Eine Abfragesprache für Multimodelle

Sebastian Fuchs
Institut für Bauinformatik, Technische Universität Dresden
sebastian.fuchs@tu-dresden.de

Kurzfassung: Es wird eine Methode zur unabhängigen domänenübergreifenden Informationsabfrage aus Multimodellen mit beliebigen Fachmodellen vorgeschlagen. Dazu werden der generische Zugriff auf Fachmodellinhalte sowie neue Aspekte des Multimodell-Filterns analysiert. Es werden die Abfragesprache MMQL und ein zugehöriger Interpreter vorgestellt, in denen diese Erkenntnisse zur Anwendung kommen.

1 Motivation

Multimodelle bieten Lösungsansätze für strukturelle Probleme der nD-Modellierung in Bauinformationsprozessen. Sie bündeln Fachmodelle unterschiedlicher Domänen und erlauben die Verbindung von deren Elementen in expliziten Linkmodellen. Da die Fachmodelle unberührt bleiben, wird auf diesem Weg eine lose und temporäre Kopplung ermöglicht. Durch den Verzicht auf ein führendes oder integrierendes Datenschema werden (1) keine Transformationsprozesse benötigt, können (2) etablierte und heute übliche Datenformate weitergenutzt und (3) verlinkte Fachmodelle neutral ausgetauscht werden. Solche verknüpften Daten bieten einen informationellen Mehrwert gegenüber alleinstehenden Fachmodellen. Zusammengehörende Informationen können über die persistenten Links automatisch ausgewertet werden, anstelle manuell vom Menschen immer wieder flüchtig neu zugeordnet werden zu müssen. Somit erscheint das Multimodell gegenüber einem Benutzer wie ein einziger abgeschlossener Informationsraum. Mit dem Übergang von bauwerksorientierter zu prozessorientierter Arbeitsweise erlangt die domänenübergreifende Bereitstellung von Informationen wachsende Bedeutung; bspw. bei der Erstellung von Controlling-Kennwerten, der Vorbereitung von Simulationen oder der Betrachtung neuer Aspekte wie Energieeffizienz.

Zwar ist es bald möglich, Multimodelle zwischen Softwareanwendungen auszutauschen - z.B. per RIB iTWO (Demharter et al. 2011) oder GRANID von gibGREINER (Hienz, 2011) - jedoch beschränkt sich der erzielbare Informationsgewinn auf die Anwendungsdomänen der jeweiligen Applikation sowie die implementierten Filtermechanismen. Bisher gibt es noch keine unabhängige Methode um auf Informationen in Multimodellen mit beliebigen Fachmodellen zuzugreifen. Erst mit einer generischen und vollständigen Abfragemöglichkeit kann das breite Einsatzpotential von Multimodellen in vollem Umfang realisiert werden. In dieser Arbeit wird eine solche Methode in Form einer neu entwickelten deklarativen Abfragesprache MMQL mit textueller Syntax sowie zugehörigem Interpreter vorgeschlagen.

2 Verwandte Forschung

Katranuschkov (2000) stellt eine Mappingsprache vor. Die Sprache ist deklarativ und dient der Überführung von Informationen in andere Datenstrukturen. Es werden allgemeine Prinzipien komplexer Datenstrukturen im Bauwesen analysiert. Die vorgeschlagenen Mapping-Patterns stellen unter Anderem Prinzipien zum Zugriff auf Datenstrukturen dar, die auch in Filtersprachen anwendbar sind. Willenbacher (2000) präsentiert einen linkbasierten Ansatz zur Bauwerksmodellierung, bei dem die ursprüngliche Information auf Partialmodelle

aufgeteilt wird. Der Fokus der Arbeit liegt auf der dynamischen und semiautomatischen Erstellung und Verwaltung von Mappings, realisiert durch einen hybriden Modellansatz mit zentraler Komponente. Bormann (2007) stellt eine räumliche Anfragesprache vor, die auf geometrische und topologische Aspekte von Bauwerksmodellen spezialisiert ist. Ye (2009) präsentiert eine Methode zum Zugriff auf Bauprojekt-Managementdaten über ein zentrales Datenmodell mittels Data Warehouse-Methoden und hochentwickelten Visualisierungstechnologien. Wender (2009) beschreibt eine Umgebung zur Informationssuche in domänenspezifischen Dokumenten mittels Identifikatoren, welche mit Repräsentationen von Realwelt-Objekten verlinkt sind. Tulke (2010) schlägt eine Methode zur Integration der Terminplanung basierend auf Bauwerksmodellen vor. Dabei wird eine allgemeine Sprache zum Verlinken spezifiziert, deren Fokus auf räumlichen Anfragen und Objektteilung zur Identifikation verlinkbarer Objekte liegt.

