1995a): *Structure of a Product Database Supporting Model Evolution*. in Proc. CIB W78 - Workshop on Modeling of Buildings Through Their Life-cycle, Stanford University, CA.

Eastman C., Jeng T.-S., Assal H., Cho M. & Chase S. (1995b): *EDM-2 Reference Manual*. Tech. Report, Center for Design and Computation, UCLA, Los Angeles, CA.

Eastman, C. & Jeng, T-S. (1999): *A Database Supporting Evolutionary Product Model Development for Design.* Automation in Construction, 8 (3).

Eastman C. (1999): *Building Product Models: Computer Environments Supporting Design and Construction*, CRC Press, Boca Raton, Florida.

Eastman C. (2001): *Overview of CIS/2 (expanded)*, http://www.coa.gatech.edu/~aisc/document/newCIS-2overview2.pdf

Elmasri R. & Navathe S. B. (2002): *Grundlagen von Datenbanksystemen.* Pearson Studium, München.

Ekholm A. (1994): *A systemic approach to building modeling - analysis of some object-oriented building product models*. Preproceedings, CIB W78 Workshop on Computer Integrated Construction, VTT, Espoo, Finland.

Eriksson H-E. & Penker M. (2000): *Business modeling with UML: Business patterns at work*. John Wiley & Sons, New York.

Fehringer C. (2004): *Durchgehend Planen – Building Information Model*, in: AUTOCAD Magazin 06/2004, WIN-Verlag GmbH & Co. KG.

Firmenich B. (2002): *CAD im Bauplanungsprozess: Verteilte Bearbeitung einer strukturierten Menge von Objektversionen.* Dissertation, Bauhaus-Universität Weimar, Shaker Verlag, Aachen.

Firmenich B. (2004): *A novel modelling approach for the exchange of CAD information in civil engineering*, ECPPM 2004 - eWork and eBusiness in Architecture, Engineering and Construction, Dikbas A. & Scherer R. J. (eds.), A.A. Balkema Publishers, Leiden, The Netherlands.

Firmenich B., Koch C., Richter T., Beer D. (2005): *Versioning structured object sets using text based Version Control.* Proceedings of the CIB-W78 Conference 2005 - Distributing Knowledge in Building, Dresden, Germany.

Fischer M. & Froese T. (1992): *Integration through standard project models*. Vanier D, Russell T (ed.), Computers and information in construction; Montreal, May, Canada.

Froese T. (1992): *Integrated Computer-Aided Project Management Through Standard Object-Oriented Models.* Ph.D. Thesis, Dept. of Civil Engineering, Stanford University.

Froese T. (1996): *STEP Data Standards And The Construction Industry.* Proceedings of the 1996 Annual Conference of the Canadian Society for Civil Engineering (Vol. 1), Edmonton, Alberta, CSCE, Montreal, Canada.

Froese T., Rankin J. & Yu K. (1997): *Project Management Application Models And Computer-Assisted Construction Planning In Total Project Systems.* International Journal of Construction Information Technology, Special Issues on Integrated Environments, Vol. 5, No. 1.

Gellert W., Küstner H., Hellwich M, Kästner H. (1986): *Kleine Enzyklopädie Mathematik*, Bibliographisches Institut Leipzig.

Genesereth M. & Fikes R. (1992): *Knowledge Interchange Format 3.0*, Reference Manual, Technical Report Logic-92-1, Computer Science Dept., Stanford University, CA.

Gielingh W. (1988): *General AEC reference model (GARM) an aid for the integration of application specific product definition models*. Christiansson P, Karlsson H (ed.), Conceptual modelling of buildings. CIB W74+W78 seminar, Lund university and the Swedish building centre.

Gielingh W. (2005): *Rethinking Conceptual Structures and their Expression – Part 1: An Essay about Concept Formation and Symbology.* Proceedings of the CIB-W78 Conference 2005 - Distributing Knowledge in Building, Dresden, Germany.

Gomez-Perez A. & Corcho O. (2002): *Ontology languages for the semantic web*. IEEE Intelligent Systems, January/February.

Goncalves R. J. (2004): *A Framework for Multilevel Standard Protocols and Interoperability*, UNINOVA, Ph.D. Thesis, Lissabon, Portugal.

