KOPPLUNG VON FACHAPPLIKATIONEN MIT VERTEILTEN, DYNAMISCHEN MODELLVERWALTUNGSSYSTEMEN

André Borrmann, Thomas Hauschild, Reinhard Hübler
Bauhaus-Universität Weimar, Informations- und Wissensverarbeitung

Kurzfassung: *Digitale Bauwerksmodelle sind eine geeignete Basis für die rechnergestützte integrale Planung von Bauwerken. Um den verschiedene Sichten der beteiligten Fachdisziplinen auf den Planungsgegenstand gerecht zu werden und die Komplexität des Modells auf ein handhabbares Maß zu reduzieren, ist eine Partitionierung in Domänenmodelle notwendig, die sich auch in der hybriden Topologie des konzipierten Informationssystems wiederspiegelt. Die wegen der Individualität der Planungsaufgaben und a-priori nicht bekannten Anforderungen notwendige Dynamik der Domänenmodelle kann durch den Einsatz dynamischer Modellverwaltungssysteme (MVS) realisiert werden, deren Clients mit Hilfe einer generischen Schnittstelle auf die Planungsinformationen zugreifen. Zur Integration konventioneller Fachapplikationen mit statischem Datenmodell dienen Konnektoren, die die generische Schnittstelle auf ein gegenständliches Niveau adaptieren. Für die im Falle des Online-Zugriffs notwendige ständige Synchronisation zwischen dem MVS-Server und seinen Clients wurde ein Notifikationsmodul entwickelt, dessen Implementierung gezeigt wird. Abschließend wird die Möglichkeit der Abbildung und Prüfung von Konsistenzbedingungen im MVS mit Hilfe von Constraints erörtert.*

0. Einleitung

Der Bauplanungsprozess ist durch eine komplexe Aufgabenstellung und eine hohe Zahl von Beteiligten aus unterschiedlichen Fachdisziplinen gekennzeichnet. Die große Menge gegenseitiger Abhängigkeiten zwischen den Planungsleistungen zwingt die Beteiligten zu einer intensiven Kooperation.

Ziel der Integrationsbemühungen ist die Unterstützung dieser Kooperation durch die Gewährleistung eines reibungslosen Datenaustauschs. Der hier verfolgte Ansatz zur

Integration der bereits gut durch den Einsatz von Rechnern unterstützten Teilprozesse sieht die Verwaltung aller anfallenden Planungsdaten eines Vorhabens in einem gemeinsamen, verteilten Informations- und Kommunikationssystem vor.

Ziel ist es, durch den Einsatz geeigneter informationstechnischer Mittel ein hohes Maß an Parallelität von Planungsaktivitäten bei gleichzeitiger Gewährleistung der Konsistenz der Planungsdaten zu erreichen. Während die Parallelisierung zu einer potentiellen Verkürzung der Planungszeiträume führt, können durch effektive Konsistenzsicherungsmaßnahmen auf semantisch hohem Niveau Planungsfehler und damit verbundene Folgekosten vermieden werden.

Um eine erneute manuelle Eingabe bereits digital verwalteter Daten zu vermeiden, muss das System zur Datenverwaltung von vornherein alle Phasen des Lebenszyklus eines Bauwerks unterstützen, angefangen bei der Planung, über die Errichtung und die Nutzungsphase bis hin zur Umnutzung bzw. dem Rückbau.

1. Dynamische Bauwerksmodelle

Die rechnerinterne Verwaltung von Planungsdaten kann mit Hilfe eines digitalen Bauwerkmodells realisiert werden. Bei der Formulierung eines solchen Modells ist eine Vorgehensweise nach Prinzipien der objektorientierten Analyse sinnvoll, deren Ergebnis ein objektorientiertes Datenmodell ist. Die Ausprägung eines solchen Modells durch Instanziierung beinhaltet die Daten eines konkreten bzw. zu planenden Bauwerks.

Die im Bauplanungsprozess involvierten Disziplinen besitzen unterschiedliche Sichten auf den gemeinsamen Planungsgegenstand, was sich in der Verwendung fachspezifischer Taxonomien wiederspiegelt. Diese verfügen zwar über eine gemeinsame Basismenge von Begriffen mit identischer Bedeutung, darüber hinaus beinhalten sie aber auch Begriffe, die nur in der jeweiligen Domäne Bedeutung haben oder in verschiedenen Disziplinen unterschiedlich interpretiert werden.

