Datenmodellierung im Zeitalter agiler Softwareentwicklung

Sebastian Graf
PROMATIS software GmbH
Ettlingen

Schlüsselworte
Agile Softwareentwicklung, Datenmodellierung, Entity Relationship, UML

Einleitung
Diskutiert man heute mit Fachleuten über das Thema Datenmodellierung, dann stellt man häufig fest, dass der Begriff sehr vielfältig belegt ist. Noch vor wenigen Jahren war klar, dass sich hinter einem Datenmodell eine formale logische und physische Beschreibung der Strukturen einer zumeist mit objektrelationalen Datenbanken umgesetzten Anwendung verbarg. Die logische Sicht wurde in der Regel mit der Entity-Relationship-Methode (ER-Methode) abgebildet. Spricht man heute hingegen über Datenmodellierung, dann wird darunter sehr oft ein fachliches Klassen- oder UML-Modell verstanden, welches zur Umsetzung der Anwendungskomponenten herangezogen wird. Mit dieser Verschiebung der Begrifflichkeiten geht leider ebenfalls die Erkenntnis einher, dass ein relationales Datenmodell nicht mehr benötigt wird und sich die Datenbankstrukturen quasi als Abfallprodukt aus dem fachlich orientierten Klassenmodell ergeben. Diese Vernachlässigung der relationalen Sicht einer Anwendung führt in der Praxis in der Regel zu größten Problemen:

- Physische Datenmodelle orientieren sich an der dialogorientierten Denkweise von Klassenmodellen und lassen Aspekte der performanten Batch-Verarbeitung außen vor.
- Beim Übergang von einem Klassenmodell werden komplexe Abbildungen von Vererbungsstrukturen in ein dazu passendes objektrelationales Modell vollkommen übersehen. In der Regel wird bei der Umsetzung der Weg des geringsten Widerstandes gewählt, was im Endergebnis wiederum zu mitunter völlig unperformanten Abfragen und überbordenden Ressourcenverbräuchen führt.
- Durch das Fehlen einer relationalen Beschreibung der Datenbankstrukturen sind viele Unternehmen heutzutage gar nicht mehr in der Lage nachzuvollziehen, wie Daten eigentlich gespeichert werden. Ein Umstand, der bei einer notwendigen Nachweispflicht recht schnell zum unternehmensweiten Problem werden kann.
- Ebenfalls sind in vielen Unternehmen komplexe Versorgungsstrecken, über die Daten von einem Subsystem via ETL-Prozessen in andere Abnehmersysteme verteilt werden, völlig unklar, da die zugrunde liegenden ETL-Prozesse ebenfalls nicht strukturiert abgebildet wurden.

Der Vortrag zeigt auf, welche Probleme sich in Summe aus einer unzureichend durchgeführten Datenmodellierung im Laufe der Zeit ergeben und wie diesen Problemen rechtzeitig begegnet werden kann. Ein Schwerpunkt liegt dabei auf der Transformation von Klassenmodellen in objektrelationale Modelle sowie der Diskussion wie im Zeitalter der agilen Softwareentwicklung trotzdem stabile und tragfähige objektrelationale Modelle erstellt werden können, die allen Anforderungen an den stabilen Betrieb einer Anwendung gerecht werden.

Datenmodellierung - aber bitte richtig!

„Natürlich haben wir für unsere Anwendung ein Datenmodell!“ Diesen Satz höre ich immer, wenn ich zu Beginn eines Kundeneinsatzes nach einem solchen Modell frage. „Herr Meier wird Ihnen das Modell umgehend per E-Mail zur Verfügung stellen.“ Die Zeit bis zum Eintreffen dieser E-Mail verbringen meine Kollegen und ich in der Regel damit, darauf zu wetten, woraus wohl der Anhang dieser E-Mail besteht. Mögliche Kandidaten sind: PowerPoint-Diagramme, unstrukturierte Word Dokumente, endlose Excel-Listen, deren einzelnen Tabellen in einem neuen Reiter dokumentiert werden, Visio-Zeichnungen, UML-Diagramme, Codeausdrucke wahlweise in Cobol oder Java, textuelle Beschreibungen von Dateiformaten oder Zugriffsstrukturen (mapping.xml) usw. Dabei sei klar gesagt, dass es sich bei dieser Liste nicht etwa um eine dramatisierende Überhöhung dessen handelt, was in vielen deutschen Unternehmen unter dem Begriff Datenmodell verstanden wird, sondern es handelt sich hierbei um die Realität, welche man als Datenmodellierer vorfindet. Aus Sicht eines Modellierers muss allerdings gesagt werden, dass die oben beschriebenen Artefakte nichts, aber auch wirklich gar nichts, mit einem Datenmodell, wie es im Kontext dieses Dokuments verstanden wird, zu tun haben. Warum dies so ist und was die minimalen Anforderungen an ein Datenmodell bzw. ein Modellierungswerkzeug sind, soll im Folgenden kurz beschrieben werden.