Greb S. & Klauer T. (2004): *Prozessmodellierung und Workflow-Management im Bauwesen.* Zimmermann J., Geller S. (Hrsg.): Forum Bauinformatik 2004 - Junge Wissenschaftler Forschen, Shaker Verlag, Braunschweig.

Haas W. (1998): *The Exchange of 3D CAD Building Models Using STEP AP225 – Scope, Functionality and Industrial Practice,* Proc. of the 2nd European Conference on Product and Process Modelling in the Building Industry, Amor R., Scherer R.J. (ed.), Building Research Establishment, Watford, Great Britain.

Haas W. (2000): *STEP and its application in the construction industry,* Proc. of the 3rd European Conference on Product and Process Modelling in the Building Industry, A.A. Balkema, Rotterdam, 2000.

Hakim M. (1993): *Modeling Evolving Information About Engineering Design Products: An Object-Centered Approach Combining Description Logic and Object-Oriented Modeling*, Ph.D. Thesis, Carnegie Mellon University, Pittsburgh.

Hanff J. (2003): *Abhängigkeiten zwischen Objekten in ingenieurwissenschaftlichen Anwendungen.* Dissertation, TU-Berlin.

Hannus M., Karstila K. & Tarandi V. (1995): *Requirements on standardised building product data models.* in: Scherer R. J. (ed.), Product and Process Modelling in the Building Industry, Proc. ECPPM'94, Dresden, Balkema, Rotterdam, The Netherlands.

Härder T., Reuter A. (1983): *Principles of transaction-oriented database recovery*. ACM Computing Survey.

van Harmelen F., Patel-Schneider P. F. & Horrocks, I. (2001): *Reference description of the DAML+OIL ontology markup language*, http://daml.org/2001/03/reference.html.

Hartmann D. (ed.) (2000): *Objektorientierte Modellierung in Planung und Konstruktion, Forschungsberichte*, Wiley-VCH Verlag, Weinheim.

Hauschild T. (2003): *Computer Supported Cooperative Work - Applikationen in der Bauwerksplanung auf der Basis einer integrierten Bauwerksmodellverwaltung*, Dissertation, Bauhaus-Universität Weimar.

Haymaker J., Ackermann E. & Fischer M. (2000): *Meaning Mediating Mechanism: A prototype for constructing and negotiating meaning in collaborative design*, 6th International Conference on Artificial Intelligence in Design; Kluwer Academic Publishers, Dordrecht, The Netherlands.

Haymaker J., Suter B., Fischer M., Kunz J. (2004): *The Perspective Approach: Enabling Engineers to Construct and Integrate Geometric Views and Generate an Evolving Project Model,* Working Paper Nr 081, Center For Integrated Facility Engineering, Stanford University.

Haymaker J. (2005): *Formalizing and Managing the Dependencies between Models.* Proceedings of the CIB-W78 Conference 2005 - Distributing Knowledge in Building, Dresden, Germany.

Herrmann U. (1991): *Mehrbenutzerkontrolle in Nicht-Standard-Datenbanksystemen*, Informatik-Fachberichte 256, Springer-Verlag, Berlin-Heidelberg, Germany.

ISO 10303-1 IS (1994): *Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 1: Overview and Fundamental Principles*, International Organisation for Standardisation, ISO TC 184/SC4, Geneva.

ISO 10303-11 IS (1994) /Cor.1:1999/: *Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 11: Description Methods: The EXPRESS Language Reference Manual*, International Organisation for Standardisation, ISO TC 184/SC4, Geneva.

ISO 10303-21 IS (1994) /Cor.1:1999/: *Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 21: Implementation Methods: Clear Text Encoding of the Exchange Structure*, International Organisation for Standardisation, ISO TC 184/SC4, Geneva.

ISO 10303-22 IS (1998): *Industrial Automation Systems and Integration -- Product Data Representation and Exchange -- Part 22: Implementation Methods: Standard Data Access Interface*, International Organisation for Standardisation, ISO TC 184/SC4, Geneva.

ISO/IEC 14750 (1999): *Information technology - Open Distributed Processing - Interface Definition Language*, International Organisation for Standardisation, ISO/IEC/JTC 1/SC 7.