Um den verschiedenen Sichten der beteiligten Fachplanerbereiche gerecht zu werden und gleichzeitig die Komplexität des Datenmodells zu begrenzen, hat sich die Verwendung von Partialmodellen durchgesetzt. Sie bilden die Taxonomie einer einzelnen Domäne ab und werden daher auch Domänenmodelle genannt. Die Kohärenz der Partialmodelle untereinander wird mit Hilfe von domänenübergreifenden Verknüpfungen gewährleistet. [1]

Zwei Aspekte des Erzeugens und der Verwendung von Planungsdaten legen die Notwendigkeit einer Dynamisierung der verwendeten Domänenmodelle nahe: Zum einen impliziert der Unikatcharakter der zu errichtenden oder zu revitalisierenden Bauwerke, dass ein für alle Bauvorhaben gültiges Produktmodell nicht zu identifizieren ist.

Bei der initialen Konfiguration des IuK-Systems muss daher eine Anpassung der verwendeten Datenstrukturen an die besonderen Spezifika des konkreten Bauprojektes

stattfinden, bei der im allgemeinen standardisierte bzw. für bestimmte Bauwerkstypen vordefinierte Produktmodelle als Basis genutzt und ggf. um spezielle Entitäten erweitert werden.

Zum anderen resultieren aus dem langen Lebenszyklus des Bauwerks nicht vorhersehbare Anforderungen an die Struktur der zu verwaltenden Daten: Das System muss es erlauben, dass bereits vorhandenen Klassen des Domänenmodells zur Laufzeit Attribute hinzugefügt und Subklassen von ihnen gebildet werden können.

Für derartige Modifikationen am Datenmodell ist ein Zugriff auf die Meta- bzw. Modellierebene notwendig. Folgerichtig muss der Zugriff auf die Instanzdaten auf eine generische Art und Weise erfolgen, da Bezeichner von Klassen bzw. Attributen a priori nicht bekannt sind. Ebenso müssen für eine korrekte Darstellung der Instanzen dynamisch erzeugter bzw. modifizierter Klassen die ins System integrierten generischen Fachanwendungen auf die Metaebene zugreifen, um die vorhandenen Daten interpretieren zu können. [2]

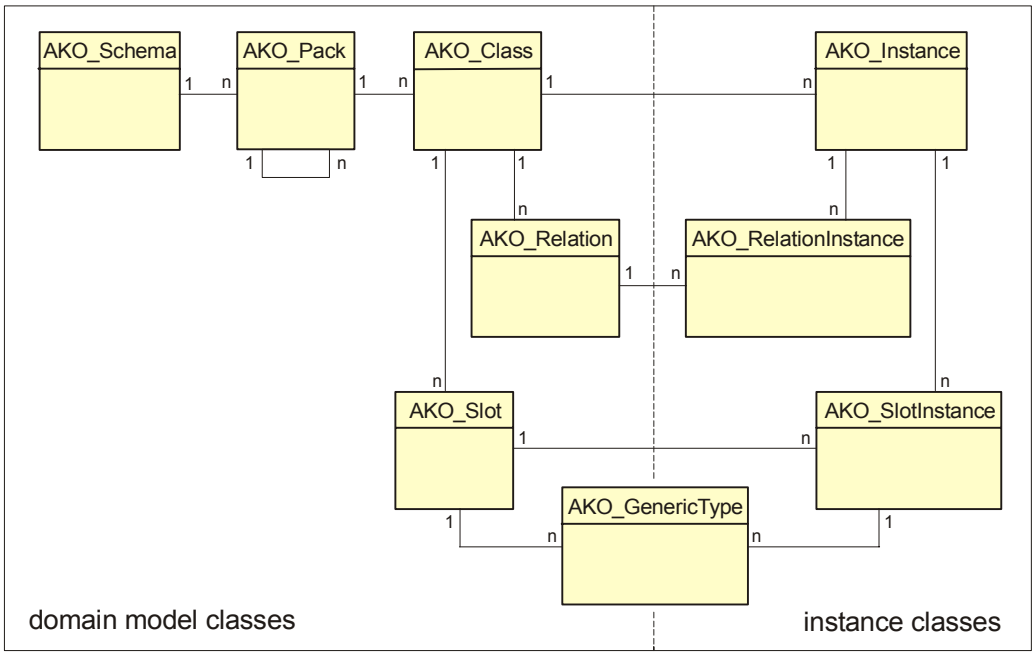

Abb. 1: Das AKO-Metamodell

Technisch wurden diese Anforderungen durch den Einsatz eines dynamischen Modellverwaltungskerns umgesetzt, der auf der Ebene des AKO-Metamodells operiert. (Abb.1) Der Zugriff auf das MVS geschieht mit Hilfe einer diesem Metamodell entsprechenden API, der sogenannten AKO-Schnittstelle. [3]

