Metadata Management and Data Warehousing[*]

Martin Staudt[†] Anca Vaduva[‡] Thomas Vetterli[†]

[†] Swiss Life, Information Systems Research (CH/IFuE),
P.O. Box, CH-8022 Zurich, Switzerland
<firstname>.<lastname>@swisslife.ch

[‡] University of Zurich, Department of Computer Science,
Winterthurerstr. 190, CH-8057 Zurich, Switzerland,
vaduva@ifi.unizh.ch

Abstract

This report gives an overview of metadata management in general (Part I) and on the role of metadata for data warehousing (Part II). Because of the complexity and extensive applicability of metadata, a compact, precise definition of the notion may hardly be provided. Therefore, we explain metadata by illustrating the use and the forms it may take within various application areas. In the case of data warehousing, we present a classification of metadata along certain dimensions and we discuss significant aspects of metadata management that have to be considered for the construction of a data warehouse system. Furthermore, this report provides a comprehensive survey and analysis of the state of the art of metadata management in industry and research. The most important standards for representation and interchange of metadata, commercial products and research projects are presented and discussed (as far as the available information allows) for both, the general case and the particular case of data warehousing.

[*]This work was supported in part by the Swiss Federal Office of Professional Education and Technology under grant KTI-3979.1 (SMART).

Contents

Introduction **7**

I Metadata Management **9**

1 The Role of Metadata **9**

2 Application Areas for Metadata **10**
2.1 Information Management Systems . . . 10
2.1.1 Databases and Integrated Database Systems . . . 10
2.1.2 Information Retrieval Systems . . . 11
2.1.3 The Web . . . 12
2.2 Software Engineering . . . 12
2.3 Multimedia . . . 13
2.3.1 Geographical/Spatial Information Systems . . . 14
2.3.2 Digital Libraries . . . 14
2.4 Telecommunication . . . 15

3 Standards for Metadata Management **15**
3.1 Standard Dimensions . . . 16
3.2 RM-ODP . . . 17
3.3 ISO/IEC 11179 . . . 18
3.4 MOF . . . 19
3.5 UML . . . 19
3.6 OIM . . . 20
3.7 IDEF . . . 21
3.8 CDIF . . . 22
3.9 MDIS . . . 22
3.10 IRDS . . . 23
3.11 PCTE . . . 23
3.12 XML . . . 24
3.13 RDF . . . 24
3.14 XMI . . . 25
3.15 XIF . . . 25
3.16 Standards for Business Models . . . 25
3.17 The Zachman Framework . . . 26

4 Commercial Support for Metadata Management **27**
4.1 UREP . . . 27
4.2 Microsoft Repository . . . 28
4.3 Platinum Repository . . . 28
4.4 Rochade . . . 29

5 Research Projects on Metadata Management 30
5.1 ConceptBase . . . 30
5.2 H-PCTE . . . 31
5.3 Lore . . . 32
5.4 Ontology Management . . . 33
5.5 Information Integration . . . 34
5.6 Website Management . . . 34

6 Conclusions 35

II Metadata for Data Warehousing 37

1 Data Warehousing 37
1.1 Data Warehouses versus Operational Systems . . . 37
1.2 Building the Data Warehouse . . . 38
1.3 Using the Data Warehouse . . . 39
1.4 Designing the Data Warehouse . . . 40
1.5 Maintaining the Data Warehouse . . . 40

2 Metadata for Data Warehousing 41
2.1 Objectives . . . 41
2.2 Classification of Metadata . . . 43
2.2.1 Criterion "Type" . . . 43
2.2.2 Criterion "Level of Abstraction" . . . 44
2.2.3 Criterion "User View" . . . 44
2.2.4 Criterion "Origin" . . . 45
2.2.5 Criterion "Purpose" . . . 45
2.2.6 Criterion "Producing/Consuming Time" . . . 45
2.3 Metadata Management . . . 46
2.3.1 Functionality . . . 46
2.3.2 Repository Architecture . . . 47

3 Standardization Efforts for Data Warehouse Metadata 48
3.1 MDAPI . . . 48
3.2 CWMI . . . 49
3.3 OIM Extensions for Data Warehousing . . . 49
3.4 The Zachman Framework applied to Data Warehousing . . . 50

4 Commercial Data Warehouse Metadata Management 50
4.1 Metadata Integration Approaches . . . 51
4.2 Platinum . . . 51
4.3 Rochade . . . 51
4.4 Oracle . . . 52
4.5 IBM . . . 53
4.6 Ardent Software . . . 53
4.7 Informatica . . . 54

5 Data Warehouse Research Projects **55**
5.1 Whips 55
5.2 Squirrel 56
5.3 DWQ 56
5.4 Meta-FIS 58
5.5 System 42 58
5.6 CC-DWS 59
5.7 SMART and SIRIUS 59
5.8 MMDWE 60

6 Conclusions **60**

Introduction

The topic is as old as data exist: metadata have ever been needed to describe the meaning or properties of data with the aim to better understand, manage, and use that data. A classical example are libraries. The books (data) may be classified, managed and retrieved only by means of appropriate metadata (i.e., title, author and content keywords).

Metadata is commonly understood as any information needed in information technology in order to analyse, design, build, implement and then use computer systems [8]. In the case of information systems, metadata particularly facilitates managing, querying, consistent use and understanding of data.

Many recent efforts within both the academic and industrial community have concentrated on issues related to metadata. The generation, storage and management of metadata promise to better support the exploitation of the huge amount of data available nowadays in every conceivable electronic form. Since everything computers work with is inherently data and (a kind of) metadata accompanies any data, the notion may be found in any thinkable application domain and takes various forms depending on its use. The notion of "meta" is closely related to modelling. For the modelling of complex information systems, at least 4 modeling levels are required. Each level is the "meta" level for the next lower level, that means it contains the modelling constructs (or the modeling language) used to define the information on the level below. To start with, on the lowest level, level 0, there are the actual data items (e.g., the customer data). The levels above contain the metainformation: level 1 contains metadata (e.g., the database schema), level 2 specifies the schema used to store the metadata (the so-called metadata schema). Usually, level 2 also includes common modeling languages like UML that, for example, may directly be used to define a database schema. Level 3 contains the meta meta model that unifies the different modelling languages specified on the level 2. However, the concept of metadata is relative, not absolute and the delimitation between data and metadata is often blurred: what may be considered metadata in one context, may look like data in another.

This paper aims to clarify the concept of metadata, to discuss its use and the problems related to metadata management in general and in the particular case of data warehousing. To this end, we present the state-of-the-art of metadata management in industry and research and we address aspects like the use of metadata in different application areas, existing standards for interchange, representation and modelling of metadata, commercial products for management of metadata in various fields, and existing research related to metadata.

During the last years, data warehousing gained significant importance for competitive enterprises. In order to cope with the complexity of building, using and maintaining a data warehouse, a metadata management system is indispensable. It may be used by other components in the data warehouse system or by human beings to effectively and efficiently achieve their particular tasks. The significance of metadata management in data warehousing is the basic motivation for Part II of the survey which provides an overview of the topic; we classify metadata along certain dimensions and present the existing standards, commercial products, and existing research projects so far.

To summarize, this survey is organized as follows. Part I discusses the use of metadata in software systems (Section 1) and presents an overview of relevant areas in connection with metadata management in Section 2. Section 3 provides a survey of existing standards for representation and interchange of metadata. Commercial products that promote metadata repositories are addressed in Section 4 and research actions related to metadata in Section 5.

In Section 6 we give a summary and draw some conclusions concerning future developments.

Part II focuses on data warehousing. First of all, the basic concepts are introduced in Section 1. Section 2 presents the objectives of metadata management for data warehousing and shows the various existing metadata types. This classification is thought as the first step towards a uniform conceptual model for metadata management. Additionally, we discuss various aspects regarding the state-of-the-art of metadata management. In this context, Section 3 and Section 4 present existing standards and representative commercial solutions for metadata management in data warehouse systems. Section 5 introduces existing research projects wrt. to their main focus and goals. Section 6 concludes Part II with a summary and enumerates certain open problems.

Part I
Metadata Management

1 The Role of Metadata

The main purpose of metadata is to provide a consistent documentation about the structure, the development process and the use of a computerized system. Inherently, the availability of documentation helps understanding the system and efficiently supports the work of all "actors" involved in the development and exploitation of the system: end-users, system administrators, and application developers. More precisely, the generation and management of metadata may contribute to achieve the following tasks and objectives:

improving interaction with the system. Interaction may be performed either by means of simply browsing or querying the system or by advanced access and analysis applications. Metadata documents what exists in the system, how to use the system, the existing business concepts and terminology, details about predefined queries and reports. Furthermore, it allows to pose precise, well-directed queries and generally reduces the cost to a user of accessing, evaluating, and using appropriate information. Sometimes metadata is even indispensable for query and retrieval. For example, certain approaches use query parsers that need metadata in order to establish the syntactical correctness of a query (e.g, to determine the existence of an attribute). Furthermore, metadata is necessary to understand the application domain and its representation in the data model (in the classical case the database schema) in order to adequately apply and interpret the results.

improving data quality. In information systems, data has to be consistent, up-to-date, accurate and complete. Thus, metadata may reveal where the data came from, what happened to it, what it means, and who is responsible for its quality.

supporting the system integration process. The integration of heterogeneous database and information systems (e.g., the construction of multidatabase or federated database systems) or simply the interoperability of various data sources require metadata about the structure and the meaning of the individual data sources. Also the integration of individual software components (which are not necessarily information systems) depends on the availability of descriptions of interfaces and other metadata regarding the constituent components.

supporting system maintenance, analysis and design of new applications and business process modeling. Metadata increases control and reliability of the application development process by providing information about the structure, meaning and origin of data and by providing documentation of the existing applications and software that has to be extended.

Two other objectives of metadata management that particulary gained importance for data warehousing during the last years, will be reconsidered in Part II:

automatization of various administration tasks. Metadata partly specify control mechanisms that are directly accessed and used at runtime.

increasing flexibility. This is possible if certain semantic aspects (i.e., program code parts), which are subject to steady change, are explicitly stored as metadata instead of being hidden within the programs. At runtime, this metadata can be read and interpreted. In this way, software may be easily extended and adapted when new requirements arise.

Metadata is produced, accessed and updated by either human users and/or software tools. End-users and technical users (like database administrators or application programmers) themselves may enter the metadata to be stored for later use. Software tools or certain components of the sofware system may also be producers of metadata which describes data and information handled by those components.

Possible metadata consumers are either components of the system or humans. Depending on the particular tasks they have to achieve, users query metadata to look for information about the system.

2 Application Areas for Metadata

Different user communities employ metadata in different ways in order to achieve their tasks in fields like earth and space sciences, medicine, cultural heritage, education, law and, of course, business. Various conferences with focus on metadata [3, 4, 7, 8] prove the world-wide interest that exists in providing, managing and using metadata to improve daily electronic data processing of data.

This section intends to give an overview of the areas where work on metadata management exists. We briefly illustrate typical metadata and its use for each area and present some approaches to manage it. Note that for space reasons, some application fields (like metadata for statistical applications [39]) are not discussed, in spite of their incontestable importance.

With respect to their metadata characteristics, we distinguish between four classes of applications which inevitably have many overlaps: *Information Management Systems*, *Telecomunication*, *Software Engineering* and *Multimedia*.

2.1 Information Management Systems

Among others, information systems include traditional database management systems (DBMS), integrated database systems (e.g, federated systems, multidatabase systems, mediated query systems [29]) and content-based retrieval systems.

2.1.1 Databases and Integrated Database Systems

One of the first explicit use of metadata was in the context of *DBMS*. The data dictionary [30] (also called the system catalog) contains descriptions of the database structure, constraints, physical storage information (like storage details of each file, keys and indexes), access rights, and so on. It may additionally include application documentation about users, the process of database design, and about which applications use which parts of the database. Data dictionaries are usually modeled and stored using the same data model as the DBMS they belong to. Consequently, the DBMS software itself can be used for their management.

With the advent of general information systems providing the integration of heterogeneous data sources (e.g., databases, flat files, web sites), metadata have expanded in scope. It covers the metadata of the original autonomous sources (e.g., the corresponding data dictionary entries), of the target system (e.g, the global schema), as well as the metadata associated with the processes of schema and data integration. That means, metadata includes

1. descriptive information about data content and data structure, and
2. information regarding the processing of data.

Besides the information previously mentioned for data dictionaries, descriptive metadata contains information about the types and sources in the integrated system, navigational metadata on how to access a source, and how the data is formatted.

The specification of metadata regarding data processing embodies two aspects. The first concerns information that keeps track of the processes integrating the data and the schema, e.g., mapping rules between local schemas and local access operations into (global) schemas and access operations of another data model. These mappings are necessary, for example, in federated database systems or when database systems are migrated.

Second, metadata includes information regarding existing application programs that access and use data in the information system. In particular, documentation for the end-user about how to use application programs, which predefined queries are available, and technical information about these aspects need to be stored and managed.

Often, metadata are spread across various operational data sources and are individually accessed by different tools or applications. Some of today's commercial products provide interoperable repositories where the shareability of metadata is ensured by a common interchange format (see Section 3). However, the ideal solution is to use a logically centralized repository containing all available metadata of the system. In this way, consistency and maintainability of metadata are facilitated. The repository is shared by all tools for administration and exploitation of the information system. Since the repository provides a documented and user-extendable metamodel (possibly stored itself in the repository), extension of the system for additional information types and sources is automatically supported. For interacting with the repository, administration and exploitation tools require a comprehensive set of operations that constitutes the application programming interface. Also a user interface is needed for querying and (possibly) updating metadata of interest.

The ideas presented in this section will be refined for data warehouse systems (Part II), which are a particular way to integrate heterogeneous data sources. Thus, some of the issues like metadata types and examples disscused in Part II, are valid for this section as well.

2.1.2 Information Retrieval Systems

An information retrieval system organizes and structures data in order to support vague search criteria. The result is a possibly ranked list of information items. In these systems, metadata is handled in a rather primitive way. It originates from full-text indexing techniques and resource descriptions, and is stored in simple files which are read when queries are submitted to the system. For example, in [43] and [117], metadata for a document is the corresponding document vector that is automatically generated by conventional indexing methods and consists of the indexing features and their weights. This metadata is stored in so-called inverted files which establish the basis for the search strategy, and is then used to estimate the relevance of the documents appearing in the output of a query.

Assumptions about user goals and preferences are another kind of metadata that may be used, if available, to further refine the results of queries. In addition, vocabularies, thesauri and ontologies[1] are used as metadata for the extraction of information from text documents.

2.1.3 The Web

Finding information on *the Web* is strongly related to information retrieval. The well-known search engines simply rely on searching indices of words and phrases established through parsing of web documents. However, additional metadata may be extracted and used for querying the web. This metadata provides a structured representation of the data comparable with a database schema and is derived from the link structure determined by the interconnections between web pages. In this context, web site management systems [37] may be built. Problems related to data homogenization and integration are similar to those encountered when building heterogeneous database systems. Thus, a metadata repository containing source descriptions and mediated schema information may exist as well. In this case, the formulation and processing of queries may use the metadata repository.

Recent efforts have concentrated on the enhancement of the WWW with machine-understandable semantics in order to improve access to the huge quantity of information and services on the Web. HTML already provides the HEAD tag and the META tag [2] that can be used to include name/value pairs describing further properties of the document, such as author, expiration date, and a list of keywords. This information may be used by search engines to identify, index, and catalog documents. Additional types of tags may be introduced [75] for denoting the semantics of data stored in HTML pages and tools for utilizing these embedded semantics (i.e., metadata).

Meanwhile, XML is about to become the standard language for metadata on the Web (see 3.12). Using XML, the specification of metadata enables not only information retrieval, filtering and support for software agents, but also resource cataloging and security management.

2.2 Software Engineering

Within software engineering, the notion of metadata often appears in relation with CASE-tools and includes development and design documentation, application models or schemas, interface definitions, source code or the content of makefiles [15]. In CASE environments, the shareability of metadata is indispensable. Metadata produced by single tools may be needed by other tools in the system. The reason is that often one tool does not meet all requirements, for example, to cover all life cycle stages. Thus, similar to information management systems, metadata has to be stored in repositories which may interoperate if appropriate interchange standards are supported (see Section 3).

However, metadata also takes other forms when developing software. During the last years, for example, object-oriented programming and component-based software development have addressed the aspect of metadata management.

[1] A *vocabulary* is a language dependent set of explained words used in a local context. A *thesaurus* manages synonyms of words and related terms (e.g. sub- and superconcepts). An *ontology* provides explicit and precise descriptions of primitive concepts and relations in a particular domain. Concepts can have many descriptive attributes, may be partially described at any level of granularity, and may be viewed from many perspectives [38, 17].

Inherently, object-oriented technology has increased the need for metainformation. The aim is to keep track of defined classes, methods, instances and interdependencies between them. Metainformation is either gathered at compile time and preserved for use at runtime (e.g., defined classes or methods) or produced at runtime (i.e., additional runtime information similar to debugging information). Metainformation is the basis for browsers and analysis tools. Classes, generic functions, and methods are made accessible as objects which may be inspected. An example is the Java Reflection Core API [65] that supplies an introspection facility for classes and objects in the current Java Virtual Machine.

The concept of *reflection* [45, 108] implies that a piece of software comes with enough information to be self-describing and has access to this information in order to use it at runtime. If a system not only uses this metainformation but also manipulates it and thus has an "open implementation", we speak about *metaprogramming* [72]. Related to this idea, there are the efforts of [6, 5] which support the extension of the system model at runtime. These so-called *dynamic (or active) object models* force developers to make their applications more configurable, flexible and adaptable. This requires to persistently store the object model outside the application (e.g, in a database system) and interpret it. In this way, the object model may be easily changed at runtime (usually by means of special purpose user interfaces) and the changes immediately result in a modified behavior.

During the development of component-based software, metadata is used for the specification and design of self-documented components, for dynamic analysis, and for supporting version management. A typical example is Java Beans [65].

Following the same idea, architectures for distributed computing like Microsoft's (D)COM or OMG's CORBA (Common Object Request Broker Architecture) [99] use metadata to describe all available services and component interfaces in a distributed architecture. Thus, independently developed components may dynamically "discover" each other and collaborate at runtime. The key solution is the availability of an *Interface Repository*. It contains the interface specifications of each object that may be recognized by the system, i.e., the descriptions of all registered component interfaces, the methods they support, and the parameters they require (method signatures). These interfaces are described in a standardized interface definition language, IDL. At runtime, the components use the metadata to discover how to send messages to be understood by other components. Metadata is generated automatically by the IDL-language precompiler. The Interface Repository has an API that allows components to dynamically access, store, and update metadata information.