Ein Datenmodell besteht aus zwei grundlegenden Elementen: dem logischen und dem physischen Modell. Das logische Modell beschreibt alle für die Datenhaltung relevanten fachlichen Aspekte einer Anwendung. Zur Erstellung strukturierter, logischer Datenmodelle hat sich im Kontext relationaler Datenbanksysteme die Entity-Relationship-Methode etabliert und bewährt. ER-Modelle bestehen aus zwei wesentlichen Elementen: den Entitäten und deren Attributen, welche Objekte der realen Welt beschreiben, und den Beziehungen (Relationen), die diese Entitäten untereinander eingehen. Für jedes Attribut einer Entität ist der Datentyp zu definieren und es ist zu beschreiben, ob es sich bei dem Attribut um ein identifizierendes Schlüsselattribut, um ein Pflichtattribut oder um ein optionales Attribut handelt. Beziehungen werden in der Regel über ihre Kardinalität und ihre Modalität beschrieben. Die Kardinalität beschreibt, wie oft eine Beziehung bestehen kann. Möglich sind hier 1:1, 1:n oder n:m Beziehungen. Die Modalität hingegen beschreibt, ob die Beziehung zwischen zwei Entitäten bestehen kann oder ob sie bestehen muss.

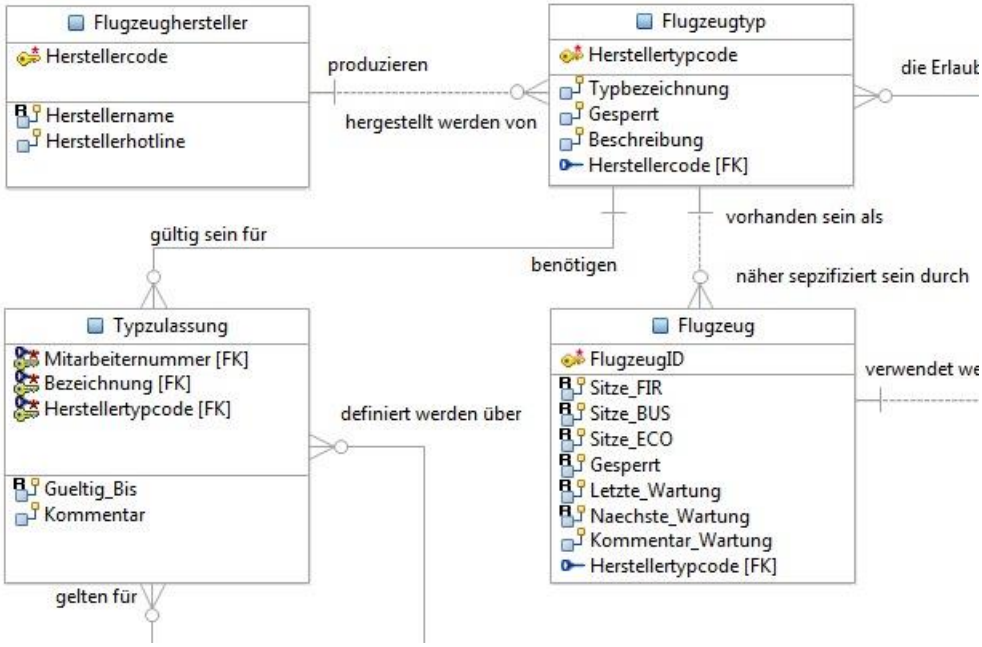

Abbildung 1: Beispiel für ein logisches Datenmodell

Im Gegensatz zum logischen Datenmodell beschreibt das physische Datenbankmodell alle technischen Aspekte der Implementierung der Entitäten und Beziehungen aus dem logischen Modell. Dazu gehören unter anderem die folgenden Objekttypen: Tabellen, Views, Indexe, Constraints, Zugriffsberechtigungen, Informationen zur Partitionierung von Tabellen und der physischen Anordnung der Daten (Clusterbildung), Stored Procedures, Datenbanktrigger u.v.m.

