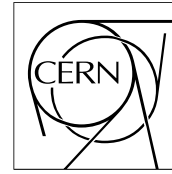




The Compact Muon Solenoid Experiment

CMS Note

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland



February, 2000

Seamless Integration of Information

Frank van Lingen

CERN, Geneva, Switzerland

Email: Frank.van.Lingen@cern.ch

Abstract

This document gives an overview of issues related to integration of information. Different aspects are discussed. Furthermore, several projects related to integration of information are described. Also several standards that relate to information integration are discussed. Finally, these issues are related to information integration within a high-energy physics environment.

Keywords: heterogeneous databases, multidatabases, temporal databases, meta- model, meta- data, ontologies, mediator, data management.

1	INTRODUCTION	5
2	HETEROGENOUS DATABASES	6
2.1	<i>WRAPPER</i>	7
2.2	<i>MEDIATOR</i>	7
2.3	<i>ONTOLOGY</i>	8
3	META- DATA AND META- MODELING	9
4	TIME	11
4.1	CHANGE OF A MODEL THROUGH TIME	11
4.2	VERSIONING	12
4.3	TIME MODELING WITHIN DATABASES	12
4.4	SCHEMA EVOLUTION	13
4.5	ACTIVE DATABASES	13
5	SELECTED PAPERS	14
5.1	KERHERVÉ & GERBÉ 1997	14
5.2	KUTSCHE & SÜNBÜL 1999	15
5.3	SERGE ABITEBOUL 1999	18
5.4	JENNIFER WIDOM & ALON LEVY 1999	19
5.5	RAM 1999	20
6	PROJECTS	21
6.1	MIX	21
6.2	TSIMMIS	21
6.3	INFOLEUTH	22
6.4	HERMES	22
6.5	DAISY	23
6.6	AMASE	23
7	COMMERCIAL PRODUCTS	23
7.1	ARDENT	23
7.2	MERLIN	23
7.3	META- INTEGRATION WORKS (MIW)	24
7.4	ETI	24
8	STANDARDS	25
8.1	ODMG	25
8.2	OMG	26
8.3	MDC	26
8.4	STEP	26
8.5	EXPRESS	27
8.6	XML	27
8.7	META- DATA INTERCHANGE SPECIFICATION	29
8.8	MOF	29
8.9	OIM	30
8.10	DOM	30
9	FUTURE WORK	30
10	CONCLUSION	31
10.1	DOMAIN SPECIFICATION	31
10.2	COMMON EXCHANGE FORMAT	31
10.3	HIGH-LEVEL QUERY LANGUAGE	31
10.4	BUILDING BLOCKS FOR VIEWS	31

LIST OF FIGURES	34
GLOSSARY	35
REFERENCES	37

1 Introduction

Scientific data sets are getting not only larger and larger but the complexity is increasing, meaning that the extraction of meaningful knowledge requires more and more computing resources. Furthermore, scientific collaborations are getting larger and more geographically dispersed. This leads to need of catalogues and indexes of data archives and the ability to select and integrate data objects, to define complex processing. [Wil99] Identified several areas that are important:

- Information modelling
- Standard scientific data objects/models. XML is seen as a candidate for this. Using XML one can build schemas that describe certain data collection using these schemas information can be retrieved from different data collections.
- Database interoperability. Wrappers can be used for this but can be too limited. Large scale data manipulation requires the tight integration of data and computing sources.

Areas that deal with large amounts data stored in different repositories are:

- Geographic information systems. Different repositories contain data of different aspects of a landscape. (see [Wari98], [Gob99])
- Biology. Data about RNA and DNA strings is stored in different databases. (see [Crite98])
- Libraries. (see [Birm98])
- Astrophysics (see [Cheu95])
- Gravitational wave observatory (see [Wil99])

As discussed earlier we see that data integration is an issue of different scientific areas. What do these areas have in common? They deal with large amounts of data. Data is not located in one database but is distributed over many data sources. To analyze this data, often data from different sources and different formats needs to be correlated. Applications need to use the same data in order to have consistent and reliable outputs. This data can be located in different data sources. To achieve this, methods, models, and middleware is developed for seamless integration of this data. If we look at these characteristics we see that they also apply to the high energy physics environment (see [Wil99]). With the upcoming LHC experiment the amount of data will increase even more. It is only logical to look at ways to deal with this data as efficiently as possible. Thus looking at the problem of how to integrate these different data sources. Why do you want to integrate it on a high level? Low level integration means building dedicated data bridges between every two specific sources. If you have n data sources you will need $n(n+1)/2$ data bridges. And if you add a new data source to this collection you will have to build n data bridges to integrate this data source with the other data sources. A high-level solution prevents this from happening.

In the CMS experiment many different data repositories are used. To ensure that these data repositories can inter-operate and can provide convenient modes of access for physicists, it is necessary to develop suitable underlying data models which are sufficiently flexible to address the needs of physicists. The inter-operation of, and data exchange between, multiple data repositories can be facilitated by the use of some common repository description. Typically, the movement of information from one repository (e.g detector construction) to another (eg. calibration) for use by physicists would be significantly eased if it was possible to access the respective repositories through a descriptive detector database (or *meta-model*). This sharing of information is also required between the construction database and large physics simulation programs (eg GEANT4) for accurate detector simulation. A common meta-model used in an HEP experiment would act as a road-atlas to the underlying data model and would allow the provision of query facilities to enable cooperation and collaboration between, otherwise distinct, 'islands of information' in the experiment.

This meta-model would lead to a seamless integration of information. Users/applications would only access the "virtual database" (the meta-model and the data sources connected to it). This would lead to a couple of benefits with respect to the CMS experiment:

- To use CAD models in the simulation program for the purpose of validating the geometry.
- To allow the reconstruction program to use the simulation geometry.
- To allow the CAD models to include the information measures and information used for reconstruction purposes.

- To exploit the advantages of a unified CMS detector description to serve different CMS offline applications via different re-presentations.
- To allow construction databases to retrieve information about specific detector parts as designed by the engineers.

But most importantly it would ensure consistency of numerical data in the whole CMS experiment. This benefit enables applications and construction databases to use the same up-to-date data and draw conclusions from it that relate to each other and to the CMS detector as a whole.

The problem discussed above relates to several research areas of the database community:

- Heterogenous databases
- Mediators and wrappers
- Ontologies
- Meta- data (modeling)
- Temporal databases

This article will give an overview of the state-of-the-art in the research areas mentioned earlier. In each of these areas we discuss several articles and projects that reflects the current view in this research area. Furthermore, we will discuss some projects that relate to the problem mentioned above and discuss some standards that are currently used. Finally, we discuss how these aspects can be used to tackle this problem

The main idea of this paper is to give an overview of the current state of the art in data integration. This work is by far not a complete summary of all the research in the field of data integration. It should serve as a primary trigger for ongoing Ph.D. work to get a deep and profound understanding of data integration. Furthermore, some topics are only touched briefly and so need more attention at the later stages of the Ph.D. program. However, raising questions is another essential aim.

2 Heterogenous databases

Heterogeneous database systems or multidatabases provide integrated access to multiple databases. Generally this is done in two steps:

- Accept a query, determine the appropriate sets of information sources to answer the query, and generate the appropriate sub queries or commands for each information source.
- Obtain results from the information sources, perform appropriate translation, filtering, and merging on the information, and return the final answer to the user or application.

This is usually referred to as the lazy approach (see [Widom95]). In an eager approach:

- Information from the source that may be of interest is extracted in advance, translated and filtered as appropriate, merged with relevant information from other sources, and stored in a centralised repository.
- When a query is posed, the query is evaluated directly at the repository, without accessing the original information sources.

This eager approach is usually referred to as data warehousing (see [Widom95]). A heterogeneous system consists usually of the following subsystems (Figure 1)

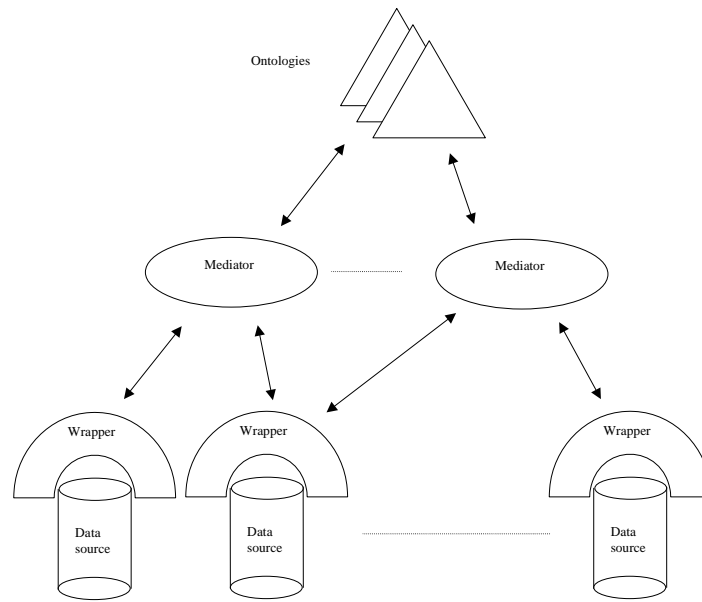


Figure 1. Subsystems of heterogeneous databases.

A wrapper is used to translate the database data into a common format. A mediator divides queries of users into sub queries for databases. Furthermore, it integrates results from sub queries into one uniform result for the user. An ontology enables users to access the databases using concepts of the real world. This architecture is used in many articles about heterogeneous database systems (see: [Kuts99], [Kash95], [Wie99], [Birm98]). The mediator architecture was first proposed by [Wie92] and is used as a reference architecture in many applications for data integration (see [Sub99], [Bar99], [Gar97]). The next sections will discuss the subsystems in more detail.

With respect to data integration there is a new technique that could aid in this process: XML (see [Bos99], [Ang99]). In [Levy99] and [Widom99] it is argued that XML has the potential to be used for integrating different data sources. Some Research directions with respect to XML and databases are discussed in [Widom99] (see section 5.4).

2.1 Wrapper

A wrapper can be used to "wrap" the source and make it available to a mediator. Wrappers handle query processing on individual sources while mediators perform queries on multiple sources.

Data in different data sources can have different Structures. Data can be stored in object-oriented databases, relational databases, or even web pages. Data can be structured (eg. in databases) or semi structured. "Semi structured data is data that has some structure, but maybe irregular and incomplete, and does not necessarily conform to a fixed schema (eg. HTML documents)" (see [Chaw97]).

2.2 Mediator

A mediator is an intermediate between the data sources and the applications/users or between different data sources. It has knowledge about where data is located, and decouples queries from applications/users, into sub queries suited for different databases. Sometimes people also speak about a mediator architecture (see [Wie99] or [Kuts99]). This refers to the architecture described in Figure 1. [Wie92] uses the following definition: "mediators are software agents which act as translators for data, encapsulating all the routine work of converting data from one format to another." [Gar97] gives the following requirements for mediators:

- There must be a common data model that is more flexible than the models commonly used for database management systems. A mediator model must support:
 - A rich collection of structures, including nested structures as is found in the type system of typical modern programming languages.

- Graceful handling of missing information or related information of widely different structures.
- Meta- information, that is, information about structures themselves and about the meaning of the terms used in the data.
- There must be a common query language to allow:
 - New mediators to join old ones for augmented functionality.
 - New sources to provide into a to an existing mediator.
- There must be tools to make the creation of new mediators and mediator systems easier than would be the case if everything was built from scratch.

Although a mediator is a general concept, it also contains a domain specific part. This domain specific part is related to the fact that a mediator has to decompose queries and combine query results. To do this efficiently a mediator needs the context (a domain) that is related to the data sources.

2.3 *Ontology*

In databases (or database systems) there are many names/structures to store certain information. Users usually have no knowledge of these technical details; they want to access the data using "world" concepts. Thus what is needed is a mapping of these concepts to the data related to it. This is an ontology or viewpoint. There are several definitions of the term ontology. We will discuss the definition that is closely related to software engineering and is also discussed in [Bez98] and [Kayser98]. The following definition is used: An ontology is an explicit and precise description of concepts and relations that exist in a particular domain such as a given organization, a study field, an application area, etc. The main properties of an ontology are sharing and filtering. Sharing means that an agreement may exist between different agents based on the acceptance of common ontologies, that they have the same understanding of a given concept. Filtering is linked to abstraction. Consider models of reality. These models, by definition, take into account only a part of the reality. Their usefulness is based on their ability to filter out a lot of undesirable characteristics. An ontology defines what should be extracted from a system in order to build a given model of this system. A model is always built for a given purpose, usually of understanding some aspects of the source system. This purpose should be clearly defined and associated with the ontology. According to [Bez98] this ontology definition corresponds closely to the classical definition of a meta- model as it is used in [OMG97]. An ontology contains the concepts and the relations that are relevant to a given modeling task. [Bez98] makes a distinction between three levels of information within an ontology:

1. Terminological. This is the basic set of concepts and relations constituting the ontology. It is sometimes called the definition layer of the ontology.
2. Assertional. This part is sometimes called the axioms layer of an ontology. It is a set of assertions applying to the basic concepts and relations.
3. Pragmatical. This is the so-called toolbox layer. It contains a lot of pragmatical information that could not fit in 1) or 2). For example, if there is a concept of class defined in 1), then 3) could contain the way to draw this concept on screen or paper, as a rounded square for example, numerous number of pragmatical data may be found in an ontology, for example the way to serialize or to pretty print the different concepts and relations of the ontology or an API expressed in IDL. It is of importance that this pragmatic information could be kept packaged and close to the basic concepts it is related to.