Zur Verwaltung von Domänenmodellen ist neben der Nutzung des AKO-Metamodells auch die Verwendung des von der Object Management Group (OMG) in den *Meta*

Object Facilities (MOF) spezifizierten Metamodells denkbar. [4] Dieser Industrie-Standard ist potentiell weit verbreitet und lässt die baldige Existenz ausgereifter Softwareprodukte erhoffen. So könnten beispielsweise MOF-Repositories als Modellverwaltungsserver eingesetzt werden und MOF-Browser zur Navigation in Domänenmodellen dienen.

2. Hybride Topologie des Gesamtsystems

Ein Planungsteam ist zumeist in mehrere kleine Einheiten (Planungsbüros) strukturiert, deren Mitglieder identische Aufgaben und Ziele innerhalb des Gesamtprojektes besitzen und die gleiche Sicht auf den Planungsgegenstand haben. Während die Beteiligten eines solchen Fachplanerbereiches im allgemeinen keiner räumlichen Trennung unterworfen sind, sind die geographischen Abstände zwischen den einzelnen Planungsbüros als potentiell sehr groß anzunehmen.

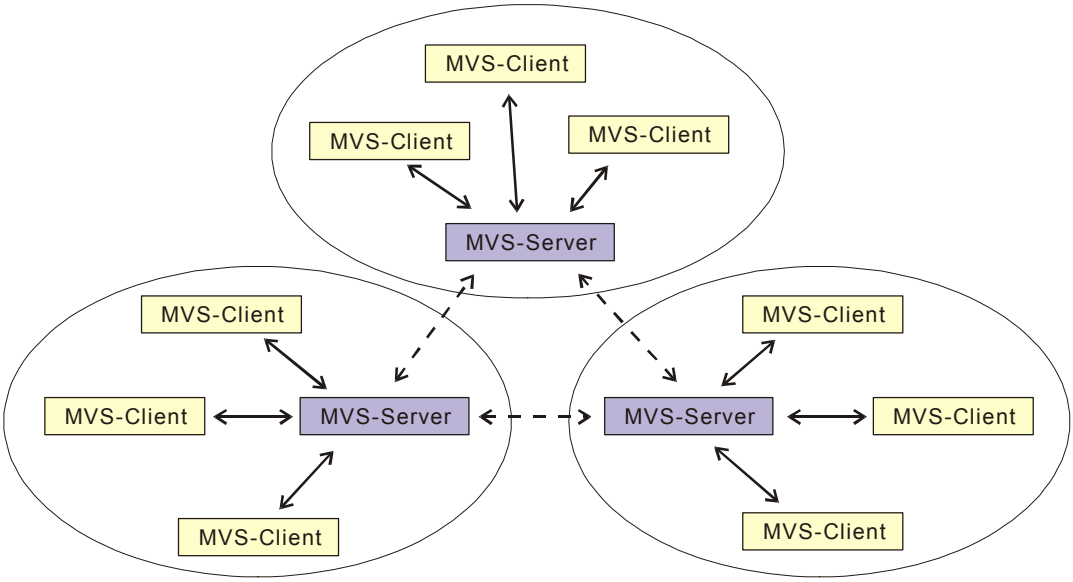

Abb.2: Hybride Systemarchitektur

Eine zentrale Verwaltung der Domänenmodelle und ihrer Instanzdaten scheidet u.a. wegen dem geringen Durchsatz und der hohen Latenz bei Datenübertragungen sowie der nicht garantierten Verfügbarkeit des Kommunikationsmediums in Weitverkehrsnetzen aus. Das entwickelte System besitzt daher eine hybride Topologie, die zudem auf physischer Ebene die Partitionierung des Bauwerkmodells in Domänenmodelle wiederspiegelt. (Abb.2)

Domänenintern werden die Planungsdaten zur Gewährleistung ihrer Konsistenz zentral gehalten: die Infrastruktur einer Domäne setzt sich aus einem Modellverwaltungsserver

(auch Domänenmodellserver) und einer beliebigen Anzahl von Clients zusammen, die die Schnittstelle des Systems zu den einzelnen Fachplanern bilden.

