An Enterprise Model Repository: Architecture and System

HEESEOK LEE, Korea Advanced Institute of Science and Technology
JAE-WOO JOUNG, Korea Advanced Institute of Science and Technology

The management of organizational knowledge has become increasingly important since knowledge was identified as the new basis of competition in post-industrial society. One of the key issues in knowledge management is to put knowledge into a repository where it can be easily stored, reused and re-created. This paper proposes a model repository and a type of knowledge management using a new repository system. The emphasis of the model repository is on abstracting knowledge in the form of schematic models. Major benefits include model independence and model integration. A metaschema is constructed according to six conceptual components that can include a variety of models. A case and prototype system illustrate the practical feasibility of model repository and knowledge management.

Because of the pressure of competition and the rapidly changing business environment, organizational knowledge emerges as an important reusable resource. Knowledge management is based on the viewpoint that an organization can take proper steps to clear obstacles lying ahead and give good services to its customers by not only differentiating internal knowledge resources for competitive edges, but also managing them efficiently (Dutta, 1997). Davenport and Prusak (1998) define knowledge as a fluid mix of framed experience, values, contextual information, and expert insight. Typically, knowledge can be categorized into two types: tacit and explicit. Tacit knowledge is based on personal subjective experience and hard to formalize. In contrast, explicit knowledge is transmittable in a formal and systematic manner.

Nonaka and Takeuchi (1995) investigated the dynamic knowledge creation phases, such as socialization, externalization, combination and internalization through reciprocal reactions between tacit knowledge and explicit knowledge. Effective knowledge management provides companies with strategic benefits, speeds changes, reduces cost, and enhances quality. Information technologies (eg., a groupware like Lotus Notes or Internet) can help represent, transmit, and reduce knowledge. This may result in explicit knowledge.

The repository is a general tool for knowledge management; it can be a part of other tools such as a CASE (Computer Aided System Engineering) tool, groupware, expert systems, and decision support systems. It includes information about data, processes, physical hardware, human, and physical resources (ISO, 1990). Initial repositories have been applied to document and program source management in the development of an information system. Recently, they support wider business areas: group communication, strategic decision making, and process improvement (Visaggio, 1994; Berlin et al., 1993). The repository is one of the best tools for knowledge management. Related to knowledge management are system functions such as representation and delivery. These representation and delivery capabilities are essential for systematic knowledge management. Representation techniques, such as model, graph, and formula, are adopted for knowledge abstraction. Good representation allows users to understand and communicate with each other more effectively. At the same time, good delivery systems guarantee easy and timely knowledge acquisition.

In representing and delivering knowledge, an important problem is the variety and change of knowledge representation. For instance, most repositories support popular
models, such as the Entity Relationship Diagram (ERD) and the Data Flow Diagram (DFD). However, with the enlargement of scope and dramatic increase of technical progress, users tend to use new approaches, such as object-oriented models, business process models and decision support models. In these cases, the metaschema of a repository has to be changed, but many current commercial repositories are difficult to extend or change the metaschema. This difficulty may pose a problem in the successful use of a repository for knowledge management. Furthermore, repository users may use different models in different times. The emphasis of our approach is on how to manage various models and their results, which are produced by different users in different times.

Knowledge reuse and its re-creation are other major issues for knowledge management. In this reusing and re-creating processes, the diversity of representation can be a problem for delivery of knowledge. Users may want to reuse knowledge, which can require methods in which they are not experienced. For example, an analyzer of business processes may want to reuse knowledge about an organization, with its management criteria and information systems as well. Typically, those are represented by different methods, but sometimes users want to compare, transfer, and merge knowledge represented by different models. Most commercial repositories are limited in supporting these requirements. For this support, repositories or other tools for knowledge management need various functions and an integrated framework.

This paper proposes an architecture and implements a system for a knowledge repository called Enterprise Model Repository (EMR). The emphasis is on the schematic model, a representation method for knowledge. EMR attempts to solve two issues: model independence and integration. EMR is likely to alleviate the limitations of representation and delivery of knowledge, and thus can support knowledge management more effectively.