Furthermore, CORBA provides an *Implementation Repository* which is a runtime repository of information about the classes a server supports, the objects that are instantiated, and their IDs. Also additional information associated with the implementation of the distributed components is provided like, e.g, trace information, audit trails, security, and other administrative data.

2.3 Multimedia

Compared to data management in other areas already presented, the management of multimedia data imposes new requirements. The reasons are multiple [18, 105]: exact-match query processing is often not applicable to digital media, and content-based search is either not possible or performance is poor because queries have to be executed on large sets of raw data. Therefore, the availability and management of metadata is indispensable. The idea is to enhance collections of images, video and audio documents with a partial structure that allows easy access, classification, query and retrieval. In this case, performance of the most costly task for multimedia documents, the namely retrieval, is improved [103] because metadata is of less volume than the data itself.

Metadata consists of general keywords or other established items like "author" or "content". There is also media type specific metadata [105] that represents features like texture of images, frequencies of audio recordings, font size of text or even very specific attributes like speaker characteristics (e.g., female or male) for speech documents and properties like camera motion and lighting for video documents, etc. To summarize, according to [69] and [70], metadata may be classified as content-independent (e.g., location, modification-date of an image) and content-dependent. The latter category may be further divided into direct content-based metadata (e.g., document vectors) and content-descriptive metadata, like textual annotations and keywords describing the content of an image.

In the following, we address two subfields that have particulary focused on metadata during the last years: *geographic and environmental information systems* and *digital libraries.*

2.3.1 Geographical/Spatial Information Systems

Metadata is particularly needed in geographic and environmental information systems to improve the availability and quality of the data. One outstanding aspect of this domain is the extensive national and international cooperation of communities to establish standards for sharing geospatial metadata. The most representative example is the foundation of the U.S. Federal Geographic Data Committee (FGDC) [35] which coordinates the development of the National Spatial Data Infrastructure (NSDI). The NSDI encompasses policies, standards, and procedures for organizations to cooperatively produce and share geographic data throughout all levels of government, the private and non-profit sectors, and the academic community. The work of NSDI has materialized in two metadata standards that have to be adhered by any federal agency maintaining collections of spatial data: the *Spatial Data Transfer Standard* and the *Content Standards for Digital Geospatial Metadata.* The former is a language for communicating spatial data across different platforms. The latter provides a common terminology and definitions for the documentation of geospatial data(e.g., names, definitions and value domains of certain data elements). In particular, metadata provides information about geospatial datasets for the following topics: identification (e.g., title, geographical area covered, currency), data quality (e.g., positional and attribute accuracy, completeness, consistency), spatial data organization (e.g., the method used to represent spatial positions as raster or vector), spatial reference information (e.g, parameters for map projections or grid coordinate systems), etc.

2.3.2 Digital Libraries

By nature, *Digital Libraries* [78] rely on concepts as presented in 2.1.2 and later in 2.4. The field evolved from the need to search, retrieve, access, evaluate, and acquire appropriate information from collections of data available in any digital form ranging from full text to multimedia material like video and audio. Various projects, like the Stanford Digital Library [12], the Alexandria Digital Library [70] and MeDoc [13] confirmed the importance of metadata for describing digital media in a digital library. From the perspective of producers, there are two kinds of metadata which complement each other [81]. Human-created metadata specifies the scope, intent and function for each collection in order to locate and use it easier.

Also, classifications are possible if keywords or brief subject descriptions are available. On the other hand, automatically generated metadata (like summaries, indices, etc.) supports the retrieval of information and the evaluation of queries.

According to [12], there are two kinds of metadata:

- Metadata for information objects (e.g., documents) refers to the description of information objects aiming at the support for the major functions of digital libraries (i.e., organize, access, evaluate and use information). There are remarkable efforts to standardize this kind of metadata. Starting with, MARC [80] is a resource cataloging standard for the representation and communication of bibliographic and related information in machine-readable form. However, the drawback is its complexity that requires professional catalogers resulting in high costs per record. The alternative is *Dublin Core* [118] which proposes a simple, minimal set of attributes to describe the essential features required to facilitate the discovery of document-like objects in a networked environment. This metadata set consists of 15 metadata elements, including usual attributes such as "Author", "Title" and "Subject". The existence of various metadata sets for information objects requires an architecture that supports the integration and interoperation of them. The Dublin Core effort produced the so-called *Warwick Framework* which consists of the specification of a broader container architecture for supporting the integration and interoperation of various metadata sets.
- Metadata for information services (e.g., resource discovery service, search services) aims at facilitating communication among disparate services. In this context, the Stanford Digital Library [12] provides an infrastructure that encourages interoperability among heterogeneous, autonomous digital library services.

2.4 Telecommunication

Telecommunication refers to the transmission of all types of data (text, voice and video) from one computer or device to another. Examples include electronic messaging (fax, mail, news) or document delivery utilizing the EDI standard (electronic data interchange). Inherently, functions to transfer documents over networks have attached metadata for identification (e.g., subject, author), format specification (e.g., compressed, pdf, ps) or metadata defining the mode of transfer. Also, access control management requires metadata like user name, password and public key.

During the last years, *electronic commerce* [123, 100] emerged as a complex application domain that combines, among others, technical issues, business, legal, policy, and socio-economic aspects. The technical side encompasses networks (in particular internet), integration and interoperability of data sources, search and resource discovery, security and access control. As a consequence, the metadata for electronic commerce subsumes all the types known from the individual application domains mentioned before, including metadata for information retrieval, for user and data authentication, and dictionaries and thesauri for legal and socio-economic aspects.

3 Standards for Metadata Management

In this section, we describe and classify various standards which are relevant for metadata management. We do this with a narrow focus: only software engineering standards and some more general standards which might have an impact on data warehouse metadata management are considered. The standards we describe in this section are however not directly related to data warehousing - emerging standards in this arena can be found later in Part II.

3.1 Standard Dimensions

Relevant standards can be classified along the following three dimensions:

1. Primary Application Area
2. Modeling Level
3. Scope

The **Primary Application Area** dimension defines the origin of the standard and the main field of applicability. We distinguish between the following categories:

- *Software Engineering* standards,
- *Repository* standards describing the architecture of repositories,
- *Semistructured Data* standards which have their origin in the world-wide web and are used to describe semi-structured data, and
- *General* standards.

For the **Modeling Level** dimension it is widely recognized that (at least) a four-level architecture is required to adequately model complex information systems:

- *Level 0* contains the actual data items (e.g., the customer data),
- *Level 1* contains the information models behind the data (e.g., the schema defining a customer),
- *Level 2* contains the modeling languages (e.g., the available constructs to build models),
- *Level 3* provides the basis for using different modeling languages within a development environment.

A *standard for metadata* describes and standardizes the syntax and semantics of a modeling language; it thus deals with level 2 of the architecture. A *metastandard for metadata* (situated on level 3 of the architecture) describes and standardizes how standards for metadata have to be built (e.g., it defines the vocabulary, rules and guidelines one has to adhere to when defining a standard) - therefore, a standard for metadata can be regarded as an instance of a metastandard for metadata.

The **Scope** dimension differentiates between standards describing the *representation* of metadata and standards defining how to *exchange* metadata, e.g. between different tools.

An overview of relevant standards can be found in Table 1. Table 2 gives a cross-reference between standards and standardization bodies. As the reader might recognize, there is a plethora of standards. As of this writing, there is a certain convergence towards standards like UML (see 3.5), OIM (see 3.6) and XML (see 3.12). Other standards like CDIF (see 3.8) will most likely align and adapt.

3.2 RM-ODP

The RM-ODP standard (Reference Model of Open Distributed Processing) [58] is a general, representational metastandard. Despite the fact that this standard is concerned with the specification of open, distributed systems, we do not classify it as a software engineering standard, because it has a much broader scope. The general goal of ODP standardization is the development of a framework for system specification and the corresponding infrastructure components. The RM-ODP provides the framework and enables *ODP standards* to be developed specifying components that are mutually consistent and can be combined to build infrastructures matching user requirements. RM-ODP defines

- a division of an ODP system specification into *viewpoints*, in order to simplify the description of complex systems,
- a set of general concepts to specify those viewpoints,
- distribution transparencies to hide aspects of ODP systems that arise through their distribution, and
- principles for assessing ODP-conformance of systems.

An ODP-compliant system is specified from five different viewpoints:

1. The *enterprise viewpoint*, which is concerned with the business activities of the specified system. It covers the role of the system in the business, and the human user roles and business policies related to it.
2. the *information viewpoint*, which is concerned with the information that needs to be stored and processed in the system. The information model is extracted from the individual components and provides a consistent common view which can be referenced by the specifications of information sources and sinks, and the information flows between them.
3. the *computational viewpoint*, which is concerned with the description of the system as a set of objects that interact through interfaces.
4. the *engineering viewpoint*, which is concerned with the mechanisms supporting system distribution
5. the *technology viewpoint*, which is concerned with the detail of the components from which the distributed system is constructed. A technology specification defines how a system is structured in terms of hardware and software components.

	Metastandards		**Standards**	
Primary application	**Representation**	**Exchange**	**Representation**	**Exchange**
General	RM-ODP		ISO/IEC 11179	
Software Engineering	MOF		UML,OIM,IDEF	CDIF, MDIS
Repository			IRDS, PCTE	
Semistructured Data	RDF		XML	XMI, XIF

Table 1: Classification of Metadata standards

	ISO	IEC	ITU	OMG	NIST	IEEE	EIA	ECMA	MDC	W3C
RM-ODP	x	x	x							
11179	x	x								
MOF				x						
UML				x						
OIM									x	
IDEF					x	x				
CDIF	x	x					x			
IRDS	x	x								
PCTE	x	x						x		
MDIS									x	
XML	x									x
RDF										x
XMI				x						
XIF									x	

Table 2: Crossreference standardization body vs. standard

The viewpoints should be considered as projections onto certain sets of concerns rather than as layers or design methods. The general concepts make it possible to relate the viewpoint specifications and to assert correspondence between the representations of the system in different viewpoints.

The RM-ODP framework addresses two audiences, system developers and standard writers. An implementable standard conforming to the provision of some ODP requirement specifies one particular solution. There may be different standards for the same ODP requirement, according to different design choices or technological development.

RM-ODP has been adopted by ISO/IEC (ISO/IEC 10746) and ITU[2](ITU-T Recommendations X.901 to X.905).

3.3 ISO/IEC 11179

The ISO/IEC 11179 family of standards is a general, representational standard. This six-part standard about specification and standardization of data elements gives concrete guidance on the formulation and maintenance of discrete data element descriptions and semantic content (metadata) that shall be used to formulate data elements in a consistent, standardized manner [54, 91]. The six parts of the standard are as follows:

- *Framework for the Specification and Standardization of Data Elements* - introduces and discusses fundamental ideas of data elements essential to the understanding of the rest of the standard.

- *Classification of Data Elements* - provides procedures and techniques for associating data element concepts and data elements with classification schemes for object classes, properties and representations. This part develops a set of principles, methods and procedures for specifying what is needed (at a minimum) in a taxonomy/ontology for the description of object class, property, representation, data element concepts, and data elements.

[2]formerly CCITT

- *Basic Attributes of Data Elements* - specifies attributes of data elements. It is limited to a set of basic attributes for data elements, independent of their usage in application systems, databases, data interchange messages, etc.
- *Rules and Guidelines for the Formulation of Data Definitions* - provides guidance on how to develop unambiguous data element definitions.
- *Naming and Identification Principles for Data Elements* - provides guidance for the identification of data elements.
- *Registration of Data Elements* - provides instruction on how a registration applicant may register a data element with a central registration authority and the allocation of unique identifiers for each data element.

ISO/IEC 11179 standardizes metadata attributes to be attached to element definitions. To this end, it describes some important meta-properties and behaviours. However, only conceptual elements are described. It does not specify schemas or APIs that a registry must support, and as a result of this, it provides no software interoperability.

3.4 MOF

The MOF Standard (Meta Object Facility) [95], adopted by OMG (Object Management Group), is a representational Software Engineering metastandard. This object-oriented standard defines a metametamodel with sufficient semantics to describe metamodels in various domains. The main purpose of OMG MOF is to provide a set of CORBA interfaces that can be used to define and manipulate a set of interoperable metamodels. Initial proposals adressed metadata interoperability in the object analysis and design domain (see 3.5). Additional domains will be covered in the future (see Part II, 3.2).

MOF is organized in three packages as follows:

- The *Model Package* contains the main parts of the MOF Model. The abstract *ModelElement* class classifies the elementary, atomic constructs of models. It is specialized into *TypedElement, Feature, Tag, Constraint, Namespace* and *Import* class.
- The *Reflective Package* contains the portion of the MOF Model supporting reflection - the ability of objects to discover and utilize information about their structure and features.
- The *Facility Package* contains model elements that support repositories, tools, etc. which take advantage of the MOF Model.

OMG has issued a series of standards adhering to the four-level architecture introduced in 3.1. The MOF metastandard constitutes level 3 of this architecture, while the UML standard, introduced in the next subsection, deals with level 2.

3.5 UML

The UML (Unified Modeling Language) standard, adopted by OMG, is a representational Software Engineering standard. UML is a language for specifying, visualizing, constructing, and documenting artifacts of software systems, as well as for business modeling and other

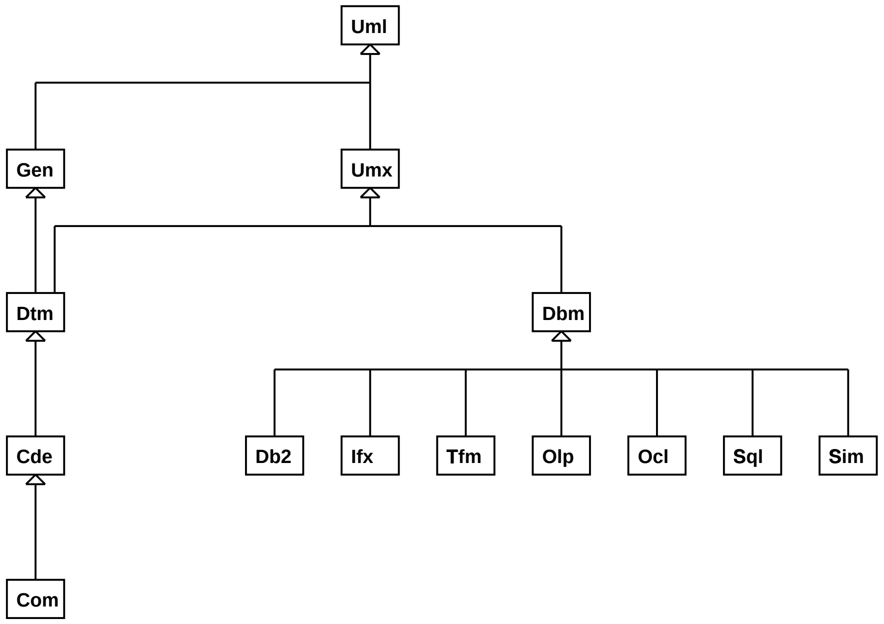

Figure 1: Open Information Model

non-software systems. It is to a large extent compatible with the architecture for system distribution defined in RM-ODP [85] (see 3.2).

The vocabulary of UML encompasses three kinds of building blocks [19]:

1. *Things* - comprising structural things (class, interface etc.), behavioral things (messages, states), groupings (packages), and annotations (notes).
2. *Relationships* - comprising Dependency, Association, Generalization and Realization.
3. *Diagrams* - enabling the graphical presentations of things and their relationships. UML includes nine types of diagrams, namely Class Diagram, Object diagram, Use Case diagram, Sequence diagram, Collaboration diagram, Statechart diagram, Activity diagram, Component diagram and Deployment diagram.

3.6 OIM

The Open Information Model (OIM) is a representational Software Engineering standard [87], developed by Microsoft and passed to the Meta Data Coalition[3] which is a group of vendors and users interested in "the definition, implementation and ongoing evolution of a metadata interchange format and its support mechanisms". OIM is a hierarchy of models to support software engineering. Figure 1 shows the dependencies between the individual models of OIM using the triangular arrow UML notation, pointing from specialization to generalization. In the following, we will shortly describe each individual model; models specifically pertaining to data warehousing are described in Part II, Section 3.3.

[3]http://www.mdcinfo.com

- The top model is **Uml**, realizing a subset of the UML standard (see 3.5). However, the behavioral aspects, namely the packages for collaborations, state machines, and interactions.
- The **Gen** *Generic Model* provides a set of general-purpose interfaces, relevant across diverse information models.
- The **Umx** *Uml Extension Model* provides a set of general extensions to Uml that are common across a number of models, as well as extensions found to be useful to ensure commonality in lower level information models (for example, extended details of attributes in common between the Cde and Dbm models).
- The **Dtm** *Data Type Model* extends the Uml model with a number of interfaces that cover numerous common data types, e.g. Enumeration, Time, Integer.
- The **Cde** *Component Description Model* comprises a set of generic component-related Uml extensions describing run-time (executable) components and their specifications.
- The **Com** *Component Object Model* defines a set of COM-specific extensions to Cde. The interfaces provide additional information specific to COM components, information that is not covered by the generic Cde. The model also includes a set of interfaces extending the Data Type Information Model (Dtm), which allows COM-specific data types to be represented.
- The **Dbm** *Database Model* describes schema information for relational databases, e.g. tables, columns, keys, queries, constraints, indexes, triggers, stored procedures.
- The **Db2** *DB2 Database Information Model* includes schema information about IBM's DB2 databases that is not covered by the more general Dbm model.
- The **Ifx** *Informix Model* comprises a set of Informix-specific extensions to the Dbm model.
- The **Ocl** *Oracle Model* comprises a set of Oracle-specific extensions to the Dbm model.
- The **Sql** *SQL Server Model* comprises a set of Microsoft SQL Server-specific extensions to Dbm.

3.7 IDEF

The IDEF (Integration Definition) pair of standards, adopted 1993 by NIST (National Institute of Standards and Technology), is a representational Software Engineering standard. IDEF consists of two parts, namely IDEF0 [93] and IDEF1X [94].

The IDEF0 standard is based on SADT (Structured Analysis and Design Technique), developed by Douglas T. Ross and SofTech, Inc. The result of applying IDEF0 to a system is a model that consists of a hierarchical series of diagrams, text, and glossary cross-referenced to each other. The two primary modeling components are functions (represented in diagrams by boxes) and the data and objects that interrelate those functions (represented by arrows).

The purpose of the IDEF1X standard is to model data in a standard, consistent and predictable manner in order to manage it as a resource. The basic constructs of an IDEF1X model are:

- *entities* representing sets of real or abstract things which have common attributes or characteristics, represented by boxes;
- *relationships* between those things, represented by lines connecting the boxes;
- *characteristics* of those things represented by attribute names within the box.

