Model Driven Architecture für Data Warehouses: Computer Aided Warehouse Engineering – CAWE

Christian Kurze, Peter Gluchowski

Lehrstuhl für Wirtschaftsinformatik,
insb. Systementwicklung und Anwendungssysteme,
Technische Universität Chemnitz

1 Einleitung

1.1 Problemstellung und Motivation

Die Model Driven Architecture (MDA) ist ein Standard der Object Management Group (2003a) zur modellgetriebenen Software-Entwicklung. Hinter dem Ansatz verbirgt sich im Kern die Idee einer strikten Trennung von Spezifikation und Implementierung eines Systems. Modelle auf verschiedenen Abstraktionsebenen und automatisierte Transformationen zwischen Modellen setzen diese Idee um. Als neuerer Ansatz im Kontext der MDA adressiert die Architecture Driven Modernization (ADM) die Modernisierung bzw. das Reengineering von Systemen und stellt formale Modelle für diesen Kontext bereit. ADM umfasst sowohl die fachliche als auch die technische Domäne von Systemen (Khusidman und Ulrich 2007, S. 1).

Data Warehouses stellen insbesondere Informationen zur Unterstützung strategischer und taktischer Entscheidungen bereit und verkörpern dadurch zentrale Komponenten moderner Entscheidungsunterstützungssysteme. In der letzten Dekade hat das Data Warehousing einen hohen Reifegrad erreicht und flächendeckende Akzeptanz gefunden. Datenschutz, Datensicherheit und Compliance stellen zunehmend höhere Anforderungen an die Entwicklung und Dokumentation von Data Warehouses sowie an das Metadatenmanagement. Zusätzlich ist eine Schwerpunkverschiebung der Initiativen auf die fachliche Seite – weg von der IT – zu verzeichnen (Kimball et al. 2008, S. xxxi).

Diese Punkte stellen den Mehrwert einer modellgetriebenen Data Warehouse-Entwicklung heraus: Eine detaillierte Systembeschreibung auf mehreren Abstraktionsebenen reduziert die Komplexität und dient als Zusammenstellung der Anforderungen an das System und damit einhergehend auch als Dokumentation für Endnutzer und Entwickler; gleichzeitig aber auch als solide Basis für die Imple-

mentierung des Zielsystems, da die Befolgung des MDA-Paradigmas die Transformation in ein reales System erlaubt.
Die aktuelle Literatur, die MDA auf die Entwicklung von Data Warehouses anwendet, berücksichtigt den ADM-Ansatz nur unzureichend. Dennoch erweist sich vor allem das Framework von Mazón und Trujillo (2008, S. 45) als ein belastbarer Ausgangspunkt für die Entwicklung von CAWE-Werkzeugen auf Basis des MDA-Paradigmas. Die Autoren zeigen in einer Fallstudie die praktische Umsetzbarkeit ihres Ansatzes, geben allerdings keine konkreten Hinweise zur Implementierung. Ähnlich der genannten Arbeit konzentriert sich der vorliegende Beitrag maßgeblich auf die Datenhaltungsschicht einer Data Warehouse-Lösung, speziell auf die Modellierung mehrdimensionaler Datenstrukturen sowie der Generierung von Datenbankschemata aus diesen Modellen. Dabei steht der ADM-Ansatz für ein tragfähiges Fundament zur Durchführung eines Reverse-Engineerings auf bestehenden Data Warehouse-Systemen – daraus resultieren Datenmodelle für bereits existente Systeme.

Das vorgestellte Werkzeug ist sowohl für die Wissenschaft als auch für Praktiker relevant. Es stellt eine Plattform für Forscher bereit, um aktuelle Methoden des Software-Engineerings auf die Spezifika des Data Warehousings anzupassen; Praktikern hilft es bei der Bewältigung von komplexen Fragestellung im Rahmen der (Weiter-) Entwicklung von Data Warehouse-Systemen.

Aus den Ausführen ergeben sich unmittelbar zwei zentrale Forschungsfragen: Wie sind die Ansätze MDA und ADM in eine Architektur für ein CAWE-Werkzeug, das zunächst die Aspekte der Datenmodellierung abdeckt, zu integrieren? Wie ist eine solche Architektur zu implementieren?

Die vorliegende Arbeit stellt sich der Aufgabe, diese Lücke zu schließen. Die präsentierte Software-Architektur integriert sowohl MDA als auch ADM. Bereits existierende Frameworks zum effizienten Umgang mit Modellen werden auf das Problem angewendet, um durch eine prototypische Implementierung die Umsetzbarkeit der Architektur zu zeigen.