Der MVS-Server übernimmt die klassischen Aufgaben der Datenhaltung: Gewährleistung des verteilten Zugriffs, Sicherung der Persistenz, Kontrolle der Nebenläufigkeit, Überwachung der Zugriffsrechte und ggf. Steuerung von Transaktionen. Aus technologischer Sicht sind für die Implementierung dieser Primärdienste eines MVS-Servers neben der Kombination eines CORBA-ORBs mit CORBA-Diensten, vor allem auf serverseitigen Komponenten beruhende Technologien wie EJB[1] oder CCM[2] geeignet. [5] [7]

Der zentrale Projektserver dient rein administrativen Zwecken: Er verwaltet Informationen über die technische Infrastruktur, sowie Authentisierungs- und Authentifizierungsdaten.

3. Zugriffsmodi

Es sind grundsätzlich zwei verschiedene Modi hinsichtlich des Zugriffs von Clients auf die vom MVS domänenzentral verwalteten Daten möglich: Zum einen kann der Datenbestand der Clientapplikationen fest mit dem des MVS gekoppelt sein, d.h. ein kontinuierlicher Datenabgleich stattfinden. Die Arbeit im sogenannten *Online*-Modus ist vor allem beim gleichzeitigen Zugriff mehrerer Fachplaner auf die selbe Teilmenge der Domänendaten (synchrone Kollaboration) sinnvoll und eine notwendige Vorraussetzung für die Bereitstellung von (relaxierten) WYSIWIS[3]-Funktionalitäten. [2]

Für eine Reihe von im Entwurfsprozess auftretenden Szenarien, wie der Prüfung von Planungsvarianten und der Durchführung umfangreicher Berechnungen ist eine feste Kopplung der Datenbestände des Clients mit dem des MVS nicht wünschenswert. Anwendungsprogramme, die in solchen Szenarien zum Einsatz kommen, arbeiten daher im *Offline*-Modus. Dabei wird zunächst eine Kopie der betreffenden Daten beim Client erzeugt, die die Anwendungssoftware frei manipulieren kann. Nach Abschluss der Bearbeitung werden die veränderten bzw. neu erzeugten Daten in das MVS eingespeist.

Während diejenigen Inkonsistenzen, die infolge paralleler Bearbeitung entstehen können, beim Online-Zugriff von vornherein gänzlich vermieden werden, muss im Falle der Offline-Bearbeitung die Konsistenz durch ein in das MVS integriertes Versionsmanagement gesichert werden.

Mit Hilfe eines in die Fachanwendung eingebetteten Moduls kann der Nutzer die Art des Zugriffs je nach gewünschter Bearbeitungsart frei wählen. Dieses Modul übernimmt zu-

---

[1] Enterprise Java Beans, Spezifikation von Sun Inc., http://java.sun.com/products/ejb

[2] CORBA Component Model, Spezifikation der Object Management Group, OMG Document 02-06-65

[3] What You See Is What I See

dem die Bekanntgabe von Planungskonflikten, die im Fall des zeitlich parallelen Bearbeitens der gleichen oder voneinander abhängigen Daten im Offline-Modus auftreten können.

4. Benachrichtigungsmodul

Für die während der Arbeit eines oder mehrerer Fachplaner im Online-Modus notwendige Synchronisierung der Datenabstände von MVS-Clients und MVS-Server war die Entwicklung eines Benachrichtigungsmoduls notwendig. Die Synchronisation findet dabei auf Grundlage einer Umsetzung des Beobachter-Musters statt: Die clientseitigen Applikationsobjekte, die für die adäquate Darstellung der Daten und die Behandlung der Nutzerinteraktion zuständig sind, beobachten die vom MVS-Server verwalteten Objekte, die sowohl das Domänenmodell als auch dessen Instanzen repräsentieren. (Abb.3) Letztlich ist dies eine verteilte Anwendung des Model-View-Controller-Paradigmas, die auch als Semantic-Presentation-Split bezeichnet wird. [6]

