Cheng Hsu, Gilbert Babin, M'hamed Bouziane, Waiman Cheung, Laurie Rattner, and Lester Yee. "Metadatabase Modeling for Enterprise Information Integration." Journal of Systems Integration. 2(1). January 1992. pp. 5-39.

Metadatabase Modeling for Enterprise Information Integration*

Cheng Hsu†[1], Gilbert Babin[2], M'hamed Bouziane[3], Waiman Cheung[2],
Laurie Rattner[4], Lester Yee[2]

Revised July 1991

Rensselaer Polytechnic Institute
Troy, New York, 12180-3590

1 Associate Professor of Decision Sciences and Engineering System, Rensselaer Polytechnic Institute, Troy, New York
2 Department of Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, New York
3 Digital Equipment Corporation, Nashua, New Hampshire
4 Assistant Professor of Management Information Systems, University of New Mexico, Albuquerque, New Mexico

* A preliminary and shorter version of this paper is presented in:

Proceedings 1st International Conference on Systems Integration, IEEE Computer Society Press, 1990, pp 616-624.

† Please send all correspondence to Cheng Hsu, 5219 CII, Rensselaer Polytechnic Institute, Troy, NY 12180-3590. Telephone: (518) 276-6847.

Cheng Hsu, Gilbert Babin, M'hamed Bouziane, Waiman Cheung, Laurie Rattner, and Lester Yee. "Metadatabase Modeling for Enterprise Information Integration." Journal of Systems Integration. 2(1). January 1992. pp. 5-39.

Abstract

An underpinning to the notion of Computer Integrated Enterprises is information integration; that is, the integration of information resources and decision logic across the enterprise so as to achieve fundamental synergies. This concept requires certain basic extensions to two previously separate paradigms: information modeling and metadata management. In particular, both paradigms must consider not only data resources but also contextual knowledge in a unified way; further, they have to converge as a single, integrated method rather than belonging to two distinct stages of a life cycle. Toward this end, a modeling system is developed based on the Two-Stage Entity-Relationship (TSER) approach [3, 4, 5, 7] and the metadatabase method [5, 6, 8].

This paper presents the metadata modeling system, focusing on its basic concepts, design, and current implementation. In addition, the prototype environment of the metadatabase that this system creates is illustrated through some examples taken from a Computer Integrated Manufacturing case.

Cheng Hsu, Gilbert Babin, M'hamed Bouziane, Waiman Cheung, Laurie Rattner, and Lester Yee. "Metadatabase Modeling for Enterprise Information Integration." Journal of Systems Integration. 2(1). January 1992. pp. 5-39.

1. Introduction

The basic objectives of an integrated enterprise information system are generally considered to include the following: data sharing via common databases and communication links, shared executable programs via common system and application interfaces, common user interfaces, and a common enterprise model. In addition, from a decision-making perspective, the system might also be envisioned as the basis for a fusion of decision regimes in the enterprise toward functional synergies throughout.

Achieving integration, however, is a challenging problem for several reasons: The data structures used in subsystems may be complex and often incompatible with each other, and many tasks require a large volume of data processing and communications. Moreover, with the trend toward computerization, there is an increasing requirement to consolidate information resources with databases and their application knowledge, and to effect real-time access to information resources across the enterprise.

Since the early 1980's, this problem has drawn growing national research attention. Examples include the Air Force's Integrated Information Support System project [15], the National Institute of Standards and Technology's Integrated Manufacturing Data Administration System [9], MULTIBASE by Computer Corporation of America [10, 14], and the Computer Integrated Manufacturing Program at Rensselaer Polytechnic Institute [5, 6].

One approach to achieving the desired commonality is to impose homogeneity such as using a single software technology or physical design for all subsystems. However, there are two fundamental issues that make this approach impractical to completely redesign systems in the near future. Firstly, the large investment in existing systems cannot easily be replaced. Secondly, ever-evolving technology and enterprise and subsystem requirements make it necessary to allow for plural system implementations. Thus, research is being pursued to make it possible to achieve the objectives of integration while still preserving the flexibility of subsystem design and reasonable independence in implementation. The ongoing efforts on the topic of distributed and heterogeneous databases [11,13,16] are an important class of research in this general direction. More fundamentally, however, it really represents a basic premise that integration will be achieved at a logical level through information rather than relying entirely on standards or other similar means aiming at attaining physical homogeneity. This notion is referred to as *information integration*, which signifies both the integration of enterprise information resources and the integration of enterprise functional systems through information.

A promising approach to the integration problem based on this concept is to gain control of the organization's information resources (databases, control knowledge, and physical information processing resources) at the metadata level while allowing autonomy of functional subsystems at the actual data level. In other words, the integration is accomplished through globally coordinating the local systems' data models and contextual knowledge which govern how these systems interact with each other to achieve synergy, as opposed to managing directly their transactions in conventional manners such as global serialization and schema integration. Toward this end, an information integration approach using a metadatabase is developed [5,6,8]. The metadatabase, serving as an enterprise kernel for integration, contains a global information model describing organization subsystems including their data resources, application knowledge, and interactions in the contexts of the overall enterprise. In addition, the metadatabase identifies the control strategy and its implementation concerning what and how information is shared and under what circumstances it is used across these subsystems.

Therefore, this approach presents new challenges in concept and technology not only for the established areas of data and knowledge engineering, but also for the evolving efforts on metadata management and information modeling. Essentially, the requirements for a metadatabase entail (1) new methods integrating data and knowledge throughout the information management life cycle (from modeling to representation and processing), (2) new methodology unifying enterprise metadata management with information modeling, and (3) new technology and architecture utilizing the potentials of metadatabase to simplify the complexity of global information control.

These three fundamental requirements stem from the basic technical characteristics of enterprise-wide information integration; on this basis, the metadatabase model is operationalized to facilitate solving integration problems. The first requirement is basically due to the scope of metadata envisioned herein. The adaptive, run-time, and on-line nature of information systems development and management in computerized enterprises implies the second requirements; i.e., enterprise information resources have to be developed and managed on a *continual* basis and in a cohesive manner for such environments. The third requirement implements the benefits of metadatabase for distributed data and knowledge administration, which is a problem not yet satisfactorily resolved by traditional distributed technology.

The current implementation of the metadatabase approach is described in this paper, with a focus on metadatabase modeling. The basis for metadatabase modeling is the Two-Stage Entity-Relationship (TSER) approach which is extensively documented in [3, 4, 6, 7]. The modeling system itself is discussed in detail in this paper to satisfy the first requirement mentioned above. The implementation and operation of the metadatabase are also discussed through a prototype currently under development at Rensselaer, thereby illustrating the other two requirements. A general description of some basic metadatabase processing methods is included; their details, however, are beyond the scope of the paper (see [6,8] for more details).

