Invited paper to be presented in CIKM (Conference
on Information and Knowledge Management),
Baltimore, MD, November 28 - December 2, 1995.

An Extensible Knowledge Base Management System for Supporting Rule-based Interoperability among Heterogeneous Systems *

Stanley Y. W. Su, Herman Lam,
Javier Arroyo-Figueroa, Tsae-Feng Yu and Zhidong Yang

Database Systems Research and Development Center
University of Florida

An Extensible Knowledge Base Management System for Supporting Rule-based Interoperability among Heterogeneous Systems *

Stanley Y. W. Su, Herman Lam,
Javier Arroyo-Figueroa, Tsae-Feng Yu and Zhidong Yang

Database Systems Research and Development Center
University of Florida
Gainesville, Florida 32611-6125
Email: su@cis.ufl.edu

Abstract

The main objective of a virtual enterprise (VE) is to allow a number of organizations to rapidly develop a working environment to manage a collection of resources contributed by the organizations toward the attainment of some common goals. One of the key requirements of a virtual enterprise is to develop an information infrastructure to support the interoperability of distributed and heterogeneous systems for controlling and conducting the business of the virtual enterprise. In order to achieve the objective and to meet this requirement, it is necessary to model all things of interest to a virtual enterprise such as data, human and hardware resources, organizational structures, business constraints, production processes, and activities in work management. Additionally, a system is needed to manage the meta-information and the shared data and to provide both build-time and run-time services to the heterogeneous systems to achieve their interoperability. In this paper, we describe the modeling requirements for a virtual enterprise and show how a global, mediated VE conceptual model can be constructed at build-time and be used by a knowledge base management system (KBMS) to provide run-time support for the operation of a virtual enterprise. The KBMS differs from the traditional database management system (DBMS) in that it provides not only the traditional database management services (such as persistent, object management, transaction management, etc.), but also a set of knowledge base management services. Most notably, the KBMS provides a request/event monitoring service which monitors the invocation of the methods which automatically triggers the processing of rules by a rule processing service when certain methods are invoked. We shall also describe how we apply the KBMS technology in the R&D efforts of a project called the National Industrial Information Infrastructure Protocols (NIIIP) to provide a rule-based interoperability among heterogeneous systems.

* Acknowledgement: This work is supported by the Advanced Research project Agency under ARPA Order #B761-00. It is a part of the R&D effort of the NIIIP Consortium. The ideas and techniques presented here are those of the authors and do not necessarily represent the opinion of other NIIIP Consortium members.

1 Introduction

In this age of global economy, many large and complex projects require the cooperation and collaboration of multiple organizations throughout a nation or across multiple nations. For the purpose of carrying out the projects, virtual enterprises (VEs) need to be formed by these organizations to pull together the best of their resources. In order for a virtual enterprise to success, it is essential that the organizations participating in the VE are able to rapidly and flexibly develop a common working environment to manage and use their resources toward the attainment of their business goals. Data and computer resources are key resources which need to be shared. However, they are generally managed by dissimilar software systems running on heterogeneous computing platforms. Thus, one of the key requirements of a virtual enterprise is to develop an information infrastructure to support the interoperability of distributed and heterogeneous systems for the purpose of accessing the diverse data and program resources.

In recent years, the data and knowledge engineering communities have been very active in researching and building systems for accessing data stored in distributed and heterogeneous database systems. These systems are called heterogeneous database management systems, multi-database systems, or federated database systems [1, 3, 6, 13, 15, 19, 22, 29] depending on the tightness of schema integration and the degree of local autonomy that is allowed. The emphasis of these existing systems is to achieve the sharing of data managed by heterogeneous database management systems (DBMSs). Other efforts have been made to tackle various technical problems associated with the reuse of programs in legacy systems and the sharing of data managed by information systems not just DBMSs (e.g., file management systems, CAD systems, in-house developed application systems, etc.). Some good examples of these efforts can be found in [5, 7, 11, 12, 30, 31]. Our work is closely related to these works. However, in our work, we focus on the development of a knowledge base management technology to solve problems related to the interoperability of diverse software systems by using a semantics-rich, extensible object model,

its modeling language, and a supporting KBMS to specify and control their interoperations.

Under the support of the Advanced Research Project Agency (ARPA), a group of industrial companies, universities, and government organizations have formed a consortium (a virtual enterprise) to undertake a project called the National Industrial Information Infrastructure Protocols (NIIIP). The goal of the project is to establish an open, standards-based software infrastructure protocol for integrating heterogeneous and distributed processes, data and computing environment across the US manufacturing base. The main objectives of the project are to 1) commercialize NIIIP Consortium members' products, 2) incorporate legacy manufacturing information systems to preserve existing investments, 3) provide functionality and usability to enable virtual enterprises, and 4) encourage widespread adoption of the NIIIP technology. The consortium is building its technology by integrating a number of existing technologies; namely, the communication technology, the information technology, the object technology, and the workflow and knowledge management technology [20]. The knowlege base management technology presented in this paper is a part of the University of Florida's contributions to the NIIIP Consortium's R&D effort. It provides an information modeling facility for modeling the resources of interest to a virtual enterprise such as data, human and hardware resources, organizational structures, business constraints, production processes, and work management activities. Additionally, it provides a knowlege base management engine for managing the meta-information and the shared data needed by other NIIIP VE components and for providing both build-time and run-time services to the heterogeneous systems to achieve their interoperability.

The organization of the remainder of the paper is as follows. In the next section, an abstract model of the system architecture of NIIIP and the modeling requirements for a virtual enterprise are discussed. We describe how a mediated global VE conceptual model can be constructed at build-time and be used by a knowledge base management system (KBMS) to provide run-time supports for the operation of a virtual enterprise. In Section 3, the KBMS which has been under development at the University of Florida in the past several years is described. In Section 4, we describe how this KBMS technology is applied in the NIIIP project to provide rule-based interoperability. Finally, a summary and some concluding remarks are given in Section 5.