1.2 Forschungsdesign

Zur Beantwortung der aufgestellten, durch praktische Relevanz getriebenen Forschungsfragen, ziehen die Autoren den Design Science-Ansatz heran. Dieser verfolgt die Zielstellung, praktischen und theoretischen Herausforderungen durch die Erstellung und Evaluation von IT-Artefakten zu begegnen. Die so erstellten Artefakte lösen die aufgezeigten organisatorischen Probleme (Hevner et al. 2004, S. 77; Peffers et al. 2008, S. 49). Der von Peffers et al. (2008, S. 50ff.) vorgestellte Forschungsprozess findet in der vorliegenden Arbeit wie folgt Anwendung:

- *Problemidentifikation und Motivation:* Das Einleitungskapitel legte die Problemstellung immer komplexerer Data Warehouse-Architekturen und -strukturen dar und motivierte die Verwendung von MDA-Konzepten als tragfähigen Lösungsansatz.

- *Definition der Zielstellungen der Lösung:* Die gegebenen Forschungsfragen repräsentieren die Zielstellung: Es ist eine Software-Architektur zu entwickeln, die es erlaubt, MDA und ADM im Rahmen der Modellierung mehrdimensionaler Datenstrukturen einzusetzen.
- *Design und Entwicklung:* In dieser Phase finden verwandte Arbeiten zur Erreichung der gesetzten Ziele Anwendung. Die Umsetzung der Architektur erfolgt exemplarisch anhand eines Prototyps.
- *Demonstration:* Die Anwendung einer Fallstudie zeigt die Eignung des Prototyp zur Lösung des gestellten Problems.
- *Evaluation:* Im Rahmen der Evaluation ist der Prototyp auf Realwelt-Probleme anzuwenden. Die hier identifizierten Ergänzungen dienen als Input für einen Kreislauf zur weiteren Verbesserung und Erweiterung.

Die Ergebnisse eines jeden Schritts sind in den folgenden Abschnitten des vorliegenden Papers zusammengefasst. Abschnitt 5 fasst die gesamte Arbeit zusammen und gibt einen Ausblick auf zukünftige Aufgaben.

2 Verwandte Arbeiten und Zielstellungen von CAWE

2.1 Model Driven Architecture (MDA) und Architecture Driven Modernization (ADM)

Die Object Management Group (OMG) vereint als non-profit Industriekonsortium Unternehmungen um den ganzen Globus. Der Zweck seiner Gründung besteht in Beschleunigung sowie Senkung von Komplexität und Kosten in der Softwareentwicklung. Detaillierte Spezifikationen führen zu interoperablen, wiederverwendbaren sowie portablen Softwarekomponenten auf Basis standardisierter Modelle (Object Management Group 2003a, S. 1-2).

Der Mehrwert der MDA liegt in der Möglichkeit, maschinenlesbare Applikations- und Datenmodelle zu erstellen, die eine langfristige Flexibilität der Implementierung, Integration, Wartung, Test und Simulation sicherstellen. Die bereits genannte strikte Trennung zwischen funktionaler Systemspezifikation und Implementierungsdetails einer konkreten Zielplattform führt zur Einführung von drei sogenannten Viewpoints auf ein System. Im Kontext der MDA ist ein Viewpoint als Abstraktionsstufe zu verstehen, die für die jeweilige Stufe unnötige Details zur Erlangung eines vereinfachten Modells ausblendet. Jeder Viewpoint wird durch ein korrespondierendes Modell repräsentiert. Das *Computation Independent Model* (CIM), auch Domänen-Modell genannt, nutzt als implementierungsunabhängige Sicht das Vokabular der Anwendungsdomäne. Hauptzweck ist es, eine Brücke zwischen Domänenexperten und Entwicklern zu schlagen. Das *Platform Independent Model* (PIM) bleibt in dem Sinne plattformunabhängig, dass es für verschiedene Plattformen ähnlichen Typs zur Anwendung kommt. Die Ergänzung des PIM um zu-

sätzliche Details bezüglich der Umsetzung auf einer konkreten Zielplattform führt zum *Platform Specific Model* (PSM) (Object Management Group 2003a, S. 2-5). Aufgrund verschieden starker Ausdrucksstärken von Modellierungssprachen ist es durchaus möglich, dass mehrere, zueinander in Beziehung stehende, Modelle zur Beschreibung eines Systems herangezogen werden. Dieser Sachverhalt ist vor allem auf CIM-Ebene von Relevanz. Ergänzend zu Modellen auf verschiedener Abstraktionsebene spielen Modelltransformationen eine zentrale Rolle in der MDA: Sie konvertieren verschiedene Modelle des gleichen Systems ineinander; beispielsweise das CIM in ein PIM und dieses wiederum in ein PSM und anschließend in Code. Abbildung 1 fasst das beschriebene Gesamtkonzept zusammen.