ISO 12006-3 (2001): *Organization of information about construction works*, International Organisation for Standardisation, BS ISO/PAS 12006-3

Junge R., Beetz K. & Liebich T. (1997): *Product data model and implementation strategies for a distributed environment*. Drogemuller R. (editor), International council for building research studies and documentation. Workshop, Cairns, Queensland, Australia.

Junge R. & Liebich T. (1998): *Product Modelling Technology – the Foundation of Industry Foundation Classes.* Proceedings of the Second European Conference on Product and Process Modelling in the Building Industry, Watford, U.K., printed by Clowes Group, Beccles, Suffolk, England.

Juli R. & Scherer R. J. (2002): *iCSS - Ein integriertes Client-Server-System für das virtuelle Planungsteam.* VDI-Berichte 1668. VDI Verlag, Düsseldorf.

Karstila K. (2001): *ST-1 Steel Frame Constructions*, http://cic.vtt.fi/iai/IFCModelDevelopment/sheets/R3_ST1_A4.pdf.

Karstila K. (2004): *General overview of exchange scenarios*. Groove-Workspace IAI Structural, IFC structural scenarios.xls.

Katranuschkov P. (1995): *COMBI: Integrated Product Model.* in: Scherer R. J. (ed.), Product and Process Modelling in the Building Industry, Proc. ECPPM'94, Dresden, Balkema, Rotterdam, The Netherlands.

Katranuschkov P. & Hyvärinen J. (1998): *Product Data Server for Concurrent Engineering in A/E/C*, Proc. of the 2nd European Conference on Product and Process Modelling in the Building Industry, Amor R., Scherer R.J. (ed.), Building Research Establishment, Watford, Great Britain.

Katranuschkov P. (2001): *Eine Interoperabilitätsmethode zur Unterstützung von Concurrent-Engineering-Prozessen.* Dissertation, Technische Universität Dresden.

Katranuschkov P., Gehre A. und Scherer R. J. (2003): *An ontology framework to access IFC model data*, e-journal ITcon, www.itcon.org, Volume 8, Special Issue eWork and eBusiness.

Katranuschkov P., Gehre A., Scherer R.J., Wix J. und Liebich T. (2004): *User Requirements Capture in Distributed Project Environments: A Process-Centred Approach.* Proceedings of the ICCCBE-X Conferenz, Weimar.

Katz R. H. (1990): *Towards a Unified Framework for Version Modeling in Engineering Databases.* ACM Computing Surveys, Volume 22 (4), ACM Press.

Keller M., Menzel K., Schäffler H., Entzian K., Steinmann R., Streller K., Illieva D. (2004): *State of the Art kollaborativer Szenarien in der Baubranche.* Projektbericht, Arkos – AP1.4: Analyse der Anforderungen und des Umfeldes.

Koivu T.J. (2001): *Future of Product Modeling and Knowledge Sharing in the FM/AEC Industry.* e-journal ITcon, www.itcon.org, Vol.2.

König M., Klinger A., Berkhahn V. (2004): *Process modelling in building engineering*, ECPPM 2004 - eWork and eBusiness in Architecture, Engineering and Construction, Dikbas A. & Scherer R. J. (eds.), A.A. Balkema Publishers, Leiden, The Netherlands.

Koutamanis A. (1990): *Development of a computerized handbook of architectural plans*, Ph.D. Thesis, Delft University of Technology.

Kraft B. & Wilhelms N. (2005): *Visual Knowledge Specification for Conceptual Design.* Proc. of the 2005 ASCE International Conference on Computing in Civil Engineering, Cancun.

Krapp C.-A., Krüppel S., Schleicher A., Westfechtel B. (1998): *Graph-Based Models for Managing Development Processes, Resources, and Products*. Lecture Notes In Computer Science; Vol. 1764, Selected papers from the 6th International Workshop on Theory and Application of Graph Transformations, Ehrig H., Engels G., Kresowski H.-J., Rozenberg G. (eds.), Springer-Verlag, London, UK.