The concept of metadatabase is further discussed in Section 2, while Section 3 provides an overview of the TSER modeling methodology. Section 4 describes the design and functionalities of the metadatabase modeling environment; it is then illustrated in Section 5 with an example taken from a computerized manufacturing case. This modeling system is incorporated into a metadatabase prototype to demonstrate the whole metadatabase approach, as presented in Section 6. The conclusion and discussion of future work is the subject of Section 7.

2. The Metadatabase

The notion of metadata is as old as the traditional data dictionary systems which are the subject of the recent Information Resources Dictionary System by the National Institute of Standards and Technology (NIST) [2]. This notion is further extended in the emerging technology of repository for systems development using Computer-Aided Software Engineering (CASE) tools, as promoted by IBM and other firms in the industry. Most of the previous metadata systems are limited in that they (1) are tightly coupled with a single database system; (2) perform only the passive, schema and "dictionary" type of functions; and (3) focus narrowly on data models and structures. To bring the problem into prominence, we formulate the metadata system for information integration as a *metadatabase*. This concept asserts and emphasizes the role of the metadatabase as an on-line kernel for *all* systems in the enterprise towards functional synergies and the fact that it is, in its own right, a combined data and knowledge base capable of both passive and active uses in integration.

2.1. Goals and Approach: The Concurrent Architecture

The objective of the metadatabase model is to achieve enterprise information integration over distributed and potentially heterogeneous functional systems while allowing these systems to operate concurrently (i.e., without relying on schemata integration and global serialization). Three progressive levels of functionality are provided; namely, repository (or passive), global query (or semi-active), and systems integration (or active).

From a user's perspective, a passive metadatabase is essentially an organization-wide repository whereby users can obtain information in a timely manner without having to rely on inexact personal communication, knowing where information resides, nor how to access it. The metadatabase provides an enterprise view and local views of the information contained in the individual subsystems, thereby supporting decision-making activities of those who are not users of specific subsystems. This view and decision logic consolidation is a type of integration that facilitates system-wide information sharing and management. In addition, the metadatabase also supports "run-time" systems development with "build-time" CASE capabilities. Since the metadatabase contains existing information models

covering a life cycle of systems development, the metadata can be used for developing new applications or maintaining old ones without starting from scratch; e.g., providing such capabilities as "what-if" analyses for design alternatives and reasonable initial models.

When the metadatabase is employed directly in the processing of global information, a more active role in integration is assumed for it. The contents, i.e., metadata, will be used to provide model-assisted or knowledge-based capabilities for global information retrieval and update. While the metadatabase does describe such activities, it will not interfere with routine data transfers between subsystems nor with self-contained subsystem operation.

A number of methods and techniques have been developed for the metadatabase to effect both its passive and active functionalities.

At the core, the metadatabase employs a new representation method called the GIRD model which is described in Section 2.2 to (1) combine data models at functional, structural and implementation levels with contextual knowledge, (2) represent the combined metadata into a generic metadatabase structure, and (3) support new metadata processing methods. The resultant system, therefore, achieves an important class of metadata independence: It represents the logic of the enterprise as a whole rather than being coupled with individual (local) database systems; it corresponds logically to local schemata but does not supercede them; and it incorporates new models into it through ordinary metadata transactions without necessitating metadatabase recompilation. Based on the GIRD model, new processing methods combine relations, rulebase and routine processing techniques into a unified metadata management environment referred to as the metadata manager. It provides enterprise users with metadata analysis and design capabilities as well as the usual passive repository functions. These capabilities allow the users to perform new applications or changes to existing ones with either the life cycle approach or the rapid prototyping methodology. It is worth noting that among them is the ability to support "what if" simulations with the metadata; e.g., assisting application managers to know just how the current application is related to others, so that the impact of changes is understood before they are undertaken.

In addition to the metadata manager, the metadatabase management system includes a global query manager and a system integrator manager. The global query manager makes use of a set of methods utilizing the metadatabase as on-line assistance for query formulation and processing. In particular, these methods enable a direct approach without relying on an active, governing global schema as the intermediate between user-oriented enterprise models and local heterogeneous implementations. They allow query formulation through direct traversal of functional and structural models whereby users articulate in terms of information models rather than specific syntax of query languages and schemata. The user is not expected to provide technical details of any local systems nor to learn a technical query language (e.g., SQL). Metadata are tapped into to assist semantic ambiguities detection, diagnostic feedback, and even information analysis. In a similar way, the direct approach further uses structural

and implementation models to optimize global queries, generate local-bound code, and assemble results; all with assistance from the metadata. On-line intelligence is also provided through contextual knowledge throughout these stages. Especially, operation rules (e.g., business rules) and data conversion rules in the metadatabase are utilized to resolve semantic ambiguities and data incompatibilities among local systems.

The active role of the metadatabase effects integration of multiple systems through a concurrent architecture implementing the contextual knowledge and global data management requirements in the enterprise information models [6]. In order to enable such capabilities, a new programming approach has been formulated: the Rule-Oriented Programming Environment (ROPE) which calls for a new "rule" section on top of the usual data typing for efficient execution, management and maintenance of distributed knowledge models. Figure 1 is an illustration of the architecture of the manufacturing enterprise applications using the ROPE concept.

[insert Figure 1 about here]

The contextual knowledge are formulated as (1) operating rules for event-based data processing, (2) control rules for inter-system data flow, and (3) decision rules for global, concurrent decision making; thus are represented and implemented as production rules. Similarly, global data management requirements are also derived from the structural models into rules. Together, they are distributed into shells according to the concurrent architecture and managed through the system integrator manager.

Each local functional system is "empowered" with its own shell responsible for monitoring the events that have a global concern, executing the rules assigned to it, and communicating with other shells. The concurrency is, therefore, achieved not only at the usual level of allowing local autonomy (i.e., the shells do not intervene with routine, local operations), but also at a fundamental level of not requiring global serialization of transactions (i.e., each shell executes its own rules without the direct control of the metadatabase). The metadatabase systems integrator, instead, functions basically as a distributed rulebase management system in this regard and globally manages the (changes to) rules in these shells. The creation, maintenance and processing of these shells is conducted by using ROPE.

2.2. Representation and Storage: The GIRD Model

Because the nature and structure of metadata are complex, a representation method is needed to organize the contents, and a modeling methodology is needed for obtaining the metadata. The TSER - based Global Information Resources Dictionary (GIRD) model has been developed as a representation method [8]. An overview of the model is given in Figure 2, which is also an example of the functional

level representation using the TSER modeling methodology (see Section 3.1). The detailed data structures (TSER structural level constructs) of the GIRD can be found in [8].

[Insert Figure 2 about here]

The GIRD model represents the four major classes of metadata mentioned in Section 2.1: enterprise functions (as reflected in application systems), functional models, structural models, and implemented information resources (including users). Each of them may include both data models and contextual knowledge. Note that from an information management's perspective, each class of metadata represents a basic stage in the information systems life cycle and reflects the perspective with which a certain group of users understands the enterprise as a whole. Each class is represented as a rounded rectangle in Figure 1. When implemented, all four classes are structured and stored together in a unified way and can be retrieved both individually and collectively to recover the original models at each distinct modeling stage, traverse models across stages to facilitate enterprise-wide management, or to use parts of existing models to develop models of new or evolving systems.