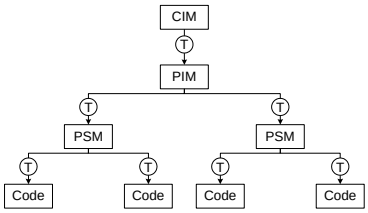

Abbildung 1: Grundkonzept der MDA (Mazón und Trujillo 2008, S. 44)

Die Meta Object Facility (MOF) stellt eine vierstufige Metamodellierungsarchitektur als Rahmen für MDA bereit. Jede Ebene definiert die jeweils darunter liegende: Die unterste Ebene enthält die zu beschreibenden Daten. Auf zweiter Ebene finden sich Modelle zur Beschreibung der Daten – sogenannte Metadaten. In der dritten Ebene sind Metamodelle anzusiedeln, die wiederum die Struktur der Metadaten definieren. Das Meta-Metamodell auf vierter Ebene beinhaltet die Beschreibung von Struktur und Semantik der Metamodelle. Es stellt also die abstrakte Sprache zur Beschreibung verschiedener Arten von Metamodellen bereit (Object Management Group 2006).

Die Architecture Driven Modernization (ADM)-Task Force der OMG setzt sich eine Spezifikation zur Modernisierung bereits existierender Applikationen zum Ziel (Object Management Group 2009). Den Kern des Ansatzes bildet das in Abbildung 2 dargestellte Hufeisenmodell (Kazman et al. 1998, S. 155; Khusidman & Ulrich 2007, S. 2; Khusidman 2008, S. 4). Es teilt sich horizontal in drei Ebenen: *Business Architecture*, *Application and Data Architecture* sowie *Technical Architecture*. Vertikal ist zwischen IST- und SOLL- bzw. Quell- und Zielsystem zu unterscheiden. Das Hufeisen beschreibt verschiedene Pfade von der ursprünglichen Lösung zur neuen Zielarchitektur. Der kürzeste Weg verbleibt dabei innerhalb einer Ebene. Veränderungen innerhalb der technischen Architektur, z.B. neue Hardware, beeinflusst lediglich eine Ebene. Der längste Transformationspfad, der in Abbildung 2 gezeigt ist, verläuft durch alle drei Ebenen. Unabhängig von dessen Länge ist jeder Transformationspfad durch drei Schritte gekennzeichnet: Strukturextraktion aus

bestehenden Systemen, Definition der Zielarchitektur sowie die Transformation des IST-Systems in das spezifizierte SOLL-System.

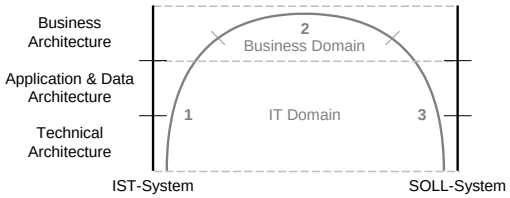

Abbildung 2: Prinzip der Architecture Driven Modernization (Khusidman & Ulrich 2007, S. 2)

2.2 Data Warehousing und MDA/ADM

Im Kontext des Data Warehousings existieren verschiedene CIMs, beispielsweise ADAPT (Bulos und Forsman 2006), DFM (Golfarelli et al. 1998) oder ME/R (Sapia et al. 1998). Allerdings konnte sich bislang keiner dieser Ansätze als allgemein akzeptierte multidimensionale Modellierungstechnik durchsetzen. Insgesamt erweist sich die Definition eines Modellierungskonzeptes, welches alle mehrdimensionalen Besonderheiten abdeckt, als schwierig. Auf PIM/PSM-Ebene konkurrieren drei Ansätze gegeneinander: Das auf relationalen Datenbanken basierende ROLAP, auf multidimensionalen Datenbanken basierendes MOLAP sowie eine hybride Kombination beider Herangehensweisen, HOLAP (Chaudhuri und Dayal 1997, S. 72). Der Unterschied zwischen PIM und PSM für ein ROLAP-System besteht beispielsweise darin, dass ein PIM weder Datentypen noch konkrete Indexierungsverfahren beschreibt; ein PSM hingegen auf die speziellen Gegebenheiten eines konkreten Systems eingeht.

