The Challenge of Process Data Warehousing

Matthias Jarke [1, 2] Thomas List [1] Jörg Köller [1]

[1] RWTH Aachen, Informatik V (Information Systems), Ahornstr. 55, 52056 Aachen, Germany
{jarke,list,koeller}@informatik.rwth-aachen.de

[2] GMD-FIT, Schloss Birlinghoven, 53754 Sankt Augustin, Germany

Abstract

The management of organizational knowledge is becoming a key requirement in many engineering organizations. In many cases, it is difficult to capture this knowledge directly, as it is hidden in the way-of-working followed by networks of highly qualified specialists. Moreover, much of this knowledge is strongly context-dependent, so rules to be followed must be augmented by adequate situation analysis. Hardware and software tools used to support these processes are strongly heterogeneous, involving significant effort of usage and very different kinds of data.

In this paper, we propose *process data warehouses* as a means to remedy these problems. A process data warehouse, according to our approach, is centered around a knowledge-based metadata repository which records and drives a heterogeneous engineering process, supported by selected materialized instance data. We follow a concept-centered approach expanding ideas from the European DWQ project and illustrate our solution with a prototypical process data warehouse for chemical engineering design developed within the Collaborative Research Centre IMPROVE at Aachen University of Technology.

1. Introduction

Data warehouses have established themselves in the information flow architectures of business organizations for two main reasons: firstly, as a *buffer* between operational and transactional tasks on the one hand, and analytical strategic tasks on the other. Secondly, to capture the *history* of business transactions for purposes of archiving, traceability, experience mining and reuse.

In this paper, we claim that the same basic arguments apply to engineering applications. In these applications, the *buffer function* of data warehousing may be even more important. Research results are often obtained by expensive simulations or even more expensive laboratory experiments, such that analytic processing on demand from information sources is only possible with exceptional effort.

Similarly, from the viewpoint of *history management*, many engineering organizations complain that simulations and experiments are repeated unnecessarily, or at least, that too few lessons for analogous cases concerning promising or useless simulation/experimentation are drawn beyond the experiences of individual engineers. Several organizations are therefore embarking on large-scale traceability or process-capture programs [Ros98, Ram98]; other organizations pursue the introduction of large-scale document management systems [GSS99] (e.g. the Documentum product) that make at least a coarse-grained representation of products and processes available electronically.

This trend is particularly strong in the research-intensive and law-suit prone process industries (chemicals, oil, food, pharmaceuticals, biotechnology) where global competition with many mergers is largely decided by timely and cost-effective invention of novel products with high market potential. In these industries, data exchange standards, interoperation standards, web-based information distribution and portals, groupware and workflow are being developed within companies and on a scale of worldwide cooperation and competition. However, few (if any) coherent approaches have emerged.

Process data warehousing is proposed as a solution strategy for some of these issues. We define a process data warehouse (PDW) as a data warehouse which stores histories of engineering processes and products for experience

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 2000

reuse, and provides situated process support. According to our approach, a PDW synchronizes features from document management systems, engineering databases, and traceability tools through active repository technology. To make the idea more precise, we report in this paper about a research effort in which we extend a recent approach to the design of data warehouses (developed in the European DWQ project [JaVa97]) to the case of process data warehousing in the domain of chemical engineering. This research is carried out in the context of the IMPROVE Collaborative Research Center at RWTH Aachen [NaWe99] which investigates IT support for cooperative chemical engineering, and the European CAPE-OPEN initiative in which the chemical industries worldwide are attempting to standardize open interfaces for process simulation software [Bra*99]. We are aware of at least two major German organizations in the chemical and pharmaceutical industry where similar systems are in progress, in part based on earlier research results of ours. It has to be stressed that our experiences are still preliminary and point out more challenges than actual solutions – therefore the title of the paper.

The paper is structured as follows: In section 2, we summarize the DWQ method for concept-driven data warehouse development. In section 3, we point out the major extensions required for process data warehousing in the chemical engineering domain and describe the principle solution strategies followed in the above-mentioned projects. These strategies are illustrated, in section 4, by a demonstration scenario we have implemented in the IMPROVE environment. Lessons learned and a summary of the remaining challenges conclude the paper.

2. The DWQ Approach

A large body of literature addresses the problems introduced by the data warehouse approach, such as the trade-off between freshness of data warehouse data and disturbance of OLTP work during data extraction; the minimization of data transfer through incremental view maintenance; and a theory of computation with multi-dimensional data models. For an overview, see [JLVV99].

The goal of the European DWQ project [JaVa97] was to study a systematic methodology for data warehouse design by developing, prototyping and evaluating comprehensive Foundations for Data Warehouse Quality, delivered through *enriched metadata management facilities* in which specific analysis and optimization techniques are embedded.

The main difference of the DWQ architecture to the standard data warehouse architecture in the literature [CGH+94] is the addition of an explicit *conceptual perspective* according to which the goals of the data warehouse, the quality of the available sources, and the interests of the clients can be characterized. Such a concept-driven architecture is of particular relevance in our process engineering context where data representations differ so widely that their interrelationships can only be described at a conceptual level.

The DWQ method consists of the six steps shown in figure 1. In this figure, the dark-grey boxes indicate logical data objects, such as relations, queries, or materialized (possibly multi-dimensional) views. The light-grey boxes describe conceptual models; in DWQ, these are externally represented as extended Entity-Relationship models, internally modelled using Description Logic formalisms from artificial intelligence to allow for subsumption reasoning. The two ovals describe related support at the operational level which we do not discuss further in this paper: aggregate query optimization and view refreshment. The whole process is administered through a metadata repository which has been implemented using the ConceptBase system [JGJ+95].

In the following subsections, we briefly describe the main steps and point to literature where more details about the underlying theory or applications can be found.