Bezogen auf die Verbindung der beiden Modelle ist beim Einsatz eines Modellierungswerkzeugs darauf zu achten, dass beide Modelle durch entsprechende Generatoren bidirektional ineinander überführt werden können. Es muss also möglich sein, eine Änderung an einem der beiden Modelle automatisch und verlustfrei in das jeweils andere Modell überführen zu können. Ferner muss es möglich sein, aus dem physischen Modell Skripten zu generieren, die die Anlage der Objekte auf einer vorhandenen Datenbank erlauben. Hier ist insbesondere darauf zu achten, dass das verwendete Werkzeug den kompletten Funktionsumfang der Zielplattform unterstützt und nicht nur rudimentäre Skripten erstellt werden, die anschließend von einem Datenbankadministrator noch aufwändig überarbeitet werden müssen.

Eine weitere zentrale Anforderung an ein Datenmodellierungswerkzeug besteht in der vollständigen Unterstützung eines Reverse Engineering-Prozesses. Was unter Reverse Engineering zu verstehen ist und welche Vorteile in dieser Vorgehensweise bestehen, wird in einem der folgenden Abschnitte ausführlich beschrieben.

Kontextmodelle und das unternehmensweite Datenmodell
Normalerweise besteht die Datenlandschaft eines Unternehmens nicht nur aus einem einzigen Datenmodell. In der Regel wird pro Anwendung ein Datenmodell erstellt. Bei einer wachsenden Anzahl an Datenmodellen entsteht so in der Regel sehr schnell der Wunsch über ein unternehmensweites Datenmodell zu verfügen, in dem alle Datenbankobjekte abgebildet werden. Bei der Schaffung eines solchen Modells sind diverse Fallstricke zu beachten:

Zunächst ist sicherzustellen, dass alle Datenmodelle stets auf einem aktuellen Stand sind – ein Datenmodell, welches einmal initial erstellt wurde und dann im Laufe der Zeit nicht ständig an Änderungen angepasst wird, ist schließlich wertlos. Diese Anforderung an die Aktualität aller Datenmodelle lässt sich normalerweise über organisatorische Regelungen erreichen. So ist es äußerst zweckmäßig die Entwicklungsprozesse dahingehend auszurichten, dass Änderungen auf Test- und Produktionssystemen erst dann durchgeführt werden, wenn diese in das entsprechende Datenmodell eingepflegt wurden. Flankierend kann festgelegt werden, dass die relevanten Änderungsskripte alle aus dem Datenmodellierungswerkzeug generiert werden müssen und in der gesamten Organisation keine manuell geschriebenen Skripte mehr akzeptiert werden.

Zum Aufbau eines unternehmensweiten Datenmodells empfiehlt sich ein divide-and-conquer-Ansatz. Dabei werden alle Einzelmodelle in einem globalen Übersichtsmodell zusammengeführt. Praktischerweise wird dabei für jedes Modell ein Übersichtsdiagramm erstellt (s.u.), welches alle Schnittstellen zu anderen Systemen aufzeigt und welches für die systemübergreifende Navigation und Auswertung von Datenmodellen verwendet werden kann. Nicht alle Tool-Hersteller bieten diese Möglichkeit der Integration unterschiedlicher Modelle. Hier ist bei der Anschaffung eines Werkzeugs genau zu prüfen, welche Anforderungen an ein unternehmensweites Modell bestehen und wie diese mit dem Modellierungswerkzeug umgesetzt werden können.