In diesem Punkt unterscheidet sich die Auffassung der Autoren gegenüber der von Mazón und Trujillo (2008, S. 47). Grafische Notationen wie DFM oder ME/R sind in der vorliegenden Arbeit den CIMs zugerechnet. Gemäß Definition sind diese Modelle zur Kommunikation zwischen Endanwendern und Entwicklern gedacht (Object Management Group 2003a, S. 2-5). Auf plattformunabhängiger Ebene sind ROLAP-, MOLAP- und HOLAP-Modelle voneinander abzugrenzen. PIMs sind für eine Reihe verschiedener Plattformen ähnlichen Typs konzipiert (Object Management Group 2003a, S. 2-6). Ein relationales Star-Schema beispielsweise ist für verschiedene konkrete relationale Datenbanksysteme geeignet. Das PSM reichert ein PIM mit Spezifika einer konkreten Plattform (Object Management Group 2003a, S. 2-6) dahingehend an, dass, am Beispiel des Star-Schemas, spezifische Datentypen, Indexierungsmechanismen, Tabellenpartitionierungen und weitere Merkmale eines konkreten Datenbanksystems angegeben sind.

Das Framework von Mazón und Trujillo (2008, S. 45) deckt verschiedene Ebenen des Data Warehousings ab: Datenquellen, ETL-Prozesse, mehrdimensionale Datenmodellierung eines zentralen Data Warehouse, mehrdimensionale Wür-

fel sowie Applikationsmodelle, beispielsweise für das Reporting, die freie Datenanalyse oder das Data Mining. Im Gegensatz zu Mazón und Trujillo (2008, S. 45) fassen die Autoren die Modellierung eines relationalen Warehouses und mehrdimensionaler Würfel in eine Ebene der mehrdimensionalen Datenmodellierung zusammen. Diese Zusammenfassung resultiert aus der Unterscheidung zwischen ROLAP, MOLAP und HOLAP. Jede Realisierungsform erfordert verschiedene Beschreibungen. Im Falle von HOLAP existieren beide Beschreibungsformen und bedingen einander. Jede Schicht einer Data Warehouse-Anwendung wird von allen drei Viewpoints (CIM, PIM, PSM) aus modelliert und anschließend in Code transformiert. Die CIM-Ebene stellt dabei den wesentlichen Input für mehrdimensionale Daten- und Applikationsmodelle auf PIM-Ebene dar. Abbildung 3 fasst das Rahmenkonzept zusammen.

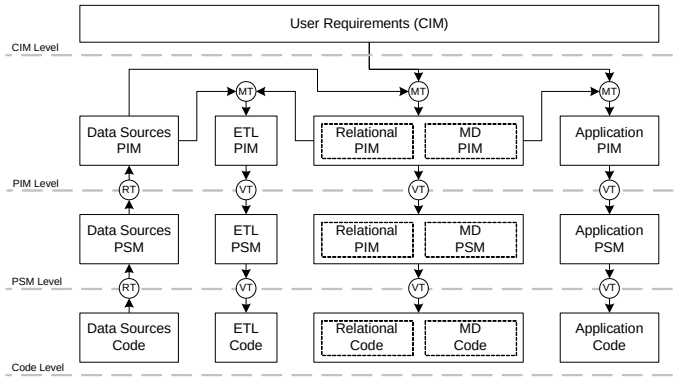

Abbildung 3: MDA-Framework für die Entwicklung von Data Warehouses (Mazón und Trujillo 2008, S. 45)

Die Überführung von Modellen ineinander erfordert eine Reihe von Transformationen innerhalb des Frameworks. Zur Verwendung in einem modellgetriebenen Vorgehen ist es notwendig, zunächst sämtliche Quellsysteme durch Modelle zu beschreiben. In den meisten Fällen existieren diese nicht und sind, ausgehend von Quelltexten bzw. operationalen Datenbanksystemen, zu entwickeln. Diese Transformationen sind als RT (Reverse Transformation) in Abbildung 3 dargestellt. Transformationen, die mehrere Modelle zu einem Zielmodell zusammenführen, sogenannte Merging Transformations, sind als MT abgebildet. Ein ETL-Modell auf PIM-Ebene vereint beispielsweise Modelle der Datenquellen mit Modellen der mehrdimensionalen Zielsysteme. Vertikale Transformationen (VT) transformieren Modelle in darunterliegende Abstraktionsebenen in Richtung Code.