The rest of this paper is organized as follows. First, conventional repositories are reviewed and their future directions are explored. Second, knowledge management systems including EMR are compared. Third, the architecture of EMR is introduced and its conceptual metaschema is built. Finally, a prototype is developed to illustrate the usefulness of the EMR.

A Repository for Knowledge Management

Data Dictionary, Encyclopedia, and Repository

It is generally accepted that a repository evolves from a data dictionary or encyclopedia (Tannenbaum, 1994; Noushin & Kuilboer, 1995). Early data dictionaries and encyclopedia were used for the management of information systems. Until recently, researchers have focused on various advanced features of a repository to enhance more effective use of enterprise information resources. Even though it is difficult to categorize them clearly, a data dictionary, encyclopedia, and repository have different features in these developing states. Table 1 summarizes the characteristics of a repository, comparing two different phases of it.

A data dictionary, an encyclopedia, and a repository have different management focuses. While a data dictionary is developed for metadata management of a database, an encyclopedia is used for document management in the development of an information system. However, the management focus of a repository is to provide enterprise information resources based on the corporate strategy (Moriarty, 1990). This coverage of a repository is more comprehensive than a data dictionary or an encyclopedia, which supports a physical database and information system development. The coverage of a repository includes various aspects of business and system environment, and this breadth is relevant to the scope of organizational knowledge.

In the representation method, initial data dictionaries support only a text, but encyclopedias use a few typical models in information system development, as well as text. In contrast, a repository contains more diversified representation methods, such as various models, documents, and hypertexts. In order to use a specific representation method, relevant metaschema are required. A more diversified representation requires more complicated and changeable metaschema. Thus a repository requires advanced features of metaschema.

### Table 1: Repository Development Trend

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Method</th>
<th>Data Dictionary</th>
<th>Encyclopedia</th>
<th>Repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major Management Focus</td>
<td>Metadata</td>
<td>Documentation</td>
<td>Information Resource</td>
<td>Enterprise</td>
</tr>
<tr>
<td>Coverage</td>
<td>Physical Database</td>
<td>Information System Development</td>
<td>All Aspects of Business and System Environment</td>
<td></td>
</tr>
<tr>
<td>Representation Method</td>
<td>Text</td>
<td>Text and Typical Methods such as ERD</td>
<td>Various Models, Documents and Hypertext</td>
<td></td>
</tr>
<tr>
<td>Relationship with IRDS, and CDIF, etc.</td>
<td>No Relationship</td>
<td>Loose Relationship</td>
<td>Close Relationship</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>Not Specified</td>
<td>Three Level or Four Level</td>
<td>Four Level</td>
<td></td>
</tr>
<tr>
<td>Relationship with Tools</td>
<td>No Relationship</td>
<td>Dependent on Tools</td>
<td>Independent on Tools</td>
<td></td>
</tr>
<tr>
<td>Extensibility of Metaschema</td>
<td>Easy to Extend</td>
<td>Hard to Extend</td>
<td>Easy to Extend</td>
<td></td>
</tr>
</tbody>
</table>

Copyright ©2000, Idea Group Publishing.
In an effort to get a common solution, major vendors and user groups have developed repository standards such as the Information Resource Dictionary System (IRDS), the Case Data Interchange Format (CDIF), the Portable Common Tool Environment (PCTE), and the A Tool Integration Standard (ATIS). They provide a common data structure, basic functions, and interfaces between different tools as guidelines for developing a repository (Toyoma & Yamamoto, 1992). These repository standards are still being changed, and have not yet matured (Simon, 1995; Bordoloi et al., 1998). However, major repository standards share issues for the progress of a repository; a repository is more closely linked with these standards than it is with a data dictionary and encyclopedia.