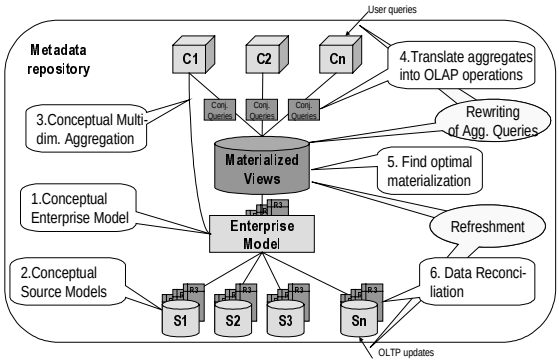

Figure 1: DWQ Data Warehouse Development Process

2.1 Source Integration (Steps 1 and 2)

The DWQ approach to source integration is incremental: Whenever a new portion of a source is taken into account, the new information is integrated with an "Enterprise Model", and the necessary new relationships are added. The main concepts used in DWQ source integration are shown in figure 2 (a corresponding mapping exists on the client side of the data warehouse):

- The *Enterprise Model* is a conceptual representation of the global concepts and relationships that are of interest to the data warehouse application. It provides a consolidated view of the concepts and relationships that are important to the enterprise, and have so far been analysed. Such a view is subject to change as the analysis of the information sources proceeds. The Description Logic formalism we use is general enough to express the usual database models, such as the Entity-

Relationship Model, and the Relational Model [CDL+99]. Inference techniques associated with the formalism allow for carrying out several reasoning services on the representation. The formalism is hidden from the user of the DWQ tools who only uses a graphical interface.

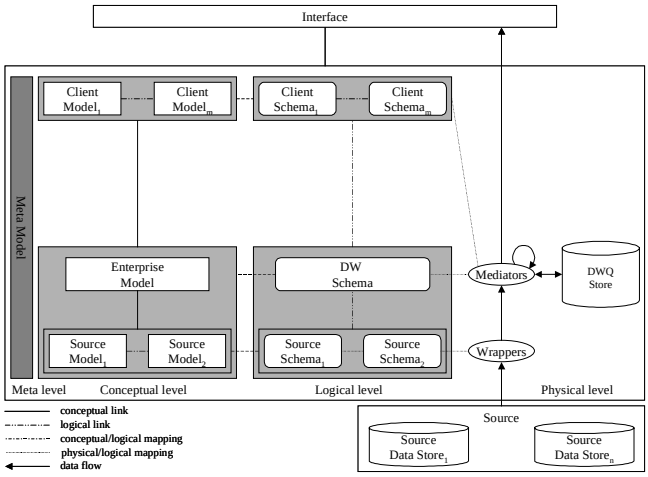

Figure 2: DWQ Architecture for Data Integration

- For a given information source S, the *Source Model* of S is a conceptual representation of the data residing in S. Again, the approach does not require a source to be fully conceptualized. Source Models are expressed by means of the same formalism used for the Enterprise Model.
- Integration does not simply mean producing the Enterprise Model, but rather being able to establish the correct relationships both between the Source Models and the Enterprise Model and between the various Source Models. We formalize the notion of interdependency by means of *intermodel assertions* [CL93]. An intermodel assertion states that one object (i.e., class, entity, or relation) belonging to a certain Model (either the Enterprise or a Source Model) is always a subset of an object belonging to another Model. This simple declarative mechanism has been shown to be extremely effective in establishing relationships among different database schemas (see also [Hul97]). We use a logic-based formalism to express intermodel assertions, and the associated inference techniques provide a means to reason about interdependencies among models.
- The logical content of each source S, called the *Source Schema*, is provided in terms of a set of definitions of relations, each one expressed as a query over the Source Model of S. The logical content of a source represents the structure of data expressed in terms of a logical data model, such as the Relational Model. The logical content of a source S, or of a portion thereof, is described in terms of a view over the Source Model associated with S (and, therefore, of the Conceptual Data Warehouse Model). Wrappers map physical structures to logical structures.
- The logical content of the materialized views constituting the Data Warehouse, called the *Data Warehouse Schema*, is provided in terms of a set of definitions of relations, each one expressed in terms of a query over the Conceptual Data Warehouse Model. How a view is actually materialized starting from the data in the sources is specified by means of *Mediators*.

The following tasks work on this structure within steps 1 and 2 of figure 1:

- *Enterprise and Source Model construction.* The Source Model corresponding to the new source is produced, if not available. Analogously, the conceptual model of the enterprise is produced, if not available.
- *Source Model integration.* The Source Model is integrated into the Conceptual Data Warehouse Model. This can lead to changes both to the Source Models, and to the Enterprise Model. Moreover, intermodel assertions between the Enterprise Model and the Source Models and between the new source and the existing sources are added to the Conceptual Data Warehouse Model. The designer can specify such intermodel assertions graphically, and can invoke various automated analyses supported by the Description Logic formalization.
- *Source and Data Warehouse Schema specification.* The Source Schema corresponding to the new source (or, corresponding to a new portion of the source) is produced. On the basis of the new analysed source, an analysis is carried out on whether the Data Warehouse Schema should be restructured and/or modified.

In all these tasks, the metadata repository stores the values of the quality factors involved in source and data integration, and helps analyse the quality of the design choices. The Quality Factors of the Conceptual Data Warehouse Model and the various schemas are evaluated and a restructuring of the Models and the schemas is accomplished to match the required criteria.

2.2 Multidimensional Aggregation and OLAP Query Generation

The next two steps consider the client side, symmetric to the source integration; they will therefore be described in less detail.

Step 3 in Figure 2: The conceptual modeling language underlying the enterprise and source models, and the corresponding modeling and reasoning tools, have been extended to the case where concepts are organized into aggregates along multiple dimensions with multi-hierarchy structure [FS99]. Data warehouse designers can thus define multi-dimensional and hierarchical views over the enterprise conceptual model in order to express the inter-

ests of certain DW client profiles, without losing the advantages of consistency and completeness checking as well as semantic optimisation provided by the conceptual modeling approach.

Step 4: The thus defined "conceptual data cubes" can either be implemented directly by MOLAP data models, or supported by a ROLAP mapping to relations. Faithful representation of client views requires a careful design of an OLAP relational algebra, together with the corresponding rewritings to underlying star schemas [Vass98].

2.3 Design Optimization and Data Reconciliation

Step 5: Viewed from the logical level, our conceptually controlled approach to source integration has created a schema of relations implementing the enterprise model. This schema could be implemented directly as an operational data store (ODS). In a data warehouse with lots of updates and completely unpredictable queries, this would be the appropriate view materialization.

Conversely, the mapping of multi-dimensional aggregates to ROLAP queries creates a set of view definitions (queries). The materialization of these queries would be the optimal solution (storage space permitting!) in a query-only data warehouse with hardly any updates.

Typical data warehouses have less extreme usage patterns and therefore require a compromise between the two view materialization strategies. The DWQ project has investigated solutions to this combinatorial optimisation problem [ThSe97, LiTS98].