Luiten B., Luijten B., Willems P., Kuiper P. & Tolman, F.P., (1991): *Development and Implementation of Multilayered Project Models.* Mangin, J-C., Kohler, N. and Brau, J. (ed.), Computer Building Representation for Integration, Pre-Proceedings of the second international workshop, Ecole depolytechnique federale de Lausanne

Luiten B., Froese T. M., Björk B.-C., Cooper G., Junge R., Karstila K., Oxman R. (1993): *An Information Reference Model For Architecture, Engineering, And Construction.* Proceedings of the First International Conference on the Management of Information Technology for Construction, K. Mathur, M. Betts, and K. Tham, Eds. World Scientific, Singapore.

Lockemann P. C., Schmidt J. W. (Hrsg.) (1987): *Datenbank-Handbuch.* Springer-Verlag, Berlin-Heidelberg, 1987.

Lockley S.R., Rombouts W.Th. & Plokker W. (1995): *The COMBINE Data Exchange System.* in: Scherer R. J. (ed.), Product and Process Modelling in the Building Industry, Proc. ECPPM'94, Dresden, Balkema, Rotterdam, The Netherlands.

Lockley S.R. & Augenbroe G. (2000): *Data Integration with Partial Exchange*, Proc. of International Conference on Construction Information Technology, INCITE 2000, Hong Kong.

MacKellar B. & Peckam J. (1998): *Multiple Perspectives of Design Objects.* Artificial Intelligence in Design, eds. John Gero and Fay Sudweeks, Klewer Academic Publishers.

Mayer R. J., Painter M. K. & deWitte P. S. (1992): *IDEF Family of Methods for Concurrent Engineering and Business Re-engineering Applications.* Knowledge Based Systems Inc. http://www.idef.com/pdf/IDEFFAMI.pdf

Meißner U. F. (2005): *DFG-Schwerpunktprogramm 1103 – Vernetzt kooperative Planungsprozesse im Konstruktiven Ingenieurbau.* http://www.dfg-spp1103.de/

Meißner U.F., Rüppel U. (2003): *Vernetzt-kooperative Planung mit Computern - Grundlagen und Methoden der Bauinformatik.* Gürlebeck K., Hempel L. & Könke C. (Hrsg.), digital Proceedings, 16. Internationales Kolloquium über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen (ikm 2003), Weimar.

Munch B. P. (1993): *Versioning in a Software Engineering Database - the Change Oriented Way*, Ph.D. Thesis, The Norwegian Institute of Technology.

Nagel R. N., Braithwaite, W. W., Kennicott P. R. (1980): *Initial Graphics Exchange Specification IGES*, Version 1.0, NBSIR 80-1978, National Bureau of Standards, Washington, DC.

van Nederveen S., Tolman F. (2001): *Neutral Object Tree Support For Inter-Discipline Communication In Large-Scale Construction*, e-journal ITcon, www.itcon.org, Vol. 6.

Neiling M. (2004): *Identifizierung von Realwelt-Objekten in multiplen Datenbanken.* Dissertation, BTU Cottbus, 2004.

Neufert E. (1950): *Bauentwurfslehre*, Bauwelt Verlag, 15. Auflage, Berlin.

Ogden C. K., Richards, I. A. (1923): *The Meaning of Meaning*, Harcourt Brace Jovanovich, New York.

OKSTRA (1999): *Objektkatalog für das Straßen- und Verkehrswesen: Historisierung*, Dokument T0002, http://www.okstra.de/docs/t0002.pdf

Olbrich M. (1998): *Relationenorientiertes Modellieren mit Objekten in der Bauinformatik,* Dissertation, Universität Hannover.

Pahl P. J., Damrath R. (2000): *Mathematische Grundlagen der Ingenieurinformatik*, Springer-Verlag, Berlin.

Pan J., Anumba C. J. (2005): *A Framework for Project Information Management in A Semantic Web Environment.* Proceedings of the 2005 ASCE International Conference on Computing in Civil Engineering, Cancun.

Persson-Dahlqvist A., Asklund U., Crnkovic I., Larsson M., Svesson D. (2002): *Product Data Management and Software Configuration Management - Similarities and Differences*, MRTC report ISSN 1404-3041 ISRN MDH-MRTC-54/2002-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, Sweden.