A database system can have different ontologies (see [Kash95]). Several ontologies can aid the user in looking for the same thing. This can result in ontology mismatches. [Pep97] discusses these mismatches. The mismatches are divided into two main categories: conceptualisation mismatch and explicitation mismatch. These categories are again subdivided into categories for each mismatch. The authors try to give the solution or transformation for these mismatches. [Kash95] stresses the importance of an ontology as follows: "the key to utilising the knowledge of application domain is identifying the basic vocabulary, consisting of terms (or concepts) of interest to a typical user in the application domain." Furthermore, [Kash95] sees the problem of mismatches. The following is said about it: "there have to be agreements on the terms used by the different designers. These agreements can be the basis of the construction of the ontology and are called ontological commitments." The term ontology is not always related to databases. [Bez98] argues there will be a change from object-oriented programming to ontological driven modeling.

The mediator uses a domain description to efficiently retrieve and combine data. An ontology describes a

domain using concepts. It should be noted however that an ontology is not only a mechanism to be able to communicate between users and applications. But that it can also be used as a mechanism to communicate between applications. This can be very useful in agent technology.

3 Meta- data and Meta- modeling

Meta- data is data about data. Because of the complexity and size of data it becomes important to have data that gives information about data. Consider for example the Internet. If you want to find an author called John Doe you do not want to get all John Does. It would be nice if the pages had meta- tag "author" and the search engine would look for that. Unfortunately this is not the case. Many pages are still in plane text. But this might change in the future with the upcoming of XML (see chapter 8). XML has the ability to define a meta- data context around the data.

Meta- data is not only important for the Internet, but for large data sources in general. Meta- data can be utilized in designing "flexible databases". If you have the proper meta- objects to store data you do not need to recompile or redesign your database, whenever the content of the database changes. Besides flexibility, meta- objects prevent an class explosion in your database. Instead, you have a small set of meta- objects. This was one of the goals of the CRISTAL project (see [Dras99], [Goff99], [Goff99+2], [Estr98])

One has to be careful with respect to defining meta- objects. Are the meta- objects "meta-" enough? Can it handle all the content changes of the database? One can solve this problem by defining meta- meta- objects. These objects define meta- objects. However, the same problem occurs on the meta- meta- level.

[Kash95] identifies 3 different types of meta- data:

- Content dependent meta-data.
- Content descriptive meta-data.
- Content independent meta-data.

Besides meta-data, one can also speak about meta- models (models about models). Examples of such models are UML and CDIF. Meta- models can also be used in database design. First a meta- model is designed encapsulating all the important aspects of the data of a certain company/project. Using this meta- model, database models can be designed as an "instance" of the meta- model, representing a certain view or projection of the data. The meta- model would also enable users and applications to access data stored in the different databases, creating a uniform and consistent view. [Dras99] discusses this problem with respect to high-energy physics.

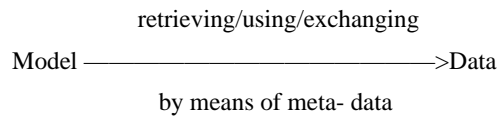
There are several reasons to use a meta- model:

- Better management of data. If you want to store large amounts of data you can build a model to store it in. However this can lead to a "class explosion" (in an object-oriented database). To manage this one can define a meta- model. Depending on your data it is even possible to define a meta- meta- model.
- In time, data stored in the database changes. Furthermore, the structure of data can change, and even new kinds of data can emerge. A meta- model enables generality, thus coping with his change.
- Besides to be flexible in time and space another requirement can be to exchange data with other applications. However these applications can use different models for this data. Therefore it can be desirable to have certain levels of meta- models.

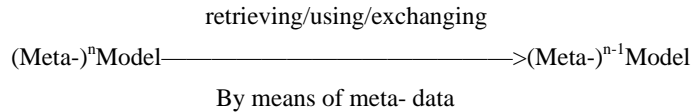
Another aspect of "flexible" databases is that data should be independent of the applications that use it. A consequence of this is that data should be self describing. This can be accomplished using meta- data. Using this approach, databases become more general, and because of self-description can be used by different applications more easily. [Foot98] discusses several patterns that can be used for this so-called "meta-programming". [Kova99] also discusses some patterns for this. These patterns are enriched design patterns based on the patterns discussed in [Gamma95].

[Kerv97] and [Kerv98] discuss this problem in a more general context and also discuss the need for a meta- data management system. Meta- data management should incorporate basic operations and corresponding tools, to extend and adapt core meta- models. Core meta- models are used to "instantiate" meta- models for specific areas of interests (eg. GIS, Digital libraries, HEP)

The relation between meta- data, model, and data is that meta-data enable users/applications to search for certain data in an efficient manner. It is a "road map" for the model to locate data. This leads to the following recursive relation:



One can generalize this idea to a higher level:



Using this observation, one can construct the following architecture (see Figure 2). A database contains the model that is connected by meta- data to the data. In a relational database this model consists of the tables defined in this database. The meta- data are the fields, types and tables. Using this idea one can construct the meta- database. The "model" of the meta- database is the meta- model and the "data" are the databases. These are related to each other by means of meta- data. In a single database a view is usually a new set of data generated by means of a query. In a meta- database however one can see a view as a new database generated by a set of queries. The meta- model can be viewed as a mediator (see [Wie92]). The following question now arises: how do you know if this meta- model is sufficient enough?

OMG defines 4 levels of (meta-) data (models) in the MOF [OMG99]. When metamodels and meta- data are discussed it is mostly related to the four levels described in the MOF. However, are four levels enough? Why not five levels? Or six? The first level is the data level. To store this data in an efficient manner one needs a model (data model in a database). However, you do not know if this model is rich enough. By defining a metamodel one is able to extend the model in a natural way. If data needs to be exchanged between different models one needs a high-level description of how this data will be exchanged. To connect these models that both have a metamodel one needs a meta meta model. This meta- metamodel enables the exchange of this data. However, how do you know if this meta- metamodel is rich enough? One defines a meta- meta- metamodel. This is a fifth level. But is never discussed. One reason for this can be that we are just starting to discover what meta- models are. At the moment we are at level 3 (metamodel). Using meta- modelling one can solve certain problems on the lower level. However, certain problems will appear again on a higher level. For example, it was thought that a metamodel would solve the problems related to standardisation. This is partially true. It can build bridges between different "data islands". What happens is that different clusters of these islands emerge (GIS, biology, astronomy, high energy physics). All having their own standards. In the future data will become more complex. To be able to obtain an overview of the data and to enable exchange of data one has to ask: do we need another metalevel? Maybe four levels will not be enough.

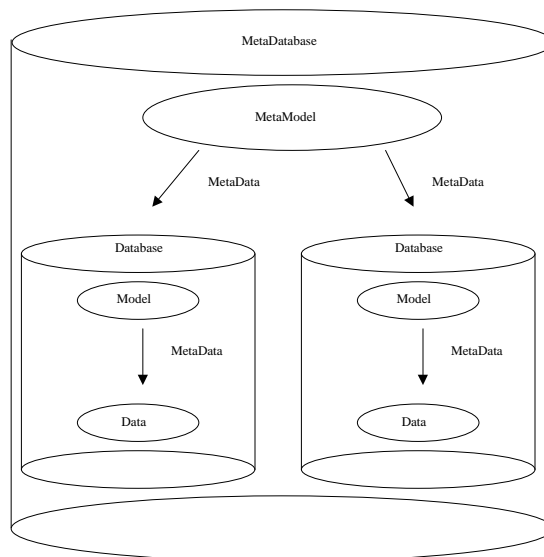


Figure 2. Meta-Database architecture.

4 Time

Time is an important aspect within database models. Lots of data that is stored in databases is related to time. For example geographic information, medical information (a medical history of a patient) or financial information. Data in the CMS detector databases can also change over time. If these data sources are connected to a meta- model then this meta- model should also be able to cope with time and be able to change accordingly. Another aspect is the integration of data that is related to time (so-called temporal data integration). In literature there is not much said about the subject. It is mostly related to integrating sensor data. However, if we look at geographic information systems then data integration is closely related to time. But this is not mentioned explicitly. This can mean that temporal data integration within geographic information systems is not a difficult problem. If we look at how time is attached to geographical data than it is usually a time stamp. If we look at commercial geographic information systems like Arcview we see that time is not explicitly supported. Time can be modeled by defining an attribute time. Time within a database or database systems can relate to several issues:

- Change of a model through time
- Versioning
- Time modeling within databases
- Schema evolution
- Active databases

4.1 Change of a model through time

The metamodel should also be able to cope with change. It cannot be foreseen how the databases that are connected will evolve in the future. Furthermore, new databases can be connected to the metamodel. The metamodel should be able to cope with this. This leads to a couple of requirements/wishes with respect to the metamodel:

- The metamodel should be able to assimilate new models and change itself accordingly.
- The metamodel should be general enough. It should encapsulate the models of the different data sources that are connected to it.
- In order to cope with change it should be able to "learn" from the schema of the connected databases.
- It is not possible to determine if the metamodel is "meta" enough. A solution to that can be to define a meta meta model.
- A metamodel should cope with change because it is meta enough. However, if it is not rich enough change of the metamodel should be automated by using certain rules. This refers to an active data model and a mechanism to define rules.
- The change referred to in the previous item can be on the model level or on the schema level. This low level change refers to schema evolution.

Coping with change in models is referred to as "dynamic object models" or "active object models". Other names are: reflective architecture or meta architecture (see [John98]). An active object model is an object model that provides "meta" information about itself so that it can be changed at run time. According to [Foot98] active object models arise as domain specific frameworks evolve to address an ever widening range of domain specific needs. If you change the object model, the system changes its behaviour. For example, workflow systems can have an active object model. Active object models can be very useful in business systems (see [Riehle98], [Witt98] and [Nak98]). The systems are deployed in an environment that changes quickly. [Foot98] discusses some issues that arise when building active object model are:

- Both systems and users must adapt quickly to changing requirements.
- Building dynamic object is hard. This is also mentioned in [John98].
- Once built, dynamic objects allow for rapid alterations to your program.
- Changing a program to meet new requirements is usually slow and complicated.

- Active object models have the ability to change on the fly and can be difficult to develop, hard to understand, and hard to maintain.

An advantage of active object models is the ability to cope with change. Another advantage is that it can reduce the lines of codes needed to build a certain application. Some disadvantages according to [John98] are: it is difficult to build. Furthermore, a system based on a dynamic object model can be slow. Related to active object models are several patterns (see [John98] and [Foot98]).

4.2 Versioning

Object versioning is a capability that permits an application to create separate versions of individual objects, or in some cases also versions of collections of objects. Versioning can occur on different levels. First you have versioning on the data level but you can also have versioning on higher levels (model, meta- model, etc.) because there is the relation between data, model, meta- model, (and maybe higher levels) versioning can be complex task. What is the impact of generating a new version on a certain level with respect to the lower levels? Furthermore if you have a set of objects in a certain level how to deal with versioning when the version of the different objects is not synchronous (not the same version at the same time). [Kova99] discusses this problem and introduces a version pattern.

4.3 Time modeling within databases

Time modeling in databases is usually referred to as temporal databases. A temporal database supports some aspects of time and supports reasoning about time. Consider the following scenario: you want to correlate event data with the geometry of the detector at the moment this event data was generated. This means you have to retrieve geometry data valid at a certain moment in time. Furthermore, during construction, components of the detector are assembled. These components are registered in a production database management system. These components contain timestamps. For example, it can contain a history of when it was produced, when it was shipped, when it was tested etc. A lot of research in temporal databases has been done in the relational field. Resulting in a lot of extensions to the entity relation model with the concept of time. These extensions also lead to an extension of SQL (TSQL). [Jens97] gives an introduction into this field.

Time can be represented in different ways (see [Snod92] and [Snod85]). The most general model of time in a temporal logic represents time as a partial ordered set. Linear time can be seen as an ordered set. Recurrent events can be modeled using a cycle model of time. In the context of databases two time dimensions are of general interest: valid time and transaction time. A database that supports both is called a bitemporal database. There is also another time concept called decision time. It is the time that a decision occurred. Besides adding time to information one can also have information that is temporarily indeterminate ("do not know exactly when"). Another "time" can be the life span of objects or collection of objects. It refers to the time that objects exist in the model.

In real-time databases valid time is used for data items that have immediate counterparts in the real (physical) world. Transaction time in real-time databases is used by transactions to set parameters of a real-time system. Furthermore, these transactions used to have a time constraint (deadline). According to [Snod95] research in temporal databases and real-time databases have proceeded independently with little cross fertilization.[Snod95] also argues that real-time databases can benefit from the temporal models designed in the temporal database research community.

