# Integrating Information Semantics for Concurrent Engineering

Michael N. Huhns
Microelectronics and Computer Technology Corporation
Information Systems Division
3500 West Balcones Center Drive
Austin, TX, U.S.A. 78759-6509
(512) 338-3651 or huhns@mcc.com

**Abstract**

The goal of our research is to develop methods for integrating separately developed information resources to enable them to be accessed and modified coherently. We are providing the computational infrastructure, from network communications to consistency maintenance, for cooperative concurrent engineering. In this paper, we describe the efforts of the Carnot Project at each of these levels, focusing on our techniques for achieving integration and consistency at the semantic level. The techniques function at both compile time and run time. At compile time, a static global ontology, provided by the Cyc knowledge base, is used to develop semantic mappings among resources. The mappings are based on schema-level information provided for each resource. At run time, a distributed truth maintenance system is used to maintain semantic consistency of data. The truth maintenance system is executed by knowledge-based agents—one for each user, resource, and application—that actively monitor integrity constraints. In addition, the agents help locate and provide access to the most appropriate information for each user or application.

## 1 Introduction

World-wide production of manufactured goods is currently being affected by five related factors: 1) there are pressures for a shorter time-to-market, forcing a need for all aspects of product engineering—from conceptualization through delivery and maintenance—to be considered simultaneously, 2) there are changes in the artifacts of production, in that many products that used to be standardized are being specially designed for each customer, and more complicated products are being attempted, 3) there are increasing data, knowledge, and experience being accumulated about all aspects of product engineering, which can be used to aid future product development, 4) there are now a plethora of tools for aiding product engineering, including tools for simulation, visualization, layout, test, aesthetics, compliance with standards, and manufacturability, and 5) the scope of the problem has increased to the point that teams of engineers are typically required. The overall trend in each of the five factors has been towards increasing the complexity of the engineering

task. This in turn has placed additional demands on the computational aids for engineering, with the foremost demands being for interoperability and coherent access to all the relevant information available.

The purpose of the Carnot Project is to provide for the integration of a variety of data and processing resources, first within an enterprise, and then, across enterprises. The Carnot Project is developing the infrastructure for the concept of an enterprise-wide information space that allows easy access for application developers and engineers to the variety of different kinds of data and information that exist within large enterprises, multinational corporations, regional companies, etc. The project also addresses the problems of trying to coordinate interactions among companies. See Figure 1. This is the context for what is seen as a global information system, spanning businesses and various other kinds of providers of services.