Step 6: The physical-level optimisation is fully integrated with the conceptual modeling approaches because it works on their outcomes. Conversely, the resulting optimal design is now implemented by data integration and reconciliation algorithms, semi-automatically derived from the conceptual specifications. The views to be materialized are initially defined over the ODS relations. There can be several qualitatively different, possibly conflicting ways to actually materialize these ODS relations from the existing sources. These ways are generated by a further set of rewritings that can be derived from the source integration definitions [CDL+99].

The problem of data reconciliation arises when data passes from the application-oriented environment to the Data Warehouse. During the transfer of data, possible inconsistencies and redundancies are resolved, so that the warehouse is able to provide an integrated and reconciled view of data of the organization. In the DWQ methodology, data reconciliation is based on (1) specifying through Interschema Assertions how the relations in the Data Warehouse Schema are linked to the relations in the Source Schemas, and (2) designing suitable mediators for every relation in the Data Warehouse Schema. In step (1), interschema correspondences are used to declaratively specify the correspondences between data in different schemas (either source schemas or data warehouse schema). Interschema correspondences are defined in terms of relational tables, similarly to the case of the relations describing the sources at the logical level. [CDL+99] distinguish among three types of correspondences, namely Conversion, Matching, and Reconciliation Correspondences. By virtue of such correspondences, the designer can specify different forms of data conflicts holding between the source, and can anticipate methods for solving such conflicts when loading the Data Warehouse. In step (2), the methodology aims at producing, for every relation in the Data Warehouse Schema, a specification of the corresponding mediator, which determines how the tuples of such a relation should be constructed from a suitable set of tuples extracted from the relations stored in the sources.

3. Extensions for Process Data Warehousing: The Case of Chemical Engineering

The DWQ approach assumes a business data warehouse setting with a coherent conceptual model and limited heterogeneity in the data sources and client applications, based on relational or possibly semi-structured data models.

In this section, we discuss the extensions we found necessary to support process data warehousing in the chemical engineering domain. These extensions are mainly twofold. At the conceptual level, the "enterprise model" of the DWQ approach has to be split into a set of loosely connected partial models which look at different facets of the chemical engineering process. Coherence between these partial models is achieved through the so-called process flowsheet, a data structure (and visualization) which turns out to be the central communications medium between the different kinds of specialists cooperating in a design process. This is the key access structure to heterogeneous sources and documents.

At the logical and physical level, heterogeneity of the process engineering tools is far greater than traditionally considered in OLTP data sources. Therefore, an intermediate standardization step is necessary, not only at the level of data but also at the level of services; this is due to the fact that often the data of engineering tools are not sensibly accessible directly but only via the tool services.

3.1 Partial Models in Chemical Process Engineering

Chemical engineering is the combination of physical, chemical, biological, and informational operations on a chemical plant with the aim of transforming input materials in a manner that a material product with desirable properties concerning type, behaviour, and composition results. Besides this main goal, the chemical engineering process is heavily influenced by considerations of cost and time, but also by environmental side effects, such as energy consumption, detrimental side products, water heating or pollution, and the like.

Process engineering encompasses the handling of these processes in all stages from the early design phase to the operation of a plant. The overall development of a chemical process is a complex task that starts with the conceptual design. It does not end with the mapping to physically available equipment for operation and control, but accompanies the whole lifecycle, often over decades.

The decisions made in the different design steps depend on various factors: Chemical properties of the materials involved in the process (reactions, temperature, pressure,...), technical feasibility, environmental and safety issues, cost and time. To assess all these properties, mathematical simulation models are built and calculated in the design phase. These models may differ widely in complexity and the kind of data they produce or need. Hence, the tools involved in the design process are highly heterogeneous. This includes flowsheeting tools for conceptual design, process simulators for predicting the reactions and behaviour of the chemical components in a process or numerical solvers for the large non-linear problems typically occurring in this context.

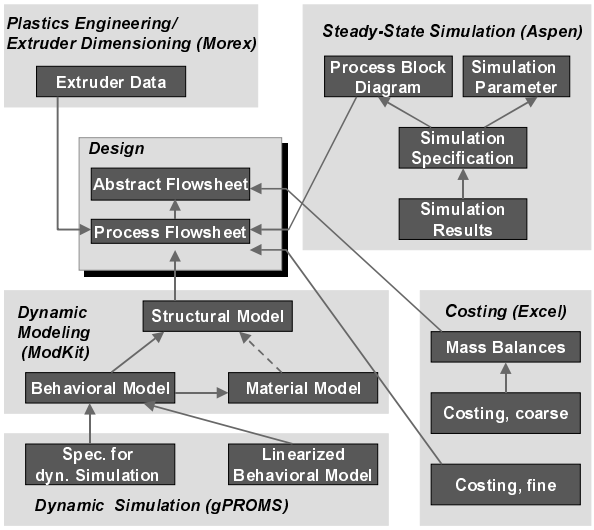

Figure 3: Partial Models and Tools

At the conceptual level considered in the DWQ approach, this heterogeneity implies that a coherent "enterprise model" cannot easily be built. Instead, a multitude of partial information models is being considered with poorly understood interconnections. In IMPROVE, such information models are being systematically developed for the important process engineering perspectives [BaSM99]. However, empirical studies of process engineering work demonstrate that one family of closely related sub-models, visualized through flowsheets of the chemical process, has a clearly dominating role in the communication between different designers. Our modeling strategy, described in more detail in [JLW99], has therefore been to use a conceptual model of flowsheeting as the kernel of a metadata model to which all other information models are related. To illustrate the central role of the flowsheet, figure 3 shows the partial models (and related tools) considered in the IMPROVE demonstrator.

In the simplest case, a flowsheet looks like a kind of data flow diagram with a lot of fancy graphical types for different kinds of devices (corresponding to processes) and connections (corresponding to dataflows). However, this simple analogy is complicated by at least three factors. These factors are illustrated in figure 4 that instantiates some aspects of figure 3.

Firstly, flowsheets may describe very complex processes and evolve in complex refinement structures, including operations such as enrichment of object definitions, decomposition of functions, specialization of choices, and realization of functions by (combinations of) device types. This can be seen in the middle of figure 4.

