Cheng Hsu and Gilbert Babin. "A Rule-Oriented Concurrent Architecture to Effect Adaptiveness for Integrated Manufacturing Enterprises." in International Conference on Industrial Engineering and Production Management. Mons, Belgium. June 1993. pp. 868-877.

A Rule-Oriented Concurrent Architecture to Effect Adaptiveness for Integrated Manufacturing Enterprises

Cheng Hsu and Gilbert Babin

Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, Troy, NY, USA, 12180-3590

ABSTRACT

The challenge facing today's manufacturing enterprises is not just how to integrate "islands" of information across the enterprise, but also how to keep them in sync with the changing needs and technology. A new approach for solving some of the problems related to growth and change in an integrated environment involving multiple information systems working together is developed. This approach entails (1) a metadatabase serving as the knowledge base for enterprise integration, (2) a concurrent architecture using the metadatabase for integration, and (3) a rule-oriented programming environment (ROPE) for implementing and managing the concurrent architecture. The paper presents the ROPE method, which contributes in its own right to the concurrent processing of distributed rulebase systems.

1. ADAPTIVENESS AND INTEGRATION: A NEW APPROACH

Manufacturing enterprises in today's global market place are hard pressed to deal with diversity, in both products and technologies. They typically need to customize their products (and therefore processes) to respond to customer's rapidly changing needs, and, at the same time, need to integrate multiple systems that are autonomous, distributed, and heterogeneous. These place a great challenge to the underlying information technology.

1.1. The Problem

Interoperability, local autonomy, and concurrent processing are major problems facing the integration of multiple data and knowledge systems. Their solution, however, requires new analyses and insights beyond the previous understanding and formulation of these issues. We submit that the key is adaptiveness.

Presently, integration in most environments is conceived without also considering evolution. Even when change is included in the design, the technology available currently is not sufficient to support evolution in distributed, heterogeneous environments. A case in point is the fact that most connections of systems are achieved by hard-coding the links among them. Even when these links are implemented using rule-based shells, these shells themselves are fixed in the sense that they cannot evolve while maintaining global synergies. This makes future growth and change extremely costly, since redesign, conversion, and at least, recompilation of systems and links are required. In a way, this situation can be compared to the days when the database technology was developed to alleviate the maintenance problem facing file systems. What is needed now, is similar development on architecture and software methods at an enterprise level for multiple databases.

Our objective is to develop new capabilities enabling the enterprise to grow (i.e., incorporating new or legacy systems into the environment) and change its integrated information system as its needs change. This objective is illustrated in a scenario shown in Figure 1, where an integrated environment includes (1) systems that are designed according to some standards (the bottom three), (2) systems that are developed using new technologies beyond the standards, and (3) systems that were in existence before the standards. In addition, any types of systems can be added to or deleted from the environment. Ideally, a;; new, legacy, and changed systems can be included on-line and real-time without interrupting neither their own operation nor any other systems' operation. To bring the problem into prominence, we shall refer to this problem as adaptiveness in integrated environments involving multiple (distributed and heterogeneous) systems. A new approach for solving this problem is developed at Rensselaer. It entails (1) a metadatabase serving as the knowledge base for enterprise integration, (2) a concurrent architecture using the metadatabase for integration, and (3) a rule-oriented programming environment (ROPE) for implementing and managing the concurrent architecture.