Reed K. A. (1988): *Product modeling of buildings for data exchange standards: from IGES to PDES/STEP and beyond*, Christiansson P, Karlsson H (ed.), Conceptual modelling of buildings. CIB W74+W78 seminar, Lund university and the Swedish building centre.

van Rees R, Tolman F. & Beheshti R. (2002): *How BcXML Handles Construction Semantics*, Proceedings of the CIB-W78 Conference 2002 - Distributing Knowledge in Building, Aarhus, Denmark.

van Rees, R. (2003): *Clarity in the Usage of the Terms Ontology, Taxonomy and Classification.* Amor R (ed.) Proceedings of the CIB W78's 20th International Conference on Construction IT, CIB Report 284, Waiheke Island, New Zealand.

Reimar U. (1991): *Einführung in die Wissensrepräsentation: netzartige und schema-basierte Repräsentationsformate*, Teubner, Stuttgart.

Rochkind M. J. (1975): *The Source Code Control System.* IEEE Transactions on Software Engineering, Dec. 1975, Volume SE-1, Number 4.

Rudolph D., Stürznickel T., Weissenberger L. (1993): *Der DXF-Standard*, Rossipaul Verlag, München.

Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W. (1991): *Object-Oriented Modeling and Design*, Prentice Hall, New York.

Rumbaugh J., Jacobson I. & Booch G. (2004): *The Unified Modeling Language Reference Manual*. 2nd edition, Addison-Wesley.

Rüppel U. (1998): *Ganzheitliche Projektbearbeitung auf der Basis dokumentenorientierter Workflows.* Tagungsband der Fachtagung "Finite Elemente in der Baupraxis - FEM '98", Verlag Ernst&Sohn, Berlin.

Scheer A.-W., Adam O., Hofer A., Zang S. (2004): *ArKoS - Architektur Kollaborativer Szenarien.* Forschungsoffensive "Software Engineering 2006", Eröffnungskonferenz 01.-03. Juli 2004, Berlin, http://www.bitkom.org/files/documents/06_Beitrag_ARKOS.pdf.

Scherer R. J. (1995): *The EU Project COMBI - Objectives and Overview*. in: Scherer R. J. (ed.) Product and Process Modelling in the Building Industry, Proc. ECPPM'94, Dresden, Balkema, Rotterdam, The Netherlands.

Scherer R. J. (2000): *Client-Server System for Concurrent Engineering - ToCEE Final Report*. EU/CEC ESPRIT Project 20587, Technische Universität Dresden.

Scherer R. J., Weise M., Katranuschkov P. (2004): *A Cooperation Model for the Control of Diverging Design Data*, German-Russian Symposium on Construction Informatics, Moskau/St. Petersburg.

Scherer R. J., Eisfeld M. (2005): *A Situated Planning Assistant for Conceptual Design of Building Structures*, Proceedings of the 2005 ASCE International Conference on Computing in Civil Engineering, Cancun, Mexico.

Schöning U. (2001): *Algorithmik*, Spektrum, Akademischer Verlag, Heidelberg, Berlin.

Schroeder U. (1995): *Inkrementelle, syntaxbasierte Revisions- und Variantenkontrolle mit interaktiver Konfigurationsunterstützung.* Dissertation, Aachen: Shaker-Verlag.

Schwarze J. (2001): *Projektmanagement mit Netzplantechnik.* 8. Auflage. Reihe: NWB-Studienbücher Wirtschaftswissenschaften, Verlag Neue Wirtschafts-Briefe, Herne/Berlin.

Sciore E. (1994): *Versioning and Configuration Management in an Object-Oriented Data Model.* VLDB Journal, 3(1), Jan. 1994.

Sheth A. P. & Larson J. A. (1990): *Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases*. ACM Computing Surveys, Vol. 22, No. 3.

Smolka G. (1992): *Feature-constrained logics for unification grammars.* Journal of Logic Programming, Vol. 12.

Spinner A. (1989): *Ein Lernsystem zur Erzeugung komplexer Kommandos in Programmierumgebungen*, Dissertation, TH Darmstadt.

Sriram D. (2002): *Distributed and Integrated Collaborative Engineering Design*. Sarven Publishers.

Steinmann R. (2005): *Web-site of IAI's ISG, the Implementer Support Group*, http://www.bauwesen.fh-muenchen.de/iai/iai_isg/.