With the concept of time one can also introduce different granularities of time (day, month, year). Besides different granularities one can also make a distinction between different temporal domains: instant (a point in time), interval, a period. [Ste97] says that is also possible to model temporal properties using data/time attributes (in object-oriented modeling) instead of introducing a more formal concept of time within object-oriented modeling. Constraint checking needs to be enhanced in databases when adding the time dimension to data. Time enables the generation of new constraints. [Ste97] discusses an extension of the object model with time. This TOM (temporal object model) adds timestamps to the names of instances. It is therefore not an extension of types but an extension of object ID. An object ID is extended with a time stamp, thus creating a temporal object ID. Besides attaching time stamps to objects is also possible to associate time stamps to relations. In an object model a relation consists of two object IDs. In the temporal object model a time stamp can be attached to these relations.

[Gor97] discusses the fact that there are several temporal models. Most of these models were designed to support the temporal needs of the particular application, or group of similar applications. However, on the abstract level there are similarities in the temporal features:

- Each temporal model has one or more temporal primitives, namely, time instant, time interval, time spent, etc.
- Some temporal models require their temporal primitives to have the same underlying granularity, while others support multiple granularities and allow temporal primitives to be specified in different granularities.
- Most temporal models support linear model of time, while some support a branching model of time.
- All temporal models provide some means of modeling historical information about real world entities and/or histories of entities in the database.

These common entities suggest a need for combining the diverse features of the temporal domain using a single infrastructure. According to [Gor97] a temporal model can be classified along four design dimensions:

- Temporal structure
- Temporarily representation
- Temporal order
- Temporal history

A temporal structure provides an underlying ontology and domain for time. In [Gor97] this is a hierarchical structure (a tree). Different concepts of time are identified: interval, point, discrete time and continuous time. The temporal re-presentation refers to a calendar. The temporal order defines an order for time. This order can be linear, sub linear or can have branches. In the temporal history different histories are identified with respect to time. The four temporal design dimensions have a "has a" relationship with each other. A temporal history "has a" temporal order "has a" temporary representation "has a" temporal structure.

The diversity of time extensions for data models suggest that it is difficult to incorporate them into these models and that there are many possibilities. What time do you need? Valid time (time points, time intervals), transaction time, lifespan, or even any other time?

4.4 Schema evolution

The term schema evolution refers to the changes undergone by a database's schema during the course of the database's existence. It refers especially to schema changes that potentially require changing the representations of objects already stored in the database. In the ODMG standard, schema is defined using the Object Definition Language (ODL), which is based upon the Interface Definition Language (IDL) of the Object Management Group (OMG)

In addition to changes to the database schema, the affected objects, i.e. the persistent instantiations of the schema, may be converted in a number of ways, eg.

- Immediate. All affected objects within a federation are changed using a special upgrade application,.
- Deferred. Affected objects are changed only when accessed. This has the advantage that only those objects that are used are changed, but does mean that access times can be unpredictable during the conversion period.
- On-demand. Affected objects in specific containers, databases, or within the entire federation are updated upon user request.

4.5 Active databases

Another requirement on data can be that it has to be consistent with the environment (see [Erik98]). This means that if the environment changes data and/or views must change accordingly. A traditional database is not suited for this. Active database however can cope with these requirements. Using events/rules that are evaluated by conditions that can execute certain actions it is possible to update data and views. It is not always possible to determine all rules/events in advance. Therefore, it is possible to use dynamic rules. The rules can be adapted without recompiling the system. Consider the following scenario: A set of queries that is deployed on different databases to generate a new database that will be used in a simulation. This database can be seen as a view on the data of the databases it extracted the data from. However, this data can change in time. In a simulation you want the most up-to-date data without having to look for it every time. The solution to this can be a set of rules that will update the view if data changes.

5 Selected papers

These papers discuss specific parts in the area of data integration. The papers are chosen such that they complement each other. [Kerh97] discusses the need for a meta- data management on different meta- levels. [Kuts99] elaborates on a general structure for large scale heterogeneous information systems and tries to identify what is needed to build subsystems. [Abit99] discusses how views are related to XML, while Jenifer Widom and Alon Levy describe future research directions for XML, databases and data integration. [Ram99] describes a project that uses a semantic model and deals with temporal data.

5.1 Kerhervé & Gerbé 1997

[Kerh97] examines modeling and meta- modeling for meta- data. Already different application areas use meta- data specific to their domain. According to [Kerh97] this individual meta- data specification is not sufficient any more, and research should go in the direction of integration, federation, and inter-operation of existing propositions and standards. This includes issues as:

- Extensible meta- data models that can be adapted to specific application domains.
- Tools allowing integration and interoperation of meta- data sets coming from different sources and representing different standards.
- Extensible meta- data managers offering basic services to support essential functions such as access, transfer, discovery or analysis for the development of specific applications.

Among different meta- data sets proposed for applications such as GIS, and distributed multimedia systems, a common subset of meta- data can be identified. Defining a core model for this meta- data is a way to describe this subset. If this model should be extendible, one should also build a meta- model for meta- data. The open environment provided by the Internet and the growing need to share information requires a focus on interoperability between these data models, systems and tools. [Kerh97] suggests an approach for a meta- data management allowing the integration as well as the extension of existing proposals. Associated to this meta- core model should be defined a set of the basic operations as well as the corresponding tools allowing to extend and adapt the core meta- data model. The tools for extension and adaptation lead to specify extensible meta- data managers. According to [Kerh97] designing extensible meta- data managers requires to address modeling and meta- modeling issues in order to provide interoperability among existing meta- data sets as well as extensibility to existing propositions. Meta- data management has been identified as an important issue while designing and implementing applications using large volumes or complex data. Nevertheless, meta- data management has also been introduced in different management systems such as database systems, distributed systems or knowledge management systems without explicitly referring to the concept of meta- data. But it is important to have a coherent view on these different levels. According to [Kerh97] such systems should be revisited with a meta- data management perspective in order to be able to address fundamental issues for meta- data managers with the experience gained in the development of these systems. [Kerh97] discusses several examples in which meta- data management systems are identified:

- DBMS in relational databases. As a homogeneous way of defining and manipulating databases, relational DBMS use the concept of meta- database that is, a database describing other databases. A meta- database is itself defined as a set of tables.
- Corporate memory management systems. It is used to store projects. Four levels of information were defined:
 - Level one is the project data level. It contains all the objects generated during the project and constitutes the project database. At this level meta- data describes characteristics of the project objects himself such as document or code size, or history of activities.
 - Project objects are defined according to a model defined in the second level and called the project model level. The project model can be considered as the project database schema and is composed of several concepts. Concepts of the project model depend on the type of project and management in the system.
 - Projects are conducted according to methods. This is the content of levels 3 of the meta- model level. The meta- model defines concepts that are used in the method.
 - Level 4 is the knowledge model. It is a description of knowledge objects. It is composed of concepts, conceptual relations and grounds that structure concepts and conceptual relations.

Using examples [Kerh97] formulates several requirements for meta- data modeling. The examples showed various meta- data on different levels. To manage data on different levels extensible meta- data managers should support:

- Several modeling levels.
- Homogeneous manipulation of these different levels.
- Extensibility of the corresponding models and meta- models.

The kernel of meta- data managers should then use a representation formalism allowing this extension, adaptation and interoperability. A root modeling level defined as the knowledge level should be introduced to support interoperability and extensibility. Based on these requirements [Kerh97] defines a modeling architecture. Figure 3 shows this architecture. This closely resembles the four levels described in [OMG99] with the addition that every level has its meta- data that connects two levels. The meta- data modules act as a "road atlas" for the consecutive levels. Conceptual graphs were chosen as the technique to support this architecture. Conceptual graphs are discussed in [Sow84].

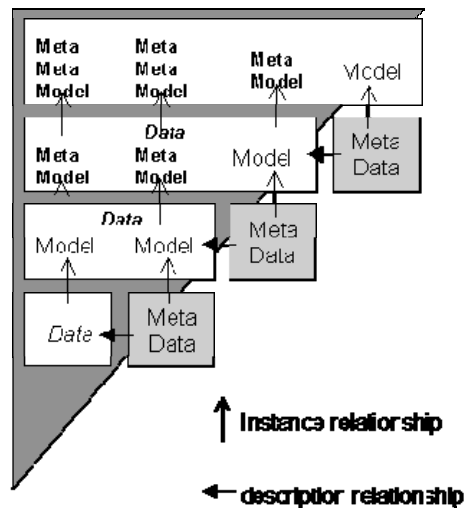


Figure 3. Four level Modeling Architecture.

5.2 Kutsche & Sünbül 1999

[Kuts99] discusses several characteristics for large-scale Information Systems:

- Being widely distributed over networks, they need 'transparent' means of communication / interoperation by appropriate middleware.
- Being strongly heterogeneous in their technical and organizational context, in type, structure and logical dependencies of their content, they require the capability of handling syntactical and semantic heterogeneity in this process of integration and federation of many different, but logically related information resources.
- Being subject to continuous change and evolution, they need capabilities of integrating legacy data sources and systems in an efficient and modular way (plug-in, plug-out).

[Kuts99] suggest the following methodology based on three principles:

- (object-oriented) information modeling on the bottom and top layers in parallel.
- Making explicit use of meta- information as a basic concept.
- Componentization in a software-architectural sense.

Whereas the communication and interoperation problem has come into a stage of ripened and applicable solutions over the past decade, given by protocols, standards and software as by object-oriented middleware and the Internet technology, neither semantic data integration /federation techniques nor the software development process itself for such complex information infrastructures have become similarly clear and well-established frame work.

[Kuts99] tries to identify important features of meta- data as a general concept beyond particular application domains: the support of evolution and flexibility of large distributed, heterogeneous information systems. First a general architecture is discussed. This architecture is composed of data sources, wrappers, mediation services, and a presentation layer (see Figure 4). Note that this architecture closely resembles the architecture displayed in Figure 1. This architecture is discussed in several other articles (see chapter 2).

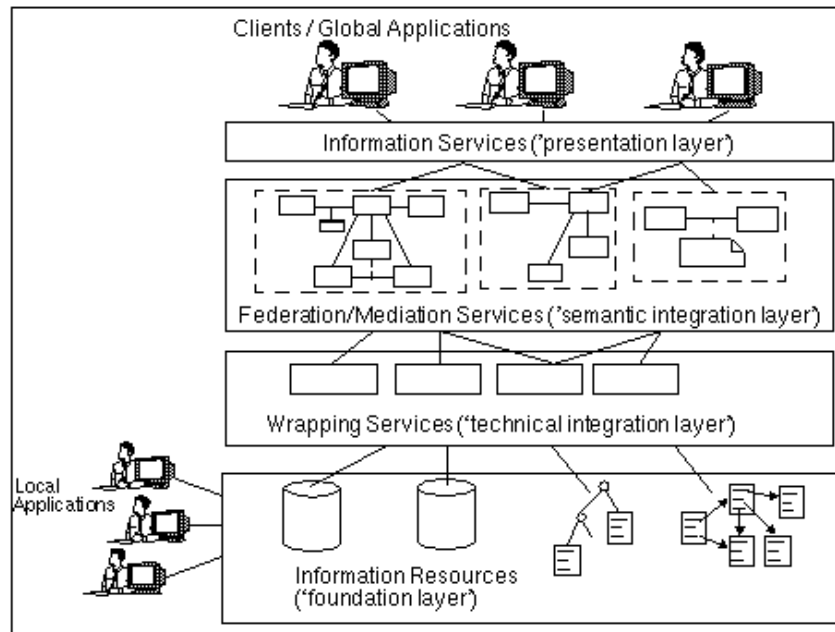


Figure 4. General architecture

[Kuts99] calls it a "reference architecture". [Kuts99] restricts itself to mediation aspects of semantic integration. According to [Kuts99] an important aspect of heterogeneous information systems is evolution. There are two reasons for this that are related to the structure displayed in Figure 4:

- At the bottom level, there will be a continuous change by newly developed, modified or additionally offered information sources.
- From the top level, the desire for new information services will appear as soon as the global value of the given information structure is recognized by a relevant number of users, and an idea of possibly new sources exists.

This gives rise to the following software engineering requirements:

- Integrating a new source into the global system by analyzing its kind, structure and content and making it technically available to the global view by appropriate wrappers.
- Specifying a new service based on the knowledge of the information need on-top, the knowledge of the available contents at bottom, and relating these to each other, which essentially means the modification of mediators or construction of new ones.
- Implicitly, this might lead to the requirement of new architectural components in the integration layers, eg. by recognizing more general patterns of wrapping or mediation and therefore restructuring the 'middleware'.

The software engineering requirements and observations made above are used in the information and modeling phase. During the information analysis and modeling phase different abstraction levels can be identified:

- The domain of interest in general (using/building ontologies and classifications as well as structural and behavioral principles of the application area and context given),
- The corresponding foundation layer (in general, by reverse information modeling of the relevant parts of the underlying data resources and legacy systems)
- An analysis of the current (and possibly future) information needs in this setting (by modeling use cases and scenarios, and their relation to the foundation layer).