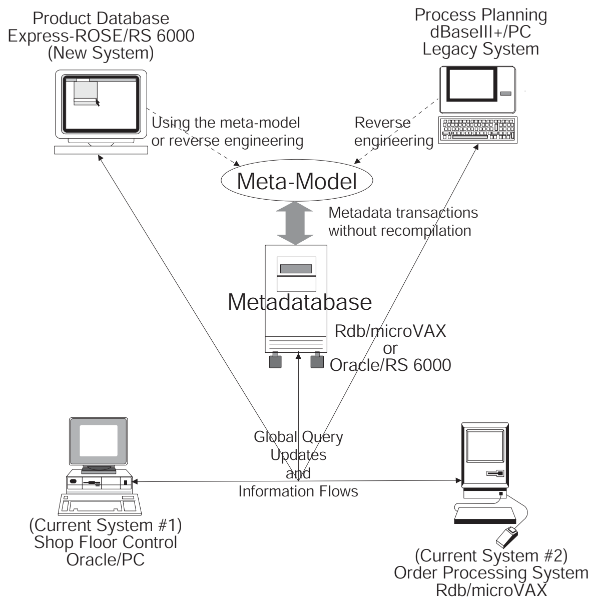

Figure 1 : Adaptiveness and Integration

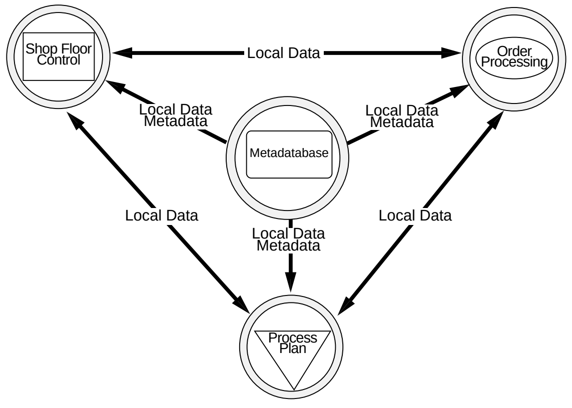

Figure 2 : The Concurrent Architecture Using a Metadatabase

1.2. The Approach

As implied in the scenario, the metadatabase [1] serves as the basis of the new approach to solving the problem. The metadatabase model [1-2] is among the first efforts in the field of integration of enterprise information systems in heterogeneous, autonomous, and distributed environments. Its continual progress has been supported with promising results verified through a prototype and joint investigations with Digital Equipment Corporation and IBM. The model is characterized by the unique solution approach converting the integration problem from one that deals directly with data instances to one that controls through mainly metadata. As such, it provides a global metadata model [3] and a concurrent architecture [4-5] to achieve metadata independence and concurrent processing for multiple systems (in a way similar philosophically to the classical three-schema architecture concept of the database model to achieve data independence).

The concurrent architecture is depicted in Figure 2. The metadatabase itself provides an integrated enterprise model for the multiple information systems, their databases, and the interactions between the different systems; i.e., the information contents and their contextual knowledge. The metadatabase approach (1) uses the enterprise model to assist end users performing global queries free of both technical details and a hierarchy of integrated schemata; (2) distributes the contextual knowledge to empower these local systems to update data and communicate with each other without central database control; and (3) incorporates legacy, new or changed local models into its generic structure of metadata to support evolution without system redesign or recompilation. The shells in the concurrent architecture, therefore, implement the distributed (localized) knowledge which, in turn, is managed by the metadatabase.

This model offers a basic approach to solving the problem of adaptiveness in integration. New methods, however, must still be developed to enable the distribution, execution, and control of contextual knowledge in the concurrent shells in an adaptive manner.

1.3. The Rule-Oriented Programming Environment

The Rule-Oriented Programming Environment (ROPE) method is developed to meet the above need. Its detailed architecture, algorithms, and languages are empirically studied and verified through the prototypical multidatabase environment at Rensselaer.

This method also lays the foundation to further reap the promises of ROPE as a new software engineering paradigm for general interoperability, concurrent processing, and knowledge management in application systems, with or without the metadatabase. An analysis for general principles of adaptiveness is provided, which promises to facilitate achieving system interoperability without having to change the local applications.

Based on the above discussion of the problem, we first analyze the previous results in the light of adaptiveness in Section 2; we then presents the ROPE method in Section 3. Current implementation and empirical results are discussed in Section 4, with on-going efforts summarized in Section 5.

2. AN ANALYSIS OF THE ADAPTIVENESS PROBLEM

2.1. Basic Challenges

The concurrent architecture of the metadatabase has its roots in the classical three-schema concept of (single) databases. The difference represents the shift of emphasis (and needs) away from supporting heterogeneous application programs at the same sites and towards integrating multiple systems across potentially wide-area networks of different sites and environments. The fundamental challenges in this new situation are essentially how to: **manage multiple systems, achieve open systems architecture, retain local autonomy, and allow for system evolution**. They are not sufficiently supported in previous results.