2 NIIIP System Architecture and Modeling Requirements

In order to develop an information infrastructure to support the interoperability of distributed and heterogeneous systems and to control and conduct the business of a virtual enterprise, it is necessary to model all aspects of the virtual enterprise such as data, human and hardware resources, organizational structures, business constraints, production process, and work management activities. The resulting conceptual model is an abstraction and a computational representation of the real-world objects, each of which can be a physical entity, abstract thing, concept, relationship, function, process, or anything of interest to a virtual enterprise. It captures the structural and behavioral properties of these objects as well as the constraints associated with them.

In this section, we first describe an abstract model of the system architecture introduced by NIIIP to support the operation of a virtual enterprise. In this context, we then describe the modeling requirements for a virtual enterprise and show how a mediated global VE conceptual model can be constructed at build-time and be used by human and software clients for accessing data and software resources at run-time.

2.1 NIIIP System Architecture

An abstract model of the NIIIP system architecture is shown in Figure 1. It is based on the Common Object Request Broker Architecture (CORBA) [21] introduced by the Object Management Group (OMG). The main goal of NIIIP is to provide an infrastructure to support the formation and operation of an industrial virtual enterprise (VE). An industrial virtual enterprise is formed by a group of organizations to design, manufacture, and distribute products. Member organizations possess data resources and program services which are to be shared by other members of the virtual enterprise. In the NIIIP environment, these resources and services are provided by a set of servers, which are physically distributed, but are interconnected through an Object Request Broker (ORB), as shown in Figure 1. The interface to the services provided by each server is defined in a NIIIP Common Language (NCL) [28], resulting in a number of "local schemas", as described in the next subsection. The modeling constructs of NCL allows each local schema to describe its services in terms of their data properties, associations, keyword constraints, rules, and methods. The integration and mediation of the local schemas, along with the definitions of other global virtual enterprise resources, forms the mediated global schema (This process will be described in Section 2.2.). Both human and program clients can access object services provided by the servers through the interfaces specified in the global schema.

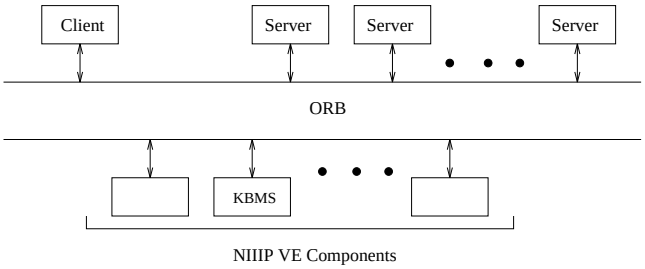

Figure 1: An Abstract Model of NIIIP System Architecture

The accesses are controlled by a set of NIIIP VE Components shown at the bottom of Figure 1 which provide session, workflow, data management, agent, negotiation, knowledge management, mediation, and other services. These VE Components are servers which are implemented by the NIIIP consortium members to control the orderly and proper (without violating security and access constraints) accesses to object services. Just like the services provided by legacy systems, they are distributed and heterogeneous servers which communicate with one another through the ORB. Thus, in an abstract form, all the software systems (legacy systems/applications and NIIIP VE servers) are connected to and communicate through the ORB. They may act as clients as well as servers depending on whether they are the requesters or the providers of services. Those functionalities of the systems and their data which are useful to the virtual enterprise are exposed to other systems as objects having well-defined interfaces. For additional information on the NIIIP project and the NIIIP system architecture, please refer to [20].

2.2 VE Modeling Requirements

In order for a virtual enterprise to represent, manage, and control the use of the shared, distributed, and heterogeneous resources, it is necessary to define the computational models of these resources and their inter-relationships by using an information modeling facility. Furthermore, the access and use of these models by either human users or computer programs should be supported by an operation invocation and/or a querying facility and its underlying information processing engine. We define some modeling requirements below and describe a processing engine in Section 3.

Object-oriented Representations of Existing Data and Application Systems' Resources. Member organizations possess all sorts of data and application systems' resources which are generated and used by different types of information users and/or heterogeneous information systems (e.g., relational application systems, CAD systems, SDAI/STEP applications, etc.). In a virtual enterprise environment, these data and application systems' resources can be uniformly modeled as objects in an object-oriented framework. Their data properties, associations (i.e., semantic relationships between or among object types and their instances), constraints and operations (or methods) are defined by their object types resulting in a number of "object-oriented local conceptual schemas". Each of these schemas captures the semantic contents of the data and application systems' resources of a member organization that are deemed useful for the business operations and activities of a virtual enterprise. This step of the modeling process is shown at the bottom of Figure 2.

Modeling of Virtual Enterprise Components. As described in Section 2.1, a number of VE Components are being developed by the NIIIP consortium to control the access and use of data and software resources of heterogeneous systems. They can be modeled in an object-oriented paradigm as object classes and their inter-relationships by various association types. These components' interfaces need to be explicitly defined so that their functionalities (or object services) are made known to NIIIP's human and software clients as well as to the components themselves. Activation of these services are done by either local message passing or by remote method calls through an ORB or other multi-transport data services. The models of these components are again captured in a number of object-oriented local conceptual schemas.