During this phase meta- data and meta- models are identified. Furthermore, it is important to look if it's possible to integrate models of existing meta- information systems. [Kuts99] identifies three general categories of meta- information: domain dependent, the domain independent, and model correspondence information. Domain dependent relates to the meta- information in classical Information Systems (domain specific information model). Domain independent meta- information is related to the domain specific meta- model. The latter domain is a more general domain. Model correspondence information is meta- data that connects different models on different meta- levels. Figure 5 shows the different models and meta- data.

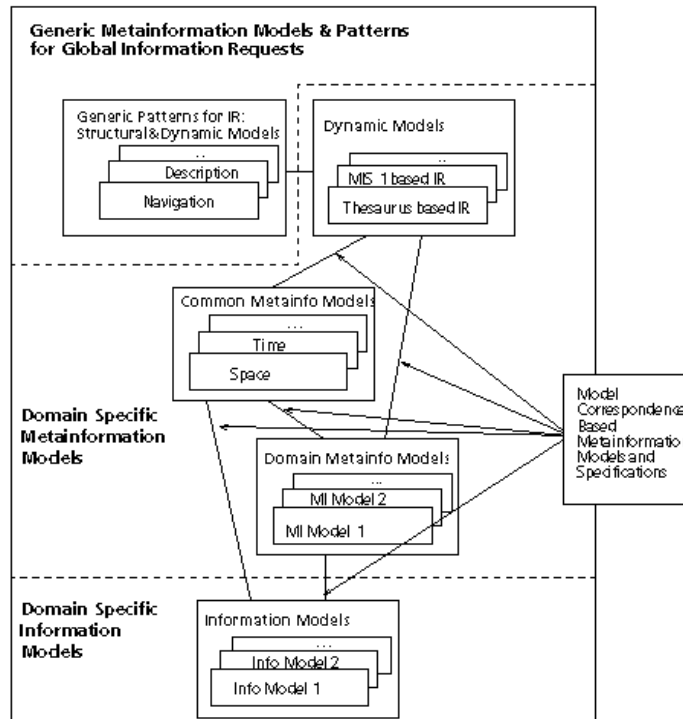


Figure 5

The second phase is to look at semantic integration. According to [Kuts99] the modeling techniques as provided by UML are not satisfactory. Therefore they have to be complemented by more precise specification texts, which means an annotation of more formal specification parts to the mainly graphical models. [Kuts99] identifies several meta- information components that can be used in a mediator:

- Information models
- Meta- information models
- Model Correspondences
- Dependencies
- Invariants

Figure 6 displays the different Meta- information components in connection with the "reference architecture"

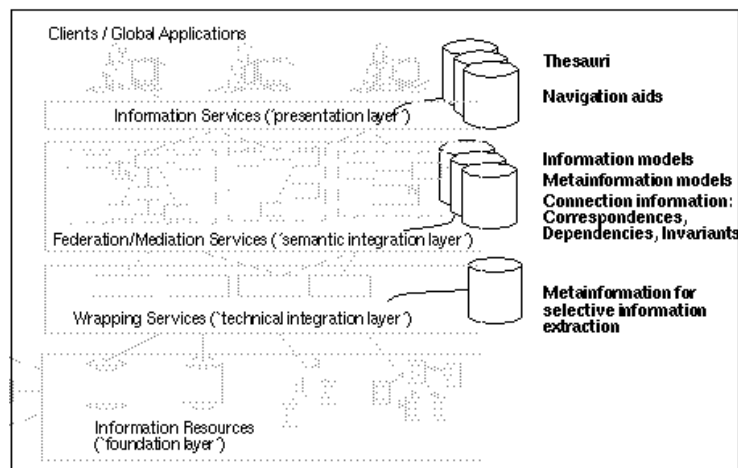


Figure 6. Meta- data within the “reference architecture”

5.3 Serge Abiteboul 1999

In [Abit99] Abiteboul discusses views and XML. The notion of views is essential for databases and for data sources in general. Therefore it is important to have a concepts of views in XML. XML already has the ability to define views, namely XSL. However according to Abitoul there are more general views than this. [Abit99] says that a view specification should rely on a data model and a query language as in the relational database world. However, due to the nature of XML it should be as the ODMG model based on objects. [Abit99] argues that views for XML are more important than for classical databases, most XML documents are primarily used as a common model for otherwise heterogeneous data. He suggested a possible architecture for this. It is based on three components:

- The data server that may be a database, a XML repository, or any (possibly wrapped) source capable exporting XML data.
- The view server that restructures data to construct the view, possibly deals with access rights, and integrates data from several sources.
- An XML view document that is handled by a standard web browser and interacts with the view server, eg., to obtain data.

[Abit99] discusses several notions on views:

View = query

This "view" comes from the relational world and is certainly true for XML sources (providing there is a query language) the same problem occurs with XML sources as with relational databases: propagation of updates. [Abit99] argues that XML views require more than standard database view technology, but that this database technology can be used in solving some of the problems.

View = world of objects

[Abit99] believes that XML data is object based by nature and strongly supports the DOM (Document Object Model) viewpoint. Since there are objects you may as well introduce methods for instance to define virtual data or conversion functions. This "viewpoint" is very similar to the object database world. Thus most notations about views in Object-oriented databases can be imported in XML. Some relevant object-oriented database views are:

- Virtual values: these are essentially like relational views.
- Virtual classes: a set of database objects may be logically grouped into a view class, and, as such, they may acquire new interfaces.
- Imaginary classes: an imaginary class allows to (virtually) create a set of objects that exist only in the view and do not correspond to database objects.

View = semi structured data

The structures used in semi structured data (usually a labeled graph) resembles the structure of XML. The

management of such data yields novel issues. The most important one is query optimization.

View + = structured data

XML should not only support semi structured data, but structured data as well. If XML data has a structure this should be exploited. This can boost performance.

View = changing world

Once a view is generated, it can change in the future because data is being updated. One should not recompile the complete view but only the updates. The active view system uses a specification language for this. The specification of an application includes definitions of actors involved in the application. Each actor is specified by:

- The data and operations available to this particular actor.
- The activities this actor may be engaged in and the data and operations available in each.
- Some active rules that notably specify the sequence of activities (a workflow component) but also the events this factor once to be notified of (a subscription component) and those that have to be logged (a tracing component).

According to [Abit99] databases can bring to XML the control of updates.

View = work space

In particular, a view will typically be on a different machine than the data sources. Thus, a view should be thought of as a workspace that in particular might contain previous queries and the result of these queries.

5.4 Jennifer Widom & Alon Levy 1999

[Widom99] discusses research opportunities of the database community with respect to XML. According to [Widom99] XML has the potential of changing the web into a database, using the XML query mechanism. The current state of query processing on the web is:

- Data embedded within HTML pages needs to be pre-processed by special purpose, page specific parsers before queries can be posed. Otherwise, ad hoc queries are limited to simple keyword based searches that understand documents as streams of words.
- Data stored within traditional database management systems generally is accessed on the web only through simple and rigidly formed based interfaces.

If data would be in XML format or XML compatible it would enable users to pose XML queries over this data. One of the first things that have to be done according to [Widom99] is encoding information into XML. Furthermore, XML raises several research topics for the database community. These topics focus on a database like treatment opposed to focusing on XML as an information integration tool:

- The ability to map XML encoded information into a true data model.
- Resolving conflicts that arise when mixing concepts of documents and databases.
- Theoretical results and practical techniques for designing XML databases are needed, to the extent possible within the relatively free form nature of XML.
- The relationship between XML's optional document type definitions and traditional database schema needs to be understood and exploited.
- And appropriate query language for XML needs to be defined.
- Database updates in an XML setting must be considered, with the focus on environments that are heavily read oriented.
- Efficient physical layout and indexing mechanisms are required for large storage of XML data.
- View mechanisms.

Before XML came into focus Jennifer Widom worked on two projects that use a language similar to XML; Lore and TSIMMIS (see chapter 5). These projects used a language called OEM (object exchange model) to integrate data. In 1999 Lore was migrated from OEM to XML. Widom sees Lore as a project to explore the possibilities of XML. According to [Widom99] much of the research done at Lore can be translated or extended to XML. This is something that will be investigated the next years.

The [Levy99] paper is an addition to the [Widom99] paper. In [Widom99] the main focus was on database issues. [Levy99] focuses more on XML as a way to integrate data sources. Furthermore, [Levy99] argues that no new issue is raised by XML. However the way that is thought about database problems should change. [Levy99] discusses the following research issues with respect to XML and data integration:

- Languages for source descriptions: A key element of the data integration system is a language for describing the contents and capabilities of data sources. These descriptions provide the semantic mapping between the data in the source and relations in the mediated schema.
- Query re-formulation algorithms.
- Translation among DTD's of XML documents.
- Obtaining source descriptions: when the number of data sources grows, it becomes more difficult to manually write their content descriptions.

Another issue that is important and can benefit from database research is measures of complexity. Besides the measures of complexity for databases there are two new measures of complexity for XML:

- Number of XML sources.
- Degree of irregularity in the data.

5.5 Ram 1999

[Ram99] discusses a model for the integration of data sources within a geographic environment. Within a geographical information system there are often different kinds of data sources. Data is gathered by sensors (satellites, weather balloons, earth observatory systems, etc.). All these sensors store data in different data formats. However, if you want to do an analysis you will need different data sources and you will need to correlate the data from these data sources. One of the main bottlenecks is the inability to conveniently understand and access the data. Many times users will need to assemble the necessary model data from different data sets which further compounds the problem. Based on this [Ram99] developed a semantic model that explicitly captures the spatial and temporal characteristics of GIS data and their interrelationships. This model is part of an overall system that can be used to solve some of the problems in accessing data from heterogeneous GIS sources. The model is based on the unifying semantic model (USM). The USM is an extended version of the entity relation model. This model (USM*) extends the USM model to support spatial and temporal objects typically found in GIS data. The model also captures the behavior of dynamic and process oriented spatial objects used for decision support simulations. According to [Ram99] these constructs are typically missing in most Object-oriented models and programming languages. Although Object-oriented data models and capture the dynamic behavior of spatial objects, such models hide this behavior in the methods associated with each object class. USM* allows explicitly to capture and show this information. USM* has constructs that define dynamic entities and cause-effect relationships. USM*uses the special constructs as the basis for high-level user interfaces to heterogeneous data sources. According to [Ram99] USM*has several advantages over existing data models:

- The semantic model allows users to browse meta data.
- Users access the data through the semantic model rather than having to learn GIS has comandns.
- The semantic model helps to interface the simulations system with the data without the intervention of a technician.
- The model allows access to the spatial and temporal data because it explicitly captures spatial and temporal entity classes and relationships.

[Ram99] says about this: "to the best of our knowledge, this is the only model that allows explicit definition of temporal and spatial aggregates as well as dynamic entity classes. At the same time, the model allows users, such as policy-makers, or database managers to integrate heterogeneous data sources without knowing the physical structure." Several lessons were learned from this:

- The meta data repository is very important because it lets users not only define data but extract it.
- Since the simulation output can also be accessed through the semantic model, it is important to provide schema evolution. For example, fire may cause changes in the vegetation of the landscape, which will need to be captured back into semantic model that represents the databases.
- The system must provide access to data that exists in the various databases as well as data that can be

derived from the existing data. This is where the semantic model proves extremely useful.

Figure 7 shows the architecture of the system.

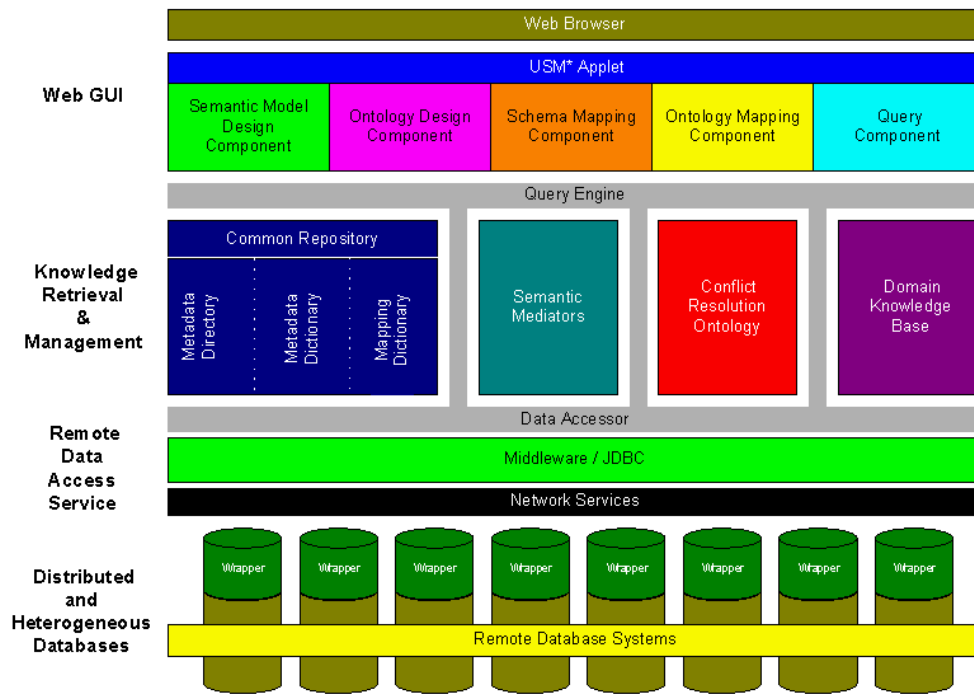


Figure 7. USM* architecture

6 Projects

Depending on how data is integrated, data integration systems are also referred to as Intelligent Information Integration (I3). This section discusses several projects with respect to data integration. There are numerous other projects. We mention a few of them: LORE, OBSERVER, KRAFT, and COIN. The projects focus on different aspects of data integration. MIX focuses on mediation and XML. TSIMMIS and DAISy use a mediator architecture. Another important aspect is their self describing language. HERMES also uses a mediator approach but integrates knowledge based on reasoning systems. AMASE uses a meta- database (catalogue) to integrate data from different data sources.