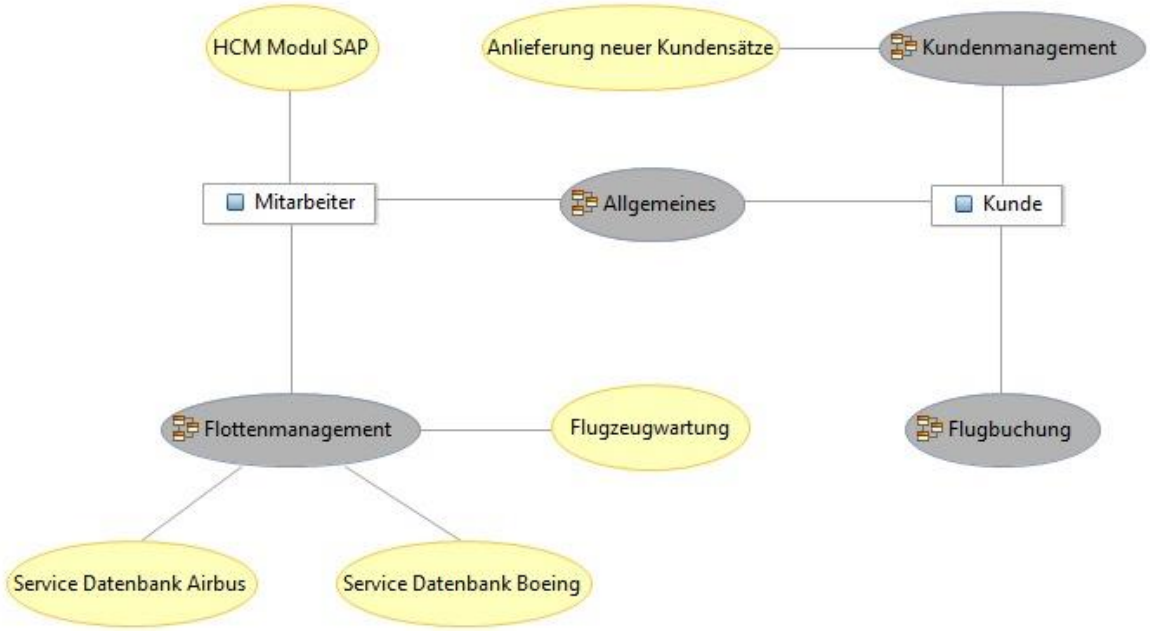

Abbildung 2: Beispiel eines Kontextdiagramms

Transformation Klassenmodell nach Datenmodell

Eine der zentralen Herausforderungen im Bereich der Datenmodellierung besteht eigentlich nicht in der Beherrschung der oben beschriebenen ER-Methode und deren Umsetzung mit einem geeigneten Werkzeug. Diese Tätigkeiten sind aus Sicht eines erfahrenen Datenmodellierers Standardabläufe. Probleme bereitet in der täglichen Praxis vielmehr die Verbindung zwischen der objektorientierten und der objektrelationalen Welt. Im Bereich der Anwendungsentwicklung hat sich heute das UML-Verfahren insbesondere bei Java-basierten Anwendungen durchgesetzt. Hier werden zunächst Anforderungen an die Funktionalität einer Anwendung über User Stories und Aktivitätsdiagramme beschrieben, die schließlich über diverse Zwischenschritte in ein objektorientiertes Klassenmodell münden. Auf Basis dieses Klassenmodells und der funktionalen Beschreibung werden dann die benötigten Anwendungskomponenten entwickelt. In der Praxis werden dabei immer wieder zwei folgenschwere Fehler gemacht: Zum einen werden in den User Stories ausschließlich Anforderungen an die Dialogkomponenten einer Anwendung berücksichtigt. Anforderungen an Batch-Prozesse, die vielfach einen großen Teil einer Anwendung ausmachen, werden komplett ignoriert, was dann in der Regel zu einem Datenmodell führt, welches für performante Batch-Verarbeitung komplett ungeeignet ist. Der zweite Kardinalfehler, der immer wieder gemacht wird, besteht in der unreflektierten Übernahme des objektorientierten Designmodells in ein vollkommen identisches Datenmodell. Dieser Fehler geschieht in der Regel deshalb, weil beim Übergang vom OO-Modell zum ER-Modell entweder die Anwendungsentwickler, die für das Designmodell verantwortlich sind, mit den Anforderungen an ein gutes ER-Modell hoffnungslos überfordert werden, oder weil erst gar kein versierter Datenmodellierer bei der Transformation hinzugezogen wird. In diesen Fällen können bereits in dieser frühen Phase eine kommende Schieflage des Projekts und bevorstehende Refactoring-Orgien vorhergesagt werden. Wird hingegen ein erfahrener Modellierer in dieser frühen Phase hinzugezogen, so wird er immer wieder Fragen nach den Anforderungen der Batch-Anteile der Anwendung stellen und in der Lage sein, auf Basis der fachlichen Modelle bereits Rückschlüsse auf die zu erwartenden Datenbankzugriffe zu ziehen. Dadurch besteht die Möglichkeit, die vorgeschlagenen Modelle entsprechend zu überarbeiten, um es in der Folge gar nicht erst zu Performance-Engpässen aufgrund eines unpassenden Datenmodells kommen zu lassen. Dieser Sachverhalt soll am Beispiel der Generalisierung exemplarisch beleuchtet werden:

