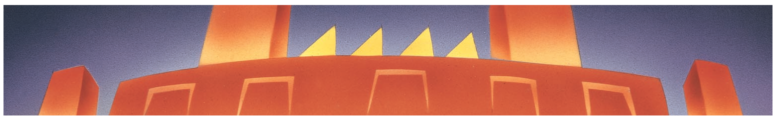
Martin Hardwick, David L. Spooner, Tom Rando, and K. C. Morris



Sharing Manufacturing Information in Virtual Enterprises

Two manufacturers cannot collaborate if the systems in one cannot process the data in the other. The prototype information infrastructure described here uses standards to let the applications of one manufacturer process the databases of another.

Recent advances in communications technology make it possible for manufacturers to transmit data to each other in fractions of a second. However, if these corporations do not use the same software tools, understanding of the data can be delayed for weeks or months while employees purchase and learn to use new tools. This article describes ongoing research into an information infrastructure that seeks to use standards (formal and de facto) to reduce the problems that occur when manufacturers want to use different tools to process each other's data.

Communicating information between different software tools is a problem common to many application domains. For example, many researchers who have collaborated on research projects are familiar with arguments over which word processor to use for a particular project. In manufacturing, the multifaceted nature of design information makes communications particularly difficult. A mechanical design may contain geometric, tolerance, material, process control, and many other kinds of information. An information infrastructure is needed to allow manufacturing applications to communicate efficiently. The infrastructure should allow engineers to use familiar applications whenever possible. A successful infrastructure will reduce the time to market for new products while letting multiple organizations apply their specialties to product development.

Manufacturing has a long history of reducing communications problems through standards. Some of the most significant advances in the industrial revolution occurred when standards were established. For example, drafting standards for drawings represented a significant contribution in the 18th and early 19th centuries. In the modern era, electronic communications are more important every year. Standards are needed so manufacturers can communicate efficiently through the Internet. The required standards are becoming increasingly complex as the range of information the industry wants to communicate becomes wider. Demand for these standards is increased by the desire to allow corporations to communicate in virtual enterprises.

A Prototype Information Infrastructure

This article describes a prototype information infrastructure for virtual manufacturing enterprises. This infrastructure combines the Internet with the STandard for the Exchange of Product Model Data (STEP) for data exchange (see the first sidebar, "Development of the STEP Standard") and the Common Object Request Broker Architecture (CORBA) standard for interoperation of application systems.[1]

[1] Microsoft OLE also defines a framework that allows applications to interoperate. CORBA is used here because, at the time of writing, OLE did not allow applications to interoperate over wide-area networks and did not use a general-purpose definition language to define its interfaces.

The prototype shows how applications described by CORBA can be applied to data defined by STEP on the Internet. The combination allows manufacturers to share information about products over the Internet while using their favorite tools to process the information. To create the infrastructure, the data definition language of CORBA, IDL, is combined with the data definition language of STEP, EXPRESS. These two languages have different purposes: IDL describes interfaces to applications; EXPRESS describes normalized data models. Both can be used to describe objects for manufacturing applications.

The next section describes some barriers preventing communications between manufacturing organizations; the section that follows describes an information infrastructure for manufacturing; the section after that describes unresolved issues; and the concluding section summarizes the entire article.

Problem Definition and Related Work

The design and manufacture of new products frequently requires the talents of many specialists. When many corporations combine their specialties to create a product, the result can be called a virtual enterprise. A virtual enterprise must be able to form quickly in response to new opportunities and dissolve just as quickly when the need ceases. From the information management point of view, communicating industrial information within a virtual enterprise offers many challenges. Barriers hindering communications include:

- Insufficient security controls. The corporations participating in a virtual enterprise are independent and frequently compete against one another.
- Loss of control over projects. Techniques used to control a project in one corporation do not generalize well to multiple corporations because of each corporation's different operating practices.
- Inability of application systems to interoperate. The data produced by the systems of one corporation cannot be read and processed by the systems of other corporations. This problem is further subdivided into:
- Semantic interoperability. Two applications cannot process each other's data because they cannot understand the internal organization of each other's data.
  — Code interoperability. Two applications cannot make use of each other's functionality because they cannot invoke one another's resources.
  — Unfamiliar technologies and application systems. A corporation's technologies and application systems are often complex and require extensive training to be used correctly.

