Ph.D. Thesis

Conceptual Models and Model-Based Business Metadata to Bridge the Gap between Data Warehouses and Organizations

Conducted for the purpose of receiving the academic title
'Doktorin der Sozial- und Wirtschaftswissenschaften'

Advisors

Prof. Gerti Kappel
Institute of Software Technology and Interactive Systems
Vienna University of Technology

Prof. Juan-Carlos Trujillo-Mondejar
Department of Software and Computing Systems
University of Alicante, Spain

Submitted at the
Vienna University of Technology
Institute of Software Technology and Interactive Systems

by

Veronika Stefanov
9902228
Lerchenfelder Str. 156/16
1080 Wien

Vienna, November 2007

Contents

Thank you very much!

For your advice, interest, supervision, support, friendship, questions, criticism, information, praise, laughter, help, insights, ideas, love:

Andrea Berg, Andrea Schauerhuber, Beate List, Bente Knoll, Birgit Korherr, Brigitte Ratzer, Christian Breiteneder, Christian Huemer, Christiane Floyd, Christiane Ulbricht, Dimitra Fellner, Doris Kastner, Elisabeth Reitinger, Elke Michlmayr, Emilio Soler, Gerhard Kramler, Gerti Kappel, Helga Stefanov, Horst Kargl, Ille Gebeshuber, Jesus Pardillo, Johann Geyer, Johanna Dohnal, Jose-Norberto Mazon, Juan-Carlos Trujillo, Julia Lindenmair, Katharina Kaiser, Katrina Vanura, Leopoldine Rücker, Luigi Mangiacapra, Manuel Wimmer, Marianna Tolar, Marijana Zeleznikar, Marion Murzek, Marko Banek, Marko Smiljanic, Martin Morandell, Martina Seidl, Martina Umlauft, Michael Schadler, Michael Strommer, Monika Lanzenberger, Nevena Stolba, Nicole Strohmann, Olaf-Michael Stefanov, Petra Brosch, Rahel Bekele, Sabine Graf, Sonja Hnilica, Sonja Willinger, Stefanie Scherzinger, Ulli Pastner, Ute Riedler, Vera Kuzmits, Viktoria Stefanov, Wiebke Beckmann, and many more[1].

[1]This research has been funded by the Austrian Federal Ministry of Science and Research (the former Austrian Federal Ministry for Education, Science, and Culture), and the European Social Fund (ESF) under grant 31.963/46-VII/9/2002.

Abstract

Data warehouse systems are used by decision makers for performance measurement and decision support. Measures such as the number of transactions per customer or the increase of sales during a promotion are used to recognize warning signs and to decide on future investments with regard to the strategic goals of the organization.

Currently, the main focus of the data warehouse research field is on database issues. The data warehouse's interaction with the organization and the way it supports the organization's strategic goals have not yet been considered in depth. Conceptual models that describe the data warehouse from various viewpoints, including an outside view of the data warehouse system, its environment and expected usage, are missing. Moreover, even though the data in the data warehouse by its very nature has to be closely related to the concerns of the organization, current data warehouses also lack sufficient business metadata that would inform users about the organizational context and implications of what they are analyzing.

This thesis targets the relationship between the data warehouse and the structure, behavior, and goals of the organization.

In order to describe this relationship, a conceptual modeling language was developed. It consists of models for describing the interdependencies between data warehouses and business processes, including so-called active data warehouse solutions; a model for identifying business objects such as customers and products in the data warehouse data model, and for constructing data models that comply to the state models of such business objects; as well as a model of data warehouse usage, which includes modeling the users, user groups, and user skill levels, the intensity with which they use the data warehouse infrastructure, temporal issues such as the required time and urgency of data access, and indicators of the relative importance of data warehouse usage.

This thesis also introduces an approach to using models to enhance the way users access the data in the data warehouse. It presents model-based business metadata, which links enterprise models such as business process models or goal models to the data model of the data warehouse through the mechanism of model weaving. A prototype illustrating how models can be weaved and used for business metadata in a business intelligence tool has been developed as part of this thesis.

Kurzfassung

Data Warehouse-Systeme werden für die Analyse großer Datenmengen und zur Entscheidungsunterstützung verwendet. Kennzahlen wie zum Beispiel die Veränderung der Verkaufszahlen während einer Aktionsphase oder die Anzahl der Transaktionen pro Kunde dienen zur Erkennung von Trends und stützen Investitionsentscheidungen, die von Unternehmenszielen geleitet werden.

Der Zusammenhang zwischen dem Data Warehouse und der Organisation, die es verwendet, war bisher noch nicht Gegenstand eingehender Untersuchungen. Es gibt keine Modelle, die ein Data Warehouse aus verschiedenen Blickwinkeln - so auch von außen, die Interaktion mit seiner Umgebung oder seine Verwendung - beschreiben könnten. Obwohl die im Data Warehouse enthaltenen Daten eng mit dem Unternehmen verknüpft sind, enthalten heutige Data Warehouses keine Business-Metadaten, die die BenutzerInnen über den Kontext und die Implikationen der Daten informieren könnten.

Diese Dissertation befasst sich mit der Beziehung zwischen dem Data Warehouse und der Struktur, dem Verhalten und den Zielen der Organisation.

Um diese Beziehung zu beschreiben, wurden neue konzeptionelle Modelle entwickelt. Ein Modell beschreibt die wechselseitigen Abhängigkeiten zwischen Geschäftsprozessen und dem Data Warehouse, und erlaubt es dabei auch, sogenannte aktive Data Warehouse-Systeme zu erfassen. Ein weiteres Modell indentifiziert Business-Objekte (zum Beispiel Kunden oder Produkte) im Datenmodell des Data Warehouses, und ermöglicht die Konstruktion von Datenmodellen, die das Zustandsmodell dieser Objekte abbilden. Weiters wurde ein Modell entwickelt, das die Verwendung des Data Warehouses beschreiben kann, und zwar im Bezug auf BenutzerInnen und Gruppen, ihre Kompetenzstufen, die Nutzungsinensität, benötigte Zeit und Dringlichkeit, sowie die Wichtigkeit des Zugriffs.

Diese Dissertation enthält außerdem einen Ansatz zur Unterstützung der BenutzerInnen bei der Dateninterpretaion. Basierend auf Modellen und Modellierungstechniken werden dem Data Warehouse Business-Metadaten hinzugefügt. Diese Business-Metadaten verknüpfen Unternehmensmodelle (zum Beispiel Geschäftsprozess- und Zielmodelle) mit dem Datenmodell, unter Verwendung von Weaving-Modellen. Ein Prototyp für modell-basierte Business-Metadaten, der auf in der Praxis verwendeten Data Warehouse-Komponenten aufsetzt, wurde im Rahmen der Dissertation entwickelt.

Chapter 1

Introduction

Contents

1.1 Problem Statement and Research Question

Data warehouse systems integrate data from heterogeneous sources to support the analysis of the behavior, the development, and the results of an organization [KRRT98]. Measures such as the increase of sales during a promotion, the number of transactions per customer, or system response time are used by organizations to recognize warning signs and to decide on future investments with regard to their strategic goals.

Business Intelligence (BI) is a wider concept, but sometimes used interchangeably with data warehouse. In general, BI is used to describe all kinds of applications and technologies for storing, analyzing, and providing access to data intended to support enterprise users to make better business decisions [GRC04]. BI covers the entire data warehouse environment from data storage to analysis, and includes charting tools, data mining algorithms, and alerting mechanisms.

A data warehouse is tightly interwoven with the organization that surrounds it. If data warehouse and BI systems are to support business users in making the right decisions, they have to be aligned with the strategy and goals of the business organization. The data that is fed into the data warehouse describes and mirrors the structure and behavior of the organization. Which measures are meaningful and what the values implicate depends on what the goals of an organization are and how it intends to do business.

Currently, the main focus in the data warehouse research field is on database issues, such as view maintenance, aggregation of data, query rewriting, indexing, data quality, and schema integration [Vas00]. The context of the data warehouse [MSAP07, GRC04], its interaction with its surroundings, how it is influenced by the organization, and the way it supports the organization's strategic goals, have not yet been considered in detail.

Conceptual models in the area of Data Warehousing are strongly data-orientated [Riz04] and do not allow for describing data warehouse context. Models that describe the data warehouse from various viewpoints, including an outside view of the data warehouse system, its environment and expected usage, are missing. Moreover, even though the data in the data warehouse by its very nature has to be closely related to the concerns of the organization, current data warehouses also lack sufficient business metadata that would inform users about the organizational context and implications of what they are analyzing [Sar01] (e.g., the business process that has supplied the value, or for which goals this metric is important; as opposed to access restriction properties or the name of the source data column, which are classical technical metadata).

The context of the data warehouse and how it is related to the organization can be used to support the design of data warehouses as well as to help the users who need to understand the data provided by the data warehouse correctly. Data warehouses interact with business processes, are related to the organizational structure, and their data measures the fullfillment of goals.

This thesis targets the relationship between the data warehouse and the organization with two interrelated research questions:

How can the relationship between the data warehouse and the structure, behavior, and goals of the organization
(1) be formally described?
(2) support the interpretation of data?

1.2 Research Goals, Field and Scope

Enterprise models are used to formally represent the basic building blocks of an organization, its structure, behavior and goals [WRK01]. All aspects described in such a model are relevant to the data warehouse in one way or another.

During the past few years, the field of Model Engineering has experienced a considerable development and increase in the number of tools, techniques, and standards. Even though some approaches are less mature than others, it is possible to create new models based on well-known standards, to manage models from various domains together, and to extend or to integrate existing models. Models can be used to generate program code, and models can be used as a way to store information.

Following from the research questions formulated in Section 1.1, two goals were defined for this thesis:

Develop a Conceptual Modeling Language for Data Warehouse Design and Usage. *How can the relationship between the Data Warehouse and the structure, behavior, and goals of the organization be formally described?* A number of diagrams to show how the organization is related to the data warehouse, to model how the organization interacts with the data warehouse, and how its structure and behavior are mirrored by the data warehouse can be used for this purpose.

Create Model-Based Business Metadata. *How can the relationship between the Data Warehouse and the structure, behavior, and goals of the organization support the interpretation of data?* Business metadata describes the business context of the data, its purpose, relevance, and potential use. It is different from technical metadata which specifies, e.g., how the data is structured and stored. Knowledge about the organization, captured in an enterprise model, can be linked to the data warehouse by means of model weaving ([dFBJ$^+$05], a Model Engineering technique) and used to gain business metadata that can be displayed to users in an analysis tool. *Model-based*[1] business metadata means that the approach uses models, modeling techniques, and modeling tools for creating, storing, managing and editing the metadata.

These goals represent different ways of applying knowledge about the relationship between the data warehouse and the organization, and they achieve contributions in different areas (see Section 1.6). This thesis positions itself in a multidisciplinary research field between Model Engineering, Enterprise Modeling, and Data Warehousing, because it applies modeling techniques to data warehouse as the application area, and aims at bringing data warehouse and Enterprise Modeling closer together, as visualized in Figure 1.1.

The scope of this thesis is limited to models on the conceptual level, and covers the relationship between the organization and the data warehouse only, i.e., it concerns the data warehouse itself, and not data warehouse development projects (which have their own structure, behavior, and goals, and also interact with the organization), technical details of data mapping and data warehouse design, or data warehouse design methodology.

1.3 Conceptual Models

Regarding the first goal, *how the relationship between the data warehouse and the structure, behavior, and goals of the organization can be formally described*, conceptual models have been

[1] as opposed to *model-driven*

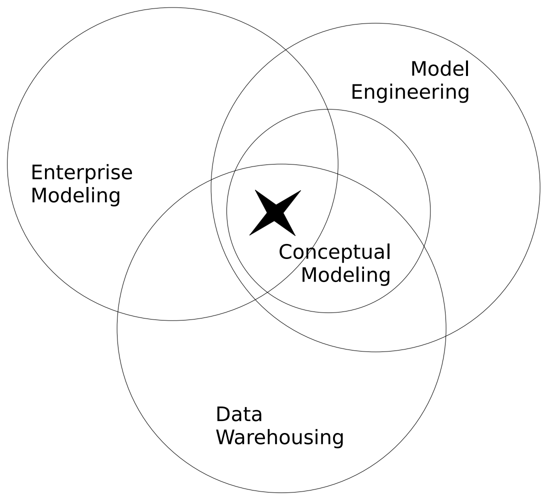


Figure 1.1: Research Field(s)

developed for different aspects of this relationship.

Most of the models are based on UML 2.0 and implemented as UML Profiles. Chapter 4 also includes models based on ARIS and EPCs.

1.3.1 Business Processes

Chapter 4 describes models that link business process models and data warehouse models. Data Warehouse information is accessed by business processes. A so-called Active data warehouse may also automatically initiate changes in the the control flow of business processes. The models introduced in Chapter 4 allow to show

- where and how business processes use a data warehouse environment,
- which parts of the processes depend on which parts of the data warehouse, and
- how the data warehouse impacts the business process control flows.

1.3.2 Business Objects

Data Warehouse users analyze business objects relevant to an enterprise organization (purchase orders or customers) and are interested in the states of these objects: e.g., a customer is either a potential customer, a first time customer, a regular customer or a past customer; purchase orders may be pending or fullfilled. Business objects and their states can be distributed over the data warehouse's data model, and appear implicitly in measures, dimension attributes, levels, etc.

Chapter 5 introduces a model for business object states in a data warehouse and 14 correspondence patterns, that

- show where the business objects and states can be analyzed in the data warehouse
- show how the business object states relate to different aspect of a multidimensional data warehouse data model
- provide hints for constructing a data warehouse data model.

1.3.3 Data Warehouse Usage

Today's data warehouse systems provide many different services to different kinds of users. Chapter 6 describes how four aspects of data warehouse usage can be captured by a model. The model can be used to find answers to questions such as:

- Who are the users and how are they grouped together?
- Which part of the data warehouse system do they use? How do they use it?
- How intensely are which parts of the data warehouse being used by which users?
- When do users need to use which part, and how time critical is it?
- How important is a certain access pattern?

1.4 Model-Based Business Metadata

To achieve the second goal, *how the relationship between the data warehouse and the structure, behavior, and goals of the organization can support the interpretation of data*, weaving models are used to link conceptual enterprise models with the data warehouse data model. Chapter 3 describes how business metadata can be created from these links. Section 3.3 introduces business metadata based on enterprise goals and metrics, followed by Section 3.4 on business metadata derived from the enterprise's structure, behavior and products.

A prototype that illustrates the creation and use of business metadata and is built on a real-world data warehouse, including example models and data, is described in Chapter 7.

1.5 Evaluation

The conceptual models and the business metadata described in Sections 1.3 and 1.4 have to be evaluated. The contribution of this thesis are not the lower-level models themselves (e.g., a model of the current situation in company A, or the data in system B), but new metamodels, i.e., new modeling languages for describing concepts that cannot be described sufficiently with existing modeling languages.

Conceptual models are difficult to evaluate, and there are no generally agreed-upon evaluation criteria for modeling languages [Fra00, Fra06]. The purpose of such a language is to allow the creation of models that describe some aspect of reality in a useful, concise, expressive, and easily understandable way. A metamodel in this sense includes the definition of the language, as well as the notation used. As described in detail by [Fra00] and [Fra06], the level of proficiency that users have with a certain language greatly influences their opinion of the language. Therefore there can be no universal idea of "easily understandable" or "expressive". Furthermore, approaches such as comparing the number of different concepts that can be described with a language are also ambiguous in their evaluation, as it is not decidable whether a metamodel with more or fewer concepts is "better" for the resulting models.

Existing related approaches to conceptual modeling in the area of Data Warehousing are applied to examples, case studies, and scenarios (see [TPGS01, LMT04b, LMTS06, ASS02, BSHD98, FS99, TBC99, AGS97] for examples). A comparison to existing models is used to show that and how the proposed approach provides a contribution to the research area.

The models presented in this thesis were all applied to several examples and scenarios, including the models behind the model-based business metadata. The examples are all either (simplified) real-world scenarios or adapted from literature. The prototype described in Chapter 7 illustrates the use of the models within a data warehouse tool with example data and queries well-known in the Data Warehousing community.

1.6 Contributions and Beneficiaries

The conceptual models described in Section 1.3 provide benefits during the earlier phases of the data warehouse lifecycle, whereas business metadata (Section 1.4) mainly supports the operational phase. The conceptual models provide:

Increased visibility and a bigger picture The models provide a straightforward way to make relationships visible. They allow the analysis of the implications of changing scenarios (e.g., removing a component, increasing the number of users) on various levels of detail.

Improved communication The models visualize the overall structure of data warehouses on the conceptual level, thus replacing the custom of creating *ad hoc* diagrams and drawings for the communication with users and decision makers.

Facilitated requirements analysis Cases where there are no links between organizational entities and the data warehouse, which may indicate business requirements that are not yet addressed, can be recognized.

Streamlined data warehouse evolution and re-engineering Some models (e.g., the data warehouse usage models described in Section 6) can be used to support the design of personalized user interfaces or user access controls. The models generally allow to prioritize the subprojects according to business needs. They are used to identify critical patterns and to identify parts of the data warehouse that are not used or not used very often, or not used for important purposes.

Documentation and maintenance The models can also be used to support estimates of the cost of usage, as well as for risk management: If the data quality in a certain area is bad, a data mart fails, or data is corrupted, an integrated model shows which business processes or users will be affected.

Business metadata adds background information directly to the data warehouse:

Improved data interpretation, enhanced usability and user acceptance of data By relating the measures in the data warehouse to enterprise goals and organizational concepts, users are better able to interpret the performance of the enterprise, and to understand the implications.

Facilitated requirements analysis Data warehouse requirements analysis and (re-)design are notoriously challenging tasks, because the business context of a data warehouse is difficult to extract from user interviews and practically impossible to store directly in the multidimensional data structures. Business metadata captures this information.

Streamlined data warehouse evolution and re-engineering As a by-product, the weaving model used to create the business metadata is used for model validation, as it identifies missing or superfluous tables and measures in the data warehouse, as well as omissions in the enterprise model.

The beneficiaries of this thesis fall into two major groups:

- All people involved in designing, building and maintaining a data warehouse (i.e., the architects and designers, as well as the future users) benefit from conceptual models. Their tasks are facilitated, and their project communication is improved by capturing volatile and implicit knowledge.
- Users and maintainers benefit from improved interpretation through business metadata.

Chapter 2

State of the Art

This chapter contains the background knowledge about the state of the art required by the later chapters. The following sections cover the state of the art in the areas Data Warehousing and Business Intelligence (Section 2.1), Enterprise Modeling (Section 2.2), and Model Engineering (Section 2.3).

In Data Warehousing, the research focus lies on database issues [Riz04, Vas00]. In Enterprise Modeling, the Data Warehouse is either seen as a generic data source or remains hidden behind BI metrics and measurement systems.

The concepts and approaches that are situated between these three areas – and thus comparable to the ideas presented in this thesis – are quite few: Section 2.1.4 on data warehouse requirements, Section 2.1.5 on business metadata, and Section 2.1.7 on data warehouse context contain descriptions of related approaches.

Contents

2.1 Data Warehousing and Business Intelligence

This section gives an overview of the current state of the art of main aspects of Data Warehousing, and how they are related to and used in the following chapters of this thesis.

2.1.1 Fundamentals

There is no overall definition of a "Data Warehouse". The most commonly quoted definition is:

> A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management's decision-making process [IH94].

In many modern organizations, Data Warehouse systems are meant to represent a single source of information for analyzing the status, the behavior, the development and results of an organization [LM04, KRRT98]. Analysts and decision makers analyze measures – such as the number of transactions per customer or the increase of sales during a promotion – with regard to the goals and the strategy of the organization, and use them to recognize warning signs or trends and to decide on future investments.

A Data Warehouse can be seen as a "long-term buffer between transactional processing and analytic processing" [JLV$^+$01]. A Data Warehouse is separate from operational systems, and usually can be seen as a kind of meta-database. It integrates data from various heterogeneous sources and provides it to analysts.

2.1.2 Architecture and Data Flow

There is a popular, long-running controversy between the between the data warehousing approaches as taught by Bill Inmon [Inm07, Inm02] and Ralph Kimball [Kim07, KR02]. Basically, Inmon advocates one central data warehouse per company, which stores all data in third normal form. Data marts for individual needs are sourced from this data warehouse. It is a top-down approach, where everything is tightly integrated, and it takes longer to create an initial project. Kimball, on the other hand, suggests a bottom-up approach, where the data warehouse is a conglomerate of the various data marts, and data is stored in the multidimensional model (see Section 2.1.3.1). This approach is faster to deploy and more flexible, but also harder to maintain, and may create redundancies and be hard to integrate.

There are three basic types of data warehouse architectures [JLV$^+$01], of which there may be modified or hybrid forms:

Centralized In a centralized architecture, there is one data warehouse, into which all data is imported and which supplies all analysis clients with data. This is mainly suitable for organizations with centralized operational systems.

Federated In a federated environment, the central data warehouse is virtual, meaning that the data is logically integrated into a central datastore, but stored in separate physical databases. Each analysis site has a data mart which stores the data relevant for the clients of that department, including several levels of detail.

Tiered In a tiered architecture, the central data warehouse is physical, but there exist additional local data marts along several tiers, each of which stores summaries of the data of the preceding tier. The data supplied to clients is not as detailed as in the federated environment.

There may be also different tiers on the source side, where data is integrated not directly but over several steps. Scalability and response time, along with cost, are major parameters when defining the architecture of a data warehouse system.

The integration of data from the sources into the data warehouse – Extract, Transform, Load (ETL) – is a challenging task. Models on the conceptual level are well researched. The models provide a functional [VSS02], static [CGL$^+$98], or dynamic [BFMB99] view of ETL. How conceptual ETL models can be transformed into logical schemata and optimized for performance is still an open topic. [VSG$^+$05] and [SVS05] focus on the modeling and optimizing the logical level, and [Sim05] presents a design method that includes a transformation of conceptual to logical ETL models.

For the approaches presented in the following chapters, the underlying architecture as well as ETL issues are not of primary concern, as the issues are treated on the conceptual – or at most logical – level.

2.1.3 Data Models

Data modeling in general is seen to be structured in three levels: *conceptual*, *logical*, and *physical*. Whereas the meaning of "physical" is quite clear (a description of the system that can be directly implemented on (one kind of) hardware/software platform/base system and/or serve as documentation of such a system), the definitions of "logical" and "conceptual" vary and are even sometimes used interchangeably. "Conceptual" definitely is the top-most level, remote from the target platform. With data models, the Entity-Relationshop model [Che76] can be used for both logical and conceptual modeling, depending on the level of detail and the intent of the modeler.

Conceptual models are typically used at the beginning of the lifecycle of a product, to describe the product and its environment, its actions and interactions with other systems and components on a high level of abstraction. Conceptual models are not meant to lead directly to an implementation, but instead to help clarify ideas, facilitate communication between people from different domains, without anticipating design decisions (e.g., platform).

The focus of this thesis lies on conceptual modeling, as the aim is to describe the relationship between the data warehouse and its environment, and not implementation details.

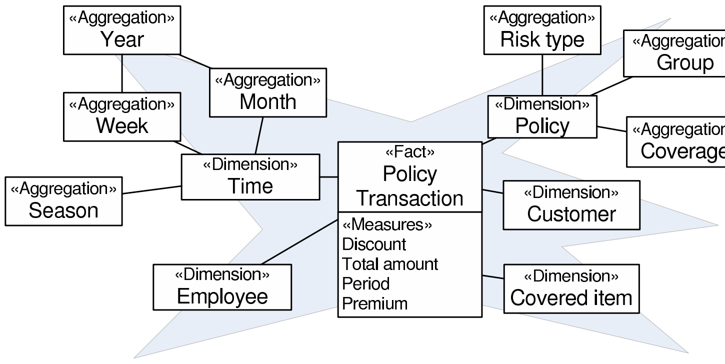


Figure 2.1: Example multi-dimensional model with five dimensions

2.1.3.1 Multidimensional Data Model

Data Warehouse applications involve complex queries on large amounts of data, which are difficult to manage for human analysts. Relational data models "are a disaster for querying because they cannot be understood by users and they cannot be navigated usefully by DBMS software" [KR02]. In Data Warehousing, data is often organized according to the multidimensional paradigm, which allows data access in a way that comes more natural to human analysts. The data is located in n-dimensional space, with the dimensions representing the different ways the data can be viewed and sorted (e.g., according to time, store, customer, product, etc.).

A multidimensional model, also called star schema or fact schema, is basically a relational model in the shape of a star (see Figure 2.1 for an example). At the center of the star there is the *fact* table. It contains data on the subject of analysis (e.g., sales, transactions, repairs, admissions, expenses, etc.). The attributes of the fact table (e.g., cost, revenue, amount, duration, etc.) are called *measures* or *fact attributes*. The spokes/points of the star represent the *dimensions* according to which the data will be analyzed (sorted/aggregated by month, by customer). The dimensions can be organized in hierarchies that are useful for aggregating data (e.g., day, month, year). Stars can share dimensions, thus creating a web of interconnected schemas that makes drill-across operations possible.

There is no standard conceptual data warehouse model, metamodel, reference model, or benchmark, even though a lot of candidates exist. Among the reasons for this are [RALT06]:

- It is not clear what the most relevant multidimensional properties are, both in the research community and in industry.
- Commercial tools focus on drawing logical schemata, thus there is no push from the industry side towards conceptual modeling.

- The models on the conceptual level are richer than on the logical level and cannot be completely transformed into logical schemata.

Multi-dimensional data models are very well researched. Depending on the focus of the authors, most approaches apply different criteria to the “ideal” multi-dimensional model, be it that the model allows several levels of details, supports more relational algebra operations, provides tool support, or has a more “intuitive” notation. In [Riz04], the approaches to data warehouse data modeling are divided into three main areas: extensions of the E/R model, object-oriented models and ad hoc models (not based on an existing paradigm). [VS99] compared 16 so-called logical models for OLAP databases, and divided the academic efforts into cube-oriented models and relational model extensions. [ASS02] and [BSHD98] also provide extensive comparisons of multi-dimensional models.

[ASS02], [LMTS06], and [NTW00] are examples of object-oriented models and/or based on the Unified Modeling Language (UML).

[FK04], [SBHD99], [FS99] and [TBC99] extend the Entity-Relationship Model with specializations for multidimensional modeling.

Models do not have to be based on existing paradigms, there are lots of examples of “ad hoc” multidimensional models available: [GMR98a], [HLV00], [GL97], [LW96], [CT98], and [AGS97].

For purpose of this thesis, the UML-based approach of [LMTS06], first introduced in [TPGS01], is used in Chapters 3, 4, 5, and 6, and [SBHD99] in Chapter 4.

As stated in Section 2.1.2, data warehouses do not necessarily store data in multidimensional models. It is common to have multidimensional data structures on the logical level, and the data actually stored in a relational database in third normal form. Apart from native multidimensional storage and normalized data in a relational database, data may also be stored in object-oriented databases, as XML data, etc. Especially though the increase of web data to be included in data warehouse systems, the importance of semi-structured data and other kinds of storage has increased.

2.1.4 Data Warehouse Requirements

The problems covered in this thesis are partly also encountered in data warehouse requirements analysis. Requirements are difficult to gather, change over time, are not well communicated across organizational boundaries, contain implicit knowledge and must take the available data sources into account. Data Warehouse requirements and design approaches can be roughly classified into two categories [WS03]:

Supply- or Data-Driven In this case, the available data sources are the primary concern. They are analyzed in depth, and their schemata or parts of them are (sometimes semi-automatically) transformed into multidimensional models [HLV00, GMR98b].

Demand- or Requirement-Driven In this case, the information requirements of the future users (i.e., queries) are taken as input for constructing the multi-dimensional models [MTSP05b, PACW06].

There are also hybrid approaches that try to reconcile both approaches [GRG05, CDN$^+$06], and a third idea, namely to use design patterns [JS05].

A strong precondition for successful data warehouse requirements analysis seems to be a common vocabulary between IT people and business decision makers [RALT06]. The models described in the following chapters, and the way business metadata can be derived from models, are a promising step in this direction. Section 6.6 provides an outlook on how a comprehensive requirements analysis approach could grow from these ideas.

2.1.5 Technical and Business Metadata

Metadata is simply "data about data", the contrary usually is called master data, main, core or principal data. [MDC99] defines it as "descriptive information about the structure and meaning of data and of the applications and processes that manipulate data". Data Warehousing and Business Intelligence systems contain large amounts of metadata for describing and managing their analysis data. This includes mainly *technical metadata*, e.g., data types, data access restrictions, data traces, etc. The term *Business Metadata* is used for data that describes the business context of the core data, its purpose, relevance, and potential use [Sar01].

Technical metadata suffers from the great heterogeneity of the data warehouse software in use, which creates a constant need for metadata transformation and integration. There are two industry standards specified by multi-vendor organizations: The Open Information Model (OIM) by the Meta Data Coalition (MDC), and the Common Warehouse Metamodel (CWM) by the OMG (see [VVS00] for a detailed comparison). MDC has contributed to OMG on the CWM as a standard metamodel. The CWM is based on standards such as UML, MOF and XMI (see Section 2.3.2ff.), but has not found general acceptance, neither in research nor industry.

The term "Business Metadata" has been in use for some time and the concept is commonly described as useful and desirable, but detailed approaches or implementations are rare. [Sar01] linked data warehouse business metadata with technical metadata, in order to provide a better context for decision support. Several business metadata categories like goals, organizational elements, processes, events, measures, etc., and a number of desirable characteristics such as evolution of navigation between metadata and data, are defined. The business metadata is described with UML classes and associations and then linked directly to the technical metadata within the same model. The approach only covers the metadata itself and does not use separate conceptual models of the business context.

[BG04] includes a metamodel of business metadata, which is said to be of importance to end users. The abstract core element of the model is called BusinessObject and has a number of specializations:

BusinessTerm A term or word that has a specific meaning to users.

Terminology A (hierarchical) collection of logically related BusinessTerms.

BusinessGoal A goal of a business unit; it may contain a hierarchy of sub-goals.

BusinessFigure A metric for measuring efficiency of processes; it may be part of a metric hierarchy.

BusinessRule There are two types of BusinessRules: *ActionRules* are preconditions for actions to be taken, *InferenceRules* describe domain knowledge.

Report Reports are analyzes or summaries of any kind available to users.

These elements are clearly related to the approach described in Chapter 3. But [BG04] does not supply details on how this business metadata can be gathered or provided to end users.

The Pentaho Open Source BI Project [Pen07c] (see Section 2.1.6.2) has introduced a subproject on metadata, which has had it's first "General Availability" release (a kind of stable release) in October 2007. At the time of writing it is therefore too early to assess the acceptance or development of this project. *Pentaho Metadata* provides an additional layer of abstraction on top of the data model, which consists of a *domain model*, *business models*, and *business views*. The models are comparable to a (simplified) re-mapping of the underlying data model, and can be enriched with any kind of additional metadata properties. Contrary to the "business" vocabulary and names used by Pentaho, the pre-defined properties that are provided with the released version are all technical metadata of different kinds, e.g., display fonts and colors, security values, or aggregation rules. The metadata can be exported in CMW format.

Chapter 3 describes an approach for creating business metadata for Data Warehouses by weaving (see Section 2.3.5) enterprise models (see Section 2.2) with data warehouse data models.

2.1.6 Tools, Products, and Suppliers

2.1.6.1 Closed Source

According to [VM07], the top five vendors of Business Intelligence in the year 2006 were Business Objects (US$ 894 million), SAS (US$ 679 million), Cognos (US$ 622 million), Hyperion/Oracle (US$ 529 million) and Microsoft (US$ 480 million), as shown in Figure 2.2,

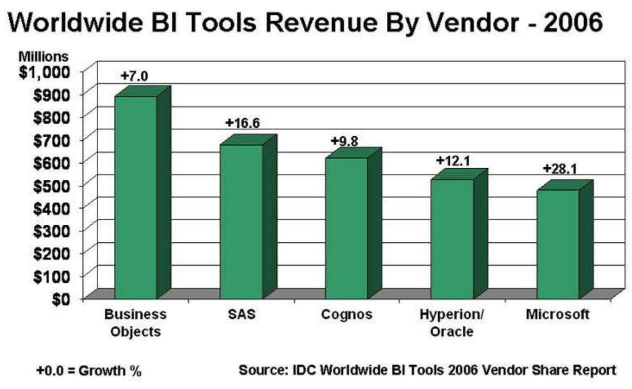


Figure 2.2: The top five BI vendors in 2006, by revenue in US$

Vendor	BI Products
Business Objects	Business Objects XI platform
SAS	SAS Enterprise BI Server, various
Cognos	Cognos 8 Business Intelligence
Hyperion/Oracle	Hyperion System 9
Microsoft	SQL Server, SQL Server Analysis Services, Reporting Services, Excel 2007

Table 2.1: Core products of the top vendors

followed by MicroStrategy, SAP, Oracle, SPSS, Information Builders Inc., Actuate Corp., and IBM. The top five vendors share about 48% of the market, and the top ten come to almost 65%, a value that has been steadily growing during the last years. Table 2.1 lists the main BI products of the top five vendors.

2.1.6.2 Open Source

In a study sponsored by Actuate [Act07], JasperSoft [Jas07c] and Pentaho [Pen07c] that took place between November 2005 and January 2006 [Eve06], 83% of the respondents were at least considering to deploy open source business intelligence software. The projects at the time were mostly small to medium size, but were expected to grow with final deployment (24% were expected to have between 200 and 1000 users, and 37% between 1000 and 20000). The users of the systems that were already deployed at the time were 37% operational users, 19% executives, and 23% senior executives.

Open source business intelligence (OSBI) has become a new buzzword. The vendor-product situation is different from the closed source situation, as the relationship between products, projects, components, companies and foundations is more complex.

For example, two of the most well-known mature BI components, the Mondrian OLAP server ([Pen07a], see Section 7.2.3.3) and JPivot, a tag library for rendering an OLAP table with JSP to HTML, ([JPi07], see Section 7.2.3.1), are used by many projects and commercial products. Mondrian is at the core of at least four successful open source BI suites: Jasper-Analysis [Jas07b], OpenI [Ope07], Pentaho [Pen07c], and SpagoBI [Eng07].

Having learnt from the Linux, J2EE, and database market, many OSBI suppliers offer two versions of their product, a free version and a "professional" version with additional support or advanced features that is sold/licenced for money.

The Eclipse Rich Client platform [Ecl07], which has become popular as a Java IDE, now acts as a kind of common denominator for many open source BI projects. The Business Intelligence and Reporting Tools (BIRT) [BIR07] initiative (lead by Actuate) is a long time Eclipse project and has been part of the large Eclipse releases "Callisto" (2006) and "Europa" (2007), which also gave it increased visibility in the Java/J2EE community. The J2EE stack allows for easy mixing of open source and "conventionally licenced" products, thus providing an easy entry into OSBI components. Many projects use the Eclipse platform to offer wizards, designers or other supporting tools for their platforms. For example, JasperSoft has created the iReport Eclipse plug-in [Jas07a] for adding reporting features to Java applications, and Pentaho has created a cube designer for Mondrian as an Eclipse plugin [Pen07b].

Interoperability and open standards naturally are a major issue for open source BI. MDX (multi-dimensional expressions, see Section 7.2.3.2) and XMLA (XML for Analysis) are the two most wide-spread query languages for OLAP queries. All the larger open source BI components speak MDX, and many also use XMLA.

The prototype for model-based business metadata described in detail in Chapter 7 uses Mondrian and JPivot.

2.1.7 Data Warehouse Context and Relationship to Business Issues

As mentioned previously, the research focus in Data Warehousing lies on database issues [Riz04]. Nevertheless, a small trend towards a broader view on Data Warehouses and BI can be detected in the following two approaches.

In [GRC04], the authors link BI with Business Performance Management (BPM). They describe a business monitoring architecture, where a tightly integrated "right time" infrastructure feeds users in the business domain with information about business processes. The envisioned architecture includes indicators and rules that are derived from and describe the business domain. The data warehousing and business intelligence systems are described together with their business process context.

The Pentaho Project Open Source [Pen07d] aims at redefining business intelligence as a process-centric. The architecture requires every BI component to have the ability to be included in a (business) process (e.g., by being called as a web service). The process-centric approach includes logging, auditing, and scheduling features. From this viewpoint, the edges between BI and operational processes become blurred. When using this framework throughout the whole lifecycle of the application it becomes necessary to acquire a much broader view of the organization in which the BI system is used.

2.2 Enterprise Models

An enterprise model formally represents the basic building blocks of an organization, its structure, behavior and goals. It is usually organized into several aspects that can be modeled individually but also related to each other [WRK01]. The Architecture of Integrated Information Systems (ARIS) [Sch99] is a typical example for such an enterprise model. Other similar approaches include CIMOSA [KV92] and MEMO [Fra02].

The term "Enterprise Model" is used in this chapter in a much wider sense than commonly in Databases and Data Warehousing, where the term often denotes Enterprise *Data* Models. Other similar or largely equal concepts include "Enterprise Architecture", "Enterprise Ontology", "Business Model", and "Reference Model".

Figure 2.3 shows the outline of a generic enterprise model, organized into five aspects: The enterprise strives to achieve goals, acts through processes, has an organizational structure, produces products and uses software applications. In the enterprise model, an organization chart can be used to describe the organizational structure, i.e., the dependencies between the departments, groups and roles that exist within the organization. Similarly, business process models describe the structure of business processes with control flows, inputs and outputs. The products, applications, and strategic goals can also be modeled separately, as well as connected to the other aspects in a single model. Such an overview model can connect all models to show for example how processes fulfill goals, are performed by organizational roles, fall into the responsibility of departments, and use applications to produce products for other departments.

The aspects of an enterprise model are often described on several levels (see below in Section 2.2.1), from a conceptual description via intermediate steps down to implementation or close to implementation of an information system that is well-aligned with the enterprise. The critical issues are how well the enterprise model can really describe the enterprise organization, and how the information can be transformed from level to level in such a way that no relevant information lost and building such an information system is feasible.