In der Objektorientierung gibt es das Konzept der Vererbung bzw. der Generalisierung, welches traditionelle objektrelationale Datenbanken nicht kennen. Im vorliegenden Beispiel wird das Objekt „Person“ näher durch die Objekte „Mitarbeiter“ und „Kunde“ beschrieben. Alle drei Objekte verfügen über eigene Attribute, wobei die Attribute der Person jeweils an den Kunden und den Mitarbeiter vererbt werden. Hier stellt sich die Frage, wie ein solches Konstrukt in einer objektrelationalen Datenbank umgesetzt werden soll. Prinzipiell gibt es dafür drei mögliche Verfahren:

- **Rollup**: Hier werden alle drei Objekte in einer Tabelle „Person“ persistiert, wobei die Attribute der drei Objekte als Spalten in der Tabelle zusammengefasst werden.

- **Rolldown**: Hier werden die drei Objekte in zwei Tabellen „Mitarbeiter“ und „Kunde“ persistiert. Die Attribute der Person werden dabei in den beiden Tabellen aufgenommen.

- **Separate Table**: Bei dieser Umsetzung werden alle drei Objekte in einer separaten Tabelle persistiert und mit entsprechenden Beziehungen versehen.

Bei der Wahl der Umsetzungsvariante sind viele Faktoren der späteren Nutzung der Tabellen zu berücksichtigen. Existieren z.B. nur Zugriffe, die immer auf Kunden und Mitarbeiter bezogen sind, und nicht zwischen beiden Typen unterscheiden, so mag ein Rollup die beste Lösung sein. Ist jedoch das Gegenteil der Fall, dann wird es sehr wahrscheinlich besser sein, über eine Trennung der Objekte nachzudenken. In der Realität ist die Entscheidung für eine der drei Umsetzungsvarianten deutlich vielschichtiger. Tatsache ist jedoch, dass die Entscheidung über das passende relationale Modell nicht dem Zufall überlasen werden sollte. In der Praxis stellt man leider immer wieder fest, dass sich die Anwendungsentwickler bei einer fehlenden Unterstützung durch einen Datenmodellierer in der Regel für einen Rollup entscheiden. Schließlich hat man es dann nur mit einer Tabelle zu tun und der Entwicklungsaufwand ist vermeintlich deutlich geringer. Diese Entscheidung kann sich aber in der Folge der Entwicklung als kapitaler Fehler herausstellen. Insbesondere dann, wenn komplexe Batch-Prozesse ins Spiel kommen, bei denen es in erheblichem Maße auf Verarbeitungsgeschwindigkeit ankommt. Klar ist auch, dass eine spätere Korrektur einer solchen Fehlentscheidung zu erheblichen Aufwänden führt. So muss das Datenmodell und der Anwendungscode angepasst werden, was in der Regel mit einem sehr kostspieligen Migrationsprojekt einhergeht.