IRDS and CDIF provide four-level data architectures as common data structures of a repository. The standard of IRDS is classified into an IRD Schema Definition, IRD Schema, IRD, and Application Database (ANSI, 1991). In CDIF, they are called meta-meta model, metamodel, model, and data (EIA, 1991). From the data to metamodel, each layer is the instance of the upper layer. Meta-meta model describes the structure of metamodels and metamodel does that of models. For example, if an instance such as ‘James’ or ‘Paul’ is the data, ‘name’ can be a model. The corresponding metamodel can be an ‘entity.’ The meta-meta model can be a ‘data model subject area.’ This hierarchy gives a consistent view of the data structure for a repository. While many encyclopedias are developed according to three levels of architecture, repositories have to be based on a four-level architecture.

Tool independence is another major issue in developing a repository. With this capability users can choose a repository without consideration of such tools as CASE tool and Decision Support System (DSS). Repository standards such as IRDS or CDIF support import and export functions with standardized syntax representing specific metaschema. Tool independence is a target in developing a repository.

Extensibility is a major issue for developing a repository, which requires the ability to accommodate the changing needs of an organization (Bordoloi, Sirac, and Lakhanpal, 1998). This adjustment area includes new tools, models and model components. There are, however, different levels of ease in modification, depending on the original architecture of a repository. Convenient modification can be assessed by whether or not users can extend the metaschema of their repository. Because an encyclopedia is based on specific CASE tools, it is difficult for users to extend this metaschema. Relatively, researchers seek an easy way by which users can modify metaschema of their repository. Extensibility is closely related with architecture, representation, and tool independence, and this connection is a major topic for this research.

A Knowledge Repository

Repositories have progressed in their architectural and functional aspects, in accordance with technical and managerial requirements. A knowledge repository is a type of repository for knowledge management. It needs the characteristics of repository that are described in Table 1, together with some different features. The required characteristics of a knowledge repository may be seen from the objectives of knowledge management within the organization. Organizations may have three major objectives in knowledge management.

First, they want to gather and store people’s knowledge, and to share with those who do not have that knowledge. Organizational knowledge can disappear for various reasons. People can quit their jobs or have an unfortunate accident. Thus many organizations try to accumulate knowledge in a knowledge base, which must support functions for access and storage. Supporting this need, a knowledge repository has to be linked with various working situations, and to be based on an effective metaschema for various representational methods.

Second, an important objective of organization is knowledge reuse. An organization may use a knowledge repository for facilitating knowledge reuse. Even though knowledge is successfully acquired and stored, reuse of it is not a simple task. The increase of amounts and types of knowledge can make the reusing process complicated. One role of a knowledge repository is reducing this difficulty. Searching or scanning functions can mitigate reusing efforts. For effective functions in searching or scanning, a metaschema of the repository is designed according to a relevant logical framework.

Third, in problem-solving processes, organizations want to re-create knowledge according to their new requirements. Typically, people may try to re-create knowledge by starting from their current knowhow. Another role of a knowledge repository is supporting knowledge re-creation. Support for this analytical and integrative activity may involve major functions.

To support these three objectives of knowledge management effectively, it is necessary to choose appropriate representation and delivery methods. Until recently, explicit knowledge means documentation in most practical knowledge management systems (Davenport & Prusak, 1998). However, simple documentation is not sufficient for supporting either easy reuse or the re-creation of knowledge. Users need more refined and structured representation methods in various working environments (Dutta, 1997). The knowledge repository can give more structured way to capture knowledge that is complicatedly mingled and reuse and re-create explicit knowledge.

A Comparison of Knowledge Management Systems

Although there are many alternatives for supporting knowledge management, we concentrate ourselves on three knowledge management systems: groupware, Internet-based systems, and repository. Although repository, groupware and Internet-based systems have different usage goals, they can
be applied to support knowledge management. They have the common system components: representation schema, interface, and various functions (representatively search function) for knowledge management.

Lotus Notes is the most popular groupware, which supports document databases, group communication, and remote database connection. Internet-based systems excel at publishing information across various hardware platforms and transferring multi-media data based on hypertext links (Davenport and Prusak, 1998). However, these systems focus on text and documents as core representation methods, and support a simple search engine. For more flexible representation, a model repository can be an alternative.