Making communications more efficient requires overcoming these barriers, which are being addressed in several ways. For example, the computer and communications industries are developing secure and trusted communications protocols [4] and the security community is developing access controls [2]. Preventing the loss of control requires sophisticated workflow control systems [9] and engineering-change control systems [5], as well as experience. The need for training in the use of technologies is reduced by the information infrastructure described in this article, which focuses on

Development of the STEP Standard

The designers of STEP had to produce an extensible specification for product data for use by all countries of the world. The specification had to describe data used by many different kinds of applications in many different contexts, leading to the definition of normalized object models, called information models.

STEP uses the EXPRESS data definition language to define its information models; this language was defined by the STEP community. Many other languages were tried before EXPRESS was developed, but it was found that these languages were not well suited for use by mechanical engineers and were unable to create models of the complexity required for STEP. EXPRESS has a range of rich data structures, including an advanced form of inheritance. The STEP normalization process requires that a data structure be used to represent an integrity constraint whenever possible. If a data structure is not available for this purpose, the modelers describe the constraint using a procedural language that is also part of EXPRESS. Like any language, EXPRESS has weaknesses, but it is more powerful than most other popular data-definition languages.

STEP information models are developed through a four-stage process that should be familiar to most database designers:

- In the first stage, the scope of the model is defined. The result is called an Application Activity Model (AAM).
- The second stage defines the information requirements of the processes identified by the AAM. The result is called an ARM.
- The third stage maps the ARM into the normalized STEP database. The result is called an AIM. In this stage, each ARM entity is mapped to an equivalent set of AIM entities. Mapping the ARM to the AIM may require new entity definitions to be added to the AIM if information of the required kind has not previously been modeled in STEP.
- The fourth, and final, stage develops tests for the new information model to verify that instances of the AIM model meet all constraints described by that model. The AIM is the standard. The other models assist users and developers in understanding and deploying that standard.

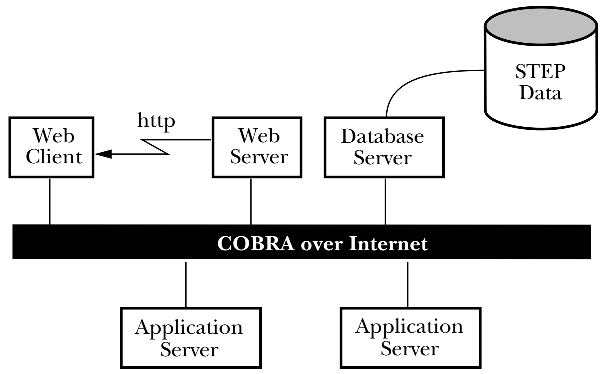


Figure 1. Architecture for the information infrastructure

interoperation of application systems.

Technology used to communicate data must be cost-effective, flexible, and portable. As a result, standards (formal and de facto) play an important role in developing these technologies. The information infrastructure described here depends on three standards:

- The World-Wide Web, which encapsulates communications protocols to organize and access data across the Internet [11].
- The STEP standards for exchange of product model data, which allow the semantics of manufacturing information to be understood by multiple applications [6].
- CORBA, which allows applications to use one another's resources by supporting message calls between objects through a network [1].

Figure 1 shows an abstract view of an information infrastructure for virtual industrial enterprises. Connected through this communication mechanism are a number of servers, repositories, and browsers. Some or all of these components are duplicated at each corporation in a virtual enterprise. The specific collection of servers for a particular corporation depends on the goals of that corporation. (See the second sidebar, "Using the Information Infrastructure," for an example of how the infrastructure is applied in practice.)

Figure 2. Some STEP definitions used in the database

```
ENTITY product
            id          : identifier;
            name        : label;
            description             : text;
            frame_of_reference              : SET [1:?] OF product_context;
END_ENTITY
```

– Product is the EXPRESS entity used in AP203 to describe a product [ISO94c]. This entity is an entry point into databases described by AP203. All of the entities that describe a product (including its geometry) may be found from this entity, but the navigation can be complex because the pointers in a STEP model describe constraints so they are sometimes arranged in the wrong direction for navigation.