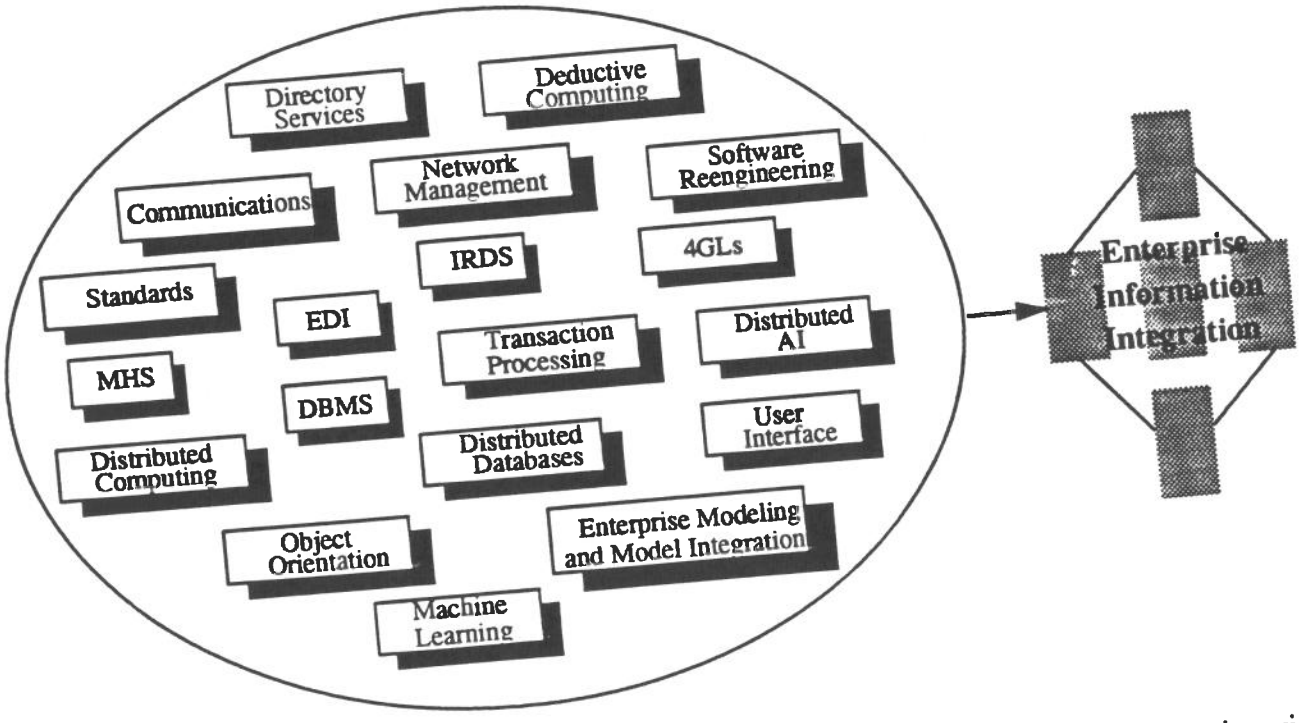


Figure 1: Technologies being used and integrated by Carnot to provide for enterprise-wide information spaces

## 1.1 Aspects of Enterprise-Wide Information Spaces

### 1.1.1 Heterogeneity

Today's corporate computing environments are heterogeneous, containing many independent information resources of different types, such as a database management system with its databases, an expert system with its knowledge base, an information repository, or an application program. Unfortunately, the resources are often mutually incompatible in syntax

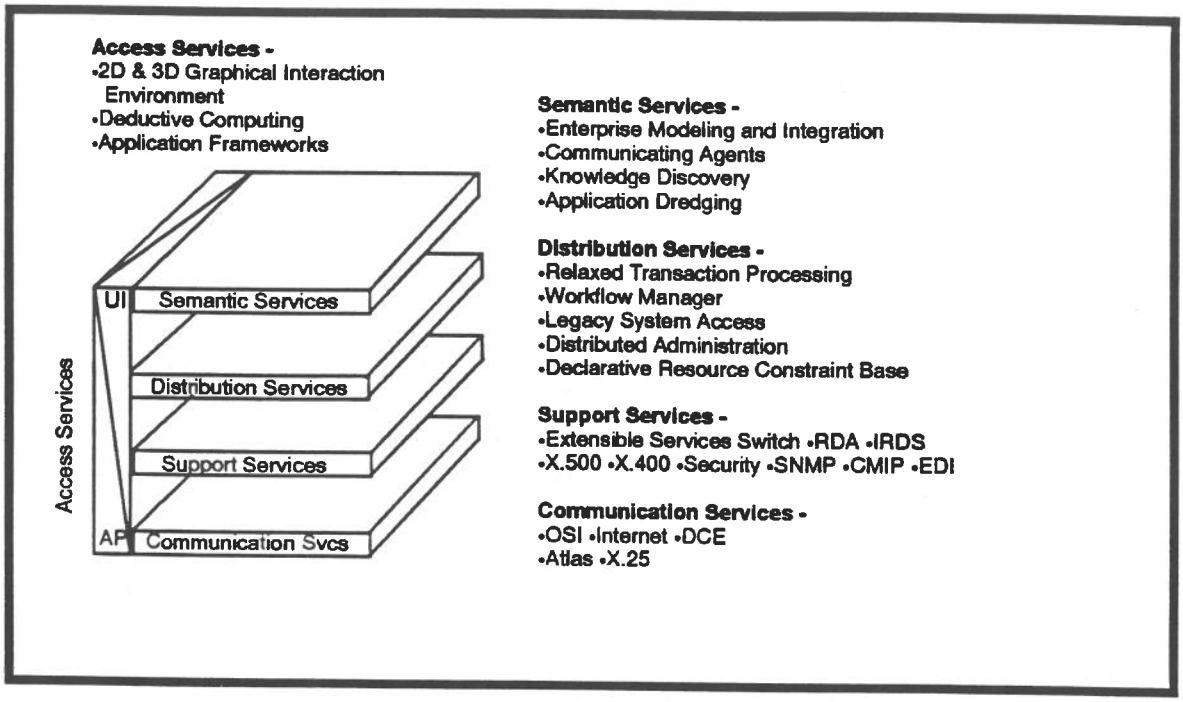


Figure 2: Layered architecture of the Carnot system

X.500 directory services. Work is progressing on additional support services, such as Information Resource Dictionary System (IRDS) interfaces, EDI, security via the ISO Upper Layers Security Model, and Network Management via SNMP, CMIS, and CMIP.