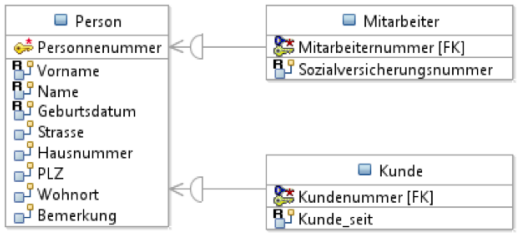

Abbildung 3: Beispiel einer Generalisierungsbeziehung

Agilität und Datenmodellierung

In den letzten Jahren haben sich agile Entwicklungsverfahren zunehmend durchgesetzt. Viele Unternehmen setzen Scrum oder Kanban ein oder denken zumindest über deren Einsatz nach. Alle agilen Verfahren basieren auf der Überlegung, komplexe Projekte in sehr kleine und überschaubare Einheiten zu unterteilen, deren Dauer meistens im Bereich von ein bis zwei Wochen liegt. An dieser Stelle sollen nicht die Vor- und Nachteile agiler Verfahren betrachtet werden. Trotzdem muss auf Basis diverser praktischer Erfahrungen mit diesen Verfahren darauf hingewiesen werden, dass sich agile Verfahren aus Sicht des Autors in keiner Weise für die Entwicklung eines Datenmodells eignen.

Man stelle sich nur vor, welche Qualität ein Datenmodell haben wird, welches evolutionär über kleine Entwicklungsabschnitte entsteht ohne jemals das Modell in seiner gesamten Komplexität betrachtet und ansatzweise verstanden zu haben. Leider ändern auch neue und innovative Entwicklungsverfahren nichts an der unumstößlichen Tatsache, dass das Datenmodell einer Anwendung dem Fundament eines Bauwerks gleicht, mit dem die Stabilität des Gesamtkunstwerks steht und fällt. Damit soll nicht gesagt werden, dass sich Agilität und Datenbanken per se nicht vertragen. Aber man sollte in einem agilen Projekt auf jeden Fall bereits in einer sehr frühen Phase anstreben, zu einem hochgradig stabilen Gesamtmodell zu kommen, auf dem die Entwickler anschließend in diversen Entwicklungsabschnitten aufbauen können. Wird das Datenmodell selbst in kleinste agile Einheiten unterteilt und ergibt es sich im Projektverlauf quasi evolutionär, dann muss auch hier mit hochaufwändigen Rework-Szenarien und Refactoring-Maßnahmen gerechnet werden.

Reverse Engineering

Der Vorgang des Revers Engineering stellt ein Verfahren dar, ein Datenmodell ausgehend von einer bereits existierenden Datenbankanwendung zu entwickeln. Dieses Verfahren kommt immer dann zum Einsatz, wenn es für eine bestehende Anwendung kein Datenmodell gibt (z.B. weil auf die Erstellung eines solchen Modells im Rahmen der Entwicklung verzichtet wurde). In der Regel unterstützen alle Hersteller von Datenmodellierungswerkzeugen diesen Prozess. Beim Reverse Engineering wird zunächst auf Basis der existierenden Datenbankstrukturen durch Auslesen der Dictionary-Informationen ein physisches Modell erstellt, welches dann in einem zweiten Schritt in ein logisches Model überführt werden kann. Trotzdem ist dieser Prozess sehr aufwändig, da zumindest auf der logischen Ebene eine große Menge an Dokumentation über die Fachlichkeit der Objekte nachzudokumentieren ist, da diese Informationen normalerweise nicht im Dictionary der Datenbank vorgehalten werden. Darüberhinaus kann in der Praxis beobachtet werden, dass viele Unternehmen leider nach wie vor auf die Implementierung von Beziehungen über Fremdschlüssel auf der Datenbank verzichten. Diese somit fehlenden Informationen, die ja einen wesentlichen Teil eines logischen Datenmodells ausmachen, müssen im Zuge eines Revers Engineering ebenfalls nachträglich manuell erfasst werden, was mit unter einer sehr zeitaufwendigen Forschungsarbeit entspricht.