Figure 2a. A Product Entity in EXPRESS

```
ENTITY composite_curve
            SUBTYPE OF (bounded_curve);
            segments            : LIST [1:?] OF composite_curve_segment;
            self_intersect      : LOGICAL;
DERIVE
            n_segments          : INTEGER := SIZEOF(segments);
    closed_curve   : LOGICAL := segments[n_segments].transition <> discontinuous;
WHERE
            WR1: ((NOT closed_curve) AND (SIZEOF (QUERY (temp <* segments |
                        temp.transition = discountinuous)) = 1)) OR
                        ((closed_curve)AND (SIZEOF (QUERY(temp <* segments |
                        temp.transition = discountinuous)) = 0));
END_ENTITY; – composite_curve
```

– An entity describing a geometry data type. This entity uses many of the features of EXPRESS including the use of derived attributes (keywork: DERIVE) to describe attribute values that can be computed from the necessary set of attrubutes, and the use of local constraints (keyword: WHERE) to describe rules that cannot be encapsulated using data structures.

Figure 2b. A Composite Curve in EXPRESS

Implementation

An information infrastructure for virtual industrial enterprises should have the following qualities:

- Performance. The data needed for an operation should be delivered in a single communication, because every additional communication causes additional delay.
- Concurrency. The data delivered should contain only necessary information, because delivering (and locking) unnecessary information prevents others from working concurrently on the database.
- Comprehension. The systems that receive the data should be able to process that data.

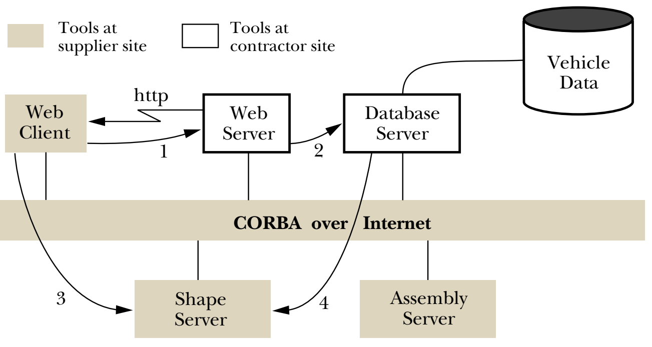
Systems can process data only when they understand the internal organization of the data. The complexity of manufacturing data is a key issue in manufacturing applications [10]. A small administrative database for a university may be defined using as few as 20 relations. Databases for product information seem to start at the equivalent of 300 relations and grow larger as the product being modeled becomes more complex. The number of definitions needed for manufacturing applications is also a significant cost concern. If an application contains 300 definitions and requires 100 lines of code on average for each definition, the application contains 30,000 lines of definition code. Manufacturing applications are difficult to write and maintain because of this abundance of code.

The STEP standard is designed to allow different manufacturing applications to share product data. The STEP designers created normalized object models, called "information models," for manufacturing applications. These models are written in an object-oriented data-specification language called EXPRESS [7, 12]. The models are normalized for the same reasons relational databases are normalized—to make the data more accurate and to make it easier for multiple applications to share the data [3]. Normalizing

Using the Information Infrastructure

Suppose that a vehicle manufacturer forms a virtual enterprise with one or more smaller supplier companies to produce a customized version of one of its vehicles. We concentrate on the vehicle manufacturer and a supplier that provides a customized door for the vehicle. The figure below describes how the door supplier can access and display a geometric shape stored in a database belonging to the vehicle manufacturer using four steps:

- The user uses a World-Wide Web browser to navigate to a page containing data of interest and selects an icon for a geometric shape.
- The web server communicates with the database server, and the user is asked if he or she wants to check this object out of the database server.
- The checkout is validated, and an identifier for the shape is sent to an application server, or shape server.
- The application server requests the data needed to service the request from the database server and processes the request by displaying the shape.



1. User navigates to a page containing data of interest using a World Wide Web browser and selects an icon for a geometric shape.

2. The Web server communicates with the database server and the user is asked if he wants to check this object out of the Database Server.