First, consider the traditional results in distributed environments: Distributed Database Management Systems (DDBMS) vs. federated heterogeneous distributed database management systems (or multidatabases [6]). Multidatabase systems differ from DDBMS in two aspects: heterogeneity and autonomy [7]. The different databases in a DDBMS environment may actually be implemented on different platforms, but they must at least share the same DBMS and the same global schema. This is not necessarily the case with multidatabases. However, autonomy is the most distinctive character of multidatabases. While certain degree of local autonomy is always needed in any database systems (e.g., external schemata), multidatabase systems tend to leave more autonomy to the local applications than DDBMS.

From our perspective, an application is fully autonomous when it does not need to comply to any of the following: (1) conforming to integrated schemata (e.g., converting a legacy schema to some global standards), (2) directly cooperating with any global controller (e.g., serialization manager or any direct supervision of transactions and their communications), and (3) requiring users to possess specific knowledge about the global environment in which the local systems operate (e.g., cannot use the local system in the same way with the integration as it would be without). Available results on multidatabases still do not support full local autonomy.

Some of the unresolved technical issues that made local autonomy a crucial need include not only computing problems such as network scaling and distributed query and transaction processing [8] but also the lack of a meta-model and metadata management capabilities. The latter is necessary to support new or legacy systems that do not confirm to global standards.

Previous methods on distributed query and transaction processing fundamentally limit the attainable level of autonomy of local applications. At the heart of the limitation is the criterion used by virtually all methods for achieving consistency among multiple concurrent jobs: serializability (von-Neumann model)[9]. It assures instantaneous correctness for any data in the (distributed) database at the expense of imposing a global controller. Both performance and autonomy are severely hindered this way.

The metadatabase model, on the other hand, employs a new criterion, event/usage correctness, and thereby remove the need for global serialization and foster full concurrent processing among local systems. This, coupled with its metadata independent structure, contributes to the above four basic challenges and paves the way to a solution to the adaptiveness problem.

3. RULE-ORIENTED PROGRAMMING ENVIRONMENT (ROPE)

3.1. ROPE and the Metadatabase

The metadatabase is employed to achieve integration and adaptiveness through its metadata content and the concurrent architecture (Figure 2).

The metadatabase is a repository containing the global model of the different applications of the enterprise and metadata describing each application [3]. The global model includes the interrelationships among the different applications' data models as well as the consolidated enterprise-level data semantics. In addition, the metadatabase also contains the contextual knowledge of data and the logic underlying the different applications and their interactions. Therefore, the metadatabase describes both the behavior (application logic) of each application and the enterprise's integrated behavior.

Using the metadatabase approach to integration, the system modeler initially creates a global model of the application(s) to be implemented into the metadatabase; this process may include either top-down modeling using any appropriate methods or bottom-up reverse engineering, or both. Models integration may be involved. New applications would be incorporated into the global model using the same methodology and the content of the metadatabase would be updated with the new metadata. Similar process applies to deletion and revision as well. All these changes are conducted as ordinary metadata transactions using the metadatabase management system [10]. This process is illustrated in the scenario in Figure 1.

The system integrator manager is triggered by a change in the operational rules or the structural model. Its primary function is to decompose the operational rules and the data modeling rules (extracted from the structural model of the application) and distributes them to the corresponding application local shell. The local application shell receives these rules and updates its local rulebase accordingly; the shells are ready to use the new rules. Because it processes metadata (the distributed and decomposed metadatabase rules) as opposed to data instances, changes in the applications are reflected in the metadata only, making those shells both structurally stable and behaviorally adaptive with the knowledge being directly derived from the metadatabase, the integration is achieved by its proper decomposition and distribution, and by having the shells process it automatically.

The key requirement is a new method that will (1) abstract and consolidate the application systems' global behavior into a knowledge model using the metadatabase approach, (2) analyze and distribute the knowledge to the different applications to effect the required global behavior, and (3) implement and process the distributed knowledge for the applications in an autonomous and concurrent way. We want all application systems to remain unchanged while the distributed knowledge empowers each of them to operate independently and achieve global synergy as well as provides a "masked appearance" of uniformity to enterprise users.