3 Multimodell-Informationen

Um domänenübergreifende Informationen aus einem Multimodell gewinnen zu können, muss der Zugriff auf die zugrunde liegenden Daten geregelt werden. Dabei sind folgende zwei Aspekte zu untersuchen: (1) die *Datenstruktur zur Bündelung und Verlinkung von Fachmodellen*, sowie (2) eine *Struktur zum generischen Zugriff auf Fachmodellinhalte*. Aus diesen Regelungen muss auch hervorgehen, welche Fachmodelle überhaupt in Multimodellen anwendbar sind. Solche Fachmodelle werden *Elementarmodelle* genannt.

Fuchs et al. (2011) schlagen zur Lösung von Aspekt (1) das *Generische Multimodell* vor; ein Datenschema sowie zugehörige Constraints und Metadaten zum neutralen Austausch und Zugriff auf Multimodelle. Elementarmodelle werden dort als übertragbare Instanz eines Datenmodells mit einer abgegrenzten Fachdomäne und einer vereinbarten Semantik definiert. Das bedeutet dass ein Elementarmodell nicht zwangsläufig ein korrespondierendes explizites Schema benötigt. Beispielsweise sind XML-Dokumente ohne XSD möglich. Lediglich die Bedeutung der Daten muss bekannt sein. Weiterhin wird davon ausgegangen, dass die verlinkbaren Elemente eine innerhalb des Elementarmodells eindeutige ID besitzen. Ausweichlösungen für Modelle ohne explizite ID werden ebenfalls beschrieben.

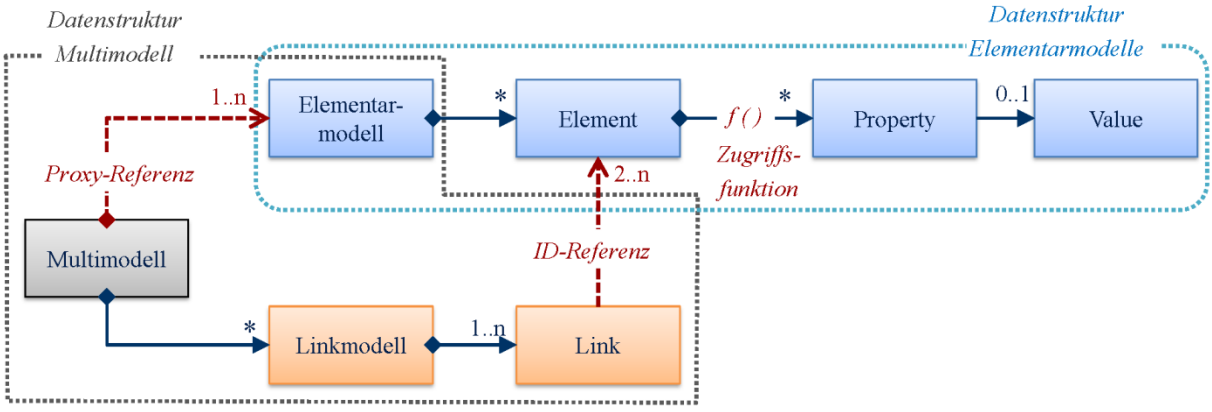

Abbildung 1:Schematischer Aufbau von Multimodellen

Diese Definition wird hier um eine Struktur zum generischen Datenzugriff erweitert. Dabei wird der innere Aufbau von Elementarmodellen ideell abgebildet. Um eine möglichst große Anzahl von Fachmodellen repräsentieren zu können, muss diese Abbildung genügend abstrakt und den Erfordernissen des Generischen Multimodells angepasst sein. Abbildung 1 zeigt den resultierenden Aufbau von Multimodellen. Der Bereich *Datenstruktur Multimodell*

spiegelt Aspekt (1), respektive das Generische Multimodell, wider. Der Bereich *Datenstruktur Elementarmodelle* repräsentiert die im Folgenden beschriebene ideelle Elementarmodell-Datenstruktur (Aspekt 2).