3. The checkout is validated and an identifier for the shape is sent to an Application Server (Shape Server).

4. The Application Server requests the data needed to service the request from the Database Server and processes the request by displaying the shape.

the object models in STEP is necessary but expensive—necessary because manufacturing data needs to be shared by as many applications as possible, and expensive because normalization often makes a model harder to process (see Figure 2a).

As an experiment, we created a STEP database for an axle of the Humvee all-terrain vehicle. The database was created by translating a Pro/Engineer CAD system model of the axle to STEP. The information model for the database contains 500 data definitions.[2] A file containing the database for the axle contains two megabytes of data, stored as 80,000 instances. If this experiment is scaled up to create a complete database for a motor vehicle, we postulate that the database will be 1,000 times larger. Note that this database covers only the mechanical assembly data for a Humvee vehicle. When STEP expands to include other kinds of data, the number of definitions in the database and the number of data instances increase further.[3]

The information infrastructure requires applications to process information defined by STEP. On one level, this can be achieved by translating information described by EXPRESS into the IDL language of CORBA. (See the third sidebar, "Relationship Between the STEP and OMG Standards.) On another level, an application system must be able to process the information described by the STEP model. For example, a CAD system must find the data in a STEP database needed to display a shape. The following IDL specification defines an operation a CAD system might offer to other systems that want to display geometric shapes.

```
interface CAD {
BOOL display (in shape S);
};
```

When an application requests this operation, the system that displays the shape must find all the details about the shape. This can be difficult because the information required may be linked to other information through a variety of relationships in the database. For example, each instance of the product in Figure 2a may contain product versions defined by product definitions using shapes [8]. The infrastructure needs to find this information and send it to the server in a form that can be processed efficiently.

Figure 3 describes the architecture we used to build an application server. The top half of the figure shows the process used to define the interfaces of the server. The bottom half shows how an instance of an application server responds to client requests by processing a combination of STEP and CORBA data.

The top half of Figure 3 shows two EXPRESS models:

- The first, called the Application Interpreted Model (AIM), is defined by STEP and describes the full STEP database.
- The second, called the Application Reference Model (ARM), is also defined by STEP, but for information purposes only. This model describes the objects in the AIM that can be processed by the application servers. When a client requests an operation on one of these objects (using CORBA), the application server receives the request and processes it by reading the details of the object as described by the AIM. The details are delivered to the application server by a database server.[4] The

[2] The information model, known as AP203 [8], defines information for three-dimensional configuration-controlled design.

Relationship Between the STEP and OMG Standards

The STEP standard and the OMG standards have been developed by two nearly disjointed standards communities. Yet the integration of the two standards promises to satisfy urgent needs in the industrial sector, as well as to enhance the two standards communities. STEP gains a new implementation method in the form of distributed objects, and OMG gains a facility that serves the broad (and expanding) domain of product information.

The current approach to integration entails three efforts:

- Mapping EXPRESS to IDL;
- Mapping STEP services to the object services of OMG; and
- Consolidating the EXPRESS and IDL language bindings.

The goal of consolidation is to provide compile-time portability between the C++ language bindings of EXPRESS and of IDL. The idea is that a mapping from EXPRESS to IDL specifies the mapping of EXPRESS constructs common to all language bindings (e.g., primitive types, application entities, collection types). This mapping forms the basis for a number of programming language bindings (via IDL) to STEP. However, these bindings do not support the complete range of EXPRESS semantics; the remaining semantics may be implemented in application code by subtyping from constructs in the IDL bindings. The end result is that IDL-binding applications can be compiled into STEP-binding implementations and that STEP-binding applications can be supported by implementations whose base functionality is provided by an IDL implementation.

[3] As this issue went to press, more than 30 models were in the process of being defined by STEP. They range from the relatively general model described by AP203 to more specific models for sheet metal die design (AP207) and printed circuit board design (AP210).

[4] The delivery protocol conforms to the specifications described by the OMG persistent object service [1].