### 1.2.3 Distribution Services

The distribution services provide the basic facilities needed to coordinate the execution of work across a variety of heterogeneous resources, including databases, files, and other applications used in running a business. It is with these services that the Carnot system begins to add unique value to the standards efforts.

The main element in the distribution services is a distributed shell environment called the Extensible Services Switch (ESS), which coordinates the control flow and information flow among components of the various Carnot services. ESS is based on the MIT actor model. Distributed transaction/query managers build ESS scripts that reflect current business realities and accumulated corporate folklore. These managers build the scripts through interactions with directory services, repository managers, and deductive knowledge in Carnot's declarative resource constraint base. The declarative resource constraint base is essentially a collection of predicates that expresses business rules, interresource dependencies, consistency requirements, and contingency strategies throughout the enterprise. The declarative constraint base may adapt to a changing environment without modifying the application programs that manage the transactions; new applications can easily be added to the environment.

### 1.2.4 Semantic Services

The semantic services provide a global or enterprise-wide view of all the resources integrated within a Carnot system, rather than the view that occurs at the distribution services layer, which essentially treats each resource as accessible but separate. The semantic services have a view of the distributed environment in which a common language can be used for communicating among resources. No attempt is made to make everything appear as a single centralized system, but rather as a common semantic framework.

This framework includes 1) MCC's Cyc knowledge base, whose ontology and reasoning mechanisms are used as a backdrop for representing sets of concepts that may be found in various databases and other resources; 2) MCC's Reasoning Architecture software, which provides mechanisms for independent, communicating agents, which in turn can encapsulate large groups of expertise as separate active entities within the Carnot system; and 3) schema integration and database design tools for updating, reconceptualizing, and generating information schema.

In addition to representing the semantics of the enterprise-wide information space, we incorporate reasoning agents in the environment. These agents are able to make use of the resources that are in the environment and to provide access to those resources for users and applications in a way that keeps the users and applications from having to know the details of navigating through the information space.

### 1.2.5 Access Services

The access services provide mechanisms for manipulating the other four Carnot services. The access services allow developers to use a mix of user interface software and application software to build enterprise-wide systems. Some situations (such as background processing) utilize only application code and have no user interface component. In other situations, there is a mix of user interface and application code. And finally, there are situations in which user interface code provides direct access to functionalities of one or more of the four services.

# 2 Dimensions of Semantic Integration

Enterprise modeling is a corporate activity that produces models of the information resources, information flows, and business operations that occur in an enterprise. In many cases, models of information resources are already available. For example, the information present in a database is modeled by the schema for the database. Similarly, the information present in an object-centered knowledge base is modeled by the ontology of the objects. The goal of our semantic integration research has been to develop a method for integrating such existing models that overcomes the above incompatibilities, yielding coherency and consistency. The method provides logical connectivity among information resources via a semantic service layer that automates the maintenance of data integrity, and provides an approximation of global data integration across systems, thus enabling the resources to be used coherently.

Two approaches have been suggested previously for integrating heterogeneous databases [2]. The *composite approach* introduces a global schema to describe the information in the given databases. Users and applications are presented with the illusion of a single, centralized database. Explicit resolutions are specified in advance for any semantic conflicts among the databases. However, the centralized view may differ from the previous local views and existing applications may not execute correctly any more. Further, a new global schema must be constructed every time a local schema changes or is added.

The *federated* [6] or the *multidatabase* [8] approach presents a user with a collection of local schemas, along with tools for information sharing. The user resolves conflicts in an application-specific manner, and integrates only the required portions of the databases. This approach yields easier maintenance, increased security, and the ability to deal with inconsistencies. However, a user must understand the contents of each database to know what to include in a query: there is no global schema to provide advice about semantics. Also, each database must maintain knowledge about the other databases with which it shares information, e.g., in the form of models of the other databases or partial global schemas [1]. For $n$ databases, as many as $n(n-1)$ partial global schemas may be required, while $n$ mappings would suffice to translate between the databases and a global schema.