A Model Mart is a model repository linked with ER-Win, which is a popular commercial CASE tool of Logic Works Ltd. It supports storage, communication, and reuse of a model. ARIS repository is another model repository linked with ARIS tool of IDS Ltd. ARIS tool is used for analysis for business processes, data flow, and organization. It consists of four different models. Table 2 shows a comparison between EMR and the other four systems (Logic Works, 1997; IDS, 1996).

Systems are developed in the client-server and stand-alone, except Model Mart. Client-server environment guarantees connection and easy communication between multiple sites and users. In addition, supporting stand-alone environment is necessary for private knowledge management. For effective use of the system, users have to be involved in developing Internet-based systems. That is why most Web browsers do not support document management or a search engine. However, four other systems are developed externally, i.e., users are involved indirectly. Representation is important, because functions for knowledge management partly depend on it. In representation, Lotus Notes and Internet-based systems support texts and documents. Especially, Internet-based systems are based on hypertext, which has benefits in flexible link with other hypertext or documents. Three other model repositories use text and model as core representation methods. For transferring information between other tools, specific interfaces are required. Lotus Notes supports interfaces with Internet and other major Lotus family software. Internet-based systems communicate data with various office automation software, such as Microsoft Office. Model Mart and ARIS Repository are directly linked with specific tools, but the EMR is not developed for specific tools and specified for interface. Interfacing with other tools requires bridge algorithms. Finding these algorithms is thus a future task of EMR.

In extensibility of representation, the EMR is significantly different from other systems. Even though the scope is different, four systems can represent a more limited scope. However, the EMR supports many kinds of models according to its model independent architecture. Thus EMR can include more diversified models and descriptions than can the other four systems. Reusability is deeply related to the searching capability. Lotus Notes and Internet-based systems support a search by domain, categorizing by a representative keyword. Model Mart and ARIS repository support simple character search as well as domain. As a searching method, EMR supports integrated search by domain, similarity, and reference. Users can also choose a searching unit flexibly, by determining a specific function for extension. These integrated methods allow users to search more accurately and efficiently. For creating new knowledge, analysis for previous knowledge is important. Generally, Lotus Notes and Internet-based systems support only reuse of previous knowl-

<table>
<thead>
<tr>
<th>Table 2: Comparison of Knowledge Management Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Viewpoint</strong></td>
</tr>
<tr>
<td>System Environment</td>
</tr>
<tr>
<td>User Involvement in System Development</td>
</tr>
<tr>
<td>Core Representation</td>
</tr>
<tr>
<td>Interface with Other Tools</td>
</tr>
<tr>
<td>Extensibility in Representation</td>
</tr>
<tr>
<td>Search Method</td>
</tr>
<tr>
<td>Analysis for Knowledge</td>
</tr>
</tbody>
</table>

Journal of Database Management Jan-Mar 2000 19
edge. However, model repositories assist some kinds of analysis, such as comparison and transfer. Comparison means viewing different aspects simultaneously; transfer is transforming a model into another. Model Mart supports transfer only and ARIS repository supports comparison and transfer. However, EMR supports reference and extension as well as comparison and transfer. According to this conceptual comparison, EMR has a few advanced functions in representation, search and integration, which are critical elements for knowledge reusability and re-creation.

Enterprise Model Repository

Architecture

A model is a structured representation method for physical objects, concepts, and systems that help organize, clarify, and unify knowledge (Jorgenson, 1995). By the use of the model, users can represent business processes, managerial rules, data structures, and logical descriptions about objectives and their state of behaviors. In addition, a model gives benefits in facilitating analysis and communication. Schematic models especially have been a core representation method of the business environment and information systems. Thus a model repository can be an alternative for a knowledge repository in effective knowledge management.

The Enterprise Model Repository (EMR) is a type of knowledge repository based on various schematic models. A schematic model is a diagram that depicts conceptual features. The DFD and ERD are good examples. EMR focuses on two major systematic features for supporting knowledge management effectively. First, EMR is based on an independent architecture, in which users can register for a model with model components without changing the database structure. In traditional repositories, it is difficult, if not impossible, to change data structure. Within an independent architecture, users can freely use their own model types without having to take into consideration the metamodel structure. Second, in order to support effective reuse and re-creation, EMR is based on an integrated framework, which guarantees that users can easily search, compare, refer, and merge knowledge represented by different models.