Several extensions to the IDEF standards have been proposed [74], but none of those have been adopted by a standards body. IDEF3 defines a method of capturing process descriptions. IDEF4 deals with object-oriented design and IDEF5 provides a method for describing ontologies.

IDEF1X (as adopted 1993) revealed several serious weaknesses [101]. It does not have constructs for mapping between models (i.e., no language for defining views). The interchange of models among tools from different vendors is neither supported by a standard meta schema nor even a file exchange format. Furthermore, IDEF1X does not allow multiple inheritance and provides no connection to process or organizational modeling. Finally, although IDEF1X defines models, it currently has no facilities for tying those models to database schemas. To overcome these limitations, a major extension to IDEF1X (IDEF1X-97) has been proposed [51]. IDEF0 and IDEF1X-97 have been adopted as IEEE Standards (IEEE 1320-1 and IEEE 1320-2).

3.8 CDIF

The CDIF (Case Data Interchange Format) family of standards, standardized by the EIA (Electronic Industries Association)[4] and adopted by the ISO (ISO/IEC JTC1/SC7/WG11), is an exchange Software Engineering standard. It defines the structure and content of a transfer to exchange metadata between two CASE tools. The CDIF standards cover many of the important modeling techniques used by CASE tools today (e.g. data-flow modeling, data modeling, physical relational data base design, etc.). These so-called "subject areas" are views of the underlying CDIF Integrated Meta-model. CDIF is another example of a four-level architecture (see 3.1), the integrated metamodel being on level 2. This metamodel is separated from the CDIF transfer format (level 2), thus distinguishing between the 'what' and the 'how'. Both models are united by the underlying CDIF metametamodel (level 3).

An actual transfer of metadata between two tools is usually via files, an OMG IDL binding is also available. The CDIF meta-model is extensible, which allows two tools supporting CDIF to exchange information that is not defined in the CDIF Integrated Meta-Model. Specifically, this involves having the exporter warn the importer about extensions to the standard CDIF Integrated Meta-Model before transferring information that uses those extensions. CDIF makes no statement about whether or not the importer needs to be able to retain the contents of information transferred.

3.9 MDIS

MDIS (Meta Data Interchange Specification) [109] is a Software Engineering exchange standard from the Meta Data Coalition (MDC). MDIS provides a metamodel that addresses the main types of data models (relational, hierarchical, network), and a standard import/export mechanism that enables the exchange of these metadata objects between tools. MDC has

[4]http://www.eigroup.org

also announced a convention to represent business rules and plans to integrate MDIS with OIM (see 3.6).

3.10 IRDS

The IRDS (Information Resource Dictionary System) standard, developed by ISO/IEC [56, 57, 59] is a representational repository standard. It addresses the requirements and architecture of a dictionary system. An Information Resource Dictionary (IRD) is defined as a shareable repository for the definition of information resources relevant for an enterprise [1]. This may include information about:

- *data* needed by the enterprise;
- the computerized and possibly non-computerized *processes* which are available for presenting and maintaining such data;
- the available physical *hardware* environment on which such data can be represented;
- the organization of human and physical *resources* which can make use of the information;
- the *human* resources responsible for generating that information.

An IRDS is a system which provides facilities for creating, maintaining and accessing an IRD and its definition.

The IRDS standard proposes a four-layer architecture, similar to the CDIF standard.

3.11 PCTE

The PCTE (Portable Common Tool Environment) standard from ECMA (European Computer Manufacturer's Association) [31, 32, 33, 34] and ISO/IEC [55] is a representational repository standard. PCTE describes the basis for a standard software engineering environment, including a repository and communication between tools.

The functionality of PCTE is divided into different modules, e.g. the core module (consisting of the data types and operations), and various additional modules (e.g. access control, auditing, accounting, monitoring etc.). Therefore, different levels of conformance to the PCTE standard exist, for example "conformance to the PCTE standard with auditing".

The most important aspect of PCTE with regard to the process of constructing and integrating portable tools is the provision of an object base (the repository) and a set of functions to manipulate the various objects in it. The basis for the Object Management System is derived from the Entity Relationship (E/R) data model and defines objects and links as basic items of a PCTE object base. Objects are entities (in the E-R sense) which can be designated, and can optionally have

- *attributes*, which are primitive values representing specific properties of an object.
- *links* representing associations between objects.
- *contents*, which can be viewed as a particular attribute type suited to store data (e.g. source code, binary code etc.).

The object base is governed by a typing mechanism. All entities in the object base are typed. Type rules are defined for objects, links and for attributes.

The PCTE object base may be split according to a number of schema definition sets (SDSs). Each SDS provides a consistent and self-contained view of the data in the object base. A process at any time views the data in the object base through a working schema. A working schema is obtained as a composition of SDSs in an ordered list.

The mapping from CDIF (see 3.8) to PCTE is described in [34].

3.12 XML

The XML (Extensible Markup Language), accepted by the World Wide Web Consortium (W3C) as a recommendation [115], is a representational standard for semistructured data. XML is a subset of SGML (Standard Generalized Markup Language (ISO 8879)). XML is about to become the successor of HTML (Hypertext Markup Language), the current language for defining web pages. Unlike HTML, an XML document does not include presentation information. Visual presentation of XML documents is achieved with technologies such as XSL (Extensible Style Language) or CSS (Cascading Style Sheet). XSL documents are themselves well-formed XML documents.

XML is a metamarkup language allowing to define the tags actually needed in a given context, in contrast to HTML and similar markup languages which define a fixed set of tags describing a fixed number of elements. XML allows metadata markers to be embedded within a web document. The XML syntax uses matching start and end tags, such as <name> and </name>, to mark up information. Therefore, web designers can provide metadata that will help people find information and help information producers and consumers communicate with each other base on a common vocabulary. XML is a low-level syntax for representing structured data. An XML document structure is defined with a DTD (Document Type Definition), which can be included in the web page, or can be stored externally through a link to another page.

Specialized and domain-specific languages can easily be defined on the basis of XML. Various groups already specified "their" language, e.g. MathML [116]. However, in order to understand semantics for data based on XML in a standardized, interoperable manner, an "upper" level is required. This metameta level, namely RDF, is described in the next subsection.

3.13 RDF

RDF (Resource Description Framework), accepted by the W3C as a recommendation [9], is a representational metastandard for semistructured data. RDF establishes a framework for describing resources that makes no assumptions about a particular application domain, nor does it define the semantics of any application domain. RDF actually is an application of the Extensible Markup Language (XML). The basic RDF data model consists of three object types:

1. *Resources*: all things being described by RDF expressions are called resources. Resources can be anything named via an URI (Uniform Resource Identifier). Specific examples of resources are an entire web page, a certain part of a web page (e.g. an XML element within the document source), or an object that is not directly accessible via the web.

2. *Properties*: specific aspects, characteristics, attributes or relations used to describe a resource.

3. *Statements*: a specific resource together with a named property plus a value of that property for that resource is an RDF statement. These three individual parts of a statement are called subject, predicate, and object, respectively.

Meaning in RDF is expressed through reference to a schema which defines the terms that will be used in RDF statements. A schema is the place where definitions and usage restrictions for properties are documented.

3.14 XMI

The XMI (XML Metadata Interchange), proposed by IBM, Oracle, Unisys and others as a submission to OMG's SMIF RFP (Stream-based Model Interchange Request For Proposal) [97], is an exchange standard for semistructured data. Although XMI uses MOF (see 3.4) as the metameta model, we classify it as a standard in the semistructured data area, because the exchange of the actual models is based on XML (see 3.12). The proposal consists of:

- a set of XML Document Type Definition (DTD) production rules for transforming MOF based metamodels into XML DTDs;
- a set of XML Document production rules for encoding/decoding MOF based metadata;
- design principles for XMI based DTDs and XML streams;
- concrete DTDs for UML (see 3.5) and MOF (see 3.4).

Because XMI is based on XML, it can be used for metadata interchange with and between non-CORBA based metadata repositories and tools.

3.15 XIF

XIF (XML Interchange Facility), Microsoft's specification for metadata interchange [88], is an exchange standard for semistructured data. Like XMI (see 3.14), XIF is based on XML and used to exchange OIM-based models (see 3.6). Vendors planning to support XIF are NCR, PLATINUM, Siemens Nixdorf, Viasoft, Sybase, Unisys. Microsoft has handed over the XIF and OIM specifications to the Meta Data Coalition.

3.16 Standards for Business Models

Leaving the mostly content independent standardizing efforts for metadata, certain initiatives defining standard models for capturing and formalizing business aspects deserve attention, too. The standards mentioned before, in particular UML, are usually used for representing and visualizing those models.

Complementing OIM in this direction, MDC recently published drafts of model descriptions for Business Engineering [110] and Knowledge Management [111]. The former comprises structures which allow the specification of business goals, organizational elements, business processes and business rules. The latter aims at formulating controlled vocabularies and employs the notions of thesauri, glossaries and indexes. The submodel for business rules relies partly on results from the GUIDE Business Rules project[5] initiated in November 1993 to formalize an approach for identifying and articulating the rules which define the structure of an enterprise and control its operations. The GUIDE organization (Guidance for Users of Integrated Data Processing Equipment) is an international association with users of IBM or open architectures as members. The main purposes of GUIDE in connection with the information systems and technology industry are:

- to communicate user needs in all technical areas of interest to suppliers of information technology equipment, services, and software,
- to review, comment and exchange information on products and services,
- to influence the development and use of computer industry standards.

A business rule is a statement that defines or constrains some aspect of the business. The GUIDE Business Rule Project defined a conceptual model of business rules in order to express (in terms meaningful to information technology professionals) just what a business rule is and how it applies to information systems. Three types of business rules have been identified:

1. *structural assertions*: terms (being words or phrases with a specific meaning for the business) or statements of facts expressing certain aspects of the structure of an enterprise;
2. *action assertions*: statements of constraints or conditions that limit or control the actions of the enterprise;
3. *derivations*: statements of knowledge which is derived from other knowledge in the business.

A business rule is atomic in that it cannot be broken down or decomposed further into more detailed business rules.

Related models for representing business rules, goals and activities stem from the area of business process modelling and usually form the basis of corresponding modelling tools. One example is the ARIS House [104] from IDS embedded in the ARIS House of Business Engineering framework which is one of the earlier proposals for providing an overall model of business applications. Originally intended even for IT related aspects and as such a competitor to various CASE-tool models extended by business model packages, the current ARIS metamodel is restricted to business specific parts. The close collaboration of IDS with SAP gives this model a noticeable position in the market.

3.17 The Zachman Framework

A very global view on the architecture of information systems was proposed by John A. Zachman [121]. The units of his framework can also be understood as organization scheme for all kinds of metadata involved in building and using an information system and have therefore become widely recognized during the last years.

The Zachman framework is organized as a matrix which describes the various ways the stakeholders of an organization view the business and its systems. The rows of the matrix represent the five different basic roles that people play in the creation of a product:

[5] http://www.guide.org/ap/apbrules.htm

1. *The planner* is concerned with positioning the product in the context of its environment, including specifying its scope.
2. *The owner* is interested in the business deliverable and how it will be used.
3. *The designer* works with the specifications for the product to ensure that it will, in fact, fulfill the owner's expectations.
4. *The builder* manages the process of assembling and fabricating the components in production of the product.
5. *The subcontractor* fabricates out-of-context (and hence reuasable) components which meet the builder's specification.

The columns of the matrix represent the six important dimensions of the architecture of an information system, namely data, function, network, people, time and motivation. The Zachman framework supports commonly adopted techniques like entity relationship diagrams, data flow diagrams and the like - each of these techniques is placed in context with the others to provide a complete picture of the information system to be modeled.

4 Commercial Support for Metadata Management

According to a recent Gartner Group survey [41], there are five vendors selling repository products which are classified as "leaders" in this market segment, a sixth vendor is classified as a "visionary" (Unisys). The five leaders are Platinum, Microsoft, Viasoft, IBM and Oracle. We will present the "stand-alone" repository solutions from Viasoft, Unisys and the offerings from Platinum/Microsoft, who recently joined forces in the repository area. We will skip the products of IBM and Oracle, because their repositories are bundled with software engineering products: in the case of IBM with TeamConnection, their software configuration management tool; in the case of Oracle with Designer (formerly Designer/2000), their analysis & design platform.

4.1 UREP

The Universal Repository (UREP) from Unisys corporation[6] is an object-oriented, extensible repository. Unisys, one of the co-submitters of the MOF proposal, is actively working to follow both the MOF and UML standard in their product. To exchange metadata with other repositories and tools, CDIF is supported. In addition, Unisys has demonstrated metadata exchange (with the Microsoft repository) via XMI and XIF.

The repository services provided by UREP are based on the service sets defined by the PCTE standard (see 3.11). These services include:

- *version control*: maintaining a history of information;
- *transactions*: enabling users to define and enact transactions (short and long transactions);
- *user and session management*: identifying a repository user, defining access rights etc.;

[6]http://www.marketplace.unisys.com/urep/

- *metadata service*: extending the information model.

Two types of models exist within UREP: The *Technology model* describes repository types and operations that represent generic software technologies (database paradigms, business analysis, etc.). The repository types and operations that a particular tool uses, belong to the *Tool model.* Both types of models are extensions of the UREP information model. The metadata service provides mechanisms for:

- defining object modeling constructs such as types, relationships, integrity rules etc.;
- extending the UREP information model;
- reusing object types and operations that are defined in the repository, and
- extending repository functionality by defining subtypes and by overriding inherited operations.

UREP has been licensed by different companies to be included in their product palette, e.g. by Sybase corporation and BEA systems.

4.2 Microsoft Repository

The Microsoft repository[7] is an object-oriented, extensible repository, consisting of two major components:

- a set of COM-based APIs to describe information models and
- a (relational) repository engine (either Microsoft SQL Server or Microsoft Jet, the database system in Microsoft Access) which serves as storage component for these models.

The Microsoft repository is targeted towards software vendors and users wishing to support the management of metadata in a variety of scenarios including software development and data warehousing, using OIM as the underlying model.

Microsoft Repository 2.0 ships with Visual Studio 6.0 and SQL Server 7.0. Microsoft has a collaboration agreement with Platinum (see 4.3) to port the repository to non-Microsoft platforms. According to Microsoft, there are more than 75 third parties shipping tools that use the repository for their metadata management.

4.3 Platinum Repository

Platinum Technology Inc.'s repository[8] currently exists in two versions:

- *Repository/MVS* is a mainframe-based repository, first released in 1988, using DB/2 as its storage platform.
- *Repository/OEE* (Open Enterprise Edition) is a client/server solution which stores its data in Oracle, Sybase or Microsoft SQL/Server.

[7]http://msdn.microsoft.com/repository/
[8]http://www.platinum.com

Both versions provide models based on the E/R approach. The Platinum repository provides full bi-directional interfaces to many CASE tools, including ERwin, Bachman, Sterling/Cayenne and Oracle Designer. Scanners (a specialized set of tools for parsing and importng file definitions, SQL statements etc.) are available for all popular languages.

Platinum and Microsoft have a collaboration agreement, giving Platinum the right to port the Microsoft repository (see 3.6) to non-Microsoft platforms and to leading RDBMSs (Oracle, DB2, Sybase). According to Platinum, the next release of Repository/OEE, called the PLATINUM Enterprise Repository V2.0, will contain all the functionality currently available in Repository/OEE and will be based on the Microsoft-PLATINUM Repository V2.0 engine.

4.4 Rochade

Viasoft's Rochade repository[9] is, together with Platinum's products (see 4.3), the market leader for centralized, "stand-alone" metadata management environments. Rochade uses an object/network-based database structure which handles objects and links between them. The following four basic building blocks are maintained:

- *item types* describing groups into which similar items are categorized,
- *attribute types* describing aspects of an item,
- *link types* describing relationships between items, and
- *rule types* describing processing associated with an item/link/attribute or another rule.

The heart of the Rochade repository is the repository information model (RIM). A RIM provides a scoping mechanism for accessing the repository and defines the details necessary for supporting Rochade subject areas, projects and external tools. The RIM of Rochade provides unlimited extensibility. Its data architecture recognizes five data levels, each level describing and defining the level beneath it.

Level 0 is the bottom layer, containing the actual business data managed outside the repository.

Level 1 the "meta-information" level, describing the application systems and the information architecture of an organization. Rochade identifies information at this level as items, and each item has attributes and links to other items.

Level 2 the schema level; defines (within a series of models) what can be managed at the meta-information level.

Level 3 the metaschema level; defines the building blocks for repository models, including the RIM. Structures like meta-entity-type or meta-relationship-type are not predefined in Rochade.

Level 4 the metanetwork level; allows the metaschema building blocks to be defined. According to Viasoft, this fourth level is the most critical model within a repository because it provides the power to define concise, semantically rich models at the schema level. The metanetwork level consists of only two structures: meta-node type and meta-rule type.

[9] http://www.viasoft.com

> Meta-node types have been used to define item types and attribute types. Meta-rule types have been used to distinguish the behavior of item types that represent entities from those that represent relationships.

It is interesting to note that the Viasoft architecture is not congruent with repository standards like PCTE (see 3.11) or IRDS (see 3.10) which both stipulate a four-level architecture.

As the competing Platinum repository product, Viasoft's Rochade offers rich interface support to other tools, e.g. Bachman, System Architect, ERwin.

5 Research Projects on Metadata Management

Metadata are relevant in many research areas as already mentioned in Section 1. In this section we concentrate on metadata *management* issues, in particular on projects providing systems for that. We shortly look at three research prototypes developed in the database community which are mainly intended to serve as a general purpose store for metadata. Then we turn to a strongly related topic, namely the management of ontologies, and finally discuss the role of metadata for information integration tasks and website management.

5.1 ConceptBase

The ConceptBase[10] deductive object base system [61] developed at Aachen University of Technology is a client-server database system for conceptual information, i.e. metadata. The representation language for this purpose is Telos [89], originally a knowledge representation language for encoding requirement models of information systems. The foundation of Telos is a simple data structure which uniformly represents objects, classes, meta classes, attributes, and class-instance as well as subclass-superclass relationships.

Telos also provides for a logical sublanguage based on many-sorted first order logic. The main predicates available allow to express the basic object-oriented structuring principles plus certain arithmetic and aggregation operators. Formulas in this language are employed for specifying deduction rules and integrity constraints and constitute the main building block of the query and view language of the systems. Queries and views are described as specific derived class definitions (with a nested frame syntax) and state necessary and sufficient membership conditions for objects explicitly stored in the extension of some ordinary class.