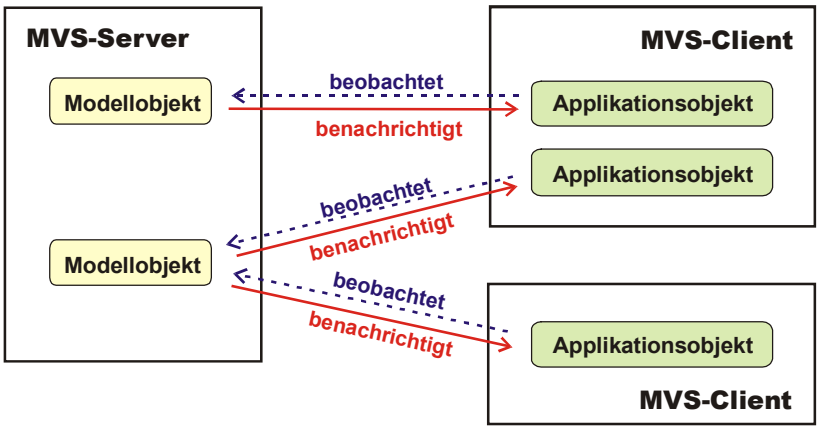

Abb.3: Funktionsweise des Benachrichtigungsmoduls

Die Applikationsobjekte werden bei Modifikationen an den von ihnen beobachteten MVS-Objekten über das in das MVS integrierte Benachrichtigungsmodul benachrichtigt. Als technologische Grundlage der Implementierung eignet sich wegen seiner hohen Flexibilität und Robustheit der entkoppelt kommunizierende *CORBA Notification Service*. [7]

Durch den Einsatz der *Typed Communication* ist eine starke Formalisierung der Benachrichtigungen mit Hilfe einer in IDL definierten Schnittstelle möglich. Diese Benachrichtigungsschnittstelle ist symmetrisch zur Zugriffsschnittstelle (AKO), d.h. zu jeder Modifikationsmethode existiert genau eine Notifikationsmethode.

Der Aufruf einer solchen Notifikationsmethode übermittelt möglichst viele Informationen über die vorgenommene Modifikation, um einen erneuten verteilten Aufruf durch den

Client zu vermeiden. Die Benachrichtigungsschnittstelle muss von den Beobachtern eines MVS-Objektes so implementiert werden, dass der Zustand der Clientapplikation mit dem des Gesamtsystems konsistent gehalten wird.

Da alle Empfänger von Benachrichtigungen als CORBA-Objekte implementiert werden müssen, erweist es sich zur Schonung der Ressourcen als sinnvoll, lediglich sogenannte Beobachtermanager als verteilte Objekte zu implementieren, die die Weiterleitung der Notifikationen an die als Beobachter angemeldeten Applikationsobjekte übernehmen.

Dank der Nutzung von Delegate[4]-Objekten im Adressraum der MVS-Clients konnte die AKO-API um Methoden zum An- und Abmelden von Beobachtern erweitert werden. Damit ist es möglich, Beobachter direkt an MVS-Objekten bzw. deren Delegates anzumelden, wodurch die interne Funktionsweise des Benachrichtigungsmoduls für den Anwendungsprogrammierer vollkommen transparent bleibt.

Die unter Nutzung des Benachrichtigungsmoduls implementierte generische Beobachterapplikation *GenObs* überwacht alle durch das MVS verwalteten Objekte und protokolliert deren Modifikationen. Sie kann der entfernten Überwachung und Fehleranalyse eines Modellverwaltungsservers dienen, sowie zur Dokumentation des Planungsprozesses genutzt werden.

5. Konnektoren für die Anbindung konventioneller Fachapplikationen

Existierende Applikationen der computergestützten Planung basieren i.a. auf einem statischen Datenmodell. Neben den generischen Anwendungen, die auf die Dynamik des Domänenmodells reagieren können, werden auch zukünftig Anwendungen Teil des Gesamtsystems sein, deren Ausschnitt aus dem Domänenmodell nur einer geringen oder gar keiner Dynamik unterworfen ist.

Das entwickelte Konzept der Konnektoren ermöglicht den Anschluss von Applikationen mit statischem Datenmodell an ein dynamisches Modellverwaltungssystem. Dabei sind vor allem solche Anwendungen geeignet, deren internes Datenmodell weitgehend konform zum im MVS vorgehaltenen Domänenmodell ist.