Therefore, the technical nature of the problem is concerned with *concurrent knowledge representation and processing*. This knowledge method must be able to (1) control the application logic of local systems without the use of a central controller, (2) enhance the local applications with a distributed knowledge capability for global behavior without changing them, and (3) transfer the needed knowledge both between the metadatabase and the local applications, and among the local applications themselves.

The concurrent architecture of the metadatabase model (Figure 1) is employed as the basic structure to develop the new method and solve the problem. Instead of having the applications changed to fit a control method (e.g., serialization), we use the knowledge about the model in the metadatabase to build customized control shells around each application. The functionality of each shell depends on the specific knowledge of particular application, therefore, each shell is in essence different. However, we can create these shells in a manner that allows their basic structure to be identical, with the differences only coming from the specific knowledge they process.

The shells must be able to (1) communicate with each other, generating a global behavior, (2) react to changes in the local applications that have global repercussions, and (3) process the knowledge they receive from the metadatabase. In addition, the shells should be efficient to implement, change and operate; why should one have shells if it is easier to modify the application?

ROPE is the method used to develop and manage the shells needed for the concurrent architecture of the metadatabase. It defines: (1) how the shells are built, (2) how the shells behave, and (3) how the shells are managed. Therefore, from the perspective of computer science, ROPE is a (new class of) distributed rulebase system featuring concurrent processing.

The ROPE approach prescribes three principles: (1) rules representing processing logic are separated from the program code by placing them in a distinct "rulebase section" for easy modifiability, (2) communications among shells are conducted through a message system, and (3) the rule processing and the message system are implemented into local environments and combined into the shells. As such, the local shells are invisible to the users, but control the "global" behavior of the local systems.

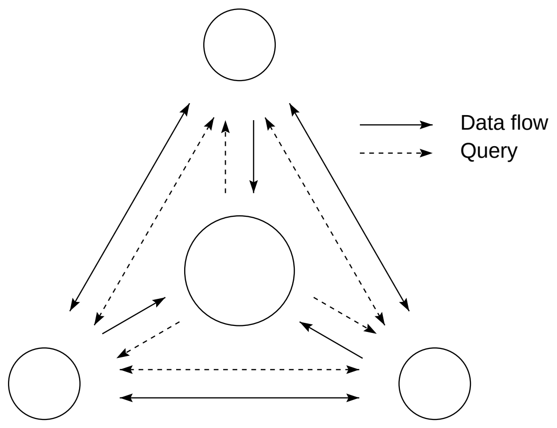

Figure 3 : Global Information Processing

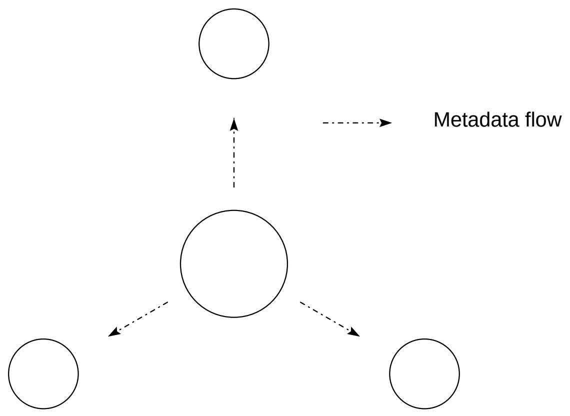

Figure 4 : Rule Updating Process

3.2. Concepts and Objectives

System Integration.

In ROPE, systems integration is achieved in two ways. First, ROPE processes global queries at local nodes. Figure 3 shows how the query and data flow across the different shells. The global query system of the metadatabase management system shell (circle in the center) sends requests for local queries to be executed by the different shells [11]. The resulting data are assembled by the global query system or by one or more remote shells; this decision is made by the global query system to minimize the amount of data flowing across the different applications. In that latter case, the local shells (outer circles) may send subqueries to the other shells, along with data. The different local shells can also send a request for a global query to be processed to the metadatabase management system shell. Therefore, the shells can be extended to include additional intelligence judging whether or not a local user's query/request is global. The metadatabase contains all necessary knowledge and ROPE provides the mechanism; the actual implementation, however, may require embedded interface, which is more a design problem than ROPE principles.