Since the class-instance relationship is stored explicitly in ConceptBase, an object is allowed to have multiple classes. Moreover, a class may be an instance of other classes, called meta classes. Even meta classes may be grouped into classes called meta meta classes, and so on. There is virtually no limit in this hierarchy though most applications do not go beyond 4 levels (instances, classes, meta classes, meta meta classes). The lowest level can be associated with data, the class level corresponds to the schema level (of databases), the meta class level encodes modeling languages (e.g. the entity-relationship language), and the meta meta class level can be used to link multiple modeling languages. Since attributes are understood as links between objects and in particular are objects themselves, they are also involved in the multi-level hierarchy: the attribute of a class works as attribute category for the attributes of its instances. Integrity constraints, deductive rules and queries can be formulated at any of these levels. Predicates describing attributes on one level e.g., are based on the attribute categories of the next upper level.

[10]http://www-i5.informatik.rwth-aachen.de/CBdoc/

The view language of ConceptBase forms the basis for view-specific C++ API's [106] which allow automatic initialization and incremental maintenance of view data derived from the object base contents and managed in client applications. Basic API's are available for several other languages. The ConceptBase server is implemented as a main-memory database with strict serialization of transactions.

In a number of different application settings, ConceptBase was able to support various data representation frameworks like the IRDS standard, but also showed the necessary flexibility for dedicated and application-specific modelling schemas. [66] gives a comprehensive overview on applications and experiences with the system. We can distinguish between three different categories:

1. *model integration:* Such application projects require a customized collection of inter-related modeling languages.
2. *model analysis:* Such applications handle large models whose content has to be analyzed to find hidden properties or faults.
3. *distributed co-operation:* Here, a group of human experts has the task to process highly inter-related information.

In category 1, software and requirements engineering projects like DAIDA [60] and NATURE [63] used the system both for design and runtime support. From the domain and application model down to implementation descriptions of an information system to be built, all levels can be described by Telos objects, even refinements on the same level and dependencies between the different levels. In the data warehouse context, ConceptBase was used in the DWQ project (see Part II, Section 5.3). A typical example for category 2 are business process modelling projects: the method PFR e.g., used in commercial applications, was complemented by a Telos schema and a stepwise refinement methodology based on query classes [92]. An example of category 3 is a terminology server system KONTAKT [114] used for establishing medical terminologies.

5.2 H-PCTE

H-PCTE[11] is a PCTE implementation maintained at the University of Siegen. An H-PCTE object base contains typed objects and typed relationships. Both can be further specified by attaching attributes. The object types organized in a multiple-inheritance hierarchy may be combined into external schemas constituting a simple view mechanism. Versioning of objects is supported explicitly and even schema level operations, i.e. the dynamic creation and change of types, are allowed and complemented by appropriate object modifications.

While the PCTE standard is restricted to an API for navigational access to objects only, the H-PCTE system offers both a set-oriented algebraic [48] as well as an SQL-like query language named P-OQL. The latter is an extension of the OMG standard OQL with regard to the PCTE object model.

From a database point of view, H-PCTE offers various levels of transaction isolation including nested transactions and fine-grained locking. H-PCTE is, like ConceptBase, a main-memory based system. It is both possible to reserve fragments of an object base for

[11]http://pi.informatik.uni-siegen.de/pi/hpcte/

exclusive access and uploading in the address space of a single application, or to work on a shared adress space with several applications.

Java clients can access H-PCTE via the Java-API which employs an additional server process for bridging application and repository and in particular for providing a notification mechanism: observer tags are attached to objects inside the repository which are of interest to an application; changes are then steadily reported back. In addition, the H-PCTE system comes with a palette of generic administration and browsing tools.

H-PCTE was mainly applied as a repository component for software engineering environments, such as the ECMA based PIROL system [46] and the UniForM WorkBench [68]. Software objects stored in such environments range from specifications and proofs up to test descriptions. Relationships between objects represent development links (e.g. between different versions produced in various phases of the development process), dependency links (e.g. imports) and additional references. The schema mechanism of H-PCTE can be used to implement workspaces that coexist and are subject to integration.

Futhermore, H-PCTE served to support real-life applications such as telephone directories on the Web. The notification mechanism enables applications for distributed document editing in general [71]. Changes in one client editor are directly made available to another client by immediate propagation of each update step.

5.3 Lore

The Lore system[12] being developed at Stanford University is a so-called light-weight repository in the sense of a very simple basic data model and at least in the beginning of the project with read-only and single-user functionality [84]. Although the main application of Lore is the management of semistructured data its origins lie in the TSIMMIS information integration project. The first data model Lore supported was OEM (Object exchange model) which basically consists of nodes and labeled links between them. This model serves in TSIMMIS as a mediating language used by a net of mediators each of which providing access to data sources of various kinds. The TSIMMIS project proposed specification rules relating the source contents resp. schema to OEM, and enabled query translation for OEM queries to queries executable on the sources. In this context, Lore was used as a repository for OEM descriptions.

The increasing importance of XML led to a replacement of OEM in Lore. The main difference between both languages in fact can be reduced to a missing analogue for DTD's in OEM which however was consistent with the original idea of schemaless data. As soon as a DTD exists, it imposes schema restriction on the respective data. The flexibility of XML is obvious as well as its applicability for managing all kinds of metadata which themselves are at least in part semistructured by nature.

The query language Lorel [10] is an OQL adaption and emphasizes the importance of path expressions derived from the link labels between objects. All other objects reachable from one object by traversing paths with the same label sequence are considered as solutions satisfying such expressions.

The database functionality of Lore was implemented from scratch including a cost-based query optimizer exploiting a dedicated index mechanism. An important feature of Lore is the DataGuide as a summary of the database contents. Querying schemaless data requires some

[12]http://WWW-DB.Stanford.EDU/lore/

idea on the structure of data. Furthermore, database statistics, warnings, index mechanisms etc. have to be anchored somewhere. A Lore DataGuide [44] provides therefore a schema reconstruction or condensation (e.g. by collecting the set of different label names among the links starting from an object). The DataGuide is also realized as a network of objects and can be maintained incrementally.

Lore as a repository for XML data is currently extended by another system called Ozone which aims at enabling coexistence with ODMG conform object bases.

5.4 Ontology Management

Ontology management [98, 38] is a classic field of Artificial Intelligence where structural information (including concepts, their definitions, relationships and properties) is collected and maintained - often in domain-specific contexts - in order to support various applications like natural language processing or machine learning. With the extended definition of metadata in Section 1, ontologies can be understood as metadata, too.

Most ontology management systems employ frame-based languages for describing concepts and additional powerful assertion languages with first-order expressiveness plus constructs to specify modals, contexts and meta statements.

Probably the most famous and also most general ontology management initiative is **CYC**[13]. Started at MCC Corp. in the mid 80's and shifted to a startup called Cycorp after 10 years of development, CYC is a huge global collection of commonsense knowledge. The main idea is to encode information which is not available from other sources but nevertheless contributes to a universal schema of the world for such general applications like the Web. One product of Cycorp actually is the existing ontology itself; in part it is available on the Web. The Cyc Knowledge Server system [77] implements the representation language CycL. Term definitions and assertions in this language can be stored and accessed via an API and various interactive tools and browsers. In addition, Cycorp offers a natural language interface and a so called semantic integration bus for connecting the Knowledge Server to external sources such as databases and web pages. The restricted database facilities of Cyc recently have led to approaches where selected portions of Cyc knowledge bases (so called microtheories) were used to generate database supported information systems including application specific API's.

The Knowledge Systems Lab at Stanford University provides the **Ontolingua Server**[14] which is a wide spread tool for describing ontologies. The Ontolingua activities are part of the DARPA-sponsored program on High Performance Knowledge Bases (HPKB) and aim at the support for distributive collaborative construction and effective use of large and reusable ontologies. The Ontolingua data model is a superset of the frame-based Knowledge Interchange Format (KIF) a proposed standard for data representation in knowledge-based systems. Besides browsing and editing facilities, Ontolingua offers advanced facilities for comparing different ontologies (including derivation of mappings between them), analysis regarding inconsistencies and incompleteness.

An additional outcome of the Ontolingua project is the definition of OKBC [23], the ODBC analogue for knowledge base systems, which consists of a number of general operations for manipulating and accessing knowledge bases. Among various other application areas,

[13] http://www.cyc.com

[14] http://ontolingua.stanford.edu

Ontolingua is currently employed to construct a World Fact Book, containing a substantial collection on all aspects of knowledge about the world's nations.

5.5 Information Integration

While information integration actually is an old topic, it has received growing interest with the explosion of Internet and Web technologies during the last years. Formerly, the attention concentrated on, e.g., integrating *databases* in distributed environments through global schemas and transparent query access. This schema integration task is even more relevant if the data sources to be integrated have heterogenous formats or are only weakly structured. A number of research initiatives were launched, in part supported by the DARPA I^3 framework, among others the **TSIMMIS** [40] and **Garlic** projects [113, 47] at Stanford and IBM Almaden. Most of them propose a *virtual* integration approach realized by source specific wrappers solving the format mismatch, and mediator components responsible for central integration and query services.

As mentioned above, the metadata necessary for integration in TSIMMIS, the schema descriptions and additional information which enables the generation of mediators, is supported by the Lore repository system. The Garlic kernel is an independent middleware component with a system catalog for recording schema information and statistics about registered sources.

The **Information Manifold** [79] developed at AT&T Labs is another example of a uniform interface to several hundred information sources. A special focus in this project was the idea of a personalized (virtual) information repository with user profiles as segments of an overall ontology used to access the information sources. The description logic system CLASSIC [20] was used to maintain the schema information and profiles and also for certain reasoning tasks, such as subsumption checking.

The Telos language used in ConceptBase is also basis of an environment implemented at Queens University Kingston. In order to query passive sources like web pages or software source files, it is assumed that a preceding analysis performed by application-specific extraction tools obtains metadata from the sources and makes them available for querying a repository system [82].

5.6 Website Management

A special case of information integration deals with managing websites. One approach to control the often chaotic structure of websites and to facilitate clearly defined update and access strategies is the model-based generation of pages, either dynamically on demand or even statically. In most cases, the generation is embedded in a general architecture which also considers existing data sources to be included through wrappers and a mediation layer for virtual integration. Existing proposals, e.g., as developed in the ARANEUS [11] and STRUDEL [36] projects, usually differ with regard to the formal data model (and its degree of structuring) selected for representing the source schemas as well as the target schema of the website. These descriptions in fact are metadata, which comprise the skeletons of the final web pages, its relationship to the data sources, and transformations between them. The ARANEUS approach uses relational views as abstraction and emphasizes the advantage of adopting relational schema integration methodologies while STRUDEL employs a semistructured data model. Systems like Lore and ConceptBase are obviously well suited to serve as repositories even in such environments. A recent project on generating Hyperbook applica-

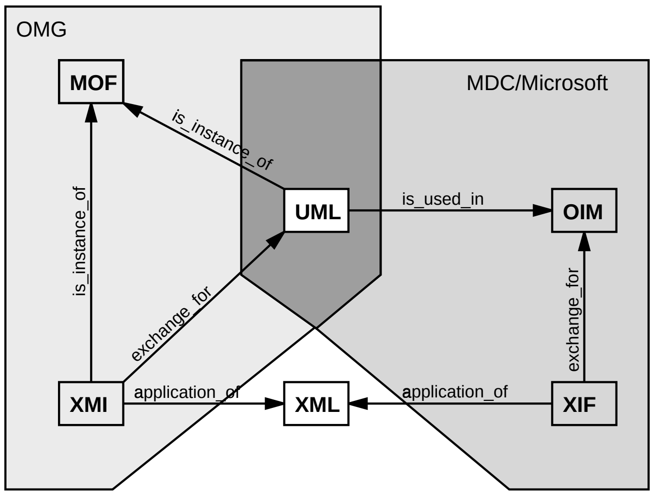

Figure 2: Standardization results

tions in the Web at the University of Hannover [90] proves this by storing book structures, page structures, possible access paths etc. in ConceptBase.

6 Conclusions

In the preceding sections we have discussed application areas, standards, tools and research activities dealing with metadata. The relevance of metadata and the broad applicability of metadata management tools should be obvious.

Meta data have in general a more heterogenous structure than pure data of certain applications: They describe data and system aspects on different levels of abstraction and formalization and for different types of users. This yields specific requirements for the flexibility of both, representation and exchange languages. Integration at all levels can be done by introducing meta meta models or even higher model abstractions.

The effect of standardization efforts for meta data management is still unclear. We can observe two different competing streams of standardization (cf. Figure 2) with regard to metadata representation and exchange, namely Microsoft strongly influencing the metadata coalition by OIM, and its rivals Oracle, IBM etc. who contribute to the OMG activities around CORBA.

One common intersection is UML as a general purpose modelling language suited for metadata and XML as a basis of exchange standards. These are also the first general formalisms (apart from HTML and SQL) adopted by Microsoft. Furthermore, XML might be suited as a general (low-level) metadata representation language, using, e.g. specific DTD's for handling UML models. The shared meta meta layer given by MOF for the OMG aligned standards is missing, the components of OIM are only in part open and in particular platform dependent.

On the tool-side, the Microsoft repository is of increasing importance due to the dominating role of Microsoft in software development and standard application tools. Whether the cooperation with Platinum will make their concepts including OIM available for the Unix and mainframe world, will turn out in the future. First commercial database solutions for XML, e.g., from Software AG (Tamino[15]), Oracle (Oracle 8i[16]) and Object Design (eXcelon[17]), have become available already and can be considered as alternatives to the classic repositories.

The research tools for metadata management shortly introduced in Section 5 feature specific benefits, which are even relevant for their commercial counterparts. For example, ConceptBase with its powerful logic-based language used for rules, integrity constraints and queries yields excellent analysis and inference facilities for meta data. Its data model has a clean formal semantics and is still the most flexible one concerning modelling layers and the definition of modelling constructs. Although PCTE obviously did not have the intended standarization effect on repository systems, the Java-based notification mechanism implemented in H-PCTE allows comfortable support for collaborative metadata editing and browsing tools, and even is important for handling federated metadata. Lore is one of the first XML-oriented repositories with its main intention to solve integration problems on semistructured data. In this context, also classical database problems like query optimization and view maintenance based on a declarative query language were tackled. Lore's Data Guide, as a kind of schema extraction facility, contributes to enabling the coexistence of structured and non-structured data in one common container.

Other research activities mentioned deal with managing special kinds of metadata, not only structural information but also very complex behavioral specifications that are used for reasoning, mapping to executable formats, etc.

Two important problems with metadata management concern their maintenance and integration. Metadata (like data) lose their worth if they do not reflect the actual state of the system and enterprise world. The maintenance of metadata is only in part a question of interoperability between software systems, in particular for generated metadata from, e.g., systems or process design. On the other hand the maintenance problem requires organizational concepts (responsibility, authorization, etc). This holds for manual documentation and also non-IT metadata. The idea of one global metadata base managed by a central repository - sometimes called Organization Memory (OM) - becomes unrealistic at the latest when one crosses enterprise or organization borders, often even earlier. Proprietary metadata mangement solutions will continue to exist. Therefore, federated and distributed metadata management naturally yields *meta*data integration and coordination tasks. Concepts of meta meta models and meta standards, which sometimes might appear to be artificial, present themselves in a different light since they consider meta meta data for supporting these tasks. However, the very general understanding of metadata (actually as data but semantically in certain contexts as data on (the handling of) data) makes this difference obsolete as long as representation and exchange languages are generic enough (like XML) and as long as the inter-level relationships between data, metadata, meta metadata etc. are not needed explicitly.

[15]http://www.softwareag.com/tamino/
[16]http://www.oracle.com/database/oracle8i/
[17]http://www.odi.com/excelon/main.htm

Part II
Metadata for Data Warehousing

1 Data Warehousing

Starting with the pioneering work of Inmon [52], the popularity of data warehousing grew exponentially during the last years. Competitive organizations are just on the way to build data warehouses or to extend, reengineer, and improve already existing one(s). The significance of the topic is reflected in the huge amount of material available today: literature [16, 25, 27, 52, 73, 83, 86], web sites[18], and an abundance of software products for building and exploiting data warehouses.

Data warehousing is a collection of concepts and tools which aims at providing and managing a set of integrated data (the data warehouse) for business decision support within an organization. In this way, important business trends may be discovered and explored and better and faster decisions may be achieved regarding multiple aspects of the business like sales and customer service, marketing, and risk assessment.

The tasks of data warehousing comprise the processes of designing, building, using, and maintaining a data warehouse. We distinguish four phases. The data warehouse system is first designed at *designtime*, then the warehouse is populated at *buildtime*, and finally it is employed at *runtime*. *Maintenance* partly reiterates the first two phases.

After a brief explanatory introduction in Subsection 1.1, we consider these phases.

1.1 Data Warehouses versus Operational Systems

A data warehouse typically incorporates data collected from a variety of heterogeneous data sources (database systems, flat files, indexed files, so-called legacy systems[19], Web pages, etc.) including current operational enterprise-internal systems as well as external data sources. Data has to be extracted, cleansed, transformed and stored in an integrated form in the data warehouse.

Compared to traditional operational systems which focus on transaction processing (including consistency and recoverability), data warehouse systems aim to fulfill certain functional and performance requirements regarding efficient information extraction from the data. In particular, data warehouses are designed for ad-hoc, complex queries with an optimized response time. This may not be achieved in operational systems where database schemas are normalized and a multitude of table joins would be required for complex queries inherently yielding weak performance.

Regarding the timeliness of the data, an operational system provides current data values while a warehouse supplies a view of the business over a period of time by means of historical data, necessary for business trend analysis. Note that the data in the warehouse is typically updated only at certain points in time and thus it does not necessarily contain the most

[18]http://www.datawarehousing.com/, http://www.datawarehouse.com/, http://www.datawarehousing.org/, http://www.dw-institute.com/, http://www.dmreview.com/, http://pwp.starnetinc.com/larryg/

[19]The term *legacy systems* is used for all those 'old-fashioned' applications in enterprises which store and manage data mostly in proprietary formats and which have grown over time often in a chaotic manner with dubious effects on data quality and interoperability.