Dokumentation

Ein weiterer wichtiger Aspekt, der bei der werkzeuggestützten Erstellung von Datenmodellen unbedingt beachtet werden muss, ist die Erstellung geeigneter Datenmodelldokumentationen. Normalerweise bieten alle Tool-Hersteller hier eine Grundmenge von Berichten, mit denen ein Datenmodell in unterschiedlichen Formaten bereitgestellt werden kann. Üblicherweise kommen hier Word- oder HTML-Berichte zum Einsatz, die dann z.B. in einem Intranet publiziert werden können. Diese Art der Publikation von Datenmodellen bietet zwei entscheidende Vorteile: Zum einen muss nicht jeder Entwickler, der auf Basis des Datenmodells Anwendungscodes entwickeln muss, mit einer Lizenz für das Modellierungswerkzeug ausgestattet und in der Verwendung des Werkzeugs geschult werden. Zum anderen vereinfacht sich der Prozess der ständigen Aktualisierung der Dokumentation durch eine automatische Generierung der betroffenen Dokumente immens. Viele Unternehmen fordern eine ständige Aktualisierung der Dokumentation aller Datenmodelle (Stichwort ISO-Zertifizierung). In einzelnen Fällen, wie z.B. im Finanzdienstleistungsumfeld, geben sogar externe Partner die konkreten Eckwerte einer solchen Dokumentation vor. Somit lohnt es sich vor Anschaffung eines Modellierungswerkzeugs zu prüfen, welche Dokumente über das Werkzeug automatisch generiert werden können und ob der Anwender die Möglichkeit hat, die vorhandenen Standardberichte an seine spezifischen Anforderungen anzupassen.

Datenverteilungsstrecken

Der Begriff der Datenversorgungsstrecke wird normalerweise immer mit der Versorgung eines Data Warehouse Systems mit Daten aus diversen operativen Systemen in Verbindung gebracht. Dabei werden aus den operativen Systemen Daten entnommen, in einem Zwischenschritt entsprechend für

das Zielsystem transformiert und dann ins Zielsystem geladen. Man spricht hier von einem ETL-Prozess (Extract-Transform-Load). Darüberhinaus gibt es aber in der Praxis in diversen Fällen durchaus die Anforderung, auch operative Daten in verschiedenen Systemkomponenten redundant vorzuhalten. Erschwerend kommt noch hinzu, dass nicht selten Daten über mehrere Stufen verteilt werden. Aus Sicht der Datenmodellierung ergibt sich hier die Anforderung, zu jeder Zeit genau sagen zu können, wie die Daten in einem Zielsystem versorgt werden und wo ein Datum seinen Ursprung hat. Existiert hier keine geeignete Dokumentation, dann erfordert die Beantwortung dieser Frage sehr oft erheblichen Forschungsaufwand. Im besten Fall ist die Information in den Köpfen der jeweiligen Spezialisten vorhanden. Auch hier kann die Datenmodellierung einen erheblichen Beitrag zur Vereinfachung leisten. So bieten viele Tool-Hersteller heute Werkzeuge an, die auf den logischen und physischen Datenmodellen aufbauen. Mit ihnen können in so genannten Mapping-Modellen unterschiedliche Quell- und Zielsysteme verbunden und die Transformationen strukturiert dokumentiert werden. Auf diesen Mapping-Modellen können anschließend Analyseprozesse aufsetzen, die ein Zielfeld über diverse Stufen bis zum Ursprung verfolgen können und die dabei durchlaufenen Transformationen transparent machen können. Diese Vorgehensweise hat nicht nur den Vorteil der Transparenz: Die Informationen in den Mapping-Modellen können einem Entwickler, welcher für die Implementierung einer Versorgungsstrecke verantwortlich ist, unschätzbare Dienste leisten. In einzelnen Fällen können sogar die SQL-Anweisungen, die zur Versorgung des Zielsystems benötigt werden, direkt aus den Mapping-Modellen abgeleitet werden.