We base our methodology on the composite approach, but make four changes that enable us to combine the advantages of both approaches while avoiding some of their shortcomings. First, we use an existing global schema—the Cyc knowledge base [7]. The schemas of individual resources are compared and merged with Cyc but not with each other, making a global schema much easier to construct and maintain. Using Cyc is significant, because of 1) its size: it covers a large portion of the real world and the subject matter of most information resources, 2) its rich set of abstractions, which ease the process of representing predefined groupings of concepts, 3) its knowledge representation and inference mechanisms, which are needed to construct, represent, and maintain a global schema, and 4) its typing mechanism, which is used to integrate and check the consistency of query results.

Second, unlike most previous work on integration, we use not just a structural description of the local schemas, but all available knowledge, including 1) schema knowledge, i.e., the structure of the data, integrity constraints, and allowed operations; 2) resource knowledge, i.e., a description of supported services, such as the data model and languages, lexical definitions of object names, the data itself, comments from resource designers and integrators; and 3) organization knowledge, i.e., the corporate rules governing use of the resource.

Third, we capture the mapping between each individual resource and the global schema in a set of *articulation axioms*: statements of equivalence between components of two theories [5]. The axioms provide a means of translation that enables the maintenance of a global view of all information resources and, at the same time, a set of local views that correspond to each individual resource. An application can retain its current view, but use the information in other resources. Of course, any application can be modified to use the global view directly to access all available information.

Fourth, we consider knowledge-based systems (KBSs) as well as databases. How a (KBS) is integrated depends on which of the following three roles it plays in an information system:

**The KBS as an Information Resource:** This is the weakest case, since no assumptions are made about the applications of which the given KBS is a resource. Thus, in this case, the KBS is integrated in approximately the same way as a database. Even so, this

case is interestingly different because the KBS is not simply an information store—it also contains rules. These rules can aid the process of integration.

**The KBS as a Driver:** The KBS is the main application, and integration of the other resources is done relative to this application. In this case, the KBS maintains knowledge about the other resources and about how to access them. Also, rules in the KBS should fire off of all appropriate databases directly. Thus, the global schema required depends on the KBS; our approach aims to captures much of this dependency.

**The KBS as a Federating Mechanism:** Semantically, the KBS serves as an intermediary between applications and databases. The KBS receives queries and updates and converts them to appropriate queries and updates on the views it integrates. The KBS maintains representations of the models of the databases in its charge. It embodies the specifications of appropriate conflict resolution algorithms, and specifications of how to distribute subqueries among databases and merge the results.

# 3 Semantic Transactions with Global and Local Views

As shown in Figure 3, a resource is integrated by specifying a syntax and a semantics translation between it and the global schema. The syntax translation provides a bidirectional translation between a local data manipulation language, $DML_i$, and the global context language, GCL, which is based on extended first-order logic. The semantics translation is a mapping between two expressions in GCL that have equivalent meanings. This is accomplished by a set of articulation axioms, having the form $ist(G\ \phi) \Leftrightarrow ist(S_i\ \psi)$ where $\phi$ and $\psi$ are logical expressions and *ist* is a predicate that means "is true in the context." This axiom says that the meaning of $\phi$ in the global schema $G$ is the same as that of $\psi$ in the local schema $S_i$. At most $n$ sets of axioms are needed for $n$ resources.

After integration, one can access the resources through the global view, which gives the illusion of a single information resource, but requires that GCL be used. Queries and updates can also be issued against a local view. In that case, they are first translated into GCL and then into different $DML_i$ and distributed to appropriate information resources. Thus applications need not be modified to access the extra information that becomes available.

To illustrate the idea, we describe how transactions are processed semantically through the global and local views of two integrated databases. The two databases have the same domain (hotels), but use different data models. The Fodor database, shown in Figure 4, uses an object-oriented data model. A hotel is represented as a class called `FodorInfo`. The features of hotel are represented as fields of the class or as other object classes pointed at by `FodorInfo`. The AAA database, shown in Figure 5, uses the relational model. A hotel is called `AAAInfo` and represented as a relation. The features of hotel, such as name and address, are represented as columns. Note that these schemas represent different perspectives and different information about hotels.