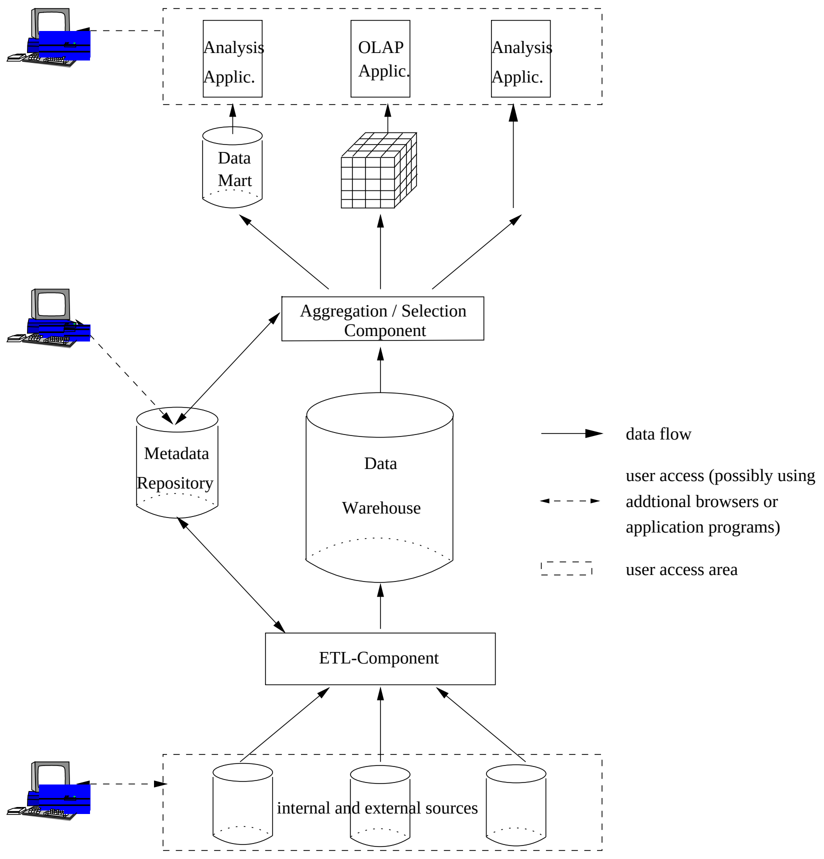
Analysis Applic.
OLAP Applic.
Analysis Applic.
Data Mart
Aggregation / Selection Component
Metadata Repository
Data Warehouse
data flow
user access (possibly using addtional browsers or application programs)
user access area
ETL-Component
internal and external sources

Figure 3: A data warehouse system at buildtime and runtime

recent data values. However, this s not a problem with regard to the periodical or long-term analysis tasks for which a data warehouse is mainly built.

More detailed comparisons between operational systems and data warehouses may be found in [16, 24, 120].

1.2 Building the Data Warehouse

Figure 3 depicts a typical architecture of data warehouse systems with the interactions that happen between the components at buildtime and runtime.

The warehouse is populated with the data extracted from the sources during two individual processes: the initial *loading* phase performs data extraction, cleaning, transformation, and storing into the target source (the data warehouse), and the regular *refreshment* that propagates (through the same activities, extraction, cleaning, etc.) changes into the warehouse. We consider the main components that participate in the loading and refreshment process: operational sources, ETL-component, data warehouse, data marts, aggregation/selection component, and metadata repository.

Data sources may be legacy systems, database systems, flat files, indexed files, WWW, etc. They may contain structured, unstructured or semi-structured data and follow different data models. Traditional operational database systems support so-called online transaction processing (OLTP) applications. Operational data sources are used in daily business, independently of the data warehouse system.

As the name suggests, the **E**xtraction, **T**ransformation and **L**oading **(ETL) component** extracts data from the sources, transforms it, and stores it in the data warehouse. Data transformation may take various forms: reconciliation of syntactic and semantic differences between operational sources, consolidation, summarization, aggregation, and mapping from local data models to the global one. In particular, for achieving data quality, duplicates are eliminated, data regarding the same entity is compared and false data is rejected, data is converted into a uniform type and representation format (character, number) supported by the target platform, into a common measuring unit specification (currency, length), and so on. Further examples of possible transformations include: the calculation of derived data (like the age infered from the date of birth), enrichment of data (like completion of addresses based on the value of the postal zip code), and so on.

The data warehouse represents the kernel of the data warehouse system. Ideally, it is an enterprise-wide collection of integrated data. Data is usually stored in relational databases that underly special forms of normalized schemas.

The **aggregation and selection component** performs further processing steps. For example, it calculates the views to be stored in the data marts or prepares data to be fed into the analysis applications.

Data marts are data stores that are subordinated to a data warehouse. This means they render information views of the same single integrated pool of data. Data marts are built with the aim to provide specific application requirements of a certain group of users, e.g., of a department or a geographical region.

The metadata repository plays a key role at buildtime (and, as we will see, at runtime and designtime as well). Figure 3 indicates the existence of interactions between the metadata repository, the ETL-component, and users.

The metadata repository is a way to automate loading and refreshing of the data warehouse. This means, the metadata is accessed during loading and refreshing and governs the extraction, transformation and loading of data into the data warehouse. In addition, the repository provides the warehouse administrator with the information necessary to keep track of the execution of loading and refreshing processes. In this way, the database administrators have full knowledge and understanding of the data available and of data processing (where the data comes from, how it was transformed, what it represents, how it can be used).

1.3 Using the Data Warehouse

Data warehouse are built for analysis purposes. The analysis involves examining data and possibly identifying relationships that may exist between different elements. Analysis relies on various interactive tools, data mining applications or simple reports based on queries and browsing. The results have to be presented in a user-friendly, understandable manner. The user has to interpret them in order to be able to take appropriate decisions.

The most popular analysis means are **OLAP[20]-tools** which enable users to examine data within a multidimensional model allowing to instantaneously retrieve and summarize data. The multidimensional model provides measures (i.e, business facts to be analyzed like sales or shipments) and dimensions for these measures (i.e., the context in which the measures have values assigned, e.g, products, customers, time, region). Usually, dimensions

[20] Online Analytic Processing

have associated hierarchies that specify aggregation levels for viewing the data. The data has to be periodically extracted from the data warehouse and, together with the precomputed results of complex operations (e.g., aggregations), has to be stored in a multidimensional model, as provided by multidimensional database systems. Since the data is appropriately organized query performance is significantly increased.

Data mining offers more powerful means for extracting information. It aims at the automatic detection of implicit, previously unknown and potentially useful patterns in data and their translation into valuable information used for strategic decisions. Successful application domains include fraud detection, loan approval, and portfolio trading. Data mining techniques comprise classification, clustering, association discovery, and deviation and change detection techniques [16]. Note that for effective data mining, data quality has to be high (i.e., as few as possible missing, incorrect and stale data).

The metadata repository has again interactions with both the aggregation component and the user in order to better extract and analyze information from the data warehouse.

1.4 Designing the Data Warehouse

The design phase comprises defining the structure of the warehouse and developing software that carries out data extraction, cleaning, schema and data integration, and loading into the warehouse. The process covers also the development of end-user applications for extracting information from the warehouse (e.g., browsing, OLAP, and other analysis applications).

Regarding the design of the data warehouse schema, note that for optimized query processing, relational data warehouses are often represented by means of a star schema or a snowflake schema [73, 16, 27, 83]. Those require less table joins than the traditional relational model and thus allows for more efficient query processing.

The development of ETL-components is a rather complex task. With regard to its extensibility and flexibility, there are three alternatives to design ETL-software. The first option uses *hard-coded scripts*, which however are difficult to maintain. The second option allows the specification of transformation rules outside of the scripts. The idea is to improve the ability to quickly update the application as the rules change or the system is extended with other data sources. These systems are based on *code-generation*. In contrast, there are *engine-based* systems, also called *metadata driven* systems. The specification of transformation rules is directly read and interpreted by an engine that controls the entire loading and refreshing process of the data warehouse.

The metadata repository plays a key role for data warehouse design and the development of software components as well. Designers and application developers may get information to achieve their tasks from the repository where, for example, results of CASE-tools, design experiences, and usual documentation of the system are stored.

1.5 Maintaining the Data Warehouse

Maintenance or administration are performed by warehouse administrators and are directly coupled with building and using the data warehouse. This process concerns the periodic refreshment of the data warehouse, updating of software when application requirements change, e.g., additional data sources have to be appended, their structure changed, etc. It may also require to specify further transformation rules or to calculate derived data or additional aggregation and summarization results. The development of new applications that access the data warehouse may be seen as being part pf maintenance as well.

The maintenance process of data warehouse systems requires the permanent updating of the metadata repository which may be happening either automatically or manually, depending on the metadata type and use. The manual updating of metadata can fall under the responsibility of different people. For example, metadata that refers to the terminology of business users is not necessarily the concern of warehouse administrators but of people responsible for knowledge management in an enterprise.

2 Metadata for Data Warehousing

As a consequence of the growing popularity of data warehousing, metadata management in this area gained significant attention during the last years. Especially industry recognized the considerable potential that lies in an effective metadata management.

In the following, we discuss the goals of metadata management, present a possible classification of metadata along certain dimensions, and introduce some aspects that have to be considered when building a metadata management system.

2.1 Objectives

We recapitulate the objectives of metadata management introduced in part I, section 1 and refine them for the particular case of data warehousing. The generation and management of metadata generally serves two purposes: (1) to minimize the efforts for administration of a data warehouse and (2) to improve the extraction of information from it. The first objective mainly concerns

- *automation of various administration processes for information systems.* During the execution of processes (like loading and refreshing of data warehouse systems), the metadata is automatically accessed and used. Its task is to control these processes and to keep track of them.

- *supporting system integration.* Inherently, both schema and data integration rely on metadata that includes information about the structure and the meaning of data sources and of the target system. Additionally, transformation rules to be applied to the original data are required and may be stored as metadata as well.

- *enforcing complex security mechanisms.* Access control in a data warehouse system may require sophisticated methods. Problems are related to data transformation that may change the value of data in the warehouse compared to the operational sources. For example, the operational sources may contain harmless information about single figures of an enterprise but the summarization of the values stored in the data warehouse may be a top secret information. On the other hand, individual incomes of the employees on the source site are secret, but the total volume stored in the data warehouse may not be a critical information. Metadata should provide the access rules and user rights for the whole data warehouse system.

- *supporting analysis and design of new applications and business process modeling.* Metadata increases control and reliability of the application development process by providing information about the meaning of the data, its structure, and origin. Furthermore, metadata regarding design decisions adopted for existing applications may be reused or updated for new ones.

- *improving the flexibility of the system and the reuse of existing software modules.* Semantic aspects that are likely to change frequently (so-called business rules) are explicitly stored as metadata outside the application programs (scripts, end-user applications, etc). These business rules can be updated without much effort in accordance with possible new requirements. They may be reused also in a different context, either the entire code or the design knowledge behind it. As a consequence, maintenance of the system is substantially easier, and the software may be extended and adapted without difficulty. At runtime, an engine accesses and interprets the metadata in order to achieve the particular tasks of manipulating, cleaning, transforming, and using data.

The second objective refers to the effective extraction of information from data. In particular, metadata should be available for achieving the following tasks:

- *improve data quality.* Data quality includes dimensions like consistency (whether the representation of data is uniform and no duplicates, no data with overlapping and confusing definitions exist), completeness (whether data is missing), accuracy (the conformity of the stored with the actual value, including precision and confidence of the data), timeliness (whether the recorded value is up-to-date). Furthermore, data quality requires correctness that has to be defined in accordance with the given application (e.g., whether attributes have expected values). To this end, quality assurance rules have to be defined, stored as metadata and checked each time the data warehouse is refreshed. In addition, high data quality requires the support of data tracking. Metadata provides information about the creation time and the author of the data, the source of the data (data provenance), the meaning of data at the time it was captured (data heritage), and the path followed from source to the current site (data lineage) [21]. In this way, users may reconstruct the path followed by data during the transformation process and verify the accuracy of returned information.

- *improve query, retrieval and answer quality.* Metadata allows to pose precise, well-directed queries and reduces the cost to users of accessing, evaluating, and using appropriate information. Metadata may help revealing associations between data elements.

- *improve data analysis.* Methods for data analysis cover a large spectrum, starting with simple reporting applications that include summarizations, continuing with OLAP, and ending with complex data mining applications. In this context, metadata is necessary to understand the application domain and its representation in the data model (in the classical case the database schema) in order to adequately apply and interpret results. Furthermore, metadata may be automatically accessed by analysis applications and used to perform their specific computations.

The presented objectives may be supported by metadata in two ways, passively and actively:

- by providing the necessary information that may help users to achieve their tasks, i.e., administration of the system, maintenance and building of new applications, query, retrieval, and data analysis. The basic requirement, namely understanding of the data warehouse system, is only possible if documentation about the structure, the development process and the use of the data warehouse system is available.

- by storing elements which are automatically accessed by the data warehousing processes at execution time. In this way, the processes are "metadata driven" since the metadata specifies what has to happen and simultaneously keeps track of them. Obviously, humans also have access to these elements in order to better understand the system and to update the processes in accordance with new requirements.

The availability of a metadata management system as a unique documentation source for users brings other benefits as well: sharing of knowledge and experience, reuse of expertise, elimination of ambiguity and guaranteed consistency of information within the enterprise. A unique terminology is enforced. In this context, note that data warehousing concerns not only technical problems related to correct system integration and efficient use of data. In fact, the more unpredictable and hard to handle aspects are the agreement upon a common understanding and the consistent enterprise view that should be enforced by the data warehouse. Once captured and stored, metadata promises to be the key solution for reducing such technical and business problems. It aims at both, supporting effective information processing within the data warehouse and ensuring a means for people to communicate, understand, and interpret information provided by the data warehouse system.

2.2 Classification of Metadata

In the following we identify six dimensions for classifying metadata in data warehousing. Metadata characteristics play an important role in establishing the structure of the metadata repository (see 2.3).

2.2.1 Criterion "Type"

We distinguish between metadata concerning *primary data* in the data warehouse system and the *data processing*, that means, metadata regarding the processes running within the data warehouse.

- **Metadata for primary data.** Primary data consists of all data managed by the data sources, the data warehouse, the data marts, and the applications. Thus, the corresponding metadata includes information related to the structure of data sources, data warehouses, and data marts. In this context, we distinguish between metadata concerning the entire schema (e.g., schema description, statistical values as, e.g., the number of entries in the database) and metadata associated with parts of the schema. Examples include quality attributes that specify the credibility of single attributes (e.g., birthdate). Furthermore, the database schema may be extended with additional attributes that associate various values to individual entries in the database, as e.g. the updating date of records, or information regarding the data collection (what, where, who, when and why). Both the attributes and the values provided for each entry can be understood as metadata. Code tables are other classical examples where no clear distinction exists between metadata and data. Code tables contain codes used in everyday acquisition of data and their textual descriptions (e.g., 0 stands for male and 1 for female).

- **Data processing metadata.** This is information associated with data processing: information regarding the loading and refreshment process, the analysis process, and inherently the administration. Examples are rules specified for data extraction, transformation, and aggregation which are defined by means of executable specification languages. They have to be accompanied by a textual explanatory description, usually in natural language (see 2.2.2). According to the operation type performed, rules may be further classified as filter, joiner, aggregator, etc.

  Process metadata also includes logfiles and schedules for establishing the time and order of the execution of (parts of) processes.

Note that a special category constitutes metadata concerning the organisation of the enterprise. This category may be both considered as belonging to the first class (when organizational information is simply handled as primary data) or as a stand-alone class. Organizational metadata includes administrative data, information related to the staff of the enterprise, as e.g., user rights to access the data warehouse, data sources, and data marts.

2.2.2 Criterion "Level of Abstraction"

In analogy to database design steps, knowledge may be provided on three levels of abstraction: conceptual, logical and physical. The conceptual perspective includes the entire description of the business using natural language. For example, main business entities like "customer", "partner" are defined and their features and relationships with other entities explained. Rules that govern extraction/transformation are described by means of natural language in order to be understandable for any system user. Also information related to the use of the system, defined queries, views and existing analysis applications are provided at this level.

Logical metadata maps the conceptual perspective to a lower level that includes for example the relational schema of the databases, the description of extraction/transformation rules in pseudocode (or using a mathematical language), etc.

The physical perspective provides the implementation level. It contains the corresponding SQL code of business rules, index files of relations, and code of the analysis applications. When the distinction between logical and physical metadata is not justified either for end-users or technical users, the two levels may be merged into a single one.

2.2.3 Criterion "User View"

This criterion is related to the informational objective of the metadata. The same information and structure may be seen from different perspectives, depending on the users that need it. As a consequence, metadata may be divided into subclasses that satisfy the interest of certain user groups. For example, some metadata may be relevant for certain departments, for knowledge managers (needing additional information beyond standard reports and OLAP results), for business users, or for more technical users (e.g., data warehouse administrators, analysis application programmers). In this context, a common distinction is *business* (or application-driven) versus *technical* metadata.

Business metadata is needed by end-users to better understand the application and thus better use the information system; it comprises application-specific documentation (user profiles, access maps, usage tips, navigational aids), business concepts and terminology, details about predefined queries and reports. Also, context information like measurement units specification (currency, length), date format (American or European convention) or dictionaries, thesauri, and domain-specific ontological knowledge are considered as metadata.

In contrast, technical metadata is used by database administrators who develop and maintain the system and by analysis application developers. Classical examples are the data dictionaries of the sources, data warehouse, data marts, and the code of data transformation rules.

Metadata classes defined on the basis of user views (in particular business and technical metadata) are not disjoint; they provide different extracts of the same collection of metadata, and may overlap as well.

2.2.4 Criterion "Origin"

This classification takes into consideration the origin of the metadata: the tool that produced it (the ETL-component, a CASE-tool used during warehouse design), the source that provided it (e.g, the data dictionary of the operational systems, of the data warehouse or data marts), or the system the metadata has been imported from, etc. Metadata may also be produced by certain users (business user, database administrator) which should be individually identifiable. In this case, they represent the origin.

2.2.5 Criterion "Purpose"

As a counterpart to the previous criterion, a "purpose" or "use" criterion may be used to classify metadata according to activities such as extraction or transformation, building a multidimensional view, data mining, reporting, etc. However, the distinction between the different classes is vague since some metadata (e.g., schema description) may be used for most purposes: administration and maintenance, refreshment and analysis.

Furthermore, we may coarsely identify (in an orthogonal dimension) metadata for informational and for controlling purposes. Note that all metadata may serve the informational purpose, so the latter category may possibly be a subset from the former. The controlling purpose class may be divided into subclasses with regard to the operations applied to them: the metadata is either only read (schedules for processes) or read but then directly exploited (transformation rules used by metadata-driven engines), or both read and also updated (logfiles).

2.2.6 Criterion "Producing/Consuming Time"

Metadata is divided in accordance with the time it was captured or generated:

- designtime collected metadata (e.g., schema definition of sources and of the data warehouse, access rights, transformation rules, etc.),
- buildtime generated metadata (e.g., logfiles, data quality attributes, particularly data tracking),
- runtime generated metadata (e.g., logfiles, schedules, usage statistics, job specifications).

In analogy, the classification according to consumption time specifies when the metadata may be needed: at designtime, when the system is developed (dictionaries, CASE tools metadata, reverse engineering tools metadata, data mining metadata), at buildtime (schedules, transformation rules, data quality rules), or at runtime (configuration files, OLAP metadata).