Enterprise *application systems* are the highest level descriptions of the basic organization subsystems. This level reflects an executive-level view of the subsystems and is primarily focused on how these subsystems interact with one another. The *functional* models (e.g., semantic data models, data flow models, and high-level CASE models) are more detailed in scope and describe the major modules and the decision logic contained within each subsystem. A functional model reflects the enterprise view of the mid-level managers who see their enterprise as a series of logically related processes. *Structural* models (data models and rule models) represent the data-structure and rule-structure translations of the higher-level, functional models. This class of metadata describes the underlying logical schemata of databases and knowledge bases and reflects the enterprise as viewed by information systems people.

Each of these three views supports decision-making for both system development and information management at some level in the enterprise. By reflecting the different orientations that enterprise members use in describing the organization. All of these metadata (models) may either describe individual (local) subsystems or pertain to a consolidated view for the enterprise as a whole. In the metadatabase model, we assume the existence of a global structural model in conjunction with individual functional models as a minimum degree of logical completion.

Finally, the *resources* class of metadata describes the actual (distributed, heterogeneous) implementation models of local subsystems at a global level. That is, existing on top of local subsystems' own metadata (e.g., Oracle schema and access methods in VMS), this class of metadata reflects the hardware and software resources in the enterprise and provides information on directory,

global access (security), and control and communication networks to support the full functionality of the metadatabase system.

The various relationships among these four views are depicted by the five rounded diamonds in Figure 2.

The GIRD representation employs the TSER modeling methodology. TSER provides one set of constructs for functional modeling and another set for data- and rule-structure modeling. (As noted previously, Figure 2 is an example of a TSER functional model.) It also defines and includes algorithms that automatically generate structural models from the functional models [3]. This methodology supports the information modeling task of the metadatabase approach.

3. The Modeling Method: TSER

The Two-Stage Entity-Relationship (TSER) model [3] was first developed to integrate some tasks of system analysis with database design in complex enterprises and was later expanded to include knowledge representation. It entails two levels of modeling constructs devised respectively for semantics-oriented abstractions (i.e., the functional constructs defined below) and cardinality-oriented (normalized) representations (i.e., the structural constructs defined below) of data and production rules. The constructs allow for top-down system development, as well as bottom-up design, i.e., reverse engineering of existing applications or software packages into the TSER constructs. There are rigorous TSER algorithms which map from semantic to structural models and these algorithms ensure that the resulting structures are in at least third normal form [3]. TSER algorithms also integrate views, thus allowing systematic consolidation of any number of data models. The integrity constraints built into the TSER constructs are used to facilitate the management and control of the metadatabase. The basic elements and functionality of TSER are summarized in Section 3.1.

3.1. The Modeling Constructs

A. Functional (Semantic) Constructs: User-oriented semantic-level constructs for object-hierarchy and processes representation; used for system analysis and information requirements modeling; and referred to in TSER as the Functional (or SER) Level Modeling Constructs. Used exclusively to capture semantics.

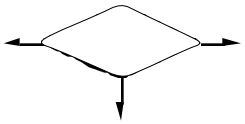
Subject:

Primitives: Contains data items (attributes), functional dependencies (among data items), *intra-*

SUBJECT rules (triggers and dynamic definitions of data items belonging to a single SUBJECT), and class hierarchy (generalizes and aggregates SUBJECTs).

Description: Represents functional units of information such as user views and application systems, and is analogous to frame or object.

Context:



Primitives: Contains *inter*-SUBJECT rules (characterized by references to data items belonging to multiple SUBJECTs), typically includes directions of flows for logic (decision and control) and data (communication, etc.).

Description: Represents interactions among SUBJECTs and control knowledge such as business rules and operating procedures and is analogous to process logic.

Note:

(1) The full contents (as applicable) must be specified for all SUBJECTs at the leaf level of the SUBJECT hierarchy. The class hierarchy implies integrity rules for applications, but its presence is not required.

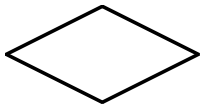
(2) Rules are constructed in the form of (a subset of) predicate logic where all clauses must only consist of the logical operators and the data items that have been declared or defined in the SUBJECTs (except for certain key words such as *do* and *execute*.). A data item may be defined to represent an executable routine, algorithm, or mathematical expression.

B. Structural (Normalized) Constructs: Used as a neutral normalized representation of data semantics and production rules from functional model for logical database design; and referred to in TSER as the structural (or OER) model. There are four basic constructs described below.

Operational Entity (OE):

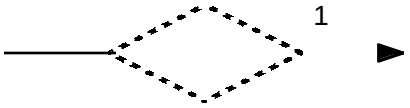
Entities identified by a singular primary key and (optional) alternative keys and non-prime attributes.

Plural Relationship (PR):



Association of entities characterized by a composite primary key and signifying a many-to-many and independent association.

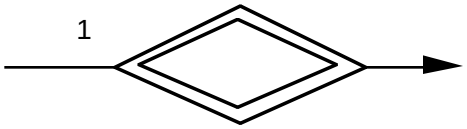
Functional Relationship (FR):



A many-to-one association that signifies characteristic or inheritance relationships. FRs represent the referential integrity constraint implied by the existence of foreign keys.

The arrow side is called the *determined* side and points to either an OE or a PR, while the other side is called the *determinant* and is also linked to either an OE or a PR. The primary key of the *determined* side is included as a non-prime attribute (i.e., a foreign key) of the *determinant* side.

Mandatory Relationship (MR):



A one-to-many *fixed* association of OEs. MRs represent the existence-dependency constraint, and are symbolized as a double diamond with direction.

The "1" side is linked to the *owner* OE while the arrow side points to the *owned* OE.

Note:

(1) In both top-down design and reverse engineering, the structural model is typically *derived* automatically from the functional model by using the TSER normalization and mapping algorithms.

(2) While there usually are multiple functional models representing different views or application systems of an enterprise model, there always exists only one integrated structural model for the global system.

C. The Mapping Algorithms: Used to map the functional models into structural models and to link both types with their respective metadata representations (i.e., the GIRD model). These algorithms which are reported in [3,4,6], generate and decompose views, produce relations or other data structures, and determine integrity constraints for schema design. They also include procedures for creating the metadatabase structure and other logical database and rulebase designs for certain generic models.

In sum, underlying these constructs are two types of primitives: data items and predicate logic. Therefore, the basic structure of TSER metadata is characterized by (1) data representation as relations, (2) knowledge representation in the form of production rules, and (3) the two representations being tied via data items.

3.2. The Modeling Methodology

From the perspective of an information modeling process, TSER may be described as follows. First, we might emphasize that there are actually a variety of modeling perspectives supported by TSER, including data modeling, knowledge modeling, and functional modeling that combines both. Each of these three may be undertaken individually and independently.