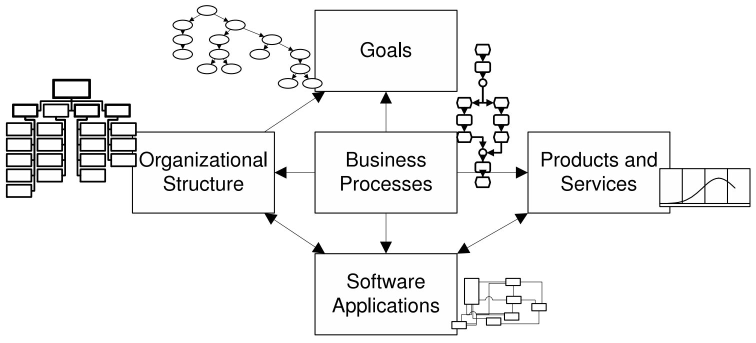


Figure 2.3: Example of a generic enterprise model

2.2.1 Enterprise Modeling Frameworks

There are many kinds of frameworks for Enterprise Modeling. The following examples are intended to illustrate the similarities and differences.

Zachman Framework The Zachman Framework [fFAZ07] consists of a 6 x 6 matrix. The columns represent *questions* or basic modeling issues: What (Data), How (Function), Where (Network), Who (People), When (Time), Why (Motivation), whereas the rows provide the *stakeholders*: Visionary/Planner (Scope Context Boundary), Owner (Business Model Concepts), Designer (System Model Logic), Builder (Technology Model Physics), Implementer (Component Configuration), and Worker (Functioning Enterprise Instances). For the What/Data column, the entries in the matrix from top to bottom are: General list of concepts, entity-relationship model (conceptual/semantic), logical data model, physical data model, data definition, data.

Architecture of Integrated Information System (ARIS) The ARIS concept [Sch99] consists of four main *views* or *perspectives*: Organization, Data and Function group around the central Control view. Each view is subdivided into three *levels*, “Fachkozept” (conceptual), “DV-Konzept” (logical), Implementation (physical). Each perspective can be modeled separately. Elements of the three supporting perspectives can then be integrated in the Control perspective (i.e., the functions are connected to their input or output data and assigned a responsible role).

Computer Integrated Manufacturing Open System Architecture (CIMOSA) CIMOSA [KV92] has three dimensions: *Views*, *building blocks*, and *modeling levels*. The views are Function, Information, Resource and Organization, and the modeling levels called are Requirements Definition, Design Specification, and Implementation Description. The building blocks consist of a general, a partial, and a particular level.

Multi-Perspective Enterprise Modelling (MEMO) The MEMO framework [Fra02] is organized in four aspects: resource, structure, process, and goal, and three interrelated perspectives: strategy, organization, and information system.

2.2.2 Basic Models

Frameworks for enterprise modeling use different kinds of domain models for describing their individual components.

2.2.2.1 Business Processes and Workflows

Business process modeling languages are used to describe different aspects of processes, such as the layout and control flow of the sub-processes and functions, involvement of organizational units (who does what, who is responsible, who provides data, etc.), alternatives and error handling, intermediate and final outputs/products, etc. Often, there are several levels of models involved, for different levels of detail and/or abstraction. For a detailed overview and comparison of current business process modeling languages, see [LK06].

Event-Driven Process Chain (EPC) EPCs [KNS92] were developed in 1992 at the Institute for Information Systems of the University of Saarland, Germany, in collaboration with SAP AG. It is the key component of SAP R/3's modeling concept for business engineering and customizing. The EPC is based on the concepts of stochastic networks and Petri nets. EPCs are used in the control view of ARIS (see above in Section 2.2.1). An EPC represents the control flow structure of the process as a chain of alternating events and functions. It integrates the design results of the different views: Functions, events, information resources, and organization units are connected into a common context by the control flow. The resulting model is the EPC. EPCs are used in Section 4.2.

UML Activity Diagram UML Activity Diagrams [OMG05c] originate in software control flow models, but are increasingly popular for business process models, also due to the wide availability of UML modeling tools and knowhow. Specific business process elements are not provided by the UML specification, but can easily be added via profiles and stereotypes (see Section 2.3.2). Section 4.3 describes a UML profile for BI that extends activity diagrams.

Other Other popular models used for business process modeling include the Business Process Definition Metamodel (BPDM [OMG04], a UML Profile specified by the OMG [OMG07]), the Business Process Modeling Notation (BPMN [BPM04], similar to UML activity diagrams), the Integration Definition Method 3 (IDEF3, [MMd$^+$95]), Petri Nets [Pet62, Pet66], and Role Activity Diagrams (RAD [HRG83]).

Chapter 4 links data warehouse models with business processes models.

2.2.2.2 Organizational Structure

Enterprise modeling frameworks typically include a model of the structural organization. Such models are often similar to or based on Entity-Relationship models [Che76] or UML class diagrams [OMG05c], meaning that they model relationships between entity types with binary or n-ary relationships and cardinalities, but do not require many very specific elements or control structures. For example, the organization chart used in the Organization perspective of ARIS places *organizational roles* and *organizational units* (i.e., departments) in a simple, usually hierarchical structure.

2.2.2.3 Goals and Strategy

A core part of every enterprise model is the goal model. "Increase market share" or "reduce operating costs" are typical enterprise goals. Goals form the basis for business decisions and the way a company does business. What is relevant and important for business performance measurement can be read directly from the enterprise goal model. They govern the design of business processes and the way the organization behaves. Nevertheless, a goal model is basically very simple, and enterprise goals are long term goals that should remain stable a lot longer than business processes, role definitions, and operating rules. Therefore, they provide excellent metadata for a data warehouse (see Chapter 3).

For enterprise goals in particular, there is often a distinction between three levels of goals: strategical, tactical and operational. In order to be able to transform high level enterprise goals of the strategic level via tactical level goals to every-day operational goals, a goal is decomposed via a causal transformation or operationalization into one or more subgoals, which in turn can be further decomposed, thus creating a hierarchy (cf. [LK97]).

In the Goal Question Metric approach [BCR94], originally aimed at software quality improvement, measurement and evaluation is based on a three-level hierarchy in which the goals (of an organization) form the first, the conceptual level. Goals are the starting point of all measurement activities and provide the organizational context according to which the measurement data can be interpreted.

Different kinds of goals, including enterprise goals, are often used in software engineering for requirements elicitation. For example, the *i** methodology [Yu97] provides an agent- and intention-oriented way to analyze requirements for software (and other) systems. The focus of i* is on interaction between autonomous agents, their actions and strategies.

2.2.3 Metrics and Enterprise Performance Measurement

In any competitive environment, it is essential for individuals to know where they stand in comparison to their competitors. Companies must continually strive to stay ahead of the competition [LK03]. Only by measuring and comparing the performance it is possible to identify problems and potential improvements. [KN92] postulate that the implementation of a measurement system in an organization suffices to influence the behavior of the people involved.

It is generally accepted that measurement should be as holistic as possible, covering many different aspects and perspective of the enterprise organization [Kit96, Kue00]. Typical examples of such performance measurement approaches include the Balanced Scorecard (BSC [KN92]) and the approach for Stakeholder-Driven Performance Measurement by [Kue00].

Using Data Warehouses and Business Intelligence (see Section 2.1) is one way for an enterprise organization to collect and analyze performance measurement data. Identifying suitable metrics/measures remains a challenging task.

2.3 Model Engineering

The following chapters employ modeling techniques for various purposes. This section aims to give an overview over the concepts used.

2.3.1 Models and Metamodels

Modern modeling languages are structured in several levels of models or metamodels. Such a meta-architecture can be structured in different ways, but basically every level describes the elements that are available to the next-lower level [HKKR05]. The elements of a lower-level model are said to *instantiate* or *conform to* the corresponding higher-level (meta-) model elements.

The approaches presented in the following chapters of this thesis focus on the model and metamodel level. New model types are created by extending existing or defining new metamodels. The models that correspond to these metamodels then allow to describe aspects of the relationship between Data Warehouses and the structure, behavior or goals of the organization in a way that is not possible with existing models.

2.3.2 Unified Modeling Language (UML)

The Unified Modeling Language (UML) [OMG05c] is specified by the Object Management Group (OMG) [OMG07]. Together with the Meta Object Facility (MOF) ([OMG03a], also a specification by the OMG[1]) it forms the basis for the Model-Driven Architecture (MDA, see

[1]MOF is the metamodel of the UML metamodel

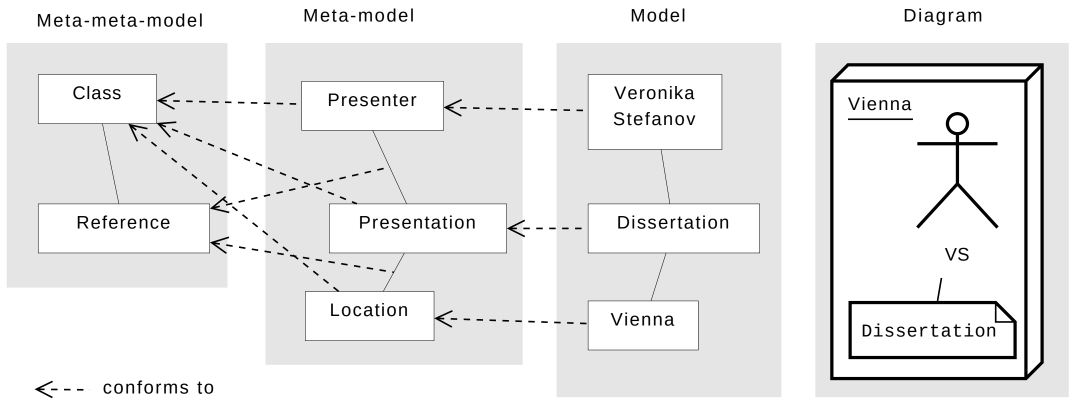


Figure 2.4: Corresponding model levels with an example diagram (not UML)

below).

UML is a multi-purpose modeling language, that has its foundation in object-oriented software modeling. It provides a number of diagrams for different purposes (e.g., use cases, structure, intra-object behavior, inter-object-behavior, deployment of software artifacts, etc.). The diagrams are comparable to different views on a common model, i.e., the same model element may appear in different diagrams [RJB04].

The UML 2.0 metamodel has a fine-grained structure that contains [HKKR05]:

12 language modules Structural models (e.g., the class diagram), behavioral models (e.g., the activity diagram), and the language extension mechanism.

2 syntax levels Abstract (model elements) and concrete (their notation) syntax.

4 conformity levels If a tool conforms to a level, it supports all elements of that level: from foundation (only object-oriented structural modeling) via basic and intermediate to complete.

To allow modelers to create more specific model elements for their domains, UML contains an extension mechanism, the so-called profiles and stereotypes. Profiles are a *lightweight* extension mechanism, which is inherent to the language and allows for continued interoperability of the extended models with different modeling tools. Chapters 4, 5, and 6 include UML profiles.

The Object Constraint Language (OCL) [OMG05b] is a declarative language commonly used for adding conditions or rules to UML models. It can be used for querying any MOF-based model and is used by model transformation languages such as ATL [ATL07], as well as the QVT standard [OMG05a]. The UML profiles in Chapters 4, 5, and 6 use OCL constraints.

2.3.3 Model-Driven Architecture (MDA)

The OMG's Model-Driven Architecture (MDA) [OMG03b] is an approach to software design that uses standards such as UML and MOF. In MDA, humans create software artifacts by designing high-level models which are then transformed via several intermediate models to deployable program code. A platform-independent model (PIM) is transformed to a platform-specific model (PSM) with the help of a platform model. A platform in this case may be anything from an operating system to a programming language, application framework or vendor, which makes several PIM-PSM distinctions possible for the same models. Another notable model is the computation independent model (CIM), which is intended to be a kind of domain or business model, used next to the PIM, but not necessarily transformable.

The modeling approaches described in the following chapters do not use MDA directly, but are related to it. Some of the techniques and tools used, e.g., EMF (see Section 2.3.4) and model weaving (see Section 2.3.5), originate from MDA.

2.3.4 Eclipse Modeling Framework (EMF)

The Eclipse Modeling Framework (EMF) [EMF07] is an open-source framework based on the Eclipse platform [Ecl07] that offers features such as code generation from model specification, model editor generation, and a persistence API for XML/XMI[2] (de)serialization of model instances. The EMF metamodel is called Ecore and conforms to EMOF (Essential MOF, as opposed to CMOF, Complete MOF). Ecore models are similar to UML class diagrams in their functionality (object-oriented structural models).

EMF is used by many other Eclipse projects (e.g., WTP [WTP07], GMF [GMF07], BIRT [BIR07], TPTP [TPT07]) and commercial tools (IBM [MDG$^+$07], Borland [Bor06], Omondo [Omo07], ...) and enjoys a large and active community.

2.3.5 Model Weaving

For many modeling purposes, one large general "one size fits all" model is not advisable [BB02]. Creating several smaller, specific domain models allows for a better separation of concerns. Separate issues can be expressed in separate domain models, which are then related to each other by model weaving [dFBJ$^+$05] as a linking mechanism.

Model weaving is an operation that links two or more (meta-)models [BJT05, BJRV05]. The result is a weaving model containing links between elements from the involved models, as shown in Figure 2.5. The links may contain additional mapping information such as calculation formulas. Weaving models are models of the relationships between (other) models. They allow for easier handling of complex relationships between models. Weaving

[2]XML Metadata Interchange [OMG05d]

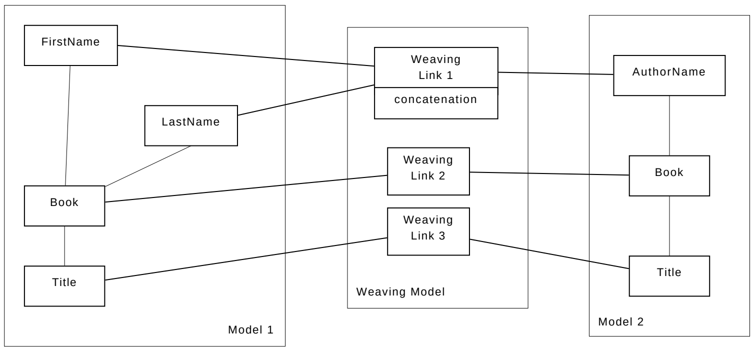


Figure 2.5: A waeving model

does not imply model or data transformation, but may be a prerequisite to these tasks. A weaving model is a "normal model" that can be stored and edited, accessed and analyzed with modeling tools. Among the application areas of model weaving are database schema matching, model transformation specification, visual programming, and ontology mapping [BJV04].

Model weaving superficially resembles techniques used in ETL or EAI, but the intention behind it is different. A weaving link does not necessarily imply that the two elements it connects should in some way be transformed from one into the other. Rather, it simply indicates that the two elements share some semantic link, e.g., "lies in the responsibility of", "is measured by", "affects", etc. Still, weaving can of course be used as a preliminary step to transformation, by indicating transformation sources and targets and then using the weaving model as an input for a transformation language. In the approach presented in Chapter 3 it is employed for annotating the data warehouse data with business metadata. This is done by linking different kinds of domain models (enterprise models and data warehouse data models), and therefore does not imply transformation.

Model weaving was chosen because it offers a number of advantages: By adhering to the "Everything is a model" principle [Béz05], it is possible to capture practically *all* information in terms of models, also the relationships and correspondences between models. This makes it possible to store, analyze and edit the links with the same modeling tools as the models themselves. Weaving avoids having one large metamodel "for everything", but instead keeps the individual metamodels separate, easy to handle and focused on their domains, while at the same time they are interconnected into a "lattice of metamodels" [Béz05].

Advanced modeling tools such as the ATLAS Model Weaver [AWM07, dFBJ$^+$05] (available as an Eclipse [Ecl07] plug-in) support model weaving. Weaving works best if the participating (meta-)models are based on the same (meta-)metamodel. The metamodels used in Chapter 3 are compliant to the Meta Object Facility (MOF) [OMG03a].

Chapter 3

Model-Based Business Metadata

Data in a data warehouse describes events and statuses of business processes, products and services, goals and organizational units, and generally mirrors every aspect of the structure and behavior of the organization. Business performance is judged regarding the achievement of goals. Business users are accustomed to their own vocabularies and concepts, and data interpretation is greatly improved by knowledge about context.

Surprisingly, information about the relationship between the data warehouse data and the organization is not available to the data warehouse users or even recorded in a suitable way. This chapter presents an approach that uses enterprise models and modeling techniques to record the at present mainly implicit knowledge about this relationship.

Contrary to the approaches presented in Chapters 4, 5, and 6, this chapter does not introduce new conceptual models and diagrams, but uses modeling techniques to link existing models, in order to derive business metadata, an additional level of abstraction on top of the data-oriented data warehouse structure. The applicability is illustrated by the prototype described in Chapter 7.

Contents

3.1 Introduction

Data Warehouse systems represent a single source of information for analyzing the status, the behavior, the development and results of an organization [LM04, KRRT98]. By describing events and statuses of business processes, products and services, goals or organizational units, the data in the data warehouse mirrors the organization. Information about this relationship between the data warehouse data and the business processes, products, etc. is usually available in the organization in the form of enterprise models and documents, and is used during the design phase of the data warehouse.

To define and analyze their objectives and goals, enterprise organizations use different types of goals models (see Section 3.2.2). Metrics are derived to measure the achievement of the goals. There exist many modeling approaches for enterprise goals as well as for metrics.

Surprisingly, the knowledge about this relationship is not made available to the data warehouse users or even recorded in a suitable way. Due to the data-oriented nature of Data Warehousing, the knowledge of how the data warehouse measures relate to enterprise goals, business processes or products is not easily accessible to data warehouse users. As this is mainly implicit knowledge, it is also more likely to be lost or forgotten.

Business users are accustomed to their own vocabularies and concepts, and data interpretation is greatly improved by knowledge of context. Using and understanding traditional data-oriented Data Warehousing frontends therefore requires additional effort from the users. If knowledge about the business context is left to chance, data analysis is bound to miss important points and becomes more errorprone.

The problem addresses here in this chapter is that these two aspects - the data warehouse and the enterprise organization - are described separately and not related to each other. There is a need for combining these two aspects. If their relationship is made explicit, it can be used to enhance the way users access and interpret data in the data warehouse.

There is a need for describing the relationship between the data in the data warehouse and the organization that surrounds it. This chapter proposes an approach that allows to show:

- How the enterprise goals and metrics are mirrored in the data warehouse data model
- Which parts of the data warehouse data is created by which business process or part of it
- How business processes have impact on the values of the data warehouse data
- Which parts of the data warehouse data measures which (sub)process or products
- Which organizational units and roles are measured along with the processes
- Where the products and deliverables of the processes are found in the data warehouse data structure

Indeed, adding context and background information to a data warehouse has been an open question in Data Warehousing for years. The term *Business Metadata* is used for data that describes the business context of the core data, its purpose, relevance, and potential use [Sar01]. There is general agreement on the usefulness and desirability of business metadata [BG04, Sar01]. But how to create or derive business metadata is still very much an open question.

Today's data warehouses provide their users with very powerful tools for analyzing large amounts of data, but supporting the actual data interpretation by decision makers, i.e., with business metadata, has not yet been a primary concern in research.

Models for data warehouse data structures are well researched and established (see Sect. 3.2.1). There exist numerous domain-specific modeling approaches to describe the data structures on the conceptual, logical and physical level. Enterprise organizations using a data warehouse are aware of these models and rely on them to design and describe their data warehouse.

The approach presented in this chapter makes use of the knowledge already available in the organization in enterprise (meta-) models to derive business metadata. Model weaving is used to store and manage the relationships between the data warehouse data model and a model describing the enterprise organization. The business metadata provided via the weaving model helps to improve the understanding and interpretation of the data warehouse data by the users. It basically creates an additional level of abstraction on top of the data warehouse data, which is aligned to the concepts that are well-known to the users, e.g., a goal- or business process-oriented view. *Model-based* – as opposed to *model-driven* – means that the approach presented here uses models, modeling techniques, and modeling tools for creating, storing, managing and editing the metadata.

The approach provides the following contributions:

- It makes the implicit relationships between the data in the data warehouse and the structure, the behavior and the goals of the enterprise organization visible and accessible.
- By relating the measures in the data warehouse to enterprise goals and organizational concepts, users are better able to interpret the performance of the enterprise, and to understand the implications.
- Creating the weaving links is a comparatively small investment for valuable metadata that gives meaning to data warehouse measures.
- Data warehouse requirements analysis and (re-)design are notoriously challenging tasks, because the business context of a data warehouse is difficult to extract from user interviews and practically impossible to store directly in the multidimensional data

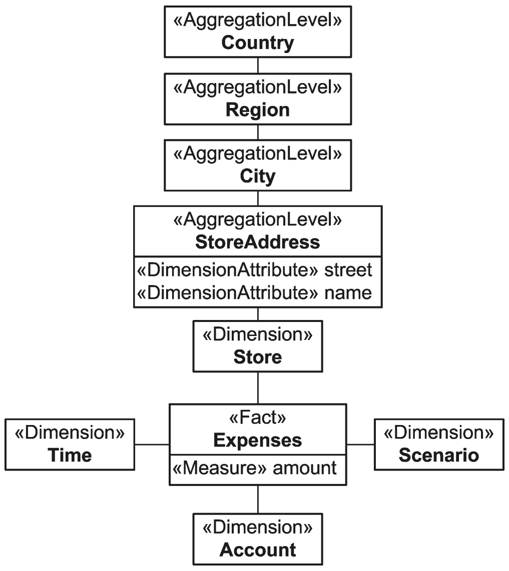

«AggregationLevel» Country
«AggregationLevel» Region
«AggregationLevel» City
«AggregationLevel» StoreAddress
«DimensionAttribute» street
«DimensionAttribute» name
«Dimension» Store
«Dimension» Time
«Fact» Expenses
«Measure» amount
«Dimension» Scenario
«Dimension» Account

Figure 3.1: Example multi-dimensional model in UML notation

structures. Weaving enterprise models with data models makes context information accessible, and does so without disrupting the involved models.

- As a by-product, the weaving model can be used for model validation, as is identifies missing or superfluous tables and measures in the data warehouse, as well as omissions in the enterprise model, and the link to enterprise goals and business strategy can help to evaluate data warehouse investments, and to justify them.

This chapter is structured as follows: The next section gives an overview over the multi-dimensional data model, enterprise models, and the concept of model weaving. Section 3.3 introduces the weaving model for business metadata based on enterprise goals, followed by a more general model with business processes, products, and organizational units in Section 3.4.

Parts of this chapter have been published in [SL06] and [SL07c].

3.2 Background

3.2.1 Multidimensional Data Models

The main data model in Data Warehousing is the multidimensional model, also called star schema [CD97]. Figure 3.1 shows an example. For details on multidimensional modeling, see Section 2.1.3.1.

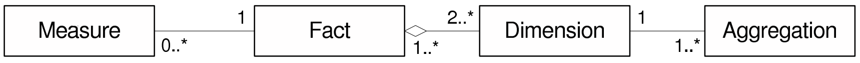


Figure 3.2: Core metamode elements for multi-dimensional modeling (cf. [LMTS06])

For the purpose of deriving business metadata, a data model that supports weaving is necessary. This chapter chooses the object-oriented approach of [LMTS06], first presented in [TPGS01] and then further developed to a UML profile in [LMTS02] and [LMTS06].

A UML profile is a domain-specific extension to the UML modeling language [OMG05c]. The profile used here adapts the UML class diagram for multi-dimensional modeling, i.e., the base class of the stereotypes is *Class*. It allows to create detailed models of the conceptual characteristics of multidimensional data models. Figure 3.2 shows the main elements of the UML profile and their relationships as a metamodel. It allows to model any number of *Fact* tables. Each fact table can have any number of optional *Measures* and must have at least two *Dimensions* connected to it, at least one of which is usually a *Time* dimensions. Dimensions may be shared between facts and have one or more *Aggregation Levels*, which form the aggregation hierarchy.

There is no universally accepted, generic metamodel for multi-dimensional modeling. This fhapter uses the metamodel shown in Figure 3.2 in lieu of generic metamodel. The corresponding model in this case is a UML model. Yet, for this approach, the data model does not necessarily have to be a UML model. The only prerequisite for the data model is that its metamodel is available, and that it allows to model facts, dimensions and measures.

The Expenses fact (Fig. 3.1) has four dimensions: *Time*, *Account* (e.g., IT, Marketing, etc.), *Scenario* (e.g., Actual, Forcast) and *Store*. The levels of the dimensions are only shown for the store dimension. Each entry in the fact table contains information about a single expense incurred. In this example there is only one measure that can be analyzed for each expense: the *amount*. Aggregations such as "total value of an account in all stores in one year" become possible by selecting aggregation levels from the dimensions. Several such facts can be connected by sharing the same dimensions, creating a more complex multi-cube model.

See [LMTS06] for the additional, more detailed elements not shown here, such as the attributes of the dimensions, as well as logical constraints on the model elements. They are defined in the Object Constraint Language (OCL) [OMG05b] and cover aspects of multidimensional data models such as ensuring that the classification hierarchies have the form of directed acyclic graphs, as well as additivity rules, derived measures, and degenerate dimensions.

3.2.2 Enterprise Models

An enterprise model formally represents the basic building blocks of an organization, its structure, behavior and goals. It is usually organized into several aspects that can be mod-

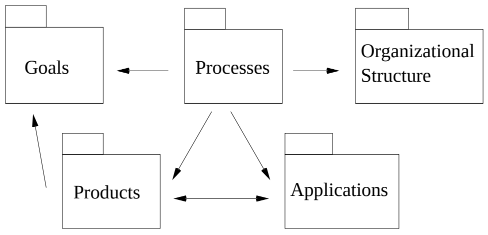


Figure 3.3: Example of a generic enterprise model

eled individually but also related to each other [WRK01]. The Architecture of Integrated Information Systems (ARIS) [Sch99] is a typical example for such an enterprise model. Other similar approaches include CIMOSA [KV92] and MEMO [Fra02]. Thus, the term "Enterprise Model" is used in this chapter in a much wider sense than commonly in Databases and Data Warehousing, where the term often denotes Enterprise *Data* Models.

Figure 3.3 shows the outline of a generic enterprise model, organized into five aspects: The enterprise strives to achieve goals, acts through processes, has an organizational structure, produces products and uses software applications. For a more detailed description of Enterprise Modeling, see Section 2.2.

All aspects of the enterprise model are related to the data in the data warehouse, because data in the data warehouse is based on and mirrors the structure and behavior of the enterprise. Enterprise models are therefore used in this approach for business metadata.

3.2.3 Model Weaving

In order to gain business metadata, it is necessary to link different kinds of domain models (enterprise models and data warehouse data models). For this task, the technique of model weaving [BJRV05], as described in Section 2.3.5, is used in this chapter.

Model weaving superficially resembles techniques used in ETL or EAI, but the intention behind it is different. A weaving link simply indicates that the two elements share some semantic link, e.g., "lies in the responsibility of", "is measured by", "affects", etc. In this chapter weaving is employed for annotating the data warehouse data with business metadata, and therefore does not imply transformation.

Advanced modeling tools such as the ATLAS Model Weaver [AWM07, dFBJ$^+$05] (available as an Eclipse [Ecl07] plug-in) support model weaving. Weaving works best if the participating (meta-)models are based on the same (meta-)metamodel. The metamodels in this chapter are compliant to the Meta Object Facility (MOF) [OMG03a].

3.3 Business Metadata from Enterprise Goals and Metrics

An enterprise model (Sect. 3.2.2) usually serves as a single point of information in the organization. From the enterprise model, this section first uses enterprise goals as the basis for the business metadata, because goals form the basis for business decisions and the way a company does business. What is relevant and important for business performance and is therefore required from the data warehouse can be read directly from the enterprise goal model.

3.3.1 Enterprise Goals

A core part of every enterprise model is the goal model. "Increase market share" or "reduce operating costs" are typical enterprise goals. Goals form the basis for business decisions and the way a company does business. What is relevant and important for business performance measurement can be read directly from the enterprise goal model. They govern the design of business processes and the way the organization behaves. Nevertheless, a goal model is basically very simple, and enterprise goals are long term goals that should remain stable a lot longer than business processes, role definitions, and operating rules. Therefore, they provide excellent metadata for a data warehouse.

Based on the description of the goals, the enterprise derives metrics that measure the level of achievement of the goals and indicate the performance of the enterprise. These metrics are not identical but closely related to the measures in the data warehouse. In the early 1990s, business goals and strategy experienced a revival in theory and practice with approaches like the Balanced Scorecard (BSC) [KN92]. The BSC's focus is on vision and strategy. Companies define goals from various perspectives, which are then translated into measures. The BSC does not mention the behavior which will lead to the fulfillment of a goal. Rather, people are assumed to be guided by the measures they have to fulfill. Measures and not the desired operations are communicated to the employees. The goals and measures give the long-term focus, while the conditions under which people operate are constantly changing.

In the Goal Question Metric approach [BCR94], originally aimed at software quality improvement, measurement and evaluation is based on a three-level hierarchy in which the goals (of an organization) form the first, the conceptual level. Goals are the starting point of all measurement activities and provide the organizational context according to which the measurement data can be interpreted.

Different kinds of goals, including enterprise goals, are often used in software engineering for requirements elicitation. For example, the *i** Methodology [Yu97] provides an agent- and intention-oriented way to analyze requirements for software (and other) systems. The focus of i* is on interaction between autonomous agents, their actions and strategies.

For enterprise goals in particular, there is often a distinction between three levels of goals: strategical, tactical and operational. In order to be able to transform high level enterprise goals of the strategic level via tactical level goals to every-day operational goals, a goal is decomposed via a *causal transformation* or *operationalization* into one or more subgoals, which in turn can be further decomposed, thus creating a hierarchy (cf. [LK97]).

DWH development projects also have goals (timeliness, cost, etc.), as does the data warehouse itself (data accuracy, availability, response time, etc., cf. [JLV$^+$01]). These goals are not used here, because providing knowledge about these goals in the form of business metadata would not improve data interpretation.

3.3.2 Enterprise Goal Metamodel (EGM)

The Enterprise Goal Metamodel presented here incorporates features from a number of existing goal modeling approaches (cf. [KN92, LK97, Kav04, KL04, Yu97]) It is aimed at providing a sufficiently detailed and comprehensive, yet concise model of the main concepts that are needed to model the context of a data warehouse.

Figure 3.4 shows the Enterprise Goal Metamodel (EGM). For sake of clarity and readability, it is shown as two separate graphics: Figure 3.4(a) explains goal decomposition hierarchies and relationships between goals and Figure 3.4(b) shows the other elements of the metamodel related to goals. The model uses the notation of the UML 2.0 class diagram [OMG05c].

Figure 3.5 shows example goals for Figure 3.4(a). In the EGM, a *Goal* may participate in a goal hierarchy via a *Goal Decomposition*. The goal decomposition connects a higher-level goal with a number of lower-level subgoals. A goal may have only one set of subgoals but may participate itself as a subgoal in more than one goal hierarchy. Therefore it can be related to only one goal decomposition in the role of a *satisfied* goal but to many in the role of a *satisfier* goal. The goals "reduce out-of-stock" and "increase freshness" in Figure 3.5 are subgoals of "satisfy customers". From the viewpoint of the "AND" goal decomposition in Figure 3.5, the four lower-level goals are satisfier goals, and "satisfy customers" is the satisfied goal. The type of a goal decomposition is either *AND* or *OR*, depending on whether all or only some of the subgoals have to be satisfied to satisfy the upper level goal.

Orthogonally to the goal hierarchy, goals can be seen to *influence* each other in various ways. The fulfillment of one goal might be detrimental to another goal, or the goals may be related to each other in such a way that if one of them is satisfied, this also supports the other goal. Therefore, there are two influencing relationships between goals: *support* and *conflict*. Both may occur between any number of goals, e.g., a goal can support several goals and conflict with others at the same time.

Figure 3.4(b) (illustrated with values in Table 3.1) shows that for each *Metric* assigned to a goal it is necessary to define a *Target Value*. Because target values usually change over time,

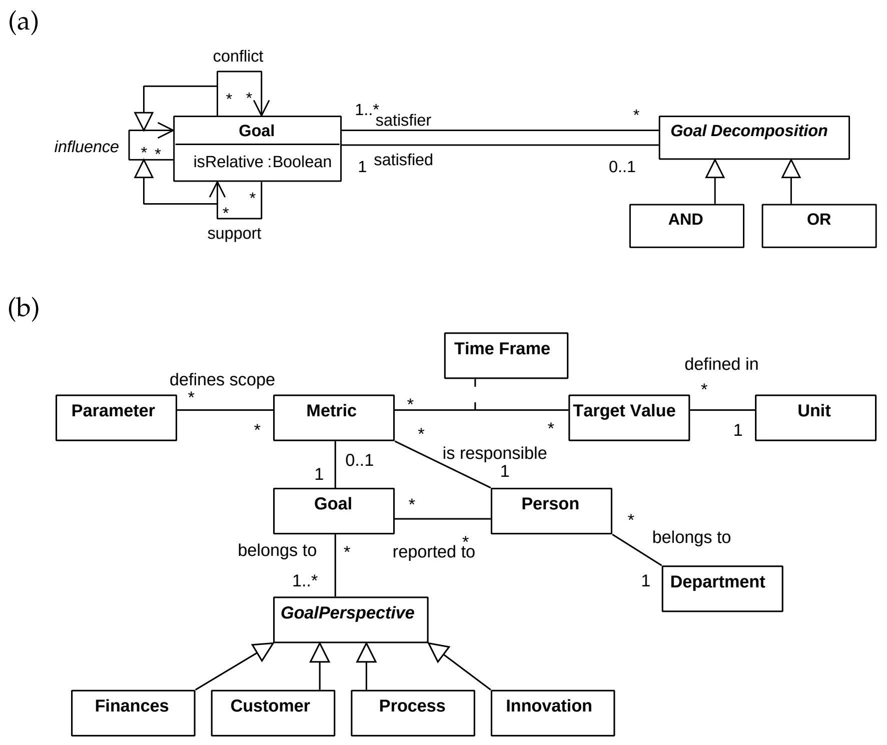


Figure 3.4: Metamodel of the enterprise goal model

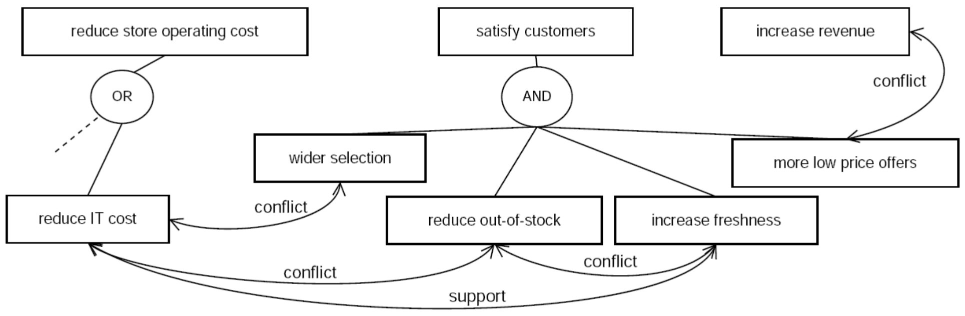


Figure 3.5: Sample goal hierarchy model corresponding to the metamodel in Figure 3.4(a)

Table 3.1: Details for the example goals from Figure 3.5, as defined in Figure 3.4(b)

Name	Value
Goal name	reduce “out of stock”
Goal perspective	Customer
Reported to + contact info	Sales Dept., Ms. Baker, Tel. 5800193
Metric name	number of days in month with no items on stock
Metric target value + unit	< 1%
Metric responsible + contact info	Mr. Jones, Tel. 5800234
Conflicting/supporting goals	conflicts with “increase freshness” and “reduce IT cost”
Goal this goal satisfies	“satisfy customers
Goal name	increase freshness
Goal perspective	Customer
Reported to + contact info	Mr. Groop, groop@dpt.company.com, Ms. Fitt, Tel. 5800156
Metric	avg. time in warehouse for product group “fresh goods”
Metric target value + unit	< 8 hours
Metric parameter(s)	Warehouse, product group
Metric responsible + contact info	Mr. Stephens, Tel. 5800655
Conflicting/supporting goals	conflicts with “reduce out of stock”, supports “reduce IT cost”
Goal this goal satisfies	“satisfy customers”

a *Time Frame* for each combination of metric and target value is necessary. Goals can be *relative* goals (“increase the value by x”), or absolute goals (“the value should be x”), indicated by the attribute *isRelative* (shown in Figure 3.4(a)). This influences the semantics of the time-frame: For a relative goal and its metric, it means that the change is to be achieved during this time, whereas for an absolute goal it indicates the validity period of the target value. A metric can be constrained with *Parameters*, which define the scope: The goal “reduce inventory cost” has none, “reduce inventory cost of top-20 products” has one and “reduce inventory cost of top-20 products in region x” has two parameters. Goals can be *reported to* a *Person* belonging to a *Department*. For each metric there has to be a *responsible* Person. To indicate the general focus of the goals, they are assigned to *Perspectives*. These perspectives are generic and can be adapted to the analysis needs of the company. Figure 3.4(b) shows four perspectives according to the Balanced Scorecard. Person and Department are part of the organizational aspect.

3.3.3 Weaving Model

In order to gain business metadata, the enterprise goals have to be linked to the data warehouse data. Figure 3.6 shows the weaving model linking the data (right) and enterprise goal (left) metamodels presented in Sect. 3.2.1 and 3.3.2. It consists of three links: two binary and

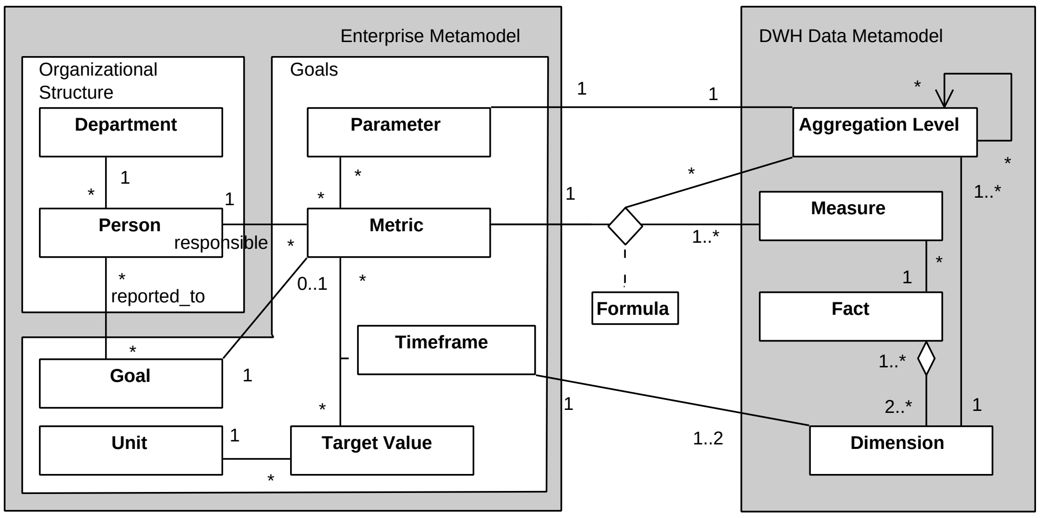


Figure 3.6: Weaving model connecting the goal model with the data model

one ternary link.