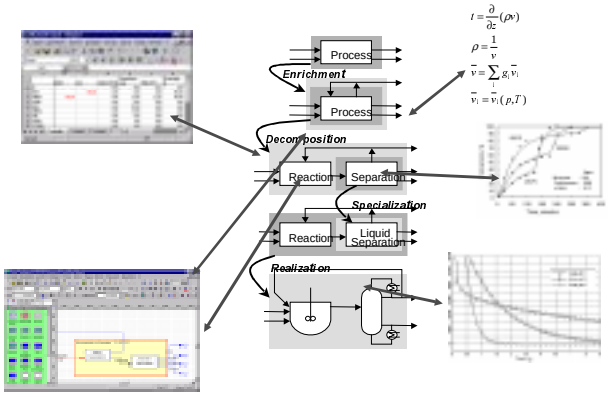

Figure 4: Flowsheet Hierarchy connected to Heterogeneous Data

Second, process synthesis decisions are made under uncertainty about their impact. A complete analysis of all design choices is impractical due to the high effort in setting up simulations or laboratory experiments. However, decisions under uncertainty may mean backtracks in the engineering process, especially if not accompanied by deep experience or available theory. The heterogeneity of representations required is illustrated in the figures surrounding the conceptual model hierarchy in figure 4.

Last not least, there is a huge and continuously growing number of different devices, connections, and specialized functions that can be used in chemical engineering; estimates speak about roughly 50.000 types to be considered. The information models of a meta database for process data warehousing must therefore be easily extensible by new product and process knowledge, and cannot be mapped one-to-one in tool functionality. Indeed, even the storage, search and consistency analysis of such large schemata becomes a problem [BaJa99, Satt98].

Summarizing, the requirements of the process engineering domain strongly support the case of a concept-driven approach as proposed in DWQ. They also require further refinements of metadata handling concerning information model integration, structural and behavioural refinement, the interplay of design and analysis, and the extensibility with a growing body of knowledge. In section 4, these issues are demonstrated through an "extended situation analysis" function where only basic structural knowledge is hard-coded in design tools such as a flowsheet editor. Detailed domain knowledge is available in declarative form in the metadata repository of the process data warehouse. The situation knowledge required for context-adequate application of the domain knowledge is collected from either the PDW itself or by callback queries to other engineering tools.

3.2 Dealing with Heterogeneity at the Technical Level

The diversity of information models at the conceptual level is exacerbated by diversity of data formats and service accessibility at the technical level of chemical engineering tools and materials databases. Current process engineering environments often hide this problem through monolithic software architectures with fixed means of access. These make it close to impossible to include company-specific knowledge or home-grown specialist tools.

The European process industries have therefore embarked on the CAPE-OPEN initiative [BJM+99, GCO00] in order to accomplish a standardization of simulation interfaces, such that a component-based approach can be followed. This standard has been defined at the conceptual level through UML models – semi-formal source models in the sense of the DWQ approach. At the implementation level, the standard is defined in both DCOM [MS00] and CORBA [Vin97,OMG]. In IMPROVE, we are using the CORBA version. Both the conceptual aspect and the tool service integration are illustrated below, focussing on the aspect of heterogeneous process simulation as an example.

Process simulators are complex software systems designed for creating mathematical models of manufacturing facilities for processing and/or transforming materials. Simulation allows the chemical engineer to interactively predict the behaviour of an existing or proposed chemical process. It enables the assessment of process alternatives with respect to economic, safety, and environmental performance without actually building a plant, thereby reducing cost and speeding up the design phase of the plant. Process simulators follow different internal architectures, e.g. block modular systems, equation based systems and simultaneous modular systems. Surveys of current approaches for process modeling and simulation can be found in [PB94, Maq95, JM96].

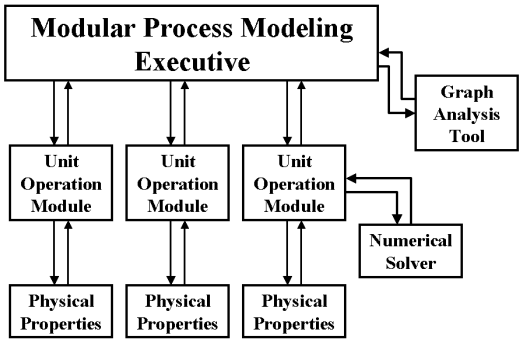

Figure 5: CAPE-OPEN Standard Components

As illustrated in figure 5, CAPE-OPEN has identified the following standard components of a process simulator from a conceptual point of view [CO98]:

- **Unit Operation Modules,** often just termed units, represent the behavior of physical process steps (e.g. a mixer or a reactor). They are linked to the simulation flowsheet which represents an abstraction of the plant structure. They compute the quality of a material stream of their outlet ports if the according information is given at the inlet ports. The simulation models are assembled from predefined libraries of unit operation modules into a flowsheet which represents the overall plant.
- **Physical Properties (Thermodynamics) Packages:** An important functionality of a process simulator is its ability to calculate thermodynamic and physical properties of materials (e.g. density or boiling point). Thermodynamic packages are complex and highly optimized pieces of software. Typically, they consist of a database containing a lot of simple properties for a set of chemical species. Based on these such packages offer FORTRAN or C programs to calculate more complex properties using the properties in the database.
- **Numerical Solvers:** The mathematical process models of a unit operation or a complete plant are large and highly non-linear. An analytical solution is impossible. Therefore iterative, numerical approaches are used to either solve the equations of a single unit operation module or to solve the overall flowsheet.
- **Simulator Executive:** This is the simulator's core which controls the set-up and execution of the simulation, i.e. analyzing the flowsheet and calculate the units. Furthermore, it is responsible for a consistent flowsheet set-up and error checking.

Figure 6 shows the set-up of a prototypical CORBA-based CAPE-OPEN compliant simulator we built jointly with Aachen's process engineering group. The distributed objects communicate via the CORBA object bus using the CAPE-OPEN standard interfaces.