Elementarmodelle bestehen aus 0..n Elementen. Elemente sind die verlinkbaren Einheiten und besitzen eine ID. ID und Element-Instanz sind äquivalent. Elemente besitzen 0..n Properties. Ein Property ist eine von seinem Element aus zugreifbare Einheit hinter der sich ein Value, die eigentliche Information, befindet. Dieser Wert muss nicht gesetzt sein. Der Zugriff auf die Properties erfolgt über Funktionen, sogenannte Property Accessors. Diese können trivial sein und bspw. den Zugriff von einem XML-Element auf sein Attribut realisieren. MMQL unterstützt syntaktisch den Zugriff über Pfade auf verschachtelte Properties sowie das inhärente Property *ID*. Property Accessors können aber auch semantisch hochwertig sein und unter Anwendung zusätzlicher Logik bspw. in einem IFC-Modell die geometrische Höhe einer Wand liefern. Die Abbildung von Fachmodell-Datenstrukturen wird so teilweise an die Implementierungen für die jeweiligen Elementarmodell-Typen delegiert. Diese Implementierungen können sich an generische Vorlagen - bspw. für XML-Datenstrukturen - anlehnen oder, wie am Beispiel IFC erwähnt, semantisch hochwertige Funktionen bereitstellen. Auf diese Art und Weise sollen möglichst viele Fachmodelle zur Verwendung in Multimodellen befähigt werden.

Definition: Elementarmodelle sind übertragbare Instanzen eines Datenmodells mit einer abgegrenzten Fachdomäne und einer vereinbarten Semantik. Das Datenmodell muss sich in die ideelle Elementarmodell-Datenstruktur *Element→Property→Value* überführen lassen.

Tabelle 1: Mögliche Überführung von Meta-Datenstrukturen in die ideelle Elementarmodell-Datenstruktur

Meta-Datenstruktur	*Elementarmodell*	*Element*	*Property*
Relationale Datenbanken	Datenbank	Tabelle	Spalte
XML	Dokument	Element	Elementinhalt, Attribut
Express	Dokument	Entität	Attribut
CSV	Dokument	Zelle	Proxy-Property „Value“
INI	Dokument	Sektion	Schlüssel
JSON	Dokument	Objekt	Eigenschaft

Tabelle 1 zeigt mögliche Überführungen von übergeordneten Datenstrukturen in die beschriebene ideelle Elementarmodell-Datenstruktur. Für inhärent objekt- bzw. klassenorientierte Meta-Datenstrukturen können Komplexeinheiten (bspw. Klassen, Entitäten, Typen) auf Elemente abgebildet werden. Deren atomare Merkmale (bspw. Feld, Attribut) werden auf Properties abgebildet. Analog können auch relationale Datenbanken behandelt werden. Beim Format *Comma Separated Value* (CSV) können die Zellen als Elemente aufgefasst werden – ihre ID bestimmt sich dabei über Zeilen- und Spaltennummer. Da eine Zelle keine weiteren untergeordneten Merkmale hat, wird für ihren Inhalt ein ideelles Property „Value“ eingeführt. Somit sind alle Fachmodelle in Multimodellen nutzbar, die auf Basis abbildbarer Meta-Datenstrukturen definiert sind und deren Elemente per ID angesprochen werden können. Beispielsweise sind die Fachmodelle GAEB-DA-XML, XPDL

oder MS Project nutzbar, da sie XML basiert sind. IFC ist als Express-Instanz ebenfalls nutzbar sowie Dokumente von Tabellenkalkulationen im CSV-Format.