The architecture of EMR is classified into interface, function, and holding area. Figure 1 shows the basic components of EMR.

Interfaces consist of input/output screen forms for direct access and standardized transferring algorithms between EMR and other tools, including CASE tools, other repositories, groupware, and a decision support system. For interfacing other tools, a repository needs specific programs based on various standards. Some CASE tools and repository standards provide their own interfacing file formats. For other tools, specific interface programs must be developed. Users can access EMR and use functions in heterogeneous environments through these interfaces.

Functions of EMR are categorized into three types: model management, model instance management, and supporting functions. Functions for model management and model instance management consist of create, read, update, and delete (CRUD). With model management functions, users can create, read, update, and delete models. Functions for model instance management enable users to create, read, update, and delete knowledge represented by models. Supporting functions have various functions for knowledge reuse and re-creation. These functions consist of scanning, searching, reporting, and transferring, etc. The basic components are metaschema model, model instance, metadata and their relationships.

A typical use of EMR is as follows. Before creation of the model, the repository manager considers the requirements and evaluates alternative models. After selection of a model, metamodel information is registered into an EMR through interfaces. According to the stored metamodel, instances can be registered and reused.
Occasionally, various supporting functions are required for reuse of models and model instances. For example, for those who are not familiar with specific models, scanning or searching functions will be helpful. At the same time, the searching function is indispensable for the reuse of model instances. After a search, the output is represented by a diagram or report, and sometimes transferred into other models. Occasionally, users find problems with the models. Repository managers have to gather information and add or change model components. The business and technical environments can be a cause for model change. In these cases, they can update, change, or even remove the model.

**Model Independence**

The architecture of EMR is based on model independence. Model independence means that users can add or update models without changing the data structure for the model. The complexity and dynamics of a business and technical environment frequently require addition or change. If users have to change the metaschema whenever a model is updated, it is difficult to accommodate their repository. Model independence is an important feature for an effective knowledge repository.

Simplifying the data structure of a repository from the viewpoint of model registration and change can enable architecture to be abstracted, as shown in Figure 2. Figure 2 also shows the differences in dependent and independent model data structures in a repository.

Three types of data structure include hierarchy, layers of model type, and model component types. In dependent model data structure, the model type is given and users can not add or change it. The model can be changed by vendors only, and thus users may not create their own metaschema for their representation.

A few repository standards and systems have been suggested that have common structures and guidelines in changing the data structure of repositories. According to a four-layer architecture, model components can be unified or generalized for various purposes; this fact can be used for the extension of metaschema, as in an IRD schema or meta model in IRDS and CDIF (ANSI, 1990; EIA, 1991). The semi-dependent data structure guarantees in a general way, a means to add and update new models with model components. However, in these cases users have to reconstruct the data structure for the layer for model instance. Even though the model type and model components can be registered according to the first and second layer, the metaschema for the third layer should be rebuilt.

An independent model data structure need not rebuild its metaschema even though new models and model components are registered. As shown in Figure 2, a model component is classified into model component type, and linked with model type and model instance. Through this metaschema, every model component belongs to a pre-defined category. Thus, users need not rebuild a metaschema, because any model component can be registered according to the category.

The core issue in this approach is how to classify every model component. An idea for classifying model components determines how model components appear, i.e., their model form. Figure 3 illustrates a general classification of model components.