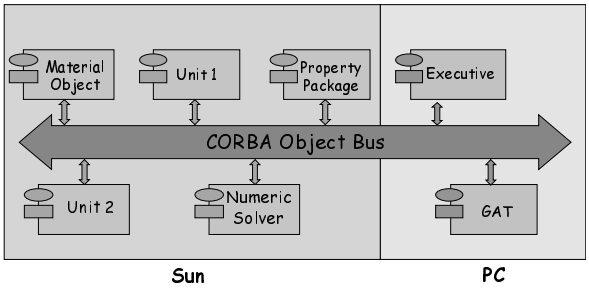

Figure 6: CAPE-OPEN CORBA Prototype Set-Up

Two different units are linked by a flowsheet in the executive component. To solve the flowsheet, i.e. run the simulation, the executive calls the graph analyzer (GAT) component to determine the order of unit calculations to be done. Then all unit simulation models are called in the order calculated by the GAT. To enable the units to calculate themselves, the executive has to configure them and to provide input values. It sends CORBA references for property packages and numerical solvers to the units which it has obtained from a component repository provided by the CORBA system. Now the units are ready to perform thermodynamic calculations (using the properties package) and solve their internal mathematical models.

Before a unit actually can be calculated it has to know what chemical components are at its input ports. This information is also provided by the Executive by creating a Material Object and handing over its reference to the unit. The Material Object is the central mechanism of exchanging complex data between the different CAPE-OPEN components and encapsulates all kind of chemical properties data. It is also used for data exchange to a property package. After the unit calculation is finished the unit creates new material objects carrying information about the materials at the units output ports. Then this material object is handed over by the executive as input data for the next unit in the flowsheet.

The example implies that the use of CAPE-OPEN components in a process data warehouse requires explicit representations of the material object, the units, and the property packages. In section 4, we show how these components can be conceptually embedded in our approach.

3.3 A Metadata-Driven Architecture

As an additional complication over the basic DWQ approach, chemical engineering tools often cover aspects from more than one partial model within the enterprise model. The mapping between logical and conceptual representations becomes considerably more complex. It is not even clear whether we have a data-to-data mapping at all, or whether it is more appropriate to think of engineering tools as document-producing and document-consuming objects, such that process data warehousing becomes an issue of content extraction from document histories rather than a data model mapping with incremental updates.

Indeed, traceability reference models drawn from studies in large software organizations – a vaguely similar domain – indicate that, when capturing an engineering process, we must model three main aspects [RJ00]: how the engineering product evolves through decisions made on content objects, in what media and with which persistence and legal value this is documented, and what is the contribution structure of stakeholders and analysts who made the content-oriented and administrative decisions. The resulting highest-level meta model of a process data warehouse is shown in figure 7.

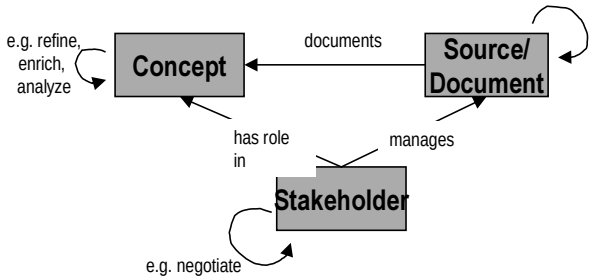

Figure 7: Meta Meta Model of Process Data Warehouse

Our implementation strategy foresees to integrate commercial tools available for data management (e.g. an object relational database or specialized engineering databases such as Comos Pt [Inn00]), document management (e.g. Lotus Notes, Documentum), and trace/dependency management (e.g. DOORS, SLATE) with the wrapped and mediated tool data via a knowledge-based metadata repository. The implementation of this full architecture is still in progress. In the prototype illustrated in the next section, only a product database and a couple of chemical engineering tools have been interfaced with the metadata repository, while others have been emulated by putting example instances in the metadata repository itself.

4. Application example

In this section, we present an example that shows how the standard interfaces from CAPE-OPEN supplement the process data warehouse in the chemical engineering domain. Our prototype combines techniques for integrating the highly heterogeneous information sources in the application domain with the standard interfaces for unit operations and physical properties packages. The process data warehouse client that operates on the prototype (called "cross-tool situation analysis") uses the domain knowledge captured in the meta database to give guidance to the chemical engineer via a process-integrated flowsheet tool [JLW99], by analysing the current development situation via the product state of several source database or tools.

4.1 Scenario

In the early conceptual design stage a chemical engineer draws a flowsheet of the plant. The blocks in the flowsheet (called devices) represent functions, such as mixing, reaction or separation. These devices are then further decomposed and realized in concrete apparatuses.

Figure 9: Conceptual Flowsheet of the example

As a (very simple) example, we consider the flowsheet in figure 9. Two of the three devices (mixer and CSTR - continuous stirred tube reactor) are already given through unit operations that realize specific functions (mixing and reaction). The developer's task is to find a suitable realization for the third device (Separation). This realization can range from a single device to a complex combination of different devices, with or without backflow into earlier devices. The detailed setting is: Two input streams feed an initial mixer. The substances A and B are fed through the streams into a mixing device. The mixture is then fed into a reactor of type CSTR. The result is the product C and the (unwanted) by-product D. These substances now have to be separated. Assume that the mixing device is completely specified. The unit to simulate the reactor is known (including the equations for the reaction). Now the task of the chemical engineer is to find a useful realization for the separation unit.

The extended situation analysis function of the process data warehouse is able to provide some hints of which unit operations should be considered for this task if some of the properties of the reactor's output stream (containing C and D) are known: temperature and pressure of the mixture in the stream, the fraction of each substance in the stream and its boiling temperatures.

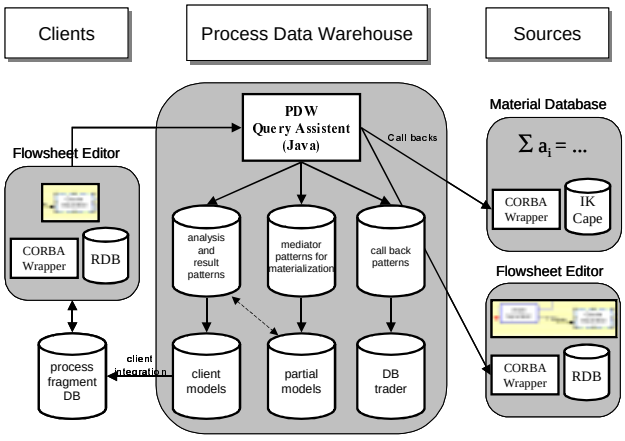

Figure 10: Architecture of the Process Data Warehouse

Figure 10 describes the functionality of the cross-tool situation analyser of the warehouse. A client tool (here: the flowsheet editor) calls the PDW Query Assistant, a Java-based control program. The call contains an identifier for the selected flowsheet element (the separation device) and an operation to be executed on the device (realize). The warehouse itself contains several sub-meta databases that are queried during the processing of the call:

- **Call back patterns** are used to determine which additional information is needed to answer the request.
- The **DB trader** contains information from which tool or database and how this information can be accessed. The calling client will be one of the tools that are accessed. In the example, a larger part of the flowsheet is needed to classify the context of the call.
- The **mediator patterns for materialization** are then used to materialise the additional data into the data warehouse such that they become instances of the **partial models** of the data warehouse. See section 4.3 for details of these integration steps.
- The **analysis and result patterns** are now applied to calculate useful results for the original query.
- The presentation of the results is highly dependent on the client tool that initiated the query. The **client model** is used to transform the result in a suitable form. The special process integrated features of the flowsheet tool can be used to directly insert a proper refinement of the separation into the flowsheet. Details on the flowsheet tool and its interaction with the process data warehouse can be found in [JLW99].