6.1 MIX

The MIX project (Mediation of Information using XML) is a collaboration between the UCSD Database Laboratory and the Data-intensive Computing Environments (DICE) group at SDSC. The goal of the project is to study, develop, apply and evaluate systems for mediation across heterogeneous information sources. MIX is a successor of TSIMMIS and relies on the mediator architecture described in [Wie92]. Data is wrapped into an XML format. Using the DTD of the "XML documents" data can be queried. A query language XMAS is used for this purpose. XMAS is a high-level query language based on XML-QL. Because this language was not rich enough XMAS was developed. XMAS allows object fusion and pattern matching on XML documents. To present data in an appropriate view, a DTD view is constructed. This construction is not done automatically. A graphical user interface BBQ (Blended Browsing and Querying) allows users to look for data. More information can be found at: [MIX] and [Bar99]

6.2 TSIMMIS

"The goal of the TSIMMIS project is to develop tools that facilitate the rapid integration of heterogeneous information sources of both structured and unstructured data" [Gar95]. The system consists of the following components:

- Translator (wrapper). Converts the data objects to a common information model. This information model is self describing (OEM). All objects and sub objects have labels that describe their meaning.
- Mediators. Mediators embody knowledge of a specific type of information. Although not mentioned in the documents it can be regarded as an agent.
- Classifier/extractor. Converged plain text into structured data.
- MOBIE (mosaic based information explorer). A browser for the user.
- Constraint management.

"TSIMMIS is not a single global database containing the integrated information, nor even a global database schema. Furthermore, mediators and translators are not required to produce objects with a fixed schema or type, but each query determines the schema for objects that match the query." [Gar95]

One of the goals of TSIMMIS is mediation and translation generation. The object exchange model is used to reach this goal. It can be viewed as an Object-oriented model, but the also as a form of first order logic (see [Gar97]). More information about this project can be found at [TSIM].

6.3 InfoSleuth

InfoSleuth is a system for the integration of heterogeneous sources developed by MCC (microelectronics and computer technology corporation, Austin Texas). The different sources are integrated in a dynamic way. The dynamic integration is made possible by using a network of cooperating agents. The system uses the following agents:

- User agent. Enables the user to access the system by using domain ontologies. (User interface)
- Resource agents. Accesses the information sources. It translates queries from a common query language to queries that can be understood by the different information sources (wrapper).
- Ontology agents. Provides overall knowledge of ontologies.
- Broker agent. Aims at finding the re-sources required to solve a query.
- Task execution agent. Routes requests to the appropriate resource agents and decomposes queries into sub queries (mediator).
- Data analysis agent.
- Monitor agent. Stores records of the agent interactions.

InfoSleuth allows different formats and re-presentations of ontologies. The differences in formats and re-presentations are allowed by use of an ontology meta- model. Agents communicate via a standard set of KQML performatives. The Knowledge Interchange Format (KIF) is used as a communication mechanism. More information can be found at [INFO]

6.4 HERMES

HERMES is a Mediator system being developed at the University of Maryland. Unlike TSIMMIS, which integrates semi-structured data sources, HERMES focuses on integrating knowledge based on reasoning systems. Specifically, it provides support for amalgamating knowledge from relational, object-oriented, spatial, and temporal domains into a single reasoning system. Integration is achieved by hooking up each component system to the semantic model, called the Generalised Annotated Program (GAP) framework, an extension of logic programming. A prime motivation behind the design of HERMES is to modularize the activities involved in creating a mediator. Centrally stands the logical separation of the construction of the mediation code, and integration of new domains. HERMES uses a high-level declarative, logic-based language for the construction of mediators. Several tools can assist the mediator author. These tools are related to domain integration, conflict resolution and semantic integration, information pulling and semantic integration. The last tool is used for the fact that information can be integrated in different ways. A mediator author can select one or more of these approaches. HERMES also uses an internal clock to keep track of the current time. The value of the clock will maintain temporal data on the creation of various data sources. It is sometimes difficult for a mediator to locate the data in the different data sources. To solve this problem in HERMES a yellow page server is created. This server has as input a topic and returns a set of structures that enables the mediator to locate the data. HERMES is based on mediation architecture described in [Wie92]. More information can be found at: [HER] and [Sub99]

6.5 DAISY

Provides both high-level user interfaces and program level interfaces and program level access across heterogeneous databases. A higher level data structure specification can describe different database schemes. The specifications are used to create information mediators. Data is mapped to an internal format. To accomplish this, the system uses a high-level data structure specification (HLDSS). Using two of these specifications and a transformation rule specification, data from one database scheme (source) can be put into schema of another database (target). The HLDSS resembles very much XML. The specification is a grammar with annotations attached to it. For every two databases a bridge needs to be built to accomplish this. In DAISY this is called an information bridge.

Furthermore, the importance of self-description is discussed in [Morg96]. "We believe that self description is an important organising principle for future highly distributed very large heterogeneous information networks. This would enable the automatic self registration of software modules and software agents simply by their being presented to a meta- data repository."

Furthermore, [Morg96] says: "We are currently designing a Meta- data repository to make explicit and utilise knowledge and semantics needed to mediate among different islands of terminology and meaning."

6.6 AMASE

NASA has a lot of astrophysics data that is the result of different missions. This data is stored in different data sources. AMASE is an object-oriented astrophysics catalogue that enables users to query these data sources in an uniform view. The catalogue is considered to be a meta- database that references to a low-level astrophysics data. One of the reasons for choosing an object-oriented database is that a significant fraction of the data is semi structured or has complex data types that are not supported by traditional relational systems. The object relational database management system of Illustra was used to implement this. This schema of a remote site is first mapped to the AMASE schema. In AMASE all the scientific and mission data attributes are linked to an astronomical object. In the future AMASE will be extended to integrate with other data archives, utilities with user defined views, and object across identification (see [Cheu98], [Cheu95]). More information can be found at [AMA].

7 Commercial products

This section discusses several commercial products the deal with data integration. Most products deal with the automation of building data bridges between two data sources or relate only to relational databases.

7.1 Ardent

Ardent has a product called meta- stage. It uses meta- brokers to integrate data from different relational sources via a meta- hub. The meta- hub contains elementary meta- meta- data to accomplish this. The question is how you know if these elements are sufficient for building meta- data. At the moment it is not possible to edit these elementary meta- meta- data. Furthermore, the product is focused on a relational database. Information about meta- stage can be found at [ARD].

7.2 Merlin

The INEEL data integration mediation system was built to address the data integration issues of a specific environmental restoration domain. The system was designed to be domain independent, such that a variety of domains could utilize IDIMS's integration capabilities as long as the appropriate domain knowledge is provided. The system consists of a mediator (or mediators) and wrappers. The mediator is used to describe a domain specification. For every data source a wrapper describes the mapping between the data source and the domain specification. The mediator and the wrappers use an extension of ODL to make these descriptions. The mediators use MODL, and the wrappers use WODL. The following extensions were made to ODL:

- Relationship expressions to describe how two classes or related.
- Data source mapping specifications (in the wrapper subsystems only).
- Source specific data transformations (in the wrapper subsystems only).

QEM (query exchange model) is a query re-presentation that is developed and used in IDMS. The reason for this was that the query representation languages were not rich enough. QEM is re-presented in tree form and is

similar to a parse tree. Furthermore, it is closely related to OEM. OEM is used for the exchange of data between wrappers and mediators. OEM is an exchange model developed at Stanford University in the LORE project. After that it was used in the TSIMMIS project. The techniques described above are incorporated in the Merlin product. Merlin is a middleware tool that has been developed to solve data access and integration issues. It uses these techniques to create a virtual database. This virtual database is the mediator described above. In all reports about IDIMS and Merlin, relational databases are used in examples. However it is not explicitly mentioned that the tool is only for relational databases (it is also not explicitly mentioned that you can use object-oriented databases). More information can be found at: [MER] and in [Wie97+2]

7.3 Meta- Integration Works (MIW)

MIW is a tool set for managing the movement of data. It includes a model manager, model browser, model converter, model comparator, and the data bridge builder. Using the data bridge builder users can build conversion modules from one repository to another. MIW already contains conversions for the most popular repositories/models. This tool is the Meta- Integration Model Bridge. The MIMB is a tool that enables the transfer of models from one modeling tool to another modeling tool. Furthermore it allows direct access to schemas of relational databases or object-oriented databases. It also is able to convert XML data to different models or repositories. The MIMB supports besides others the following models/repositories:

- Rational Rose v4.0 to 2000
- Microsoft Visual Modeler 2.0 (same as Rose)
- Direct access to the schema of "live" Relational (RDBMS) and Object (OODBMS) databases: Sun/JavaSoft JDBC v2.0 driver (ODBC)
- Microsoft Repository 2.0 via XML Interchange Format (XIF)
- UniSys Repository 2.0 (via XMI)
- W3C XML's Specifications (DTD) 1.0

[MIW] Gives a more detailed description of the product.

7.4 ETI

ETI provides a tool that can be used to convert data from one source to another. The tool consists of four major subsystems:

- Environment editors. ETI's Editors are used to specify data conversions, administer the software and customize its features. Users specify data mappings and transformations
- Data system libraries. DSLs give ETI•EXTRACT the ability to access any type of database or file structure from relational and non-relational DBMSs to legacy file structures. DSLs enable the software to generate programs in the language most appropriate for the database being accessed and the platform, which it resides. Users can tailor DSLs to meet site-specific requirements, and companies can even create their own DSLs to access proprietary databases or systems.
- Meta-Store. The Meta-Store is a centralized repository both for information used to perform conversions and for meta- data about conversion projects. Included in the Meta-Store are definitions of the data access and storage systems; the computing environment; and the schemas, interrelationships, and test and transformation logic that constitute the data integration project. The Meta-Store provides support for versioning of meta- data to reflect changes and enhancements. Finally, meta- data in the Meta-Store can be shared with a variety of products and repositories throughout an organization's IT infrastructure using mechanisms such as ETI's meta- data Facility™ (MDX), the Meta- Data Coalition's Meta- Data Interchange Specification (MDIS) or the Microsoft® Repository Translator.
- Generation engine. The Generation Engine generates the actual code used to extract, transform, move and load data. The engine works behind the scenes, translating natural language statements into code, developing process flow steps, generating code and managing its execution.

Main idea behind this tool is to aid users in building information bridges between different data sources. More information can be found at [ETI].

8 Standards

Databases exchange information with the detector description database, other databases, or applications. It is therefore important that there are some standards in exchanging/describing information. This section discusses several standards. Using standards has several advantages. Information is easy to exchange with other applications (that use this standard). Tools that support this standard can aid in the development of applications (reuse). The standards discussed in this section relate to several aspects of data integration. Exchange (MDIC, OIM). Solid modelling (STEP, EXPRESS) (this is used with in the GEANT simulation package). Meta data description (XML, DOM, MOF). Object-oriented databases (ODMG). Furthermore, several standardization organizations are discussed.

8.1 ODMG

ODMG adds database functionality to object programming languages. ODMG extends the semantics of the C++, Smalltalk and Java object oriented programming languages to provide database programming capability, while retaining native language compatibility. A major benefit of this approach is the unification of the application and database development into a seamless data model and language environment.

The ODMG standard sits at the intersection of three standards domains: databases, objects and object oriented programming languages. Rather than defining a completely new standard from the ground up, the ODMG standard builds upon the existing OMG, SQL-92 and ANSI programming language standards to define a framework for application portability. The ODMG standards' functional components include an Object Model, an Object Definition Language, an Object Query Language, and language bindings to C++, Smalltalk and Java.

The ODMG Object Model provides the unifying basis for the entire ODMG standard. The ODMG Object Model extends the OMG Object Model with capabilities like relationships and transactions to support database functionality.