The first link is between the Parameter describing the focus or scope of the metric (e.g., Region is the parameter when a value is given by *by region*) in the EGM and an Aggregation Level (e.g., per Month on the Time Dimension or per Region on the Store dimension). These are similar concepts, which can be easily mapped. The corresponding dimension to a level is provided by the data model.

Weaving links can connect more than just one element on either side. The second, most complicated link in this weaving model connects a Metric with a Measure and optional Aggregation Levels. A metric roughly corresponds to a fact attribute. The fact attribute itself is not aggregated, but the metric can be restricted by parameters: This has to be indicated on the data model side by adding the corresponding aggregation base(s) to the link. Additionally, because the fact attribute itself contains only the absolute value of a variable, while the metric related to it might contain an average, a percentage, or a rate of change, this weaving link can contain a formula (e.g., *(IT cost/total cost)*100* for the *percentage of IT costs*).

Finally, the third link allows us to handle the relationships between the Timeframe of a metric's target value and a Dimension containing time values in the data model[1]. A timeframe is a time period, indicated by start and end point, whereas a time dimension contains single points in time. Therefore the weaving link connects one timeframe with two points

[1]This can be ensured by an OCL [OMG05b] constraint, e.g.,
```
Context TimeFrame
inv:  self.dimensions-> forAll(d|d.isTime = true)
```

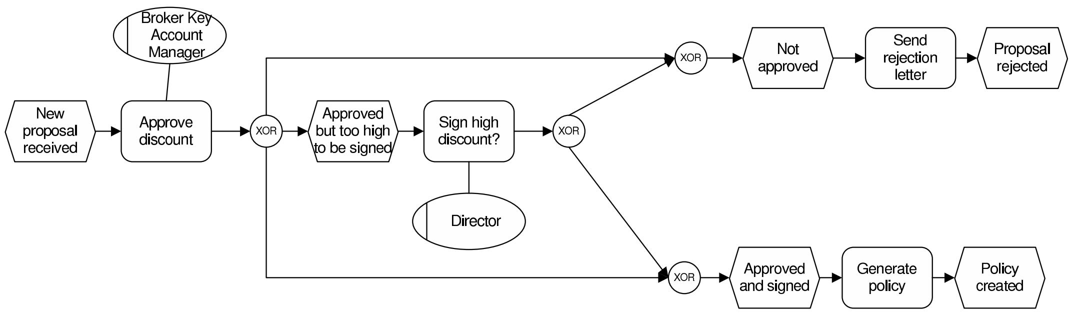


Figure 3.7: Example Event-Driven Process Chain (EPC): Insurance policy creation

on the time dimension. Or if the timeframe has the format of "until x", with one point.

Analysis tools and applications can use the business metadata derived through the weaving links, similarly to technical metadata, to enhance the way users access and interpret data. Where before there were basically only numbers, there now is context and explanation. Through knowing which goal it measures, it becomes clearer what a certain value means and why it is important. The actual values of the measures can be compared to the target values of the metrics. This business metadata can be also incorporated directly into the user interface of analysis tools. Chapter 7 describes a prototype and an example that illustrates the applicability of this approach to business metadata.

3.4 Business Metadata from Enterprise Structure, Behavior and Products

This section extends the weaving model to include further aspects of enterprise models. The additional links make business metadata based on business process and functions, organizational structure and products available.

3.4.1 The Architecture of Integrated Information System (ARIS) and the Event-Driven Process Chain (EPC)

The Architecture of Integrated Information System (ARIS) concept [Sch99] involves dividing complex business processes models into separate views, in order to reduce the complexity. There are three main views focusing on functions, data, and the organization, and an additional view focusing on the integration of the other three.

The EPC has been developed within the framework of ARIS and is used by many companies for modeling, analyzing, and redesigning business processes. For details on EPCs and business process modeling, see Section 2.2.

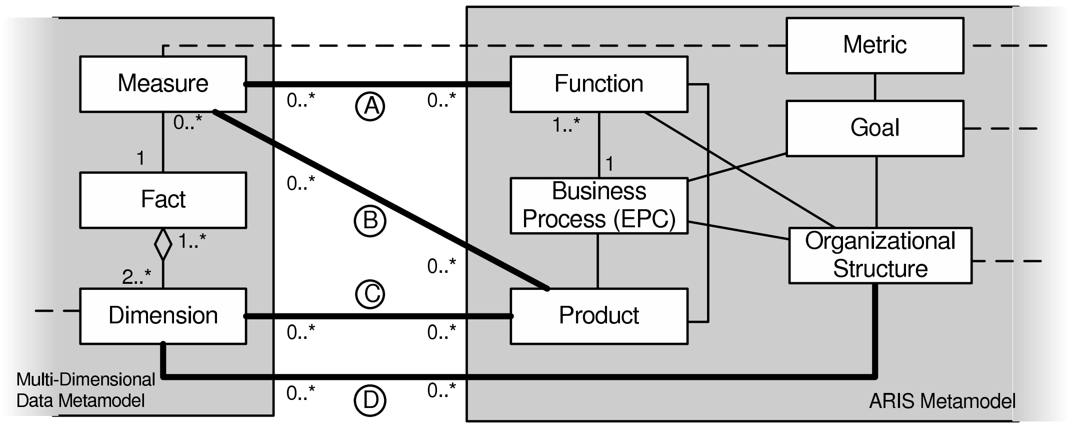


Figure 3.8: Weaving model between data model to the enterprise model. See also Figure 3.6.

The EPC was chosen for this approach because of its wide-spread use in many companies for modeling business processes, and because of its flexible view concept, that allows to separate the different aspects of a business process.

Figure 3.7 shows a simplified example process from an insurance company. The process starts with the arrival of a proposal for an insurance policy (as created and sent by an insurance broker for example). The broker key account manager checks whether the discount is appropriate. There are three possible outcomes of this check: The proposal can be either approved and signed, or not approved by the key account manager, or, if it is approved but the discount is above a certain amount, it has to be signed by a director. For each signed proposal, an insurance policy is generated, whereas if it was not approved, a rejection letter is sent.

3.4.2 Weaving Model

This section presents a weaving model for connecting enterprise models to data warehouse data, in order to derive business metadata. Business metadata allows the data warehouse users to access the context of the data warehouse data. The weaving model stores information about the relationship between the data warehouse and the structure and behavior of an enterprise organization, e.g., which business process or part of it impacts which part of the data warehouse.

Figure 3.8 shows the weaving model (strong lines) between the two existing metamodels (grey): On the left side the core model elements of a multidimensional data model, as described in Section 3.2.1, and on the right side a subset of the model elements of the ARIS Framework's metamodel, as described in Section 3.4.1. Both metamodels are actually larger, as hinted by the blended edges.

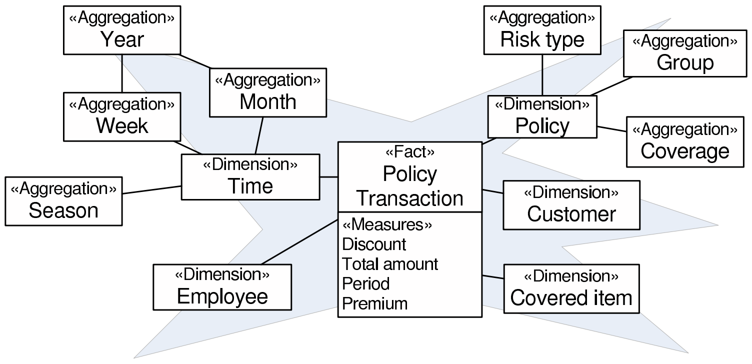


Figure 3.9: Example multidimensional data model (see also Section 2.1.3.1)

Between these two (meta-)models, this section has introduced four weaving links (strong lines, (A) - (D)). They store the relationships between the two domains and allow us to derive business metadata.

This approach makes use of the knowledge already available in the organization in the form of enterprise models for business metadata. The business metadata helps to improve the understanding and interpretation of the data warehouse data by the users. For the viewpoint of the users, it is an additional level of abstraction on top of the data warehouse data. Business metadata should be aligned to the concepts that are well-known to the users, e.g., provide a business process-oriented view.

The weaving model makes formerly implicit relationships between the data in the data warehouse and the structure and behavior of the enterprise organization visible and accessible. The four weaving links shown in Figure 3.8 are described in the following.

The measures of the data warehouse cubes are linked to functions (link A) and products (link B) in the ARIS model. Functions supply measure data to the data warehouse as they create or change values of business objects. Data in the “Policy Transaction” cube (see Figure 3.9) is created each time a process creates or changes the values of a policy. For example, the measure “discount” of the policy transaction cube is set by the function “Approve discount” (see Figure 3.7 in Section 3.4.1). Knowing which function supplies a measure implies knowing the overall business process to which the function belongs. The same applies to the fact cubes of measures. This transitive relationship allows to derive business metadata for analysis needs on different levels of detail (Figure 3.10). It can be shown which business process the data warehouse or which part of it impacts a certain measure.

Regarding products and other deliverables, their values can be found in two different places in the data warehouse: They are represented as measures (link B) as well as dimensional data (link C). For example, the insurance policy as a product will be present as a dimension of several fact cubes of the data warehouse (see Figure 3.11 for an illustration):

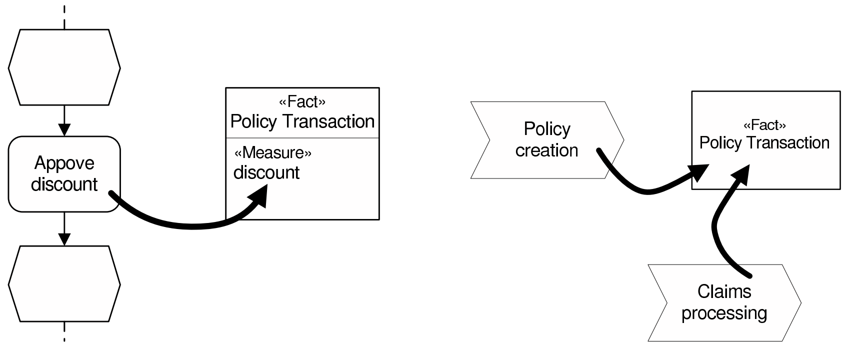


Figure 3.10: A function creates the value of a measure; several processes for a fact table

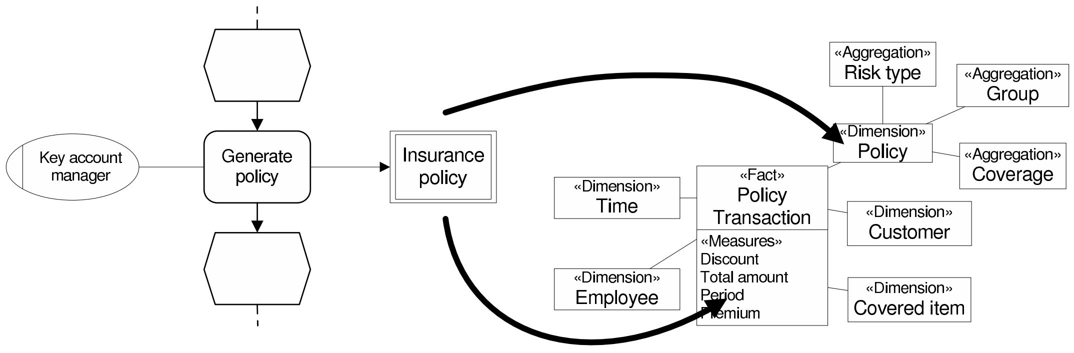


Figure 3.11: Products as dimensional data and as measures

The data on policy transactions can be aggregated e.g., by policy risk type. But the individual attributes of a policy, such as its premium or period (which may be different for each instance and change over time with each transaction) will be found in the measures.

As illustrated in Figure 3.12, elements of the organizational structure, such as units or roles, may be mirrored in the data warehouse directly as dimensional data (link D), or indirectly through other elements such as the business process and functions (via link A) they are in charge of, their products (via links B and C), or goals. Directly as a dimension, the organization may appear as employees and/or departments. Indirectly, functions of business processes have organizational roles or units assigned to them, which can be evaluated through the measures recorded for these functions. The same applies to products or goals.

Figure 3.13, as a summary of the above, schematically illustrates the use of business metadata in Data Warehousing. The user accessing data on policy transactions has several questions: Where do the values in this table come from? Which functions created them and where? What am I measuring here?

The weaving links create the connections and show that the rows of the table are created

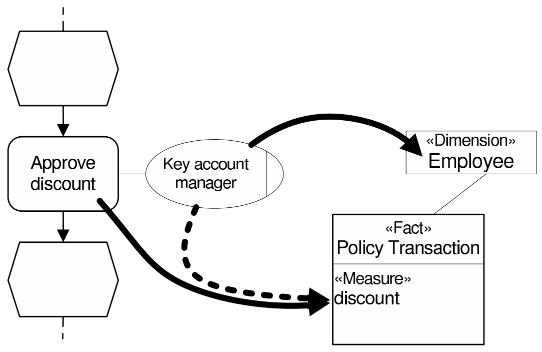
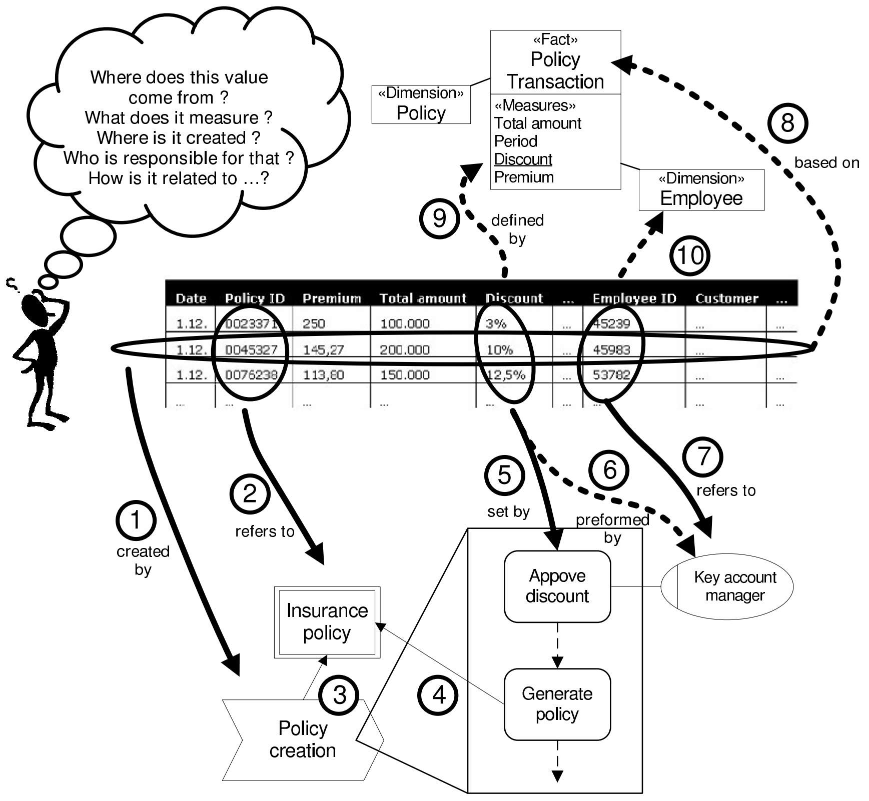


Figure 3.12: Organizational roles: directly or indirectly via other elements


Where does this value come from ? What does it measure ? Where is it created ? Who is responsible for that ? How is it related to …?
«Dimension» Policy
«Fact» Policy Transaction
«Measures» Total amount Period Discount Premium
«Dimension» Employee
8
based on
9
defined by
10
1
created by
2
refers to
5
set by
6
preformed by
7
refers to
Appove discount
Key account manager
Generate policy
Insurance policy
3
4
Policy creation

Date	Policy ID	Premium	Total amount	Discount	...	Employee ID	Customer	...
1.12.	0023371	250	100.000	3%	...	45239	...	...
1.12.	0045327	145,27	200.000	10%	...	45983	...	
1.12.	0076238	113,80	150.000	12,5%	...	53782	...	...
...	...	...	...	...	...	...	...	...

Figure 3.13: Overview of context provided by business metadata

by the business process *Policy creation* (1). The column *Policy ID* (2) refers to the *Insurance policy* which is the product of this process (3). If the users is interested in further details here, it can be shown that the insurance policy is actually the output of the function *Generate policy* within the process (4), and that the values in the column *Discount* are set by the function *Approve discount* of the same process (5). If the user is interested in the discount, more questions appear, such as "Who is responsible for that?". The user can see that the function that sets the value of D*iscount* is performed by the organizational role *Key account manager* (6) which can also be found in the column *Employee ID* (7).

The table as it is shown here refers to the fact table *Policy Transaction* (see (8) and Figure 3.9. The columns are based on either measures (9) or dimensions (10).

The knowledge captured by the weaving model can be exploited by analysis tools (also to offer better navigation or hints). Figure 10 shows how the business metadata can be displayed for the "policy transactions" cube. The organizational knowledge captured in the enterprise model becomes available to the user. Providing this information to the user directly within the analysis tool helps to improve data interpretation. The business metadata thus increases the usefulness of the data.

3.5 Related Work

The term "weaving" is also used in a different sense in Aspect-oriented Programming, where it denotes the integration of aspects into the base program [KLM+97]. See the AOSD Ontology [vdBCC05] for more general definitions that apply not only to the programming level, but also to modeling.

In [BB02], Breton and Bézivin apply model weaving to the area of workflow and process modeling. The build-time and the run-time workflow definitions are weaved together to create a binding between definition and execution of the process. The two models used are the Workflow Management Coalition's process definition metamodel [WMC07] and the OMG's Workflow Management Facility [OMG00].

There are a lot of conceptual models available for business processes, data bases or Data Warehouses. But there are almost no models available that focus on the relationship between the data warehouse and the business processes. EPCs [KNS92] incorporate a data view targeting operational data bases. EPC functions perform read or write operations on the databases and their entities. But they do not take the specific characteristics of Data Warehouses into account. The Business Process Modeling Notation [BPM04] provides data objects, which are used and updated during the process. The data object can be used to represent many different types of object, both electronic or physical. For an integrated view on Data Warehousing and business processes see Chapter 4, in terms of a model that allows to show where and how a data warehouse is used by business processes, and which parts of the business processes depend on which parts of the data warehouse.

Mazon and Trujillo [MT06] applied the MDA framework to data warehouse repository development, and aligned multidimensional conceptual data models with code. Speaking in MDA terms, they aligned a Platform Independent Model (PIM) with a Platform Specific Model (PSM) and defined a transformation between them. Starting with a conceptual model, they developed a logical model using a relational approach to build a star schema. Then, they derive the necessary code to create data structures for the data warehouse. This approach can be seen on top of this work targeting the Computation Independent Level (CIM) level, as it aligns enterprise goals, representing the business requirements as well as context, with the data warehouse conceptual data model.

Giorgini et al. focus on data warehouse requirement analysis based on goals in [GRG05]. They derive the data model from the goals, which represent a rather narrow software engineering type of goals. In contrast, the approach presented here integrates enterprise goals and aligns the data warehouse directly with business strategy.

Sarda linked data warehouse business metadata with technical metadata in [Sar01], in order to provide a better context for decision support. Several business metadata categories like goals, organizational elements, processes, events, measures, etc., and a number of desirable characteristics such as evolution of navigation between metadata and data, are defined. The business metadata is described with UML classes and associations and then linked directly to the technical metadata within the same model. The approach only covers metadata and does not use separate conceptual models of the business context. Additionally, this weaving model is focused on the details of enterprise goals and their measures, rather than on all aspects of an enterprise.

The Pentaho Open Source BI Project[Pen07c] has a metadata subproject, *Pentaho Metadata*. It provides an additional layer of abstraction on top of the data model, which consists of what is called a *domain model*, *business models*, and *business views*. The models are comparable to a (simplified) re-mapping of the underlying data model. Each element enriched with additional metadata properties, which are subject to a sophisticated inheritance hierarchy. But contrary to the "business" names of the models, at least the pre-defined properties that are provided with the released version are all technical metadata of different kinds, e.g., display fonts and colors, security values, or aggregation rules. The metadata can be exported in CMW format. At the time of writing, it is too early to assess the acceptance or development of this project, which has been released in October 2007. Nevertheless it seems likely that a tool such as the prototype presented in Chapter 7 could be integrated with Pentaho Metadata.

3.6 Concluding Remarks

This chapter has presented an approach to business metadata that is based on the relationship between the data warehouse data and the structure, the behavior and the goals of an organization. The enterprise goals and information related to them such as metrics and target values, and business processes and functions, organizational units and roles as well as the products produced by them, are taken from an enterprise model. The business metadata is created by linking this knowledge about the organization to the data warehouse by means of a weaving model.

The business metadata is then created directly from the weaving model. It improves data interpretation by explaining the relevance and context of the data, whereas the weaving model itself supports data warehouse requirements analysis, (re)design and evolution by making context visible and accessible. The approach is applied to an example.

Chapter 4

Business Processes and the Data Warehouse

Data Warehouse information is accessed by business processes, and sometimes may also initiate changes of the control flow of business process instances. Today, there are no conceptual models available that make the relationship between the data warehouse and the business processes transparent.

This chapter presents extensions of two different kinds of business process modeling languages, the Event-Driven Process Chain and the UML 2 Activity Diagram, that allow to make this relationship explicit in a conceptual model.

Contents

4.1 Introduction

In modern organizations, data warehouse systems are utilized for performance measurement [LM04] and play a crucial role, as more and more business processes require information from the data warehouse.

A business process is "a group of tasks that together create a result of value to a customer" [Ham96], and describes how work is done within an organization. When a person applies for a loan in a bank for example, the data warehouse is an integral part of the loan application business process. The applicant is scrutinized to find out if she or he has caused a financial loss previously, or has changed identity and caused damage under a different name. The business processes of designing new products in a telecommunication company or an airline, or composing the product range of a supermarket for example, requires comprehensive information on the customer behavior covered by the data warehouse.

More and more business processes depend on services provided by the data warehouse environment. The components of a data warehouse architecture provide Key Performance Indicators (KPIs), also called metrics or performance measures in other disciplines, to business processes. The data warehouse might simply offer data for decision support, or it may actively influence the control flow of the process. There are lots of examples showing how important data warehouses have become for business processes.

Today, a data warehouse is an integral part of a highly competitive business environment, where it plays a very active role. A data warehouse that is tightly coupled with the operational business, in order to reduce the time between critical business events and the actions taken, is called an active or real-time data warehouse [BR01]. Changes in the business process flow can be initiated in near real-time, allowing to move from acting reactively to proactively. For example, when an order cannot be delivered on time, the active data warehouse of a transportation company can initiate alternative scenarios, such as a different means of delivery or a higher priority for the order. Thus it becomes possible for the company to avoid the delay and the penalty. In this case, the data warehouse actively initiates changes of the process flow.

Surprisingly, this knowledge – how dynamic business structures interact with the data warehouse and how the data warehouse is being used in every day business life – is not made explicit in existing conceptual models. There is a need for an integrated model of processes and data warehouses to make the relationship between the data warehouse and the business processes more transparent.

To bridge this gap, two existing business process modeling languages, namely the Event-Driven Process Chain [KNS92] and the UML 2 Activity Diagram [OMG05c], are extended with a UML Profile for Business Intelligence (BI) Objects, to be able to create models that show

- where and how business processes use a data warehouse environment,
- which parts of the business processes depend on which parts of the data warehouse, and
- how the data warehouse impacts the business process control flows.

A data warehouse stores decision support data. But the data need to be analyzed and interpreted as well. The term Business Intelligence (BI) is used here instead of data warehouse, as it represents a broader approach to decision support data. BI is seen as all kinds of applications and technologies for storing, analyzing, and providing access to data to help enterprise users to make better business decisions. BI goes beyond data warehouses as it covers the entire data warehouse environment from storage to analysis. BI objects cover a broad range of object types.

Business process models enriched with a BI elements provide the following contributions for process managers and data warehouse designers:

- They provide a bigger picture for data warehouse designers, as it shows how the data warehouse and other BI objects are accessed by business processes and where business processes depend on or are influenced by BI.
- The models link static BI structures and dynamic business structures.
- The models show BI objects on different aggregation levels and thus enables the modeler to choose the right level of detail for different purposes or target audiences. The modeler may model a high level data repository access of a business process, e.g., the access of a data mart or data warehouse, or describe the access at a more detailed level, e.g., the access of a certain fact or entity. Furthermore, modelers may also show the access of an analysis tool.
- The models can support the design phase of a BI project, by making it possible to describe the business requirements for the data warehouses or data marts, making it possible to prioritize the subprojects according to business needs.
- They can be used to justify the costs of BI projects by pointing out the unseen relationships between the business processes and its business value, important business decisions, and BI.
- The models can also be used to support estimates of the cost of usage, as well as for risk management: If the data quality in a certain area is bad, a data mart fails or data is corrupted, an integrated model enables better reactions because it is known which business processes will be affected.
- Furthermore, the models also allow to discover parts of the data models which are not accessed at all, and to decide if these parts should be further maintained.

Section 4.2 adds a perspective to the Event-driven Process Chain [KNS92], a well-known business process modeling language made popular through SAP R/3. Section 4.3 describes the UML Profile for BI Objects, which extends the UML Activity Diagram [OMG05c], another model used for business process modeling. Section 4.4 covers related work. Parts of Section 4.2 have been published in [SLS05] and [SL05], and of Section 4.3 in [SLK05].

4.2 A Business Intelligence Perspective for Event-Driven Process Chains

Data Warehouse information is accessed by business processes, and sometimes may also initiate changes of the control flow of business process instances. Today, there are no conceptual models available that make the relationship between the data warehouse and the business processes transparent.

There is a need for an integrated model of processes and data warehouses to make the relationship between the data warehouse and the business processes more transparent. To bridge this gap, a business process modeling language is extended with an additional perspective, the Business Intelligence (BI) Perspective, to be able to create models that show

- where and how business processes use data warehouses,
- which parts of the business processes depend on which KPIs from specific areas of the data warehouse architecture, and
- how the data warehouse impacts the business process control flows.

This section presents an extension to the the Event-Driven Process Chain [KNS92] – the BI Perspective – to make this relationship explicit in a conceptual model. The additional perspective is tested with a example business process.

The business process modeling language chosen to be extended with the BI Perspective is the Event-Driven Process Chain (EPC). It is introduced in Section 4.2.1. The EPC has been chosen because of its wide-spread use and because of its flexible view concept which allows to separate the different aspects of a business process. One can easily add another perspective while keeping the original structure intact.

The BI Perspective is divided into a *Traditional BI Perspective* and an *Active BI Perspective*. It covers the two main types of interaction between the data warehouse environment and business processes: The Traditional BI Perspective (Section 4.2.2) shows the different levels and perspectives of a data warehouse environment which are relevant to a business process, from accessing the full data warehouse or a data mart, to single facts and measures, as well as KPIs in reports. Three typical usage scenarios for the Traditional BI Perspective were identified, and this section also shows how it can be applied in Business Process Dependency Diagrams.

The Active BI Perspective (Section 4.2.3) shows how an active data warehouse reacts to events that occur in a business process and how it influences the control flow of a business process. Related Work is presented in Section 4.4.

4.2.1 Event-Driven Process Chains (EPCs)

The *Architecture of Integrated Information Systems* (ARIS) [KNS92] divides complex business process models into separate views to reduce the complexity. The views can be handled independently. There are three views focusing on functions, data, and the organization (see Figure 4.1), and an additional view focusing on the integration of the other three.

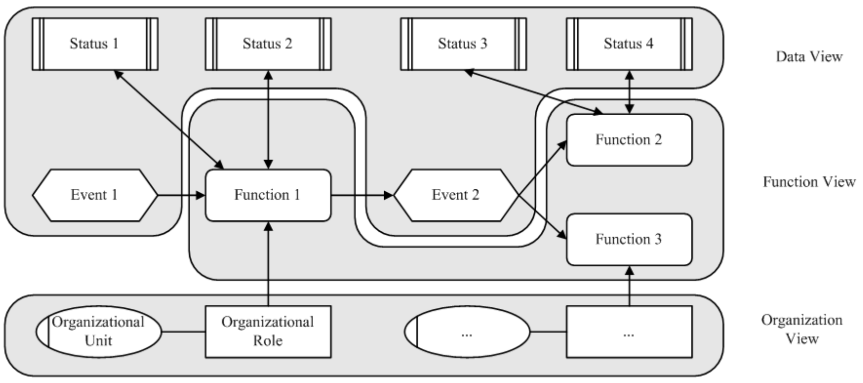


Figure 4.1: ARIS Views

The *Data View* contains events and statuses. Events such as “customer order received”, or “invoice written” are objects that represent data. Statuses such as “customer status” and “article status” are also represented by data. To provide the data view with a description method for statuses, Chen’s Entity-Relationship (E/R) model [Che76] was adopted into the ARIS framework, since it was the most widespread designing method in the area of data modeling. Today, the UML class diagram is also used [OMG05c].

The *Function View* contains the description of the activities to be performed, the individual sub-functions, and relationships that exist between the functions.

The *Organization View* represents the organizational structure. This includes the relationships between organizational units, between employees and organizational units, and employees and their roles.

The *Control View* links functions, organization units or roles and data. It integrates the design results of the different views, which were initially developed separately for reasons of simplification. The functions, events, information resources, and organization units are connected into a common context by the control flow. The resulting model is the EPC.

The EPC has been developed within the framework of ARIS and is used by many companies for modeling, analyzing, and redesigning business processes. EPCs were developed in 1992 at the Institute for Information Systems of the University of Saarland, Germany, in collaboration with SAP AG. It is the key component of SAP R/3’s modeling concept for business engineering and customizing. The EPC is based on the concepts of stochastic net-

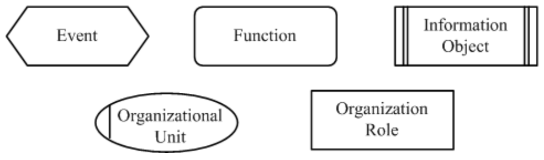


Figure 4.2: EPC Elements

works and Petri nets.

EPCs are an intuitive graphical business process description language [Sch99]. They describe processes on the level of their business logic, and are targeted to be easily understood and used by business people. An EPC represents the control flow structure of the process as a chain of alternating events and functions. A basic EPC consists of the following elements (Figure 4.2):

Functions are active elements. They model the activities within the company.

Events are created by processing functions or by actors outside of the model. An event may act as a pre-condition or post-condition of one or correspond to the post-condition of another function.

Logical operators connect functions and events. There are three types of logical operators: AND, XOR (exclusive or) and OR.

The extended EPC adds the following elements:

Organizational Units or **Roles** are responsible for performing a function.

Information Objects represent input data serving as the basis for a function, or output data produced by a function. They correspond to entities or attributes of the ER model or classes and attributes of the UML class diagram.

4.2.2 Traditional BI Perspective

This section extends the EPC with features for BI modeling to enable the creation of models that integrate information about where the process makes use of decision support data. These models make the hidden knowledge about the relationships between the business processes and BI explicit. This chapter is divided into two parts. This section describes the Traditional BI Perspective, which can be used to model scenarios in which business processes access data warehouses, data marts or reports, followed by a section about the Active BI Perspective, where the data warehouse actively influences the control flow of business processes.

4.2.2.1 Extended Metamodel of the EPC

The EPC is chosen as a basis for the model because of its wide-spread use in many companies for modeling business processes, and because of its flexible view concept, that allows to separate the different aspects of a business process. Another perspective can easily be added while keeping the original structure intact.

The EPC metamodel (white) including the Traditional BI Perspective (dark) is shown with a sufficient level of granularity in Figure 4.3. Each EPC consists of one or more *Functions* and two or more *Events*, as an EPC starts and ends with an event and requires at least one function for describing a process. A function can be either a *Complex Function* or an *Elementary Function*. Complex functions are refined by one or more other functions. A function is connected with two *Control Flow Connectors*. An event is connected with one or two control flow connectors. Control flows link events with functions, but also events or functions with *Logical Operators*. A logical operator can be either XOR, OR or AND. A logical operator is connected at least with three control flows, one or more incoming and one or more outgoing connectors.

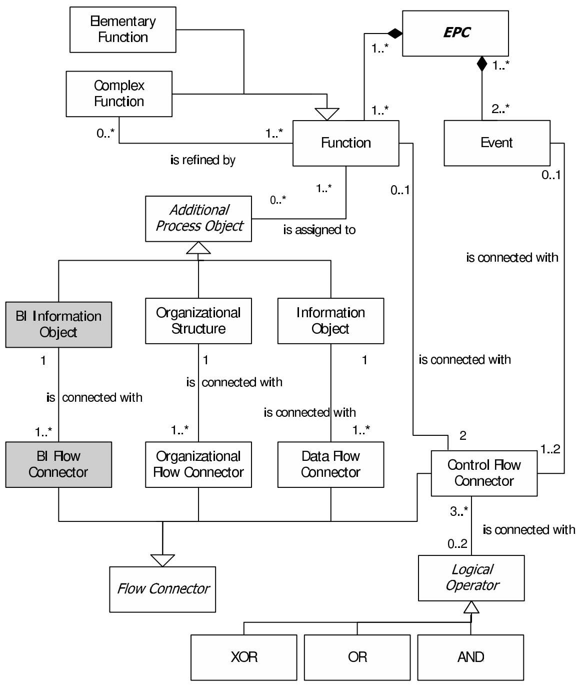


Figure 4.3: EPC Metamodel with BI Information Object and BI Flow Connector

An *Additional Process Object* may be assigned to one or more functions, for example an *Information Object* or an *Organizational Structure*. All types of additional process objects may be assigned to any function. The organizational structure is connected with one or more

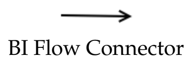


Figure 4.4: BI Flow Connector: Connects BI Information Objects to EPC Functions

Organizational Flow Connectors. The information object is connected with one or more *Data Flow Connectors*.

The EPC metamodel is extended with what is called the *Traditional BI Perspective* and introduce a *BI Information Object* as an additional process object. The detailed metamodel of the BI perspective is shown in Figure 4.5. All elements of the model are specializations of the BI Information Object. All BI information objects are Additional Process Objects in terms of the EPC, which means that they can be assigned to a function that uses the information contained in them. The BI information object is connected with one or more BI Flow Connectors, the notation of which is shown in Figure 4.4 (cf. Figure 4.22 in Section 4.3.1).

What is a BI information object? It represents the ways in which a business process might access specific areas of the data warehouse architecture, e.g., decision makers might use reports or analysis tools to obtain KPIs. The business process might see the data warehouse as a whole, or focus on individual data marts. A subprocess or function can use individual entities and attributes, depending on the level of detail of the process model. In a detailed model of a business process, the individual data entities and attributes accessed by a function could be shown. On the larger scale, a process accesses the whole data warehouse, or individual data marts. With or without data warehouses, decision makers often receive relevant data in form of reports, e.g., a report on sales data for the past fiscal year.

Three main categories of BI information objects are thus identified: *BI Data Repositories* (the different databases of the data warehouse architecture), *BI Data Objects* (the elements of the data model of a certain repository), and *BI Information Presentation Objects* (the data presentation to the user). The relationships between the categories and their objects are shown in terms of a metamodel in Figure 4.5.

4.2.2.2 BI Data Repositories

BI data repositories are the first type of BI information object that can be modeled in relation to a business process. They basically represent different types of databases as used in data warehouse settings. The types of BI data repositories occurring in a given situation depend on the data warehouse architecture in an organization. Also, several different data repositories may exist in parallel. The approach presented here is not limited to any specific data warehouse architecture, but can be applied to a wide selection of architecture types.

Depending on the architecture, different combinations of BI data repositories may occur in an organization. In large multinational organizations it is not uncommon to have more than one data warehouse. A large data warehouse often co-exists within an organiza-

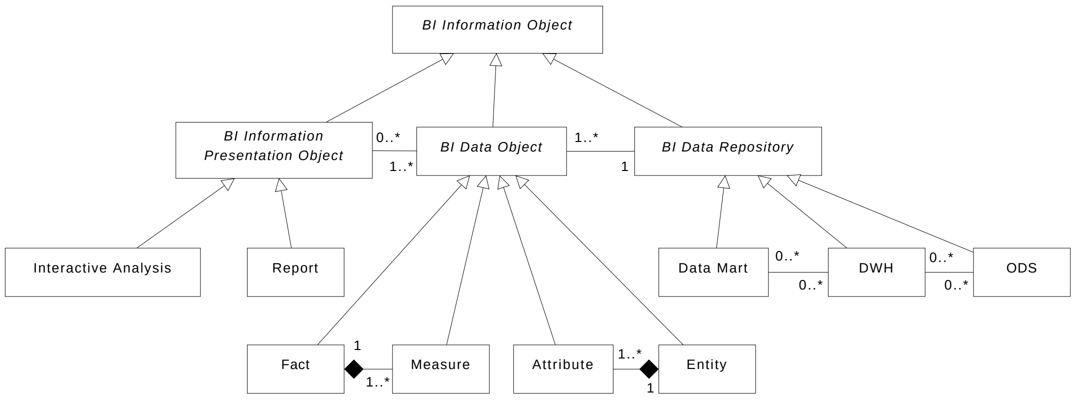


Figure 4.5: Metamodel of the Traditional BI Perspective

tion with smaller data marts (DM), departmental subsets of a data warehouse focused on selected subjects [CD97]. The data marts might be created based on the data warehouse, obtaining their data from there, meaning that a data mart acts as a kind of materialized view on the data warehouse. In another situation, the data marts may be created individually by departments without an underlying data warehouse, and then later be integrated into an organization-wide data warehouse, making possible operations that span several data marts. Also, there may be none, one or more operational data stores (ODS), located between the operational systems and the data warehouse [GRC04]. Depending on the architecture, end user applications may query individual data marts and/or the data warehouse, or even access the data in the ODS directly.

In order to allow the greatest possible flexibility and provide meaningful content in the models, three basic types of BI data repositories were identified: the *data warehouse* (DWH), the *Data Mart* (DM) and the *Operational Data Store* (ODS). They are related to each other through data dependencies. The notation for these elements is shown in Figure 4.6.