The second class of integration through ROPE is global updates and information flows; which involves direct use of the "rulebase sections". In this case, the local shells monitor the events happening in the application and database(s) they're attached to. Rules are triggered based on the status of the application or the database(s). These rules may generate automatic update commands to be performed by the other shells and query other databases to update their local databases.

Adaptability and Flexibility.

ROPE assures adaptability (1) by storing its knowledge in the form of rules, (2) by directly processing these rules in local languages, and (3) by automatically updating these rules whenever the metadata is modified and new rules propagated by the metadatabase managers (see Figure 2). Any change in the content of the metadatabase will trigger the generation of updated rules in a neutral form, which is then taken over by ROPE and implemented into the local shells in local forms (see Figure 4). Because most of its operations are defined in the form of rules, the shell is easily adapted to new situations, and provide a flexible interface between the different applications and the metadatabase management system.

4. THE BASIC ELEMENTS AND CURRENT IMPLEMENTATION OF ROPE

In order to avoid changing the different applications of the enterprise while integrating them, we believe that the pertinent application logic should be abstracted into a separate layer where the logic is represented directly as production rules and implemented as shells. Furthermore, this structure allows for the application logic to be managed globally (through the predicate logic level representation, which is primitive without artifacts such as frames and objects) and evolved outside of the applications. Thus, it constitutes a flexible environment that can respond to changes more rapidly than other approaches. The shells should be simple in structure, and be as implementation-independent as possible to enable software portability. In particular, the design entails: (1) global distribution of the rules into the local shells to assure global consistency, (2) use of message protocols for inter-shell communications, (3) adoption of local language for intra-shell communications, and (4) utilization of the metadatabase system for simplifying and optimizing the computing complexity. Seven basic elements of shell are determined as follows (see Figure 5).

4.1. The Static Structure of ROPE

Rulebase Segment

A key element of the shell is the possession of rules. The rules should be constructed in a specific section of the source file, in a separate file, or in any program readable location within the software environment. The format of the rules should facilitate the access to and execution of the rules. Logically, the Rule Segment implements the intersection of the global rulebase model and the local application logic; physically, it employs a data structure amenable to local software environments. All of the rules are originating from and globally managed by the metadatabase. Whenever a contextual rule is changed, the change is propagated to all local systems that will be affected by it. Also, when changes occur in the global data model of the enterprise, new data management rules are generated to replace the existing rules in the Rule Segment of different shells.

Network Monitor

The Network Monitor is the shell's interface with the communications network, thus it provides the window to the outside world for the local application and its shell. It receives incoming messages and passes them to the Message/Rule Processor. It is also responsible for sending messages to other nodes. This module is tailored to the particular network employed for the different applications in implementation, and requires access to the routing information needed to reach the other applications.

Local Application Monitor

This module interfaces with the local application. It "spies" on the behavior of the application and reports any globally significant event to the Message/Rule Processor, such as any change to the local database that would result in the execution of a global behavior rule. The Local Application Monitor must also be tailored to the local application (database); however, its functionality can be specified in general terms, allowing implementations on very different platforms.

Message/Rule Processor

This module is the inference engine of the shell. Based on the events it receives from the Message Monitor, the Local Application Monitor, and the enterprise data, it will trigger the appropriate rule(s) stored in the Rule Segment. There are two categories of rules: rules derived from the enterprise model and rules defining the internal behavior of the shell; both are structured into the Rule Segment. This module is generic and neutral to particular implementations.

Timer

The Timer manages a list of time events. When a time event is due for processing, the Timer notifies the Message/Rule Processor that a specific rule must be triggered. This element further mitigates the impact of specificities in local software and hardware environments for the rest of the shell.