Das beschriebene Framework beinhaltet ADM-Mechanismen lediglich in der Quellsystem-Ebene bei der Erstellung von PIMs aus operativen Systemen heraus. Aus Sicht der Autoren ist jedoch ein formaler und systematischer Weg zur Unterstützung des Verstehens, Anpassens und Weiterentwickelns von gesamten Data

Warehouse-Lösungen nötig. Diese Argumentation führt zur Postulation eines ADM-Ansatzes auf allen Ebenen des Data Warehousings: ETL, mehrdimensionale Daten und Applikationen. Ein derartiges Vorgehen ist auch für die Einführung eines modellgetriebenen Entwicklungsvorgehens für bereits im Einsatz befindliche Data Warehouse-Systeme unerlässlich: Im ersten Schritt sind für alle Systemelemente geeignete Modelle zu erstellen. Aufgrund der semantischen Lücke zwischen CIM und PIM stellt sich diese Aufgabe als schwierig heraus. Um Data Warehouse-Entwickler bestmöglich zu unterstützen, ist ein automatisiertes Verfahren für diesen Arbeitsschritt zu entwickeln.

Die Ausführungen zeigen eine weitere Anforderung an die Architektur auf: Sie muss komplexe Algorithmen für das Entdecken von mehrdimensionalen Elementen innerhalb von PIMs und deren Transformation in Elemente eines CIMs verwalten und ausführen können. Da den einzelnen Elementen zusätzliche Semantik hinzuzufügen ist, sind die Ergebnisse verschiedener Algorithmen zu evaluieren und in ein resultierendes CIM zu integrieren.

2.3 Fusioniertes Computing Independent Model

In Anbetracht des Fokus der zu entwickelnden Applikation – der mehrdimensionalen Datenmodellierung – erweist sich das Aufstellen eines CIMs, das alle Besonderheiten der mehrdimensionalen Modellierung unterstützt, als anspruchsvolle Herausforderung. Durch die Vielzahl verschiedener grafischer Notationen ist ein gemeinsames Metamodell von essentieller Bedeutung. Dieses muss eine Obermenge mehrdimensionaler Modellierungskonzepte sein, um verschiedene grafische Modellierungsmethoden mit unterschiedlich starken Abbildungsmöglichkeiten abzudecken. Ein derartiges Metamodell erlaubt die Modellierung mit verschiedenen grafischen Notationen, da diese problemlos in ein gemeinsames CIM – und damit auch ineinander – transformiert werden können.

Das Common Warehouse Metamodel (CWM), ein weiterer Standard der Object Management Group (2003b), versuchte diese Lücke durch Bereitstellung eines generischen Metamodells zur Beschreibung von Data Warehousing und Business Intelligence zu schließen. Dennoch ist das CWM zu allgemein gehalten, um alle Besonderheiten der mehrdimensionalen Modellierung abzudecken, und zu komplex, um von Endanwendern und Entwicklern gleichermaßen verstanden zu werden. Die Autoren folgen der Auffassung von Mazón und Trujillo (2008, S. 44) und verwenden eine reichhaltige semantische Notation als gemeinsames CIM. Ist ein Metadatenaustausch nach CWM-Standard nötig, so sind entsprechende Modelltransformationen anzuwenden, um CWM-konforme PIMs/PSMs zu erzeugen. Die Erstellung eines gemeinsamen CIM sollte auf Basis der Analyse verschiedener semantischer Notationen erfolgen. Zur Evaluation der Machbarkeit findet ein UML-Profil von Luján-Mora et al. (2006) Anwendung. Zukünftige Arbeiten sind auf einer breiten Literaturstudie aufzubauen, um möglichst alle mehrdimensionalen Konzepte abzudecken. Das herangezogene UML-Profil ist dennoch für die Um-

setzung im Prototypen vorläufig ausreichend, da es bereits auf einer Analyse verschiedener semantischer Notationen fußt.