To illustrate the relationships between the BI data repositories a simple repository dependency diagram is proposed. In the example diagram shown in Figure 4.7, the data warehouse depends on two independent ODS systems, and in turn supplies four data marts with data.

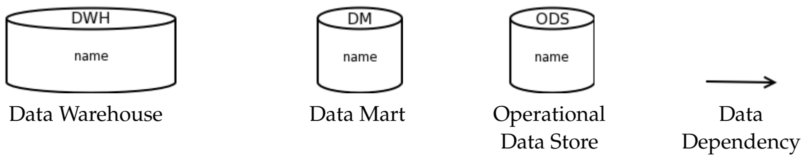


Figure 4.6: Notation of BI Data Repositories

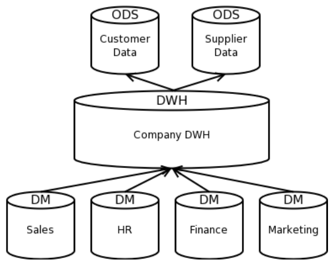


Figure 4.7: Data Repository Dependency Diagram

4.2.2.3 BI Data Objects

In order to provide a more detailed view of the data accessed by the functions of an EPC, the individual data entities contained in the BI data repositories, should also be modeled. These *BI Data Objects* are generally represented by conceptual data models. For example, if a function needs data on the revenue of a certain product range, it can be modeled to access the corresponding BI data object directly.

Depending on the type of repository, the overall architecture, and the preferences of the designers, different kinds of data models can be used. The two main types relevant to BI applications are entity-relationship modeling and multidimensional modeling [GMR98a]. In the first case, use the Entity-Relationship model [Che76] is used, and in the latter, the Multidimensional Entity Relationship (ME/R) model [SBHD99] described in Section 4.2.2.4 below.

The BI data objects of an E/R model that can be accessed by an EPC are either *Entities* or individual *Attributes*. In the case of the ME/R, they are *Facts* or *Measures*. The notation of BI data objects is described in Figure 4.8. Whether entities/facts or attributes/measures are to be used as additional process objects in the EPC depends on the granularity of the EPC functions.

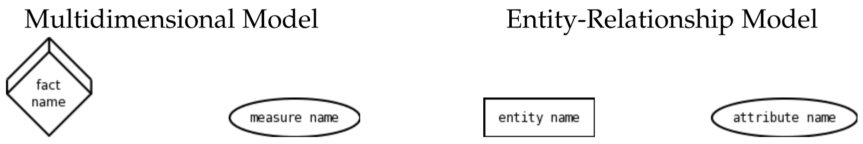


Figure 4.8: Notation of BI Data Objects

4.2.2.4 Multidimensional ER Model (ME/R)

Data Warehouse applications involve complex queries on large amounts of data, which are difficult to manage for human analysts. Relational data models “are a disaster for query-

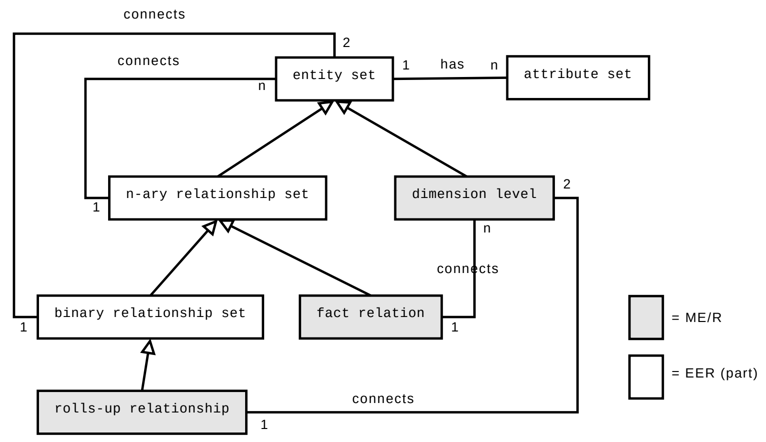


Figure 4.9: ME/R Metamodel

ing because they cannot be understood by users and they cannot be navigated usefully by DBMS software" [KR02]. In Data Warehousing, data is often organized according to the multidimensional paradigm, which allows data access in a way that comes more natural to human analysts. The data is located in n-dimensional space, with the dimensions representing the different ways the data can be viewed and sorted (e.g., according to time, store, customer, product, etc.).

A multidimensional model, also called star schema or fact schema, is basically a relational model in the shape of a star. At the center of the star there is the fact table. It contains the subject of analysis (e.g., sales, transactions, repairs, admissions, etc.). The attributes of the fact table (e.g., cost, revenue, amount, duration, etc.) are called measures. The spokes/points of the star represent the dimensions according to which the data will be analyzed. The dimensions can be organized in hierarchies that are useful for aggregating data (e.g., store, city, district, country). Stars can share dimensions, creating a lattice of interconnected schemas that makes drill-across operations possible.

The Multidimensional Entity Relationship (ME/R) Model [SBHD99] is chosen as the conceptual model for a multidimensional data model for this purpose because of its simplicity and expressiveness. ME/R extends the E/R model by adding three elements that are specializations of existing E/R elements.

The ME/R metamodel is shown in Figure 4.9. The white elements are part of the E/R metamodel and the gray elements denote the additions by the ME/R. A fact is defined as a *fact relation*, because it is a specialization of the basic *n-ary relationship set*, connecting n dimensions. Because dimensions are organized in hierarchies, they are divided into dimension levels. A *dimension level* is a specialization of an *entity set*, connected with n other dimensions via a fact relation. Within the hierarchy of a dimension, the dimension levels

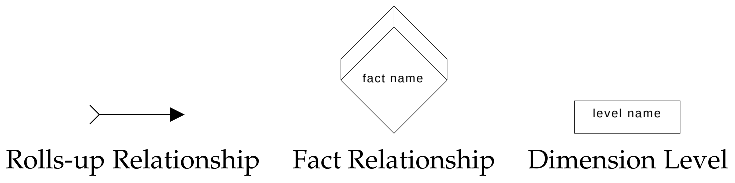


Figure 4.10: ME/R Notation

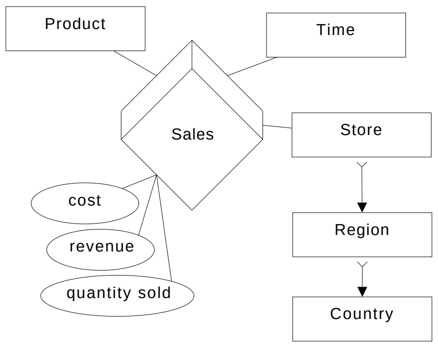


Figure 4.11: ME/R Notation: Example

are related to each other via *rolls-up relationships*. Each rolls-up relationship connects two neighboring dimension levels (e.g., product rolls up to category, or region to country). The notation of the ME/R elements is shown in Figure 4.10.

The example in Figure 4.11 shows a simple example of a fact relation, inspired by [KR02], described in ME/R notation. The *Sales fact* has three dimensions, *Time*, *Product* and *Store*. The levels of the dimensions are only shown for the store dimension. Each entry in the fact table contains information about a single sales event, meaning that one or more items of a product were sold at a store at a given time. For each sale the revenue achieved, the cost incurred, and the quantity sold can be analyzed, as well as aggregations such as "total revenue of a product in all stores in one year". Several such facts can be connected by sharing the same dimensions, creating a more complex multi-cube model.

4.2.2.5 BI Information Presentation Objects

In a data warehouse architecture of an organization employing BI techniques, there are usually tools and applications providing users with prepackaged information that has been compiled for them. These collections of information are called *BI Information Presentation Objects*, and have identified two different types: *Report* and *Interactive Analysis*. A report is a kind of document containing the KPIs (measures) related to a certain area, for example a report on sales in the south region for the 4th quarter of 2007. The values contained in a report do not change over time. An interactive analysis on the other hand is closer to a tool. It provides its users with regularly updated values and can be used for continuous performance monitoring. In the EPC it is possible to show e.g., a certain report that a function accesses. The notation for BI information presentation objects is shown in Figure 4.12.

report name
Report
Interactive Analysis

Figure 4.12: Notation of BI Information Presentation Objects

4.2.2.6 Usage Scenarios

Three main scenarios regarding the usage of the BI perspective were identified. They depend on the target user group, and offer modeling solutions for typical every-day requirements.

DWH managers are looking for the big picture, an overview of what is going on. They will use EPCs showing business processes with the BI data repositories. This allows them to find answers to questions such as "Which processes use this data mart?", "Which business processes require direct access to the data warehouse?", or "If this data mart fails, which processes are in danger?".

Business users are interested in business decisions. An EPC model showing an individual business process or sub-process in connection with reports and interactive analyzes will support questions such as "Which reports are accessed where?" and "Which important business decisions are supported by the data warehouse environment?".

DWH designers and **developers** need to understand the details of how the data warehouse environment is used. They will use a fine-grained model of a business process or subprocess with facts and entities, attributes or measures.

4.2.2.7 Example

The BI information objects introduced in the previous sections extend the EPC metamodel as additional process objects. Therefore, they can be used in any EPC diagram. A BI information object connected to a function indicates that the function requires data provided by the BI object.

The example process in Figure 4.13 does not correspond directly to one of the usage scenarios introduced in the previous section, but it is better suited as a light-weight example. It

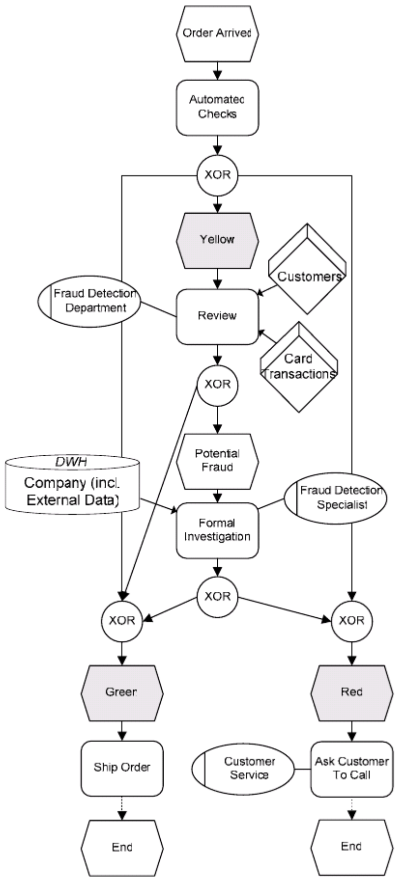


Figure 4.13: Example EPC with BI Elements

illustrates the use of the BI data object “Fact” (Figure 4.8) and the BI data repository “DWH” (Figure 4.6).

BI is widely used in the area of fraud detection. The example EPC in Figure 4.13 demonstrates the application of the Traditional BI Perspective in such a case. Any company selling goods online is faced with the problem of credit card fraud. In this example fraud detection business process, every order is first subjected to a number of automated checks. Depending on the result of the first steps, only suspicious cases are investigated further, in order to reduce the workload of the fraud detection department and reduce overall costs.

The automated checks performed for each order include, among other things, methods such as checks against lists of known offenders, identity verification of the people involved, plausibility checks, and the authorization status of the credit card.

The function *Automated Checks* is started by the event of an order arriving, and does not need access to the data warehouse. As a result of this function, the order is either *Red* (client has been involved in previous problems), *Yellow* (suspicious) or *Green* (not suspicious).

For any order classified as *Red*, the customer is asked to phone the customer service department. All orders flagged as *Yellow* are forwarded to the fraud detection department for review and further analysis. In this step, the details of the data used in the previous step are reviewed and compared to details and history of similar cases. The reviewer then decides whether the risk of potential fraud is enough to justify a formal investigation of the case. False positives generated by the automated checks are likely to be identified in this step. They are re-classified as *Green* and will be processed normally.

The function *Review* is performed by the fraud detection department. The function accesses the information contained in the facts *Customers* and *Credit Card Transactions*. The output events of this function indicate that the order has either been identified as *Potential Fraud* or as *Green*.

The orders likely to be fraudulent are analyzed in depth by a fraud detection specialist. In order to obtain certainty concerning the final assessment of the order, the investigator can explore the entire data warehouse of the company, searching for information that answers the remaining open questions. This includes access to external data sources. If the order has been successfully identified as fraudulent, it is flagged red and denied. Again, if the investigation reveals that the order is genuine, it is processed normally.

In the EPC, this step is modeled by the function *Formal Investigation*. It needs access to the whole data warehouse, which, in addition to the usual wealth of company internal data, also contains external data. This is typical of a process step that cannot be pre-defined in detail. It strongly depends on the individual situation, which information can be of use to the investigating specialist. Again, the function is followed by an XOR-operator, either leading to the event that the order is classified as definitely red or green.

4.2.2.8 BI Objects in Business Process Dependency Diagrams

Detailed diagrams such as the one in Figure 4.13 provide a high level of detail on the flow of a business process, which is not necessary for all purposes. On a macro level, business processes are modeled as black boxes whose internal complex process logic is hidden for sake of clarity. Such diagrams are useful for providing a large scale overview, e.g., over the way a company does business, using a process landscape diagram showing all processes and their interactions with other processes or customers or partners. Diagrams at a macro level give a comprehensive understanding and highlight the relationships with, or the dependencies on other objects.

Such a macro view can also be combined with BI information objects. In order to show the dependencies between BI information objects and business processes this section introduces a business process dependency diagram. This diagram may show several different types of cases: Firstly, a business process depending on one or several BI data repositories.

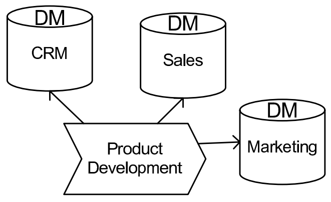


Figure 4.14: Which BI Data Repositories does a process depend on?

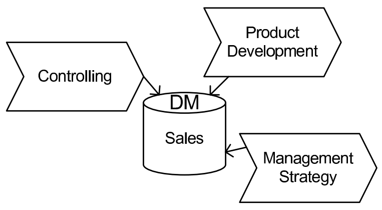


Figure 4.15: Which processes use a BI Data Repository?

This shows the variety of BI data repositories supplying a business process with data. If only one BI data repository fails, the entire business process will be affected and may even stop operating. Secondly, business processes depending on a BI data repository. This shows where the data from a certain BI data repository are used, and which business processes are affected when a BI data repository fails, or provides bad data quality. In both cases, instead of BI data repositories, reports can also be used. Figure 4.14 shows a business process dependency diagram that illustrates on which data marts a business process relies on. The process Product Development employs information provided by three data marts: CRM, Sales, and Marketing.

From another point of view, it is possible to use the business process dependency diagram to show where the data from a certain data mart is used. Figure 4.15 provides an example showing the Sales data mart providing the business processes Controlling, Product Development and Management Strategy with data.

On the macro view BI information presentation objects can also be combined with business processes. A Report supplies business processes with data and a business process depends on data from one or more reports. In order to illustrate these relationships, a business process dependency diagram with reports as shown in Figure 4.16 can be used. This shows the variety of reports supplying a business process with data. If only a single report contains invalid data, the whole process is affected.

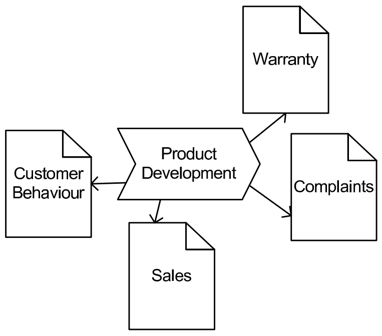


Figure 4.16: Which Reports are required by a process?

4.2.3 Active BI Perspective

In a process-driven enterprise, BI should not only be seen as a pure static retrieval of pre-processed data to support decision makers. With the control of business processes in near real-time through knowledge and awareness of current business situations, BI moves from acting reactively to proactively. An essential prerequisite for this step is a process-oriented embodiment of BI, thereby becoming an integral part of the business process. Active BI is a dynamic discovery process which continuously:

- Observes and collects events from a business environment.
- Converts the event data into meaningful business information.
- Discovers and analyzes business situations and exceptions.
- Automatically decides the most appropriate actions for a response to the business environment.
- Executes the business actions based on the decision that has been made.

This response can either change the state of a business process or notify humans or IT systems that may be interested in the outcome and result of the decision making.

This section extends the EPC with elements that enable the creation of models that integrate information about where the control flow of a business process is changed by BI data.

4.2.3.1 Characteristics of BI Processes

BI processes generate new knowledge and business information for supporting business processes. The main objectives of BI processes are to:

Table 4.1: Stages of Sense and Respond Loops

Stage	Scope	Description
Sense	What is the current state of the business environment?	Events are unified, cleaned and prepared before the actual analytical processing begins.
Interpret	What do the captured events indicate? What do the event data mean for the current situation of the organization?	Transformation of the events into business information, such as key performance indicators. Discovery of business situations and exceptions.
Analyze	Which business opportunities and risks can arise? What are the possibilities to improve the current situation of the organization?	Analysis of key performance indicators and determination of root causes for business situations and exceptions. Prediction of the performance and assessment of the risks for changing the business environment.
Decide	Which strategy is the best to improve the current situation of the organization? What are the actions required to successfully put a decision into action?	Selection of the best option for improving the current business situations and determines the most appropriate action for a response to the business environment. This step can be automated with rules or by involving humans (a decision maker selects from alternative choices; the Sense and Respond loop continues the processing with the choice of the decision maker).
Respond	Who has to implement the decision? How can the decision be put into action?	Response to the business environment by communicating the outcome to the business process with events.

- Discover situations and exceptions in business processes. For instance, organizations want to detect suspicious customer behavior (e.g., fraud behavior) which can be countered with a proactive response.
- Provide proactive responses by continuously observing and analyzing customers, business partners and the competition, business processes can be adapted and optimized (e.g., continuous update and optimization of production plans based on the current orders of customers).
- Analyze business data in real-time, in order to change processes instances during execution.

Such BI processes represents what is called a *Sense and Respond Loop*, which can be divided into 5 stages. In Table 4.1, the phases are described.

4.2.3.2 Extended EPC Metamodel

The following extends the EPC metamodel (Figure 4.3) with BI Process Objects (grey elements in Figure 4.17) that link EPCs with BI processes. Through Event Flow Connectors, a Sense Delegate receives and forwards business process events, which start the BI process.

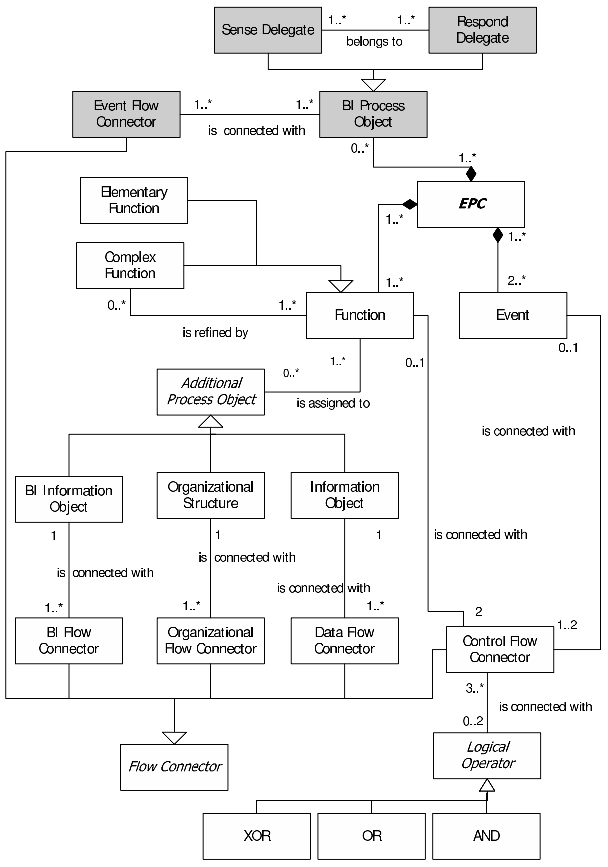


Figure 4.17: EPC Metamodel with BI Process Objects and Event Flow Connector

The Respond Delegate communicates the outcome of the sense and respond loop, i.e., the BI process.

Figure 4.18 shows a generic EPC which is supported by a BI process. Events of the EPC are linked with a sense delegate, which represents the collection and forwarding of these events which become the input for the BI process. Therefore, the sense delegate is a sensor that emits the business process events to the corresponding BI processes. On the other hand, respond delegates communicate an event about the outcome of the BI process.

4.2.3.3 Metamodel of the BI Process

The objectives listed in Section 4.2.3.1 indicate that BI processes are very event-driven in their nature and focus on processing in order to deliver results in near real-time. BI processes generally do not require much interaction with humans and external systems of business partners. Furthermore, processing steps depend mostly on the availability of pro-

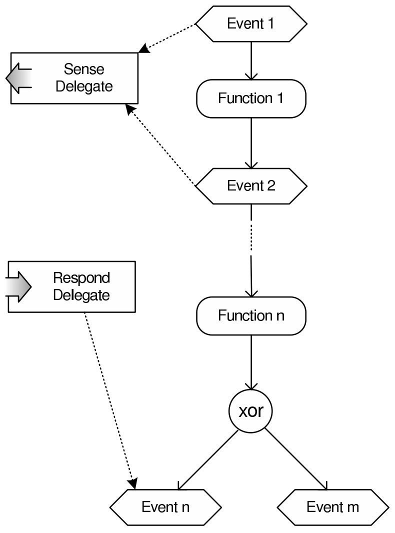


Figure 4.18: EPC with a BI Process

cessing results (e.g., analytical results, discovered situations) from other steps. Therefore, in a BI process, event flows have a higher importance than control flows.

For these reasons, this section proposes a model that is tailored to the requirements of BI processes. EPCs do not show direct data flows between functions which makes it difficult to model data-driven tasks. The BI metamodel (Figure 4.19) includes various elements which can be used to construct a BI process. External interfaces to business processes are represented by *Sense Delegates* and *Respond Delegates*. A *Processing Step* represents a unit of work that has to be performed within the BI processes (e.g., the calculation of a performance indicator or a data analysis). *Event Flow Connectors* represent event streams and connect the processing steps within the BI process. They show how events flow from one processing step to the next. *Connection Points* facilitate merging and splitting event streams.

Since the connectors between delegates and processing steps are event flows (which are essentially data flows), this model does not use any typical elements for control flows (e.g., XOR or OR elements in EPCs) to reflect that BI processes are generally data-driven in their nature. An event flow essentially represents a data flow which conforms to a certain event type. The notation of the BI process elements is shown in Figure 4.20.

Please note that the sense and respond delegates in a BI process do not have to originate from the same business process. BI processes can support multiple business processes. For instance, if a BI process is used to intelligently control the production process based on the currently placed orders, the sense delegate would be part of the order process and the respond delegate would be part of the production process. In other words, with the model presented here BI processes can be used to intelligently adjust business process behavior

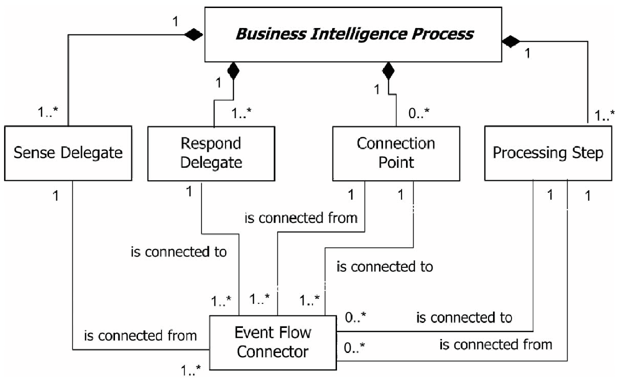


Figure 4.19: Metamodel of the BI Process

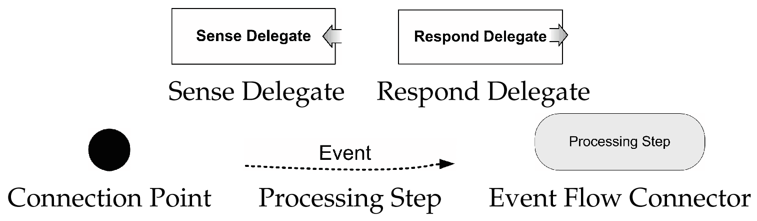


Figure 4.20: Notation of the BI Process Elements

across the entire enterprise.

In the example shown in Figure 4.21, the sense delegate *Order Request* receives order events occurring in the business process. Each event passes through a number of processing steps. In the first step, all order events are interpreted. Only if an order with a value higher than a certain threshold is identified, it is passed on to an analyzing step. In the case of this example the payment history of the customer involved is reviewed. Depending on the outcome of the analysis, the order request is either denied, e.g., the event is passed on to the *Deny Order* respond delegate, or a second analysis step, an External Credit Check, is performed. Again, depending on the outcome of the check, the order may be denied. In the case of a satisfactory outcome, a decision on a discount is made. When the amount of the discount has been decided, the *Give Discount* respond delegate responds to the business process.

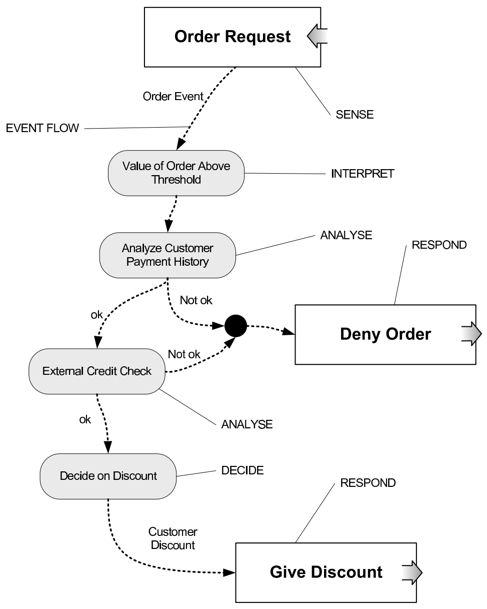


Figure 4.21: Example BI Process

4.3 Extending UML 2 Activity Diagrams with Business Intelligence Objects

Data Warehouse information is accessed by business processes. The knowledge how dynamic business structures interact with the data warehouse and how the data warehouse is being used in every day business life is not made explicit in existing models. There is a need for an integrated model of processes and data warehouses to make the relationship between the data warehouse and the business processes more transparent. To bridge this gap, this chapter extends a business process modeling diagram, namely the UML 2 activity diagram [OMG05c], with a UML profile for Business Intelligence (BI) Objects, to be able to create models that show

- where and how business processes use a data warehouse environment, and
- which parts of the business processes depend on which parts of the data warehouse.

UML profiles provide an extension mechanism for building UML models for particular domains or purposes [OMG05c]. This extension mechanism is used because the UML Profile for BI Objects provides the advantage that data warehouse people are able to view business process models and the interaction with a data warehouse in a well-known notation. In addition to the reuse of the UML notation, these models can be easily presented and edited with UML tools, as almost all UML tools support UML profiles.

The UML profile provides elements for data repositories (data warehouses, data marts, or operational data stores), data objects representing data models of data repositories (entities or facts), and presentation objects representing tools (reports or analysis tools). These BI objects can be accessed by business processes, or in this case by the actions of the UML 2 activity diagram.

Based on the metamodel in Section 4.3.1, a UML Profile for BI Objects extending UML 2 activity diagrams was developed in Section 4.3.2. The UML profile is tested with example business processes in Section 4.3.3. Section 4.4 covers related work.

4.3.1 Metamodel of Business Intelligence Objects

This section presents an extension to the UML 2 activity diagram with a UML Profile for BI Objects to enable the creation of models that integrate information about where a business process makes use of data for decision support. These models make the otherwise hidden knowledge about the relationships between the business processes and BI explicit. This section describes the metamodel of BI Objects (cf. Figure 4.5 in Section 4.2.2). The metamodel is related to the models presented in Section 5.3.1 and by [LMTS06].

What is a BI object? There are three main categories of BI objects: Data Repositories (representing the elements of the data warehouse architecture), Data Objects (representing the data model of a certain repository), and Presentation Objects (representing the means of presentation, either a static report or an interactive analysis). The relationships between the BI objects are shown in Figure 4.22.

BI objects chosen for a model depend on the target audience and the level of detail of the model. In an overview business process model suited for data warehouse managers, one might show the data warehouse or individual data marts as a whole. In a more detailed model for developers, subprocesses can be described as accessing individual entities and facts. Additionally, decision makers often receive relevant data in form of reports, for instance a report on sales data for the past fiscal year, which may also be relevant for business process modeling.

4.3.1.1 Data Repositories

Data Repositories are the first type of BI object that can be modeled in relation to a business process. They basically represent different types of databases as used in data warehouse settings. The types of data repositories occurring in a given situation depend on the data warehouse architecture in an organization. Also, several different data repositories may exist in parallel. This approach is not limited to any specific data warehouse architecture, but can be applied to a wide selection of architecture types. In order to allow the greatest possible flexibility and provide meaningful content in the models, three basic types of data

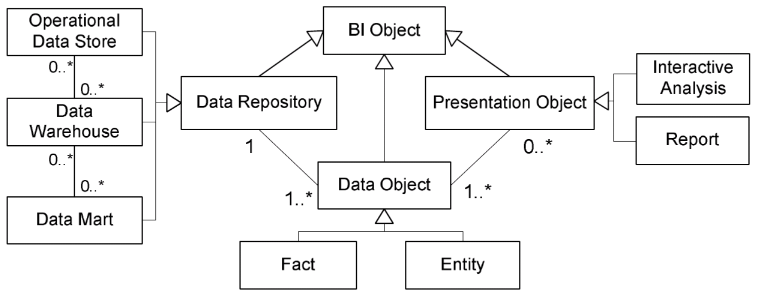


Figure 4.22: Metamodel of Business Intelligence Objects (see also Figure 4.5)

repositories were identified: the Data Warehouse (DWH), the Data Mart and the Operational Data Store (ODS).

Depending on the architecture, different combinations of BI data repositories may occur in an organization. In large multinational organizations it is not uncommon to have more than one data warehouse. Within an organization a large data warehouse often co-exists with smaller data marts, departmental subsets of a data warehouse focused on selected subjects [CD97]. The data mart might be based on the data warehouse, obtaining its data from there, and acting as a kind of materialized view on the data warehouse. In another case, each data mart may be created individually by a department without an underlying data warehouse. To make operations spanning several data marts possible, they may later be integrated into an organization-wide data warehouse. Also, there may be none, one or more ODS, located between the operational systems and the data warehouse [GRC04]. Depending on the architecture, end user applications may query individual data marts and/or the data warehouse, or even access the data in the ODS directly.

4.3.1.2 Data Objects

In order to provide a more detailed view on the data, the individual data entities contained in the data repositories can also be modeled. These Data Objects are generally represented in conceptual data models. For example, if a business process needs data on the revenue of a certain product range, it can be modeled to access the corresponding data object directly. In BI settings, there are two common types of data models: entity-relationship (E/R) models [Che76] and multidimensional models [CD97, GMR98a, KR02]. Which model is used depends on the type of repository, the overall architecture, and the preferences of the designers. The data objects of an E/R model that can be accessed by an activity of a business process are Entities. In the case of the multidimensional model, they are Facts.

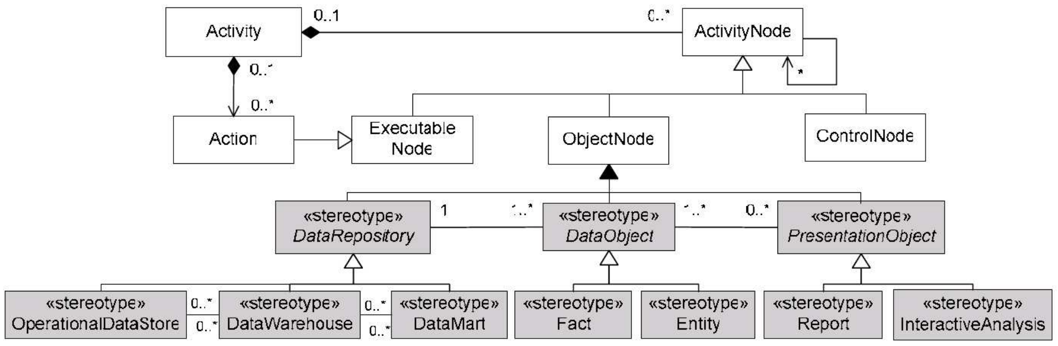


Figure 4.23: Extending the UML2 Metamodel with Stereotypes for BI Objects

4.3.1.3 Presentation Objects

In an organization employing BI techniques, there are usually tools and applications providing users with prepackaged information that has been compiled for them. We call these collections of information Presentation Objects, and have identified two different types: Report or Interactive Analysis. A report displays a predefined set of queries, for example a report on sales in the south region for the 4th quarter of 2004. The values contained in a report do not change over time. An interactive analysis is a tool, e.g., an OLAP tool. In this case, the queries or analysis operations are not predefined but can be chosen by the user. The values are regularly updated and can be used for continuous performance monitoring. In a business process model it is possible to for instance show a certain report that is accessed by an activity.

4.3.2 The UML Profile for Business Intelligence Objects

UML offers a possibility to extend and adapt its metamodel to a specific area of application through the creation of profiles. UML profiles are UML packages with the stereotype «profile». A profile can extend a metamodel or another profile [OMG05c] while preserving the syntax and semantic of existing UML elements. It adds elements which extend existing classes. UML profiles consist of stereotypes, constraints and tagged values.

A stereotype is a model element defined by its name and by the base class(es) to which it is assigned. Base classes are usually metaclasses from the UML metamodel, for instance the metaclass «Class», but can also be stereotypes from another profile. A stereotype can have its own notation, e.g., a special icon.

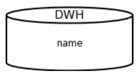
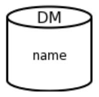
Constraints are applied to stereotypes in order to indicate restrictions. They specify pre- or post conditions, invariants, etc., and must comply with the restrictions of the base class [OMG05c]. Constraints can be expressed in any language, such as programming languages or natural language. We use the Object Constraint Language (OCL) [OMG05b] in this profile, as it is more precise than natural language or pseudocode, and widely used in UML profiles.

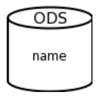
Tagged values are additional metaattributes assigned to a stereotype, specified as name-value pairs. They have a name and a type and can be used to attach arbitrary information to model elements.

We extend the UML 2 activity diagram with a UML Profile for BI Objects, creating an integrated model of processes and BI objects to make the relationship between the data warehouse environment and the business processes more transparent. Activity diagrams are used in UML for modeling processes, workflows, and computations. In Figure 4.23 shows a part of the UML 2 metamodel related to activity diagrams (light) to illustrate how the stereotypes designed (dark) fit into to the existing metamodel.

In an UML 2 activity diagram, a single activity, representing a process or part of a process, is modeled. An activity may include any number of activity nodes, such as individual actions, control nodes (e.g., splits and joins), and object nodes. These nodes can be arranged to form sequential or concurrent processes, and several activity diagrams can be connected to describe larger processes.

Table 4.2: Specification of Stereotypes: Data Repositories

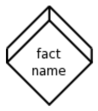
Name	***DataRepository***	
Base class	ObjectNode	
Description	A type of database used in data warehouse environments.	
	Abstract.	
Specializations	DataWarehouse, DataMart, and OperationalDataStore	
Tag Definition	**isMultidimensional**	
	Type: Boolean, Multiplicity: 1	
	Description: Indicates whether the data model of the DataRepository is a multidimensional data model	
Constraints	A DataRepository must be related to at least one DataObject:	
	`context DataRepository inv:`	
	`self.dataObject->size() >= 1`	
Name	**DataWarehouse**	
Generalization	DataRepository	
Description	A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management's decision-making process [IH94]	
Name	**DataMart**	
Generalization	DataRepository	
Description	A data mart is a departmental subset of a data warehouse focused on a single subject area [CD97].	

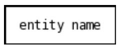
Name Generalization Description	**OperationalDataStore** DataRepository An operational data store is located between the operational systems and the data warehouse [GRC04].	

In the UML Profile for BI Objects, the class Object Node is the base class of all stereotypes. The OMG has defined an object node as an "activity node that indicates an instance of a particular classifier, possibly in a particular state, may be available at a particular point in the activity" [OMG05c]. Therefore, object nodes represent concrete instances of information objects, which are input or output parameters of an activity. They are suited for the purpose of showing when a (sub-)process accesses a BI object, as the BI objects amount to input parameters of activities.

Table 4.3: Specification of Stereotypes: Data Objects

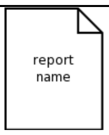
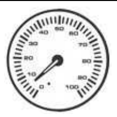
Name
Base class
Description
 	
Specializations
Constraints
	DataObject
	ObjectNode
	A data object is part of the data model contained in a data repository. Abstract.
	Fact, Entity
	A DataObject must belong to exactly one DataRepository:
	`context DataObject inv:`
	`self.dataRepository.size() = 1`
	The corresponding class must have at least one attribute:
	`context DataObject inv:`
	`self.type.allAttributes()->size() >= 1`	

Name
Generalization
Description
 	
Constraints
	Fact
	DataObject
	A fact is a data object of a multidimensional data model.
	The DataRepository containing a fact must have a multidimensional data model:
	`context Fact inv:`
	`self.isType(Fact) implies`
	`self.dataRepository.isMultidimensional`
		

Name
Generalization
Description
Constraints
	Entity
	DataObject
	An entity is a data object of an E/R model.
	The DataRepository containing an entity must not have a multidimensional data model:
	`context Entity inv:`
	`self.isType(Entity) implies not`
	`self.dataRepository.isMultidimensional`
		

As described in the metamodel in Section 4.3.1, BI objects can be classified into three larger types. We therefore have defined three abstract top-level stereotypes, «DataRepository», «DataObject», and «PresentationObject». The stereotypes «DataWarehouse», «OperationalDataStore» and «DataMart» are derived from «DataRepository». Their specifications are listed in Table 4.2. The stereotype «DataObject» can be further specialized into «Fact» and «Entity», as shown in Table 4.3. Finally, the stereotypes «Report» and «InteractiveAnalysis» are specializations of «PresentationObject», as listed in Table 4.4. The semantics of the individual elements were described in greater detail in Section 4.3.1.

Table 4.4: Specification of Stereotypes: Presentation Objects

Name	***PresentationObject***	
Base class	ObjectNode	
Description	A presentation object is a document or tool used to present information to a user. Abstract.	
Specializations	Report, InteractiveAnalysis	
Constraints	A PresentationObject must have at least one DataObject: `context PresentationObject inv:` `self.dataObject->size() >= 1`	
Name	**Report**	
Generalization	PresentationObject	
Description	A report displays the results of a predefined set of queries.	
Name	**InteractiveAnalysis**	
Generalization	PresentationObject	
Description	An interactive analysis is a tool that allows the user to freely explore information.	