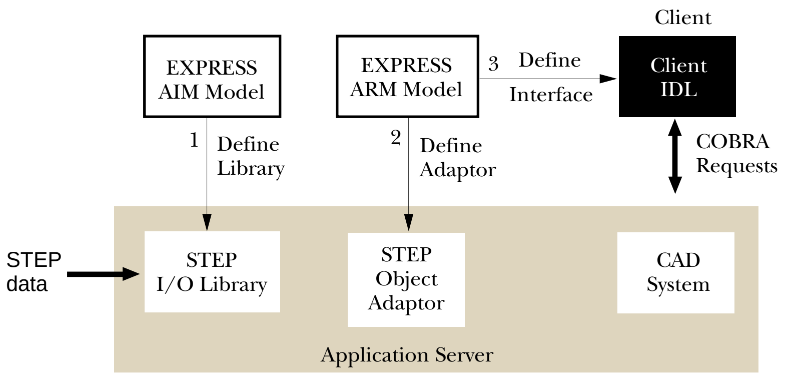


Figure 3. Defining an application server for the infrastructure

delivery is shown by the large arrow on the left side of Figure 3. The CORBA request from the client to the server is shown by the large arrow on the right side of Figure 3.

Suppose a client requests an operation on an object. This object may have been found by searching a World-Wide Web page, by querying a relational database, or as a result of another computation. An application server that can service the request is invoked by CORBA[5] and given a reference to the object. This reference contains pointers into the AIM data. The adaptor in the server requests delivery of the detailed data addressed by these pointers from the database server; the details are processed using the STEP input/output (I/O) library; and the operation is executed using the application code of the CAD system.

An EXPRESS compiler automates three functions for the application server:

- It generates code from the AIM model. This code is used by the application server to define the STEP I/O library that reads and traverses the data delivered by the database server.
- The EXPRESS compiler generates code from the ARM model to define an Object Adaptor[6] for the application server. This adaptor lets the application server receive client requests for the range of objects defined by the relevant portion of the ARM model.
- It generates IDL from the ARM model so that a client can request operations from the server. This IDL is the third model shown in the top half of Figure 3. Only the portion of the IDL that defines data structures, such as shape, is generated by the EXPRESS compiler.

[5] If multiple application servers service the request, one is selected through protocols defined by CORBA [1].

[6] In CORBA, an Object Adaptor makes resources in an application available to external systems via IDL.

```
VIEW shape
FROM (shape_definition_representation AS aim_s)
WHEN TRUE;

    off_shape  := aim_s;

    WHEN  ('PRODUCT_DEFINITION'  IN
               TYPEOF  (aim_s.definition.definition);
    BEGIN
               with_product_definition := aim_s.definition;
    END;

    user_name  := off_shape.definition.name;

    - - Expressions to calculate properties and relationships not shown

END_VIEW:
```

Figure 4a. Partial EXPRESS-V mapping for an ARM Object.

```
ENTITY shape
    off_shape  :  shape_definition_representation; - - pointers to AIM
    with_product_definition  :  property_definition;

    user_name  :  STRING;          - - user level properties
    key  :   STRING;

    describe  :  LIST OF assembly; - - relationships to other ARM objects
    shape_in  :  LIST OF part;

END_ENTITY;
```

Figure 4b. Partial EXPRESS Definition for an ARM Object.

```
interface shape {
    attribute shape_definition_representation off_shape;
    attribute property_definition with_product_definition;

    attribute string user_name;
    attribute string key;

    attribute sequence<assembly> describes;
    attribute sequence<part> shape_in;

};
```

Figure 4c. Partical IDL Description for an ARM Object.

Figure 4. Code fragments that illustrate the operation of EXPRESS-V

The methods that can be applied to a data structure are added to the IDL by programmers and database administrators. Any method can be added to the IDL—provided an application server contains code to process the method. For example, a server that contains a CAD system can display a shape.

The model used to define the IDL interfaces for the application server can be any EXPRESS model. We used the ARM defined by STEP, because a large number of applications can process the objects described by the model. However, any model can be used—provided the relationship between the AIM model and the ARM model is defined by a database administrator. For example, one application may have a view of the product database that refines that database into collections of shapes organized into assemblies—the view we used—while another application may refine the database into a view that recognizes only parts manufactured through a casting process.