2.3 Metadata Management

The previous classification can be used as the basis for the construction of a metadata management system. In the following, we focus on the functionality and structure requirements for a repository and we discuss some aspects related to its architecture.

2.3.1 Functionality

Metadata is extracted, collected and stored in a repository, which is a structured storage and retrieval system, usually implemented on top of a database system. In order to achieve the objectives proposed, the metadata repository should supply certain functionality: the *provision of information* based on the underlying metamodel, *automatic access*, *version and configuration management*, *impact-analysis*, and *notification*. We consider these aspects in turn.

Provision of information. The repository has to offer suited mechanisms for querying, filtering, navigating and browsing in order to satisfy the information needs of the users.

The structure (schema) of the repository should support *querying* of the repository according to certain conditions. For example, it should be possible to pose queries to select all single activities which make up the refreshment process, or all logical metadata related to the business element "partner". The structure of the repository should also allow the selection of metadata that has a certain origin (see 2.2.4), a certain purpose (see 2.2.5) a certain production time, etc. This require metadata to be "labeled" with keys that contain the appropriate values (for example, in relational terminology, attributes for purpose, origin, etc. should exist).

Relationships/dependencies between individual metadata elements (like, e.g., one relation has certain attributes) are important for understanding the system and usually are stored as well. Ideally, the repository should provide not only explicit but also implicit (hidden) relationships between aspects of managed metadata. As a consequence, an essential functionality requirement is the *navigation* within the metadata collection. Starting with a certain element, the user may navigate to other elements along existing relationships. Navigation is "driven" by the underlying schema which is specified on a conceptual level by the *metamodel* of the repository.

Filtering refers to the selection of relevant information when search criteria are not necessarily provided by the structure of the repository. This means, besides querying of fixed attributes, filtering presumes the search of keywords within textual descriptions. In this way, all information related to a certain topic may be provided.

Browsing requires an appropriate, user-friendly (and thus highly graphical) interface for interacting with the repository. User views play a central role for browsing, since they restrict access to information according to user interests. Each user view has a defined starting point for browsing and navigation which provides the elements the user has to start with when exploring the repository.

Metamodel. In order to effectively provide the described functionality, an appropriate metamodel has to be chosen. A metamodel is the conceptual schema of the metadata repository. It specifies metadata elements and relationships existing between them. Starting with our classification in 2.2, the *type* of metadata (recall 2.2.1) and the *level of abstraction* (2.2.2) play an important role in the metamodel. In other words, the metamodel requires elements to represent both, the process metadata and the primary metadata, and furthermore do this on different levels of abstractions (e.g., conceptual process element and logical process element). Obviously, relationships exist between different representations, allowing for navigation. Other classification criteria (2.2.4, 2.2.5, 2.2.6) may be represented through attributes of the elements in the metamodel. Note that user views (2.2.3) are enforced through adequate navigation mechanisms, not the metamodel.

Inherently, the metamodel has to be extensible in order to allow users to define application specific metadata elements, relationships, and constructs when application requirements change.

Repository Access. To be usable, the repository has to be permanently up to date. Users access it either manually and ad-hoc or indirectly via tools, e.g. storing metadata produced by a compiler. Even for manually entering metadata, appropriate tools that guide users through the structure (the metamodel) of the repository are necessary. With an adequate interface, all metadata elements and their relationships are safely introduced and stored. Tools should also make the entering of data easier by providing templates (e.g., for transformation rules), reusable components, etc.

The successful use and longevity of the repository may be guaranteed only if interfaces for interoperability with other repositories are available. This may be enforced if standards for interchange representation formats (see Section 3) are adopted.

For access by other tools, a comprehensive application program interface (API) is required.

Version and Configuration Management. Important changes of metadata (e.g., due to schema updates of operational sources schemas) require the creation of different versions and their storage in the repository. However, problems may arise with inconsistent links, or descriptions on the conceptual level which are not valid anymore when changes on the implementational level occur (e.g., changes of code). Thus, the repository has to provide notification mechanisms for detecting such errors in structure and actuality of the metadata.

Impact Analysis. This feature enables administrators to evaluate the impact of potential changes in the data warehouse system before they are made. It assumes simulations for detecting which parts of the system are affected when application requirements change. For example, changes in the schema of sources may have consequences for transformation rules: type mismatches, violations of referential integrity, etc.

Notification. The repository should provide a notification mechanism for both, interested users and tools. Changes in the repository can thus be propagated to certain tools that registered their interest in being notified. Also, users who previously “subscribed” for the repository (e.g., application developers) are informed about significant changes (e.g., new versions of the repository).

2.3.2 Repository Architecture

Ideally, an enterprise should have a single repository for managing its metadata. However, centralization does not always work in reality. The reasons are multiple: either the historical evolution of various departments may have implied an asynchronous development of repos-

	Meta Standards		Standards	
Primary application	**Representation**	**Exchange**	**Representation**	**Exchange**
General	RM-ODP		ISO/IEC 11179	
Software Engineering	MOF		UML,IDEF, **OIM, MDAPI**	CDIF,MDIS, **CWMI**
Repository			IRDS, PCTE	
Semistructured	RDF		XML	XMI, XIF

Table 3: Data Warehouse Metadata standards taxonomy

itories, or political and organizational aspects do not allow to physically manage a single repository within the enterprise. Furthermore, different tools with divergent data models, following diverse representation formats and standards have to be inevitably used for achieving the numerous tasks of data warehousing. The consequence is a multitude of proprietary metadata storage models, redundant storage of metadata in different repositories or, worst of all, no repository at all.

The ideal solution is to provide a single conceptual view of metadata existing in an enterprise. Repositories may be maintained individually and different access rights enforced on them. In this case, federated metadata management strikes a trade-off between the advantages of centralization and those of local control.

3 Standardization Efforts for Data Warehouse Metadata

In sharp contrast to the plethora of standards existing in the are of general metadata managemen, there is a rather short supply of standards in the data warehouse area. Besides a standard concentrating on multidimensional aspects, we only found an open call of OMG and a proposal of Microsoft. Table 3 contains the classification of these standards according to the schema in Part I, Section 3.

3.1 MDAPI

The Multi-Dimensional API (MDAPI) version 2.0 [112] is a structural standard defined by the OLAP Council in 1998.

The OLAP Council was established in January 1995 to serve as an industry guide and customer advocacy group. Members of the council are IBM, Oracle, Sun, Platinum, Hyperion Solutions, NCR, Cognos, Business Objects, and others.

The defined API, among other things, provides metadata functions for OLAP multidimensional databases. MDAPI is principally a specification only. The OLAP Council publishes the specification, the members of the OLAP Council provide platform dependent implementations of MDAPI. Version 2 is a read-only API and does not offer a mechanism to modify data in a connected schema or the structure of a metadata schema. The API is available for COM and Java.

The API access to metadata requires an initialization by creating so-called *connection* objects. The basic information units are *members* which can be understood as abstractions of cells plus related additional features described by *properties*. A special subtype of members are *measures* with predefined attributes like scale, precision etc. At runtime, members and their properties receive concrete value assignments corresponding to their specified data types. Members are organized in *dimensions* consisting of one or more *hierarchies* with different *levels* of detail and aggregation. Special types of dimensions are available for time and measures.

MDAPI offers two basic kinds of queries, namely *Cube* for data access and *MemberQuery* for metadata retrieval.

3.2 CWMI

In September 1998, the Object Management Group (OMG) has issued an RFP (Request for Proposal) for "Common Warehouse Metadata Interchange" [96] with deadline September 1999. Its objectives are:

1. Establishing an industry standard specification for common warehouse metadata interchange;
2. Providing a generic mechanism that can be used to transfer a wide variety of warehouse metadata;
3. Leveraging existing vendor-neutral interchange mechanisms as much as possible.

Proposals are required to cover a complete specification of the syntax and semantics needed to export/import warehouse metadata and the common warehouse metamodel. This may consist of a specification for the common warehouse metamodel, APIs (in IDL) and/or interchange formats. Proposals *must* be compatible with MOF/UML/XMI, *shall* be compatible with RM-ODP and "should be aware of" MDIS, MDAPI, OIM.

Up to now, it is not clear whether CWMI will be a structural standard defining a common warehouse metamodel or "only" an exchange standard.

3.3 OIM Extensions for Data Warehousing

OIM (see Part I 3.6) has been extended to support metadata management for data warehousing. The following models have been added:

- The **Tfm** *Database Transformation Information Model* describes what transformations (from production databases into a data warehouse) do and what data they access. The Tfm is an extension of the Database Information Model (Dbm). The Tfm model covers basic transformations for relational-to-relational translations. Transformations can be packaged into groups.
- The **Olp** *OLAP Information Model* describes information about multidimensional databases. The following elements can be described:
  - Cube: is the basic component in multidimensional analysis. It describes a fact table and one or more dimension tables which together form a star or snowflake schema.
  - Virtual Cube: is a cube which is not materialized.
  - Partition: describes the part of a cube (partitioned for performance or storage reasons).
  - Aggregation: defines pre-calculated roll-ups of data stored in a cube.

  - Dimension hierarchy: states how a dimension decomposes into sub-dimensions; defined independently of the OLAP stores using it.

- The **Sim** *Semantic Information Model* is another extension of Dbm, accommodating schema-to-semantic mappings. Specifically, the Sim model holds descriptions of semantic models and their relationships to the underlying database schema. These connections enable a user to interact with data in a database without learning data manipulation languages.

The *Legacy/Record-Oriented Information Model* includes models necessary to capture information about record definitions (Cobol, PL/I etc.) The primary purpose of this model is to allow for specifying data warehousing transformations on common (non-relational) sources. This model does not contain any information specific to a file system or database - this will be added in later information models.

Microsoft expanded OIM with supplementary sub-models for its own data warehousing tools within the SQL Server Data Warehousing Framework.

3.4 The Zachman Framework applied to Data Warehousing

Based on the general Zachman Framework (see Part I, Section 3.17), [53] develops a framework for managing enterprise knowledge. It consists of four layers dealing with modeling of product aspects, modeling of enterprise aspects, modeling the engineering of an enterprise, and finally a repository layer which reflects the lower layers and their relationships to each other. Data warehousing is understood as one central, first step from information resource management towards a comprenhensive knowledge management environment. The Zachman view is demonstrated to be useful for defining

- a data warehouse strategy in the form of principles to guide behavior (consisting of a statement, a rationale for it, and its major implications), and
- a methodology for building and extending a data warehouse.

The methodology is complemented by a metadata schema which is organized along the dimensions (columns) of the Zachman Framework. For each dimension it contains modelling constructs across all relevant perspectives: the data or entity dimension submodel, e.g., ranges from business entities (owner perspective) down to data warehouse tables (builder and designer perspective); the motivation dimension constructs covers business goals as well as data warehouse service level objectives.

The authors claim their model not to be complete concerning all aspects of data warehousing, but to serve as a launching point for designing a metaschema tailored to a specific company's requirements and priorities.

4 Commercial Data Warehouse Metadata Management

For all components in the architecture of a data warehouse system (see Section 1), commercial tools exist in abundance[21]. Each of these tools is a metadata consumer and/or producer. The creation and maintenance of metadata for these products requires a lot of effort. Besides that,

[21] An excellent resource is Larry Greenfield's site at http://pwp.starnetinc.com/larryg/index.html.

it is very well possible that the same piece of metadata has to be defined for several products. This makes the job of keeping metadata consistent error-prone and difficult to achieve. It is therefore essential to enable the sharing of metadata and automating metadata management.

4.1 Metadata Integration Approaches

Vendors are employing a variety of centralized and distributed approaches for metadata management. The concepts fall into one of three categories [119]:

1. **Central repositories** for metadata sharing and interchange;
   Naturally "stand-alone" repository vendors like Viasoft (Part I, 4.4) or Platinum (Part I, 4.3) come into play. This approach makes it absolutely necessary to have a standardized, extensible meta model like OIM in order to fulfill the needs of all the diverse products someone wants to use. Another example of this category is the solution of Oracle Corporation (4.4). Their offerings include a centralized repository to manage data warehouse metadata.

2. Metadata **interchange standards** defined by vendor coalitions;
   Such standards require that the coalition members agree on a common sub-model to exchange meta data between their tools. Each vendor is free to specify additional metadata for his own products usage. Interchange standards include MDIS, CDIF, XMI, XIF (see Part I) and CWMI (see 3.2). As a representative exponent of this approach, we present the concept of IBM (4.5) below.

3. Vendor specific **"open" product APIs** for metadata interchange;
   In this approach, a vendor offers an API that enables other parties to import and/or export metadata from their products. Examples of vendor interfaces that enjoy wide support from a variety of third-party tool vendors include Ardent Software (4.6) and Informatica (4.7).

4.2 Platinum

Platinum's repository model has been extended to allow definitions of source, target and transformation rules used in building data warehouses and data marts. The model has no constructs for metadata about business aspects of a data warehouse, e.g. dimensions, business concepts, business key figures.

4.3 Rochade

Viasoft provides an additional RIM (Repository Information Model), DW-RIM, which covers both technical and business aspects of data warehousing. Specifically, DW-RIM supports the definition of transformation rules as well as business aspects of data warehousing. Transformation rules link source and target data elements and specify both the implementation as well as the business aspect of a rule. Business elements comprise aspects like available queries, reports, business terminology, dimensions and key figures.

4.4 Oracle

The Oracle Warehouse Builder (OWB)[22] delivers a complete data warehousing solution containing the following primary components:

- *OWB Repository*: stores OWB metadata used to create the warehouse. Oracle acquired OneMeaning, Inc. - their products ("Exchange", a leading metadata interoperability tool, and "Marlow", a metadata repository) constitute the metadata part of Oracle's Warehouse Builder.

- *OWB User Interface*: is a graphical tool for modeling and rapid construction of the warehouse environment.

- *OWB Warehouse Administrator*: manages the workflow process and the entire life cycle of building and populating a data warehouse.

- *OWB Software Development Kit*: provides customers with the ability to extend the capabilities of OWB by wrapping their own data extraction programs.

- *Oracle Integrator for SAP and Peoplesoft*: enables access to these applications' data and metadata.

The OWB logical warehouse model consists of four sub-models:

1. The *enterprise data model* consists of information about tables, columns, foreign key joins etc.

2. The *warehouse data model* comprises contains the definition of data tables, summary tables, dimensions and hierarchies, extraction definitions and transformations.

3. The *software library* contains the list of sources that can be accessed by OWB. Additional sources for extraction can be defined using Oracle's Software Development Kit.

4. The *transformation library* stores the reusable formulas and expressions that perform transformations between objects; the transformation library contains a set of pre-defined PL/SQL functions and transformations.

Oracle Corporation is (together with IBM) the main opponent of Microsoft. It is therefore no surprise that Oracle announced the "common warehouse metadata standard" (CWM), which allows third-party software to integrate with the metadata repository. CWM is based on open standards such as CORBA and UML for object modeling, and XML for migration and interchange tools. CWM incorporates both technical and business metadata and covers (according to Oracle) all aspects of warehousing. It is expected that Oracle submits their CWM to the OMG's CWMI RFP (see 3.2), thereby not only proposing an interchange standard, but also a structural one.

[22]http://www.oracle.com/

4.5 IBM

IBM's Visual Warehouse[23] provides the means to define, build, manage, monitor and maintain a data warehouse environment. IBM has formed partnerships with different vendors, e.g., with

- *Evolutionary Technologies International*, whose Extract tool generates 3GL programs to extract, transform, and consolidate data from a variety of data sources, enabling transformations not easily expressible using SQL,
- *Vality Technology*, who sell Integrity, a data re-engineering tool extending Visual Warehouse with regard to data quality, and
- a couple of partnerships with vendors of analysis tools, e.g. Brio Technology, Business Objects, Cognos, Hyperion Solutions.

Metadata for both administrative and business users is integrated in the Visual Warehouse Information Catalog and is made available through an interface tailored to end-users, including the ability to navigate and search using business terms. Metadata exchange is supported via MDIS (see Part I, 3.9) and IBM's own Tag Language format. IBM also announced its intention to submit a proposal to the OMG's CWMI RFP (see 3.2).

The Visual Warehouse Information Catalog distinguishes between different categories of object types. Every object type must belong to one category. An object type's category affects how the Information Catalog handles it. The categories are:

- *Grouping*: object types that can contain other object types,
- *Elemental*: atomic object types that are the building blocks for other object types,
- *Contact*: object types that identify a reference for additional information about an object,
- *Program*: object types that represent and describe applications capable of processing the actual information,
- *Dictionary*: object types that define business terminology,
- *Support*: object types that provide additional information about the information catalog or enterprise,
- *Attachment*: the only object type belonging to this category is the Comments object type.

4.6 Ardent Software

Ardent Software[24] offers several products for data warehousing including the product line of the former Prism Corporation (Prism Warehouse Executive and Prism Warehouse Director), and also Ardent's DataStage product. Warehouse Executive is an ETL-tool which generates code (Cobol, Java, ABAP/SAP). Warehouse Directory is the metadata store. A key feature is the integrated metamodel that contains business, technical, operational and quality data.

[23] http://www.software.ibm.com/data/vw/
[24] http://www.ardentsoftware.com/

Third-party tools can also access the repository metadata in order to facilitate end-user queries and operational data management.

Ardent announced the MetaConnect initiative, allowing users to exchange metadata among different warehouse tools. The initiative is based on so-called MetaBrokers. The core of each MetaBroker is built by a patented technique which enables the decomposition and recomposition of metadata into simple units of meaning. This semantic translation facilitates the exchange of metadata between extraction, transformation and loading, business intelligence and data modeling tools. MetaBrokers use a tools standard import/export file for metadata exchange. Metadata exchange between two tools does not rely on a central repository, but only on a MetaBrokers transient storage. When an exchange has been completed, the transient storage is deleted. Currently, MetaBrokers are available for Erwin, ER/Studio, Impromptu, and Business Objects. Ardent also announced the MetaStage product as a materialized repository for their MetaBrokers.

4.7 Informatica

Informatica Corporation[25] sells the PowerMart and PowerCenter products. PowerMart is a product in the ETL-area, its metadata repository coordinates and drives a variety of core functions including data extraction, transformation, loading and management. PowerMart's Repository Manager is used to create and maintain the PowerMart Repository and its metadata. Three levels of detail exist within the repository:

- *Folders* - The highest-level logical groupings of data. This level permits sharing through Shared Folders, which may contain custom groupings of shareable transformations.
- *Mappings* - The business rules that source data must follow when it populates the data warehouse.
- *Objects* - The lowest level of detail with wich users interact; examples are source tables, data mart tables and transformations.