Every schematic model component can be classified into the Separate Unit (SU), External Link (EL), Internal Link (IL), Separate Unit Descriptor (SUD), Link Descriptor (LD), and Composite Descriptor (CD). Table 3 shows a description for these concepts and examples of these classification. SU means the separated form in a diagram. A rectangle, ellipse, or circle is a typical form. EL is the explicit linking form between SUs, and a line or arrow is a typical form. For

![Figure 2: Comparison of Model Dependent and Independent Data Structure](image-url)

![Figure 3: A Graphical Notation of Model Components](image-url)
example, class and attribute of Object Model, state of Dynamic Model, and data store of Functional Model belong to SU (Rumbaugh et al., 1991). Inheritance, aggregation, and association of Object Model and data flow of Functional Model can be typical examples of EL. IL is a type of link between Separate Units that are not visible. The relationship between classes and attributes of Object Model can be an example of IL. A descriptor is a model component addressing the Separate Unit or Link. There are three types of descriptors: SUD, LD and CD. Concepts such as types of association, actions and activity, business rule, and lead time of business process are examples of a Descriptor. Specifically, CD exists only in the combination of Separate Unit and Link. Multiplicity and ordering of Object Model is an example of CD. If a model can be classified into the framework, as shown above, it can support model integration by the similarity of model components between different models. Similar model components mean that two model components represent similar objects. For example, an entity of ERD is similar to a data store of DFD. By the use of similar model components, two models can communicate with each other.

Second, EMR uses domain as a framework for model integration. Domain means a semantically common area for models and instances. Any models and model components can be included in a specific domain. For example, ERD and Class Diagram can be grouped as a data model domain. At the same time, model instances such as customer, sales, and inventory can belong to a sales domain.

Third, EMR supports model integration by reference relationship. The initial author of a model can link an instance to other instances for reference. After that linkage, others are able to refer to the instances when they reuse the knowledge. For example, a process analyst may want to link model instances of an organization chart and an information model when he designs a new business process. When it is difficult to find the similarity or make a domain, the analyst can use a reference relationship. On the basis of this relationship, users can understand organizational and informational aspects related to the business process. As an integrated framework, those three approaches support the reuse or re-creation of knowledge.

Figure 4 depicts the process of reuse and re-creation of knowledge using EMR, a process based on the integrated framework of model integration. When users want to reuse knowledge, they can search and reuse it in a more refined way under the integrated framework. As shown in Figure 4, a user can choose the model type, key word and instance domain. According to options, the search engine of EMR finds related

<table>
<thead>
<tr>
<th>Model Component</th>
<th>Concept</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separate Unit (SU)</td>
<td>The Independent and Separated Form in a Model</td>
<td>Object Model: • Class, • Attribute, Dynamic Model: • State, Functional Model: • Process, • Data Store</td>
</tr>
<tr>
<td>External Link (EL)</td>
<td>The Explicit Linking Form between SUs in a Model</td>
<td>Object Model: • Inheritance, • Aggregation, • Association, Dynamic Model: • Event, Functional Model: • Data Flow, • Control</td>
</tr>
<tr>
<td>Internal Link (IL)</td>
<td>The Intrinsic Linking Form between SUs in a Model</td>
<td>Object Model: • Class / Attribute, Dynamic Model: • Nesting, Functional Model: • Not Exist</td>
</tr>
<tr>
<td>Separate Unit Descriptor (SUD)</td>
<td>Any Components Describing SU</td>
<td>Not Exist, Dynamic Model: • Actions &amp; Activity, Functional Model: • Not Exist</td>
</tr>
<tr>
<td>Link Descriptor (LD)</td>
<td>Any Components Describing EL and IL</td>
<td>Object Model: • Type of Association, Dynamic Model: • Not Exist, Functional Model: • Not Exist</td>
</tr>
<tr>
<td>Composite Descriptor (CD)</td>
<td>Any Components Describing SU and EL or SU and IL at the Same Time</td>
<td>Object Model: • Multiplicity, • Ordering, Dynamic Model: • Not Exist, Functional Model: • Type of Data Access</td>
</tr>
</tbody>
</table>

Model Integration

In the reuse and re-creation of knowledge, users may want to find knowledge represented by various models. Sometimes the differences in models can be an obstacle for reuse. For permitting a variety of models in the repository, an integrated framework is essential. In most repositories, model integration means having functions for transferring data between them. However, model-integration includes functions for search, comparison and reference, as well as transference between different models, in order to facilitate reusability and re-creation.