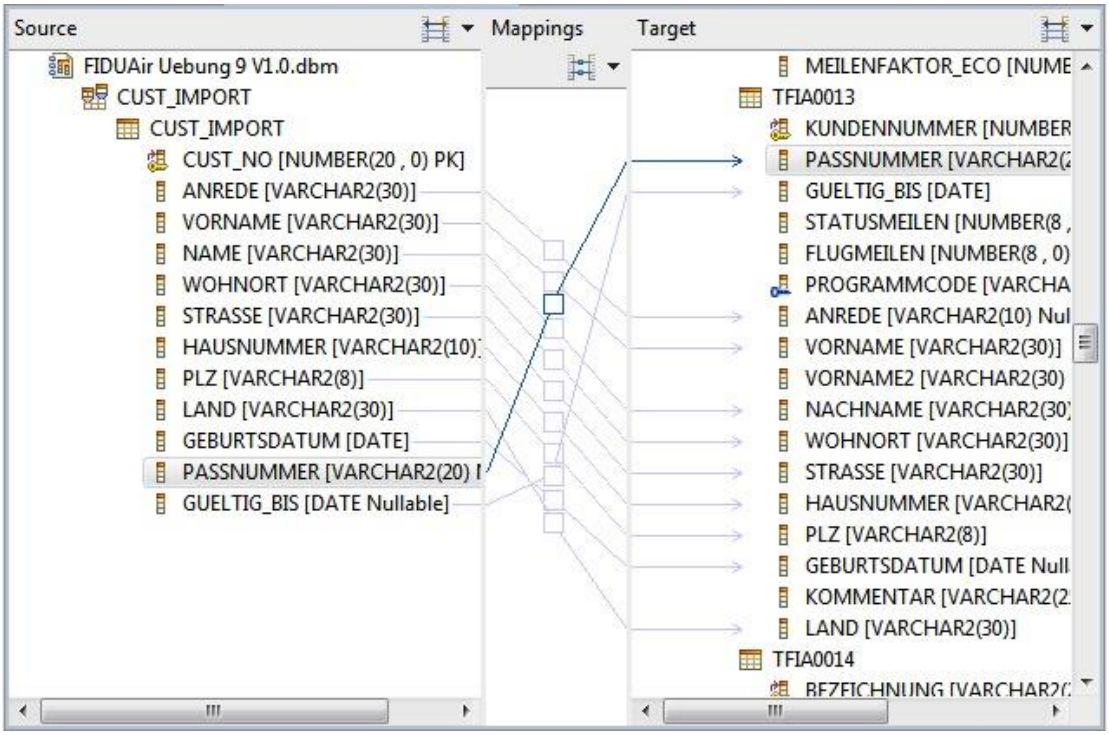

Abbildung 4: Beispiel eines Mapping-Modells

Abschließend sei noch erwähnt, dass es sich für viele Unternehmen bei der Dokumentation solcher Versorgungsstrecken nicht um einen angenehmen Luxus handelt, sondern dass auch hier beispielsweise eine erfolgreiche ISO-Zertifizierung die strukturierte Dokumentation dieser Strecken erfordert. Im Finanzdienstleistungsumfeld beispielsweise fordert die Bundesanstalt für Finanzdienstleistungsaufsicht eine lückenlose Dokumentation, auf welchen operativen Daten eine Kennzahl in einem DWH-Report basiert.

Zusammenfassung
Die Disziplin der Datenmodellierung, wie sie im vorliegenden Dokument beschrieben wurde, hat in den letzten Jahren leider an Bedeutung verloren. Vielfach wird nur noch der objektorientierte Anteil einer Anwendung betrachtet, manchmal sogar ganz auf einen modellbasierten Ansatz verzichtet und direkt mit der Erstellung von Anwendungscodes begonnen. Trotzdem ist ein Unternehmen, welches über objektrelationale Datenbanksysteme verfügt, gut beraten, das Thema Datenmodellierung fest in den Entwicklungsprozess zu integrieren. Die Vorteile, die sich daraus ergeben, werden sich auf jeden Fall durch eine deutlich erhöhte Qualität und eine spürbare Reduktion nachträglicher Fehlerkorrekturmaßnahmen innerhalb kürzester Zeit bezahlt machen.

Kontaktadresse:
Dipl.-Inform. Sebastian Graf
PROMATIS software GmbH
Pforzheimer Straße 160
D-76275 Ettlingen

Telefon:	+49 (0) 7243-2179-0
Fax:	+49 (0) 7243-2179-99
E-Mail	sebastian.graf@promatis.de
	sebastian.graf@doag.org
Internet:	http://www.promatis.de
	http://www.horus.biz