4 Multimodell-Filter

Multimodelle ermöglichen die Filterung des repräsentierten Informationsraumes auf neuartige Weise. Zusätzlich zu den bekannten Methoden zur Filterung von Fachmodellen kann die Einschränkung relevanter Elemente über die *Auswertung der Linkmodelle* erfolgen. Katranuschkov et al. (2010) demonstrieren die Idee im Kontext von IFC. Die Einführung einer formalen Abfragesprache erfordert eine grundlegend systematische Betrachtung der Linkauswertung. Diese gliedert sich in die unabhängigen Aspekte *Elementkombination* und *Linkinterpretation*.

Die Elementkombination legt fest, welche Elemente aus den Fachmodellen kombiniert werden sollen und wie diese Kombination zu erfolgen hat. Diese Operation kann in Analogie zur relationalen Algebra wie eine JOIN-Anweisung betrachtet werden. Dabei entsprechen die Elemente der Elementarmodelle den Relationen und die Linkmodelle den Mappingtabellen. Der Modus (1) *Natural* kombiniert demzufolge alle Elemente nach Maßgabe der vorhandenen Links in den Linkmodellen. Das bedeutet dass nur diejenigen Elemente weiterverarbeitet werden, welche auch verlinkt sind. Im Modus (2) *Right Outer* werden alle Elemente des rechten Arguments weiterverarbeitet. Elemente die nicht verlinkt sind, werden auf der linken Seite mit `NULL` aufgefüllt. Der Modus (3) *Full Cross Product* ignoriert vorhandene Links und bildet immer ein vollständiges Kreuzprodukt aller Elemente. Die Anzahl weiterzuverarbeitender Elemente wächst potentiell von Modus (1) nach (3). Es wird vermutet dass das Verhalten im Modus *Natural* der intuitiven Erwartungshaltung eines Nutzers entspricht; in der Syntax wird daher auf einen speziellen Modifikator verzichtet.

Tabelle 2: Syntaxübersicht für Linkauswertung

Element-kombination	*Linkinterpretation*		
	Strict	Standard	Transitive
Natural	`strict linkedwith`	`linkedwith`	`trans linkedwith`
Right Outer	`strict right linkedwith`	`right linkedwith`	`trans right linkedwith`
Full Cross Product	`cross linkedwith`		

Die Linkinterpretation bestimmt, welche Bedeutung ein gegebener Link für eine korrespondierende Menge von Elementen hat. Die Notwendigkeit dieser Betrachtung ergibt sich hauptsächlich durch die n:m-Kardinalität der Links. Ein Link kann n Elemente aus m Elementarmodellen verknüpfen. Im Zuge einer Abfrage können jedoch einige dieser Elemente ungültig werden, z.B. durch Einschränkungen bei Elementkombinationen oder aufgrund von Elementarmodellfiltern. Im Modus (1) *Strict* gelten Links nur dann als existent, wenn alle ihre Elemente gültig sind. Es werden also entweder alle Elemente eines Links oder keines weiterverarbeitet. Im Modus (2) *Standard* gelten Links als dann existent, wenn mindestens 2 Elemente aus mindestens 2 Elementarmodellen gültig sind. Es können also auch Elemente weiterverarbeitet werden, wenn Teile eines Links entfallen. Im Modus (3) *Transitive* werden Links transitiv ausgewertet. Ein Link wird dabei um all diejenigen

Elemente erweitert, die von seinen Elementen aus über andere Links erreichbar sind. Für die Weiterverarbeitung von Elementen gelten die gleichen Regeln wie im Modus *Standard*. Die Anzahl weiterzuverarbeitender Elemente wächst auch hier potentiell von Modus (1) nach (3). Für den Aspekt der Linkinterpretation wird vermutet, dass das Verhalten im Modus *Standard* der intuitiven Erwartungshaltung eines Nutzers entspricht; weswegen auch hier in der Syntax auf einen Modifikator verzichtet wurde.

Tabelle 2 zeigt die möglichen Kombinationen der Modi Elementkombination und Linkinterpretation anhand der Syntax der Modifikatoren für den `linkedwith`-Operator – dem MMQL-Sprachelement zur Linkauswertung. Die Multimodell-Abfragesprache MMQL kann auch lediglich als Elementarmodell-Filtersprache genutzt werden, indem ein Statement ohne `linkedwith`-Anweisung verwendet wird. Als Kriterien kommen dann Konditionen auf die Properties des gewählten Elements oder die Ausführung benannter externer Elementarmodell-Filter infrage.