Konnektoren adaptieren die generische Schnittstelle auf Metaniveau (AKO) auf eine spezifische Schnittstelle gegenständlichen Niveaus. Sie bieten dabei sowohl Zugriffs- als auch Benachrichtigungsfunktionalität für Applikationen mit statischem Datenmodell. Derartige Applikationen können naturgemäß nicht auf Änderungen am Domänenmodell

---

[4] Das Business-Delegate-Pattern beschreibt Vertreter der serverseitigen Geschäftslogik-Objekte auf Clientseite, die u.a. für die Transparenz der Verteilungsfunktionalität sorgen. In: D. Alur, J. Crupi, D. Malks: Core J2EE Patterns, Markt und Technik Verlag, München 2002

reagieren. Die Konnektoren behalten jedoch auch nach Modifikationen am Domänenmodell ihre Funktionalität, da die Dynamik auf Erweiterungen um zusätzliche Klassen und Attribute beschränkt ist.

Spezifische Methodenaufrufe zum Zugriff auf ein Attribut oder eine Relation werden von den Konnektoren in generische Methodenaufrufe konvertiert und an den Modellobjekten ausgeführt. Generische Benachrichtigungen vom MVS übersetzen die Konnektoren in spezifische Benachrichtigungen für die eigenen Beobachter. (Abb.4)

Die Funktionalität der Konnektoren ermöglicht dem Programmierer, statt der abstrakten Begriffe der generischen Schnittstelle konkrete Termini der Anwendungsdomäne zu verwenden. Dies führt zu einer Verringerung der Komplexität des Quellcodes sowie zu einer stärkeren Formalisierung, die logische Fehler bereits zur Kompilierzeit erkennen und vermeiden hilft.

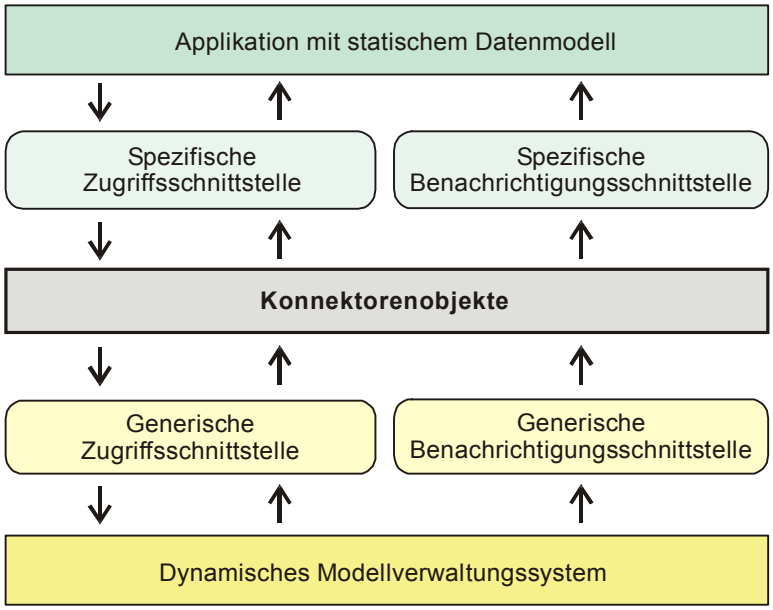

Abb.4: Funktionsweise der Konnektoren

Da sich die Konnektoren im Adressraum des Clients befinden, bleibt der verteilte Zugriff auf das MVS vollkommen transparent für den Programmierer. Zudem ist dadurch das Beobachter-Muster direkt umsetzbar: Jede Konnektorenklasse verfügt über Methoden zum An- und Abmelden von Beobachtern, die ihrerseits ein zur Konnektorenklasse gehöriges spezifisches Beobachter-Interface erfüllen müssen. Intern wird für die Implementierung dieser Methoden die Funktionalität der Delegate-Objekte verwendet.

Dank der semantischen Äquivalenz der Entitäten des AKO-Metamodells zu Konstrukten einer objektorientierten Programmiersprache ist es möglich, formale Abbildungen von AKO auf die im Projekt clientseitig eingesetzten Programmiersprachen zu definieren.