Modeling of Other VE Resources and Global Information. In addition to the modeling of existing data and application systems' resources and VE Components, other resources, such as tools, people, hardware devices, organization units and structures, and physical capital and monetary resources need to be modeled in the object-oriented framework. Some of these resources can be modeled by intelligent agents which represent the "interests" of these resources in the NIIIP environment. Furthermore, VE information which is "global" in nature and to be shared by some VE Components will have to be modeled to inter-relate the modeled objects. Examples of such global information include:

- new associations that link object types of different local schemas
- new constraints to be imposed on them
- work management models that control the VE project or production activities
- negotiation procedures and rules that deal with the requests and deliverables of services
- mediation rules and operations needed to resolve the naming, structural and semantic conflicts and discrepancies among local and global resources

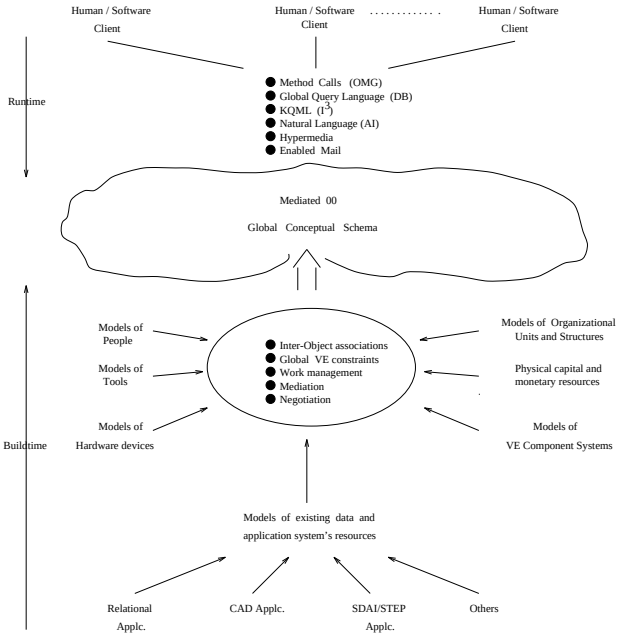

Figure 2: Modeling a Virtual Enterprise

2.2.1 Modeling VE Resources as Active Objects

An important requirement in modeling the resources of the virtual enterprise is the ability to specify constraints associated with individual objects or with the associations among objects of various types. It is important that the objects which represent various types of NIIIP resources be "active objects" so that they can interact automatically with one another and take actions (i.e, perform operations automatically) under different external and/or internal events.

The active properties of objects can be introduced by an event-condition-action-alternative-action (ECAA) rule specification language and event/request monitor and rule processing facilities provided by an active knowledge base management system. An ECAA rule consists of three parts: event (E) specifications, condition (C) specification, and action/alternative action (AA) specification:

```
RULE rule_id;
  [TRIGGERED triggered_time trigger_operation,
         [, triggered_time trigger_operation,...]]
  [CONDITION condition_clause]
  [ACTION
     statement_list]
  [OTHERWISE
     statement_list]
END_RULE;
```

The event part E contains a set of (trigger time, triggering operation) pairs. The triggering operations can be either system-defined operations such as retrieval, update, deletion

and insertion or user-defined operations such as check-out-design and display-part. The time specification can be either Before, Immediate-after, or After which indicates that the processing of the C part should take place before, immediately after, or after the end of a transaction in which the triggering operation is to be performed. The C condition part of a rule specifies a potentially very complex data condition which exists in either the global VE knowledge base or the existing heterogeneous information systems. It evaluates to True or False. If the condition is true, an operation specified in the action part of the rule will be executed. Otherwise, an alternative operation will be performed. An example of a knowledge rule in NCL is given below:

```
RULE stock_and_use;
  TRIGGERED AFTER create(),
                  update(qty_on_hand,used_in)
  CONDITION EXIST p IN THIS *> [used_in] p:Product |
                  {100 < qty_on_hand < 1000}
  OTHERWISE
    "RULE: Part::stock_and_use\n".display();
    "*ERROR* If a part is used in some product, then
     the quantity on hand should be between 100 and
     1000".display();
    abort();
END_RULE;
```

The rule, stock_and_use, is triggered after the creation of a Part instance or after an update of the attribute qty_on_hand or used_in. The condition part of the rule is a guarded expression which states if the part is not used in a product, the rest of the rule is skipped. If it is used in a product, the second expression is evaluated to make sure that the quantity on hand is within the appropriate range. If it evaluates to True, no action is taken. Otherwise, an error message is displayed and the operation is aborted. For additional information on the NCL language and NCL rules, please refer to [28].

Thus, the modeling of virtual enterprise resources in the NIIIP environment is not restricted to the modeling of structural properties provided by the traditional database management systems (DBMSs). Nor is it limited to the modeling of the structural and behavioral properties offered by the existing commercial object-oriented DBMSs. Knowledge rules with triggers are also needed to declaratively capture all sorts of semantic constraints, expert knowledge, security and integrity restriction, business policies, government regulations, design and manufacturing constraints, mediation and negotiation rules, etc. Thus, the term "knowledge base" instead of "database" or "repository" is used.

Also, the information processing engine that supports the processing of the objects in the NIIIP environment should provide request monitoring and rule processing to automatically trigger object operations. Thus, the term active "knowledge base management" instead of "database management" or "repository management" is used.

2.2.2 Global Conceptual Schema

After the local schemas and the definitions of the global virtual enterprise resources discussed above are integrated, mediation rules are then introduced to specify the mediation processes needed to bridge the structural and semantic discrepancies between the modeled properties of heterogeneous systems. The result is a mediated global schema which provides the global view of the virtual enterprise's knowledge base. The global knowledge contains 1) the meta-data useful for the controlled access to data and program resources in a heterogeneous network (note: the actual data and programs are distributed among different systems), and 2) the common data which are used by the NIIIP-developed software systems to control the accesses of the data and software resources in the heterogeneous network.

As shown in Figure 2, the establishment of the mediated global conceptual schema is the result of the VE build-time activities. At run-time, the human and software clients would make use of the conceptual views defined over the global schema to issue all their object service requests. Service requests can be issued as remote methods calls, or as query statements in some query language. Higher-level languages such as KQML [8, 9, 10, 16] or a natural language can also be used. Alternatively, a graphical user-interface or a hypermedia facility can also be used to specify clients requests. In any case, a request made in any type of language or user-interface is eventually translated into method calls which are transported through ORB or other NIIIP communication facilities to relevant objects in the NIIIP environment. It is envisaged that in a fully realized NIIIP VE, the data and meta-data which are shared among software systems to be developed by NIIIP are stored and managed by some knowledge base management services. In addition, private data used by individual NIIIP software systems (or NIIIP objects) can be stored, maintained and used by these component systems themselves.