Some of the Cyc concepts used in integrating these databases are the collections `Lodging` and `Restaurant`, and the predicates `hasAmenities`, `phoneNumber`, and `instanceOf`. Articulation axioms that map between the two database schemas and the global schema include

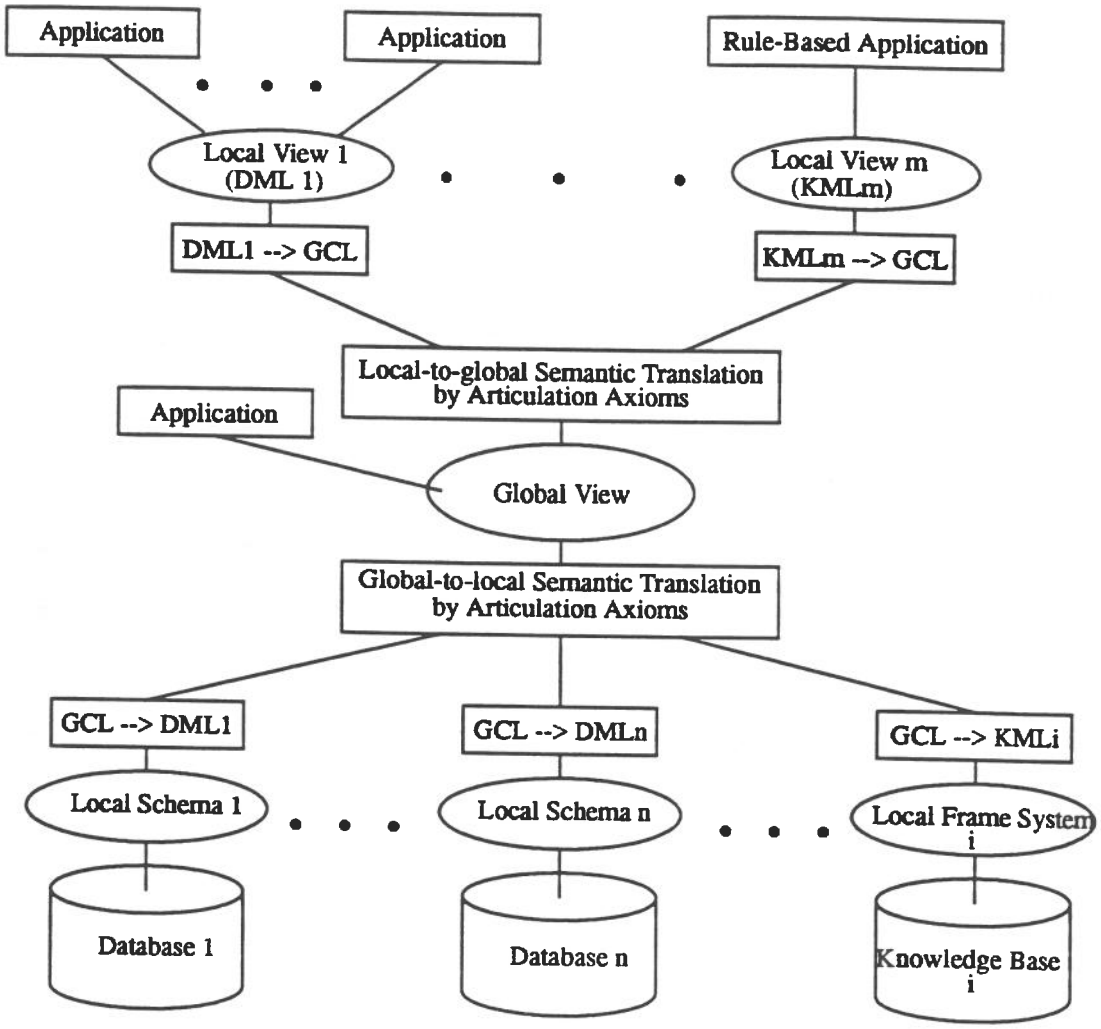


Figure 3: Global and local views in semantic transaction processing

*ist(G instanceOf(?H Lodging))⇔ist(AAA instanceOf(?H AAAInfo))*
*ist(G phoneNumber(?H ?P))⇔ist(AAA phone(?H ?P))*
*ist(G hasAmenities(?H ?F))⇔ist(AAA facility(?H ?F))*
*ist(G instanceOf(?H Lodging))⇔ist(Fodor instanceOf(?H FodorInfo))*
*ist(G hasAmenities(?H ?F))⇔ist(Fodor facility(?H ?X)∧facilityCode(?X ?F))*