Eine solche Abbildung kann zum Zeitpunkt der initialen Konfiguration des Gesamtsystems oder nach größeren Modifikationen am Domänenmodell durch einen Generator abgearbeitet werden, um aus einem im Modellverwaltungsserver vorgehaltenen Domänenmodell Konnektorenklassen und die dazugehörigen Beobachter-Interfaces in einer spezifischen Programmiersprache zu generieren. (Abb.5)

Der Aufwand zur Integration einer Fachapplikation in das konzipierte IuK-System als Client eines Domänenservers ist entsprechend gering. Unter Nutzung der zu erwartende Plug-In-Funtionalitäten von Anwendungsprogrammen im Zuge der fortschreitenden Akzeptanz standardisierter Produktmodelle kann die Integration weitgehend automatisch erfolgen.

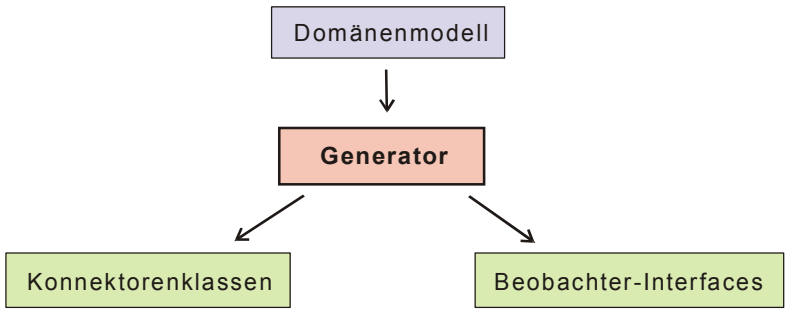

Abb 5.: Generierung von Konnektorenklassen

6. Constraint-basierte Konsistenzprüfung

Die große Menge der beim Entwurf und der Planung anfallenden Daten ist nur handhabbar, wenn ihre Konsistenz, d.h. ihre Sinnhaftigkeit gerade im Bezug auf die vielfältigen wechselseitigen Abhängigkeiten der Daten untereinander gewährleistet ist.

Eine mögliche Ursache für die Entstehung von Inkonsistenzen liegt in der Einspeisung von unvollständigen oder inhaltlich falschen Planungsdaten in das Informationssystem mit Hilfe von Fachapplikationen, die derartige Unstimmigkeiten nicht erkennen. Daneben resultieren Inkonsistenzen vor allem aus der zeitlich parallelen Offline-Bearbeitung von gleichen oder voneinander abhängigen Planungsdaten durch mehrere Planer.

Das MVS muss zur Gewährleistungen eines reibungslosen Arbeitsablaufs temporär Inkonsistenzen im verwalteten Datenbestand akzeptieren, diese jedoch ggf. aufspüren können und dem Nutzer bei deren Beseitigung assistieren. Dazu wurde das MVS um Funktionalitäten zur laufzeitdynamischen Definition von Konsistenzbedingungen (Constraints) erweitert. Sie können genutzt werden, um Bedingungen für die Gültigkeit

von Instanzdaten zu beschreiben. Constraints vervollständigen auf diese Weise die rechnerinterne Abbildung des Domänenwissens.

Für die Formulierung von Constraints wird im beschriebenen Projekt eine Untermenge der *Object Constraint Language* (OCL) eingesetzt. Sie ist Teil der UML-Spezifikation und dient ursprünglich zur formalen Spezifikation von Softwaresystemen, eignet sich aber auch zum Festlegen von Konsistenzbedingungen in Domänenmodellen. Mit ihrer Hilfe ist es u.a. möglich, komplexe Navigationsausdrücke, konditionale Konstrukte und über Kollektionen iterierende Operationen zu formulieren. [8]

Da das AKO-Metamodell nicht über Mittel zur Abbildungen von Objektverhalten (Operationen, Methoden) verfügt, müssen die Sprachkonstrukte der OCL, die der Formulierung von Vor- bzw. Nachbedingungen dienen, zunächst keine Berücksichtigung finden.

Die Constraints beschreiben Bedingungen zwischen Entitäten der Modellebene und werden vom MVS-Kern zusammen mit den betreffenden Klassen verwaltet. Der für die laufzeitdynamische Auswertung der Ausdrücke zuständige OCL-Interpreter arbeitet zur Erzielung einer für die Prüfung großer Instanzmengen notwendigen hohen Performanz im Adressraum des MVS-Servers.