5 Multimodell-Abfragesprache MMQL

Die Abfrage von Multimodell-Informationen soll mit einer Programmiersprache erfolgen. Das Sprachparadigma ist dabei mengenorientiert deklarativ, d.h. ein Anwender beschreibt das Resultat, nicht den Lösungsweg. Es wurde eine textuelle Syntax gewählt, da diese (1) ausdruckstärker als GUI-Elemente, API-Aufrufe oder eine grafische Syntax ist und (2) einen direkten Einsatz als Server-Komponente ermöglicht. Syntax und Semantik wurden weitestgehend an SQL (vgl. INCITS/ISO/IEC 2008) angelehnt um eine geringe Einarbeitungszeit und hohe Nutzerakzeptanz zu erreichen. Syntax und Semantik der Sprache können an dieser Stelle nur prinzipiell - keinesfalls vollständig - dargestellt werden.

Eine MMQL-Anfrage liefert Values von Elementarmodellen in Tabellenstruktur, das sogenannte *ResultSet*. Analog zu einer SQL-JOIN Anfrage werden dabei die Values, korrespondierend zu den m:n-Links ihrer Elemente, zeilenweise kombiniert. Der Aufbau einer solchen `select`-Anweisung ist im Wesentlichen:

```
1  // kommentar
2  USE <EDITOR multimodell_name> | <FILE pfad_zu_multimodell>
3  SELECT
4      < element.property <AS property_alias>? >*
5  FROM
6      element <AS element_alias>
7  < <STRICT | TRANS>? <RIGHT | CROSS>? LINKEDWITH
8      element <AS element_alias>? <BY linkmodell,>* >*
9  <WHERE
10     condition>*
```

Zeile 1 zeigt die Syntax von Kommentaren. Das `use`-Statement (Zeile 2) legt fest, mit welchem Multimodell gearbeitet werden soll. Im ersten Teil des `select`-Statements (Zeile 3-4) wird die Projektion deklariert, also diejenigen Properties, deren Value als Spalte im ResultSet erscheinen soll. Hier können auch Property-Accessors verwendet werden. Optional kann ein Alias vergeben werden um das Property bequemer in weiteren Ausdrücken verwenden zu können. Das erste Element wird im `from`-Teil angegeben (Zeile 5-6). Elementen kann ebenfalls ein Alias gegeben werden, bspw. zur Verwendung in Zeile 4. Alle weiteren Elemente werden über den `linkedwith`-Operator deklariert (Zeile 7-8), d.h. es wird gefordert dass die Elemente verlinkt sind. Dazu werden alle Linkmodelle in Betracht gezogen oder, wenn angegeben, nur die mittels `by` benannten. Eine Selektion kann ab Zeile 9

optional erfolgen indem Konditionen auf Properties angegeben werden. Diese Ausdrücke ähneln in Syntax und Semantik denen der WHERE-Rules in SQL. In Abschnitt 3 wurde festgelegt, dass Values nicht belegt sein müssen. Sie evaluieren dann zu `null`. Aus diesem Grund gilt in den Konditionen eine dreiwertige Logik (vgl. Codd 1979). Als Kondition können auch benannte, parametrisierte Elementarmodell-Filter ausgeführt werden, wodurch komplexe Logik außerhalb der MMQL definiert werden kann.