These databases are not accessed sequentially but in a nested way so that only these information sources are accessed that are needed to answer the specific query.

The call back queries used in these steps are not purely queries to source databases. For example, the needed simulation results of the reactor are results of an aggregation function. In this sense the results are the results of a (highly complex) query on the data warehouse store. As simulating is a time consuming and expensive task we also store the results in the data warehouse for reuse. To gain access to the units the DB trader contains meta information about the CAPE-OPEN components. As a result of the usage of the CAPE-OPEN compliant units we do not need to handle very different simulators such as Aspen-Plus, Pro/II or gProms but we have to create the CAPE-OPEN objects used by units. This is especially the material object for each substance contained in the input ports of the unit. The process data warehouse produces these CORBA objects and is then able to start the simulation of the unit.

4.2 Applying the DWQ approach

For the example application the partial models *plant* (for the flowsheet) and *material concept* (for the material properties) are of interest (Fig. 11 & 12, we omit most of

the attributes and the specializations of `ProcessStep`). The results of the simulation of the reactor have the form of material objects and thus refer to the material concept partial model. More complex simulations than those of a single unit would be represented in an own partial model for simulation results. The two partial models are connected via the *abtract plant model* that states that `Stream` is an aspect of an `abstract_stream` and therefore can contain a `material_concept`. The EER-models form a part of the conceptual enterprise model in figure 2.

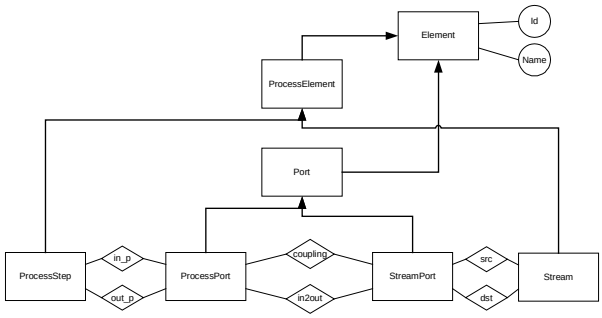

Figure 11: The Plant Partial Model

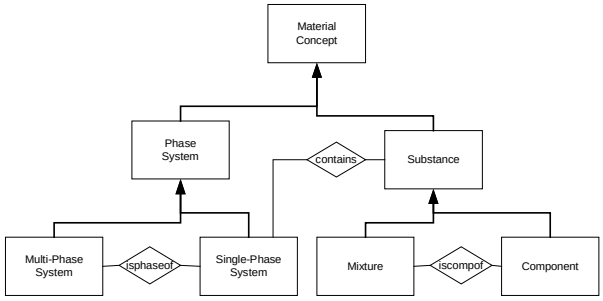

Figure 12: The Material Concept Partial Model

4.3 Source Integration

The first step in integrating data sources (such as the flowsheet editor) into the data warehouse is to reverse engineer the conceptual model from the data source. As the flowsheet editor is not a relational database but a technical tool, we built a CORBA-wrapper to access its data. This wrapper simplifies the internal data structure used by the tool by suppressing some constructs only needed for implementation purposes. The CORBA IDL now reflects some relational structure that can be reverse engineered to the EER diagram in figure 13. This diagram shows that the data in the flowsheet editor covers (parts of) both partial models from figure 11 and figure 12.

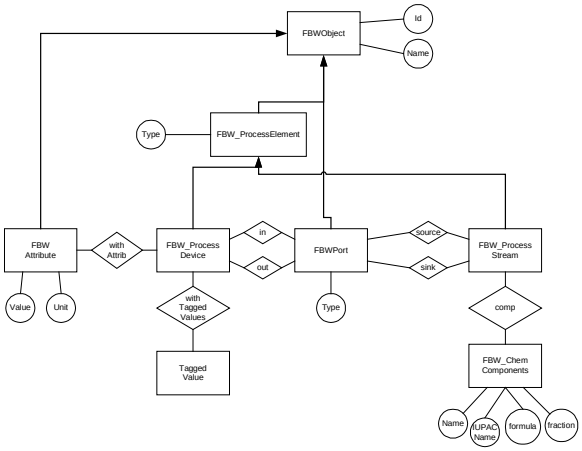

Figure 13: Conceptual Model of the Flowsheet Editor

In addition the flowsheet structure is different. There are no stream ports – these are important if a tool regards streams as decomposable objects, which the flowsheet editor doesn't. The flowsheet editor simply uses tagged values instead of a specialization hierarchy to specify the type of flowsheet devices (*FBW_ ProcessDevice*). The domain knowledge coded into the flowsheet editor was intentionally kept low because of the volatility of domain knowledge. The conceptual model is now transformed into a description logic formalism [CDL+98] and enriched through the definition of intramodel assertions that define some constraints not captured by the EER diagram.

The other information source – the physical properties package of the CAPE-OPEN standard – is seen as a collection of *material objects*. The EER model of that source consists of exactly one entity and a set of attributes. This data source in special in the way that it calculates some of its data on demand. This fact is hidden by the wrapper!