Figure 4. Code fragments that illustrate the operation of EXPRESS-V

The relationship between the AIM model and the ARM model is defined by a database administrator using a language called EXPRESS-V [13]. This language maps data instances between different EXPRESS models. Figure 4 shows code fragments to illustrate the operation of EXPRESS-V. Figure 4a shows an EXPRESS-V mapping that creates a shape object; Figure 4b shows the EXPRESS definition for the shape object, and Figure 4c shows the IDL description for the same shape object generated from the EXPRESS code by the EXPRESS compiler.

The EXPRESS-V code in Figure 4a shows how an instance of a shape object is created whenever a shape_definition_representation[7] entity is found in an AIM database. The instance contains the attributes shown in Figure 4b. The exact content depends on a number of conditions formulated in the EXPRESS-V code. (Only a fragment of the code is shown.) The contents are divided into three types of data, each created from the AIM data for the ARM data by the EXPRESS-V code:

- A series of pointers into the AIM. The AIM pointers address the underlying data that defines the ARM object. These pointers make it easier for a process to traverse that AIM data because they address all content directly.
- A series of STRING attributes. These attributes describe "user properties" that can be inserted into indexes and catalogs, such as World-Wide Web browsers and relational databases, to make the ARM objects easier to locate.
- A series of relationship pointers. These pointers address other ARM objects related to an ARM object because of underlying relationships in the AIM. For example, if a shape is used to define a version of a product, this product is addressed by the shape_in relationship. These relationships are useful when the database is being traversed.

The infrastructure shown in the figures was tested using the Humvee axle example already described. In the experiment, a STEP database was built for the axle using STEP data translated from the Pro/Engineer CAD system. The original data con-

```
VIEW shape
FROM (shape_definition_representation AS aim_s)
WHEN TRUE;

    off_shape  := aim_s;

    WHEN  ('PRODUCT_DEFINITION'  IN
                TYPEOF  (aim_s.definition.definition);
    BEGIN
                with_product_definition := aim_s.definition;
    END;

    user_name  := off_shape.definition.name;

    - - Expressions to calculate properties and relationships not shown

END_VIEW:
```

Figure 4a. Partial EXPRESS-V mapping for an ARM Object.

```
ENTITY shape
    off_shape  :  shape_definition_representation; - - pointers to AIM
    with_product_definition  :  property_definition;

    user_name  :  STRING;           - - user level properties
    key  :   STRING;

    describe  :  LIST OF assembly; - - relationships to other ARM objects
    shape_in  :  LIST OF part;

END_ENTITY;
```

Figure 4b. Partial EXPRESS Definition for an ARM Object.

```
interface shape {
    attribute shape_definition_representation off_shape;
    attribute property_definition with_product_definition;

    attribute string user_name;
    attribute string key;

    attribute sequence<assembly> describes;
    attribute sequence<part> shape_in;

};
```

Figure 4c. Partical IDL Description for an ARM Object.

[7] See ISO 10303-203 [7] for a definition of the shape_definition_representation entity.

tained 92,000 STEP AIM entity instances resolved into 84 ARM object instances by an EXPRESS-V process. Two application servers were implemented—one displayed STEP shapes, and the other displayed STEP shapes and assemblies. The first server contained the AutoCAD CAD system, and the other server contained the Pro/Engineer CAD system. World-Wide Web pages were built so that users could request operations on the ARM objects from any workstation on the Internet.

Open Issues

The architecture shows promise, but many issues need further investigation. For example:

- Performance. The architecture partitions a database into ARM objects describing views of the AIM database. Our prototype supports materialized views in which the AIM instances belonging to an ARM instance are computed before a transaction is executed. However, depending on the view, some operations may require delivery of several ARM objects to a client while others require only one object. For example, an assembly operation might require all shapes in that assembly to be delivered to its process, while an operation that displays a shape requires only that shape. In the prototype, both applications cannot be supported efficiently if the goal is to minimize the number of transmissions to a client without sending unnecessary data.
- Concurrency. The architecture lets multiple users change a database concurrently. Concurrency implies that different users may give different values to the same data instances in a database. Contradictory changes can be prevented through concurrency control or allowed through version control. Version control is interesting because of potential overlaps in the changed data. For example, one user may change the scope of an authorization while another user changes the definition that has been authorized. When the two units are integrated back into the database, conflicts between these actions must be detected and resolved [5].
- Comprehension. Restricted views of product data are easier to understand, but larger application bases exist for more general-purpose views. For example, many CAD applications can display three-dimensional shapes, but relatively few applications can process information about near net castings.