The set of constructs for functional modeling is comprised of SUBJECTs and CONTEXTs and is used to represent data semantics and knowledge as viewed from an application level or a systems analysis perspective. These constructs may be employed in their entirety or just subsets of them, depending on the perspective and tasks intended. To illustrate how SUBJECTs and CONTEXTs may be employed separately to perform some major tasks for data or knowledge modeling, they are compared to certain traditional notions in Table 1.

For instance, to perform traditional data modeling tasks, for, say, relational database design, CONTEXTs would not be required (or even the intra-subject knowledge might be omitted). A model using SUBJECTs alone would suffice the requirements of semantic data modeling, and then lead to the normalized structure (OER) through the attendant mapping algorithms. When, on the other hand, both SUBJECT and CONTEXT are used, the resulting model would encompass what is usually referred to as functional models of structured systems analysis. Again, in the case of TSER, the functional level

model would include data semantics and knowledge which would then be structured into normalized combined representations, i.e., the structural level model.

Table 1. The TSER Modeling Portfolio

Functional Constructs	Perspective	Comparable Methods
Subject (used alone)	Data Modeling	Semantic Data Models and Object-Oriented Models
Context (used alone)	Knowledge Modeling	Process and Flow Models, and Rule-Based. Models
Subject and Context	Functional Modeling	Functional Models (e.g., DFD and $IDEF_0$)

The structural modeling constructs are one type of entity and three special types of associations that refine the representation of data and production rules captured in the functional model of logic design. They, too, may be decoupled from the functional constructs and used for modeling in their own right. In this manner, they can be compared to the traditional Entity-Relationship model except for the rigorous definitions in TSER. These definitions ensure proper data structures and integrity rules for the design of databases or rulebases or their combination.

The metadata is represented by connecting the first two sets of constructs and structuring them into TSER meta-models; the result, along with other classes of metadata, is a metadatabase. This process, which corresponds to the common life cycle of systems analysis and design, is depicted in Figure 3, where a typical manufacturing enterprise is used for illustration purposes.

It is worth noting that the three classes of metadata in Figure 3 correspond to the meta-subjects in Figure 2. Therefore, three of the four classes of metadata in the metadatabase model are determined directly from the metadata modeling methodology itself, i.e., the TSER definitions and constructs.

[Insert Figure 3 about here]

In the usual case where the models of more than one system are being developed, a three-step process for basic global information modeling is used. First, a functional model (hierarchy) for each application system is created; second, each leaf-level model is mapped to its corresponding structural model; and third, the several structural models are consolidated into a single global structural model using dependency-theoretical principles (e.g., normalization). When systems integration is actively formulated on top of this basic model, step 2 would be expanded to provide a single functional model. That is, the several functional models would be integrated into a global functional model by creating and

populating *inter*-application CONTEXTs with the control knowledge and operating rules that define the interactions among application systems.

The above logic is depicted in Figure 3, which includes both top-down modeling and reverse engineering approaches. Unlike top-down modeling, which is precise and standard when using the TSER method discussed above, reverse engineering calls for a general guideline whose specificity depends largely on the particular systems to be reverse engineering on. The general guideline employs TSER functional constructs to represent the local models and then proceed from there following the usual (top-down) methodology. For example, base relations, views, or data files would be represented as subjects, with additional data semantics being modelled into functional dependencies, intra-subject rules, or contexts. A specific algorithm is presented in [6].

4. The Software Environment of TSER: a CASE System

Based on the TSER methodology, a CASE-type of integrated data and knowledge modeling environment for the metadatabase system is developed. It is referred to as the Information-Base Modeling System (IBMS). IBMS assists users to design an enterprise information system and create a Global Information Resources Dictionary (GIRD) structure for its metadatabase. Management of the metadatabase is achieved using the Metadatabase Management System (MDBMS), which provides query and system administration capabilities.

Figure 4 depicts an overview of the software architecture. The major elements of MDBMS have been described in section 2.1, while the IBMS is the focus of this section. MDBMS and IBMS are actually integrated in the sense that each can be accessed separately by users and can also invoke each other before, during or after operation. At present, the system is coded in C and LISP and is implemented on an extended relational platform using VAX/Rdb, with C and LISP shells. The implemented metadatabase structure (GIRD) includes Rdb schema, integrity constraints in C, and VAX file management utilities. Since the metadatabase modeling capabilities are primarily provided by the IBMS layer, it is discussed in detail below.

[Insert Figure 4 about here]

4.1. The Integrated Modeling and Management Environment

The first two layers shown in Figure 4, metadata management (MDBMS) and information modeling (IBMS), are coupled architecturally to form a unified physical environment. Thus, these layers reflect more of a logical functional distinction rather than any intended physical separation of implementation. This design, inspired by the metadatabase objective of facilitating enterprise

information integration, allows MDBMS users to invoke the modeling capabilities whenever necessary for active global control. In a similar way, the design also provides IBMS users with access to MDBMS as well as Rdb. Therefore, IBMS is currently designed for three types of users, namely (1) general information modelers, (2) enterprise administrators concerned primarily with the integration of heterogeneous system models, and (3) metadata managers focused on enterprise information resources management. These views of IBMS are depicted in Figure 5, which also illustrates the components and processes that create the metadatabase.

[Insert Figure 5 about here]

Modelers are the basic type of users for which IBMS has been developed. They use IBMS to create models of existing or new application systems at various levels of specification. In this capacity, IBMS allows integrated top-down analysis and design activities as well as reverse engineering and access to individual modeling capabilities. IBMS also performs interfacing/integration with selected software systems.

The other two types of users actually need both modeling and management capabilities for metadata; thus they may access the system through either MDBMS or IBMS. Enterprise administrators use the system to get an enterprise-wide picture of the information models to facilitate their managerial tasks. These users are primarily engaged in global information management; i.e., administering the heterogeneous structures of the enterprise information system as a whole. IBMS supports both access to specific information models and the integration of different information models. Lastly, metadata users are most interested in the types of information resources available in the enterprise, and the management of such resource types. This category of user is most likely to invoke MDBMS through (the assistance of) IBMS, even though they might approach their problems from the perspective of information modeling and engineering.

The following three subsections describe how the IBMS supports the information modeling and management tasks of the three classes of users.

General information modeling

Modelers are responsible for creating the functional and structural models for both data and knowledge; they may interact with any of the modeling components. Information modelers would typically start with some type of structured system analysis. IBMS facilitates such top-down analysis and design by providing a user interface to the functional model module. IBMS has automated algorithms to simplify the task of mapping these models into their respective rule/data structure models; and from these latter models to the target rulebase/database schemata.

Modelers are not restricted to this top-down use of IBMS, however, and may enter data and rule structure models directly, by using the appropriate structural model module. The results of modelers' efforts may also be input into the metadatabase. To ensure approved metadatabase updates, the decision as to whether and when to populate the metadatabase with the modeling results is made by the modelers.