4.3.3 Examples

We present three examples that demonstrate the application of the UML Profile for BI Objects developed in Section 4.3.2, each illustrating a different aspect. The first example introduces a simple UML 2 activity diagram with BI objects, the second example illustrates how UML «selection» notes can be used in combination with BI objects to provide more detail on data access, and the third example demonstrates how a more complicated business process can be modeled on a higher level of abstraction.

The example activity diagram in Figure 4.24 describes the well-known process of a passenger checking in at an airport. Two parties are involved in this activity, the passenger and the check-in desk. The process starts with the action "present documents": the passenger presents the travel documents at the check-in desk. Two items, the ticket and the passport, are passed to the "check identity" action performed by the check-in desk. In order perform its task, the action also needs access not only to the two documents but also to the entity

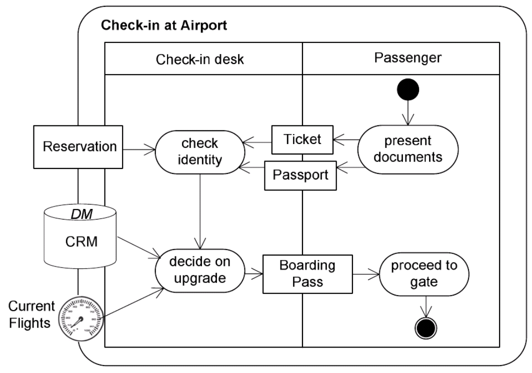


Figure 4.24: Airport check-in business process

"reservation". Therefore, it only starts if all three necessary inputs are available. After the identity check has concluded, the check-in desk decides on a possible upgrade. The action "decide on upgrade" needs data from the Customer Relationship Management (CRM) data mart. The data mart contains the frequent flyer status of the passenger in question. Data on the current flights situation, (e.g., whether another flight to the same destination is cancelled or overbooked, meaning that no upgrades are available) is provided by an interactive analysis tool. The "decide on upgrade" action therefore can only begin when the identity check has concluded and the two BI objects are available. It produces a boarding pass as output. The passenger can proceed to the gate as soon as he or she has received the boarding pass. Alternative paths, such as the identity check failing, were left out for sake of clarity of the example.

A large business process can be modeled by linking together several activity diagrams, each describing a small sub-process, such as the part of the process of designing and organizing a promotion of a single product (e.g., a 30 percent discount on a brand of soap) shown in Figure 4.25. In the initial step of choosing the product, a report on past promotions is analyzed in order to identify products suitable for a profitable promotion. Therefore, the action "analyze past promotions" has a set of products, e.g., those that seem promising, as output. In the following "choose product" action, a product is chosen based on how many items of the product were sold in the past (i.e., the sales information provided by the "Sales" fact) and whether enough items are on stock (i.e., inventory information from the ODS system). Only data on the products selected before should be read from the fact table and the ODS. In an activity diagram, a «selection» note attached to the object flow between an object node and an action can be used to specify selection behavior. In the example presented here, the OCL statement checks whether a product in the BI object – the "Sales fact" or the ODS – is contained in the list of promising products.

During a fraud detection process at an insurance company (Figure 4.26), insurance claims

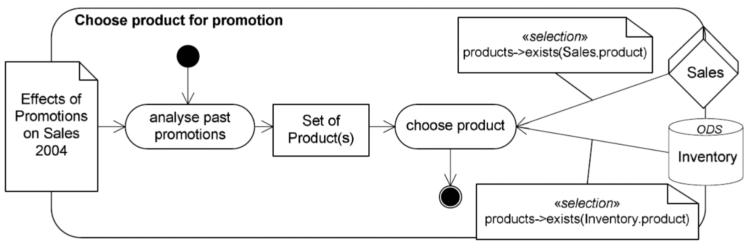

Choose product for promotion
Effects of Promotions on Sales 2004
analyse past promotions
Set of Product(s)
choose product
«selection» products->exists(Sales.product)
Sales
ODS
Inventory
«selection» products->exists(Inventory.product)

Figure 4.25: Product promotion: sub-process of choosing the product

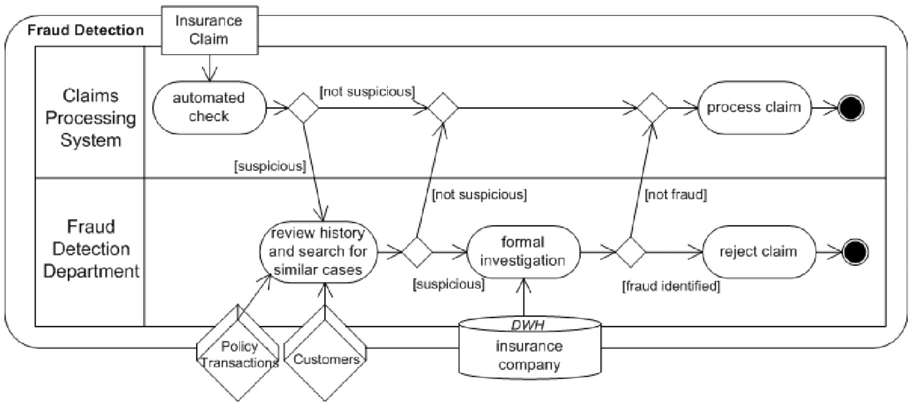


Figure 4.26: Fraud detection business process

are subjected to a three-step analysis, aimed at recognizing all potentially fraudulent claims before they might be processed and paid. The activity "fraud detection" is started by the arrival of an insurance claim. The claim is first exposed to an extensive automated check by the claim processing system. All claims judged as being suspicious are forwarded to the fraud detection department, whereas the others are processed normally. The suspicious claims are then reviewed. In this action the results of the automated check as well as the history of the customer and the insurance policy are analyzed, to identify patterns and/or similar cases. Therefore, the action needs access to two fact tables: "Customers" and "Policy Transactions". The claims that continue to be suspect are then formally investigated, whereas the claims re-established as genuine are returned to the claim processing system. The action "formal investigation" represents a thorough search for further clues in order to provide answers to any open questions. As the queries necessary in this step are different in every case and cannot be predicted, the action requires the whole data warehouse of the insurance company as input. All claims finally identified as fraudulent are rejected.

4.4 Related Work

There are a lot of conceptual models available for business processes, as well as for databases or data warehouses. But there are no models available that focus on the relationship between these two domains. The conceptual data warehouse diagrams available for the different stages of the data warehouse process, e.g., for multidimensional models [LMTS06] or ETL processes [TLM03], do not address the link to business processes at all. Business process diagrams that address the static structure of databases do not address the particularities of data warehouses and BI.

Event-Driven Process Chains (EPC) [KNS92] incorporate a data view, targeting operational data bases. To provide the data view with a conceptual model, Chen's entity-relationship (ER) model was adopted, since it was the most widespread model in the area of data modelling. Today, the UML class diagram is also used. EPC functions perform read or write operations on E/R entities or UML classes. The models presented in this section are based on a similar concept, but account for the particularities of data warehouse settings.

In UML 2 activity diagrams [OMG05c], data store nodes represent data. A UML 2 action node can perform read or write operations, comparable to the EPC function. The data store node is not necessarily linked with a UML class or database.

The Business Process Modeling Notation (BPMN) [BPM04] provides data objects, which are used and updated during the process. The data object can be used to represent many different types of object, both electronic or physical.

4.5 Concluding Remarks

This chapter has addressed the missing link in conceptual modeling between between the static structures of the data warehouse and BI and the dynamic structures of business processes. To bridge this gap, two business process modeling languages were extended with an additional components, the Business Intelligence (BI) Perspective for the Event-Driven Process Chain (EPC) and the UML Profile for Business Intelligence (BI) Objects for UML 2 Activity Diagrams.

With these models, it becomes possible to create models that show where and how business processes use decision support data, as well as which parts of the processes depend on which information from the data warehouse environment, and how an active data warehouse may impact on the business process control flow.

The BI Perspective for EPCs in divided into the Traditional BI Perspective and the Active BI Perspective, the latter addressing the specific features of active data warehouses. In the Traditional BI Perspective, elements representing the different types of data repositories that are accessed, as well as representing the data model of a certain repository, and additionally elements representing the means of presentation have been designed and incorporated into a metamodel. The Active BI Perspective adds to EPCs the concept of external BI processes that may influence the control flow of business processes. Both parts of the BI perspective have been tested with example business processes.

The UML Profile for BI Objects is specified in terms of several types of BI objects, representing the different types of data repositories, their data models and the means of presentation. These BI objects can be accessed by actions of UML 2 activity diagrams. The models were applied to several example processes.

Chapter 5

Business Object States and the Data Warehouse

Data warehouse systems allow to analyze business objects relevant to an enterprise organization (e.g., orders or customers). Analysts are interested in the states of these business objects: A customer is either a potential customer, a first time customer, a regular customer or a past customer; purchase orders may be pending or fullfilled.

Business objects and their states can be logically distributed over many parts of the data warehouse, and appear in measures, dimension attributes, levels, etc.

Surprisingly, this knowledge – how business objects and their states are represented in the data warehouse – is not made explicit in existing conceptual models. There is a need to make this relationship more accessible.

This chapter introduces the *UML Profile for Representing Business Object States in a Data Warehouse*. It makes the relationship between the business objects and the data warehouse conceptually visible.

Contents

5.1 Introduction

The data offered by a data warehouse describes business objects relevant to an enterprise organization (e.g., accounts, orders, customers, products, and invoices) and allows to analyze their status, development, and trends. *Business objects*, also known as domain objects, represent entities from the "real world enterprise" and should be recognizable to business people, as opposed to implementation objects (e.g., menu items, database tables).

Business objects can be characterized by the states they have during their lifecycle. For example, a customer can have different states: There are potential customers, first time customers, regular customers, customers who pay their bills on time, fraudulent customers, high volume customers, past customers, etc.

In many organizations today, data warehouses have grown over time and become large and complex. Business objects and their states can be found all over the data warehouse: Data relevant to analyzing, e.g., customers may be distributed over many parts of the data warehouse, and the different states of the customer may appear in many different ways, i.e., as measures, dimensional attributes, levels, etc. A business object can easily have 20 states, which might be hidden in just as many facts or even more: Potential customers in a contacts fact of the marketing data mart, regular customers in a sales fact, customers who pay on time in a payment transaction fact, etc.

Surprisingly, this knowledge - how business objects and their states are represented in the Data Warehouse - is not made explicit in existing conceptual models. When new analysis requirements appear, it is often difficult, time consuming and costly to find out whether the information about a certain state of a business object is already contained somewhere in the data warehouse, or whether the data model of the data warehouse has to be extended or changed.

The goals therefore are to

- make the relationship between the business objects that the uses want to analyze and the data warehouse visible and accessible
- show where the business objects and their states can be found in the data warehouse
- offer a new perspective on data warehouses, which emphasizes business objects and their states instead of solely fact and dimension tables
- show how the business object states relate to the data model

These goals are achieved by introducing the *UML*[1] *Profile for Representing Business Object States in a Data Warehouse* (Section 5.4), which makes the connection between Data Warehouses and business object states visible. State machines (Section 5.2) describe how objects change states in reaction to events, and are suitable for capturing business logic. This

[1]Unified Modeling Language

approach relates state machines of business objects to data warehouse elements and use them as a new viewpoint on Data Warehouses. Additionally, using the UML Profile, 14 correspondence patterns between state machines and data warehouse elements (Section 5.5) were defined.

The contributions of the UML Profile for Representing Business Object States in a Data Warehouse along with the correspondence patterns are:

- It provides a straightforward way to make visible where a business object such as a customer is available in the data warehouse and can be analyzed, and how its various states correspond to facts, dimensions and measures.
- Through the business objects, it offers a bigger picture as it links the needs of the business organization to the data warehouse that stores data on business objects for analytical purposes.
- The Profile for Representing Business Object States in a Data Warehouse allows modeling on several levels of detail and thus enables the modeler to choose the right level of detail for different purposes or target audiences. A high level overview model shows only the whole business objects and how they are related to facts and dimensions, where as a more detailed model can show measures, dimension attributes and hierarchy levels for the individual states of the object.
- By extending standard UML 2.0 state machines, the UML profile offers reuse of a well-known notation as well as tool reuse, avoiding costs of learning a new notation or additional tools.
- The models allow locating business object states in the data warehouse, and thus recognizing cases where business object states are not available at all, which may indicate business requirements that are not yet addressed. If the model shows that a business object state is available at several locations at the same time, it can be checked whether this was a deliberate design decision or whether this indicates a problematic situation.
- Together with the correspondence patterns the models can support the design phase of a data warehouse project, as they provide hints on possible facts, measures and dimensions that can be derived from the business object states.

Large parts of this chapter have been published in [SL07b].

5.2 Modeling Business Objects with UML State Machines

UML 2.0 state machines are used here to model business object states. State machines in general describe “the possible life histories of an object” [RJB04]. A state machine comprises

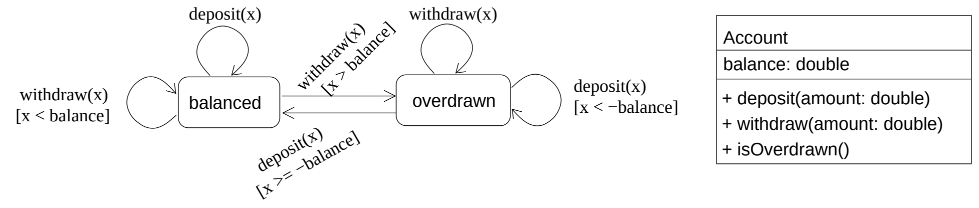
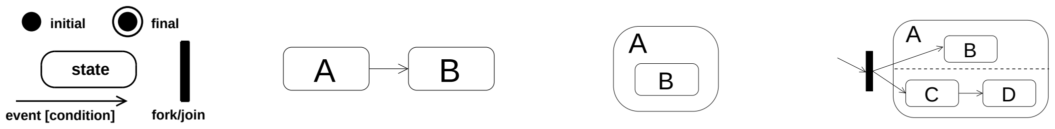


Figure 5.1: State machine for Account objects



(a) Main notation elements (b) $A \vee B$ (c) $A \wedge B$ (d) $A \wedge (B \wedge (C \vee D))$

Figure 5.2: UML State Machines: Syntax and combinations of states

a number of states, interconnected by transitions. Events trigger the transitions. There may be several transitions for one event, and they may be guarded by conditions (see Figure 5.1).

State machines are used for example in Software Engineering to achieve higher quality software. During the analysis phase of a software development process, state machines for the main business objects (account, order, etc.) are modeled. Such a state machine cannot be transformed directly to code, but is very useful for designers as it allows to recognize "illegal" or conceptually impossible state transitions and thus assess correctness of the final software product. The UML Profile for Representing Business Object States in a Data Warehouse presented in this chapter aims at bringing the power of state machines to Data Warehousing.

Figure 5.2 gives an overview of state chart elements and their use in UML 2.0. States may be nested in other states (c), and the so-called sub-machines can be divided into regions (d). This allows access to the model on various levels of detail. Figure 5.3 shows an example state machine. The object modeled here is the contract between a telecommunications provider and the user of a post-paid mobile phone.

Before it is signed, the contract is in the state "potential". After the customer has signed it, it is registered and a credit check is performed. During this phase, it may become "cancelled", if the check fails. Otherwise, it the credit is OK, the overall state is now called "current", which is a composite state that contains a lot of detail in terms of substates. If the contract is never signed, it becomes "past" after one year.

"Current" contracts are initially "normal", "fully paid", "active", and "new", as indicated

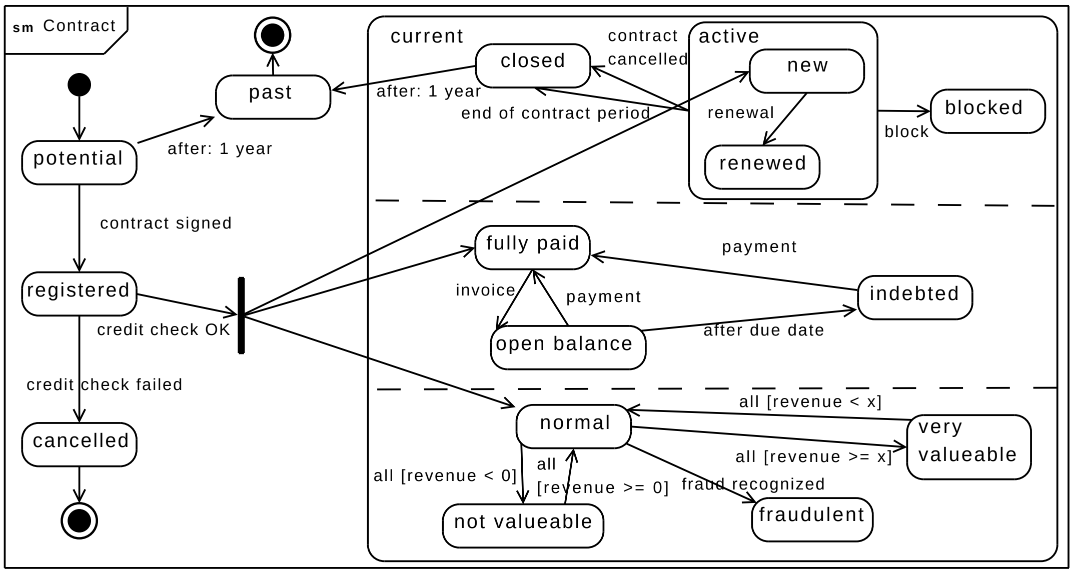


Figure 5.3: State Machine diagram for a post-paid mobile phone contract

by the transition arcs entering the composite state. The three regions within “current” are concurrent and orthogonal, meaning that at each point in time the object has a valid state in each of the three regions, and that what happens in one region does not influence the other regions.

In the first region, the contract starts as “active” and can become “closed”, either because the contract period ends or because the contract is cancelled beforehand. If a contract is blocked for some reason, it is neither active nor closed. Active contracts start as “new” and can become “renewed”. Closed contracts are moved from “current” to “past” after one year.

The middle region relates to the financial aspects of the contract: Each time an invoice is issued for a contract, it moves from “fully paid” to “open balance”, and when the payment for the invoice arrives, it moves back again. If an open balance is not paid until its due date, the contract is “indebted”. Issues regarding the amounts of the payments are not shown for sake of clarity (cf. Figure 5.1).

In the last region, the movements of the contract object between the states depend mostly on certain conditions becoming true: If the revenue gained from this contract supercedes the amount x, it becomes “very valueable”, meaning that this customer is very valueable to the company. Whereas, if the revenue falls below 0, keeping this contract is actually causing financial damage (i.e., by producing higher costs than monthly payments). The fourth state in this region is “fraudulent”: It indicates that fraud connected to this contract has been discovered.

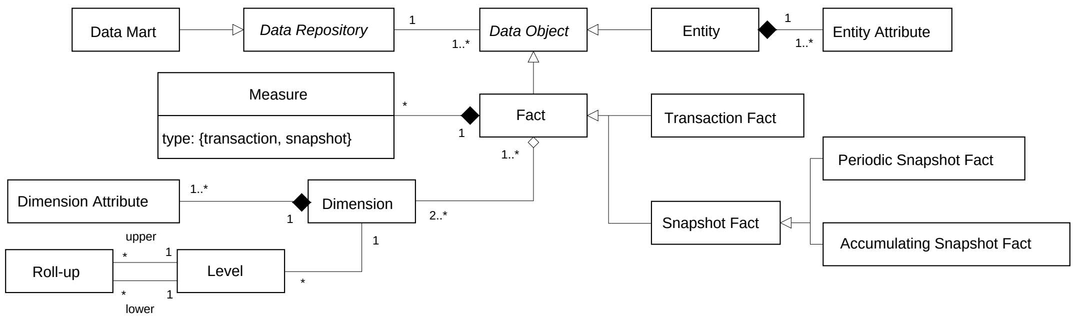


Figure 5.4: Data warehouse elements used to represent states of business objects

5.3 Business Objects in the Data Warehouse

The aim of this approach is to create conceptual models for making visible where business objects and their states are available in the data warehouse. Before UML Profile can be constructed, the data warehouse elements to be used have to be identified. This section contains a metamodel of these elements, an overview of the correspondences between the data warehouse elements and business object states, the user groups that the UML Profile is aimed at, and finally a tabular overview of the correspondence patterns.

5.3.1 Metamodel

Figure 5.4 shows a metamodel of the elements that may represent business objects in a data warehouse. There are different kinds of *data repositories*, one of which may be a *data mart*. Other subtypes of data repositories [SLK05] are not used in this chapter for sake of simplicity. Data repositories contain *data objects*, which are *facts* or *entities*, depending on the type of the repository. Entities have *entity attributes*, whereas facts may have *measures*. Each fact consists of at least two *dimensions*, which may have several *levels* connected to each other via *roll-up* relationships. Dimensions are described by *dimension attributes*. Facts come in different types: *Transaction facts* or *snapshot facts*, of which there are two sub-types, *periodic* and *accumulating*.

The elements data repository, data mart, fact and entity of this metamodel (and also the UML Profile) are based on previous work by the authors. See [SLK05] for a more detailed description.

This metamodel is related to the models presented in Section 4.3.1 and by [LMTS06].

5.3.2 Representation in the Data Warehouse

Not all elements of this metamodel are useful for all conceptual modeling needs. A number of correspondence patterns were identified along with the main usage scenarios of this approach. They are grouped into three levels and one additional category: the *overview* level, the *fact* level, the *attribute* level, and finally correspondences related to *aggregation and classification*. This subsection gives an overview; details and examples are given in Section 5.5.

Overview To show at a glance, where a business object can be found in the data warehouse, it can be linked as a whole to several data marts, fact tables, dimension tables or entities.

Fact Level The relationship of transitions and states of business objects to different types of fact tables can be shown on this level. As Kimball described in [Kim99], there are three main types of measurements, which are the fundamental grains of fact tables: *transaction*, *periodic snapshot*, and *accumulating snapshot*. A *transaction fact* corresponds to a *transition* between states, whereas *snaphot facts* correspond to *states*.

Attribute Level Showing even more detail, a state of a business object may be represented by dimensions or measures, whereas the transactions and their guard conditions are found in measures or dimension attributes.

Aggregation and classification Guard conditions on state transitions may also appear in aggregation and classification hierarchies. Nested states correspond directly to roll-up relationships.

5.3.3 User Groups

The conceptual model presented here is designed to answer the needs of two separate types of users:

Data warehouse managers and *business users* are looking for the big picture, an overview of what to find where. They will use models showing business objects with correspondences mainly on the *overview level* (see Section 5.5.1 below). This allows them to find answers to questions such as "Which business object is in which data mart?", "Is there a fact that corresponds to this business object?", "If I have this dimension, does it represent this business object?"

Data warehouse designers and developers need to understand the details of how business objects states and transactions between them can be represented in a data warehouse. They will use a fine-grained state machine model of a business object mapped to a data model. The *correspondence patterns* identified by us give hints on which elements can be used in the

Table 5.1: Correspondence patterns

	Fact	Transaction F.	Snapshot F.	Measure	Dimension	D. Attribute	Level	Roll-up	Entity	E. Attribute	Data Mart
Object of State Machine	X				X				X		X
State			X	X		X	X			X	
Transition		X							X		
Guard condition				X		X	X			X	
Nested States								X			

data warehouse model to represent the characteristics of the business objects, thus improving the creation and evolution of data warehouse models.

5.3.4 Correspondence Patterns

Based on an analysis of the typical requirements of the user groups, Table 5.1 gives an overview of which elements of the UML Profile may be linked to which in the conceptual model to show how business objects (elements on horizontal lines) are represented in the data warehouse, and which elements of the data warehouse (vertical columns) can be used to represent the states of the business objects (further details of the correspondence patterns in Section 5.5).

5.4 UML Profile for Representing Business Object States in a Data Warehouse

This section introduces the UML Profile for Representing Business Object States in a data warehouse. It provides an easy to use yet formally founded way to model the correspondences between business object states and data warehouses.

The Unified Modeling Language (UML) can be extended and adapted to a specific application area through the creation of profiles [OMG05c]. UML profiles are special UML packages with the stereotype ≪`profile`≫. A profile adds elements while preserving the syntax and semantic of existing UML elements. It contains stereotypes, constraints and tag definitions.

A *stereotype* is a model element defined by its name and by the base class(es) to which it is assigned. Base classes are usually metaclasses from the UML metamodel, for instance the metaclass `Class`. A stereotype can have its own notation, e.g., a special icon.

Constraints are applied to stereotypes in order to enforce restrictions. They specify pre- or postconditions, invariants, etc., and cannot override the restrictions of the base class [OMG05c]. The Object Constraint Language (OCL) [OMG05b] is used here, which is widely

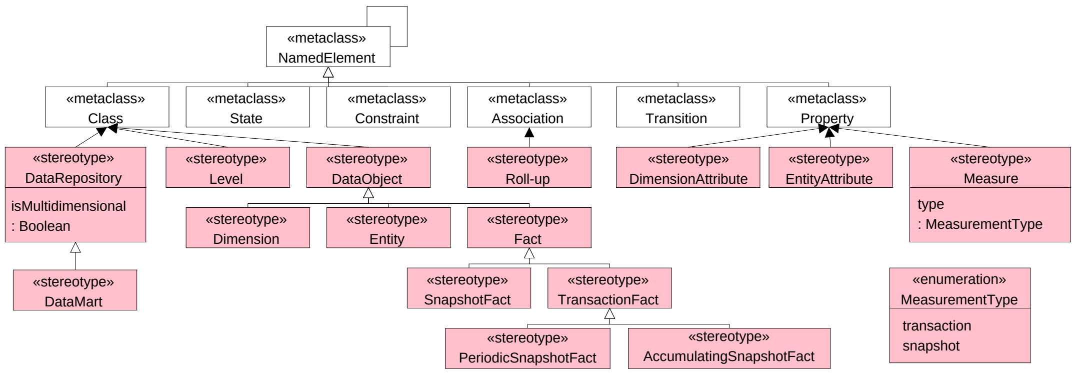


Figure 5.5: UML stereotypes for representing business object states in a data warehouse

used in UML profiles to define constraints in this profile, but any language, such as a programming language or natural language, may be used.

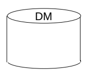
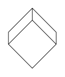
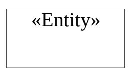
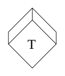
Tag definitions are additional attributes assigned to a stereotype, specified as name-value pairs. They have a type and can be used to attach arbitrary information to model elements.

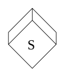
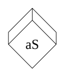
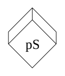
The UML Profile for Representing Business Object States in a Data Warehouse makes it possible to model how the data warehouse used to analyze business objects is related to the lifecycle and states of these objects. Its stereotypes are described in detail in Table 5.2. These stereotypes can be used directly in UML State Machine diagrams [OMG05c]. Figure 5.5 shows a part of the UML 2 metamodel (light) to illustrate how the stereotypes designed (dark) fit into to the existing UML metamodel. The relationships between the stereotypes correspond to the metamodel presented in Section 5.3.

`Class`, `Association`, and `Property` are the base classes for the stereotypes. This allows us to model their relationships with elements used in state machines such as `State`, `Transition` and `Constraint`.

Table 5.2: Stereotype definitions (see also Figures 5.4 and 5.5)

Name	**DataRepository**
Base class	Class
Description	A data repository represents a type of database used in data warehouse environments.
Tag Definition	isMultidimensional Type: AuxiliaryConstructs::PrimitiveTypes::Boolean Multiplicity: 1 Description: Indicates whether the data model of the DataRepository is a multidimensional data model
Constraints	A DataRepository must be related to at least one DataObject: `self.dataObject->size() >= 1`

Name	**DataMart**	
Generalization	DataRepository	
Description	A data mart is a departmental subset of a data warehouse focused on a single subject area.	
Name	**DataObject**	
Base class	Class	
Description	A data object is part of the data model contained in a data repository. The stereotypes Fact and Entity are derived from DataObject.	
Constraints	A DataObject must belong to exactly one DataRepository: `self.dataRepository.size() = 1`	
Name	**Fact**	
Generalization	DataObject	
Description	A fact is a data object of a multidimensional data model.	
Constraints	The DataRepository containing a fact must have a multidimensional data model: `self.oclAsType(DataObject) \ .dataRepository.isMultidimensional = true` The Fact must have at least two Dimensions: `self.dimension->size() >= 2`	
Name	**Entity**	
Generalization	DataObject	
Description	An entity is a data object of an E/R model.	
Constraints	The DataRepository containing an entity must not have a multidimensional data model: `not self.oclAsType(DataObject) \ .dataRepository.isMultidimensional` The Entity has at least one EntityAttribute: `self.entityAttribute->size() >= 1`	
Name	**TransactionFact**	
Generalization	Fact	
Description	A TransactionFact contains measures of transactions.	
Constraints	A TransactionFact may only contain transaction measures. `self.allAttributes->forAll(a | a.oclIsTypeOf(Measure) implies a.type=transaction)`	

Name	**SnapshotFact**	
Generalization	Fact	
Description	A Snapshotfact contains snapshot measures, e.g., inventories.	
Constraints	A SnapshotFact only contains snapshot measures. `self.allAttributes->forAll(a | a.oclIsTypeOf(Measure) implies a.type=snapshot)`	
Name	**AccumulatingSnapshotFact**	
Generalization	SnapshotFact	
Description	The measures of an entry in the AccumulatingSnapshotFact are gathered over time	
Name	**PeriodicSnapshotFact**	
Generalization	SnapshotFact	
Description	The measures of a PeriodicSnapshotFact are acquired periodically for all instances	
Name	**Measure**	«Measure»
Base class	Attribute	
Description	A numeric Measure is the object of analysis.	
Tag Definition	measurementType Type: MeasurementType (Enumeration) Multiplicity: 1 Description: Indicates the type of measurement: *transaction* or *snapshot*	
Constraints	A Measure must belong to exactly one Fact: `self.fact.size() = 1`	
Name	**EntityAttribute**	«Entity−Attribute»
Base class	Attribute	
Description	An attribute to an Entity.	
Constraints	An EntityAttribute belongs to exactly one Entity: `self.entity.size() = 1`	
Name	**Dimension**	«Dimension»
Base class	Class	
Description	Dimensions provide context for the measures and together are assumed to uniquely determine them.	
Constraints	A Dimension is used by at least one Fact: `self.fact->size() >= 1` A Dimension has at least one DimensionAttribute: `self.dimensionAttribute->size() >= 1`	
Name	**DimensionAttribute**	«Dimension−Attribute»
Base class	Attribute	
Description	DimensionAttributes describe Dimensions.	
Constraints	A Dimension Attribute belongs to exactly one Dimension: `self.dimension->size() = 1`	

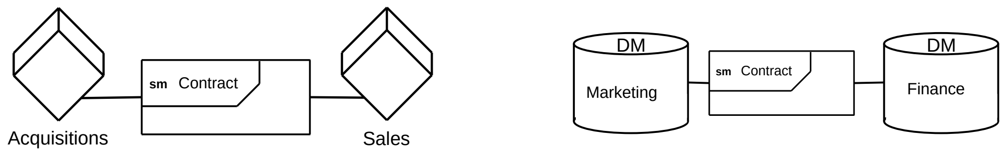
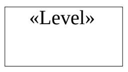


Figure 5.6: The business object "Contract" in the fact "Sales" (left) and data marts (right)

Name Base class Description Constraints	**Level** Class Levels of Dimensions are used to aggregate Measures. A Level belongs to exactly one Dimension: `self.dimension->size() = 1`	

Name
Base class
Description
 	
Constraints
	Roll-up
	Association
	A Roll-up-Association between two Levels indicates that Measures can be aggregated from the lower to the upper Level.
	A Roll-up-relationship has exactly one upper and one lower Level:
	`self.upper->size() = 1 and`
	`self.lower->size() = 1`	

5.5 Correspondence Patterns and Examples

To create conceptual models for making visible where business objects and their states are available in the data warehouse, elements from data warehouse conceptual data models are linked with state machines of business objects. This section applies the correspondence patterns already introduced in Section 5.3 to examples.

5.5.1 Overview level

The business object can be linked to one or more *data marts* to provide an overview (Figure 5.6a). Then it is possible to identify where each business object can be analyzed in the data warehouse model. A business object may correspond to a *fact table* (e.g., order, shipment, account; Figure 5.6b), a *dimension table* (e.g., product, customer, account), or in the case of a pure E/R-model to an *entity*.

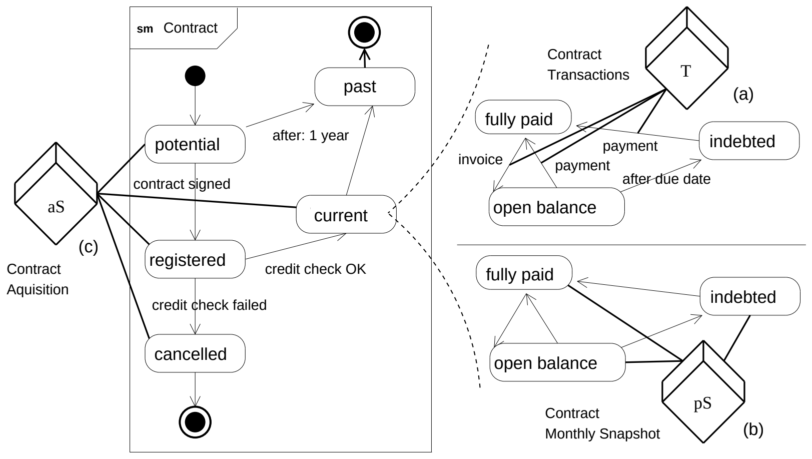


Figure 5.7: Transitions in transaction facts (a), states in snapshot facts (b, c)

5.5.2 Fact level

There are three main types of measurements or fundamental grains of fact tables: *transaction*, *periodic snapshot*, and *accumulating snapshot* [Kim99]. Looking at state charts, it can be seen that a *transaction fact* corresponds to a *transition* between states (Figure 5.7 (a)), whereas *snaphot facts* correspond to *states*. Periodic snapshots record the current state for all instances of a business object at regular intervals (b), whereas accumulating snapshots "follow" each instance as it passes from state to state, adding values to the record over time (c). In the latter case, the order in which the values are added must conform to the state chart.

5.5.3 Attribute level

To provide a more detailed conceptual model, i.e., for data warehouse design and development, this level contains correspondence patterns related to all kinds of attributes (measures, dimension attributes, entity, attributes, guard conditions).

5.5.3.1 States

A state in a state chart may play several roles in the data warehouse model. The most straightforward case is when there is an explicit "status" dimension (e.g., for account facts, insurance policies, etc.). In this case, several states of a business object are modeled by the dimension, i.e., as dimension attributes. Second, a state may be modeled as a measure. For a single state, this measure is of type boolean (either the object is in this state of not), and for several states, the measure would be an enumeration with the values corresponding to the states (which may be seen as a degenerate type of the status dimension).

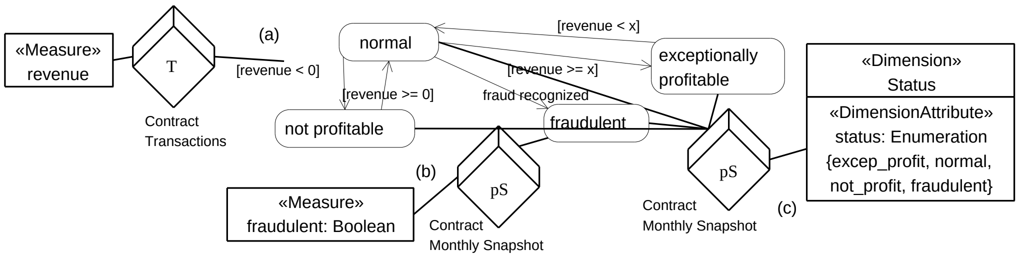


Figure 5.8: Guard condition as measure (a), states as measure (b) or dimension (c)

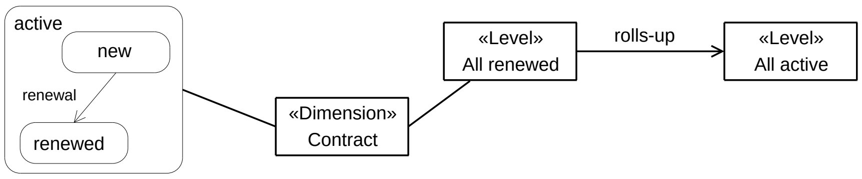


Figure 5.9: Nested states of a business object correspond to a roll-up relationship

5.5.3.2 Transitions

Transitions in state charts may be guarded by conditions. These conditions often are found in the data warehouse as measures or dimension attributes. If a guard condition contains a recognizable variable (e.g., "revenue" in Figure 5.8(a)), this variable can turn into a measure in the data warehouse model.

5.5.4 Aggregation and Classification

State charts of business objects may also provide hints regarding aggregation and classification hierarchies. First, nested states correspond to roll-up relationships, the inner state being the special case and the outer state the more general. Also, guard conditions may define levels in the hierarchy, as they separate instances into groups. In the case of a specific "status" dimension mentioned above, the states covered by this dimension then may also correspond to levels.

5.6 Related Work

Using UML profiles to model the structure and behavior of data warehouses as well as related aspects such as ETL processes has become increasingly popular, also due to the rise of the Model-Driven Architecture (MDA [OMG03b]).

Trujillo and Luján-Mora introduced a Data Warehouse design framework in [LMT04a], which is supported among others by the UML Profile for Modeling ETL processes [TLM03], a profile for Physical Modeling of Data Warehouses [LMT04b], and a profile for Multidimensional Modeling [LMTS06].

UML has also been applied to aspects such as data warehouse security. Fernandez-Medina et al. have extended UML for Designing Secure Data Warehouses [FMTVP04].

In [MTSP05a], Mazon et al. show how the MDA can be applied to Data Warehousing. The contribution widens the scope of conceptual modelling in Data Warehousing to include more than structural models.

5.7 Concluding Remarks

This chapter has addressed that fact that there are no existing models for describing how business objects and their states are represented in a data warehouse. Business object states can be distributed over many parts of the data warehouse, and appear in measures, dimension attributes, levels, etc. This information is needed e.g., when new analysis requirements appear, as it is often difficult, time consuming and costly to find out whether the information about a certain state of a business object is already contained somewhere in the data warehouse, or whether the data model of the data warehouse has to be extended or changed.