3 Architektur und Implementierung von CAWE

3.1 Konzeptionelle Architektur

Aus den zugrunde gelegten Aspekten der Metamodellierung ergeben sich die wesentlichen Anforderungen an CAWE-Werkzeuge. Ausgangspunkt der in diesem Kapitel vorgestellten Architektur bildet deshalb ein generischer Ansatz zur Anordnung von Systemkomponenten von Karagiannis und Kühn (2002). Wie Abbildung 4 zeigt, stellt das *Meta²-Modell* die Kernkonzepte zur Erzeugung von Metamodellen und Mechanismen bereit. Es steht im Zentrum der Architektur und ist mit allen weiteren Komponenten verbunden. Modelle und Metamodelle sind im *Modell-* bzw. *Metamodell-Repository* abgelegt. Beide Repositories stehen zueinander in Beziehung, um Veränderungen an Metamodellen an die entsprechenden Modelle zu propagieren, d.h. sie synchron zu halten. Auf Modelle und Metamodelle anwendbare Funktionen sind im *Mechanismen-Repository* gespeichert. Ihre Ablage erfolgt entweder komplett oder als externe Referenz im Repository. Im zweiten Fall befindet sich zusätzlich eine Schnittstellenbeschreibung zur Interaktion mit dem Baustein im Repository. Der *Persistenzierungsdienst* ist aus den einzelnen Datenspeichern ausgegliedert: Er verwaltet die persistente Ablage sämtlicher (Meta-) Modelle und Mechanismen. Der Zugriff erfolgt transparent und unabhängig von der gewählten physischen Speichermethode wie beispielsweise Datenbanksystemen oder dem Dateisystem. Zusätzlich ermöglicht der Dienst die Verteilung von (Meta-) Modellen, d.h. sehr große Modelle können in kleineren Teilmodellen persistenziert werden. Zugriff auf die einzelnen Komponenten erfolgt über die *Zugriffsdienste* per API oder Dateiaustausch. *Viewer und Builder-Komponenten* an der Spitze der Architektur erlauben Nutzung und Wartung der gesamten Plattform.

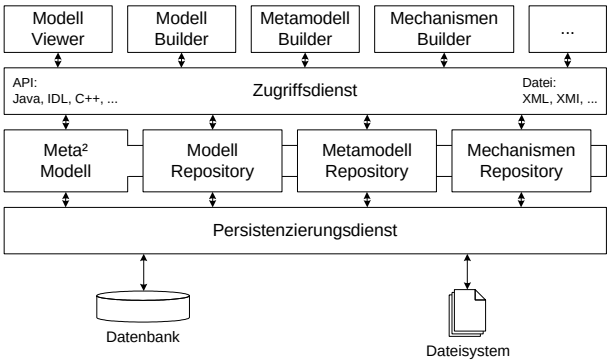

Abbildung 4: Konzeptionelle CAWE-Architektur (Karagiannis und Kühn 2002, S. 186)

Die gestellten Anforderungen der Abschnitte 2.1 und 2.2 sind durch die vorgestellte Architektur erfüllt. Sie dient als Basis für die Entwicklung von Metamodellierungswerkzeugen und deckt somit die benötigten Metamodellierungsaspekte ab. Modelltransformationen sind im Mechanismen-Repository abgelegt und lassen sich auf (Meta-) Modelle anwenden. ADM-Spezifika sind ebenfalls abbildbar: Modell-Discovery aus bestehenden Systemen ist ein Mechanismus, komplexe Algorithmen für diese Aufgabe sind durch externe Ablage ebenfalls integrierbar. Transformationen von PSMs in PIMs sind ebenfalls im Mechanismen-Repository gespeichert. Ein fusioniertes CIM, wie in Abschnitt 2.3 vorgestellt, ist im Metamodell-Repository bzw. dessen Instanz im Modell-Repository abzulegen.

3.2 Implementierung der Komponenten

Die gewählte Implementierung basiert auf dem Eclipse Modeling Project (Gronback 2009). Es handelt sich dabei um ein Open Source-Projekt mit einer großen Gemeinde an Entwicklern, die permanent an dessen Evolution arbeiten. Als Meta²-Modell findet das Ecore-Metamodell des Eclipse Modeling Frameworks (EMF) Anwendung (Steinberg et al. 2009, Kapitel 5). Die Modell- und Metamodell-Repositories sind derzeit durch einen dateibasierten Persistenzierungsdienst unterstützt. Weitere, durch EMF unterstützte Persistenzierungsmechanismen sind beispielsweise Connected Data Objects (CDO) und Teneo für den Zugriff auf Datenbanken. Das Mechanismen-Repository dient hauptsächlich zur Ablage von Modelltransformationen. Komplexere Algorithmen, insb. zum Modell-Discovery, sind in Java geschrieben. Sämtliche (Meta-) Modelle können in Form von Eclipse-Plugins bereitgestellt werden und ermöglichen somit einen transparenten Zugriff per API im Sinne der Zugriffsdienste; ein Datei-Export ist ebenso möglich. Viewer- und Builder-Komponenten sind im Rahmen von EMF ebenfalls vorhanden. Integrierte, baumbasierte Editoren lassen sich automatisch aus Metamodellen generieren; ein baumbasierter wie auch ein grafischer Editor für Metamodelle stehen zur Verfügung.

Modell-Discovery dient der Abbildung von Schemata relationaler Datenbanksysteme in Form vom Modellen – korrespondierend zu deren Metamodellen. Der Discoverer liest das Datenbankschema und legt ein Modell entsprechend der gefundenen Strukturen an. Die Implementierung ist der MoDisco-Toolbox des Eclipse Modeling Projects entnommen und entsprechend angepasst.