These issues suggest that a more sophisticated database server is needed to support multiple, dynamic views. The demonstration used a simple database server that divided all of the data in a database into a set of files—one file per ARM object.

Figure 5 shows an early version of a database server we are implementing to support multiple views. In this architecture, the "enter" component is used to divide data produced by a CAD system into modules

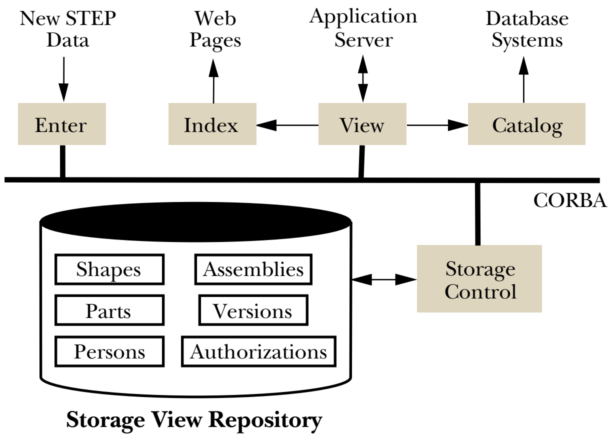


Figure 5. Multiview database server

with keys (see Figure 4). After division, the enter component gives the modules to the storage-control module for storage in a repository. The storage-control part of the server is similar to the original database server, except the new part is responsible for replacing old versions of modules with new versions and integrating the new versions into the repository. This integration requires extensions to EXPRESS-V, so the differences between the old and new versions can be identified and resolved.

The "view" component in the new database server is used to support multiple dynamic views. This component loads data from the storage view repository as necessary and computes the view needed by a client. The data in a view is then transmitted to the client. An extension designed for EXPRESS-V allows the data limits of each view to be computed at run time using a program designed by a database administrator. This program describes which AIM instances are to be included with an ARM object and which AIM instances are to be excluded [13]. For example, if a user selects an assembly object, all of the AIM instances in all of the shapes in that assembly are included in the transmission by writing lines of code to include those AIM instances with the ARM object.

Some views are materialized by the "index" and "catalog" components, because these components put information about the ARM objects in the views onto Web pages and into relational databases. When a user selects an ARM object using a Web page or a relational database query, the modules in the storage view repository containing the information needed by the object are loaded into the view component, the appropriate data limit program is executed, and the data needed to support the object is delivered to the application server.

Conclusions

If we compare the corporations in a virtual enterprise to an orchestra, CORBA defines the instruments for the orchestra and STEP defines the music. CORBA is necessary because we need to know which instruments are available. STEP is necessary because we want the orchestra to play together. By integrating STEP and CORBA, we let a database administrator decide which instruments are to be played for each tune.

Integration of STEP and CORBA must be performed on two levels:

- The languages of the two standards—EXPRESS and IDL—must be integrated so that IDL objects can process STEP data.
- The "points" of integration must be identified.

A typical STEP file contains many megabytes of data and hundreds of thousands of data instances. Groups of these instances define "themes" that can be played by the orchestra's IDL instruments. We have described how these themes can be identified using a view definition language called EXPRESS-V.

Many more issues can be investigated to make virtual enterprises communicate industrial information more efficiently. For example, the technologies needed to define a conductor for the orchestra are discussed in the section "Problem Definition and Related Work" but are missing from our architecture. Other issues that can be investigated include the management of very large databases, version control, conflict resolution, and increasing the use of higher-level semantics.

The project described in this article made the communication of industrial information within virtual enterprises more efficient by combining STEP and CORBA. Our contribution has been to apply a traditional technology—database views—to a new domain and use it to show how an information infrastructure for industrial applications can be constructed. In the article's context, the lesson learned from STEP is that more complicated databases can be shared if a richer language is used to describe these databases. The lesson learned from CORBA is that distributed applications can interoperate on a network if they define their resources using a neutral language. The lesson learned from the World-Wide Web is the power of using a standard data format to describe information. The lesson learned from our project is that views allow general-purpose manufacturing information to be shared by many different kinds of specialists.