The *UML Profile for Representing the Lifecycle of Business Objects in a Data Warehouse* was introduced in this chapter. It makes the relationship between the business objects and the data warehouse visible and accessible, as it allows to model various elements of data warehouses and their data models in combination with UML 2.0 state machines used to model the lifecycle of business objects. The models offer several levels of detail, and by using UML they provide a well-known notation and can be created with many modeling tools. Using the UML profile, identified 14 correspondence patterns between state machines and data warehouse were identified. The UML profile and the correspondences are applied to an example.

Chapter 6

Data Warehouse Usage

Data warehouse systems represent a single source of information for analyzing the status, the development and the results of an organization.

Today's data warehouse systems provide many different services to different kinds of users. People involved in designing and managing data warehouse systems need to see the big picture of how the data warehouse is being used, to have an overview of the current situation, and to be able to visualize future scenarios. Currently, there is a lack of such general models in Data Warehousing.

This chapter introduces the UML Profile for Modeling Data Warehouse Usage for modeling the different kinds of data warehouse usage on a conceptual level. It uses features of UML intended for the purpose of creating abstract, general models. The profile distinguishes four perspectives of usage, and allows to model details of the users. The UML Profile is applied to examples illustrating some of the application scenarios.

Contents

6.1 Introduction

Today's data warehouse systems provide many different services to different kinds of users: Users retrieve summaries and reports relevant to them, or analyze data with specialized visualization tools. The system may send them messages via e-mail or sms, or provide a quick overview visualization in a dashboard or an intranet portal. Users need access to data at different times, some need it occasionally, others more often, suddenly urgently, or regularly and predictably.

People involved in managing, designing or evolving today's data warehouse systems need to see the big picture of all these different ways the data warehouse is being used, in order to have an overview of the current situation, and to be able to visualize future scenarios. Overview diagrams are needed to facilitate communication with users and decision makers.

Surprisingly, today there are no existing models to describe the different aspects of data warehouse usage on a conceptual level. There is a lack of general models that provide a broader view over several aspects, even though there exist many detailed models of subareas. There is a need for a model that shows on the conceptual level:

1. Who are the users and how are they grouped together?
2. Which part of the data warehouse system do they use? How do they use it?
3. How intensely are which parts of the data warehouse being used by which users?
4. When do users need to use which part, and how time critical is it?
5. How important is it?

To fill this gap, this chapter specifies the *UML Profile for Modeling Data Warehouse Usage*. The profile uses some of the lesser known features of UML, intended for the purpose of creating preliminary models with a "less precise but more general representation" [OMG05c]. The features of the profile are grouped into four perspectives, which focus on the most common application scenarios of data warehouse usage modeling (Section 6.3). The *UML Profile for Modeling Data Warehouse Usage* (Section 6.4) offers the following contributions:

- It allows to model who uses the data warehouse, to group the users, and to model their organizational affiliation, skill level, and an approximate number of instances for each user role.
- Modelers can show how often users use something, and how time critical and how important a certain usage is, as well as active or passive usage types.
- The model allows the analysis of the implications of changing scenarios (e.g., adding a component, increasing the number of users) on various levels of detail.
- It can be used to identify critical patterns (many important accesses, rapid growth) and to identify parts of the data warehouse that are not used or not used very often or importantly.

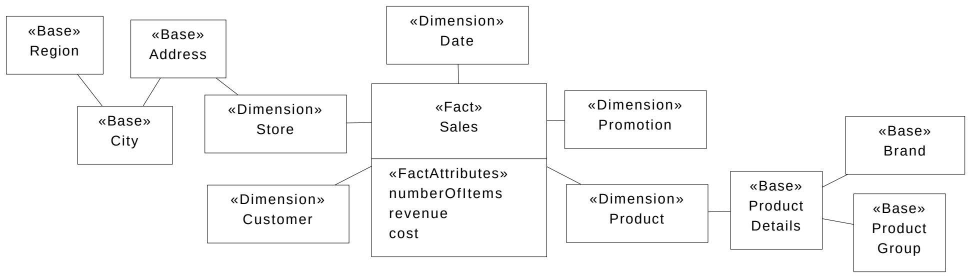


Figure 6.1: A multidimensional model

- Data warehouse usage models can be used to support the design of user access controls or personalized user interfaces.
- In general, the models make the overall structure of data warehouse usage visible on the conceptual level, thus replacing the custom of creating *ad hoc* diagrams and drawings for the communication with users and decision makers.

DWH usage models are intended to provide an overview without aiming at a design process. Compared to requirements analysis in Data Warehousing, the approach to data warehouse usage presented here is broader, and not necessarily focused on a future system to be built. The UML Profile allows to model the users in detail and does not explicitly include (design) goals of any kind. In MDA [OMG03b] terms, this approach is located in the CIM (Computation Independent) area, where models are not necessarily intended to be transformed into code.

Large parts of this paper are published in [SL07a], and Section 6.6 in [SSM$^+$08].

6.2 Background

The approach presented here applies UML to the Data Warehousing domain. It is aimed at encompassing all the different ways that users may use a data warehouse, and providing data warehouse professionals and others with a way to model a current or future system. The goal is to provide an overview over all aspects of data warehouse usage, not only focussing on the data model. Nevertheless, due to the special characteristics of data warehouse data, it is necessary to take the data model especially into account.

The main data model in Data Warehousing is the multidimensional model, also called star schema [CD97]. It is meant to provide intuitive and high performance data analysis [KRRT98]. For details on multidimensional modeling, see Section 2.1.3.1.

For modeling multidimensional data, the UML Profile as described in [LMTS06] is used. Figure 6.1 shows an example. This profile allows to model not only the core features of multidimensional models (facts, measures, and dimensions), but also many advanced features

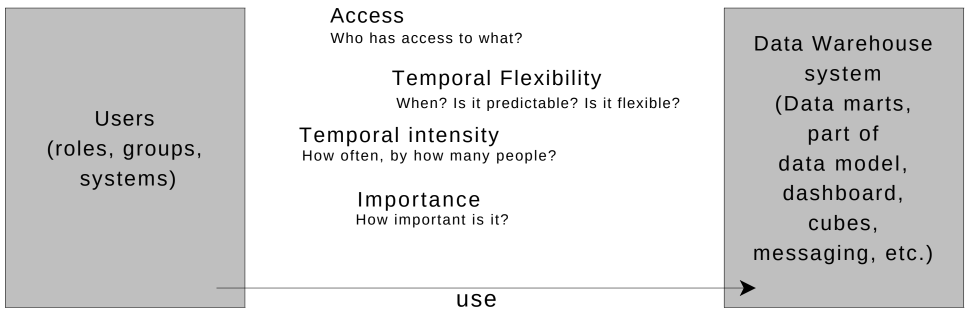


Figure 6.2: Perspectives to consider when describing the usage of a data warehouse

such as degenerate dimensions or nonstrict and complete dimensional hierarchies, and also provides three levels of detail.

6.3 Perspectives and Application Scenarios

In order to provide models of data warehouse usage that are useful to different application scenarios, a definition of the notion of usage is necessary. The goal is to achieve a broad view of usage, while maintaining concise models.

Usage occurs between different kinds of *users* (i.e., roles of users, groups of users, external users) which use different *parts of the data warehouse system* (data marts, facts, overview dashboards) in different ways (only passively, very often, more resticted), as illustrated in Figure 6.2. For greater clarity, the general notion of data warehouse usage is divided into four perspectives:

1. **Access control:** Who is allowed to use what?
2. **Temporal intensity:** How often do they use it?
3. **Temporal flexibility:** Do they have to use it at a certain time, or can it wait? Is it predictable when they need to use it?
4. **Importance:** How important is this usage?

A number of application scenarios of data warehouse usage models are identified. They vary with the target user group and the perspectives to be modeled, and offer modeling solutions for typical every-day requirements in data warehouse management, maintenance and (re-)design.

To gain an overview of the current system, *data warehouse engineers* and *architects* as well as *managers* can employ usage models containing details from the *access control perspective* and an approximate number of instances for the user roles or groups. This answers general questions such as "who is using this?", "how many people would complain if we remove this?", etc.

A more detailed model using the *access types* from the access perspective can serve as input for specifying *access restriction* policies, and/or for setting *predefined views* and queries in data access tools.

Temporal intensity and flexibility considerations can be used (a) during the *planing phase* of a data warehouse design project, user requirements have to be matched to the available resources. Data warehouse usage models offer a way to capture a general overview of both aspects. Designers can then procede from the usage models to more detailed models later on.

Additionally, if (b) changes become necessary to an existing data warehouse, usage models can help to *identify critical patterns*. For instance, if due to mergers or reorganizations the number of users in a certain area rapidly increases or decreases, the intensity and flexibility perspectives can provide an overview of the implications.

As in any real-life setting, often not all that is desirable can be achieved. With the help of usage models with elements from the *importance perspective*, managers can decide how to resolve *resource conflicts*.

6.4 The UML Profile for Modeling Data Warehouse Usage

This section introduces the UML Profile for Modeling Data Warehouse Usage. The extension mechanism of UML is used, and elements from a well-known UML Profile of the Data Warehousing domain are imported in order to achieve a conceptually sound model, with (a) tool support and (b) well-known notation elements as additional advantages gained by choosing to extend UML.

Figure 6.3 gives an overview of the UML Profile and its stereotypes and supporting enumerations, and also shows which classes are used as base classes of the stereotypes. The profile imports the profile of Luján-Mora et al. [LMTS06], and also additionally some packages of the UML metamodel (for the convenience of not having to use fully qualified names). Table 6.1 describes the characteristics of the stereotypes not shown in Figure 6.3.

Usage is defined as an InformationFlow, which is a type of directed relationship that specifies that information items circulate from sources to targets[1]. Information flows are defined in UML as a very general concept to be used in "*preliminary models, before having taken detailed modeling decisions on types or structures. One other purpose of information items and information flows is to abstract complex models by a less precise but more general representation of the information exchanged between entities of a system*" [OMG05c]. Even though [RJB04] state that these concepts "*are so vague as to question their usefulness*", this characteristic has its merits. This

[1]Sources and targets of an information flow may be: Actor, Node, UseCase, Artifact, Class, Component, Port, Property, Interface, Package, and InstanceSpecification [OMG05c].

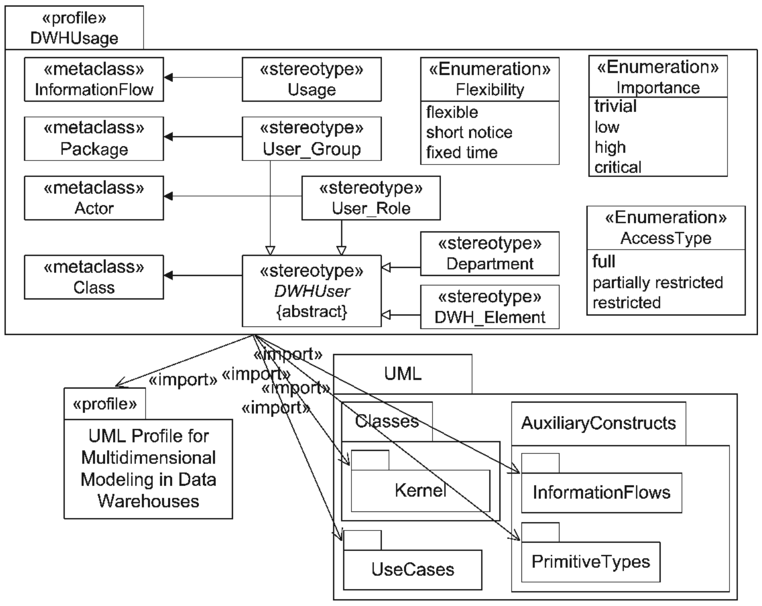
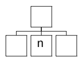


Figure 6.3: UML Profile for Data Warehouse Usage: Package contents and imports

makes information flows very suitable for this purpose, which is to provide models that capture an overview of the general structure of data warehouse usage.

Table 6.1: The UML Profile for Modeling Data Warehouse Usage

Name	***DWH User***	
Description	An entity using the data warehouse. Abstract.	
Tag Definition	**numberOfInstances** Type: Integer, Multiplicity: 1 Description: The number of instances of this role. Visualized in the icon e.g., as a number in the "head" of an actor symbol. **skillLevel** Type: SkillLevel, Multiplicity: 1 Description: The skill level of the user, i.e., whether able to write queries	
Name	**User Role**	
Description	A role that users/actors take when they access a data warehouse. One physical person (or external software system) may have several roles, and there may be several instances of one role.	n
Name	**User Group**	
Description	Group of similar roles (orthogonal to Department)	n

Name Description	**Department** Organizational department (orthogonal to User Group)	
Name Description	**DWH Element** An Element of the data warehouse system that users can access, and that can access other elements, e.g., a dashboard, a portal, etc.	
Name Description Tag Definition	**Usage** A *Usage* indicates an "information channel" [OMG05c] between the data warehouse and its users. See text below for details. **accessType**, Type: AccessType, Multiplicity: 1 Description: Indicates whether the access is (partially) restricted **temporalFlexibility**, Type: Flexibility, Multiplicity: n Default value: full Description: Indicates whether the usage is flexible in terms of time. **temporalIntensity**, Type: String, Multiplicity: 1 Description: A textual description of the intensity of usage, e.g., number of instances per time interval, as scalar or as interval, probability range, etc. **importance**, Type: Importance, Multiplicity: 1 Description: Indicates the level of importance attached to this usage.	«Usage» $\longrightarrow$
Name Stereotype Values	**SkillLevel** Enumeration {basic, intermediate, expert}	
Name Stereotype Values	**AccessType** Enumeration {full, partially restricted, restricted}	
Name Stereotype Values	**Flexibility** Enumeration {flexible, short notice, fixed time}	
Name Stereotype Values	**Importance** Enumeration {trivial, low, high, critical}	
Additionally, all elements imported from the UML Profile for MD modeling in Data Warehouses [LMTS06] (see Figure 6.3).		

The direction of the Usage arrow indicates whether the users actively initiate the access to the data warehouse or whether they wait to receive messages from the system, i.e., *push* or *pull* mode. A user analyzing OLAP data would be pull, whereas an e-mail alert is push.

6.5 Examples

This section illustrates the use of the UML Profile for Modeling Data Warehouse Usage with a number of examples. The examples each focus on a subset of the features of the profile and together provide an overview over the perspectives described in Section 6.3.

6.5.1 Users Accessing Data Warehouse Data

Figure 6.4 focuses on the questions "who needs which data?" and "who should be allowed to see what?". Diagrams of this type can be used in discussions with (future) users and in a later stage of the data warehouse design process may serve as rough input for specifying access restriction controls. Diagrams like this make it possible to identify preliminary groups of users, based on their data needs.

For each hospital admission it is recorded who was admitted (Patient dimension), what was the primary Diagnosis, which bed the patient was given (Placement) and which Insurance will cover the expenses. Health care professionals (nurses, doctors, therapists) need to access data on the patient, the diagnosis and the placement (the latter resticted to the ward they are working at), whereas the administration is interested in overall Figures of how many patients were admitted where, but should not access patient details or diagnoses. For the billing clerks of the accounting department, all data for charging the hospital bills to the insurance companies has to be accessible. Finally, if the data is made available to medical researchers (e.g., research on the seasonal occurrence and duration of certain medical conditions), only aggregated patient data and diagnoses are relevant.

6.5.2 Temporal Aspects of Data Warehouse Usage

The example shown in Figure 6.5.2 illustrates the concepts of Intensity and Flexibility as described in Section 6.3.

Consider a Sales fact, the typical example of Data Warehousing (see Section 6.2), which contains data on items sold, to be sorted, aggregated and analyzed by time, product group, store, etc. This example takes the Sales fact as a whole (a "StarPackage") and focus on the different users from various parts of the enterprise who all want to access this data.

Branch managers want to analyze the sales of their branch regularly once a week, and more or less predictably at the same time. The marketing department on the other hand will want access to sales data occasionally, but not necessarily at a given time or urgently. Product managers need to access the data with a varying intensity: If their new product is launched for example, they will watch the sales closely, but not at other times.

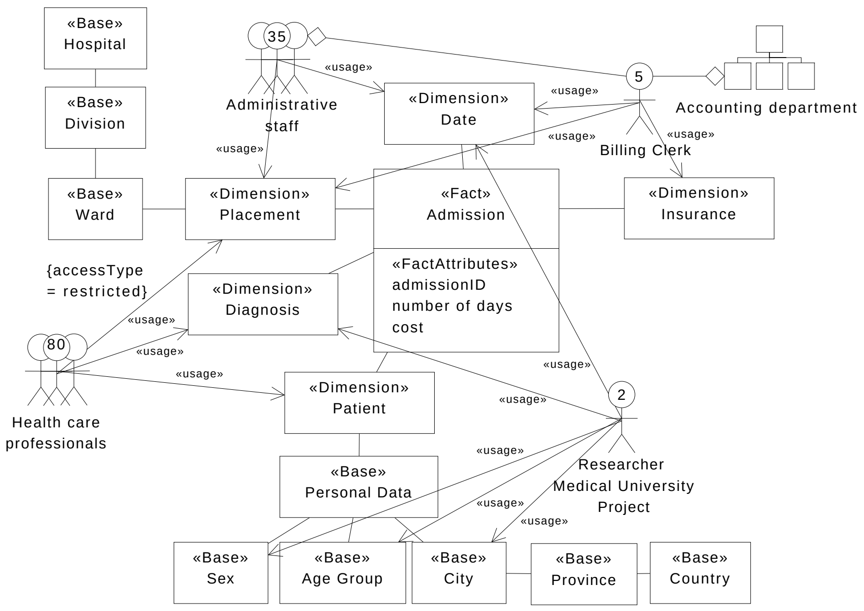


Figure 6.4: Hospital admissions: Different roles and groups

A new component that provides sales history data to sales agents is added to the system. Via this component, 150 individuals will want to access the sales data. The diagram provides an overview of the situation and supports discussions about this design decision.

6.5.3 Importance in Data Warehouse Usage

Assessing the relative importance of features is crucial for design decisions. For this purpose, it is restricted to four levels, (critical, high, low, and trivial) and subsume economical as well as "political" importance (see [Dem97] for examples of political issues in Data Warehousing) under one item, as shown in Figure 6.5.3. In this example the use of the attribute

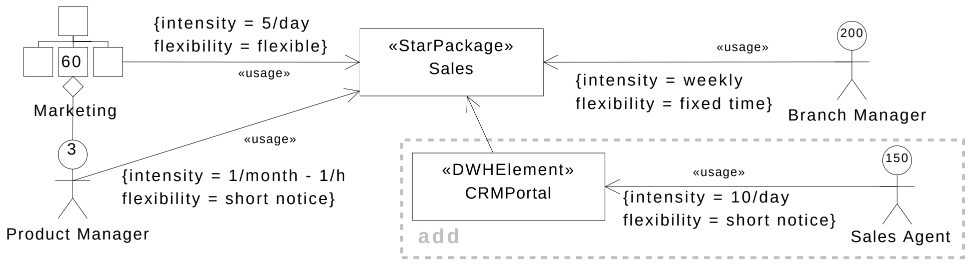

{intensity = 5/day flexibility = flexible}
«usage»
60
Marketing
«StarPackage» Sales
200
{intensity = weekly flexibility = fixed time}
Branch Manager
3
{intensity = 1/month - 1/h flexibility = short notice}
Product Manager
«DWHElement» CRMPortal
150
{intensity = 10/day flexibility = short notice}
Sales Agent
add

Figure 6.5: Temporal aspects of data warehouse usage: Intensity and Flexibility

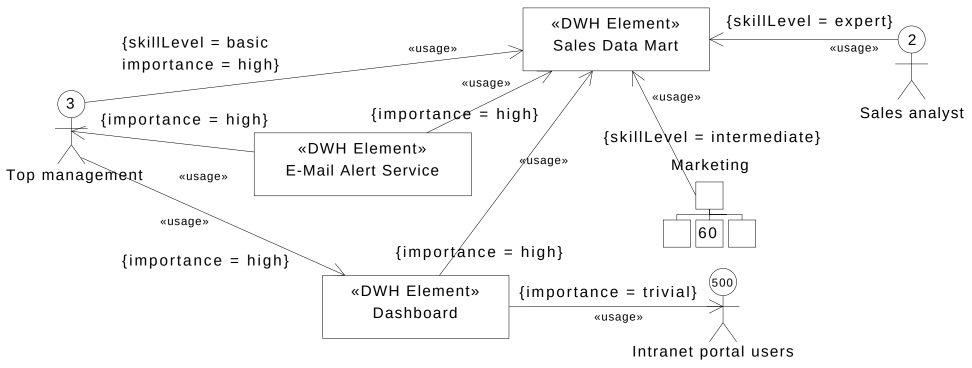


Figure 6.6: Importance, passive usage (push instead of pull), and user skills

skillLevel and passive data warehouse usage (via an e-mail alert service) are also shown.

Sales and Marketing people need to access sales data for their everyday work, which is of high importance. Top managers occasionally browsing sales data should also be treated as important, which is an example of "political" and not so much economical importance. Aggregated sales data is also fed into the Intranet portal, but this considered a "nice to have" feature of trivial importance.

6.6 Outlook: Towards a Comprehensive Requirements Analysis Approach for Data Warehousing

This chapter treats the need for a model of how people use data warehouses. Analyzing the intended and the actual usage of a system is closely related to requirements analysis. This section gives an outlook on how this viewpoint can impact the way requirements for data warehouse systems are managed.

Current approaches to data warehouse design focus on creating the correct data models. Research is very strong on constructing the optimal, usually multidimensional data model. This process has two main inputs:

1. The existing data sources.
2. The information requirements of the users.

Consolidating these two can be done in many different ways (see Section 2.1.4). Basically, it is a question of converting the data model of the sources into a data model that fits the queries of the users, as illustrated in Figure 6.7

Nevertheless, the final product of the design process is not only the data model of the data warehouse, but the whole data warehouse. This includes the physical database layout,

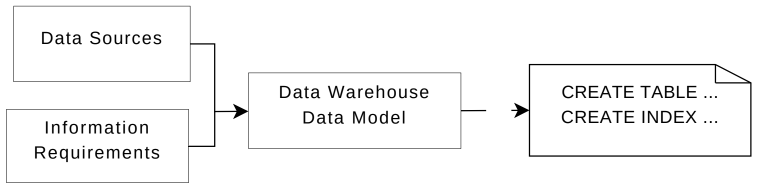


Figure 6.7: Consolidating data sources and information requirements

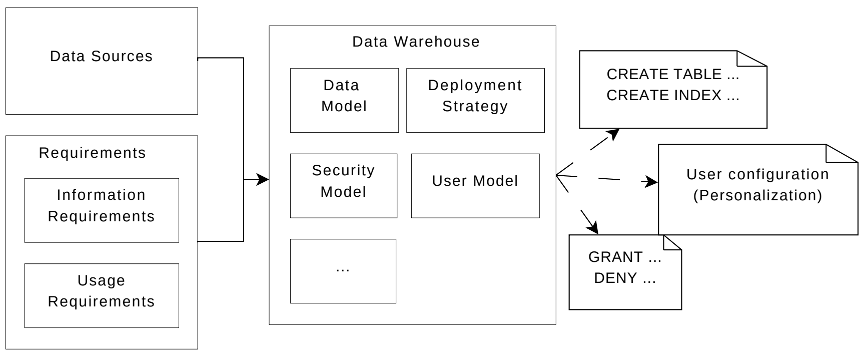


Figure 6.8: More types of usage requirements

indexing and other performance tuning techniques, user configuration, security measures, ETL interfaces, user interface configuration, etc. Additionally, quality of service (QoS) constraints have to be met, e.g., data quality, data freshness, query performance, accessibility, etc.

All these issues are inter-related and should not be considered separately, where possible. There is general agreement that is is beneficial to include security, quality or similar issues already in the first phases of requirements analysis and design, because adding them to a preliminary "naive" model later causes problems.

To summarize, there are actually three inputs to data warehouse design:

1. The existing data sources.
2. The information requirements of the users.
3. The "other" requirements of the users.

Figure 6.8 illustrates the notion described above. Based on these concepts, a future requirements analysis approach for Data Warehouses could be based on a framework as the one shown in Figure 6.9. In this framework, the information requirements are also a subitem of "usage". It can be extended to include any type of usage or QoS requirements.

There is a strong trend in Data Warehousing today from model-based semi-automated design approaches to truly model-driven engineering. With the ideas presented here, it

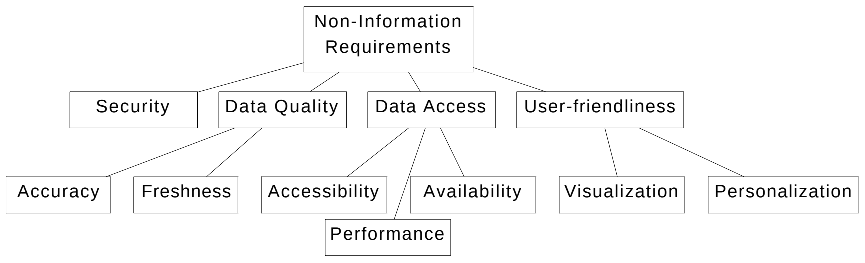


Figure 6.9: Open framework for Data Warehouse requirements analysis

could become possible to generate large parts of the data warehouse systems from models. Many items in the framework can be realized as configuration files (e.g., user settings) which have a defined syntax and can be automatically generated. Generating many parts of the data warehouse system together allows further control of details (i.e., the database tables' and columns' names are known, user groups are known, QoS thresholds are known), which can be considered together to produce better results.

In the long run, it can be envisaged that full round-trip engineering becomes possible, i.e., by analyzing actual usage (from logfiles of servers, databases and tools), deriving models, creating improved models, and finally generating a new, improved but compatible version of the system.

6.7 Related Work

The approach to modeling data warehouse usage on the conceptual level as presented in this chapter touches many different areas, with access control, temporal intensity, temporal flexibility and importance, active or passive usage, details of the users such as their skill level, number of instance or affiliation. To the best of my knowledge, there is no comparable work with the same focus. Nevertheless, regarding individual aspects, the relationship to previous work can be discussed.

Modeling the users who access different parts of data warehouse systems is also an issue for security and access control in Data Warehousing. In [FMTVP07], the authors present a UML Profile for secure data warehouses that includes user profiles and user roles contained in hierarchies. The users are granted privileges to access parts of a multidimensional data model. [PP01] define a different kind of authorization model integrated with MD modeling, also based on user roles. [KKST97] or [WJW04] are other example of approaches to modeling user access to parts of multidimensional models and/or OLAP operations.

As these approaches have a different aim in modeling users (i.e., grouping users with similar privileges for access control and security measures), they do not include the organizational affiliation of users, their skill level, or the importance, intensity or temporal flexibility of their access to the data warehouse.

This approach can also be compared to Requirements Engineering. [BLS01] describe an approach to model data warehouse requirements with UML use cases. [MPT07] and [GRG05] describe approaches for goal-oriented data warehouse requirements. [MTL07] reconcile the available data sources with the requirements, and [MT06] present an MDA approach for building conceptual and logical data models from requirements models.

The UML profile for modeling data warehouse usage is not intended as a means to create data models, but can be used to support the requirements analysis phase of a data warehouse project.

6.8 Concluding Remarks

Today's data warehouse systems provide many different services to different kinds of users. In order to have a big picture of the current situation and to visualize future scenarios, people involved in designing and managing today's data warehouse systems need an overview of all these different ways the data warehouse is being used.

This chapter has introduced the UML Profile for Modeling Data Warehouse Usage for modeling the different kinds of data warehouse usage on a conceptual level. It distinguishes four perspective of usage (access control, temporal intensity, temporal flexibility and importance) as well as active or passive usage, and allows to model details of the users such as their skill level, number of instances, functional grouping or organizational affiliation. In this model, "usage" is based on UML information flows, which are intended for a more general representation of information exchanges and import the elements of an existing profile for modeling the special multidimensional data models of data warehouses.

Chapter 7

Bermuda: A Prototype for Model-Based Business Metadata

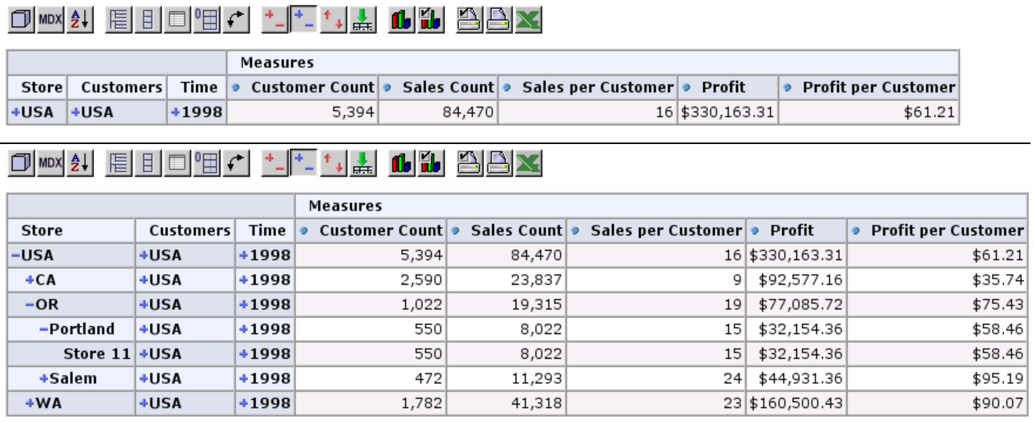
This chapter describes the prototype that was developed within the scope of this thesis in order to illustrate the applicability of model-based business metadata. Chapter 3 describes the theory behind this. Section 7.1 introduces the prototype with a typical use case, first without and then with business metadata, to illustrate its features. Section 7.2 describes the design and implementation of the prototype, and Section 7.3 provides a simple installation guide.

Contents

7.1 The Example

The example used here is based on the well-known *Foodmart* OLAP example. The Foodmart is a supermarket chain, and the example package includes data on sales and promotions, warehousing[1], and HR. The Foodmart's data, OLAP configuration and queries are for example included with the Analysis Services component of MS SQL Server, and are used as

[1]Physical warehouses, not data warehouses.



			Measures				
Store	Customers	Time	Customer Count	Sales Count	Sales per Customer	Profit	Profit per Customer
+USA	+USA	+1998	5,394	84,470	16	$330,163.31	$61.21

			Measures				
Store	Customers	Time	Customer Count	Sales Count	Sales per Customer	Profit	Profit per Customer
−USA	+USA	+1998	5,394	84,470	16	$330,163.31	$61.21
+CA	+USA	+1998	2,590	23,837	9	$92,577.16	$35.74
−OR	+USA	+1998	1,022	19,315	19	$77,085.72	$75.43
−Portland	+USA	+1998	550	8,022	15	$32,154.36	$58.46
Store 11	+USA	+1998	550	8,022	15	$32,154.36	$58.46
+Salem	+USA	+1998	472	11,293	24	$44,931.36	$95.19
+WA	+USA	+1998	1,782	41,318	23	$160,500.43	$90.07

Figure 7.1: Analyzing Sales data: Top level, and drill down on Store

sample data by open source BI projects such Mondrian or Pentaho, and also in teaching environments.

7.1.1 Without Business Metadata

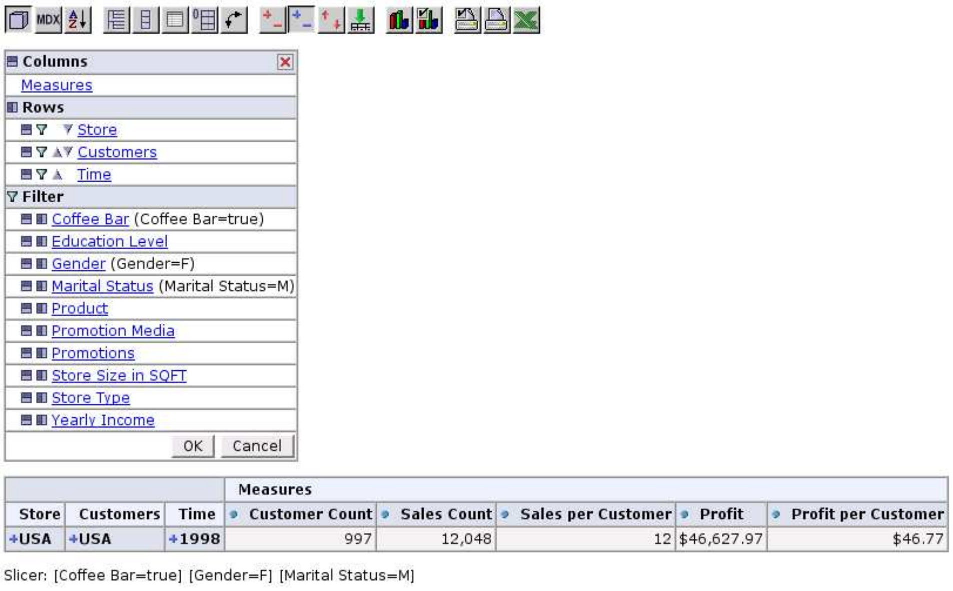
This section describes the features offered by the example OLAP application without the business metadata plugin.

The application is accessed with a Web browser, and the initial screen (at the top of Figure 7.1) shows highly aggregated data for five measures, some of them calculated from others. The data can be explored along three dimensions: *Store*, *Customer*, and *Time* (these three dimensions were pre-selected in this page and could be changed via the toolbar shown above the table; there is a forth dimension, *Product*, not used here). Figure 7.1 shows how the data can be explored with subsequent drill-down operations on the Store dimension.

Further OLAP operations are possible via the toolbar: Filters, slicers, swapping the axes, etc. Figure 7.2 shows how the *OLAP Navigator* is used to change the query to select only sales in stores with coffee bars to female customers who are married. Users can also edit the MDX queries directly, as shown in Figure 7.3.

7.1.2 With Business Metadata

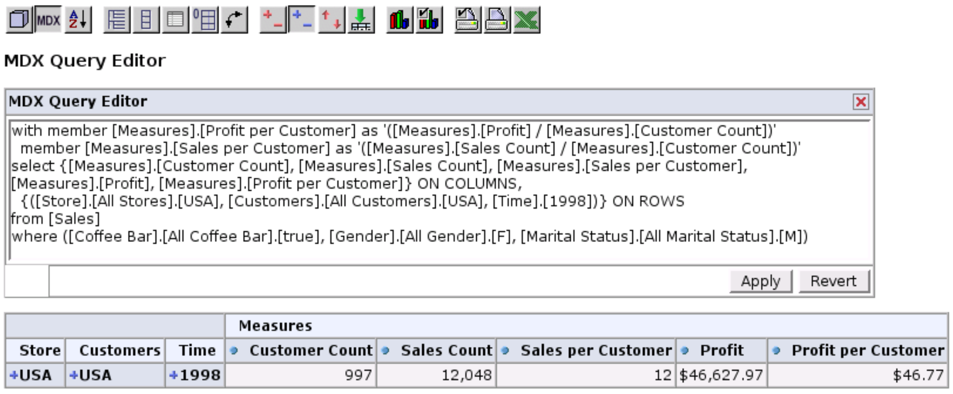
This section describes the same application as above, but with the addition of the business metadata plugin. The first part describes how the user's experience is enriched with business metadata during data analysis. The second part describes how the models behind the business metadata are defined and linked to the application.



			Measures				
Store	Customers	Time	Customer Count	Sales Count	Sales per Customer	Profit	Profit per Customer
USA	USA	1998	997	12,048	12	$46,627.97	$46.77

Slicer: [Coffee Bar=true] [Gender=F] [Marital Status=M]

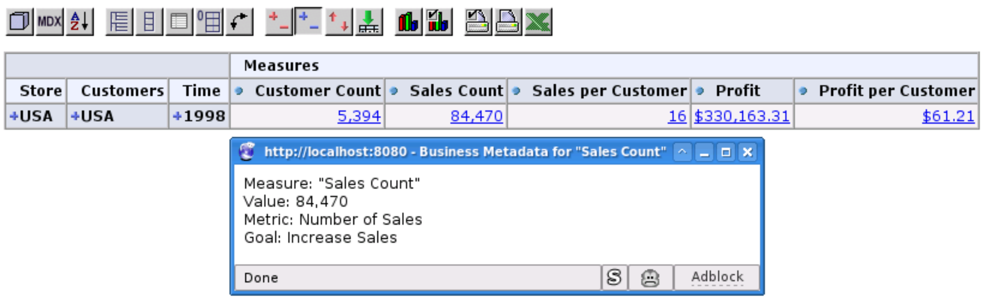
Figure 7.2: The pivot table can be customized via the Web interface



```
with member [Measures].[Profit per Customer] as '([Measures].[Profit] / [Measures].[Customer Count])'
  member [Measures].[Sales per Customer] as '([Measures].[Sales Count] / [Measures].[Customer Count])'
select {[Measures].[Customer Count], [Measures].[Sales Count], [Measures].[Sales per Customer],
[Measures].[Profit], [Measures].[Profit per Customer]} ON COLUMNS,
  {([Store].[All Stores].[USA], [Customers].[All Customers].[USA], [Time].[1998])} ON ROWS
from [Sales]
where ([Coffee Bar].[All Coffee Bar].[true], [Gender].[All Gender].[F], [Marital Status].[All Marital Status].[M])
```

			Measures				
Store	Customers	Time	Customer Count	Sales Count	Sales per Customer	Profit	Profit per Customer
USA	USA	1998	997	12,048	12	$46,627.97	$46.77

Figure 7.3: The Web interface offers a simple MDX query editor



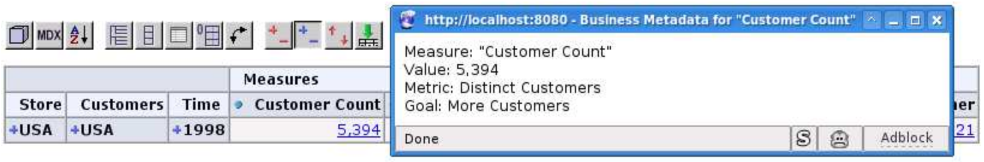
			Measures				
Store	Customers	Time	Customer Count	Sales Count	Sales per Customer	Profit	Profit per Customer
+USA	+USA	+1998	5,394	84,470	16	$330,163.31	$61.21

http://localhost:8080 - Business Metadata for "Sales Count"

Measure: "Sales Count"
Value: 84,470
Metric: Number of Sales
Goal: Increase Sales

Done

Figure 7.4: A Business Metadata Popup for Sales Count



			Measures	
Store	Customers	Time	Customer Count	ner
+USA	+USA	+1998	5,394	21

http://localhost:8080 - Business Metadata for "Customer Count"

Measure: "Customer Count"
Value: 5,394
Metric: Distinct Customers
Goal: More Customers

Done

Figure 7.5: A Business Metadata Popup for Customer Count

7.1.2.1 Accessing the Data

Figure 7.4 shows the same entry page as Figure 7.1, but the measures are now clickable links. If a measure is clicked (*Sales Count* in this case), a popup appears and displays the business metadata connected to this value.