The above may only be a partial list of modeling requirements which are necessary for running a successful business in a virtual enterprise. However, it serves to show that object-oriented modeling of existing VE resources, work management, mediation, negotiation, and agents are important concepts and techniques for controlling virtual enterprises. Also, a VE knowledge base and a knowledge base management system are needed to support the operations of the software systems to be developed in the NIIIP project (i.e., the NIIIP virtual enterprise components). As will be described in the next section, the KBMS provides not only the traditional database management functions such as persistence, naming, concurrency control, querying, recovery, security, etc. but also the added functionalities such as request monitoring, rule processing, and constraint maintenance.

3 An Extensible Knowledge Base Management System

In this section, we shall describe an extensible knowledge base management system (KBMS) which has been under development at the University of Florida in the past several years. In Section 4, we shall describe how this KBMS technology is applied in the NIIIP project to provide rule-based interoperability.

Shown in Figure 3 is the architecture of the knowledge base management system, OSAM*.KBMS [25, 26, 27], and a number of supporting tools. A set of graphical tools called XGTOOLS (its earlier version was described in [17]) can be used to graphically define, edit, and browse schemas at build-time and to query the schemas against their instances stored in the knowledge bases at run-time. The KBMS provides an information modeling language called the NIIIP Common language (NCL [28]). This language is an integration of STEP' EXPRESS [14], OMG's IDL [21] and OSAM*.KBMS' K.3 programming language. The latter is the third version of an implemented language K reported in [4, 23, 24]. The implementation of NCL is based on a trans-

lation to K.3 and its processing is supported by the KBMS. NCL allows all things of interest to a virtual enterprise to be modeled as objects. Each object class can be defined by 1) its structural properties in terms of its associations with other classes, 2) its behavioral properties in terms of methods, and 3) its constraints in terms of keywords (the frequently used constraints) and Event-Condition-Action-AlternativeAction (or ECAA) rules. The modeler of a virtual enterprise can use the graphical editor to define schemas which can then be translated into NCL in its textual form. XGTOOLS also provides the editing facility for the user to enter NCL schemas in textual forms directly. In addition to these graphical tools, an EXPRESS-to-NCL compiler is available to import the existing EXPRESS schemas into NCL. A translator is also being developed to convert the existing IDL specifications into NCL. In the NIIIP project, EXPRESS schemas and IDL specifications that define the data and program interfaces of some legacy systems can be converted into NCL and additional resources of a virtual enterprise (data, software systems developed in the NIIIP project, hardware devices, tools, organizational structures, personnel resources, etc.) can be defined in NCL.

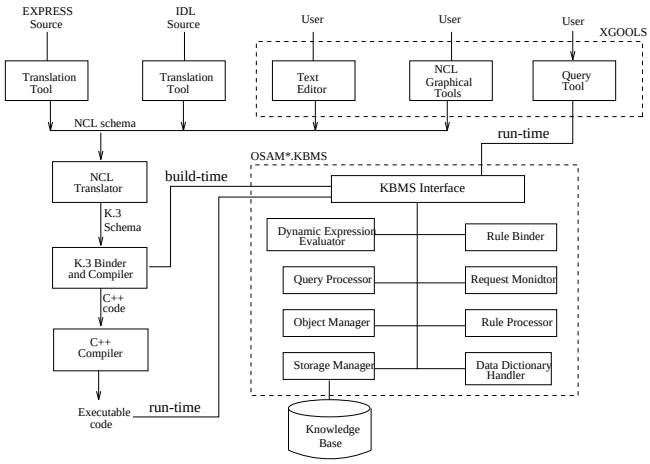

Figure 3: KBMS Components and Tools

The KBMS is extensible in the following way. The KBMS is developed based on a kernel object model. The kernel object model is used to model itself to produce a meta model (i.e., a model of the model itself). In the meta model, all the modeling constructs of the kernel such as Method, ECAA Rule, Entity, Domain and Association are defined by their corresponding meta classes. Thus, methods, rules, entities, domains, associations are first-class objects. The structural properties, methods and knowledge rules defined in each meta class represent the semantic properties of the corresponding modeling construct. Two built-in association types are recognized by the kernel object model: generalization which models the superclass-subclass or is-a relationship and Aggregation which models a-part-of, a-component-of or an-attribute-of relationship between classes. New object class types, association types, and constraints types can be added to the kernel object model by adding new meta classes, thus extending the modeling power of the underlying object model. In these meta classes, parameterized rules are used to specify the semantic properties of these new constructs. They have structures similar to the ECAA rules except that they contain variables which are bound to the attribute and class names of those user-defined classes which make use of the new class, association and constraint types. The binding process takes place during the schema compilation time. It translates parameterized rules into ECAA rules which are then bound to these user-defined classes as if the users have written the ECAA rules for the classes. In the development of the KBMS, all its operations are "driven" by the semantic contents of the meta model. Therefore, when the meta model is extended to include new class, association, and constraint types or when it is modified to specify the new semantics of some modeling constructs, the underlying object model of the KBMS is extended and/or modified. As a result, the operational behaviors of the KBMS is automatically extended and/or modified. A detailed description of the concept and technique of model, language and KBMS extensibilities is out of the scope of this paper. Interested readers are referred to our technical reports [18, 28].

The KBMS provides two general types of services: build-time and run-time services. At build-time, if NCL is used to model a software system and to implement the methods of its components (i.e., NCL can be used as a knowledge base programming language), the resulting model and method implementations are first translated into K.3 (our "internal" knowledge base programming language). Then, a K.3 compiler is used to generate .h files and C++ code that correspond to the structural and semantic properties of the K.3 schema. The resulting C++ program is then compiled and linked without the KBMS code to produce the executable code, as shown in Figure 3.