4 Anwendung einer Fallstudie

Zur Evaluation der vorgestellten Lösung kommt die Fallstudie eines Unternehmens zum Einsatz, das seine Absatzzahlen (Absatzmenge, Preis und Umsatz) hinsichtlich verschiedener Produkte, Organisationseinheiten und Kunden über die Zeit hinweg analysieren möchte. Dieses Beispiel lässt sich mithilfe eines Würfels

(Sales) mit vier Dimensionen (Organisational Unit, Customer, Product und Time) modellieren. Die gesamte Kette verschiedener (Meta-) Modelle ist in Abbildung 5 aufgeführt. Metamodelle sind durch Kreise mit weißen Quadraten, Modelle mit grauen Quadraten dargestellt. Die Ausführung einer Transformation repräsentiert eine Raute mit grauen Quadraten, wobei MM eine Modell zu Modell-Transformation, MT eine Modell zu Text-Transformation und MS Modell-Discovery (im Sinne einer System zu Modell-Transformation) kennzeichnet.

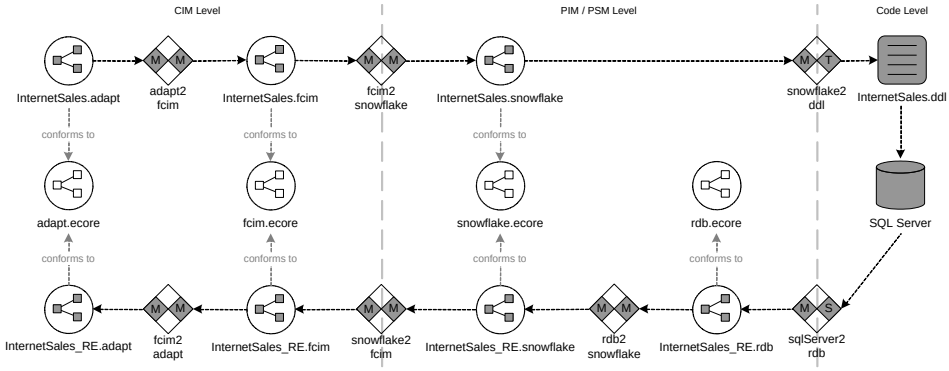

Abbildung 5: Kette von (Meta-) Modellen der Fallstudie

Das ADAPT-Modell ist zur Erhöhung der Wiederverwendbarkeit, gemäß den Ausführungen in Kapitel 2.3, zunächst in ein übergreifendes CIM zu transformieren. Aus diesem generalistischen Modell entsteht durch Transformation ein Snowflake-Schema-Modell. Das zugehörige Metamodell ist eine Erweiterung des relationalen Modells um eine zusätzliche Klassifikation der Tabellen, d.h. Fakten- bzw. Dimensionstabellen, und entstammt maßgeblich aus dem Relational Package des CWM (Object Management Group 2003b, Kapitel 6). Im nächsten Schritt sind den einzelnen Attributen der Relationen Datentypen zuzuweisen. Dieser Schritt erfolgt derzeit manuell. Aufgrund systemspezifischer Datentypen stellt dieser Schritt die Transformation des PIM in ein PSM dar; das InternetSales.snowflake-Modell ist deshalb in Abbildung 5 den Ebenen PIM und PSM zugeordnet. Ein Java Emitter Template (JET) generiert DDL-Code zur Implementierung in Microsoft SQL Server. Es erstellt Tabellen, Spalten und Fremdschlüsselbeziehungen zwischen den verschiedenen Tabellen. In der Reverse-Richtung ist das Ergebnis des Modell-Discoverers eine relationale Datenbankbeschreibung, in der durch eine zusätzliche Transformation einzelne Tabellen entsprechend ihrer Funktion (Fakt oder Dimension) klassifiziert sind.

Bis zur eigentlichen Generierung von Code ist eine Reihe von Transformationen notwendig. Auf den ersten Blick deutet dies auf hohen Aufwand hin. Eine derartige Lösung birgt jedoch Vorteile: Zur Integration eines neuen CIMs ist lediglich eine Transformation neu zu erstellen: Eine Modell zu Modell-Transformation zwischen dem neuen CIM und dem fusionierten CIM. Alle weiteren Transformati-

onen können sofort wiederverwendet werden. Das Gleiche gilt für die Anbindung weiterer Datenbanksysteme: Lediglich die Modell zu Code-Transformation sowie der Modell-Discoverer sind anzupassen.