The business metadata shown in Figure 7.4 tells the user that the name of the metric that corresponds to *Sales Count* is *Number of Sales*, and that the goal that is measured with this metric is *Increase Sales*.

The advantage of having access to business metadata become apparent here: Business users are accustomed to their own vocabularies and concepts. The users might or might not know that the "sales count" is the same as the "number of sales" metric known to them. Why these terms were chosen is usually not known to all users, and they might be different only for historical reasons. Because this knowledge about which measure corresponds to which metric is mainly implicit, it is more likely to be lost or forgotten. The corresponding goal in this case is quite obvious, but can be less obvious in other situations.

Figure 7.5 shows the business metadata for the measure *Customer Count*. In this case, the metric is called *Distinct Customers*. It might not be apparent from the names that these to values are the same. This additional information can be valueable to the users and avoid that their analysis misses important points and becomes more errorprone.

The models used in this example are very simple. The weaving model contains only one link, and from the metric it is possible to access only one additional element, the goal. As

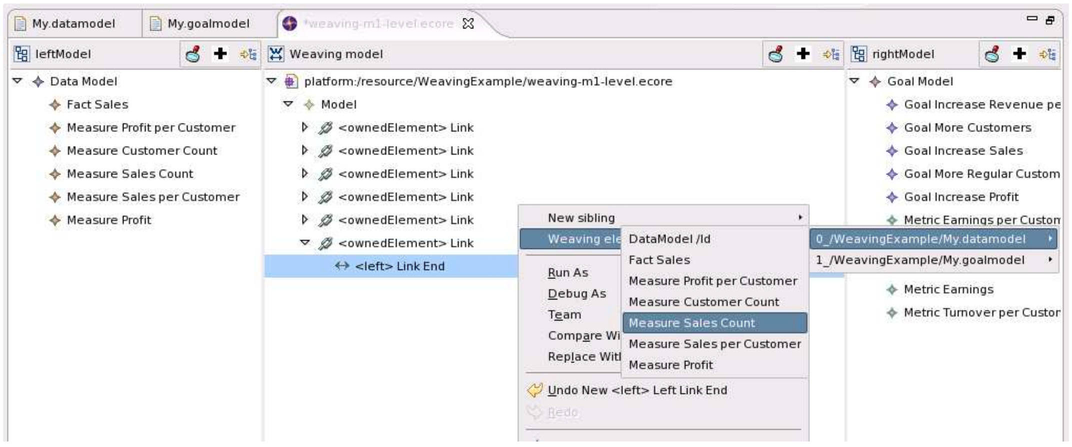


Figure 7.6: Creating a new weaving link with the weaving model editor

described in detail in Section 3, the weaving model can contain many, and more complex links. The business process that produces the metric and is measured by it, or the organzational unit that is responsible, could be added. A product model can provide metadata on the product hierarchy, and in connection with the time values, it is possible to display the target values of the metrics.

7.1.2.2 Defining the Business Metadata

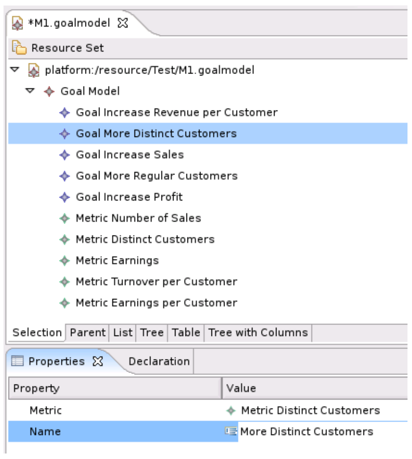
How is the business metadata linked to the Data Warehouse? Figure 7.6 shows the AMW model weaver tool. The data model is shown in the left panel. It contains the measures of the example. The panel on the right contains the goal model with some metrics that can be linked to the measures. The weaving links are created in the central panel, in this case via the context menu.

How can the business metadata be updated? The models linked by the weaving model are independent. Therefore, all elements of the goal model can be changed, as long as the link ends are not entirely removed. Figure 7.7 shows the goalmodel in the model editor. The goal *More Customers* has just been renamed to *More Distinct Customers*.

The change is visible in the weaving model editor (Figure 7.8) and also in the business metadata displayed to the user (Figure 7.9).

7.2 The Business Metadata Plugin

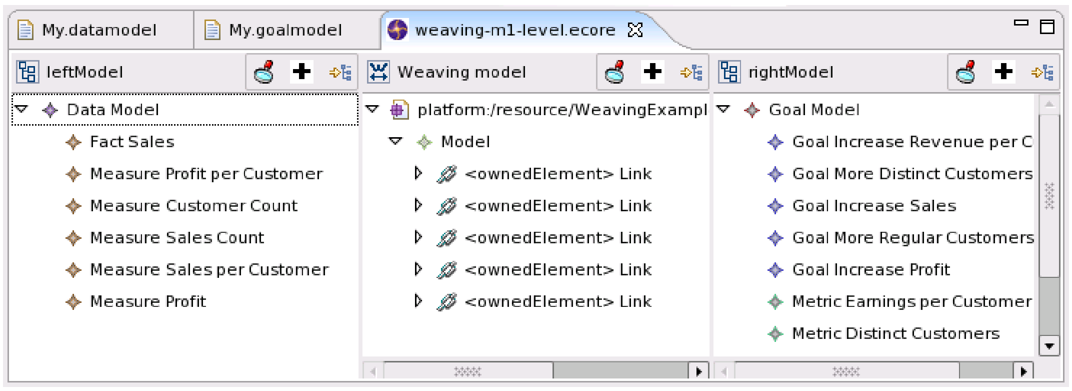
This section treats the design and implementation of the business metadata plugin. The example data and configuration as well as the tools and technologies used are described.



Resource Set
- platform:/resource/Test/M1.goalmodel
  - Goal Model
    - Goal Increase Revenue per Customer
    - Goal More Distinct Customers
    - Goal Increase Sales
    - Goal More Regular Customers
    - Goal Increase Profit
    - Metric Number of Sales
    - Metric Distinct Customers
    - Metric Earnings
    - Metric Turnover per Customer
    - Metric Earnings per Customer

Property	Value
Metric	Metric Distinct Customers
Name	More Distinct Customers

Figure 7.7: Renaming the goal in the EMF model editor

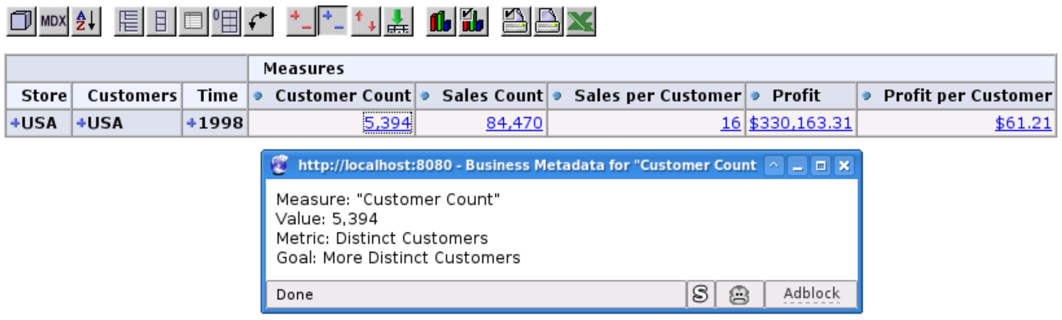


- Data Model
  - Fact Sales
  - Measure Profit per Customer
  - Measure Customer Count
  - Measure Sales Count
  - Measure Sales per Customer
  - Measure Profit

- platform:/resource/WeavingExampl
  - Model
    - <ownedElement> Link
    - <ownedElement> Link
    - <ownedElement> Link
    - <ownedElement> Link
    - <ownedElement> Link

- Goal Model
  - Goal Increase Revenue per C
  - Goal More Distinct Customers
  - Goal Increase Sales
  - Goal More Regular Customers
  - Goal Increase Profit
  - Metric Earnings per Customer
  - Metric Distinct Customers

Figure 7.8: The new name of the goal does not effect the weaving link



			Measures				
Store	Customers	Time	Customer Count	Sales Count	Sales per Customer	Profit	Profit per Customer
USA	USA	1998	5,394	84,470	16	$330,163.31	$61.21

http://localhost:8080 - Business Metadata for "Customer Count

Measure: "Customer Count"
Value: 5,394
Metric: Distinct Customers
Goal: More Distinct Customers

Done

Figure 7.9: The change in the goal model is visible in the business metadata popup

7.2.1 The Name

The name "Bermuda" was chosen as a more easily pronounceable subsitute for the original acronym "BMD", which simply stands for Business MetaData. Additionally, BMD is also the ISO 4217 code for the Bermudian Dollar, which might be seen as a hint that Data Warehousing always has to do with money. Bermuda allegedly has the highest GDP per capita in the world.

7.2.2 Overview and Architecture

The plugin is plugged into JPivot as a `TableExtension`. Table Extensions can add custom `CellBuilderDecorators` to the table's `CellBuilder`. The Cell Builder Decorators are called during the rendering of each cell of the table.

When a cell is rendered, the Cell Builder Decorator of the business metadata plugin collects the values necessary for identifying the business metadata of the current cell[2], and adds a popup link attribute with these values to the cell element. The table containing the cells is then rendered including all the links and converted to HTML by JPivot (i.e., by the WCF library). See Section 7.2.3.1 for a description of the JPivot API relevant to constructing the popup link.

When the user clicks on the business metadata link, a JSP page is opened in a popup window. With the help of the EMF and AMW libraries, the Java code behind the JSP retrieves the relevant weaving link from the weaving model by selecting the data model element that corresponds to the cell that was clicked. Then the business metadata values are retrieved from the other model (in this example the goal model) via the weaving model, and displayed in the popup page.

Figure 7.10 illustrates the structure of the business metadata plugin. The folders shown include everything necessary for deploying and running the example as shown in this chapter.

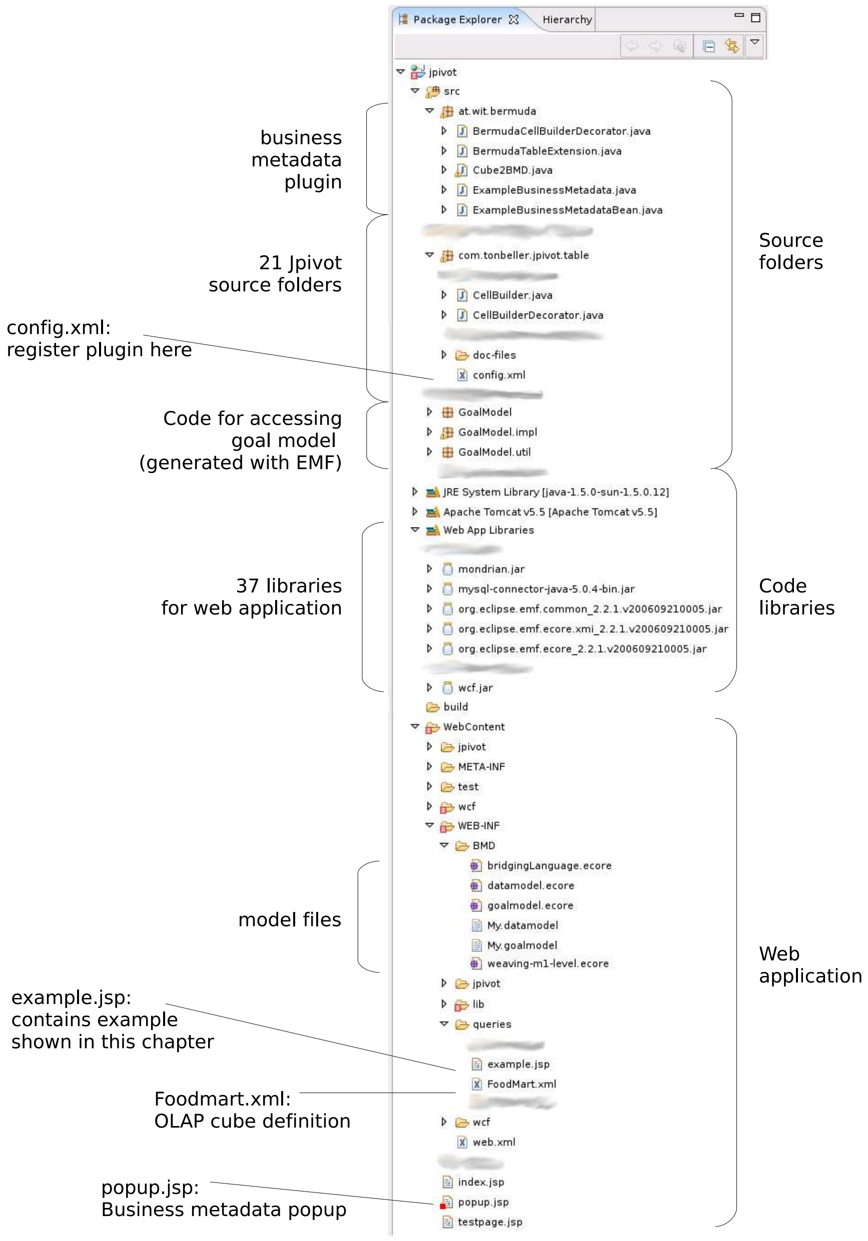
7.2.3 Data Warehouse Stack

Figure 7.11 illustrates the architecture of the data warehouse stack created with JPivot and Mondrian. The interaction between the components is visualized in Figure 7.12.

7.2.3.1 JPivot

JPivot is a JSP tag library for OLAP. It renders the result of a multidimensional query to HTML, and supports all typical OLAP navigations such as roll up and drill down, slice and dice. JPivot tags such as `<jp:table id="table01" query="#{query01}"/>` and

[2]e.g., the name of the measure and the current aggregation level; which values are necessary depends on the business metadata model


business metadata plugin
21 Jpivot source folders
config.xml: register plugin here
Code for accessing goal model (generated with EMF)
37 libraries for web application
model files
example.jsp: contains example shown in this chapter
Foodmart.xml: OLAP cube definition
popup.jsp: Business metadata popup
Source folders
Code libraries
Web application
Package Explorer
Hierarchy

```
jpivot
  src
    at.wit.bermuda
      BermudaCellBuilderDecorator.java
      BermudaTableExtension.java
      Cube2BMD.java
      ExampleBusinessMetadata.java
      ExampleBusinessMetadataBean.java
    com.tonbeller.jpivot.table
      CellBuilder.java
      CellBuilderDecorator.java
      doc-files
      config.xml
    GoalModel
    GoalModel.impl
    GoalModel.util
  JRE System Library [java-1.5.0-sun-1.5.0.12]
  Apache Tomcat v5.5 [Apache Tomcat v5.5]
  Web App Libraries
    mondrian.jar
    mysql-connector-java-5.0.4-bin.jar
    org.eclipse.emf.common_2.2.1.v200609210005.jar
    org.eclipse.emf.ecore.xmi_2.2.1.v200609210005.jar
    org.eclipse.emf.ecore_2.2.1.v200609210005.jar
    wcf.jar
  build
  WebContent
    jpivot
    META-INF
    test
    wcf
    WEB-INF
      BMD
        bridgingLanguage.ecore
        datamodel.ecore
        goalmodel.ecore
        My.datamodel
        My.goalmodel
        weaving-m1-level.ecore
      jpivot
      lib
      queries
        example.jsp
        FoodMart.xml
      wcf
      web.xml
    index.jsp
    popup.jsp
    testpage.jsp
```

Figure 7.10: The structure of the business metadata plugin and the example application

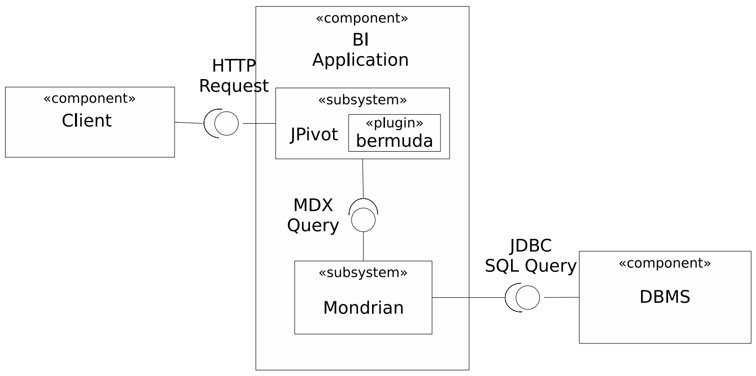


Figure 7.11: The architecture of the data warehouse stack created with JPivot and Mondrian

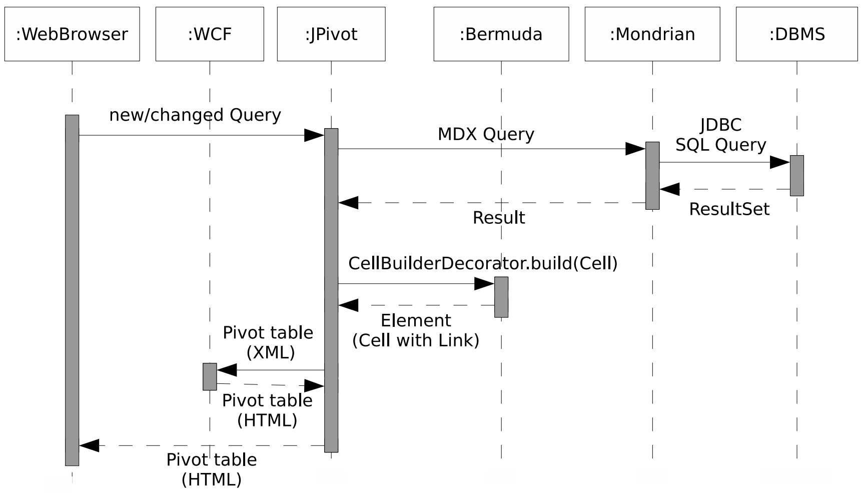


Figure 7.12: Communication between the components: The user has requested a change

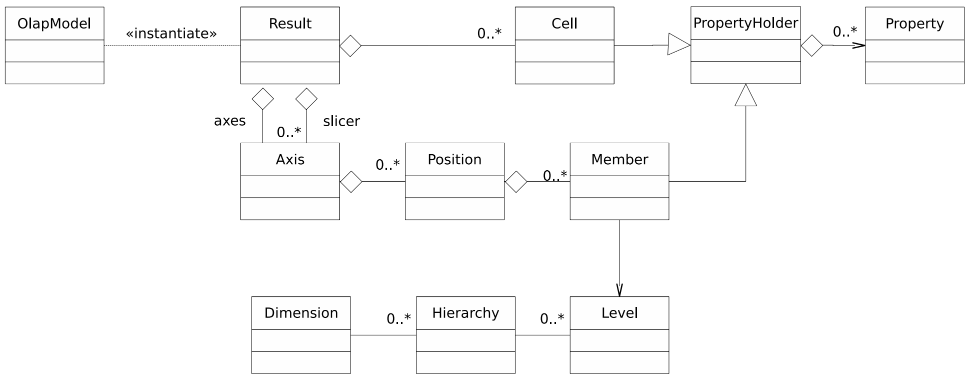


Figure 7.13: JPivot OLAP Result: The pivot table ([Ton07])

`<jp:mondrianQuery id="query01"> select ... </jp:mondrianQuery>` hide the complexity of rendering an OLAP cube from the developers of an OLAP web application. JPivot uses Mondrian as its OLAP server (but this relationship is not hard-coded) and therefore can be used with any data source that is accessible via JDBC or XML/A (see Section 7.2.3.3).

Figure 7.13 shows the JPivot package `com.tonbeller.jpivot.olap.model` that defines an implementation independent basic OLAP result (not the query, but the result that is displayed to the user). The business metadata plugin uses this part of the JPivot API to recognize the elements of the data model for which the business metadata is be fetched.

7.2.3.2 MDX

Listing 7.1: Basic MDX query used in the example (before any further OLAP operations)

```
WITH
MEMBER [Measures].[Profit per Customer] AS '([Measures].[Profit] /
    [Measures].[Customer Count])'
MEMBER [Measures].[Sales per Customer] AS '([Measures].[Sales Count] /
    [Measures].[Customer Count])'
SELECT
  {[Measures].[Customer Count], [Measures].[Sales Count], [Measures].[Sales per
      Customer], [Measures].[Profit], [Measures].[Profit per Customer]} ON COLUMNS,
  {([Store].[USA],[Customers].[USA],[Time].[1998])} ON ROWS
FROM Sales
```

The MDX ("Multidimensional Expressions") language is an OLAP query language originally introduced by Microsoft in the late 1990s which has become an industry standard. The syntax of MDX queries is vaguely similar to SQL, but an MDX query is applied to an OLAP cube[3], and the result that is returned is also a cube. For a detailed introduction to the

[3]i.e, any multidimensional data source

syntax and features of MDX see [Pea02].

Listing 7.1 shows the MDX query used in the example: Two calculated measures are defined in the preamble (WITH MEMBER), then five measures are selected to be displayed in the columns (ON COLUMNS), and three dimensions (Store, Customers, and Time) on the row axis (ON ROWS). The year 1998 and locations in the USA are preselected.

7.2.3.3 Mondrian

Mondrian is a Java OLAP server. For a given multi-dimensional query (in MDX or XML/A) it returns the resulting cube as a `mondrian.olap.Result`, which is to OLAP what the standard JDBC `ResultSet` is to relational database queries.

Mondrian can be used with any JDBC compatible data source, i.e., SQL databases and spreadsheets as well as flat files. Mondrian uses an XML file to define the multi-dimensional schema on top of the data source (which does not have to be multidimensional). Queries are written against the multi-dimensional schema and then translated at runtime by Mondrian into queries against the underlying data source. The example in Listing 7.2 maps the table *sales_fact_1998* to the cube *Sales* and its column *unit_sales* to the measure *Unit Sales*.

Listing 7.2: Mapping the data source to the multi-dimensional schema

```
<Cube name="Sales">
  <Table name="sales_fact_1998"/>
  <Measure name="Unit Sales" column="unit_sales"/>
</Cube>
```

Listing 7.3 shows the Cube used in the example (some details excluded). It contains a subset of the possibilities offered by the Foodmart database. The database already conforms to the multi-dimensional paradigm, therefore the mappings between the database tables and columns and the facts, dimensions and measures are rather straightforward.

See [Hyd07] for a detailed description of Mondrian's architecture and API, including caching and performance issues. In general, Mondrian aims to utilize as much of the underlying database's capabilities as possible.

Listing 7.3: Mondrian OLAP Cube Schema used in the example (selected elements)

```
<?xml version="1.0"?>
<Schema name="FoodMart">

<!-- Shared dimensions (to be used by several cubes) -->
  <Dimension name="Store">
    <Hierarchy hasAll="true" primaryKey="store_id">
      <Table name="store"/>
      <Level name="Store Country" column="store_country" uniqueMembers="true"/>
      <Level name="Store State" column="store_state" uniqueMembers="true"/>
      <Level name="Store City" column="store_city" uniqueMembers="false"/>
      <Level name="Store Name" column="store_name" uniqueMembers="true">
      </Level>
    </Hierarchy>
  </Dimension>

  <Dimension name="Time" type="TimeDimension">
    <Hierarchy hasAll="false" primaryKey="time_id">
      <Table name="time_by_day"/>
      <Level name="Year" column="the_year" type="Numeric" uniqueMembers="true" levelType="TimeYears"/>
      <Level name="Quarter" column="quarter" uniqueMembers="false" levelType="TimeQuarters"/>
      <Level name="Month" column="month_of_year" uniqueMembers="false" type="Numeric" levelType="TimeMonths"/>
    </Hierarchy>
  </Dimension>

  <Dimension name="Product">
    <Hierarchy hasAll="true" primaryKey="product_id" primaryKeyTable="product">
      <Join leftKey="product_class_id" rightKey="product_class_id">
        <Table name="product"/>
        <Table name="product_class"/>
      </Join>
      <Level name="Product Family" table="product_class" column="product_family" uniqueMembers="true"/>
      <Level name="Product Department" table="product_class" column="product_department" uniqueMembers="false"/>
      <Level name="Product Category" table="product_class" column="product_category" uniqueMembers="false"/>
      <Level name="Product Subcategory" table="product_class" column="product_subcategory" uniqueMembers="false"/>
      <Level name="Brand Name" table="product" column="brand_name" uniqueMembers="false"/>
      <Level name="Product Name" table="product" column="product_name"
          uniqueMembers="true"/>
    </Hierarchy>
  </Dimension>

<Cube name="Sales">

 <Table name="sales_fact_1998"/>

  <DimensionUsage name="Store" source="Store" foreignKey="store_id"/>
  <DimensionUsage name="Time" source="Time" foreignKey="time_id"/>
  <DimensionUsage name="Product" source="Product" foreignKey="product_id"/>
  <Dimension name="Customers" foreignKey="customer_id">
    <Hierarchy hasAll="true" allMemberName="All Customers" primaryKey="customer_id">
      <Table name="customer"/>
      <Level name="Country" column="country" uniqueMembers="true"/>
      <Level name="State Province" column="state_province" uniqueMembers="true"/>
      <Level name="City" column="city" uniqueMembers="false"/>
      <Level name="Name" uniqueMembers="true" />
    </Hierarchy>
  </Dimension>

  <Measure name="Unit Sales" column="unit_sales" aggregator="sum" formatString="Standard"/>
  <Measure name="Store Cost" column="store_cost" aggregator="sum" formatString="#,###.00"/>
  <Measure name="Store Sales" column="store_sales" aggregator="sum" formatString="#,###.00"/>
  <Measure name="Sales Count" column="product_id" aggregator="count" formatString="#,###"/>
  <Measure name="Customer Count" column="customer_id" aggregator="distinct count" formatString="#,###"/>
  <CalculatedMember name="Profit" dimension="Measures">
    <Formula>[Measures].[Store Sales] - [Measures].[Store Cost]</Formula>
    <CalculatedMemberProperty name="FORMAT_STRING" value="$#,##0.00"/>
  </CalculatedMember>

</Cube>
</Schema>
```

7.2.3.4 Database

The example uses the Foodmart database with MySQL. The schema of this database is shown in Figure 7.14, and Table 7.1 gives an idea of the sizes of the tables.

Table	Number of rows
sales_fact_1998	164558
customer	10281
store	25
time_by_day	730

Table 7.1: Sizes of the database tables used in the example

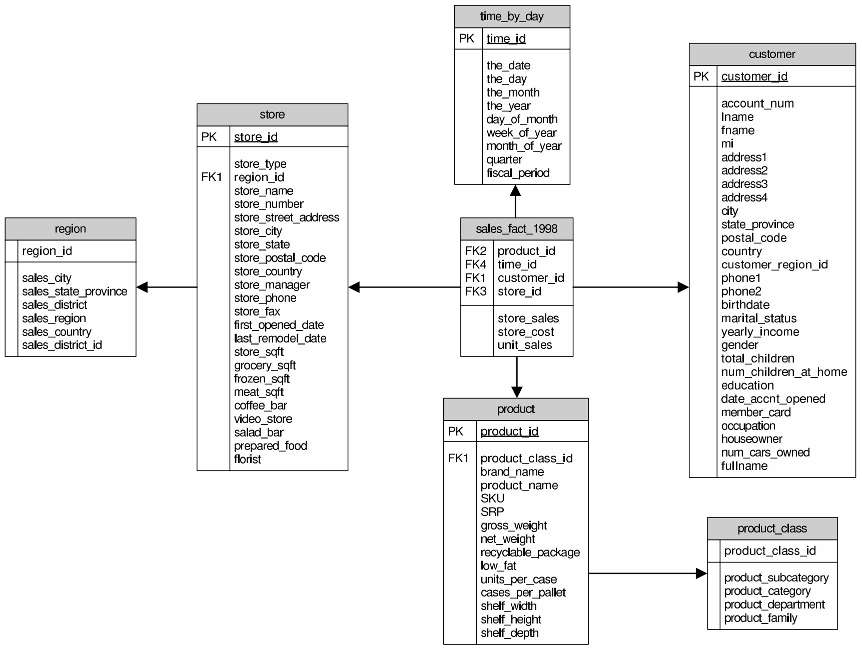


Figure 7.14: The data model of the example data (simplified from Foodmart example)

7.2.4 Modeling Stack

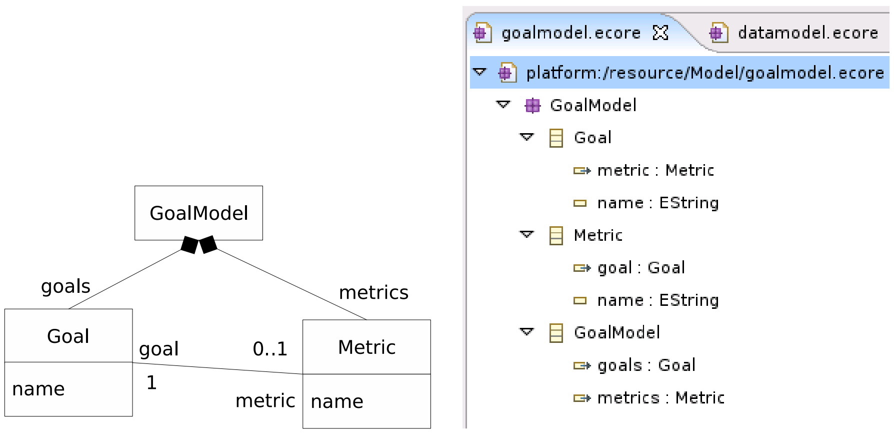
This section gives an overview over the models used in the example. They partly correspond to the examples presented in Section 3.3, but are simplified here for readability and reduced complexity. The aim of the examples is to illustrate how the information contained in (for example) a goal model can be linked to the data warehouse, and how it then can be accessed with the BI tools.

7.2.4.1 Metamodel Level

Figure 7.15 shows the simplified goal model metamodel used in the example. It contains goals and metrics, each with an attribute "name" and references from goal to metric and vice versa.

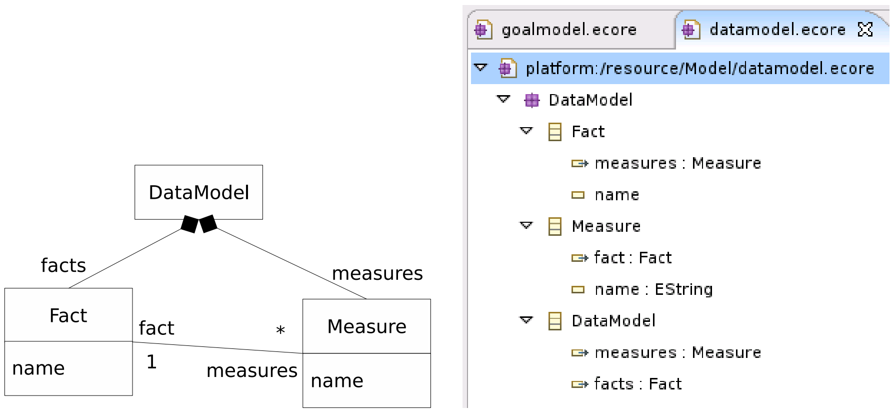
The simplified metamodel of the data model is shown in Figure 7.16, containing only facts and measures. A fact may have any number of measures, and each measure belongs to one fact.

The weaving model between these two metamodels is shown in Figure 7.17. It contains one link, between the class `Measure` of the data model and the class `Metric` of the goal model.



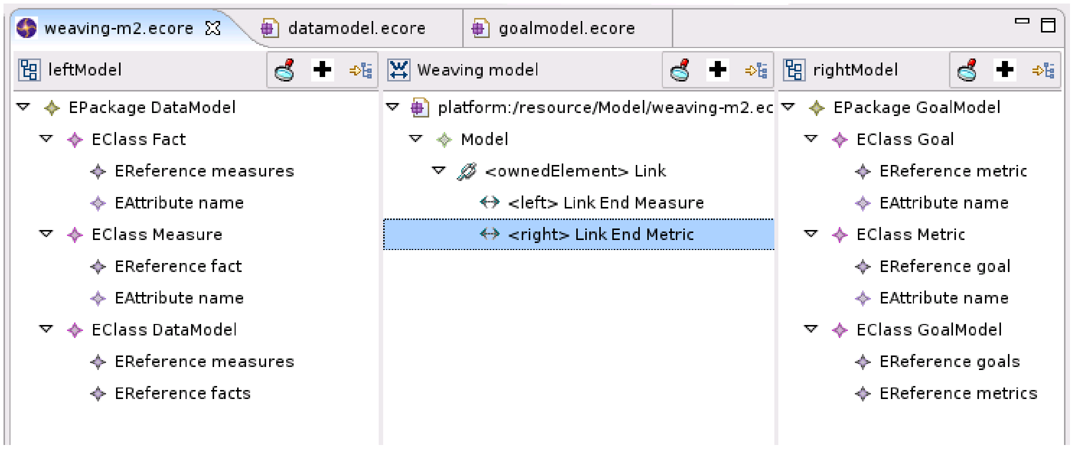
```
goalmodel.ecore    datamodel.ecore
▽ platform:/resource/Model/goalmodel.ecore
  ▽ GoalModel
    ▽ Goal
        metric : Metric
        name : EString
    ▽ Metric
        goal : Goal
        name : EString
    ▽ GoalModel
        goals : Goal
        metrics : Metric
```

Figure 7.15: The metamodel of the goal model (tree view of the EMF model editor)



```
goalmodel.ecore    datamodel.ecore
▽ platform:/resource/Model/datamodel.ecore
  ▽ DataModel
    ▽ Fact
        measures : Measure
        name
    ▽ Measure
        fact : Fact
        name : EString
    ▽ DataModel
        measures : Measure
        facts : Fact
```

Figure 7.16: The metamodel of the data model (tree view of the EMF model editor)



```
leftModel
▽ EPackage DataModel
    ▽ EClass Fact
        EReference measures
        EAttribute name
    ▽ EClass Measure
        EReference fact
        EAttribute name
    ▽ EClass DataModel
        EReference measures
        EReference facts
```

```
Weaving model
▽ platform:/resource/Model/weaving-m2.ec
    ▽ Model
        ▽ <ownedElement> Link
            <left> Link End Measure
            <right> Link End Metric
```

```
rightModel
▽ EPackage GoalModel
    ▽ EClass Goal
        EReference metric
        EAttribute name
    ▽ EClass Metric
        EReference goal
        EAttribute name
    ▽ EClass GoalModel
        EReference goals
        EReference metrics
```

Figure 7.17: A link is defined with the AMW weaver editor

7.2.4.2 Model Level

Figure 7.18 shows the parts of the data model that are used in the example, and the goal model that is linked to the data model via the weaving model. They are displayed in the EMF model editor in Figure 7.18. There are five measures belonging to one fact, five metrics (one for each measure), and five goals (one for each metric).

The datamodel file can be derived from the configuration of the BI tool. In the example presented here, the `Foodmart.xml` (Listing 7.3) contains the necessary information.

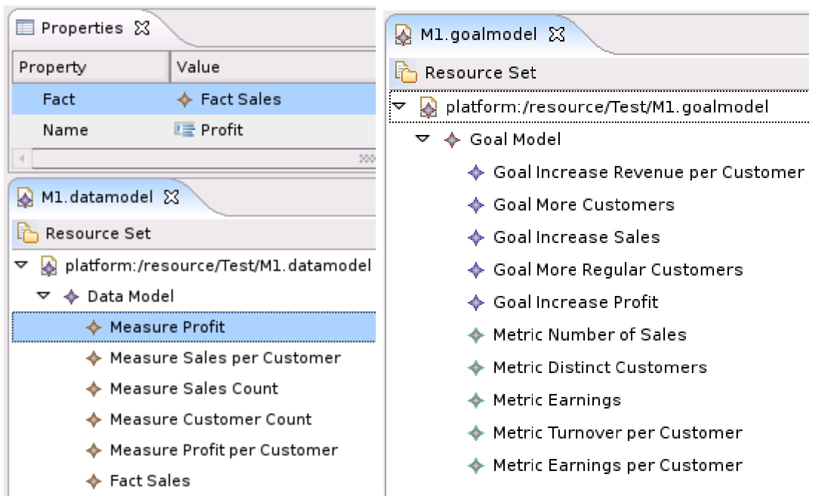
Figure 7.20 finally shows the weaving model that produces the business metadata: Each measure is linked to a metric.

7.3 Installation

Please note: The prototype was built on a number of components that were easily available during the writing of this thesis. Due to the fact that especially the modeling components (Eclipse, EMF, ATL and AMW) are undergoing frequent updates and changes, and not all versions are compatible to each other, it is very probable that at some time in the future it will be next to impossible to recreate the environment required by the prototype in its present form. Nevertheless, it should be quite easily possible to adapt the source code to newer versions of the modeling components.