While modelers may select individual IBMS modules to manually document a system at its various levels of detail, they may prefer to take advantage of the integrated algorithms that automatically map from functional models to schemata designed for the target database management system. A specific task that is facilitated by the use of these automated algorithms is the translation of existing models (e.g., functional models: IDEF, Data Flow Diagram; semantic models: Entity-Relationship, Object-Oriented; and structural (data) models: Relational, Hierarchical, Network, etc.) into target schemata.

The modeler is also responsible for creating the initial metadatabase, i.e., the standalone global information resources dictionary (GIRD). This function is done once — to establish the GIRD structure for the enterprise. Then, as the models of enterprise systems are constructed with IBMS tools, the modelers may automatically populate the GIRD structure with the metadata describing the systems. As described in Section 2.2, metadata includes descriptors of the application systems, interface systems, and the database and rulebase schemata. Thus, information describing the functional model hierarchy and the logical database designs are maintained, as are semantic constraints, synonyms and the locations (physical and logical) of the data, rules, models, and systems.

Modelers thus ensure that approved system models are contained in the enterprise repository. This responsibility is consistent with modelers' authority in the enterprise. As shown in Figure 5, this type of user is represented by the middle terminal, connected to the Information Modeling System (Module 1).

Enterprise information administration

In addition to the generic use of information systems modeling, IBMS can be used to facilitate high-level information management tasks for enterprise administrators. This type of user interacts with the parts of the IBMS system that contain the functional models of the various (and, perhaps, heterogeneous) systems and their respective data- and rule-base schemata.

These users are particularly interested in bridging the boundaries among the standalone systems as well as overseeing the structures of these systems per se. IBMS can provide standalone functional models and the schemata for their data- and rule-bases when, for example, enterprise administrators plan to develop new system resources or when they seek a better understanding of the interactions. Perhaps more importantly, IBMS also supports a central task in enterprise information administration: the design and modeling of effective interfaces/ integration across different systems.

The enterprise administration category is depicted in Figure 5 by the uppermost terminal, connected to Module 2: Information Models Integration System.

Metadata management

Similar to the enterprise administrators, the third type of users, metadata managers, requires an enterprise-wide view. This type of user may be distinguished from the enterprise administrators just as data managers may be distinguished from database managers. Metadata managers, then, are responsible for ensuring that information resources are available to all authorized enterprise users. Their primary use of the IBMS involves querying and maintaining metadata.

This type of enterprise user differs from the other types in one basic characteristic: these users are generally outside of the information systems sphere per se and are concerned with other business functions; i.e., they are functional area managers. Effective functional administration often requires information concerning various enterprise systems. Some of these information requirements are well-known and structured, while others are ad hoc. IBMS supports functional managers' need for information by providing them with information (metadata) concerning the available information resources.

This class of users is shown as the bottom terminal in Figure 5, entering IBMS through Module 3, Metadata Management System. Their primary interaction is with the Global Information Resources Dictionary (metadatabase).

4.2. Design features

The major functionalities of the modeling system are reflected in the menu of IBMS (see Figure 6). Features of the IBMS can be summarized as follows.

[Insert Figure 6 about here]

a. An integrated environment. As indicated by the flows in Figure 5, all modules of IBMS are available to any class of users; i.e., Modules 1, 2, and 3 are actually an integrated interface and control subsystem for the modeling environment. IBMS security enables users to access modules consistent with their enterprise authority and responsibility.

b. Multiple interface methods. IBMS supports three methods: menu-driven, graphic (icon), and interactive (language).

c. Automated algorithms for the following tasks:

   (1) Mapping Functional models to structural models: this mapping algorithms are based on functional dependencies specified in functional models.

(2) Mapping structural models to database schemata: IBMS currently supports two relational database systems: VAX/Rdb and Oracle. It is worth noting that IBMS not only generates the data definition language (DDL) code for the schemata but also generates code or commands implementing the integrity constraints embedded in the structural model.

(3) Populating the metadatabase: IBMS generates data manipulation language (DML) code to populate the metadatabase according to the GIRD structure.

d. Self-documenting. At all steps in the modeling process, IBMS generates a variety of reports, documentation and diagrams as requested by users.

e. Error checking when entering the functional model.

f. Access to individual modules. The users are not required to follow a prescribed modeling sequence. In particular, users can work directly with the structural model (OER), perform reverse engineering and model integration at any level, as well as conduct the customary top-down design starting with the functional model (SER).

5. Integrated Manufacturing Case

The above approach is illustrated below with a modeling example. This example is taken from an actual Shop Floor Control System at Rensselaer Polytechnic Institute Computer Integrated Manufacturing (CIM) laboratory. The CIM system consists of six functional subsystems: order entry, computer-aided product design, process planning, production planning (Manufacturing Resources Planning — MRP II), shop floor control and work in progress tracking. To illustrate the semantic hierarchy modeling capabilities, the shop floor control (SFC) system is decomposed into its three main modules: material information, work order management, and workstation control.

5.1. The modeling process

A functional model (see Table 1) was first developed using SUBJECTs and CONTEXTs for this CIM system. Figures 7a-c show the resultant functional modeling hierarchy, where Figure 7a is the highest-level functional model of the CIM system. Each subsystem was modeled as a SUBJECT (rounded rectangle) with subsystem interactions shown as CONTEXTs (rounded diamonds). Note that the SUBJECT “Shop Floor Control System” from Figure 7a is decomposed into a model of the three modules shown in Figure 7b. The sub-systems are also represented as SUBJECTs (Material Information, Work Order, Workstation) and their interactions as CONTEXTs (Temp_Store, Consume, Allocate Material, Queue). In figures 7a and 7b, the arrows emanating from the CONTEXTs indicate the flows of information and control.

[Insert Figures 7a-c about here]

As described in Section 3.1, each of the CONTEXTs in these figures contains production rules and operating knowledge that define the dynamics among SUBJECTs. Each SUBJECT, on the other hand, is characterized by an encapsulation of data semantics describing the statics of information. To illustrate the contents of SUBJECTs and CONTEXTs, Figure 7c shows the data items, functional dependencies and rules contained in two SUBJECTs (Workorder and Workstation) and the inter-subject knowledge contain in the CONTEXT (Queue).

The next step in this case was to derive a structural model from the functional model developed above. Applying TSER normalization algorithms to the data items and functional dependencies shown in the "Work Order" SUBJECT produces the structural submodel in Figure 8. After repeating this derivation for the remainder of the SFC functional model, an integrated structural model resulting from consolidating this "Work Order" submodel with similar submodels mapped from the other two SUBJECTs (Workstation and Material Information) in the SFC System is obtained, as shown in Figure 9. Similarly the completed SFC (sub)model is consolidated with the global CIM facility model (but not shown here).