Result Integrator

The Result Integrator is used by the shell to assemble the result of local queries and to perform other transformations to the result of these local queries. When a rule is launched, the data items it needs must be fetched from the different systems involved. As the result from these query return to the shell, the Result Integrator is call to join them.

Database Interface

The Database Interface assures the neutrality of the data values been processed by the shell. This is accomplished by converting the data item values into their global equivalent value, when the data is fetched, and by converting the global equivalent value into the local value, when the data is written in the local database. The implementation requirements of this module are similar to the Message/Rule Processor.

4.2. The Dynamic Structure of ROPE

Based on the static structure, we develop specific algorithms to create the shells as well as perform the tasks each module of the shell calls for. Just like the structure itself, these algorithms include both a generic nucleus and a system-specific interface to enact appropriately for different application systems. These algorithms suggest how the knowledge needs to be structured and what knowledge is needed for the shells to perform their tasks, hence defining the language requirements. The algorithms and languages constitute the dynamic structure of ROPE.

There are three major areas where new languages are needed to bring about the application systems' global behavior through ROPE. Initially, we must be able to describe and model that global behavior. Then, the knowledge must be distributed to the local shells in a form suitable for the Message/Rule Processor. Finally, we need to define a communications protocol for connecting the shells. These three areas combined with the needs of the five elements of the shell, define three basic classes of language as described below.

Shell Definitional Language

Shells are created by using this language, thus it is completely callable by the metadatabase management system or a user of ROPE. The language first defines a shell in generic terms in a global environment, then uses a code constructor to implement the generic structure in the target local environment. Shell definitional constructs include (1) the system functions defining the five elements of the static structure and their attendant algorithms, (2) the system specifications defining the interfaces with the local application, and (3) system parameters defining the requirements of the interfaces and the algorithms. These constructs constitute a generic creation of shells which is system-independent. This result is then mapped to specific software environments through the code constructor, to allow for maximum portability. The definitive syntax of the language depends on the final design of the static structure of ROPE.

Modeling and Rule Language

The Modeling and Rule Language helps in describing the knowledge (concerning the application systems) contained in both the metadatabase and the shells. It is an extension of the rule language proposed in [10].

There are two types of actions in the rules: system actions and user actions. The system actions are functions and procedures that are included in every application's shell, as integrating tools. They include the routines to process and generate the messages that the shell sends to or received from other application's shells. The user actions are all the other functions and procedures involved in the rules.

The Modeling and Rule Language must provide constructs to define: (1) the firing conditions of the rule, (2) the actions to be performed when the rule is fired, (3) the globally significant events (e.g., time event, database event), and (4) how to access the data needed to execute the rule. This language is an extension of the rule language used in [10].