During the translation process, the KBMS provides the following build-time services:

(1) Data Dictionary Handler service. The K.3 compiler interacts with the Data Dictionary Handler service to store and retrieve meta-data to and from the knowledge base to support the translation of the K.3 schema. The access to the data and meta-data stored in the knowledge base is managed by an Object Manager and a low-level Storage Manager.

(2) Rule binding for model and system extensibility. The KBMS supports object model, language, and system extensibilities. The underlying object model of NCL can be extended by adding new constraint types, association types, and class types. The semantics of the new keyword constraints, association types and class types, are specified in the corresponding meta classes in the knowledge base as parameterized rules. The Dynamic Expression Evaluator and Rule Binder services of the KBMS are used to translate these parameterized rules into bound ECAA rules. These rules are bound to the classes in which the constraint, association and class types are used.

(3) Code generation for Request Monitoring and Rule Processing. During this translation, event specifications (i.e., E part) of all ECAA rules defined in the schema are used to produce C++ code for monitoring requests for various services (i.e., method calls) and are linked with the run-time Request Monitoring service of the KBMS. In other words, the generated code monitors the invocation of the corresponding method and invoke appropriate rules. The CAA parts of all the rules are also translated into C++ methods and are linked to the run-time Rule Processing service of the KBMS.

(4) Initiation of a Knowledge Base. An NCL schema created by a client using a copy of the XGTOOLS installed at the client site can be sent to the KBMS which initiates the knowledge base by storing the meta-information in the knowledge base. At run-time, the client can then populate the instances of the knowledge base using a query language.

At run-time, the KBMS provides the following services

to a) software clients which require KBMS services or b) human clients who access the KBMS through a Query Tool:

(1) Query Processing Service. Software and human clients can access the contents of a knowledge base by issuing queries to the KBMS either through the Query Tool or from executable code. An object-oriented query language (OQL [2]) is used for this purpose. The Query Processor of the KBMS is invoked to parse and process the queries to access the knowledge base.

(2) Request Monitoring Service. When a method is invoked either from the program code or from an OQL query, the Request Monitor for that method (i.e., the C++ code generated during build-time) determines whether any rules need to be invoked before and/or after this method is executed.

(3) Rule Processing Service. If it is determined that a rule needs to be invoked, the Request Monitor will invoke the Rule Processor (i.e., the C++ methods generated at build-time that correspond to that rule) to process the rule. If another method needs to be called within the processing of a rule, that method invocation may trigger another cycle of request monitoring and rule processing.

In summary, the KBMS described in this section differs from the traditional database management system (DBMS) in that it provides not only the traditional database management services (such as persistent, object management, transaction management, etc.), but also a set of knowledge base management services. Most notably, the KBMS provides a request/event monitoring service which monitors the invocation of the methods to automatically triggers the execution of rules by the rule processing service. In the next section, we also describe how we apply the KBMS technology in the NIIIP environment to provide a rule-based interoperability.

4 KBMS Technology Applied to NIIIP

The KBMS technology developed at the University of Florida is applied to the NIIIP architecture in the following two ways:

(1) The KBMS is used as a VE component to provide build-time and run-time services for the functioning of the NIIIP VE information infrastructure as described in Section 3. It provides the knowledge base management services to manage the NIIIP global knowledge base. As described before, the global knowledge base consists of the meta-data useful for the controlled access to data and program resources in a heterogeneous network and the common data which are used by the other NIIIP Components.

(2) The KBMS provides additional build-time services to distribute the Request Monitoring and Rule Processing services to component systems to achieve a rule-based interoperability. We shall focus on this application of the KBMS technology in this section.

4.1 Conventional CORBA Environment

As described in Section 2.1, the NIIIP system architecture is based on the Common Object Request Broker Architecture (CORBA) [21] introduced by the Object Management Group (OMG). In the conventional CORBA environment, the specification of a service is separated into an interface part and an implementation part. The interfaces of all the object services are defined in a standard Interface Definition Language (IDL) [21] and are translated into program language bindings (e.g., C++/C stubs and skeletons). The implementation of these systems can be in any programming language in which there is a language binding supported by the IDL compiler. Thus, a client software written in one programming language can invoke services (i.e., methods) written in another programming language, providing interoperability among heterogeneous systems.

However, the functioning of a virtual enterprise is not simply "a bunch of programs running and calling one another". There may be constraints and rules that need to be enforced in the invocation of these services. These constraints and rules may be the result of some governmental regulations and policies or some business rules and restrictions. For example, before a service (i.e., a method) is invoked, the NIIIP system may need to invoke a service to check the security, or trigger another service to perform some negotiation. After the service is performed, other services may need to be invoked to initiate some notification and perform some accounting. To enforce such constraints and rules using the conventional CORBA architecture, program code performing the enforcement has to be incorporated in the client and server programs. In other words, the control and logical relationship and constraints among the clients and servers are embedded in program code as method calls. If some governmental regulation or business rule changes, we have to go into the code to recode the enforcement. This can be a very time-consuming and costly process. Basically, the interoperability of the CORBA architecture is "method-based".

The basic idea of the "rule-based interoperability" is to separate the code of the implementation of the methods from the constraints and rules which define the control and semantic relationships among the methods. Thus, in the NIIIP environment, the interfaces of the servers are defined in NCL which, after the translation process described in a previous section, are essentially IDL specifications plus ECAA rules which capture the semantics of keyword or other user-defined constraints, association types and class types. In this manner, rules can be used to define not only the constraints associated with data but also the control logic that inter-relate the services of NIIIP VE servers. Furthermore, if the interaction among servers and clients changes, we do not have to go into the code to recode the program which carries out their interactions. Instead, the ECAA rules can be changed to reflect the new interrelationships among clients and servers. In the following subsection, we shall present the idea and technique of using these rules to achieve the rule-based interoperability.