[Insert Figures 8 and 9 about here]

The consolidation of submodels begins by identifying the identical constructs (i.e., those having the same primary key) among the structural submodels. Each such set of identical constructs is merged into a single construct (note, during the initial mapping phase, identical constructs within submodels have been merged.) The second step in consolidation is representing the foreign keys (referential integrity constraint) arising between constructs in *different* submodels (those arising within submodels have already been specified in the initial mapping) by creating Functional Relationships (FRs). Both submodels and consolidated structural models are normalized and are amenable to input to the automated algorithms for schema generation. The same process would be applied to other subsystems (i.e., order entry, MRP II, design, process planning and WIP tracking) to obtain the enterprise information model. All structural models for these subsystems would then be consolidated to yield a global integrated model.

5.2. Use of IBMS

This section demonstrates the use of IBMS in modeling the above SFC system by integrating the SFC model with models of two other systems in the CIM laboratory and then populating the metadatabase.

The graphic mode is used to develop a functional model for the SFC system. First, a hierarchy is created by defining SUBJECTs and CONTEXTs, and decomposing SUBJECTs into lower level functional models. Figure 10 shows the functional model definition screen to create the model shown originally in Figure 7b. An important step in this process is the global definition of data items. Global definition allows the user to define the format only once for an item used by many SUBJECTs in the system. Figure 11 shows the assignment of data items to a SUBJECT (WK_ORDER). During data item assignment, IBMS helps the user by displaying a list of all defined items and marking the items already assigned to the SUBJECT. The user may also define new items that are then added to the list of defined items.

[Insert Figures 10 & 11 about here]

Once the data items have been assigned to a SUBJECT, the functional dependencies (FDs) are defined for that SUBJECT, as shown in Figure 12. IBMS provides the user with the list of data items already assigned to the subject, so that the user may easily select the ones involved in the FDs. Any number of FDs may be specified for each SUBJECT.

[Insert Figure 12 about here]

Application knowledge is stored in CONTEXTs and SUBJECTs using a production rule format. Any number of statements can be included in the “IF” and “THEN” portions of the production rule. There is no limit to the number of rules contained in either a CONTEXT or a SUBJECT. The resulting SFC functional model is partially shown in Figures 7a-c.

The functional model is then mapped into a structural model using the automated algorithms. During the mapping, IBMS determines the primary key of each SUBJECT, and may also request additional semantics (in order to resolve ambiguities) from the user. The user is allowed to replace a system-determined primary key with an alternative key if one exists (Figure 13a). In Figure 13a, since the list of alternative primary keys is NIL, an error message would result if the user attempted to select an alternative key.

Each SUBJECT is then decomposed according to dependency theory and the functional dependencies given. As shown in Figure 13b, one type of additional semantics defines existence

dependency, which implies a "fixed association," and is termed an MR in the TSER approach (see Section 3.1) The fixed association constraint declares that an instance of the *owned* side exists only as long as its *owner* does. In Figure 13b PART_ID becomes the owner of an MR also involving WO_ID. By answering "Y" (for yes) the user indicates that the system should delete any instances of a work order (WO_ID) that is associated with a deleted part (PART_ID). This is a strong integrity constraint, and reflects the semantics of this particular facility. The resulting SFC structural model (Figures 8 and 9) is mapped into a target database for its own implementation and application. Currently, IBMS supports the mapping (automated schema creation) for two systems: VAX/Rdb and Oracle.

[Insert Figures 13a-b about here]

Finally, the SFC model will be integrated with the models developed separately for the five other systems in the CIM laboratory (see again Figure 7a). The model integration and creation of the global enterprise model requires the functional and structural models of all three systems, and is achieved by selecting the options for functional model integration and structural model consolidation, respectively. Then, metadata from all the system and global models are used to populate the metadatabase.

Turning to the metadatabase itself, the significance and design objectives of the metadatabase model are demonstrated and discussed through a prototype in the next section.

6. The Metadatabase Prototype: Some Examples

A prototype metadatabase system for information integration is currently under development at Rensselaer. This system is implemented for the Computer Integrated Manufacturing Laboratory. The order entry, process planning and shop floor control systems are employed as depicted in Figure 14. The order entry system is implemented as a VAX/VMS file system, the process planning system developed as a dBASE III+ application on an IBM PC/AT, and the shop floor control system runs under PC/Oracle DBMS. The metadatabase is the fourth system and resides on a Digital MicroVAX running Rdb through the host language C. A heterogeneous hardware and software environment was intentionally developed to help convey how the metadatabase approach can help with information integration.

[Insert Figure 14 about here]

The metadatabase operates in three modes providing three levels of functionality:

1) Repository/Enterprise Information Resources Model (Passive Mode)
2) Global Query (Semi-active Mode)
3) Systems Integration (Active Mode)

In the passive mode, the metadatabase functions as an on-line logical kernel whereby users throughout the enterprise can gain access to an integrated model of the enterprise. The second level, semi-active mode, builds on the passive mode and provides model-assisted global query capabilities retrieving actual data from local systems. Finally, the third level, active mode, serves as the knowledge base supporting the control of system interactions and updates to local systems.

To best illustrate how the metadatabase achieves information integration, some examples of its operation are developed below, using a scenario for this environment. Key features and functions of the metadatabase are introduced through these examples.

6.1. Enterprise Information Resources Model (Passive Mode)

The passive mode metadatabase serves primarily for enterprise users, information managers and system developers as a stand-alone enterprise information resources repository. A variety of information ranging from summary views of an enterprise to data-element details is provided by the metadatabase in this mode.

There are four unique features of the passive metadatabase. First, it contains the contextual knowledge about data resources and combines both in a unified representation for easy retrieval. Second, the metadatabase supports life-cycle representation of models ranging from planning and analysis to design and implementation. Third, the passive metadatabase allows users to obtain overviews as well as perform "What if?" types of analysis on proposed new applications or changes to existing enterprise information resources. Finally, the passive mode metadatabase supports the integration of information modeling (using current functional views in the enterprise) with metadata management itself. These features of the passive mode are based solely on the content of the metadatabase. All the information necessary for information integration can be retrieved and then presented in the appropriate format to the user.

To show the capability of the passive mode, consider a scenario in which we are developing a new application to better support management's need to track customer orders. This example illustrates how the passive metadatabase can support both information management and systems development. The following four information requests illustrate the multiple perspectives of the systems planning, analysis, and design life-cycle for developing a new application system: an order tracking system.

(1) Investigate what information resources we have.
(2) Investigate what information we have about orders and parts.
(3) Develop an information model for the order tracking system.

(4) Investigate implementation details for the order tracking system.

What follows are excerpts from an actual run of the passive mode to provide answers to the above requests. For the first request, the metadatabase presents the existing information resources that are available — classified according to the major enterprise subsystems. We have the ability to delve into many levels of detail on any of the systems included in the metadatabase. (i.e. System Level → Functional Model → Structural Model → Data-element Level → Physical Implementation Level)

The second request embeds the knowledge that the data-element "part" would be involved in the order tracking system, and, as such, the query seeks all information relating to part (e.g., "PARTID"). The metadatabase presents this information in Figure 15.