Acknowledgments
This work has been partially funded by the National Science Foundation, Grants IRI-9224783 and ECS-9422713, and by the National Industrial Information Infrastructure Protocols Consortium, United States Air Force, Agreement F33615-94-2-4447. Any opinions expressed or implied are those of the authors.

References
1. The Common Object Request Broker: Architecture and Specification (CORBA) Revision 1.2. OMG TC Doc. 93.12.43, Object Management Group, Framingham, Mass., Dec. 1993. (Available by anonymous ftp from ftp.omg.org.)
2. Department of Defense Trusted Computer System Evaluation Criteria. DoD Document 5200-28-STD, U.S. Department of Defense, 1985.
3. Elmasri, R. and Navathe, S. *Fundamentals of Database Systems*, 2d ed., Benjamins/Cummings, Redwood City, Calif., 1994.
4. Ganesan, R., and Sandhu, R. Securing cyberspace, *Commun. ACM 37*, 11 (Nov. 1994), 29–31.
5. Hardwick, M., Downie, B., Kutcher, M., and Spooner, D. Concurrent engineering with delta files. *IEEE Computer Graphics and Applications 15*, 5 (Jan. 1995), 62–68.
6. Industrial Automation Systems—Product Data Representation and Exchange—Part 1, Overview and Fundamental Principles, ISO 10303-1. International Organization for Standardization, Geneva, Switzerland, 1994.
7. Industrial Automation Systems—Product Data Representation and Exchange—Part 11, Description Methods: The EXPRESS Language Reference Manual, ISO 10303-11. International Organization for Standardization, Geneva, Switzerland, 1994.
8. Industrial Automation Systems—Product Data Representation and Exchange—Part 203, Application Protocol: Configuration Controlled Design, ISO 10303-203. International Organization for Standardization, Geneva, Switzerland, 1994.
9. McCusker, T. Workflow takes on the enterprise. *Datamation 39* (Dec. 1993), 88.
10. Morris, K.C., Mitchell, M., Dabrowski, C. and Fong, D. Database management systems in engineering. In *The Encyclopedia of Software Engineering*. Wiley, New York, 1994, pp. 282–308.
11. Obraczka, K., Danzig, P. B., and Li, S.-H. Internet resource discovery services. *IEEE Computer, 26*, 9 (Sept. 1993), 8–22.
12. Schenck, D.A. and Wilson, P.R. *Information Modeling the EXPRESS Way*. Oxford University Press, Oxford, England, 1994.
13. Spooner, D., Hardwick, M., and Wen, J. The EXPRESS-V language manual. Tech. Rep., Laboratory for Industrial Information Infrastructure, Rensselaer Polytechnic Institute, Troy, NY, 1995 (http://www.rdrc.rpi.edu).

About the Authors
MARTIN HARDWICK is a professor in the Computer Science Department and principal investigator in the Laboratory for Industrial Information Infrastructure at Rensselaer Polytechnic Institute. **Author's Present Address:** Laboratory for Industrial Information Infrastructure, Rensselaer Polytechnic Institute, Troy, NY 122180-3590; email: hardwick@rdrc.rpi.edu

DAVID L. SPOONER is a professor in the Computer Science Department and principal investigator in the Laboratory for Industrial Information Infrastructure at Rensselaer Polytechnic Institute. **Author's Present Address:** Laboratory for Industrial Information Infrastructure, Rensselaer Polytechnic Institute, Troy, NY 122180-3590; email: spooner@cs.rpi.edu

TOM RANDO is a principal engineer in General Dynamics' Electric Boat Division. **Author's Present Address:** General Dynamics, Electric Boat Division, 76 Eastern Point Road, Groton, CT 06340-4948.

K.C. MORRIS is a computer scientist at the National Institute of Standards and Technology. **Author's Present Address:** National Institute of Standards and Technology, Building 220, Room A127, Gaithersburg, MD 20899–0001.