Stouffs R. & Krishnamurti R. (2001): *Standardization: a critical view*, Proceedings of the CIB-W78 International Conference: IT in Construction in Africa, Pretoria, South Afrika.

Straub H. (1975): *Die Geschichte der Bauingenieurkunst*, Birkhäuser Verlag, 3. Auflage, Basel.

Tarandi V. (1998): *Neutral Intelligent CAD Communication - Information exchange in construction based on a minimal schema*, Ph.D. Thesis, Royal Institute of Technology, Stockholm, Sweden.

Tichy W. F. (1985): *RCS--A System for Version Control,* Software Practice & Experience, Volume 15, Issue 7, July 1985, John Wiley & Sons, New York, USA.

Tolman F. & Poyet P. (1995): *The ATLAS Models.* in: Scherer R. J. (ed.) Product and Process Modelling in the Building Industry, Proc. ECPPM'94, Dresden, Balkema, Rotterdam, The Netherlands.

Turk Z. (2001): *Phenomenological foundations of conceptual product modelling in architecture, engineering and construction.* Artificial Intelligence in Engineering. Volume 15.

Unland R. (1994): *Optimistic Concurrency Control Revisited.* Arbeitsbericht Nr 31, Becker J., Grob H. L., Kurbel K., Müller-Funk U., Unland R., Vossen G. (Hrsg.), Institut für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster.

Vossen G. (1994): *Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme.* 2. Auflage, Addison-Wesley, Bonn/Paris.

Vossen G. & Becker J. (1996): *Geschäftsprozessmodellierung und Workflow-Management: Modelle, Methoden, Werkzeuge*, Bonn: International Thomson Publishing.

W3C (2004): OWL Web Ontology Language – Overview, http://www.w3.org/TR/owl-features.

Wagner R. A. & Fischer M. J. (1974): *The String-to-String Correction Problem*, Journal of the ACM, Volume 21 , Issue 1 ACM Press New York, USA.

Wasserfuhr R. & Scherer R.J (1999): *Towards Role Based Management of Cooperative Design Processes*, 2. International Conference on Concurrent Engineering in Construction, Helsinki, August 1999.

Watson A. & Ward M. (1996): *ISO/WD 10303-230, Building Structural Frame: Steelwork*, ISO TC184/SC4/WG3 N551 (T12).

Wedekind H., Görz G., Kötter R., Inhetveen R. (1998): *Modellierung, Simulation, Visualisierung: Zu aktuellen Aufgaben der Informatik*, Informatik-Spektrum, Nr.21, Springer-Verlag.

Weise M. & Katranuschkov P. (2001): *Eine praxisorientierte Sicht auf die Produktmodelltechnologie - Möglichkeiten und Probleme bei der Anwendung der IFC*, Forum Bauinformatik 2001, München, VDI Verlag, Düsseldorf.

Weise M., Katranuschkov P., Liebich T. (2002): *Das iCSS-Tragwerksmodell und seine Einbindung in das IFC-Modell*, VDI-Berichte 1668. VDI Verlag, Düsseldorf.

Weise M., Katranuschkov P., Liebich T., Scherer R. J. (2003a): *Structural Analysis Extension of the IFC Modelling Framework*, e-journal ITcon, www.itcon.org, Volume 8.

Weise M., Katranuschkov P., Scherer R. J. (2003b): *Generalised Model Subset Definition Schema.* Proceedings of the CIB-W78 Conference – Information Technology for Construction, Auckland, New Zealand.

Weise M., Katranuschkov P., Scherer, R. J. (2004a): *Managing Long Transactions in Model Server Based Collaboration*, ECPPM 2004 - eWork and eBusiness in Architecture, Engineering and Construction, Dikbas A. & Scherer R. J. (eds.), A.A. Balkema Publishers, Leiden, The Netherlands.

Weise M., Katranuschkov P., Scherer R. J. (2004b): *Generic Services for the Support of Evolving Building Model Data.* Proceedings of the ICCCBE-X Conferenz, Weimar.

Weise M. & Katranuschkov P. (2005): *Supporting State-based Transactions in Collaborative Product Modelling Environments.* Proceedings of the CIB-W78 Conference 2005 - Distributing Knowledge in Building, Dresden.