We specify the relations of the data sources in terms of adorned Datalog-like queries [CDL+99]. This step is needed because the mapping from the EER model to the relational model is ambiguous. Figure 14 shows an extract from the mapping of the flowsheet editor. The query for `FBWStream_R` is an example where several relationships and entities are mapped onto a single relational table.

```
FBWObject_R(I,N) <-
   FBWObject(X),Id(X,I),Name(X,N) |
       identify([I],X), I::ObjIDs,N::ObjNames

FBWPort_R(I,T) <-
  FBW_Port(X),Id(X,I),PortType(X,T) |
  identify([I],X), I::ObjIDs, T::PortTypes

FBWProcessElement_R(I,T) <-
  FBW_ProcessElement(X),Id(X,I),
  ElementType(X,T) |
  identify([I],X), I::ObjIDs,
  T::ProcessElementTypes
```

```
FBWDevice_R(I) <-
FBW_ProcessDevice(X),Id(X,I) |
identify([I],X), I::ObjIDs

FBWStream_R(I,SOURCE_PORT,SINK_PORT,
  SOURCE_DEV,SINK_DEV) <-
  Source(S,P1),Sink(S,P2),Out(D1,P1),
  In(D2,P2),Id(S,I),Id(P1,SOURCE_PORT),
  Id(P2,SINK_PORT),Id(D1,SOURCE_DEV),
  Id(D2,SINK_DEV)
     OR
  Stream(S),
  NOT Source(S,P),Sink(S,P2),In(D2,P2),
  Id(S,I),SOURCE_PORT=NULL,Id(P2,SINK_PORT),
  SOURCE_DEV=NULL,Id(D2,SINK_DEV)
     OR
  Stream(S), Source(S,P1),NOT Sink(S,P),
  Out(D1,P1),Id(S,I),Id(P1,SOURCE_PORT),
  SINK_PORT=NULL,Id(D1,SOURCE_DEV),
  SINK_DEV=NULL
     OR
  FBW_Port(P),FBW_Port(PP),NOT Source(S,P),
  NOT Sink(S,PP),Id(S,I),SOURCE_PORT=NULL,
  SINK_PORT=NULL,SOURCE_DEV=NULL,
  SINK_DEV=NULL |
  identify([I],S), I::ObjIDs,
  identify([SOURCE_PORT],P1),
      SOURCE_PORT::ObjIDs,
  identify([SINK_PORT],P2),
      SINK_PORT::ObjIDs,
  identify([SOURCE_DEV],D1),
      SOURCE_DEV::ObjIDs,
  identify([SINK_DEV],D2),
      SINK_DEV::ObjIDs
```

Figure 14: The logical Flowsheet Editor Model

The same formalism is used to specify the relations of the data warehouse in terms of the conceptual data warehouse model. The difference between these two steps is that on the source level it is a reverse engineering task, on the data warehouse level the relations are designed from the conceptual model.

The mapping of the source model onto the data warehouse model have to be given as intermodel assertions on the conceptual part and as interschema correspondences on the logical part. The intermodel assertions are constructs similar to the intramodel assertions of the conceptual source model [CDL+98]. The interschema correspondences build the basis for the generation of the mediators that actually load the data into the data warehouse. The correspondences in figure 15 demonstrate how the "missing" stream port of the flowsheet model can be specified.

```
convert_1([I,SRC_P],[PID,T]) <-
  FBW_ProcessStream(X),Id(X,I),Source(X,P),
  Id(P,SRC_P),StreamPort(P2),Id(P2,PID),
  T == 'Source'
  THROUGH new_src(I,PID)

convert_2([I,SNK_P],[PID,T]) <-
  FBW_ProcessStream(X),Id(X,I),Sink(X,P),
  Id(P,SNK_P),StreamPort(P2),Id(P2,PID),
  T == 'Sink'
  THROUGH new_sink(I,PID)
```

Figure 15: Interschema Correspondences

The small programs `new_sink` and `new_src` are used to create new identifiers for the relations in the data warehouse. They are supplemented by a code fragment that converts identifiers used in the source databases into data warehouse identifiers and makes use of a global hash table for the mapping. Similar rules specify how the device type coded as tagged value in the flowsheet editor is transferred into an instance of a subclass of `ProcessStep` (figure 11). These interschema correspondences look like this:

```
convert_reactor([I],[PID]) <-
   FBW_ProcessDevice(X),Id(X,I),with_tag(X,T),
   Reactor(R),Id(R,PID), Tag(T,'Type'),
   Value(X,'Reactor')
   THROUGH convert_device([I],[PID])
```

This correspondence is accompanied by an intermodel assertion that states that process devices of the flowsheet editor can be transformed into a reactor.

Mediators that load the data instances into the data warehouse can now be generated using the data reconciliation too [CDL+99]. So the mediators are constructed from small, partly reusable software constructs, combined with declarative specification of the tools and its models.

5. Conclusions

The move from business data warehouses to process data warehouses mirrors the historical development of operational databases in the 70s and 80s. Early databases, evolving around the relational model, focused on business applications. A few years later object-oriented databases came up mostly as a proposal to handle engineering data. Nowadays, semi-structured data models á la XML provide a complementary source of technologies which, also in our context, is becoming important for document analysis and distributed information delivery. We hope to profit from work such as [GSN99] on the generation of XML wrappers for a wide range of media which may appear as interfaces to process engineering tools.

Summarizing the experiences so far, the concept-centred approach to data warehouse design proposed in projects such as Information Manifold and DWQ appears even more important in the process engineering context. This is due to the greater syntactic variations of engineering tools which can only be bridged by semantic modeling approaches with strong formal support. This conceptual modeling approach also enabled us to make operational an extensible body of method knowledge. In the past such growing knowledge either had to remain in the heads of engineers, or led to an growing complexity and unmaintainable engineering tools. Domain standards such as CAPE-OPEN emerge as an indispensible prerequisite for our approach which assumes open tool interaction, with much of the changing body of knowledge captured in the process data warehouse and its metadata repository.

As discussed earlier, the network of tools we have linked to our PDW prototype is still limited in comparison to the richness of chemical engineering reality. Moreover, the number of experimental uses is far too small to see whether case-based method reuse (quite successful in the mechanical service domain) has a real future here. However, it is encouraging to see that one of the largest chemical engineering departments worldwide is indeed setting up a major PDW according to a simplified version of our approach, albeit only for the product perspective. In order to include the process aspect, and to broaden the scope of PDW usability beyond the initial engineering process, a key research challenge will be to understand the interplay between a document-oriented management view of process engineering and plant administration, and the concept-oriented database-like approach pursued in this work.