The PowerMart Server features parallel execution of its three process engines: The Extractor, the Transformation Engine and the Loader.

The PowerCenter product's main feature is the Global Repository, allowing to share metadata between local PowerMart installations. It plays three roles:

1. it defines and documents the operational source data store; it contains source definitions, target definitions and mappings generated during construction of the operational store;
2. it enables sharing among local data marts by maintaining common object definitions for sources, targets and transformations;
3. it maintains local data mart linkage intelligence and security information to enable trusted interaction and sharing within the enterprise.

The Metadata Exchange (MX2) Architecture provides a set of application program interfaces (APIs) which can be used by OLAP, query and access tool vendors to integrate their products with Informatica's open metadata repository. MX2 is based on UML for information modeling, and COM for object interoperability. According to Informatica, the

[25]http://www.informatica.com/

MX2-Architecture is compatible with Microsoft's OIM. The MX2 APIs allow read and write access to the underlying repository. Different vendors announced their support for the MX2 architecture, e.g. Brio Technology, Cognos, Business Objects and others.

5 Data Warehouse Research Projects

In this section we review current research projects on data warehousing and enumerate relevant activities with a link to metadata management.

5.1 Whips

The database group at Stanford University started to explore open problems related to data warehousing within their Whips project [49] in 1995. The main focus lies on developing algorithms and concepts for the data integration component in data warehouse environments rather than on data querying and analysis. The results are intended to reach goals like modularity, scalability, permanent availability (in particular through incremental view maintenance), consistency, and support for a broad heterogeneity of data sources.

The Whips architecture assumes pairs of monitor and wrapper components for each data source. Both types provide relational access for a query processing component (acting for data warehouse load and maintenance) and an integrator module which works as the main coordination instance. The actual data warehouse contents are materialized views, the data warehouse itself is also a component with a wrapper. Each view is controlled by a view manager for initialization (through the warehouse wrapper) and maintenance, supported by the query processing component. The integrator establishes communication between source monitors and view managers.

From a software architecture point of view, the Whips prototype [76] is realized as a set of distributed communicating objects on CORBA basis. The Whips metadata store keeps all information on the source schemas and view definitions and records data about active objects in the data warehouse environment. One assumption of Whips is that nearly the complete DW data integration tasks can be realized starting from declarative view specifications and generic algorithms, e.g., for handling view maintenance. Only wrapper components have to be built manually.

In the meantime, Whips produced various results[26] for many problems related to data integration and view maintenance. An interesting topic also considered is the view data lineage problem: starting from a certain data fragment in a data warehouse or data mart which, e.g., represents unexpected key figures, it is necessary to trace back the derived data to their origins in the data sources. This task has also a connection with advanced explanation facilities in data warehouse environments which in part can be accomplished by applying techniques similar to those used in data mining. A solution for the lineage problem could be supported by managing metadata about the data integration processes. The tracing algorithms presented in [26], however, assume relational sources only and pursue a similar approach as used for reaching selfmaintainability of views, namely by introducing appropriate auxiliary views. These views contain additional data which can be used instead of accessing the data sources. More complex transformations not expressible by declarative expressions or even process traces are not considered.

[26]http://www-db.stanford.edu/warehousing/publications.html

5.2 Squirrel

The Squirrel Project [50] at the University of Colorado belongs to the information integration projects which rely on materialization of derived and mediated information. Therefore Squirrel mediators resemble data warehouses or at least data marts. They locally store data and provide query and incremental update services. While data warehouses are usually decoupled from the data sources and are only refreshed in certain time intervals based on polling the data sources for changed data, mediators usually are required to provide up-to-date information for clients. Squirrel mediators constantly take change logs from relational database sources, fill them into update queues and propagate the changes into the derived materialized views.

The complete mediators, i.e., data and services to be used at runtime, are generated from specifications in a language called ISL [122]. An ISL specification consists of schema definitions for those source relations the views are based on, SQL-like export class definitions which represent the views accessible by mediator clients, and a set of conditions and correspondences.

Correspondences are predicates describing the relationship between attributes of views and source relations and are used in the 'where' part of the export classes. Correspondences may also include user-defined comparisons available as external object files. Overall, they solve the problem of object matching, when objects from different sources have to be unified in a view. The simplest case is key-based matching, but also very complex conditions can be specified with the correspondence mechanism.

The conditions state specific constraints that have to be monitored on the views and lead to certain actions, e.g. sending of warnings. ISL specifications do not contain explicit descriptions of procedures or complex processes, a limitation which is mainly justified through concentration on sources of relational structure and the assumption of reliability and consistency of the source data. This means that advanced data transformation and cleaning is not in the project focus. However, the specifications are actually the metadata describing contents and behavior of Squirrel mediators. The Squirrel project also does not consider the management of ISL specifications or the composition of specialized mediators to larger services.

The implementation is realized by compiling the specifications to an intermediate language, Heraclitus, further translation by the Heraclitus compiler and linking to executable code together with certain Squirrel libraries as runtime environment for execution and query processing.

At the intermediate level, both incremental maintenance of the views and monitoring of conditions are realized by event-triggered active rules and so-called view decomposition plans. The latter have a close correspondence to query execution plans as used for query optimization, however with a partly redundant introduction of relations for intermediate results. An advantage of this approach is that incremental maintenance and monitoring can be done by simplified active rules and simultaneously be used in various different base data update scenarios.

5.3 DWQ

The ESPRIT Long Term Research Project DWQ[27] (started in late 1996) explores foundations of quality aspects in Data Warehousing [64]. The project in particular proposes to link all

[27]http://www.dbnet.ece.ntua.gr/~dwq/

data warehousing tasks to be executed with explicit quality information and store them in a repository.

The project work is divided into 6 main packages which tackle problems concerning architecture and general quality issues, source integration, data reconciliation, multidimensional aggregation and reasoning, update propagation and query optimization.

While many results are primarily technical contributions of new techniques for the various data warehouse tasks, the most interesting efforts of DWQ in the metadata context stem from the data warehouse framework and quality-related activities [62]. One main decision consists in the adoption of two orthogonal views on data warehousing:

1. With respect to the level of abstraction, conceptual, logical and physical aspects can be distinguished: The *conceptual* perspective is directed towards the business model of information systems, while *logical* and *physical* view specify two levels relevant for realizing data warehouses.

2. With respect to the organizational focus, we have *operational* sources supporting certain departments of an enterprise, the data warehouse based on the overall *enterprise* model, and the anaylsis-oriented client side, mainly *dispositive* applications using data marts derived from the data warehouse.

While the first view plays an important role in many other areas, too, in particular in DB design and even more general in software design, the second naturally reflects the 3-tier architecture of information integration from sources to consumers with a mediation component in between. One notable aspect within the DWQ framework is the understanding of local sources, resp. their models or schemas, as views on the enterprise model, although the data flow actually runs the other way round, namely from source systems to the data warehouse.

Based on the combination of both perspectives, a meta model for data warehouses was developed and refined to illustrate the relationship between quality requirements and data warehousing tasks. Due to the nature of the project (mainly contributions to technical data warehousing aspects), components for the usage side remain on the abstract level. In a detailed way the basic model was extended, e.g., to capture source and data integration. The technical approach adopted for this aspect is based on the 'local-as-view' paradigm [22], leading to source schema definitions in terms of the global integrated model given by the data warehouse schema. This corresponds to the interpretation of source schemas as views mentioned before.

Specific quality dimensions were assigned to each data warehousing task and combined with an explicit quality model based on the Goal-Question-Metric approach. In this model, quality goals and metrics are linked by queries whose evaluation trigger a decision whether a goal is fulfilled or not by respective measurements.

DWQ definitely makes valuable contributions to a general data warehouse meta model. The available material, however, just contains several fragments (mainly covering quality representation in general and the data warehouse loading task) which have to be complemented and integrated into a more commercial setting for data warehousing with regard to platforms and tools.

5.4 Meta-FIS

The Information Systems Institute of the University of Münster works on a metadata management repository with main emphasis on management information support. In fact, the proposed ideas cover the usage aspects of data warehousing, in particular business reporting based on generated documents and business key figures [14].

As a general framework this project uses three views on management information systems. The *actor view* specifies the management personnel that has to execute certain controlling and management tasks (*task view*) and need to be provided with suitable *information objects*.

One result of the current activities is a meta model for the information object aspects using a terminology that is rather application oriented than including the system or data source perspective. The main components of this meta model are reports and business key figures, queries which are used to fill report elements and to compute key figures, and information objects represented as multidimensional structures. The project goal is to understand instances of this meta model as specifications of management information systems and to offer support for semi-automatic generation of, e.g., the schema of an underlying data warehouse. The relationship between OLTP systems originally providing the input of Management Information Systems (MIS) and the warehouse (neither schema nor process aspects) are not considered. A second contribution, however, consists in a coarse process model for developing the MIS specification.

The Meta-FIS project explores three different ways of realizing the repository: as stand-alone implementation, coupled to one of the commercial OLAP tools, and directly integrated into a standard business software application package.

5.5 System 42

System 42 is the name of a research framework[28] established in 1997 at the Bavarian Research Center for Knowledge Based Systems (FORWISS) which comprises 5 subprojects dealing with various aspects of data warehouses. Besides a market study (including product evaluation) on OLAP [28], the construction of an online information system (Line 42) on data warehousing and a number of data warehouse application projects conducted with commercial partners, two basic research projects deserve attention:

The **MISTRAL** project explores multi-dimensional index structures with regard to their applicability in data warehousing. As a supplement to a relational DBMS as a data warehouse platform, MISTRAL proposes the introduction of an extra layer between applications and the database system consisting of so-called UB-trees. The main focus lies on implementation efficiency for multidimensional queries rather than on a conceptual representation of such index structures.

Such (design) aspects are covered by the **Babel-Fish** action which aims at the development of a tool for abstract modelling and maintenance of data warehouses. This tool provides a model editor, generation components and analyzers which are centered around a metadata repository. The vision is to reach an automatic mapping of the model to executable code and procedures. Up to what point this is possible at all, is one of the open research questions to be tackled in the project. Currently the design process and methodology for data warehouses dominates the Babel-Fish activities. One main outcome concerning metadata management up to now is an extension of the Entitiy-Relationship approach to represent multidimensional

[28]http://www.forwiss.tu-muenchen.de/~system42/

data cubes. [102] introduces additional model constructs as abstractions of the respective fact and dimension tables occuring in relational star schemas.

5.6 CC-DWS

In late 1998 the University of St. Gallen launched within their Information Management research programme a so called Competence Center Data Warehouse Strategy (CC-DWS)[29]. Several companies, mainly from the services sector, represented by data warehouse project leaders and project engineers, collaborate with personnel from the University in order to define a process model for optimal data warehouse organization [67]. Resulting from an inquiry among the partner companies, metadata management turned out to be the topic with the highest priority among data warehouse problems to be discussed in the project. As a consequence, analysis of the requirements in this area was chosen to be the first major action in the second quarter of 1999.

5.7 SMART and SIRIUS

SMART is a research collaboration started in October 1998 between the University of Zurich and Swiss Life, the leading life insurance company in Switzerland, and sponsored by the Swiss Federal Commission for Technology and Innovation (KTI). SMART explicitly focuses on metadata management rather than on discussing specific techniques implementing the data warehouse, its loading and maintenance. Starting from a systematic collection of meta data requirements and a categorization with regard to type and usage, a conceptual schema of a metadata management system will be elaborated and prototypically implemented in a suitable repository system. In order to feed metadata from various external metadata producers, SMART provides a metadata definition tool with editing facilities and translators. A corresponding metadata access tool allows interactive browsing as well as general querying facilities on top of the integrated conceptual schema. Since Swiss Life also takes part in CC-DWS, synergies, in particular through a collection of requirements on metadata management from other companies, can be expected.

Furthermore, SMART is closely related to the **SIRIUS** [42] project at the University of Zurich which proposes a new kind of middleware service for data warehouse maintenance, a so-called data warehouse refresh manager, which communicates and mediates between the operational sources and the kernel data warehouse data store based on a CORBA-like platform. The maintenance task controlled by a coodinator module is distributed over a set of cooperating mediators, each of which is responsible for a fundamental data warehousing task such as loading, data extraction and data cleaning. The mediators act in a rule-based manner, where the underlying rules as well as the basic control flow structure of the coordinator and the complete data warehouse and source schema information are stored in a metadata repository. Each source has a dedicated monitor/wrapper component which detects changes on the source and issues notifications about them to the coordinator. The metadata schema developed in SMART will be used for the SIRIUS repository, too.

[29]http://datawarehouse.iwi.unisg.ch/

5.8 MMDWE

The database research group at the University of Leipzig pursues a goal similar to that of SMART within a project 'Metadata Management in Data Warehouse Environments' (MMDWE). Starting from the same observation, namely that currently a comprehensive repository solution for all kinds of metadata relevant in Data Warehousing is not existing, MMDWE develops an UML-based schema both for technical and semantic metadata. The latter comprises constructs for representing conceptual enterprise models and multidimensional structures. Both main model components are glued together by a number of shared classes like, e.g. *Entity*, *Association Attribute* and *Mapping*. Mappings are used to describe the relationships between sources and targets of transformations on the technical layer through attached transformations which filter data, aggregate data or apply general functions (e.g. join operations), but also for linking business concepts with technical entities like tables or derived relations.

A first version of the model including an insurance-domain specific sample instantiation is described in [107]. Special emphasis is put on metadata driven query generation actually exploiting the links between semantic and technical metadata. For this purpose an OQL-like intermediate query language accessing business terms is assumed. A stepwise 'unfolding' of query expressions by inspecting the involved mappings leads to the final (SQL) query executable on the data warehouse.

Future plans of the project are directed towards metadata support for data mining applications running on top of data warehouses. Metadata in this context can be used to constrain hypothesis sets and provide hints for filtering of mining results.

6 Conclusions

The preceding sections gave a survey on the central role of metadata for data warehousing and showed how this is reflected in standards, products and research activities. Of course, the general remarks on metadata maintenance and integration in Part I Section 6 also hold for data warehousing as a special case of physically combining the contents of source systems to one central new data container which then serves for decision support applications. Two aspects make the problem a distinguished one, namely the active role metadata may have wrt. process execution when building or using a data warehouse, and the diversity of tools for the various data warehousing tasks.

As soon as metadata directly controls the behavior of tools, it is by nature more tool specific than a general representation of the same aspects serving, e.g., for documentation only. The emerging data warehouse business has many key players: not only the classic DB vendors providing the target platform and consulting firms giving advice for executing the projects, but also many specialized (and at least in the beginning) comparatively small data warehouse tool vendors, selling all kinds of ETL- and OLAP- tools. One system ought to be built, maintained and used, but necessarily many tools are involved and have to work together. Metadata should provide the overall view on how the system works, what it offers and how it is used. The all-from-one-hand argumentation of certain vendors in this context was only a dummy solution since missing competence in certain special areas yield only alliances and takeovers, but up to now no integration results.

The standard side is not yet stabilized. This is not surprising since standards for specific applications require a much higher degree of detail and more precise regulations than general purpose standards. MDAPI concentrates on OLAP aspects only. CWMI is still in the call-for-proposal phase, and even which level of detail and coverage (only interfaces ?) can be expected, has not been fixed yet. Only Microsoft seems to be ahead by offering its data warehouse meta model embedded in OIM and implemented by corresponding extensions in SQL server. The Zachman Framework based proposal in [53] up to now does not seem to have any direct commercial implications.

Data warehouse metadata management is also an attractive market for central repositories like Rochade and Platinum. They offer generic meta models tailored to data warehousing needs from their own perspective. Obviously, the system's APIs, however, are not attractive enough and in particular not based on standardized query languages. As long as tool vendors continue to manage their metadata in specialized local data stores or even in a standard database with a proprietary schema which is made only in part accessible to the outside world like the MX solution of Informatica, it remains unclear whether the standalone repository vendors will reach their general goal, namely handling data warehousing metadata like any other type of metadata in a central place, accessible for all applications and users. In fact, the user side of data warehousing cannot be seen decoupled from the general representation of business metadata of a company.

The proprietary management of metadata by tools employed for data warehousing requires at least facilities of mutual exchange. Actually, this yields the metadata integration problem mentioned in Part I Section 6 in a specific application domain. The minimum requirement is an open interface, whether standardized or not. This allows building pairwise interfaces between tools and even replication of tool-specific metadata in a global repository. A virtual integration based on an integration meta meta model providing the mapping of tool metadata to a reference metadata schema, is an ambitious goal pursued, e.g., by Ardent with its metabroker technology. Unfortunately, the integration model which is used for developing metabrokers remains implicit only and is not managed by an independent control component. The announced MetaStage product may fill this gap and in addition provides a central materialization of metadata.

Research contributions with respect to metadata are directed towards specific aspects, like introducing constructs for multidimensional representation, quality aspects or business reports, instead of pursuing the establishment of a common global metadata schema for data warehousing projects. The most comprehensive steps in this direction are made by the DWQ framework which is used for embedding the quality-specific metadata aspects, and by the MMDWE project. On the other hand, various specific metadata aspects are nowhere considered, e.g.,

- a security model attached to the representation of source systems, the target warehouse, the client applications and their respective contents, but also to processes running at build and execution time,
- metadata supporting online search, explanations, and interactive support for query formulation,
- metadata-based tracing of observations in the data warehouse on the data level through involved transformations back to the data sources.

Furthermore, data warehousing does not only consist of specifying complicated processes to be executed on data, but is itself a complex process which is only in part covered by typical software development process models. The handling of these aspects and their uniform embedding in a global data warehouse metadata schema requires further research. Finally, research results and proposed architectures for handling the general information integration problem could be applied to the data warehouse metadata integration problem, too.

References

[1] Purpose of an IRDS. http://www.irds.org/purpose.html.

[2] The Web developer's virtual library, HEAD and META tags. http://WDVL.com/Authoring/HTML/Head/.

[3] *First IEEE Metadata Conference, Proceedings*, Silver Spring, Maryland, April 1996. http://www.computer.org/conferen/meta96/paper_list.html.

[4] *Second IEEE Metadata Conference*, Silver Spring, Maryland, September 1997. http://computer.org/conferen/proceed/meta97/list_papers.html.

[5] OOPSLA'98 Workshop: Metadata and Dynamic Object-Model Pattern Mining Workshop. http://www-cat.ncsa.uiuc.edu/~yoder/Research/metadata/OOPSLA98-MetaDataWkshop.html, October 1998.

[6] University of Illinois '98 Metadata and Active Object-Model Workshop, May 1998. http://www-cat.ncsa.uiuc.edu/~yoder/Research/metadata/UoI98Metadata-Wkshop.html.

[7] *The DAMA International Symposium and The Meta-Data Conference*, Atlantic City, April 1999. Technology Transfer Institute. http://www.tticom.com/dama/.