[Insert Figure 15 about here]

For the third request, the metadatabase presents a graphical diagram to help an analyst determine whether an existing model component can be reused in developing the model of the new order tracking system. If the new model cannot be built from components of any existing model, the metadatabase modeling facility is also available to develop it. This new model, when completed and approved, will be integrated with the present global enterprise model. The graphical representation of a part of a model in the metadatabase was shown in Figure 10 and was generated by reverse modeling directly from the contents of the metadatabase (i.e., Figure 10 is *computed* from metadata and not a stored graphical file).

The fourth request presents what model components will be affected if a data item is redefined or deleted. This is the metadatabase sensitivity analysis capability that a systems developer would utilize. Clearly, to involve "PARTID" in the new order tracking system one must consider the implications on existing uses of that data item as shown in Figure 16.

[Insert Figure 16 about here]

From reviewing the results shown in Figure 16, one can see that "PARTID" is involved in all three applications: Shop Floor Control, Process Planning, and Order Entry. It is also involved in a number of rules and physically implemented in a number of files. This data item is logically equivalent to four other data items (with different names and types) from other systems.

These four types of query highlight some of the functions that the passive mode provides to support managers' and developers' needs for integrated information. They also establish a foundation upon which global queries can be based.

6.2. Global Query System (Semi-active Mode)

Enterprise users often want to retrieve information from local subsystems without being burdened with technical modeling or implementation details. Toward that end, the semi-active mode supports end-users' need for local data irrespective of where such data are stored. This semi-active mode provides a global query system (GQS). The global query capability simplifies access to data from multiple (and even heterogeneous) local systems. Users interact with the single integrated global enterprise model instead of manually searching, reconciling, and consolidating information from multiple databases and file systems.

The GQS processes queries that involve any number of local systems without requiring the users to know much about the existence or contents of specific local systems. Three features make the global query system distinct from other distributed query systems. First, the system supports a syntax-free interface for end-users; i.e. the system helps and guides query formulation by applying information contained in the metadatabase. For example, GQS automatically uses the metadata concerning global equivalence among data items to build queries across local systems. Second, end-users are spared many tedious processing details since the global query system generates code to optimize queries, interacts with any number of local systems and consolidates results automatically in the background. Third, GQS has an adaptive interface for metadata retrieval; that is, the menu items provided to users for query formulation are *computed* based on the current contents of the metadatabase and the menus automatically change whenever metadatabase contents change.

Consider this global query example:

> "Find the customer order ID, part ID, part description, and quantity completed for Gilbert Babin's order which has a desired date of 10/25/90."

This request requires that we access data from all three application systems: order entry, shop floor control and process planning. (Note: The user posing this query need not to be aware that the query traverses multiple systems.) The user engages the system through the model-assisted dialog menus to formulate the query and marks the data fields needed along with any conditional statements. (In this query example, "Date desired=10/25/90" and "Customer name=Gilbert Babin.") Once this formulation is completed, GQS decomposes the query into an optimized set of local-bound subqueries that are determined by the system using the metadatabase. Then, each subquery is sent across the network to the respective local systems to be serviced; GQS disseminates these subqueries using the native query languages of the local systems involved (in this case, dBASE III+ for Process Planning, Oracle/SQL for Shop Floor Control, and file system access for the Order Entry System). Each application's local shell (Section 2.1) receives a request to process the subquery. Upon reception, the subquery is executed by calling the local DBMS with the file containing the generated subquery. The results are then sent back by the local shells to the GQS where they are assembled logically and

presented to the user. Figure 17 shows a model-assisted query formulation session. The user does not need to know the information model; it is presented here as an illustration of the model network stored in the metadatabase. This network is used by GQS itself to guide the query formulation process.

[Insert Figure 17 about here]

The shaded lines show the particular path followed by the user to formulate this query. Note the relationship between the data items (listed at the bottom of the model) and the systems in which they are stored (shown at the top). The items tagged with asterisks have been requested in the query. Also observe that the rounded boxes represent SUBJECTs and the squared boxes the corresponding data structures (OEs and PRs) of the enterprise model.

In demonstrating the passive and semi-active modes, the examples have shown how the metadatabase supports a variety of needs for integrated information. The metadatabase functionality also incorporates an active mode that manages and controls the interactions among information system beyond the previous modes. This active mode is described in the next section.

6.3. System Interactions (Active Mode)

Maintaining control of interactions among multiple systems is quite complex since individual systems tend to evolve separately over time and may be distributed physically and logically. Global control is necessary, however, to achieve logically complete information integration for an enterprise. The metadatabase facilitates this complex and difficult task by serving as a knowledge-base for the enterprise and thereby opening up many possibilities for simplification in the design of the environment. In particular, as discussed in Section 2.1, the active mode operation of the metadatabase employs a concurrent architecture incorporating distributed rule-oriented shells for local systems. In this mode, the operating, control, and decision knowledge that has been modeled in the metadatabase become operational. The individual shells (Figure 1) process these rules for their respective application.

The ROPE concept is implemented by using, at the present time, existing software technology. Specifically, the rules embodied in these shells are not hardcoded as would be the case with conventional techniques; instead, they are represented into a separate section of the shell code according to the rulebase model of GIRD. This section is interpreted by the rest of the code for execution, and is also updated directly from outside. This way, the logic of the shells can be changed, maintained, and managed in just about any ways without necessitating the shell code to be recompiled at every change.

The following example demonstrates the significance of these concepts in achieving information integration of the stand-alone Order Entry System with the Shop Floor Control System.

When an order is entered into the Order Entry System and is committed to the order database, a trigger is activated (based upon a rule in the metadatabase). This trigger fires rules that extract the new order, convert it to the destination format, export it to the Shop Floor Control System and then execute the order. The local shell is responsible for activating the trigger; i.e., it must be able to detect that a change occured in the Order Entry System database. The knowledge for inter-system data in the metadatabase is used to determine the data elements and corresponding tables that need to be monitored for changes. This monitoring can be event-triggered or temporal-based. Once a change is detected (by comparing an old copy of the table with the current one), messages are generated and sent to the applications influenced by that change. By detecting the committing of a new order, the Order Entry System active mode shell fires a series of rules that will generate messages to update the Shop Floor Control application. Another shell (Shop Floor Control active shell) receives these messages and process them.

In summary, three levels of functionality are provided by the metadatabase for information integration. In the passive mode, users have a repository of metadata. In the semi-active mode, users have the ability for ad-hoc retrieval of information from anywhere in the enterprise through a guided, enterprise-specific global query system. Finally, in the active mode, the tasks for managing the complex interactions among systems is achieved by activating the knowledge already contained in the metadatabase. Thus, metadatabase technology is used to integrate the management and control of the enterprise information resources.