Acknowledgments. This work was supported in part by the European Commission under BRITE-EURAM project Global CAPE-OPEN, and by DFG under Collaborative Research Center IMPROVE (SFB 476). Thanks are due to the partners in these projects, especially Maurizio Lenzerini, Panos Vassiliadis, Klaus Weidenhaupt, Manfred Nagl, Bertrand Braunschweig, Wolfgang Marquardt, Birgit Bayer and Christoph Quix.

6. References

[BaJa99] M. Baumeister, M. Jarke: Compaction of large class hierarchies in databases for chemical engineering. *Proc. BTW 99 (Freiburg, Germany),* Springer 1999, 343-361.

[BaSM99] B. Bayer, R. Schneider, W. Marquardt: Product Data Modeling for Chemical Process Design. In *Proc. European Concurrent Engineering Conference*, Erlangen, Germany, April 1999.

[BJM+99] B. Braunschweig, M. Jarke, J. Köller, W. Marquardt, L .v. Wedel: CAPE-OPEN - experiences from a standardization effort in chemical industries. *Proc. Intl. Conf. Standardization and Innovation in Information Technology (SIIT99)*, Aachen 1999.

[Bra*99] B. Braunschweig , M. Jarke, A. Becks, J. Köller, C. Tresp: Designing Standards for Open Simulation Environments in the Chemical Industries: A Computer-Supported Use-Case Approach. *Proc. of the 9th Annual Int. Symposium of the Int. Council on Systems Engineering*, Brighton, England, June, 1999.

[CDL+98] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati: Description Logic Framework for Information Integration, *In Proc. 6th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, Trento, Italy, 1998, pp. 2-13.

[CDL+99] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati: A principled approach to data integration and reconciliation in data warehousing. *Proc. Int. Workshop on Design and Management of Data Warehouses (DMDW'99),* Heidelberg, Germany, 1999.

[CGH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, J. Widom: The TSIMMIS project: Integration of heterogenous information sources. In *Proc. IPSJ Conference,* Tokyo, Japan, 1994, pp. 7-18.

[CL93] T. Catarci, M. Lenzerini: Representing and using interschema knowledge in cooperative information systems. *Intelligent and Cooperative Information Systems,* **2**(4), 375–398, 1993.

[CO98] CAPE-OPEN, Conceptual Design Document 2. *http://www.global-cape-open.org/CAPE-OPEN_standard.html*, 1998.

[FS99] E. Franconi, U. Sattler: A data warehouse conceptual data model for multidimensional aggregation. In *Proc. Int. Workshop Design and Management of Data Warehouses (DMDW'99),* Heidelberg, Germany, 1999.

[GCO00] Global CAPE-OPEN Web Site http://www.global-cape-open.org, 2000.

[GSN99] G. Gardarin, F. Sha, T.D. Ngoc: XML-based componetnes for federating multiple heterogeneous data sources. Proc. Conceptual Modeling – ER 99, Paris 1999, 506-519

[GSS99] J. Gulbins, M. Seyfried, H. Strack-Zimmermann: Dokumenten-Management. Springer-Verlag, 2nd edn. 1999

[Hul97] R. Hull: Managing semantic heterogeneity in databases: A theoretical perspective. *Proc. 16th ACM Symp. Principles of Database Systems (PODS)*, Tucson, AZ, pp. 51–61, 1997.

[Inn00] Comos Pt product description. http://www.innotec.de, 2000.

[JaVa97] M. Jarke, M. Vassiliou: Foundations of data warehouse quality: An overview of the DWQ project. *Proc. 2nd International Conference on Information Quality*, Cambridge, MA, 299–313, 1997.

[JGJ+95] M. Jarke, R. Gallersdörfer, M. Jeusfeld, M. Staudt, S. Eherer: ConceptBase – a deductive object base for meta data management. *Intelligent Information Systems*, **4**(2), 167–192, 1995.

[JLVV99] M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis: *Fundamentals of Data Warehouses*. Springer 1999.

[JLW99] M. Jarke, T. List, K. Weidenhaupt: A Process-Integrated Conceptual Design Environment for Chemical Engineering. In *Proc. 18th Int. Conference on Conceptual Modeling (ER 99)*, Paris, France; 1999, 520-537.

[JM96] Jarke, M., Marquardt, W. Design and evaluation of computer-aided process modeling tools. *International Conference on Intelligent Systems in Process Engineering* (Snowmass, Co, July 1995), AIChE Symposium Series, vol. 92, 1996, 97-109.

[LiTS98] S. Ligoudistianos, D. Theodoratos, T. Sellis: Experimental Evaluation of Data Warehouse Configuration Algorithms. In *Proc. 9th DEXA Workshop (DEXA'98),* Vienna, Austria, August 1998.

[Maq95] Marquardt, W. Trends in Computer-Aided Process *Modeling. Computers and Chemical Engineering*, 1995.

[MS00] Microsoft DCOM Web Site http://www.microsoft.com/com/tech/DCOM.asp , 2000

[NaWe99] M. Nagl, B. Westfechtel (Hrsg.): Integration von Entwicklungssystemen in Ingenieuranwendungen. Springer 1999.

[OMG] OMG CORBA Web Page http://www.omg.org/corba

[PB94] Pantelides, C.C; Britt, H.I.: Multipurpose process modelling environments. In: *Proc. Conf. on FOCAPD '94.* CACHE Publications, 1994.

[Ram98] B. Ramesh: Factors influencing requirements traceability practice. *Comm. ACM* 41, (12) 1998, 37-44.

[RJ00] Ramesh, B., Jarke, M.: Towards reference models of requirements traceability. *IEEE Transactions on Software Engineering*, July 2000, vol. 26, no. 7.

[Ros98] T. Rose: Visual Assessment of Engineering Processes in Virtual Enterprises. *Comm. ACM* 41, (12) 1998, 45-52.

[Satt98] U. Sattler: Terminological Knowledge Representation Systems in a Process Engineering Application. *Dissertation*, RWTH Aachen, 1998.

[ThSe97] D. Theodoratos, T. Sellis: Data warehouse configuration. In *Proceedings of the 23rd International Conference on Very Large Databases (VLDB)*, 126–135, Athens, Greece, August 1997.

[Vass98] P. Vassiliadis: Modeling multidimensional databases, cubes and cube operations. In *Pro.s 10th Intl Conf. Scientific and Statistical Database Management (SSDBM)*, 53–62, Capri, Italy, 1998.

[Vin97] S. Vinoski: CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications Magazine*, 35, 2., Feb 1997.