[8] *Meta Data Europe 99: Implementing, Managing and Integration Meta Data*, London UK, March 1999. Technology Transfer Institute. http://www.ttiuk.co.uk/.

[9] Resource Description Framework (RDF). http://www.w3.org/RDF/, 1999.

[10] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J.L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[11] P. Atzeni, G. Mecca, and P. Merialdo. To weave the Web. In *Proceedings of 23rd International Conference on Very Large Data Bases*, pages 206–215, Athens, Greece, August 1997. Morgan Kaufmann.

[12] M. Baldonado, C.C.K. Chang, L. Gravano, and A. Paepcke. The Stanford digital library metadata architecture. *Intl. Journal on Digital Libraries*, 1(2):108 –121, September 1997.

[13] A. Barth, M. Breu, A. Endres, and A. De Kemp, editors. *Digital Libraries in Computer Science: the MeDoc Approach*. Springer, 1998. LNCS 1392.

[14] J. Becker and R. Holten. Fachkonzeptuelle Spezifikation von Führungsinformationssystemen. *Wirtschaftsinformatik*, 40(6):483–492, December 1998.

[15] P.A. Bernstein. Repositories and object oriented databases. *SIGMOD Record*, 27(1):88 –96, March 1998.

[16] A. Berson and S.J. Smith. *Data Warehousing, Data Mining & OLAP*. McGraw-Hill, 1997.

[17] J. Bézivin. Who's afraid of ontologies? In *OOPSLA'98 Workshop: Model Engineering, Methods and Tools Integration with CDIF*, Vancouver, Canada, October 1998. http://www.metamodel.com/oopsla98-cdif-workshop/.

[18] K. Böhm and T.C. Rakow. Metadata for multimedia documents. *SIGMOD Record*, 23(4):21 –26, December 1994.

[19] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[20] A. Borgida, R.J. Brachman, D. McGuinness, and L. Alperin Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pages 58–67, Portland, OR, May 1989. ACM Press.

[21] M.H. Brackett. *The Data Warehouse Challenge*. Wiley, 1996.

[22] D. Calvanese, G. De Giacomo, and M. Lenzerini. A principled approach to data integration and reconciliation in data warehousing. In *Proceedings of the Intl. Workshop on Design and Management of Data Warehouses (DMDW 99)*, pages 16.1–16.12, Heidelberg, Germany, June 1999.

[23] V. K. Chaudhri, A. Farquhar, R. Fikes, P. D. Karp, and J. P. Rice. OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of the AAAI-98, Madison, WI, July 1998*, 1998.

[24] S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. *SIGMOD Record*, 26(1), March 1997.

[25] M.J. Corey, M. Abbey, I. Abramson, and B. Taub. *Oracle8 Data Warehousing: A Practical Guide to Successful Data Warehouses*. Osborne/McGraw-Hill, 1997.

[26] Y. Cui, J. Widom, and J. L. Wiener. Tracing the lineage of view data in a data warehousing environment. Technical note, Stanford University, Dept. of Computer Science, 1997.

[27] B. Devlin. *Data Warehouse: from Architecture to Implementation*. Addison Wesley, 1997.

[28] B. Dinter, C. Sapia, M. Blaschka, and G. Höfling. OLAP market and research: Initiating the cooperation. *Journal of Computer Science and Information Management*, 2(3), 1999.

[29] K.R. Dittrich and R. Domenig. Towards exploitation of the data universe; database technology for comprehensive query services. In *3rd International Conference on Business Information Systems (BIS '99)*, Poznan, Poland, April 1999.

[30] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* The Benjamin Cummings Publishing Company, 1994.

[31] European Computer Manufacturer's Association (ECMA). *Portable Common Tool Environment (PCTE) - Abstract Specification*, fourth edition, 1997. Standard ECMA-149: http://www.ecma.ch/stand/Ecma-149.htm.

[32] European Computer Manufacturer's Association (ECMA). *Portable Common Tool Environment (PCTE) - C programming language*, fourth edition, 1997. Standard ECMA-158: http://www.ecma.ch/stand/Ecma-158.htm.

[33] European Computer Manufacturer's Association (ECMA). *Portable Common Tool Environment (PCTE) - Ada programming language binding*, fourth edition, 1997. Standard ECMA-162: http://www.ecma.ch/stand/Ecma-162.htm.

[34] European Computer Manufacturer's Association (ECMA). *Portable Common Tool Environment (PCTE) - Mapping from CASE Data Interchange Format (CDIF) to PCTE*, 1997. Standard ECMA-270: http://www.ecma.ch/stand/Ecma-270.htm.

[35] Federal Geographic Data Committee, U.S. Government, Washington, D.C. *Content Standards for Digital Geospatial Metadata*, 1994. http://www.fgdc.gov.

[36] M.F. Fernandez, D. Florescu, A.Y. Levy, and D. Suciu. Web-site management: The Strudel approach. *Data Engineering Bulletin*, 21(2):14–20, 1998.

[37] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the world-wide web: A survey. *SIGMOD Record*, 27(3):59 –74, September 1998.

[38] N. Fridman Noy and C.D. Hafner. The state of the art in ontology design. *AI Magazine*, 18(3):53 – 74, Fall 1997.

[39] K.A. Froeschl. *Metadata Management in Statistical Information Processing.* Springer Computer Science, 1997.

[40] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J.D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[41] Gartner Group. *Repository Market Update - 2H98*, 1998. Research Note 30 June 1998, M-04-9463.

[42] S. Gatziu, A. Vavouras, and K.R. Dittrich. The SIRIUS approach for refreshing data warehouses incrementally. In *Proc. GI-Conf. Datenbanksysteme in Büro, Technik und Wissenschaft (BTW)*, Freiburg, Germany, March 1999.

[43] U. Glavitsch, P. Schäuble, and M. Wechsler. Metadata for integrating speech documents in a text retrieval system. *SIGMOD Record*, 23(4):34 –41, December 1994.

[44] R. Goldman and J. Widom. DataGuides: Enabling query formulation and optimization in semistructured databases. In *Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 436–445. Morgan Kaufmann, 1997.

[45] F. Griffel. *Componentware; Konzepte und Techniken eines Softwareparadigmas.* dpunkt-Verlag, 1998.

[46] B. Groth, S. Herrmann, S. Jähnichen, and W. Koch. Project integrating reference object library (PIROL): An object-oriented multiple-view SEE. In *Proceedings of the 7th Conference on Software Engineering Environments (SEE'95)*, Noordwijkerhout, Holland, April 1995.

[47] L.M. Haas, D. Kossmann, E.L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proceedings of 23rd International Conference on Very Large Data Bases*, pages 276–285, Athens, Greece, August 1997. Morgan Kaufmann.

[48] O. Haase and A. Henrich. Query processing techniques for partly inaccessible distributed databases. In *Proceedings of the 15th British National Conference on Databases*, pages 123–125, London, UK, July 1997. Springer, LNCS 1271.

[49] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge. The Stanford data warehousing project. *Data Engineering Bulletin*, 18(2):41–48, 1995.

[50] R. Hull and G. Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 481–492, Montreal, Canada, June 1996. ACM Press.

[51] IEEE IDEF1X Standards Working Group. *IDEF1X97 Conceptual Modeling, (IDEF-object) Semantics and Syntax*, 1996.

[52] W.H. Inmon. *Building the Data Warehouse.* John Wiley & Sons, 1993.

[53] W.H. Inmon, J.A. Zachman, and J.G. Geiger. *Data Stores, Data Warehousing and the Zachman Framework.* McGraw-Hill, 1997.

[54] ISO/IEC. *ISO/IEC 11179:1995-199x, Information Technology - Specification and Standardization of Data Elements.*

[55] ISO/IEC. *ISO/IEC 13719, PCTE.* http://anubis.dkuug.dk/JTC1/SC22/WG22/.

[56] ISO/IEC. *ISO/IEC 10027:1990 IRDS Framework*, 1990.

[57] ISO/IEC. *ISO/IEC 10728:1993 IRDS Services Interface 3*, 1993.

[58] ISO/IEC. *ISO/IEC 10746, Reference Model of Open Distributed Processing*, 1995.

[59] ISO/IEC. *ISO/IEC 13238-3:1998 Information technology - Data Management - Part 3: IRDS export/import facility*, 1998.

[60] M. Jarke, editor. *Database Application Engineering with DAIDA.* Springer-Verlag, 1992.

[61] M. Jarke, R. Gallersdörfer, M. Jeusfeld, M. Staudt, and S. Eherer. Conceptbase: A deductive object base for meta data management. *Journal of Intelligent Information Systems*, 4(2):167 –192, March 1995.

[62] M. Jarke, M.A. Jeusfeld, C. Quix, and P. Vassiliadis. Architecture and quality in data warehouses. In *Proc. of the 10th Conference on Advanced Information Systems Engineering (CAiSE '98)*, Pisa, Italy, June 1998.

[63] M. Jarke, K. Pohl, R. Dömges, S. Jacobs, and H.W. Nissen. Requirements information management: The NATURE approach. *Engineering of Information Systems*, 2(6):609–637, 1994.

[64] M. Jarke and Y. Vassiliou. Data warehouse quality design: A review of the DWQ project. In *Proc. 2nd Conference on Information Quality, Massachusetts Institute of Technology*, Cambridge, MA, 1997.

[65] JavaSoft. Java(tm) beans specification. Technical documentation. http://java.sun.com/beans/docs/.

[66] M. Jeusfeld, M. Jarke, M. Staudt, C. Quix, and T. List. Application experience with a repository system for information systems development. In *Proc. GI-Symposium EMISA, "Development Methods for Information Systems and their Application"*, Fischbachau, Germany, September 1999. Teubner.

[67] R. Jung and S. Schwarz. Planning success for data warehousing processes: An extended business case approach. Technical report, University of St. Gallen, Inst. of Information Management, April 1999.

[68] E.W. Karlsen. The UniForM workbench - a higher order tool integration framework. Technical report, University of Bremen, Institute of Safe Systems, 1998.

[69] V. Kashyap, K. Shah, and A.P. Sheth. Metadata for building the multimedia patch quilt. In *Multimedia Database Systems: Issues and Research Directions*, pages 297 –319. Springer, 1996.

[70] V. Kashyap and A.P. Sheth. Semantic heterogeneity in global information systems: the role of metadata, context and ontologies. In *Cooperative Information Systems: Trends and Directions*. Academic Press, 1998.

[71] U. Kelter, M. Monecke, and D. Platz. Realizing distributed editors in Java based on an active repository (in German). In *Proceedings GI-Meeting Java Information Days (JIT)*, pages 340 – 353, Frankfurt, Germany, November 1998.

[72] G. Kiczales, J. Des Rivières, and D.G. Bobrow. *The Art of the Metaobject Protocol*. The MIT Press, 1991.

[73] R. Kimball, L. Reeves, M. Ross, and W Thornthwaite. *The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses*. Wiley, 1998.

[74] Knowledge Based Systems, Inc. (KBSI). *IDEF methods*. http://www.idef.com.

[75] Y. Kogan, D. Michaeli, Y. Sagiv, and O. Shmueli. Utilizing the multiple facets of WWW contents. *Data & Knowledge Engineering*, 28(3):255 –275, December 1998.

[76] W. Labio, Y. Zhuge, J.L. Wiener, H. Gupta, H.Garcia-Molina, and J. Widom. The WHIPS prototype for data warehouse creation and maintenance. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 557–559, Tucson, Arizona, May 1997. ACM Press.

[77] D. B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11), November 1995.

[78] M. Lesk. *Practical Digital Libraries; Books, Bytes & Bucks.* Morgan Kaufmann Publishers, San Francisco, California, 1997.

[79] A. Levy, A. Rajaraman, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of 22th International Conference on Very Large Data Bases*, pages 251–262, Mumbai (Bombay), India, September 1996. Morgan Kaufmann.

[80] The Library of Congress MARC Standards. MARC Formats. Technical document. http://lcweb.loc.gov/marc.

[81] C.C. Marshall. Making metadata: A study of metadata creation for a mixed physical-digital collection. In *3rd ACM Conference on Digital libraries (DL '98)*, pages 162–171, Pittsburgh, PA, USA, June 1998.

[82] P. Martin, W. Powley, and P. Zion. A metadata repository API. In *Proceedings of the 5th International Workshop on Knowledge Represenation Meets Databases (KRDB '98): Innovative Application Programming and Query Interfaces*, pages 13.1–13.8, Seattle, WA, May 1998.

[83] R. Mattison. *Data Warehouse: Strategies, Technologies and Techniques.* McGraw-Hill New York, 1996.

[84] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3):54–66, September 1997.

[85] MCI Systemhouse Corporation. *Relationship of the Unified Modeling Language to the Reference Model of Open Distributed Computing*, 1997. http://enterprise. Systemhouse.MCI.com/uml-odp.

[86] D. Meyer and C. Cannon. *Building a Better Data Warehouse.* Prentice Hall, 1998.

[87] Microsoft. *Open Information Model.* http://msdn.microsoft.com/repository/OIM.

[88] Microsoft. *XML Interchange Format (XIF).* http://msdn.microsoft.com/repository/.

[89] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362, October 1990.

[90] W. Nejdl and M. Wolpers. KBS Hyperbook - a data-driven information system on the web. Technical report, University of Hannover, KBS Institute, November 1998.

[91] J. Newton. Application of Metadata Standards. In [3].

[92] H.W. Nissen, M.A. Jeusfeld, M. Jarke, G.V. Zemanek, and H. Huber. Managing multiple requirements perspectives with metamodels. *IEEE Software*, pages 37–47, March 1996.

[93] NIST. *Integration Definition for Function Modeling (IDEF0)*, 1993. FIPS-183: http://www.sdct.itl.nist.gov/~ftp/idef/idef0.rtf.

[94] NIST. *Integration Definition for Information Modeling (IDEF1X)*, 1993. http://www.sdct.itl.nist.gov/~ftp/idef/idef1x.rtf.

[95] Object Management Group (OMG). *Meta Object Facility (MOF) Specification*, 1997. OMG Document ad/97-08-14 and ad/97-08-15.

[96] Object Management Group (OMG). *Common Warehouse Metadata Interchange - Request For Proposal*, 1998. OMG Document ad/98-09-02.

[97] Object Management Group (OMG). *Stream-based Model Interchange*, 1998. OMG document ad/98-10-05.

[98] D. O'Leary. Using AI in knowledge management: Knowledge bases and ontologies. *IEEE Intelligent Systems*, 13(3):34 – 39, May/June 1998.

[99] R. Orfali and D. Harkey. *Client/Server Programming with JAVA and CORBA.* John Wiley & Sons, 1997.

[100] F.J. Riggins and H.S.S. Rhee. Toward a unified view of electronic commerce. *Communications of the ACM*, 41(10):88 –95, October 1998.

[101] A. Rosenthal. Toward Unified Metadata for the Department of Defense. In [4].

[102] C. Sapia, M. Blaschka, G. Höfling, and B. Dinter. Extending the E/R model for the multidimensional paradigm. In *Advances in Database Technologies, Proc. International Workshop on Data Warehouse and Data Mining (DWDM)*, Singapore, 1998. Springer.

[103] P. Schäuble. *Multimedia Information Retrieval; Content-Based Information Retrieval from Large Text and Audio Databases.* Kluwer Academic Publishers, 1997.

[104] A.-W. Scheer. *ARIS - Business Process Frameworks.* Springer, 1998.

[105] A. Sheth and W Klas. *Multimedia Data Management; Using Metadata to Integrate and Apply Digital Media.* McGraw-Hill, 1998.

[106] M. Staudt, C. Quix, and M. Jeusfeld. View maintenance and change notification for application program views. In *Proc. ACM Symposium on Applied Computing (SAC'98)*, pages 220–225, Atlanta, Georgia, February 1998. ACM Press.

[107] T. Stöhr, R. Müller, and E. Rahm. An integrative and uniform model for metadata management in data warehousing environments. In *Proceedings of the Intl. Workshop on Design and Management of Data Warehouses (DMDW 99)*, pages 12.1–12.16, Heidelberg, Germany, June 1999.

[108] C. Szyperski. *Component Software; Beyond Object-Oriented Programming.* Addison-Wesley, 1998.

[109] The Meta Data Coalition. *Meta Data Interchange Specification (MDIS Version 1.1)*, 1997. http://www.MDCinfo.com/MDIS/MDIS11.html.

[110] The Meta Data Coalition. *Business Engineering Model*, July 1999. http://www.mdcinfo.com/OIM/models/BEM.html.

[111] The Meta Data Coalition. *Knowledge Management Model Knowledge Descriptions*, July 1999. http://www.mdcinfo.com/OIM/models/BEM.html.

[112] The OLAP Council. *The MDAPI specification*, 1998. http://www.olapcouncil.org/research/apily.htm.

[113] M. Tork Roth, M. Arya, L.M. Haas, M.J. Carey, W.F. Cody, R. Fagin, P.M. Schwarz, J. Thomas, and E.L. Wimmers. The Garlic project. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, page 557. ACM Press, 1996.

[114] B. von Buol and S. Kethers. A terminology server for quality-driven cooperative terminology work. In *Proceedings of the Seventh Annual Workshop on Information Technologies and Systems (WITS 97)*, pages 267–276, Atlanta, GA, December 1997.

[115] W3C. *Extensible Markup Language (XML)*, 1.0 edition, 1998. Recommendation 10-February-1998, http://www.w3.org/TR/1998/REC-xml-19980210.

[116] W3C. *Mathematical Markup Language (MathML)*, 1998. W3C Recommendation 07-April-1998, http://www.w3.org/TR/REC-MathML/.

[117] M. Wechsler and P. Schäuble. Metadata for content-based retrieval of speech recordings. In [105].

[118] S.L Weibel and C. Lagoze. An element set to support resource discovery; the state of the Dublin Core: January 97. *Intl. Journal on Digital Libraries*, 1(2):176 –186, 1997.

[119] C. White. Managing distributed data warehouse meta data. http://www.dmreview.com/issues/1999/feb/articles/feb99_46.htm.

[120] M.C. Wu and P. Buchmann. Research issues in data warehousing. In *Datenbanksysteme in Büro, Technik und Wissenschaft (BTW '97)*, pages 61 –82, Ulm, Germany, 1997.

[121] John A. Zachman. A framework for information systems architecture. *IBM Systems Journal*, 26(3), 1987.

[122] G. Zhou, R. Hull, and R. King. Generating data integration mediators that use materialization. *Journal of Intelligent Information Systems*, 6(2/3):199–221, May 1996.

[123] V. Zwass. Electronic commerce: Structures and issues. *Intl. Journal of Electronic Commerce*, 1(1):3 –23, Fall 1996.