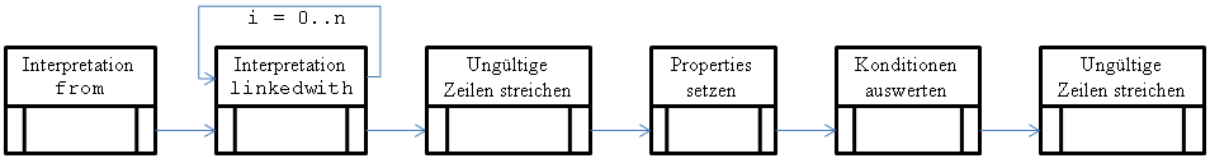

Abbildung 2:Prinzip der Interpretation einer Select-Anweisung

Abbildung 2 zeigt das Prinzip der Interpretation einer `select`-Anweisung. Nach der Auswertung des `from`-Arguments und der wiederholten Linkauswertung existiert eine Tabelle mit `n+1` Spalten kombinierter IDs. Nach Entfernung ungültiger Zeilen, die bei einigen Modi zur Linkauswertung entstehen können, werden die IDs durch die Properties der Projektion ersetzt. Dadurch entsteht das ResultSet mit seiner endgültigen Spaltenanzahl. Properties werden immer als Zeichenkette behandelt, lediglich bei Verwendung in Konditionen werden sie in einen der folgenden Typen gewandelt: String, Double, Integer, Date oder Boolean. Durch Auswertung der Kondition findet die Selektion statt – es werden diejenigen Zeilen gestrichen, die die Kondition nicht erfüllen. Schlussendlich werden noch einmal ungültige Zeilen entfernt, die nun bspw. ungewollte unvollständige Links enthalten.

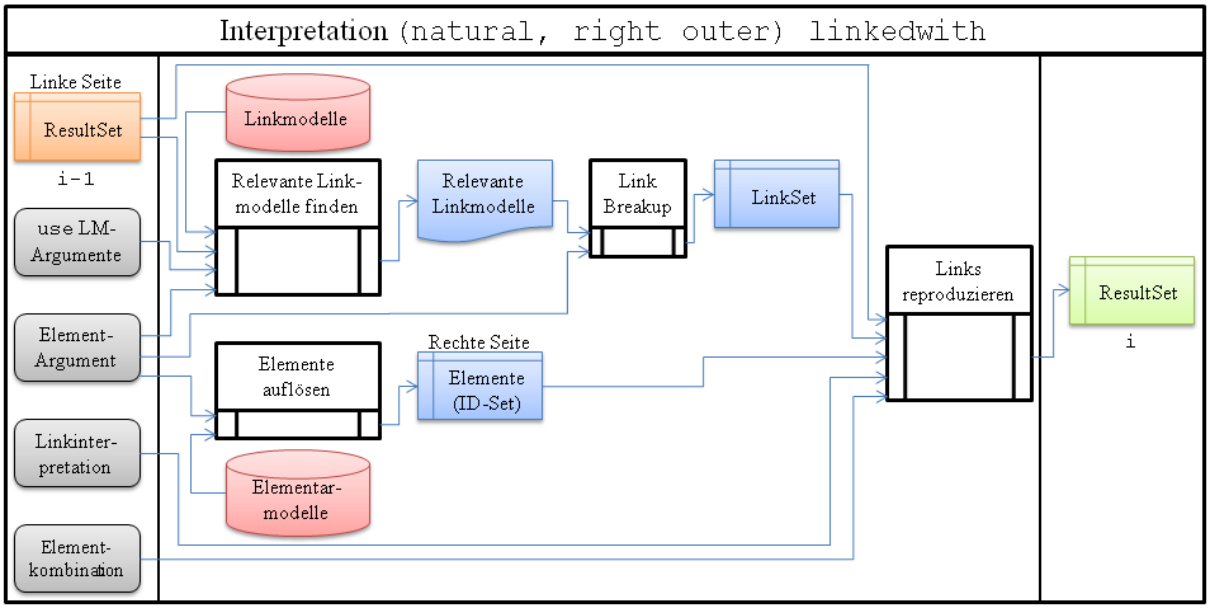

Abbildung 3:Interpreterprozess für Linkedwith-Anweisung bei Elementkombination Natural oder Right Outer

Abbildung 3 zeigt den detaillierten Interpreterprozess zur Linkauswertung. Dabei wird versucht, vorhandene Links unter Anwendung der Regeln zur Linkauswertung zu reproduzieren. Jeder Link wird dafür zeilenweise aufgespalten und entsprechend den Elementen der linken und rechten Seite kombiniert (LinkBreakup). Das Ergebnis ist ein ResultSet, das aus den Zeilen der erfolgreich teilweise oder vollständig nachbildbaren Links besteht.