7.3.1 Prerequisites

The Bermuda prototype for model-based business metadata builds on many existing components. For deploying and using it, the following applications should be provided:



Properties

Property	Value
Fact	Fact Sales
Name	Profit

M1.datamodel

Resource Set
- platform:/resource/Test/M1.datamodel
  - Data Model
    - Measure Profit
    - Measure Sales per Customer
    - Measure Sales Count
    - Measure Customer Count
    - Measure Profit per Customer
    - Fact Sales

M1.goalmodel

Resource Set
- platform:/resource/Test/M1.goalmodel
  - Goal Model
    - Goal Increase Revenue per Customer
    - Goal More Customers
    - Goal Increase Sales
    - Goal More Regular Customers
    - Goal Increase Profit
    - Metric Number of Sales
    - Metric Distinct Customers
    - Metric Earnings
    - Metric Turnover per Customer
    - Metric Earnings per Customer

Figure 7.18: Goal and data model: Measure *Profit* belongs to the *Sales* fact

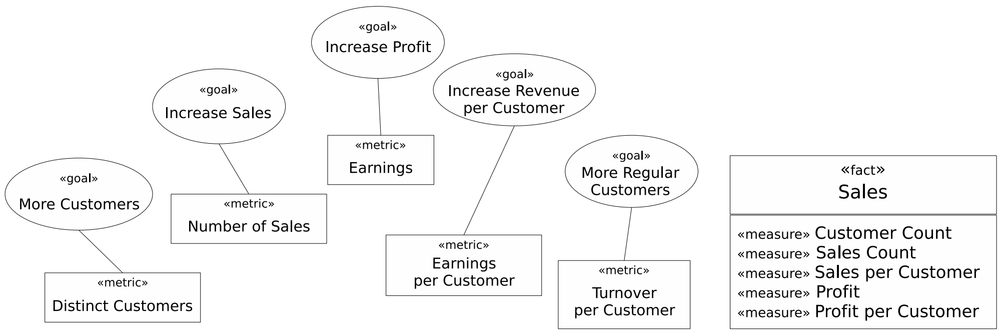
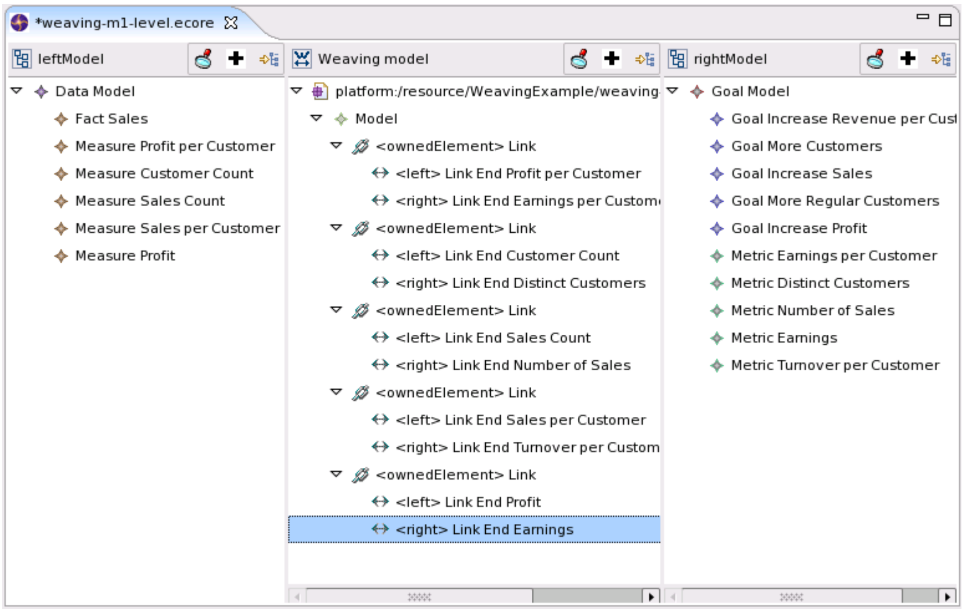

«goal» Increase Profit
«goal» Increase Sales
«goal» Increase Revenue per Customer
«goal» More Customers
«goal» More Regular Customers
«metric» Earnings
«metric» Number of Sales
«metric» Distinct Customers
«metric» Earnings per Customer
«metric» Turnover per Customer
«fact» Sales
«measure» Customer Count
«measure» Sales Count
«measure» Sales per Customer
«measure» Profit
«measure» Profit per Customer

Figure 7.19: The instances of the goal and data models (see also Figure 7.19)



```
*weaving-m1-level.ecore

leftModel
  Data Model
      Fact Sales
      Measure Profit per Customer
      Measure Customer Count
      Measure Sales Count
      Measure Sales per Customer
      Measure Profit

Weaving model
  platform:/resource/WeavingExample/weaving
    Model
      <ownedElement> Link
          <left> Link End Profit per Customer
          <right> Link End Earnings per Custom
      <ownedElement> Link
          <left> Link End Customer Count
          <right> Link End Distinct Customers
      <ownedElement> Link
          <left> Link End Sales Count
          <right> Link End Number of Sales
      <ownedElement> Link
          <left> Link End Sales per Customer
          <right> Link End Turnover per Custom
      <ownedElement> Link
          <left> Link End Profit
          <right> Link End Earnings

rightModel
  Goal Model
      Goal Increase Revenue per Cust
      Goal More Customers
      Goal Increase Sales
      Goal More Regular Customers
      Goal Increase Profit
      Metric Earnings per Customer
      Metric Distinct Customers
      Metric Number of Sales
      Metric Earnings
      Metric Turnover per Customer
```

Figure 7.20: The weaving model that links the goal model to the data model

- Java application server (Servlet/JSP container, Web server); e.g., Apache Tomcat
- JDBC-capable database; e.g., MySQL
- Foodmart example data imported into database (available from many sources and in many formats; e.g., from the JPivot project site)
- JPivot sample application configured to use the database and deploy on the server (follow the instructions provided by JPivot), JPivot source available (alternatively the sample.war (see below) can be deployed directly).

For editing the model files, the following is necessary:

- Eclipse SDK (3.2. was used during the development)
- Matching EMF, ATL and AMW plugins installed in Eclipse, including all their prerequisite plugins.

7.3.2 Files

Table 7.2 lists the files provided by the Bermuda project.

bermuda.jar	JPivot plugin, to be added to JPivot installation
bermuda-src.jar	Source code of the above
installation.txt	installation notes
models.zip	model files of the example
sample.war	sample Web application (built on JPivot sample), for models.zip

Table 7.2: Files of the project

7.3.3 Installation

The following steps are required for using the Bermuda plugin with the components listed in Section 7.3.1; alternatively the sample.war can be deployed to the server directly as a whole:

- add Bermuda class files to JPivot build path, register TableExtension in config.xml
- add Model files and JSP pages to WEB-INF directory of the JPivot sample installation
- call example.jsp from Web browser

With regard to changing or replacing the model files, they can be edited with the Eclipse installation including all plugins as described in Section 7.3.1, and then added to the models directory of the web application. Depending on the amount of change, the query provided by example.jsp and/or the OLAP cube defined in Foodmart.xml would have to be changed as well.

Chapter 8

Conclusion and Future Work

This thesis has targeted the relationship between data warehouses and the structure, behavior and goals of the organization. In order to describe this relationship in a formal way, a conceptual modeling language for Data Warehousing was developed. It consists of models for different aspects:

- Models for describing the interdependencies between data warehouses and business processes, from high level models showing large components of the data warehouse architecture all the way to detailed models of the individual attributes of the data model entities, and including active and (near-) real-time data warehouse solutions.
- Models for identifying business objects such as customers and products in the data warehouse data model, and for constructing data models that comply to the state models of such business objects.
- Models of data warehouse usage, which includes modeling the users, users groups, and user skill levels, the intensity with which they use the data warehouse infrastructure, temporal issues such as the required time and urgency of data access, as well as indicators of the relative importance of data warehouse usage.

Additionally to these new models for describing things that could not be described well or at all with existing models, this thesis also provides an approach to using models to enhance the way users access the data in the data warehouse. Through business metadata, it is possible to provide users with background information about the context and ideally the implications of what they are analyzing. This thesis has presented a model-based approach to business metadata, which links enterprise models such as business process models or goal models to the data model of the data warehouse though the mechanism of model weaving. Model weaving allows to manage relationships between elements from different models belonging to different domains. In the scope of this thesis, a total of seven weaving links have been developed, which link data warehouse elements to

- goals, metrics, and target values

- processes and functions,
- products and deliverables,
- organizational units and roles.

How models can be weaved and used for business metadata in a BI tool can be seen from a prototype developed as part of this thesis, thus illustrating the applicability of this approach.

This thesis presents the results of work conducted during the past three years. Naturally, many promising related questions could not be persued in this thesis. Because the research topic ventures into a new field, combining Data Warehousing and Enterprise Modeling, the results presented here are a starting point that hopefully will arouse interest and form the foundation on which others can build their work.

Regarding future work, promising challenges wait in many areas touched by this thesis:

Model Management and Automation Most benefits attributed to this thesis depend heavily on the models being up-to-date. It would therefore be worthwhile to investigate how far such models can be automatically constructed, or how situations that suggest that an update of the models could be necessary can be recognized. This could happen on the basis of log files or configuration files of, e.g., workflow systems, BI tools, ERP systems, etc.

Model-Driven Approaches Building on the work by [MTSP05a] and as described in the outlook in Section 6.6, the conceptual models presented in this thesis can be seen as CIM-level models in MDA. They can be used to enrich the MDA process, as they provide additional information. In the future, it may be become possible to not only construct the data warehouse data models automatically with MDA, but also access restriction specifications, user interface or personalization configurations, or performance tuning settings.

Security One of the main points of this thesis is providing users with more information. Information about the internal workings of a business organization is highly critical and has to be protected. By adding such models to the data warehouse infrastructure, the present-day problems of devising the optimal access rules to data warehouse data are aggravated.

Performance The prototype presented in Chapter 7 was designed to illustrate how models and modeling technology can be used to create business metadata, in the well-known Java J2EE web application environment. Performance issues beyond basic considerations - such as only accessing the models when required by the user - were not considered. EMF model access code is expected to scale quite well, but this was not investigated.

Data Warehouse Design and Development The scope of this thesis was deliberately limited to the relationship between the organization and the data warehouse itself, as an artifact. Nevertheless, data warehouse design methodology (a) is a topic generally worthwhile investigating, and (b) can definitely benefit from the approaches presented here.

Bibliography

[Act07] Actuate. Actuate Corporation. `http://www.actuate.com`, 2007.

[AGS97] Rakesh Agrawal, Ashish Gupta, and Sunita Sarawagi. Modeling Multidimensional Databases. In *Proceedings of the 13th Int. Conference on Data Engineering (ICDE 1997)*, pages 232–243. IEEE Computer Society, 1997.

[ASS02] Alberto Abelló, José Samos, and Felix Saltor. YAM^2 (Yet Another Multidimensional Model): An Extension of UML. In *Proceedings of the 6th International Database Engineering & Applications Symposium (IDEAS 2002)*, pages 172–181. IEEE Computer Society, 2002.

[ATL07] ATLAS Group. ATLAS Transformation Language. `http://www.eclipse.org/m2m/atl/`, 2007.

[AWM07] AWM. ATLAS Model Weaver. `http://www.eclipse.org/gmt/amw/`, 2007.

[BB02] Erwan Breton and Jean Bézivin. Weaving Definition and Execution Aspects of Process Meta-models. In *CD-ROM / Abstracts Proceedings of the 35th Hawaii Int. Conference on System Sciences (HICSS 2002)*, page 290. IEEE Computer Society, 2002.

[BCR94] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. *The Goal Question Metric Appraoch*. Wiley, 1994.

[Béz05] Jean Bézivin. On the Unification Power of Models. *Software and System Modeling*, 4(2):171–188, 2005.

[BFMB99] Mokrane Bouzeghoub, Françoise Fabret, and Maja Matulovic-Broqué. Modeling the Data Warehouse Refreshment Process as a Workflow Application. In *Proceedings of the Intl. Workshop on Design and Management of Data Warehouses (DMDW 1999)*, volume 19 of *CEUR Workshop Proceedings*, page 6. CEUR-WS.org, 1999.

[BG04] Andreas Bauer and Holger Günzel, editors. *Data-Warehouse-Systeme*. dpunkt.verlag, 2 edition, 2004.

[BIR07] BIRT. Business Intelligence and Reporting Tools Project. `http://www.eclipse.org/birt/phoenix/`, 2007.

[BJRV05] Jean Bézivin, Frédéric Jouault, Peter Rosenthal, and Patrick Valduriez. Modeling in the Large and Modeling in the Small. In *Revised Selected Papers of Model Driven Architecture, European MDA Workshops: Foundations and Applications, MDAFA 2003 and MDAFA 2004*, volume 3599 of *Lecture Notes in Computer Science*, pages 33–46, 2005.

[BJT05] Jean Bézivin, Frédéric Jouault, and David Touzet. An Introduction to the ATLAS Model Management Architecture. Technical report, LINA, 2005.

[BJV04] Jean Bézivin, Frédéric Jouault, and Patrick Valduriez. First Experiments with a ModelWeaver. In *Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2004)*, 2004.

[BLS01] Robert Bruckner, Beate List, and Josef Schiefer. Developing Requirements for Data Warehouse Systems with Use Cases. In *Proceedings of the 7th Americas Conference on Information Systems (AMCIS 2001)*, pages 329–335, 2001.

[Bor06] Borland Software Corporation. Together 2006 Release 2 for Eclipse. `http://techpubs.borland.com/together/`, 2006.

[BPM04] Business Process Modeling Notation BPMN. Specification BPMN 1.0 May 3, 2004. `www.bpmn.org/Documents/BPMNV1-0May32004.pdf`, 2004.

[BR01] Stephen Brobst and Joe Rarey. The Five Stages of an Active Data Warehouse Evolution. *Teradata Magazine*, 2001.

[BSHD98] Markus Blaschka, Carsten Sapia, Gabriele Höfling, and Barbara Dinter. Finding Your Way through Multidimensional Data Models. In *Proceedings of the 9th Int. Conference on Database and Expert Systems Applications (DEXA 1998)*, pages 198–203. Springer, 1998.

[CD97] Surajit Chaudhuri and Umeshwar Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Rec.*, 26(1):65–74, 1997.

[CDN$^+$06] Diego Calvanese, Luigi Dragone, Daniele Nardi, Riccardo Rosati, and Stefano Trisolini. Enterprise modeling and Data Warehousing in Telecom Italia. *Information Systems*, 31(1):1–32, 2006.

[CGL$^+$98] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Information Integration: Conceptual Modeling and Reasoning Support. In *Proceedings of the 3rd International Conference on Cooperative Information Systems (IFCIS 1998)*, pages 280–291. IEEE Computer Society, 1998.

[Che76] Peter P. Chen. The Entity-Relational model: Toward a Unified View of Data. *ACM Transactions on Database Systems*, 1(1), 1976.

[CT98] Luca Cabibbo and Riccardo Torlone. A Logical Approach to Multidimensional Databases. In *Advances in Database Technology - Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98)*, pages 183–197. Springer, 1998.

[Dem97] Marc Demarest. The Politics of Data Warehousing. `http://www.noumenal.com/marc/dwpoly.html`, 1997.

[dFBJ$^+$05] Marcos Didonet del Fabro, Jean Bézivin, Feédéric Jouault, Erwan Breton, and Guillaume Gueltas. AMW: A Generic Model Weaver. In *Ingénierie Dirigée par les Modèles (IDM 2005)*, 2005.

[Ecl07] Eclipse Foundation. Eclipse Platform. `http://www.eclipse.org/`, 2007.

[EMF07] EMF. Eclipse Modeling Framework. `http://www.eclipse.org/emf/`, 2007.

[Eng07] Engineering Ingegneria Informatica. SpagoBI Business Intelligence Free Platform. `http://spagobi-info.eng.it/`, 2007.

[Eve06] Dan Everett. Open Source BI Makes Inroads. *Intelligent Enterprise*, 6 2006.

[fFAZ07] Zachman Institute for Framework Advancement ZIFA. Zachman Framework for Enterprise Architecture. `http://www.zifa.com/`, 2007.

[FK04] Enrico Franconi and Anand Kamble. A Data Warehouse Conceptual Data Model. In *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004)*, pages 435–436. IEEE Computer Society, 2004.

[FMTVP04] Eduardo Fernández-Medina, Juan Trujillo, Rodolfo Villarroel, and Mario Piattini. Extending UML for Designing Secure Data Warehouses. In *Conceptual Modeling - ER 2004, 23rd Int. Conference on Conceptual Modeling*, volume 3288 of *Lecture Notes in Computer Science*, pages 217–230. Springer, 2004.

[FMTVP07] Eduardo Ferández-Medina, Juan Trujillo, Rodolfo Villarroel, and Mario Piattini. Developing Secure Data Warehouses with a UML Extension. *Information Systems*, 32(6):826–856, 2007.

[Fra00] Ulrich Frank. Modelle als Evaluationsobjekt. In Lutz Heinrich, Irene Häntschel, and Ulrich Frank, editors, *Evaluation und Evaluationsforschung in der Wirtschaftsinformatik*. Oldenbourg, 2000.

[Fra02] Ulrich Frank. Multi-perspective Enterprise Modeling (MEMO) - Conceptual Framework and Modeling Languages. In *CD-ROM / Abstracts Proceedings of the 35th Hawaii Int. Conference on System Sciences (HICSS 2002)*, volume 3, page 72. IEEE Computer Society, 2002.

[Fra06] Ulrich Frank. Evaluation of Reference Models. In Peter Fettke and Peter Loos, editors, *Reference Modeling for Business Systems Analysis*, pages 118–140. Idea Group Inc., 2006.

[FS99] Enrico Franconi and Ulrike Sattler. A Data Warehouse Conceptual Data Model for Multidimensional Aggregation. In *Proceedings of the Intl. Workshop on Design and Management of Data Warehouses (DMDW 1999)*, pages 13–1–13–10. Technical University of Aachen (RWTH), 1999.

[GL97] Marc Gyssens and Laks Lakshmanan. A Foundation for Multi-Dimensional Databases. In *Proceedings of the 23rd Int. Conference on Very Large Data Bases (VLDB 1997)*, pages 106–115. Morgan Kaufmann, 1997.

[GMF07] GMF. Eclipse Graphical Modeling Framework. `http://www.eclipse.org/modeling/gmf/`, 2007.

[GMR98a] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. Conceptual Design of Data Warehouses from E/R Schemes. In *CD-ROM / Abstracts Proceedings of the 31st Hawaii Int. Conference on System Sciences (HICSS 1998)*, volume 7, pages 334–343. IEEE Computer Society, 1998.

[GMR98b] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. The Dimensional Fact Model: A Conceptual Model for Data Warehouses. *Cooperative Information Systems*, 7(2-3):215–247, 1998.

[GRC04] Matteo Golfarelli, Stefano Rizzi, and Iuris Cella. Beyond Data Warehousing: What's Next in Business Intelligence? In *Proceedings of the ACM 7th Int. Workshop on Data Warehousing and OLAP (DOLAP 2004)*. ACM Press, 2004.

[GRG05] Paolo Giorgini, Stefano Rizzi, and Maddalena Garzetti. Goal-oriented Requirement Analysis for Data Warehouse Design. In *Proceedings of the ACM 8th Int.*

Workshop on Data Warehousing and OLAP (DOLAP 2005), pages 47–56. ACM Press, 2005.

[Ham96] Michael Hammer. *Beyond Reengineering - How the Process-Centered Organization is Changing Our Work and Our Lives*. Harper Collins Publishers, 1996.

[HKKR05] Martin Hitz, Gerti Kappel, Elisabeth Kapsammer, and Werner Retschitzegger. *UML@Work*. dpunkt.verlag, Heidelberg, 3 edition, 2005.

[HLV00] Bodo Hüsemann, Jens Lechtenbörger, and Gottfried Vossen. Conceptual Data Warehouse Modeling. In *Proceedings of the 2nd Int. Workshop on Design and Management of Data Warehouses (DMDW 2000)*, page 6, 2000.

[HRG83] A. Holt, H.R. Ramsey, and J.D. Grimes. Coordination System Technology as the Basis for a Programming Environment. *Electrical Communication*, 57(4):307–314, 1983.

[Hyd07] Julian Hyde. Mondrian Architecture Documentation. `http://mondrian.pentaho.org/documentation/architecture.php`, 2007.

[IH94] William Inmon and Richard Hackathorn. *Using the Data Warehouse*. John Wiley & Sons, New York, 1994.

[Inm02] William H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, Inc., 3 edition, 2002.

[Inm07] Bill Inmon. Inmon Consulting Services. `http://www.inmoncif.com/`, 2007.

[Jas07a] JasperSoft. iReport Visual Report Builder. `http://www.jasperforge.org/sf/projects/ireport`, 2007.

[Jas07b] JasperSoft. JasperAnalysis, JasperSoft BI 2.0 Suite. `http://www.jaspersoft.com/JasperSoft_JasperAnalysis.html`, 2007.

[Jas07c] JasperSoft. JasperSoft Open Source Business Intelligence. `http://www.jaspersoft.com`, 2007.

[JLV$^+$01] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, Y. Vassiliou, M. Lenzerini, P. Vassiliadis, and Panos Vassiliadis. *Fundamentals of Data Warehouses*. Springer, 2001.

[JPi07] JPivot. JPivot Project. `http://jpivot.sourceforge.net/`, 2007.

[JS05] Mary Elizabeth Jones and Il-Yeol Song. Dimensional Modeling: Identifying, Classifying & Applying Patterns. In *Proceedings of the ACM 8th Int. Workshop on Data Warehousing and OLAP (DOLAP 2005)*, pages 29–38. ACM Press, 2005.

[Kav04] Evangelia Kavakli. Modeling Organizational Goals: Analysis of Current Methods. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC 2004)*, pages 1339–1343, 2004.

[Kim99] Ralph Kimball. Fundamental Grains. *Intelligent Enterprise*, 2(5), 3 1999.

[Kim07] Ralph Kimball. Ralph Kimball Associates. `http://www.ralphkimball.com/`, 2007.

[Kit96] Barbara Kitchenham. *Software Metrics: Measurement for Software Process Improvement*. Blackwell, 1996.

[KKST97] Remzi Kirkgöze, Nevena Katic, Mladen Stolba, and A. Min Tjoa. A Security Concept for OLAP. In *Proceedings of the 8th Int. Conference on Database and Expert Systems Applications (DEXA 1997)*, page 619. IEEE Computer Society, 1997.

[KL04] Evangelia Kavakli and Pericles Loucopoulos. *Goal Driven Requirements Engineering: Analysis and Critique of Current Methods*, pages 102 – 124. IDEA Group, 2004.

[KLM$^+$97] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-Oriented Programming. In *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP 1997)*, volume 1241 of *Lecture Notes in Computer Science*, pages 220–242. Springer, 1997.

[KN92] Robert S. Kaplan and David P. Norton. The Balanced Scorecard - Measures that Drive Performance. *Harvard Business Review*, pages 72–79, 1992.

[KNS92] G Keller, M Nüttgens, and A.-W. Scheer. Semantische Prozeßmodellierung auf der Grundlage "Ereignisgesteuerter Prozeßketten (EPK)". *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, 89, 1992.

[KR02] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. John Wiley & Sons, Inc., 2 edition, 2002.

[KRRT98] Ralph Kimball, Laura Reeves, Margy Ross, and Warren Thornthwaite. *The Data Warehouse Lifecycle Toolkit*. John Wiley & Sons, Inc., 1998.

[Kue00] Peter Kueng. Process Performance Measurement System: A Tool to Support Process-Based Organizations. *Total Quality Management*, 11(1):67–85, 2000.

[KV92] Kurt Kosanke and François Vernadat. CIM-OSA: A Reference Architecture for CIM. In *Proceedings of the IFIP TC5 / WG5.3 Eight International PROLAMAT Conference on Human Aspects in Computer Integrated Manufacturing*, pages 41–48. North-Holland Publishing Co., 1992.

[LK97] Pericles Loucopoulos and Vagelio Kavakli. Enterprise Knowledge Management and Conceptual Modelling. In *Conceptual Modeling - ER 1997, 16th Int. Conference on Conceptual Modeling*, pages 123–143. Springer, 1997.

[LK03] Jim Lawler and Barbara Kitchenham. Measurement Modeling Technology. *IEEE Software*, pages 68–75, 2003.

[LK06] Beate List and Birgit Korherr. An Evaluation of Conceptual Business Process Modelling Languages. In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006)*, pages 1532–1539. ACM Press, 2006.

[LM04] Beate List and Karl Machaczek. Towards a Corporate Performance Measurement System. In *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC 2004)*. ACM Press, 2004.

[LMT04a] Sergio Luján-Mora and Juan Trujillo. A Data Warehouse Engineering Process. In *Proceedings of the 3rd Biennial Int. Conference on Advances in Information Systems (ADVIS 2004)*, volume 3261 of *Lecture Notes in Computer Science*, pages 14–23. Springer, 2004.

[LMT04b] Sergio Luján-Mora and Juan Trujillo. Physical modeling of data warehouses using UML. In *Proceedings of the ACM 7th Int. Workshop on Data Warehousing and OLAP (DOLAP 2004)*, pages 48–57. ACM Press, 2004.

[LMTS02] Sergio Luján-Mora, Juan Trujillo, and Il-Yeol Song. Extending the UML for Multidimensional Modeling. In *Proceedings of the 5th International Conference on The Unified Modeling Language (UML 2002)*, volume 2460 of *Lecture Notes in Computer Science*, pages 290–304. Springer, 2002.

[LMTS06] Sergio Luján-Mora, Juan Trujillo, and Il-Yeol Song. A UML Profile for Multidimensional Modeling in Data Warehouses. *Data Knowl. Eng.*, 59(3):725–769, 2006.

[LW96] Chang Li and Xiaoyang Sean Wang. A Data Model for Supporting On-Line Analytical Processing. In *Proceedings of the 5th Int. Conference on Information and Knowledge Management (CIKM 1996)*, pages 81–88. ACM Press, 1996.

[MDC99] MDC. Meta Data Coalition Open information model. `http://www.mdcinfo.com`, 1999.

[MDG+07] William Moore, David Dean, Anna Gerber, Gunnar Wagenknecht, and Philippe Vanderheyden. Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework. `http://www.redbooks.ibm.com/abstracts/sg246302.html?Open`, 2007.

[MMd$^+$95] R.J. Mayer, C.P. Menzel, P.S. deWitte, T. Blinn, and B. Perakath. Information integration for concurrent engineering (IICE) IDEF3 Process Description Capture Method Report. Technical report, Knowledge Based Systems Incorporated (KBSI), 1995.

[MPT07] José-Norberto Mazón, Jésus Pardillo, and Juan-Carlos Trujillo. A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses. In *RIGiM Workshop, ER 2007, to appear*, 2007.

[MSAP07] Tarek Ben Mena, Narjès Bellamine-Ben Saoud, Mohamed Ben Ahmed, and Bernard Pavard. Towards a Methodology for Context Sensitive Systems Development. In *Proceedings of the 6th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT 2007)*, volume 4635 of *Lecture Notes in Computer Science*, pages 56–68. Springer, 2007.

[MT06] José-Norberto Mazón and Juan-Carlos Trujillo. An MDA Approach for the Development of Data Warehouses. *Decision Support Systems*, in press, 2006.

[MTL07] José-Norberto Mazón, Juan-Carlos Trujillo, and Jens Lechtenbörger. Reconciling rRequirement-Driven Data Warehouses with Data Sources via Multidimensional Normal Forms. *Data Knowl. Eng.*, in press, 2007.

[MTSP05a] Jose-Norberto Mazon, Juan Trujillo, Manuel Serrano, and Mario Piattini. Applying MDA to the Development of Data Warehouses. In *Proceedings of the ACM 8th Int. Workshop on Data Warehousing and OLAP (DOLAP 2005)*, pages 57–66. ACM Press, 2005.

[MTSP05b] José-Norberto Mazon, Juan-Carlos Trujillo, M. Serrano, and M. Piattini. Designing Data Warehouses: From Business Requirement Analysis to Multidimensional Modeling. In *Proceedings of the 1st International Workshop on Requirements Engineering for Business Need and IT Alignment, at the 14th Int. Requirements Engineering Conference*, 2005.

[NTW00] Thanh Binh Nguyen, A Min Tjoa, and Roland Wagner. An Object Oriented Multidimensional Data Model for OLAP. In *Proceedings of the Int. Conference on Web-Age Information Management (WAIM 2000)*, pages 69–82. Springer, 2000.

[OMG00] Object Management Group OMG. Workflow Management Facility Specification, V1.2. `http://www.omg.org/docs/formal/00-05-02.pdf`, 2000.

[OMG03a] Object Management Group OMG. Meta Object Facility (MOF) 2.0 Core Specification. `http://www.omg.org/cgi-bin/apps/doc?ptc/03-10-04.pdf`, 2003.

[OMG03b] Object Management Group OMG. Model Driven Architecture (MDA). http://www.omg.org/cgi-bin/doc?formal/03-06-01, 2003.

[OMG04] Object Management Group OMG. Business Process Definition Metamodel (BPDM), 2004.

[OMG05a] Object Management Group OMG. MOF Query / Views / Transformations (QVT). http://www.omg.org/cgi-bin/doc?ptc/2005-11-01, 2005.

[OMG05b] Object Management Group OMG. UML 2.0 Object Constraint Language (OCL) Specification. http://www.omg.org/cgi-bin/apps/doc?ptc/05-06-06.pdf, 2005.

[OMG05c] Object Management Group OMG. UML 2.0 Superstructure. http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04.pdf, 2005.

[OMG05d] Object Management Group OMG. XMI Mapping Specification, v2.1). http://www.omg.org/cgi-bin/doc?formal/2005-09-01, 2005.

[OMG07] Object Management Group (OMG). http://www.omg.org/, 2007.

[Omo07] Omondo Europa, Inc. EclipseUML. http://www.omondo.de/, 2007.

[Ope07] OpenI.org. Open Source Web Application for OLAP Reporting. http://openi.sourceforge.net/, 2007.

[PACW06] Nicolas Prat, Jacky Akoka, and Isabelle Comyn-Wattiau. A UML-based Data Warehouse Design Method. *Decision Support Systems*, 42(3):1449–1473, 2006.

[Pea02] William Pearson. MDX at First Glance: Introduction to SQL Server MDX Essentials. *Database Journal, http://www.databasejournal.com/features/mssql/article.php/10894_1495511_2*, 11 2002.

[Pen07a] Pentaho Analysis Services. Mondrian Project. http://mondrian.pentaho.org/, 2007.

[Pen07b] Pentaho Corporation. Pentaho Cube Designer. http://mondrian.pentaho.org/, 2007.

[Pen07c] Pentaho Corporation. Pentaho Open-Source Business Intelligence. http://www.pentaho.com, 2007.

[Pen07d] Pentaho Corporation. Pentaho Open Source Business Intelligence Platform Technical White Paper. http://downloads.sourceforge.net/pentaho/Pentaho_Technical_Whitepaper-1-6.pdf, 2007.

[Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

[Pet66] Carl Adam Petri. Kommunikation mit Automaten. *New York: Griffiss Air Force Base, Technical Report RADC-TR-65–377*, 1:1–Suppl. 1, 1966. English translation.

[PP01] Torsten Priebe and Günther Pernul. A Pragmatic Approach to Conceptual Modeling of OLAP Security. In *Conceptual Modeling - ER 2001, 20th Int. Conference on Conceptual Modeling*, volume 2224 of *Lecture Notes in Computer Science*, pages 311–324. Springer, 2001.

[RALT06] Stefano Rizzi, Alberto Abelló, Jens Lechtenbörger, and Juan Trujillo. Research in Data Warehouse Modeling and Design: Dead or Alive? In *Proceedings of the ACM 9th Int. Workshop on Data Warehousing and OLAP (DOLAP 2006)*, pages 3–10. ACM Press, 2006.

[Riz04] Stefano Rizzi. Position Statement. In *Dagstuhl Seminar "Data Warehousing at the Crossroads"*. Schloss Dagstuhl, International Conference and Research Center for Computer Science, 8 2004.

[RJB04] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2 edition, 2004.

[Sar01] Nandlal L. Sarda. Structuring Business Metadata in Data Warehouse Systems for Effective Business Support. *CoRR, http://arxiv.org/abs/cs.DB/0110020*, cs.DB/0110020, 2001.

[SBHD99] Carsten Sapia, Markus Blaschka, Gabriele Hoefling, and Barbara Dinter. Extending the E/R Model for the Multidimensional Paradigm. In *Conceptual Modeling - ER 1998, 17th Int. Conference on Conceptual Modeling*, volume 1552 of *LNCS*, pages 105–116. Springer, 1999.

[Sch99] A.-W. Scheer. *ARIS - Business Process Modeling*. Springer, 1999.

[Sim05] Alkis Simitsis. Mapping Conceptual to Logical Models for ETL Processes. In *Proceedings of the ACM 8th Int. Workshop on Data Warehousing and OLAP (DOLAP 2005)*, pages 67–76. ACM Press, 2005.

[SL05] Veronika Stefanov and Beate List. A Performance Measurement Perspective for Event-Driven Process Chains. In *Proceedings of the First International Workshop on Business Process Monitoring and Performance Management (BPMPM 2005), 16th International Workshop on Database and Expert Systems Applications (DEXA 2005)*, pages 967–971. IEEE Computer Society, 2005.

[SL06] Veronika Stefanov and Beate List. Business Metadata for the Data Warehouse - Weaving Enterprise Goals and Multidimensional Models. In *Proceedings of the International Workshop on Models for Enterprise Computing (IWMEC) at the 9th International Enterprise Distributed Object Computing Conference (EDOC 2006)*, pages 53 –61, 2006.

[SL07a] Veronika Stefanov and Beate List. A UML Profile for Modeling Data Warehouse Usage. In *Proceedings of the 3rd International Workshop on Foundations and Practices of UML (FP-UML 2007), 26th International Conference on Conceptual Modelling (ER 2007)*, volume 4802 of *Lecture Notes in Computer Science*, pages 137 – 147. Springer, 2007.

[SL07b] Veronika Stefanov and Beate List. A UML Profile for Representing Business Object States in a Data Warehouse. In *Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2007)*, volume 4654 of *Lecture Notes in Computer Science*, pages 209–220. Springer, 2007.

[SL07c] Veronika Stefanov and Beate List. Explaining Data Warehouse Data to Business Users - A Model-Based Approach to Business Metadata. In *Proceedings of the 15th European Conference on Information Systems (ECIS 2007)*, 2007.

[SLK05] Veronika Stefanov, Beate List, and Birgit Korherr. Extending UML 2 Activity Diagrams with Business Intelligence Objects. In *Proceedings of the 7th Int. Conference on Data Warehousing and Knowledge Discovery (DaWaK 2005)*, volume 3589 of *Lecture Notes in Computer Science*, pages 53–63. Springer, 2005.

[SLS05] Veronika Stefanov, Beate List, and Josef Schiefer. Bridging the Gap between Data Warehouses and Business Processes - A Business Intelligence Perspective for Event-Driven Process Chains. In *Proceedings of the 9th International Enterprise Distributed Object Computing Conference (EDOC 2005)*, pages 3–14. IEEE Computer Society, 2005.

[SSM$^+$08] Emilio Soler, Veronika Stefanov, Jose-Norberto Mazón, Juan Trujillo, Eduardo Fernández-Medina, and Mario Piattini. Modelado de Requisitos de Seguridad para Almacenes de Datos (Towards Comprehensive Requirement Analysis for Data Warehouses: Considering Security Requirements). In *XI Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS 2008)*, 2008.

[SVS05] Alkis Simitsis, Panos Vassiliadis, and Timos K. Sellis. Optimizing ETL Processes in Data Warehouses. In *Proceedings of the 21th Int. Conference on Data Engineering (ICDE 2005)*, pages 564–575. IEEE Computer Society, 2005.

[TBC99] Nectaria Tryfona, Frank Busborg, and Jens G. Borch Christiansen. starER: A Conceptual Model for Data Warehouse Design. In *Proceedings of the 2nd Int. Workshop on Data Warehousing and OLAP (DOLAP 1999)*, pages 3–8. ACM, 1999.

[TLM03] Juan Trujillo and Sergio Luján-Mora. A UML Based Approach for Modeling ETL Processes in Data Warehouses. In *Conceptual Modeling - ER 2003, 22rd Int. Conference on Conceptual Modeling*, volume 2813 of *Lecture Notes in Computer Science*, pages 307–320. Springer, 2003.

[Ton07] Tonbeller AG, JPivot. JPivot JavaDoc API Documentation. `http://jpivot.sourceforge.net/api/`, 2007.

[TPGS01] Juan Trujillo, Manuel Palomar, Jaime Gómez, and Il-Yeol Song. Designing Data Warehouses with OO Conceptual Models. *IEEE Computer*, 34(12):66–75, 2001.

[TPT07] TPTP. Eclipse Test & Performance Tools Platform Project. `http://www.eclipse.org/tptp/`, 2007.

[Vas00] Panos Vassiliadis. Gulliver in the Land of Data Warehousing: Practical Experiences and Observations of a Researcher. In *Proceedings of the 2nd Int. Workshop on Design and Management of Data Warehouses (DMDW 2000)*, page 12. CEUR-WS.org, 2000.

[vdBCC05] Klaas van den Berg, Jose Maria Conejero, and Ruzanna Chitchyan. AOSD Ontology 1.0 - Public Ontology of Aspect-Orientation. AOSD-Europe-UT-01 D9, May 2005.

[VM07] Dan Vesset and Brian McDonough. Worldwide Business Intelligence Tools 2006 Vendor Shares. Technical Report Doc # 207422, IDC, 6 2007.

[VS99] Panos Vassiliadis and Timos K. Sellis. A Survey of Logical Models for OLAP Databases. *SIGMOD Record*, 28(4):64–69, 1999.

[VSG$^+$05] Panos Vassiliadis, Alkis Simitsis, Panos Georgantas, Manolis Terrovitis, and Spiros Skiadopoulos. A Generic and Customizable Framework for the Design of ETL Scenarios. *Information Systems*, 30(7):492–525, 2005.

[VSS02] Panos Vassiliadis, Alkis Simitsis, and Spiros Skiadopoulos. Conceptual Modeling for ETL Processes. In *Proceedings of the ACM 5th Int. Workshop on Data Warehousing and OLAP (DOLAP 2002)*, pages 14–21. ACM Press, 2002.

[VVS00] Thomas Vetterli, Anca Vaduva, and Martin Staudt. Metadata Standards for Data Warehousing: Open Information Model vs. Common Warehouse Metamodel. *SIGMOD Record*, 29(3):68–75, 2000.

[WJW04] Lingyu Wang, Sushil Jajodia, and Duminda Wijesekera. Securing OLAP Data Cubes Against Privacy Breaches. *IEEE Symposium on Security and Privacy*, 2004:161, 2004.

[WMC07] Workflow Management Coalition WMC. Interface 1 - Process Defintion Interchange Model, Version 1.14. `http://www.wfmc.org/`, 2007.

[WRK01] Larry Whitman, Kartik Ramachandran, and Vikram Ketkar. A Taxonomy of a Living Model of the Enterprise. In *Proceedings of the 33nd conference on Winter Simulation (WSC 2001)*, pages 848–855. IEEE Computer Society, 2001.

[WS03] Robert Winter and Bernhard Strauch. A Method for Demand-Driven Information Requirements Analysis in Data Warehousing Projects. In *CD-ROM / Abstracts Proceedings of the 36th Hawaii Int. Conference on System Sciences (HICSS 2003)*, page 231, 2003.

[WTP07] WTP. Eclipse Web Tools Platform. `http://www.eclipse.org/webtools/`, 2007.

[Yu97] Eric S.K. Yu. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE 1997)*, pages 226–235. IEEE Computer Society, 1997.