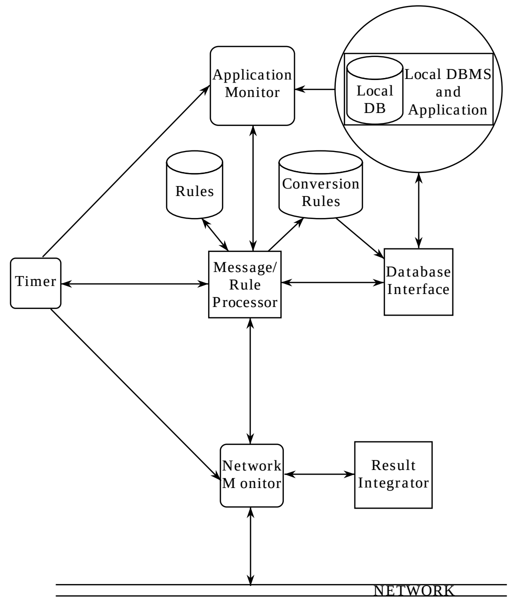

Figure 5 : ROPE Shell

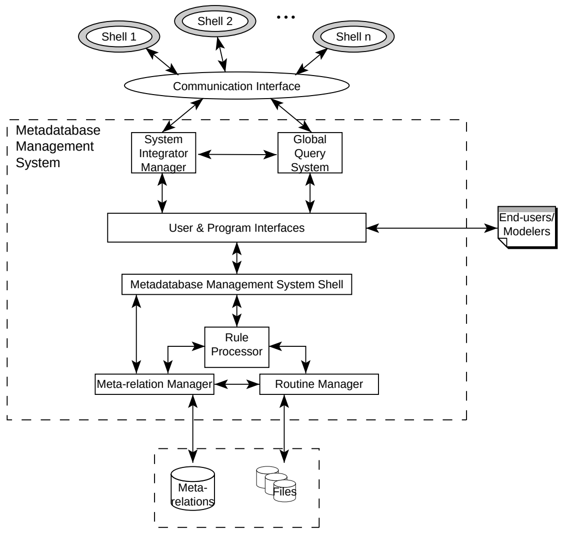

Figure 6 : The Metadatabase Management System

Message Protocol and Language

The messages are used to enable communications across the different applications. They are also used to link the metadatabase management system shell with each application's shell.

There is some minimal information that the Message Language should express in order to completely specify a message: the message identifier, the function to be performed upon reception of the message and the necessary parameters, and the origin and destination of the message. The functions can be further classified into metadata and data functions. The metadata functions are used to manage the Rule Segment of the shells. The data functions are used to enable cooperation among shells and with the metadatabase.

We have obtained promising results over the past few years, which cover most aspects of the problem analysis and conceptual solution. The empirical verification is also underway through the prototype CIM environment using the metadatabase model. The preliminary results have demonstrated the feasibility of (1) the concurrent architecture of the metadatabase, (2) the concepts of ROPE (the shell architecture, the algorithms, and languages), and (3) the adaptability of the shells, thus the behavior of the local applications.

The conceptual design of ROPE is largely completed. We separated the static and dynamic elements of ROPE, defined the different modules used in a shell and their basic functionality, and established the need for the different languages used by ROPE. The five static elements of the shell have also been defined, including a description of their functions within the shell and specific algorithms for these elements.

The Message Protocol and Language definition includes a partial list of functions the messages can perform [12].

4.3. ROPE Environment Implementation

Figure 6 shows the software architecture of the basic metadatabase environment. The Global Query System implements the model-assisted global query method. The combined data and knowledge processing method gives rise to the Metadata Manager consisting of the Rulebase Processor, the Meta-relation Manager and the Routine Manager. The Systems Integration Manager derives information flow (e.g., events and data management) rules from the information models contained in the metadatabase, manages (through ROPE) these rules when changes are needed, and distributes them to local shells using ROPE (described above). The Program and User interface represents constructs for both internal and external users.

Much of the core metadatabase research and implementation has been completed at Rensselaer as part of the industry-sponsored Computer Integrated Manufacturing (CIM) Program (through June, 1992) and the Adaptive Integrated Manufacturing Enterprises (AIME) Program (since June, 1992). Initiated in 1983, the CIM program has been sponsored by a consortium of major U.S. based manufacturers; current sponsorship is Alcoa, Digital, GE, General Motors, and IBM. Representing the vision beyond CIM, the AIME Program is jointly funded by the same consortium and the federal government.

A Metadatabase Management System (MDBMS) has been implemented on a microVAX platform using Rdb as the database engine for the metadatabase and has been demonstrated publicly. A new version using RS 6000 AIX and ORACLE has recently been developed to provide a multi-platform and distributed metadatabase environment. To facilitate user and program interaction with the metadatabase, a shell has been developed in C. Currently, users interact via a menu (in the VAX version) or an X-Window (in the AIX version) interface, while other systems interact through a query language developed for that purpose [11].

At this time, the ROPE environment is still under development. Although some modules still need to be implemented, most concepts put forward by the ROPE approach have been empirically tested using the metadatabase environment. This environment, in addition to the metadatabase, includes (1) a shop floor control system (on IBM PC, using C and ORACLE), (2) a process planning system (on IBM PC, using dBase III+), (3) an order processing system (implemented on VAX with C and Rdb, and on RS 6000 AIX with C and ORACLE), and (4) a product database (on RS 6000 AIX with Express schema, ROSE database and C++).