7. Conclusion

This paper presents a metadatabase modeling approach for information integration in computerized enterprises characterized by heterogeneous and distributed environments. The concept of metadatabase was discussed with a general description of its basic methods (Section 2), and was illustrated through a prototype that has been, and still is, under continual development at Rensselaer since 1988 (Section 6).

It is worth emphasizing that the scope of the metadatabase goes beyond the conventional notion of data dictionary or repository and includes the organization's operating (i.e., contextual) knowledge. The knowledge is combined with data models to achieve all three levels of metadatabase functionality: metadata management, global query, and (concurrent) systems integration (see Section 2.1). The inclusion of contextual knowledge in its own right also facilitates managing knowledge resources, and, increasingly, the imperative to re-use them. Specifically, the various knowledge-based systems in a complex organization should have a global repository to support enterprise-wide knowledge management. The repetitive or predetermined decision logic should be modeled with respect to its data

content to facilitate computerization; similarly, for logical consistency, communications protocols and control procedures should be related to the data resources they affect.

Since the metadatabase includes both data models and contextual knowledge pertaining to the enterprise, its own model requires new capabilities that encompass and combine these two previously separate domains of metadata. Toward that end, a metadata system is developed that includes (1) a data and knowledge modeling environment based on the Two-Stage Entity-Relationship methodology, and (2) a metadata management environment that is integrated with the modeling environment. The Two-Stage Entity-Relationship methodology has been described (Section 3) and illustrated with a case taken from integrated manufacturing (Section 5).

The metadata system includes three main functions to: (1) create functional models, structural models and implementation schemata for both data and knowledge; (2) integrate heterogeneous subsystem models across the enterprise; and (3) manage the metadata repository. The mappings between functional and structural models, as well as between structural models and implementation schemata, are automated. The resulting global information model is implemented into a metadatabase of the enterprise; creation and population of the metadatabase are fully automated according to the information model. The system also supports the creation, storage and management of the metadatabase as a standalone global information resources dictionary for all enterprise users. Further, information models at all levels can be recomputed from the metadatabase at any time for use in new systems development efforts. Whenever necessary, these models can be integrated with new models to update the global enterprise model and the metadatabase.

All of these elements are based on a single architecture and are integrated in the sense of using a unified representation method and modeling logic. As a focus of this paper, Sections 4 and 5 discussed in detail the software implementation of the modeling environment, IBMS, a major subsystem of the metadatabase management system.

The significance of the metadatabase system is demonstrated through a metadatabase prototype (Section 6) that brings together the concepts and results mentioned above. The metadatabase prototype was discussed from the users' perspective to show how this approach effects information integration. Although the demonstration system has been developed for a computerized manufacturing environment, the complexity and nature of the heterogeneity addressed are clearly applicable to computerized enterprises in other domains.

The metadatabase system remains under continual improvement and extension. Planned enhancements to the existing modeling functionality include a model-assisted user interface to facilitate the modeling task and a graphical representation of structural models. Major extensions will focus on (1) the development of a new software technology implementing the ROPE concept and a modeling language for the metadatabase, (2) the incorporation of the object-oriented paradigm into metadata modeling and processing, and (3) reverse engineering (i.e., converting relational and other traditional

schemata, semantic models, and functional or data flow models into TSER models). Also under investigation is the viability of using a control-theoretic method such as Petri-net modeling for TSER model validation.

Acknowledgement

The research is supported in part through a National Science Foundation grant DDM-9015277 and Rensselaer's industry-sponsored Computer-Integrated Manufacturing Program at the Center of Manufacturing Productivity and Technology Transfer. Sponsors of the CIM Program include Alcoa, Digital Equipment Corporation, GE, General Motors, and International Business Machines.

References:

[1] Chen, P., "The Entity-Relationship Model: Toward a Unified View of Data," *ACM Transactions Database Systems*, Vol.1, March 1976, pp.9-36.

[2] Goldfine, A. and P. Konig, *A Technical Overview of the Information Resource Dictionary System (Second Edition)*, NBS Special Publication NBSIR 88-3700, National Bureau of Standards, Gaithersburg, MD, January 1988.

[3] Hsu, C., "Structured Database System Analysis and Design Through Two-Stage Entity-Relationship Approach," *Proc. 4th International Conference on Entity-Relationship Approach*, IEEE Computer Society, Los Alamitos, CA, 1985, pp. 56-63.

[4] _____, A. Perry, M. Bouziane and W. Cheung, "TSER: A Data Modeling System Using the Two-Stage Entity-Relationship Approach," *Proc. of the 6th Entity-Relationship Approach Conference*, New York, NY, 1987, pp.461-478.

[5] _____ and C. Skevington, "Integration of Data and Knowledge in Manufacturing Enterprises: A Conceptual Framework," *Manufacturing Systems*, Vol.6, No.4, 1987, pp.277-85.

[6] _____ and L. Rattner, "Information Modeling for Computerized Manufacturing," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 4, 1990, pp. 758-776.

[7] _____ , M. Bouziane, W. Cheung, J. Nogues, L. Rattner and L. Yee, "A Metadata System for Information Modeling and Integration," *Proceedings of the International Conference Systems Integration*, IEEE Computer Society, Los Alamitos, CA, April 1990, pp. 616-624.

[8] _____ , M. Bouziane, L. Rattner and L. Yee, "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach," *IEEE Transactions on Software Engineering*, Vol. SE-17, No. 6, June 1991.

[9] Krishnamurty, V., S. Y. W. Su, H. Lam, M. Mitchell and E. Barkmeyer, "IMDAS, An Integrated Manufacturing Data Administration System," *Data and Knowledge Engineering*, Vol.3, No.2, 1988, pp.109-131.

[10] Landers, T. and R. L. Rosenberg, "An Overview of MULTIBASE," *Second Symposium on Distributed Database*, North Holland, New York, NY, 1982, pp.311-336.

[11] McLeod, D., "Interbase: An Approach to Controlled Sharing Among Autonomous, Heterogeneous Database Systems," *Data Engineering Bulletin*, Fall 1990, pp. 4-9.

[12] Navathe, S., R. Elmasri and J. Larson, "Integrating User Views in Database Design," *Computer*, IEEE Computer Society, Vol.19, 1986, pp. 50-62.

[13] Sheth, A. and J. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, Vol. 22, No. 3, 1990, pp. 203-236.

[14] Smith, J. M., P. A. Bernstein, U. Dayal, N. Goodman, T. Landers, K. W. T. Lin and E. Wong, "MULTIBASE — Integrating Heterogeneous Distributed Data Systems," *AAFIPS Proceedings*, Vol.50, 1981, pp.487-499.

[15] U.S. Air Force, *Integrated Information Support System (IISS) Report*, Integrated Computer Aided Manufacturing (ICAM), Materials Laboratory, Air Force Systems Command, Wright-Patterson Air Force Base, OH, February 1983.

[16] Wang, R. and S. Madnick, “Facilitating Connectivity in Composite Information Systems,” *ACM Data Base*, Vol. 20, No. 3, Fall 1989, pp. 38-46.