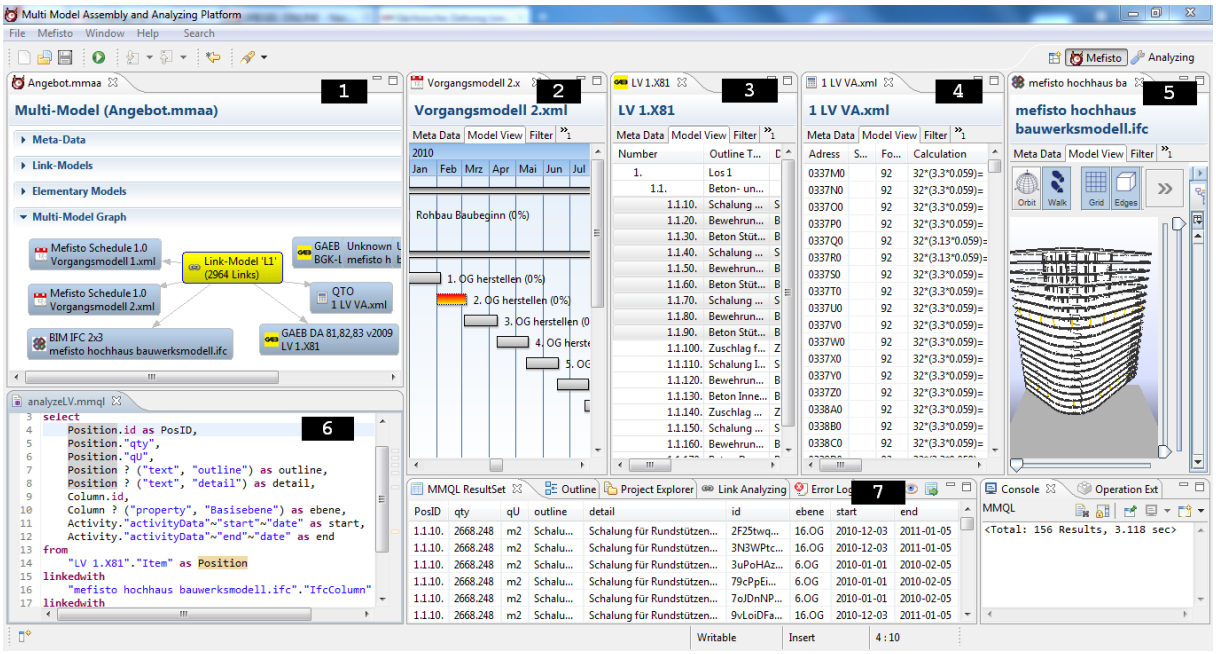

Abbildung 4:Bildschirmaufnahme Multimodell mit MMQL in M2A2

MMQL wurde mit Hilfe des Frameworks *EMFText* (Heidenreich et al. 2009) umgesetzt. Dabei wird eine textuelle Syntax (Concrete Syntax, CS) für ein Ecore-Metamodell des abstrakten Syntaxbaums definiert. Davon ausgehend wurden ein Editor sowie Interpreter-Stubs generiert und in die *Multimodel Analyzing and Assembly Platform* (M2A2) implementiert. Die M2A2-Platform ist ein Framework für Multimodell-Operationen und zur Integration von Elementarmodell-Plugins der TU Dresden (Fuchs et al. 2010). Die Software kann einschließlich der MMQL-Sprachdefinition und Beispieldateien über die Homepage des Instituts für Bauinformatik (http://tu-dresden.de/biw/cib) heruntergeladen werden. Abbildung 4 zeigt ein geöffnetes Multimodell (1), eine MMQL-Abfrage zur Erstellung eines Mittelabflussplans (6), das zugehörige ResultSet (7) sowie die Selektion der Elemente des ResultSets in den Elementarmodell-Viewern Vorgangsplan (2), Leistungsverzeichnis (3), Mengenansatz (4) und Bauwerksmodell (5).