The key concepts of the object model include:

- Attributes and relationships object properties
- Object operations (behaviour) and exceptions
- Multiple inheritance
- Events and keys
- Object naming, lifetime and identity
- Atomic, structured and collection literals
- List, set, bag and array collection classes
- Concurrency control and object locking
- Database operations

All DBMSs provide a data definition language, or DDL, that enables the user to define the schema. The ODMG Object Definition Language (ODL) is a database schema definition language extension of the OMG Interface Definition Language (IDL) that allows the standard of an object type and its attributes, relationships and operations.

ODL creates a layer of abstraction such that an ODL-generated schema is independent of both the programming language and the particular ODMG-compliant DBMS. Accordingly, ODL considers only object type definitions and ignores the actual implementation of methods. An ODL-generated schema can be freely moved between compliant DBMSs, different language implementations, or even translated into other DDLs, such as those proposed by SQL3.

OQL is an SQL-like declarative language that provides a rich environment for efficiently querying of database objects, including high-level primitives for object sets and structures. OQL provides a super set of the SQL-92 SELECT syntax. This means that most SQL SELECT statements that run on relational DBMSs tables work with the same syntax and semantics on ODMG collection classes.

OQL also includes object extensions to support object identity, complex objects, path expressions, operation invocation and inheritance. OQL's querying capabilities include the ability to invoke operations in ODMG language bindings, and OQL may be called from within an ODMG language binding for embedded operation. OQL maintains object integrity by invoking an object's defined operations, rather than using its own update

operators.

In an effort to maintain compatibility with SQL SELECT in the future, the ODMG has created a strong working relationship with the ANSI X3H2 committee, which is defining the SQL3 standard. The objective is to ensure interoperability between OQL and SQL-3.

8.2 OMG

Application integration and distributed processing are the same thing. OMG believes that an approach based on objects simplifies these problems. The mission of the object management group is to develop a single architecture; using object technology for distributed application integration. Guaranteeing: reusability of components, inter-operability and portability. To achieve this the following technical goal is set: Foster interoperability and portability for application integration through cooperative creation and promulgation of Object-oriented standards based on commercially available software:

- Single terminology for object orientation
- Common abstract framework
- Common reference model
- Common interface products

Several standards have been generated by OMG:

- CORBA
- MOF. A CORBA common facility for the management of meta- information. The MOF can be used as a meta- information repository within CORBA distributed systems.
- UML
- XMI. XMI uses XML to encode UML. This will allow UML models to be exchanged among various tools.

8.3 MDC

The Coalition allies software vendors and users with a common purpose of pushing forward the definition, implementation and ongoing evolution of a meta- data interchange format standards and their support mechanisms. MDC currently has 2 standards: OIM and MDIS. MDC is a member of OMG and OMG is a member of MDC. The goal of the MDC is to enable full-scale enterprise data management, different IT tools must be able to freely and easily access, update, and share meta- data. The only viable mechanism to enable disparate tools from different vendors to exchange meta- data is a common meta- data standard with which the different vendors' tools can comply. By choosing tools that comply with meta- data standards, purchasers can be assured of the accurate and efficient sharing of meta- data essential to meeting their users' business information needs. This will allow IS managers to build on investments in data management tools and infrastructure with each additional purchase of a standard-compliant product.

8.4 STEP

The information generated about a product during its design, manufacture, use, maintenance, and disposal is used for many purposes during its life cycle. Their use may involve many computer systems, including some that may be located in different organisations. In order to support such uses, organisations need to be able to represent their product information in a common computer-interpretable form that is required to remain complete and consistent when exchanged among different computer systems.

STEP stands for: SStandard for the Exchange of Product model data. It is one of the largest efforts to create a data standard for industrial products. The STEP standard is also known as ISO standard 10303. The STEP standard has been adopted as an American National Standard (ANS US/IPO 200).

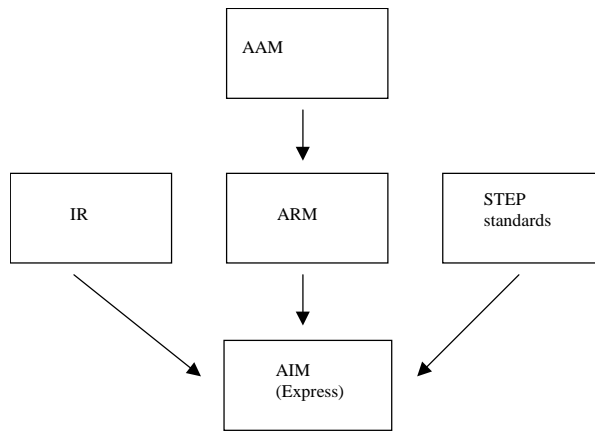


Figure 8. STEP standardization process

The STEP approach is to produce standard product models for use within specific areas of applications (called application protocols (AP's)), and to strive and co-ordinate these models across application areas as much as possible. Figure 8 lists the processes for developing an AP. An application activity model (AAM) identifies business processes in which the AP is used and shows information flows among these processes. ARM (application reference model) and IR are building blocks from which an AP is built (for example geometry, topology).

8.5 EXPRESS

EXPRESS was originally developed to provide a formal, and computer processible, means of defining the data necessary to describe a product (anything from a microchip to a battleship) throughout its lifecycle, from time of conception through its manufacture to its time of disposal. ISO 10303, commonly known as STEP (STandard for the Exchange of Product model data), uses EXPRESS as the formal specification of the required data and its relationships. EXPRESS is also used in other standards, such as EDIF for electronic printed wiring boards, ISO TC 211 for Geographic Information Systems, and will be used in forthcoming editions of the SGML and XML standards. It has also found broad applications within industry, such as POSC (Petrotechnical Open Software Corporation) for modeling oilfield exploration and production information, the Human Genome Project for data exchange between genomic databases, and the London Stock Exchange for Asset Management. Many European ESPRIT Projects use EXPRESS. US projects and industrial consortia that use EXPRESS include, among others, the CAD Framework Initiative (CFI) and the National Industrial Information Infrastructure Protocols (NIIP) project.

There are basically two aspects to EXPRESS: (1) It provides for the modeling of data and data relationships with a very general and powerful inheritance mechanism (much more than is provided in OO programming languages), and (2) it includes a full procedural programming language which is used to specify constraints on data instances. As noted, EXPRESS-G is a subset of EXPRESS as it does not include the constraint portions of the lexical language. EXPRESS models may be written in the style of Entity-Relationship, Relational, Object Oriented, or other kinds of data modeling. It may also be considered to be a Set Theoretic specification language, and some have even gone so far as to indicate that it might be classed as a higher order predicate logic language (see [Wil98])

The EXPRESS family of languages has been, and is being, developed under the auspices of ISO TC184/SC4. The EXPRESS Language Reference Manual defines a graphical subset of the lexical language called EXPRESS-G. EXPRESS-I is a lexical language for the display of data instances and also for the formal definition of test cases. EXPRESS-X, is a mapping language for data translation between two EXPRESS models that are similar in semantic meaning but which differ in their data forms. In terms of software tools for authoring, apart from EXPRESS-G which requires some drawing capabilities, the members of the EXPRESS family only require a text editor.

8.6 XML

XML stands for extended markup language and is a subset of SGML (see [OAS]). Basically XML is a grammar to define grammars. These grammars have a nested structure such that it is easy to parse. Using this, one can describe the structure of data. The grammar is defined in a DTD (document type definition). Once data is

captured in this definition, it can be parsed and displayed using XSL. XSL defines how data should be presented. XML enables you to define structure, but does not define presentation rules. The following figures show an example of XML. Figure 9 shows some cryptic password information. Figure 10 shows a document type definition that can be used to structure the information displayed in the password information. Figure 11 shows how the password information is structured into an XML file using the document type definition.

```
dschmidt:x:1201:1200:Dave Schmidt:/export/home/dschmidt:/bin/ksh
```

Figure 9. Password file information

```
<!-- Document Type Definition for passwd documents -->

<!-- password-file element -->
<!ELEMENT password-file (entry)*>

<!-- password-file element attributes -->
<!ATTLIST password-file last-updated CDATA #REQUIRED>

<!-- entry element -->
<!ELEMENT entry (username,password,userid,groupid,gcos-field,homedir,login-shell)>

<!-- superuser entity -->
<!ENTITY superuser SYSTEM "superuser.xml">

<!-- End of DTD -->
```

Figure 10. Document Type Definition

```
<?xml version="1.0"?>

<!-- DOCTYPE password-file SYSTEM "passwd.dtd" -->

<password-file last-updated="09/20/98 11:34:33">
  <entry>
    <username>dschmidt</username>
    <password>x</password>
    <userid>1201</userid>
    <groupid>1200</groupid>
    <gcos-field>Dave Schmidt</gcos-field>
    <homedir>/export/home/dschmidt</homedir>
    <login-shell>/bin/ksh</login-shell>
  </entry>
</password-file>
```

Figure 11. XML file

An advantage of XML is that it can be used to create a common language for data sources on the Internet (which can be seen as a set of data sources). Thus enabling data transfer between different heterogeneous data sources. If you have an XML file and its DTD then data can be extracted from it. It is already possible to generate XML

files from relational databases and generate relational databases from XML files (see [Buck99]). At this moment the WISDOM project tries to accomplish this for Object-oriented databases (see [Afaq99]). Instead of transforming XML into a database format, it can also be viewed as an XML repository. Because the data is structure and can be queried. At the moment there is not yet a uniform query language for XML (like SQL), but several proposals have been made. Furthermore, there is a proposal for a standard on XMLQL (see [Robie98]) by the W3C. [Widom99] discusses several research directions for XML with respect to databases (see section 5.4).

Although XML is a very recent technology, the idea of a language that can be used to integrate different data sources is not new. The Lore project at Stanford (see [Widom99]), TSIMMIS (see [Gar95]), and Daisy project (see [Morg96]) had their own languages similar to XML.

Although XML has much expressive power, there is additional information (constraints on data and structure) needed for full exchange of information. This is where XMI (XML meta- data interchange) starts. The main goal of XMI is to support the exchange of meta- data between modeling tools based on OMG MOF. To accomplish this XMI uses XML. Although XMI only supports OMG MOF models, it is assumed that XMI could be used for interchange of STEP schemas and instances (see [Wright99]).

8.7 Meta- Data Interchange Specification

To enable full-scale enterprise data management, different tools must be able to freely and easily access, and in some cases manipulate and update, the meta- data created by other tools and stored in a variety of different storage facilities. The heart of the MDIS is the core set of components that represents the minimum common denominator of meta- data elements and the minimum points of integration that must be incorporated into tool products for compliance.

The MDIS also provides for a set of optional/extension components that are relevant only to a particular type or class of tool or a specific application or architecture. Because these are used by more than one tool or application, they can and should conform to the specification definition and set of access parameters, but because they are not generic across all tools, architectures, or applications they would not be eligible for the core set, nor required for compliance.

The Meta- Data Interchange Specification draws a distinction between:

- The application meta- model - the tables, etc., used to "hold" the meta- data for schemas, etc., for a particular application;
- The meta- data meta- model - the set of objects that the MDIS can be used to describe. These represent the information that is common (i.e., represented) by one or more classes of tools, such as data discovery tools, data extraction tools, replication tools, user query tools, database servers, etc. The meta- data meta- model should be:
 - Independent of any application meta- model
 - Character-based so as to be hardware/platform-independent
 - Fully qualified so that the definition of each object is uniquely Identified

The meta- data model must be sufficiently extensible to allow a vendor to store the entire meta- model for any application. In other words, MDIS should provide mechanisms for extending the meta- data model so that additional (and possibly encrypted) information can be passed.

8.8 MOF

The OMG MOF is a generic framework for describing and representing meta-information in a CORBA-based environment. This is intended to include:

- Interface definitions for CORBA objects, COM objects, DCE
- Services
- Service types for the CORBA Trader,
- Meta-data for databases and information retrieval systems,
- Models and project management information for software
- development tools,

- Mapping descriptions for interoperability tools; eg. application level bridges.

The MOF is designed to support many different kinds of meta-information. This is achieved by treating the meta-information as information, and formally modeling each distinct kind of meta-information. These formal models are expressed using the meta-modeling constructs provided by the MOF Model. The MOF specification also defines an IDL mapping which allows models expressed using MOF Model constructs to be translated into interfaces to CORBA-based meta-information services. These interfaces can be implemented by hand or using non-standard server generation tools. The mapped interfaces for a meta-model all inherit from a standard Reflection module that supports "introspection" and meta-model independent access and update. The interfaces can be used within a MOF Repository framework, or deployed independently. The MOF is designed for representing meta- information within a CORBA environment however; it turned out to be general enough to use it also in a non-CORBA environment.