5 Kritische Würdigung und weitere Arbeitsschritte

Das vorliegende Paper stellt erste Forschungsergebnisse dar, die in weiteren Arbeiten zu vertiefen sind. Eine Einschränkung ist vor allem im Bereich der Modell-Discovery zu sehen: Nur einfache, idealtypische Snowflake-Schemata sind derzeit unterstützt. Die Autoren arbeiten derzeit an einer Auflistung mehrdimensionaler Konzepte und deren Umsetzungen im relationalen Modell als Basis für die Erweiterung der einzelnen Transformationen. Eine Evaluation an Beispieldatenbanken verschiedener Hersteller ist geplant.

Die Verbesserung der GUI bedarf ebenso weiterer Arbeitsschritte. Derzeit findet die Arbeit an Modellen auf Ebene der abstrakten Syntax statt. Werkzeuge, wie das Graphical Editing Framework (GEF) bzw. das Graphical Modeling Framework (GMF) als Bestandteile des Eclipse Modeling Projects, sind zu evaluieren und auf die gegebene Domäne hin anzupassen. Konkrete Syntax findet derzeit nur im Rahmen kleiner Icons innerhalb der baumbasierten Editoren Anwendung.

Obwohl die vorliegende Arbeit nur beschränkt generalisierbar ist, zeigt sie das Potenzial von CAWE sowie dessen Umsetzbarkeit. Weitere Arbeitsschritte müssen folgen, um die prototypische Anwendung in einem breiten Rahmen einsetzen zu können.

Literatur

Bulos D, Forsman S (2006) Getting started with ADAPT. http://symcorp.com/downloads/ADAPT_white_paper.pdf. Abruf am 2009-09-12.

Chaudhuri S, Dayal U (1997) An overview of data warehousing and OLAP technology. ADM SIGMOD Record 26(1):65-74.

Golfarelli M, Maio D, Rizzi S (1998) The dimensional fact model: A conceptual model for data warehouses. International Journal of Cooperative Information Systems 7(2-3):215-247.

Gronback RC (2009) Eclipse Modeling Project: A domain-specific language (DSL) toolkit. Addison-Wesley, Amsterdam.

Hevner AR, March ST, Park J, Ram S (2004) Design Science in information systems research. MIS Quarterly 28(1):75-105.

Karagiannis D, Kühn H (2002) Metamodelling Platforms. In: Proceedings of the Third International Conference EC-Web, Aix-en-Provence.

Kazman R, Woods SG, Carrière SJ (1998) Requirements for integrating software architecture and reengineering models: CORUM II. In Proceedings of Working Conference on Reverse Engineering, Honolulu.

Khusidman V (2008) ADM transformation. http://www.omg.org/cgi-bin/doc?admtf/2008-06-10. Abruf am 2009-07-11.

Khusidman V, Ulrich W (2007) Architecture-Driven Modernization: Transforming the enterprise. http://www.omg.org/cgi-bin/doc?admtf/2007-12-01. Abruf am 2009-07-11.

Kimball R, Ross M, Thornthwaite W, Mundy J, Becker B (2008) The data warehouse lifecycle toolkit. Wiley, Indianapolis.

Luján-Mora S, Trujillo J, Song I (2006) A UML profile for multidimensional modeling in data warehouses. Data & Knowledge Engineering 59 (3):725-769.

Mazón J, Trujillo J (2008) An MDA approach for the development of data warehouses. Decision Support Systems 45(1):41-58.

Object Management Group (2003a) MDA Guide V1.0.1. http://www.omg.org/cgi-bin/doc?omg/03-06-01. Abruf am 2009-11-17.

Object Management Group (2003b) Common Warehouse Metamodel (CWM) specification version 1.1. http://www.omg.org/spec/CWM/1.1/PDF/. Abruf am 2009-11-17.

Object Management Group (2006) Meta Object Facility (MOF) Core specification version 2.0. http://www.omg.org/spec/MOF/2.0/PDF/. Abruf am 2009-11-17.

Object Management Group (2009) Architecture-Driven Modernization task force. http://omg.org/adm/. Abruf am 2009-07-29.

Peffers K, Tuunanen T, Rothenberger M, Chatterjee S (2008) A Design Science Research methodology for information systems research. Journal of Management Information Systems 24 (3):45-77.

Sapia C, Blaschka M, Höfling G, Dinter B (1998) Extending the E/R model for the multidimensional paradigm. In Proceedings of the ER'98 Workshops on Data Warehousing and Data Mining, Singapore.

Steinberg D, Budinsky F, Paternostro M, Merks E (2009) EMF: Eclipse Modeling Framework. Addison-Wesley, Amsterdam.