6 Fazit

Es wurde eine Methode zur neutralen, domänenübergreifenden Informationsabfrage von Multimodellen mit beliebigen Elementarmodellen vorgestellt. Dazu wurde eine ideelle Elementarmodell-Datenstruktur zum generischen Zugriff auf Fachmodelldaten vorgeschlagen sowie das neuartige Prinzip der Linkauswertung als Teil des Multimodell-Filterns analysiert. Die textuelle, deklarative, mengenorientierte Abfragesprache MMQL greift diese Konzepte auf und bietet mit dem zugehörigen Interpreter die Möglichkeit zur Ausführung auf einem Computer. Der grundlegende Anwendungsaspekt von MMQL, die Überwindung von Datenstruktur-Heterogenität, spiegelt sich in Syntax und Semantik wider. Zur Anwender-Zielgruppe zählen daher Bau-Anwendungsentwickler oder ambitionierte Endnutzer.

Die vorgestellte Methode bietet noch Entwicklungsmöglichkeiten für höhere Performance. So könnten zu Beginn statische Konditionen (bspw. Vergleiche von Properties mit einem festen Wert) ausgewertet und damit die Größe der Zwischenergebnisse verkleinert werden. Fortführender Forschungsbedarf besteht für das programmatische Erzeugen von Links sowie für den Export des Filterergebnisses als neues Multimodell auf Basis des ResultSets.

Danksagung

Die Arbeit ist Teil des BMBF-Projekts MEFISTO sowie der EU-Projekte SARA, HESMOS und ISES.

Referenzen

Bormann, A. (2007). Computerunterstützung verteilt-kooperativer Bauplanung durch Integration interaktiver Simulationen und räumlicher Datenbanken. Dissertation. Technische Universität München

Codd, E. F. (1979). Extending the database relational model to capture more meaning. In: *ACM Trans. Database Syst., 4(4)*. S 397-434.

Demharter, J. und Pflug, C. (2011). Einsatz von Multimodellen zur Projektabwicklung beim Auftragnehmer. In: Scherer, R.J., Tauscher, H. und Schapke, S.-E. (Hrsg.): *MEFISTO: Management-Führung-Information-Simulation im Bauwesen, Tagungsband 2. Kongress*. Dresden, S. 43-58

Fuchs, S., Kadolsky, M., und Scherer, R. J. (2011). Formal description of a generic multi-model. In: Reddy, S. und Tata, S. (Hrsg.): *Proceedings of the 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE '11*. Washington, DC, USA: IEEE Computer Society. S. 205-210.

Fuchs, S., Katranuschkov, P. und Scherer, R.J. (2010). A framework for multi-model collaboration and visualization. In Proc. ECPPM 2010. Taylor & Francis. S. 115-120

Heidenreich, F., Johannes, J., Karol, S., Seifert, M. und Wende, C. (2009). Derivation and refinement of textual syntax for models. In: *Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications, ECMDA-FA '09*. Berlin, Heidelberg: Springer-Verlag. S. 114-129.

Hienz, R. (2011). Multimodellbasierte Planung und Ausschreibung von Bauprojekten. In: Scherer, R.J., Tauscher, H. und Schapke, S.-E. (Hrsg.): *MEFISTO: Management-Führung-Information-Simulation im Bauwesen, Tagungsband 2. Kongress.* Dresden, S. 147-171

INCITS/ISO/IEC (2008). INCITS/ISO/IEC 9075-1-2008 SQL/Framework, Part 1.

Katranuschkov, P. (2000). A Mapping Language for Concurrent Engineering Processes. Dissertation. Technische Universität Dresden.

Katranuschkov, P., Weise, M., Windisch, R., Fuchs, S. und Scherer, R.J.(2010). Bim-based generation of multi-model views. In: *Proc.CIB W78 2010*.

Tulke, J. (2010). Kollaborative Terminplanung auf Basis von Bauwerksinformationsmodellen. Dissertation. Bauhaus-Universität Weimar

Wender, K. (2009). Das virtuelle Bauwerk als Informationsumgebung für die Planung im Bestand zur Organisation und Strukturierung einer digitalen Bauwerksdokumentation. Dissertation. Bauhaus-Universität Weimar

Willenbacher, H. (2002). Interaktive verknüpfungsbasierte Bauwerksmodellierung als Integrationsplattform für den Bauwerkslebenszyklus. Dissertation. Bauhaus-Universität Weimar.

Ye, J. (2009). Integrating data models, analysis and multidimensional visualizations: a unified construction project management arena. Dissertation. University of British Columbia.