4.2 A Compilation Approach to Rule-based Interoperability

The technique used to achieve the rule-based interoperability is to use the ECAA rule specifications in NCL together with the NCL's method specifications (which are equivalent to IDL specifications) to generate language bindings for client and server programs at the build-time. At run-time, the activations of object services as specified in the rules will be carried out automatically across the ORB, meaning programs will be calling each other through the ORB following the trigger and action specifications. For this reason, the proposed technique is called "rule-based interoperability". To achieve this, we need not only a mechanism to generate program stubs and skeletons to implement the methods in NCL specifications but also a mechanism to monitor the execution of the methods (request/event monitoring) and to trigger the processing of methods which implement the CAA

parts of rules (rule processing).

Figure 4 illustrates the process of generating program stubs and skeletons from NCL specifications. NCL is used to model the resources of different organizations which form a virtual enterprise. The integrated and mediated global NCL schema gives the global view of all the resources. It is first translated into the corresponding K.3 specification which defines the semantic properties of classes in terms of attributes and superclass-subclass associations, bound rules, methods specifications, and method implementations. Next, a K.3 compiler is used to translate the K.3 specification into 1) IDL method specifications and 2) extended methods in C++ or C code for implementing the rules. The generated IDL specifications are compiled by an IDL compiler to generate stubs and skeletons which are integrated with the extended methods that implement the rules. The integrated stubs and skeletons are then used by clients and servers to achieve run-time interoperability. We shall elaborate on this concept and technique by an example.

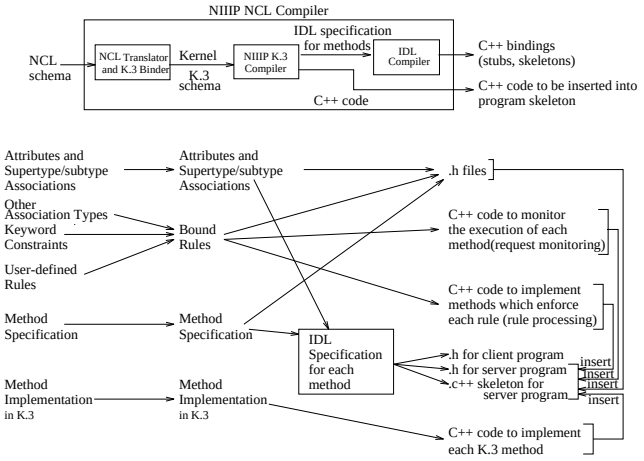

Figure 4: Generation of Program Bindings from NCL specifications

4.3 An Example of Rule-Based Interoperability

Figure 5 illustrates the compilation of a method (M1) with its associated before and immediate-after rules (R1 and R2) into the compiled code for distributed request/event monitoring and rule processing. Figure 6 illustrates the execution flow showing how a service request for M1 by a client is monitored to trigger the processing of the appropriate rules.

4.3.1 Compilation Phase

In Figure 5, the definition of a class (named EXAMPLE) in NCL is given which includes the specification of two methods (M1 and M4), and the specification of two rules (R1 and R2). As described before, NCL rules are Event-Condition-Action-AlternativeAction (ECAA) rules in which the triggering event can be the execution of any method. The actual NCL syntax for the two rules shown in Figure 5 are as follows:

```
RULE R1;
  TRIGGERED BEFORE M1()
  CONDITION X AND (*B, !C)
  ACTION
    M11();
  OTHERWISE
    M5();
END_RULE;

RULE R2;
  TRIGGERED IMMEDIATE-AFTER M1()
  CONDITION Y
  ACTION
    M4();
  OTHERWISE
    M9();
END_RULE;
```

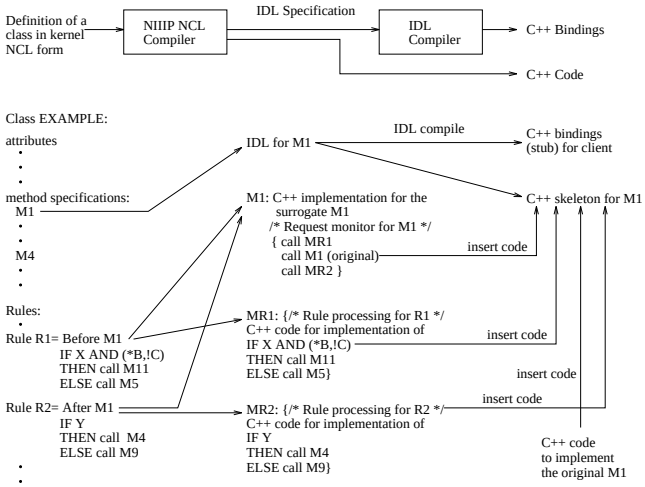

Figure 5: Compilation of an NCL method M1 and its associated rules

Rule R1 specifies that before method M1 is executed, the condition X AND (*B, !C) should be checked. This condition is an object pattern specification posted in an object-oriented query language OQL [2]. It verifies if there exists an X object instance which is associated with some object instance of B (* is the association operator) but is not associated with any object instance of C (! is the non-association operator). If the condition evaluates to True, then method M11 is called to perform some action. Otherwise, method M5 is called instead. Also, R2 specifies that immediately after method M1 is executed, condition Y is checked. If condition Y evaluates to True, then method M4 is called to perform some action. Otherwise, method M9 is called.

During the compilation of the class EXAMPLE by the NIIIP K.3 compiler, a C++ method is generated for each rule. For rule R1, the C++ code in method MR1 will evaluate the condition X AND (*B, !C) either locally or globally, and call method M11 or M5 based on the result of the evaluation. Similarly, for rule R2, a C++ method MR2 is generated.

For each method in the class, an equivalent IDL specification is generated. For example, an IDL specification would be generated for M1. Furthermore, a new implementation of M1 (i.e., a surrogate M1 in C++ code) is generated. The new implementation consists of three method calls. First, a call to method MR1 is made to process rule R1 (i.e., a BEFORE rule for M1). Then, a call is made to the original implementation of the method M1 (i.e., the original M1) to perform the requested service. Finally, a call is made

to method MR2 to process rule R2 (i.e., an IMMEDIATE-AFTER rule for M1).