5. ON-GOING EFFORTS: A CONCLUSION

This research encompasses four fundamental concepts in the field of multiple information systems: (1) interoperability, (2) local autonomy, (3) systems evolution, and (4) open system architecture. These concepts are closely related to adaptiveness. Interoperability is the property of systems that performs (database) operations across different software (database management systems) and hardware platforms, which may use incompatible models (e.g., relational database vs. object-oriented database). A system has local autonomy if its structure (database schemata) does not have to be converted to conform to a global controlling standard (integrated schemata), nor must it confer with the other applications, or a global controller, when performing tasks within its own boundaries. Systems evolution refers to the ability of a system to change its design requirements such as business/operating rules, information contents, and configurations. Finally, open system architecture represents the idea of providing a neutral environment to accommodate different manufacturers of information technology as well as supporting multiple systems. The previous literature tends to focus these concepts on new systems development, which does not reflect the full scope and real nature of the problem. This research, in contrast, stresses the need to generalize it into covering both old and new systems and considering both (one shot) development and (continual) evolution; that is, integration with adaptiveness.

Therefore, the new ROPE method contributes not only to the metadatabase solutions to the problem, but also to these general concepts in the field. Immediately, the results extend the current metadatabase technology and result in a particular method effecting adaptiveness in integrating multiple system environments. In addition, the results will facilitate the general interoperability problem in application areas beyond multidatabases. Unresolved issues in software engineering such as the management of processing logic (contextual knowledge) which is currently embedded in the procedures of application programs will also be able to utilize the new principles. For instance, the knowledge can be abstracted into a "rulebase" section in a manner similar to the data typing section of object-oriented programming and be managed using ROPE techniques. These contributions will lead to a new software engineering paradigm extending data typing into data and knowledge typing.

6. REFERENCES

1 Hsu, C., "The Metadatabase Project at Rensselaer," ACM Sigmod Record, 20(4), pp. 83-90, 1991.

2 Hsu, C. and C. Skevington, "Integration of Data and Knowledge in Manufacturing Enterprises: A Conceptual Framework," Journal of Manufacturing Systems, 6(4), pp. 277-285, 1987.

3 Hsu, C., M. Bouziane, L. Rattner, and L. Yee, "Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach," IEEE Transactions on Software Engineering, 17(6), pp. 604-625, June 1991.

4 Hsu, C. and L. Rattner, "Information Modeling for Computerized Manufacturing," IEEE Transactions on Systems, Man, and Cybernetics, 20(4), pp. 758-776, 1990.

5 Hsu, C., G. Babin, M. Bouziane, W. Cheung, L. Rattner, and L. Yee, "Metadatabase Modeling for Enterprise Information Integration," Journal of Systems Integration, 2(1), pp. 5-39, January 1992.

6 Soparkar, N., H.F. Korth, and A.S. Silberschatz, "Failure-Resilient Transaction Management in Multidatabases," Computer, 24(12), pp. 28-36, December 1991.

7 Georgakopoulos, D., M. Rusinkiewicz, and A. Sheth, "On Serializability of Multidatabase Transactions Through Forced Local Conflicts," in Seventh International Conference on Data Engineering, IEEE Computer Society Press, Los Alamitos, CA, pp. 314-323, April 1991.

8 Özsu, M.T. and P. Valduriez, "Distributed Database Systems: Where Are We Now?," Computer, 24(8), pp. 68-78, August 1991.

9 Eswaran, K.P., J.N. Gray, R.A. Lorie, and I.L. Traiger, "The Notions of Consistency and Predicate Locks in a Database System," Communications of the ACM, 19(11), pp. 624-633, November 1976.

10 Bouziane, M., Metadata Modeling and Management, Unpublished Ph.D. Thesis - Computer Sciences, Rensselaer Polytechnic Institute, June 1991.

11 Cheung, W., The Model-Assisted Global Query System, Unpublished Ph.D. Thesis - Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, November 1991.

12 Babin, G., Adaptiveness in Information Systems Integration, Unpublished Ph.D. Thesis Proposal - Decision Sciences and Engineering Systems, Rensselaer Polytechnic Institute, February 1992.