EMR supports model integration in three ways. First, it
instances. For example, in Figure 4, EMR gradually searches any model components similar to “entity.” Concurrently, EMR scans and lists related model instances in “sales domain.” By the cooperative use of these two approaches, users can find related instances represented by different models, such as DFD and Class Diagram, which users may not know before searching. Users can learn the models through their metadata.

After this searching process, according to model similarity and domain, users can choose two additional search options. When users want a reference model for instance in an output, they can search for instances showing a reference relationship with output. With the references users know the backup knowledge. For example, they can refer to the ordering process represented by Business Process Model (BPM) as shown in Figure 4. EMR also supports search extensions. Any searched model instances can be extended on the basis of the general form of the model, as described in model independence. In the EMR structure, most Separate Units are linked with Link Instances and various Descriptors, and these are also linked to other items. EMR has this information, and follows these internal relationships. Users can extend output easily and logically.

**Metaschema of EMR**

According to the architecture of EMR, the conceptual metaschema for its database was designed so as to implement model independence and integration. An ERD notation, as
Figure 5: Metaschema for Model Independence

![Metaschema for Model Independence]

- If instance of A exist, then corresponding instance of B must exist.
- If instance of B exist, then corresponding instance of A may exist.
- 1:m Cardinality
- m:m Cardinality

Figure 6

![Metaschema for Model Integration]
shown in Figure 5, represents the conceptual metaschema for its model independence. The metaschema consists of model type, model component type, and instance according to general classification of model components. The metaschema integrates three layers of data architecture of the repository. Generally, these three layers are represented by two separated metaschema such as meta-meta and meta layer, and meta and instance layer. These two schemas are used for different purposes. However, for model independence, two schemas can be integrated. Components of every layer, except Model Type, are existentially dependent on the upper layers. For example, without model type no model components exist. A model type has one or more model components; a model component has one or more Separate Unit instances, Link instances, and/or various Descriptors. A Separate Unit instance can be related to many Link instances, and vice versa. A Separate Unit instance and a Link instance have many Descriptors.

The other characteristic of EMR architecture is model integration. Figure 6 shows aspects of a metaschema of EMR supporting model integration. The concept of model similarity is represented in Similar Model Component. A model component can have many similar components. In another way of model integration, the domain concept is divided into two entities: model domain and instance domain. A model type can be related to many model domains, and a model domain can have many model types. Instances can also be related to instance domains. For reference of model instance, a reference model component is represented in the metaschema.

The metaschema of most commercial repositories is more complicated than EMR. The difference in complexity results from model independence. EMR integrates metaschema layer with meta layer by categorizing model components into six types. However, other commercial repositories need to represent various model components as they are.

**Figure 8: Register Form of Model and Model Component**

### A Case and Prototype

K Company is one of the three major companies in the confectionery industry of Korea. Recently, in the increasing competition and economic recession, K Company has redesigned core business processes and developed new information systems. Members in the projects thought that effective knowledge management is an important factor for the successful process improvement and system development. On the advice of consultants, they considered model repository a system for knowledge management. However, because they had not experienced model repository, they wanted to test a type of model repository.

Call Desk Management is a major internal process of K Company. This process starts as a request from internal users for purchase or technical service, and progresses in the internal division or major outsourcing companies. Evaluations from the past year made clear that users were not satisfied with the efficiency and quality of the Call Desk. Thus, K Company chose Call Desk management as a core business process. The Process Innovation (PI) team observed the problems of the current process, with its supporting systems, and needed to find a new way. At the suggestion of a consultant, the PI team decided to use a prototype of EMR to test the usefulness of model repository. A prototype of EMR was developed by the use of Microsoft Visual Basic 5.0 and SQL Server 6.5.

In order to represent internal processes and information systems, K Company used three kinds of models: an Entity Relationship Model, a Data Flow Model, and a Business Process Model. However, the PI team found that a new type of model was required to evaluate the current business process. For improving the Call Desk process, they had to evaluate it by various resources and criteria, such as time, cost, manpower, facility, and user satisfaction. Thus, process analysts developed a model for capturing evaluation data. This model consists of components such as business process, sequence for process description, and time and cost factors for process evaluation.