In the final step of the compilation process, the IDL compiler is used to generate the C++ bindings for all the methods which have been specified in IDL, including method M1. After the bindings have been generated, the corresponding C++ implementation code for the surrogate M1, MR1, the original M1 and MR2 can be inserted into the skeleton of M1.

4.3.2 Service Request Execution Phase

Figure 6 illustrates the execution flow showing how a service request for M1 by a client is monitored to trigger the processing of the appropriate rules. A client makes a service request by using the programming language binding (i.e., IDL stub) generated by the IDL compiler for a particular method. When the request is made, the ORB would dispatch that request to the appropriate server to invoke the corresponding method. In the case of the method M4 (see Figure 6), there is no associated rules defined for it. Thus, the code which implements M4 is executed directly and no additional overhead is incurred.

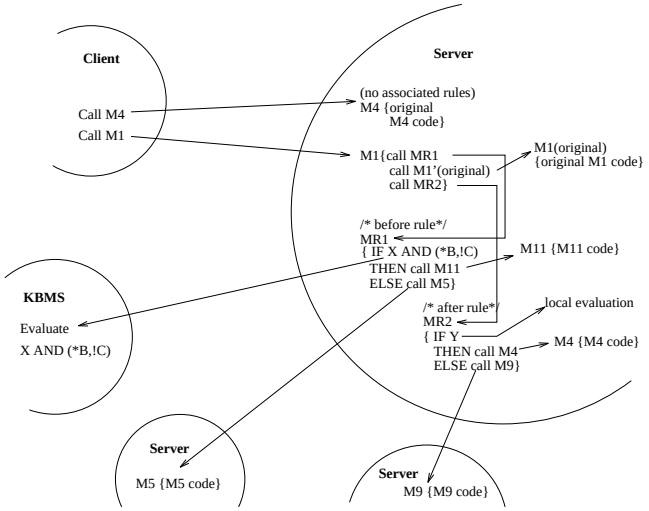

Figure 6: The Execution Flow

When a request for M1 is made, the ORB again dispatches that request to the appropriate server to invoke the implementation of M1. However, in this case, the original code for M1 is not invoked directly. Instead, the generated implementation (i.e., the surrogate M1 generated by the NIIIP K.3 compiler) is executed. First, the method MR1 is invoked. The execution of MR1 involves the checking of the condition X AND (*B, !C). In this case, let us assume that it requires a remote call to a service in the KBMS to verify the condition, as illustrated in Figure 6. The KBMS would return a True or False back to MR1. Based on the condition, MR1 either make a local call to method M11 or a remote call to M5. Note that each call to another method may trigger other rules, which will be handled by the Request Monitors and Rule Processors that have been compiled as methods and distributed among the corresponding methods.

After MR1 has been executed, a call to the original M1 is made to execute the code which implements the actual service requested by the client. After the original M1 has been executed, a call to MR2 is made to process the IMMEDIATE-AFTER rule associated with method M1. In our example, we assume the checking of the condition Y is done locally. Depending on the condition, either a local call to method M4 or a remote call to method M9 is made.

The above example shows that knowledge rules which capture all kinds of semantic information such as security and integrity constraints, expert knowledge, agent behaviors, business constraints, policies, etc. as well as rules which implement keyword constraints, association types, and class types can be used in conjunction with CORBA/IDL to achieve a rule-based interoperability in a distributed and heterogeneous environment.

5 Summary and Conclusion

In this paper, we have introduced an on-going project called NIIIP which aims to establish industrial information infrastructure protocols to facilitate the interoperability of heterogeneous computing systems in a virtual enterprise environment. We first described the overall NIIIP system architecture and the requirements for modeling various types of resources in virtual enterprises. We then described an object-oriented, extensible KBMS, a component of the NIIIP system architecture, and its build-time and run-time services to the software and human clients and servers in the NIIIP heterogeneous network. The build-time services include: (1) Data Dictionary Handler service, (2) Rule binding for model and system extensibilities, (3) Code generation for Request Monitoring and Rule Processing, and (4) Initiation of a knowledge base. The run-time services include: (1) Query Processing Service, (2) Request Monitoring Service, and (3) Rule Processing Service. We have also explained how the system extensibility is achieved by extending or modifying the underlying object model of the system. The resulting extended object model can more adequately model the complex structural and behavioral properties, constraints and associations found in virtual enterprises. Finally, we presented a compilation and distributed approach to achieve a rule-based interoperability among heterogeneous systems. We have shown that, by using the rule specification facility and the corresponding rule processing facility (rule binder and rule processor) provided by the KBMS, object classes defined in a semantics-rich object model can be translated into rules and IDL specifications. These rules and IDL specifications can then be translated into C++/C stubs and skeletons for binding client and server programs. At run-time, client and server programs can then activate their methods following the control information and logics specified in ECAA rules. The use of ECAA rules can also make a heterogeneous network system active since events can be automatically monitored and intelligent behaviors associated with objects can be automatically triggered. The rule-based interoperability and active feature are added values to the OMG/CORBA.

References

[1] R. Ahmed, et al., "The Pegasus Heterogeneous Multidatabase System," *IEEE Computer*, 24:19-27, 1991.

[2] A. Alashqur, S. Y. W. Su and H. Lam, "OQL– A Query Language for Manipulating Object-oriented Databases", in *Proc. of 15th Int. Conf. Very Large Databases*, Amsterdam, Netherlands, pp. 433-442, August 1989.

[3] M. Andersson, Y. Dupont, S. Spaccapietra, K. Yetongnon, M. Tresch, and H. Ye, "FEMUS: A Fed-

erated Multilingual Database System", Chapter 18 in Advanced Database Systems, N. Adams and Bhargava, R. (Eds.), Lecture Notes in Computer Science, Springer-Verlag, 1993, pp. 359-380.

[4] J. A. Arroyo-Figueroa, "The Design and Implementation of K.1: A Third Generation Database Programming Language," Technical Report, Database Systems R&D Center, University of Florida, August 1992.