Westfechtel B. (1991): *Structure-Oriented Merging of Revisions of Software Documents*, Proceedings of 3rd International Workshop on Software Configuration Management, ACM Press, New York.

Westfechtel B. (1999): *Graph-Based Product and Process Management in Mechanical Engineering*, in: H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg (Eds.): Handbook of Graph Grammars and Computing by Graph Transformation, Vol. II.

Westfechtel B., Conradi R. (1998): *Software Configuration Management and Engineering Data Management: Differences and Similarities*. Proceedings 8th International Workshop on System Configuration Management, Brüssel, LNCS 1439, Springer-Verlag.

Willenbacher H. (2002): *Interaktive verknüpfungsbasierte Bauwerksmodellierung als Integrationsplattform für den Bauwerkslebenszyklus*, Dissertation, Bauhaus-Universität Weimar.

Wix J. (1996): *Purpose of a Core Model*, Tech. Report, Research Project „Computerised Exchange of Information in Construction“, Dept. of the Environment, UK.

Wix J. & See R. (1999): *IFC Specification Development Guide*, International Alliance for Interoperability (IAI), PDF file.

Wix J. & Katranuschkov P. (2002): *Defining the Matrix of Communication Processes in the AEC/FM Industry: Current Developments and Gap Analysis*, Proceedings of the CIB-W78 Conference 2002 - Distributing Knowledge in Building, Aarhus, Denmark.

Wix J. & Liebich T. (2000): *Information Flow Scenario*, Deliverable D4, EU Project, IST-1999-11508 (ISTforCE).

Wix, J. & Liebich, T. (2003): *Industry Foundation Classes IFC 2x2,* © International Alliance for Interoperability, http://www.iai-ev.de/spezifikation/IFC2x2/index.htm.

Woestenenk K. (2002): *The LexiCon: Sructuring Semantics*, Proceedings of the CIB-W78 Conference 2002 - Distributing Knowledge in Building, Aarhus, Denmark.

Yum K.-K. & Drogemuller R. (1997): *Managing dynamic life-cycle-dependent building objects in a distributed computing environment*, Drogemuller R. (ed.), Intern. council for building research studies and documentation. Workshop, Cairns, Queensland, Australia.

Zarli A. (1995): *XP-Rule: A Language for the Representation of Knowledge*, Technical Report, CSTB, Sophia Antipolis, France.

Zeller A. (1997): *Configuration Management with Version Sets – A Unified Software Versioning Model and its Application*. Dissertation. Technische Universität Braunschweig.

Zeller A. & Snelting G. (1997): *Unified Versioning through Feature Logic*. ACM Transactions on Software Engineering and Methodology (TOSEM), Volume 6, Issue 4.

Zhang K., Statman R., Shasha D. (1992): *On the editing distance between unordered labeled trees*, Information Proceedings Letters 42.

Zuser W., Grechenig T., Köhle M. (2004): *Software Engineering mit UML und dem Unified Process*, 2. überarbeitete Auflage, Pearson Studium.

Bisher erschienene Dissertationen, Habilitationen und Hefte des Instituts für Bauinformatik[1]

Markus Hauser	Eine kognitive Architektur für die wissensbasierte Unterstützung der frühen Phasen des Entwurfs von Tragwerken	logos Verlag Berlin, 06/1998
Martin Zsohar	Stochastische Größen der Resonanzfrequenzen und der Verstärkung seismischer Wellen im horizontal geschichteten zufälligem Medium	Shaker Verlag Aachen, 07/1998
Christian Steurer	Modellierung und Berechnung des Ermüdungsfortschritts mit stochastischen Differentialgleichungen	Selbstverlag, 02/1999
Peter Katranuschkov	A Mapping Language for Concurrent Engineering Processes	Heft 1, 02/2001
Karsten Menzel	Methodik zur nachhaltigen, rechnergestützten Ressourcenverwaltung im Bauwesen	Heft 2, 11/2003
Michael Eisfeld	Assistance in Conceptual Design of Concrete Structures by a Description Logic Planner	Heft 3, 11/2004

[1] Bis September 2003 Lehrstuhl für Computeranwendung im Bauwesen