### Implementation of Model Independence

After selection of the model for a process evaluation, the PI team registered it into EMR. As we described, EMR help users add and update a model without changing its structure. Model and Model components were registered in the form shown in Figure 8, which shows that users can register a new model type and its model components, according to their requirements.

A user registers a new model type “PEM” and its component “Manpower”
and the notation. He categorizes it into a Separate Unit Descriptor. By this activity, the PI team can use this model in the project. Users can model real instances in three ways: direct input, drawing, and importing. Direct input means users register each instance, one by one in a registering form. This is a time-consuming task and sometimes it is necessary to supplement it with other methods. Users can register model instances by drawing or importing. Figure 9 shows an example of drawing a PEM, according to notations registered in the model component management as shown in Figure 8. A team member drew “Help Desk Process.” He drew three processes and their sequence and evaluated data: manpower and time. The PI team could register the specific process and measure it by specific criteria.

Implementation of Model Integration

The PI team members wanted to observe and analyze various perspectives: data, process, and data flow. They use the integrated framework of EMR. The PI team manager registered similar and referring model components. Figure 9 shows an example.

Support for Knowledge Reuse and Re-creation

PI team members reused model instances in the analysis and design. EMR could assist knowledge reuse and recreation under the model independence and integration. Figures 11 and 12 show an example. A database analyst searched model instances related to “purchase.” As shown in Figure 11, users key in options that they want to reuse. A PI team member selected “ERD” as model type and keyed in “purchase” as a character search and “Call Desk” as domain. EMR searched related instances. However, the outcome included items of diversified models: DFD and PEM as well as EMR. This result can be possible for the integrated framework. The entity of ERD and data store of DFD were registered as similar components, and the business process as a referential component. This output could be extended according to extension levels. Diagrams, as shown in Figure 12 depicting four different models, can also represent an output from a single search.

Benefits of EMR

After three months, the PI team completed the redesign of the Call Desk process, including personnel, business rules, and information systems. In these PI activities, EMR used a core system for information gathering, representation, evaluation, and finding alternatives. Most PI activity results were stored as new reusable knowledge after evaluation. This knowledge will be reused in
EMR supports advanced features for knowledge reuse. Model components and instances can be categorized (according to the domains defined by users) and searched in a convenient fashion. Furthermore, EMR helps refine the search results via various search functions.

- Integration of knowledge

Knowledge Integration is an impressive feature of EMR. Models and model components can be integrated via a common metaschema. Users, who are not aware of their relationships, can have an integrative view for a variety of different views by different users.

Conclusion

Many business organizations have emphasized the reuse of knowledge. Users require a much wide-scope of information than in the past. At the same time, required quality, including correctness and fitness for use, has also been increased. With this scope and quality, it is very important to represent specific knowledge. Users want to search and access any reusable information resources, and to represent them in their accustomed representation ways.

In this paper, we have shown how to solve representation and delivery problems, focusing on a schematic model through EMR. EMR is based on two major architectural properties: model independence and integration. By these architectural properties, EMR can be sustained in a changing and complex representation environment, and also facilitate reuse and re-creation between different models. A case and prototype of EMR showed that users have real benefits in representation, reuse and communication. We believe that EMR can be applied to process innovation, strategic analysis, and information system analysis and design as an enterprise system for knowledge management. However, EMR may be improved to fulfill its enterprise knowledge management more effectively as follows.

First, the scope of representation should be enlarged. There are different representation methods such as graphs, mathematical formulas, and various report forms in a business organization. These types of representation have spe-
specific structures with supporting types. The metaschema of EMR will be changed accordingly. Sustaining independence and integrated property will be a challenge.

Second, integration between high level analytical information, such as strategy and business process models with application programs, is an issue in supporting reusability in a full-life cycle of business with system development. By this integration, a company can respond to new requirements faster. We expect that an object-oriented approach gives a good solution in the near future.

Third, more generalized import and export functions, with application programming interfaces, are required for various decision support tools.

Acknowledgment

This work was supported in part by a grant from KOSEF (98-0102-0801-3).

References