[5] R. N. Bershad, D. T. Ching, E. D. Lazowska, J. Sanislo, and M. Schwartz, "A Remote Procedure Call Facility for Interconnecting Heterogeneous Computer Systems," *IEEE Transactions on Software Engineering*, 13(8), August 1987.

[6] E. Bertino, "Integration of Heterogeneous data Repositories by Using Object-oriented Views," *Proceedings of the 1st International Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan, April 1991, pp. 22-29.

[7] P. Drew, R. King, and J. Bein, "A la Carte: An Extensible Framework for the Tailorable Construction of Heterogeneous Object Stores," in Implementing Persistent Object Bases: Principles and Practice, *The Fourth International Workshop on Persistent Object Systems*, Morgan Kaufmann Publishers, Inc. 1990.

[8] T. Finin, R. Fritzson and D. McKay, "A Language and Protocol to Support Intelligent Agent Interoperability," appeared in the *Proceedings of the CE & CALS Washington '92 Conference*, June 1992.

[9] T. Finin, D. McKay, R. Fritzson, and R. McEntire, "KQML: An Information and Knowledge Exchange Protocol," in Kazuhiro Fuchi and Toshio Yokoi (Ed.), Knowledge Building and Knowledge Sharing , Ohmsha and IOS Press, 1994.

[10] T. Finin, R. Fritzson, D. McKay and R. McEntire, "KQML as an Agent Communication Language," to appear in *The Proceedings of the Third International Conference on Information and Knowledge Management* (CIKM'94), ACM Press, November 1994.

[11] J. Franchitti and R. King, "Amalgame: A Tool for Creating Interoperating, Persistent, Heterogeneous Components," Chapter 16 in Advanced Database Systems, N. Adams and Bhargava, R. (Eds.), Lecture Notes in Computer Science, Springer-Verlag, 1993, pp. 313-336.

[12] D. Heimbigner and D. McLeod, "A Federated Architecture for Information Systems," *ACM Transactions on Office Information Systems*, 3(3):253-278, July 1985.

[13] D. K. Hsiao, "Federated Databases and Systems (Parts I and II): A Tutorial on Their Resource Consolidation," *VLDB Journal*, Vol. 1, No. 1&2, July and Oct. 1992.

[14] Subcommittee 4 of ISO Technical Committee 184, "Product Data Representation and Exchange - Part 11: The EXPRESS Language Reference Manual," *ISO Document*, ISO DIS 10303-11, August 1992.

[15] W. Kim, et al., "On Resolving Schematic Heterogeneity in Multidatabase Systems," in *Distributed and Parallel Databases*, 1:251-279, 1993.

[16] Y. Labrou and T. Finin, "A semantics approach for KQML – a general purpose communication language for software agents," to appear in *the Third International Conference on Information and Knowledge Management* (CIKM'94), November 1994.

[17] H. Lam, S. Y. W. Su, et al., "GTOOLS: An Active GUI Toolset for an Object-oriented KBMS," *International Journal of Computer System Science and Engineering*, Vol.7,No.2,pp. 69-85, April 1992.

[18] H. Lam, S. Y. W. Su, et al., "Model Extensibility in an Extensible Knowledgebase Management System," submitted to *IEEE Transactions on Knowledge and Data Engineering*.

[19] W. Litwin, L. Mark, and N. Roussopoulos, "Interoperability of Multiple Autonomous Databases," *ACM Computing Surveys*, 22:267-293, 1990.

[20] NIIIP Consortium, "NIIIP Reference Architecture: Concepts and Guidelines," NIIIP Publication NTR95-01, Jan 1, 1995.

[21] OMG Committee, "The Common Object Request Broker: Architecture and Specification," *OMG Document*, Revision 1.1, No. 91.12.1, December 1991.

[22] A. Sheth, and J. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Survey*, 2(3), Sept. 1 990, pp. 183-236.

[23] Y. M. Shyy, S. Y. W. Su, "K: High-level Knowledge Base Programming Language for Advanced Database Applications", *ACM SIGMOD Int'l Conf. on Management of Data*, pp. 338-347, 1991.

[24] Y.M. Shyy, J. Arroyo-Figueroa, S. Y. W. Su, and H. Lam, "The Design and Implementation of K: A High-level Knowledge Base Programming Language of OSAM*.KBMS," accepted for publication in the *VLDB Journal*, 1995.

[25] S. Y. W. Su, V. Krishnamurthy and H. Lam, "An Object Oriented Semantic Association Model (OSAM*)," *AI in Industrial Engineering and Manufacturing: Theoretical Issues and Applications*, American Institute of Industrial Engineering, 1989.

[26] S. Y. W. Su and H. Lam, "An Object-oriented Knowledge Base Management System for Supporting Advanced Applications," *Proc. of the 4th Int'l Hong Kong Computer Society Database Workshop*, pp. 3-22, December 12-13, 1992.

[27] S. Y. W. Su and H. Lam, et al., "OSAM*.KBMS: An Object-Oriented Knowledge-Base Management System for Supporting Advanced Applications", *Proc. of the 1993 ACM SIGMOD Int'l Conf. on Management of Data*, pp. 540-541, 1993.

[28] S. Y. W. Su and H. Lam, et al. "NCL: A Common Language for Achieving Rule-Based Interoperability among Heterogeneous Systems", Submitted for publication *Journal of Intelligent Information Systems, Special Issue*, 1995.

[29] G. Thomas, et al., "Heterogeneous Distributed Database Systems for Production Use," *ACM Computing Survey*, 22:237-266, 1990.

[30] G. Wiederhold, "Intelligent Integration of Information," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington, D.C., May 1993, pp. 434-437.

[31] J. C. Wileden, A. L. Wolf, W. R. Rosenblatt, P. L. Tarr, "Specification Level Interoperability," *Proceedings of the Twelfth International Conference on Software Engineering*, Nice, March 1990.