8.9 OIM

The Open Information Model (OIM) is a set of meta- data specifications to facilitate sharing and reuse in the application development and data warehousing domains. OIM is described in UML (Unified Modeling Language) and is organised in subject areas. The data model is based on industry standards such as UML, XML, and SQL. The Open Information Model is grouped into subject areas Analysis and Design Models

- Objects and Components Models
- Database and Data Warehousing Models
- Knowledge Management Models
- Business Engineering Models

The OIM is a core model of the most commonly used meta- data types. Starting with an abstract core model and adding additional subject areas over time has proven to be the most effective strategy in model development, reducing redundancy and promoting extensibility. By first defining a domain-specific model for a generic subject area that addresses the implementation-neutral aspects of that area, and only later specialising it to one or more specific subject areas that describe the tool-specific extensions, the OIM tries to minimise redundancy and enables application neutrality sharing and reuse.

To encourage compliance - and to eliminate technology-specific dependencies - the MDC OIM relies heavily on widely accepted industry standards. It uses UML as a base model. It uses XML for meta- data interchange. And it uses SQL for data retrieval. The MDC-OIM is based on the Microsoft Open Information Model, a meta- data model and specification that is part of Microsoft Repository, a meta- data management product. OMG has been working on a similar standard initiative called the common warehouse meta- data initiative.

8.10 DOM

The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term "document" is used in the broad sense - increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

With the Document Object Model, programmers can build documents, navigate their structure, and add, modify, or delete elements and content. Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model, with a few exceptions - in particular, the DOM interfaces for the XML internal and external subsets have not yet been specified.

The DOM is designed to be used with any programming language. In order to provide a precise, language-independent specification of the DOM interfaces, the specifications are defined in OMG IDL, as defined in the CORBA specification.

9 Future work

The meta- model is considered to be a technique to integrate data from different data sources (see [Dras99] and [Kerh97]) other techniques are: A common exchange language (TSIMMIS, DAISy), ontological approach (InfoSofleuth) and mediator approach (MIX, HERMES). These techniques have been applied by different

projects. To build a system for the integration of information it is therefore necessary to look at how to integrate these techniques for solving this problem. In recent years standards emerged with respect to meta- data and data integration (see chapter 7). It is important to look at how these standards can be used in combination with these techniques. Currently there is much attention for the Grid (see [Fos99] and [Cher99]). A Grid enables users to use computer re-sources on different geographic locations. To accomplish this a middleware layer is needed. Part of this Grid is the data Grid. The data Grid enables users to send/retrieval/analyse data on different geographic locations. This data is located in different formats in different databases. This results in the need for a layer that is able to integrate data. At the moment there are several initiatives for building a Grid: Particle Physics Data Grid (see [Ppdg]) and the NASA Information Power Grid (see [Nasa99]).

10 Conclusion

The projects discussed in chapter 5 focus on different aspects of data integration. MIX and HERMES focus on the mediator concept. HERMES approaches the problem by using a reasoning system. TSIMMIS and DAISy focused mainly on self-describing languages as a tool to integrate data, while InfoSleuth is mainly concerned with ontologies. Furthermore, AMASE uses a meta- database to integrate data. The DAISy project is also working on a meta- data repository. Most projects that used a meta- model use a certain degree of first order logic. A "meta- model" alone is not powerful enough. All the different focus points in these different projects are important for this seamless integration of information. Furthermore, not everything is automated yet (plug and play). Commercial products support mainly building of information bridges between different databases. Furthermore, it is not always clear if object-oriented databases are supported. Using this knowledge, a division into subsystems can be made of the "meta- model" described in chapter 1 and [Estr98]. The meta- model should contain the following subsystems:

- domain specification
- common exchange format (language)
- high-level query language
- building blocks for views

10.1 Domain Specification

In order to integrate data there needs to be a context or domain that all data sources relate to. This domain can be seen as a "common schema" of all data sources connected to the meta- model. It enables the creation of a "virtual database" (a database that is not there but is physically re-presented by many data sources). (e.g. all data sources in the CMS experiment are related to the CMS detector). Thus the domain specification should be strongly related to the detector description. Furthermore, new databases can emerge in the future. This should not affect the domain specification. It cannot be foreseen how the CMS detector (and the domain) will change in the future. Therefore this domain should be adaptable.

10.2 Common Exchange Format

The data sources in the CMS experiment can have different structures or even have different technologies (eg. object-oriented, relational, or none!). To enable data integration there should be a common exchange format (language). Databases should be "plugged" into the meta- model without much effort. Therefore this common exchange format should enable self-description. Users and applications will access/manipulate the data in the database via the domain specification. Thus the common exchange format should enable a mapping from the data sources to this domain specification.

10.3 High-level query language

In order to access the database as one "virtual" database there should be a high-level query language that interfaces with the domain specification. This domain specification is able to translate these "queries" into a query specific to the different data sources connected to the meta- model. It is not yet clear if this language will differ (or is an extension) from the common exchange format.

10.4 Building blocks for views

Although a query language enables users/applications to access the "virtual" database, it still can be difficult to

locate data, or integrate data into a desired format. Furthermore, different applications will need a different granularity/view of the numerical data available. Data can be stored as numerical data (the actual numbers). However, this can be too detailed for certain applications (because it is too computation intensive) They will use these numbers to get a less detailed view. For example see Figure 12. The numerical data contains nine numbers. The view that is generated from this contains four numbers. Furthermore, data can change over time, thus there should be update mechanisms for the views.

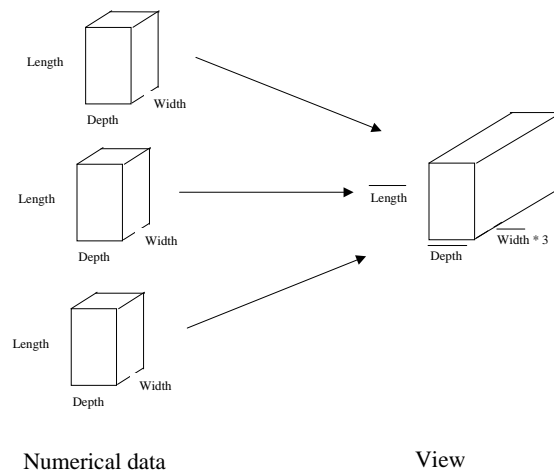


Figure 12. A view on data

Views can be seen as simple queries. But they can also be more complex. In a view you can for example specify what data you want (a set of queries). How you want to re-presented it (for example summarize the data returned by the queries into a more compact form). Furthermore, if a view contains data that can change over time you want to specify how this view will evolve over time should it stay the same or should change accordingly? These can be very complex structures. Furthermore, users do not want to build a view from scratch if they can re-use one. This means there must be the possibility to build views from views. Therefore there should be building blocks to enable this. These building blocks use the high-level query language to access the data.

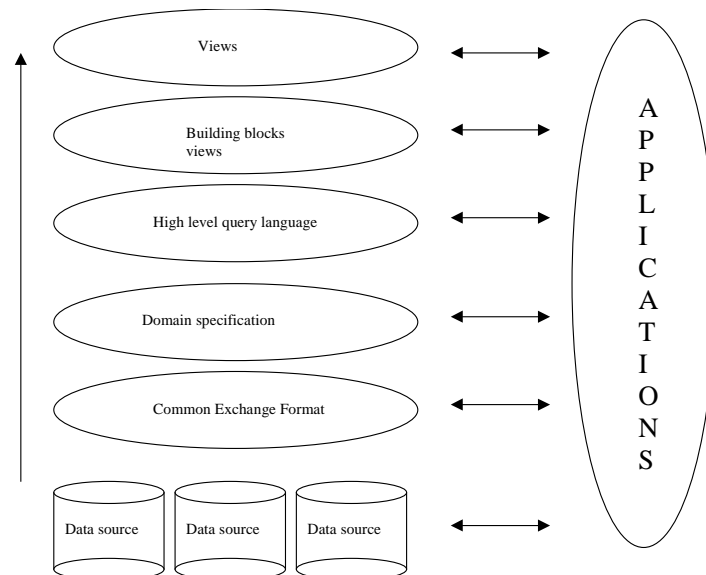


Figure 13. High level view of "meta-model"

This leads to the following architecture for meta- model (Figure 13). Applications can access the meta- model on any level. They even can access the data sources directly. However the most convenient way to access the data in a uniform way and to enable consistency is one of the levels above the domain specification level.

The architecture described in Figure 13 can be related to the meta- database architecture described in Figure 2. Figure 14 shows his relation. This figure shows the relation between the "meta- model" described in [Estr98], the actual meta- model, and the different focus points of the projects described in chapter 5. Note that the "meta- model" described in [Estr98] differs from the actual meta- model. The meta- model described in [Estr98] covers all aspects needed for information integration.

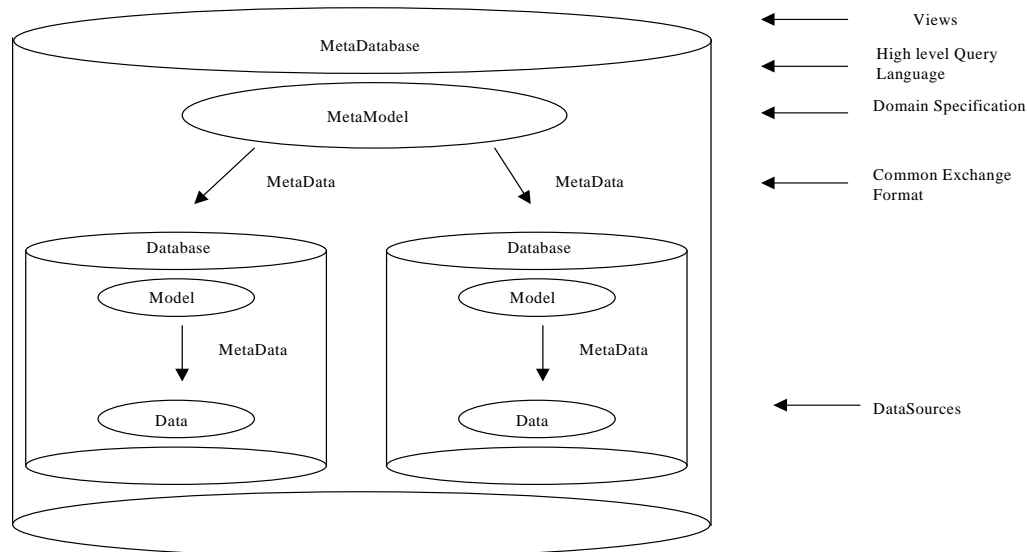


Figure 14. Relation between meta-database and "meta- model"

Some projects discussed in chapter 5 have constructed new languages (TSIMMIS, DAISy) or new exchange formats. However, today standards on languages and exchange formats are available (see chapter 7). It should be investigated how to use these standards with respect to the meta- model described in chapter 1. The advantages of using standards are obvious. The metamodel, model, and data are only parts that relate to the environment in which it will be used (physics environment). The other parts are independent of the environment. It is therefore important not to re-invent these parts but to reuse them if possible.

List of Figures

FIGURE 1. SUBSYSTEMS OF HETEROGENEOUS DATABASES.....	7
FIGURE 2. META-DATABASE ARCHITECTURE.....	10
FIGURE 3. FOUR LEVEL MODELING ARCHITECTURE.....	15
FIGURE 4. GENERAL ARCHITECTURE	16
FIGURE 5.....	17
FIGURE 6. META- DATA WITHIN THE “REFERENCE ARCHITECTURE”	18
FIGURE 7. USM* ARCHITECTURE	21
FIGURE 8. STEP STANDARDIZATION PROCESS.....	27
FIGURE 9. PASSWORD FILE INFORMATION	28
FIGURE 10. DOCUMENT TYPE DEFINITION.....	28
FIGURE 11. XML FILE	28
FIGURE 12. A VIEW ON DATA	32
FIGURE 13. HIGH LEVEL VIEW OF “META- MODEL”.....	32
FIGURE 14. RELATION BETWEEN META-DATABASE AND “META- MODEL”	33

Glossary

AAM	Application Activity Model
AIM	Application Interpreted Model
AMASE	Astrophysics Multispectral Archive Search Engine
ANSI	The American Standard Institute
AP	Application Protocol
API	Application Program Interface
ARM	Application Reference Model
BBQ	Blended Browsing and Querying
CAD	Computer Aided Design
CASE	Computer Aided Software Engineering
CDIF	CASE Data Interchange Format
CFI	CAD Framework Initiative
CMS	Compact Muon Soloid
COIN	Context Interchange project
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CRISTAL	Cooperating Repositories and an Information System for Tracking Assembly Lifecycles
	Database Integration System
DAISy	Database Management System
DBMS	Distributed Computing Environment
DCE	Data Definition Language
DDL	Data Intensive Computing Environment
DICE	Data Intensive Computing Environment
DNA	DeoxyriboNucleic Acid
DOM	Document Object Model
DTD	Document Type Definition
ETI	Evolutionary Technologies International
GAP	Generalized Annotated Program
GEANT4	toolkit for the simulation of the passage of particles through matter.
	Geographic Information Systems
GIS	High Energy Physics
HEP	Heterogeneous Reasoning and Mediator System
HERMES	High Level Data Structure Specification
HLDSS	Hyper Text Markup Language
HTML	Identity
ID	Interface Definition Language
IDL	INEEL Data Integration Mediation System
IDMS	Intelligent Integration of Information
I ³	Idaho National Engineering and Environmental Laboratory
INEEL	system for the integration of heterogeneous sources
InfoSleuth	Integrated Resource
IR	International Standard Organization
ISO	Information Technology
IT	
	Knowledge Interchange Format
KIF	Knowledge Reuse and Fusion / Transformation
KRAFT	Knowledge Query and Manipulation Language
KQML	Large Hadron Collider
LHC	a database management system for XML
LORE	Micro electronic and Computer technology Corporation
MCC	Meta- Data Coalition
	Meta- Data Interchange Specification
MDC	Meta Integration Model Bridge
MDIS	Meta Integration Works
MIMB	Mediation of Information using XML
MIW	
MIX	