Die Prüfung der Daten auf Einhaltung aller Constraints findet zu einem vom Nutzer festzulegenden Zeitpunkt statt und kann von einem MVS-Client aus angestoßen werden. Sie kann für einzelne Instanzen oder entlang von Aggregationsrelationen für ganze Instanzpopulationen stattfinden. Die Constraint-Prüfung besitzt besondere Signifikanz beim Zusammenführen von Teildatenbeständen nach deren paralleler Bearbeitung im Offline-Modus und beim domänenübergreifenden Erzeugen eines Revisionsstandes.

Beim Feststellen einer Konsistenzverletzung macht das System detaillierte Angaben über die nicht eingehaltenen Constraints und der davon betroffenen Instanzen. Auf dieser Basis kann die Ursache der Inkonsistenz lokalisiert und manuell behoben werden.

7. Ausblick

Ausgehend von der beschriebenen Funktionalität des Systems zum Formulieren und Prüfen von Konsistenzbedingungen wird zu untersuchen sein, in wie weit die aus der Forschung im Bereich der Wissensverarbeitung bekannte Technik des „Constraint-basierten Schließens“ angewendet werden kann, um dem Nutzer bei der Herstellung eines konsistenten Zustands der Planungsdaten zu assistieren.

Denkbar ist beispielsweise, dass das System zu einer vorhandenen Inkonsistenz eine Reihe von Vorschlägen zu deren Behebung generiert. Diese Vorschläge werden zu einem großen Teil aus Wertebereichen bestehen, die im Fall von geometrisch-topologischen Daten auf geeignete Art und Weise visualisiert werden müssen.

Während Konnektoren vornehmlich der Einbindung von im Online-Modus arbeitenden Fachapplikationen in das IuK-System dienen, kann eine Integration von im Offline-

Modus arbeitenden Fachapplikationen durch die Möglichkeit zum Import dateibasierter Daten erreicht werden. In diesem Zug ist auch zu prüfen, wie die initiale Konfiguration der Domänenmodelle auf Basis vordefinierter Produktmodelle weitestgehend automatisiert werden kann. Liegen Informationen über das zu verwendende Produktmodell in Form einer Datei vor (z.B. XML-Schema, EXPRESS, CORBA-IDL), so sollten diese für die Bildung des Domänenmodells im MVS genutzt werden können. In beiden Fällen ist die Definition einer Abbildung vom Metamodell der dateibasierten Sprache auf das AKO-Metamodell notwendig.

Schließlich muss die Eignung von OCL als Abfragesprache und als Mittel der Navigation im Datenbestand eines dynamischen MVS untersucht werden. Dies wird vor allem hinsichtlich der angestrebten Harmonisierung des AKO-Metamodells mit dem von UML bzw. MOF von Bedeutung sein.

Literatur

[1] Hübler, R.; Hauschild, Th.; Willenbacher, H.: Distributed Cooperative Building Models for Revivification of Buildings, International Association for Bridge and Structural Engineering (IABSE) Symposium 2002, Melbourne 2002

[2] Hauschild, Th.; Hübler,R.; Schleinitz, M.: Unterstützung von Gruppenarbeit im Bauwerksentwurf auf Basis eines dynamischen objektorientierten Bauwerks-Modelliersystems, CAD 2000, Berlin 2000

[3] Kolbe, P.; Ranglack, D.; Steinmann, F.: Eine Schnittstelle für dynamische Objektstrukturen für Entwurfsanwendungen. Internationale Konferenz für Anwendungen der Informatik und Mathematik in den Ingenieurwissenschaften IKM '97: Berichte, Bauhaus–Universität Weimar, 1997

[4] The Object Management Group: The Meta Object Facilities Specification, Version 1.4, OMG Document 02-04-03, Framingham, MA 2002

[5] The Object Management Group: The Common Object Request Broker Architecture and Specification, Version 2.2. OMG Document 98–02–33, Framingham, MA, 1998

[6] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software. Addison-Wesley, Bonn 2001

[7] The Object Management Group: Notification Service Specification, Version 1.0. OMG Document 00-06-20, Framingham MA, 2000

[8] The Object Management Group: The Unified Modeling Language Specification, Version 1.5. Chapter 6: The Object Constraint Language Specification. OMG Document 03-03-13, Framingham MA, 2003