MOBIE	Mosaic Based Information Explorer
MODL	Mediator ODL
MOF	Meta- Object Facility
NASA	North America Space Agency
OBSERVER	Ontology Based System Enhanced with Relationships for Vocabulary hETerogeneity Resolution
ODL	Object Definition Language
ODMG	Object Data Management Group
OEM	Object Exchange Model
OIM	Open Information Model
OMG	Object Management Group
OODBMS	Object Oriented Database Management System
OQL	Object Query Language
POSC	Petrotechnical Open Software Corporation
QEM	Query Exchange Model
RDBMS	Relational Database Management System
RNA	RiboNucleic Acid
SDSC	San Diego Supercomputer Center
SGML	Standard Generalized Markup Language
SQL	Structured Query Language
STEP	Standard for Exchange of Product model data
TOM	Temporal Object Model
TSIMMIS	The Stanford-IBM Manager of Multiple Information Sources
TSQL	Temporal SQL
UCSD	University of California, San Diego
UML	Unified Modeling Language
USM	Unified Semantic Model
WISDOM	Project to retrieve/insert XML data in Object Oriented database
WODL	Wrapper ODL
XMI	XML Meta- data Interchange
XML	Extended Markup Language
XMLQL	XML Query Language
XSL	Extended Stylesheet Language

References

- [Abit99] S. Abiteboul, "Views and XML", Proceedings. ACM Symp. on Principles of Database Systems, pp. 1-9, 1999.
- [Afaq99] Muhammad Anzar Afaq, Syed Shafqat Saeed, Heinz Stockinger, "Object Oriented Database Serialization Using XML", CMS Note, 1999
- [AMA] Amase, <http://tarantella.gsfc.nasa.gov/>
- [Ang99] Jonathan Angel, "XML: Ready for Prime Time", Network Magazine, 1999
- [ARD] Ardent software, : <http://www.ardentsoftware.com/>
- [Bar99] Chaitanya Baru, Amarnath Gupta, Bertram Ludäscher, Richard Marciano Yannis Papakonstantinou, Pavel Velikhov, Andreas Yannakopoulos, "XML-Based Information Mediation with MIX", Sigmod99, 1999
- [Birm98] Bill Birmingham et al. "EU-NSF Digital Library Working Group on Interoperability between Digital Libraries Position Paper", <http://galileo.iei.pi.cnr.it/DELOS/NSF/interop.htm>, 1998
- [Bez98] Jean Bézivin, "Who's Afraid of Ontologies?", workshop at oopsla, 1998
- [Bos99] Jon Bosak, Tim Bray, "XML and the Second Generation WEB", Scientific American, 1999
- [Buck99] Lee Buck, "XML, Schemas, and Extensibility", Extensibility, 1999
- [Chaw97] Sudarshan S. Chawathe, Serge Abiteboul, Jennifer Widom, "Representing and Querying Changes in Semistructured Data", 1997
- [Cheu95] Cynthia Cheung, Dave Leisawitz, Nick Roussopoulos, Stephen Kelly, Jane Wang, Gail Reichert, David Silberberg, "AMASE: An Object-Oriented Meta-database Catalog for Accessing Multi-Mission Astrophysics Data", http://adc.gsfc.nasa.gov/~blackwell/amase_paper/amase.html, 1995
- [Cheu98] Cynthia Y. Cheung, Nick Roussopoulos, Stephen Kelley, James Blackwell, "A Search and Discovery Tool AMASE", ADASS'98, 1998
- [Crit98] T. Critchlow, M. Ganesh, R. Musick, "Meta-Data Based Mediator Generation", LLNL, 1998
- [Dras99] Jelica Draskic Jean-Marie Le Goff Ian Willers Florida Estrella Zsolt Kovacs Richard McClatchey Marton Zsenei, "Using a Meta-Model as the Basis for Enterprise-Wide Data Navigation", IEEE Meta-data Conference, 1999
- [Erik98] Eriksson, J., "Real-Time and Active Databases: A Survey", In Proceedings of the 2nd International Workshop on Active, Real-Time, and Temporal Database Systems, volume 1553 of Lecture Notes in Computer Science, pages 1--23. Springer.
- [Estr98] F. Estrella, Z. Kovacs, J-M. Le Goff, R. McClatchey, "The Design of an Engineering Data Warehouse Based on Meta-Object Structures", International Conference on Conceptual Modeling (ER), 1998
- [ETI] <http://www.eti.com/>
- [Foot98] Brian Foote, Joseph Yoder, "Metadata and Active Object-Models", OOPSLA 1998
- [Gamma95] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns", Addison Wesley, 1995
- [Gar95] H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and Jennifer Widom. "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS". In Proceedings of the AAAI Symposium on Information Gathering, pp. 61-64, Stanford, California, March 1995
- [Gar97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, J. Widom. "The TSIMMIS approach to mediation: Data models and Languages". In Journal of Intelligent Information Systems, 1997
- [Gob98] Stefan Göbel Karen Lutze, "Development of meta-databases for geospatial data in the WWW", ACM GIS vol 11, 1998, pp94-99
- [Goff99] J.-M. Le Goff, R. McClatchey, "The CMS ECAL Detector Database System", 1999
- [Goff99+2] J.-M. Le Goff, I. Willers, R. McClatchey, Z. Kovacs, "Design Patterns for Description-Driven Systems", CMS Note 1999

- [Gor97] I. Goralwalla, M.T. Özsu, D. Szafron. "A Framework for Temporal Data Models: Exploiting Object-Oriented Technology", to appear in Proc. of 1997 Conference on Technology of Object-Oriented Languages and Systems (TOOLS USA 97), Santa Barbara, California, July 1997
- [HER] Hermes, <http://www.cs.umd.edu/projects/hermes/index.html>
- [INFO] InfoSleuth, <http://www.mcc.com/projects/infosleuth/>
- [Jens97] C. S. Jensen and R. T. Snodgrass, "Temporal Data Management", IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 1, January/February 1999, pp. 36-44.
- [John98] Ralph E. Johnson, "Dynamic Object Model" 1998 <http://st-www.cs.uiuc.edu/users/johnson/papers/dom/>
- [Kash95] Vipul Kashyap, Kshitij Shah, Amit Sheth, "Meta-data for building the MultiMedia Patch Quilt", 1995
- [Kayser98] D. Kayser, "Ontologically Yours" Invited Talk, ICCS'98, Montpellier, France, LNAI #1453, M.L. Mugnier & M. Chein eds, August 1998
- [Kerh97] Brigitte Kerhervé, Olivier Gerbé, "Models for Meta-data or Meta- models for Data?" , Proceedings IEEE Meta-data conference, 1997
- [Kerh98] Brigitte Kerhervé, Olivier Gerbé, "Modeling and Meta- modeling Requirements for Knowledge Management" , workshop Oopsla98 ,1998
- [Kova99] Zsolt Kovács, "The Integration of Product Data with Workflow Management Systems Through a Common Data Model", dissertation of Faculty of Computer Studies and Mathematics, University of the West of England, Bristol, 1999
- [Kuts99] Ralf-Detlef Kutsche, Asuman Sünbül, "Meta-Data Based Development Strategy for Heterogenous, Distributed Information Systems" , proceedings of IEEE Meta-data Conference, 1999
- [Levy99] Alon Levy, "More on Data Management for XML" , 1999
- [MER] Merlin <http://id.inel.gov/merlin/index.html>
- [MIX] MIX, <http://www.db.ucsd.edu/Projects/MIX/>
- [MIW] <http://www.meta-integration.net/Products/MIW/Description.html>
- [Morg96] Matthew Morgenstern, "An Integration Platform for Heterogenous Databases: The DAISy System" a DARPA funded project, 1996
- [Nak98] Hiroaki Nakamura and Ralph E. Johnson , " Adaptive Framework for the REA Accounting Model" , OOPSLA'98 Business Object Workshop IV
- [Nasa99] <http://www.nas.nasa.gov/ipg/ImplementationPlan/IPG.Plan.pub.pdf>
- [OAS] Robin Cover, "The SGML/XML Web Page" , <http://www.oasis-open.org/cover/xml.html>
- [OMG97] OMG, "Unified Modeling Language UML Notation Guide" , AD/97-08-05, Object Management Group, Framingham, Mass., November 1997
- [OMG99] Object Management Group, "Meta- Object Facility (MOF) Specification" , <http://www.omg.org/cgi-bin/doc?ad/99-09-04>
- [Pep97] Pepijn R.S. Visser, Dean M. Jones, T.J.M. Bench-Capon and M.J.R. Shave, "An Analysis of Ontology Mismatches; Heterogeneity versus Interoperability" , AAAI spring symposium on Ontological Engineering, Standford University USA, 1997
- [Ppdg99] Particle Physics Data Grid, <http://www.cacr.caltech.edu/ppdg/>
- [Ram99] Sudha Ram, Jinsoo Park, George L. Bal, "Semantic-Model Support for Geographic Information Systems" , Computer IEEE 1999
- [Riehle98] Dirk Riehle, "Why a Bank Needs Dynamic Object Models" , position paper OOPSLA 98
- [Robie98] Jonathan Robie, Joe Lapp, David Schach, "XML Query Language (XQL)" , WWW3C proposal, 1998
- [Snod92] C.S. Jensen, J.Clifford, S.K.Gadia, A.Segev, R.T.Snodgrass, "A Glossary of Temporal Database Concepts" , SIGMOD Record, Vol. 21, no 3 1992 7
- [Snod95] Gultekin Ozsoyoglu Richard T. Snodgrass "Temporal and Real-Time Databases: A Survey" IEEE Transactions for Knowledge and Data Engineering, August 1995

- [Sow84] Sowa, J.F., “*Conceptual Structures: Information Processing in Mind and Machine*” 1984, Addison-Wesley.
- [Ste97] Andreas Steiner, Moira C. Norrie, “*Implementing Temporal Databases in Object-Oriented Systems*”, 5th International Conference on Database Systems for Advanced Applications, 1997
- [Ste97+2] A. Steiner and M. C. Norrie, “*A Temporal Extension to a Generic Object Data Model*”, Technical report 1997.
- [Sub99] V.S. Subrahmanian, Sibel Adali, Anne Brink, Ross Emery, James J. Lu, Adil Rajput, Timothy J. Rogers, Robert Ross, Charles Ward” *HERMES: Heterogeneous Reasoning and Mediator System* “ Fourth Annual Electrical Engineering, Computing and Systems Research Review Day Friday, April 30, 1999
- [TSIM] TSIMMIS, <http://www-db.stanford.edu/tsimms/tsimms.html>
- [Wari98] P.C.H. Wariyapola, S.L.Abrams, A.R.Robinson, K.Streitlien,N.M.Patrikalakis,P.Eliseeff,H.Schmidt, “*Ontology and Meta-data Creation for the Poseidon Distributed Coastal Zone Management System*”, 1998
- [Widom95] Jennifer Widom,”*Research problems in Data Warehousing*”, ACM,1995
- [Widom99] Jenifer Widom,”*Data Management for XML*”, IEEE Data Engineering Bulletin, Special Issue on XML, 22(3) 44-52, september 1999
- [Wie92] Gio Wiederhold, “*Mediators in the Architecture of Future Information Systems*”, IEEE computer magazine, 1992
- [Wie97+2] Gio Wiederhold, Joshua Hui, Bhujanga Panchapagesan, Stephan Erickson, Lynn Dean, “*The INEEL Data Integration Mediation System*”, White paper, <http://id.inel.gov/merlin/index.html> , 1997
- [Wie99] Gio Wiederhold, “*Mediation to Deal with Heterogenous Data Sources*”, Interop Zürich, 1999
- [Wil98] Peter R. Wilson “*STEP and EXPRESS* “ (An additional contribution to an Invitational NSF Workshop on Distributed Information, Computation and Process Management for Scientific and Engineering Environments, 15--16 May, 1998.)
- [Wil99] Roy Williams, Paul Messina, Fabrizio Gagliardi, John Darlington, Giovanni Aloisio, “*Large Scientific Databases*” workshop Annapolis, Maryland, USA, 1999 September 8-10
- [Witt98] Weerasak Witthawaskul,” *Position Statement: Active Object-Model*”, OOPSLA 1998
- [Wright99] Ann M. Wrightson, “*XML study notes*”, <http://helios.hud.ac.uk/staff/scomaw/xmlstudy/xminotes.htm> , 1999