Based on its local view of the AAA database, an application might issue the following query for the phone numbers of hotels that have a restaurant:

```
SELECT phone FROM AAAInfo WHERE facility = "Restaurant"
```

This local SQL query is first translated into GCL by the SQL-GCL syntax translator:

```
instanceOf(?L AAAInfo) ∧ instanceOf(?R Restaurant) ∧ facility(?L ?R) ∧ phone(?L ?P)
```

This expression is then mapped by articulation axioms into a new expression whose semantics is meaningful in the global schema:

```
instanceOf(?L Lodging) ∧ instanceOf(?R Restaurant) ∧ hasAmenities(?L ?R) ∧ phoneNumber(?L ?P)
```

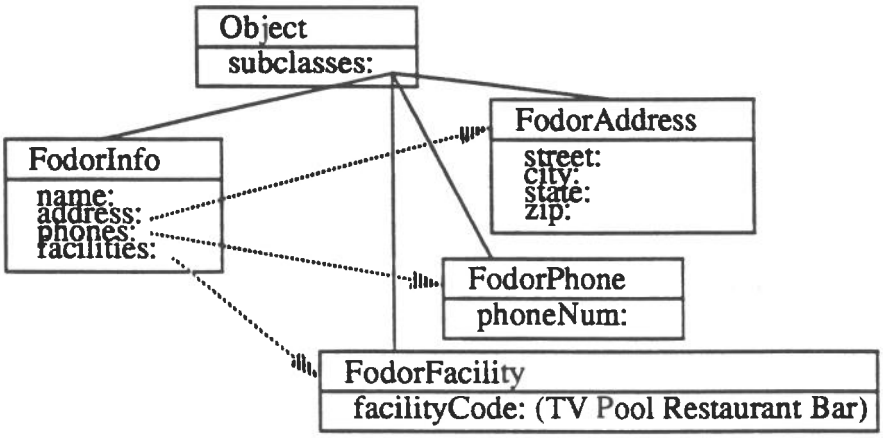


Figure 4: The object-oriented schema for the Fodor database

| RELATIONS | COLUMNS |
|---|---|
| AAAInfo | name* address rateCode lodgingType phone facility |
| AAADirection | address* direction |
| AAACredit | name* creditCard* |
| AAARate | name* season* 1P 2P1B 2P2B XP fCode |

Figure 5: A relational database schema for the AAA Tour Book database

This is then translated into different local queries using the appropriate articulation axioms in reverse. The translation for the Fodor local schema is

```
instanceOf(?L FodorInfo) ∧ facilities(?L ?F) ∧ facilityCode(?F ?R)
    ∧ instanceOf(?R Restaurant) ∧ phone(?L ?P) ∧ phoneNum(?P ?N)
```

These queries are then translated into appropriate $DML_i$ before being sent to the databases. For example, the query sent to the Fodor database is the following object-oriented expression (using ITASCA$^{TM}$ syntax):

```
(SELECT (FodorInfo phones phoneNum)
        (= (path (some facilities) (some facilityCode)) "Restaurant"))
```

After the transactions are executed, the distributor assembles the results in the local view. In this example, the result is a column of phone numbers, because the local view of the AAA database is in the relational model. Note that these phone numbers come from two databases and the list may be much longer than that from the AAA database alone. However, users and applications need not be aware of the extra source of information.

# 4 The Development of Articulation Axioms

The articulation axioms for an information resource are developed in three phases: 1) *schema representation*, in which a Cyc context containing a model for the resource is produced, 2) *concept matching*, in which concepts from the model are matched with concepts in Cyc's base context—the global schema, and 3) *axiom construction*, in which the matches are converted automatically into articulation axioms by instantiating templates for these axioms with terms from the matches. The matching phase requires human interaction: frames may have to be created in the global schema and the model of the local schema may have to be augmented with additional properties (semantics) to ensure a successful match.

## 4.1 Schema Representation

In this phase, we represent the given schema as a set of frames and slots in a Cyc context created specially for it. These frames are instances of frames describing the data model of the schema, e.g., (for a relational schema) `Relation` and `DatabaseAttribute`.

We define three types of frames for representing schemas: 1) `DatabaseSchema` frames, describing the schemas for different data models, 2) `DatabaseComponent` frames, describing the major components of schemas, such as relations and entities, and 3) `DatabaseLink` frames, describing different kinds of links used to refine and relate the major components. Every schema and every one of its components (relation, attribute, etc.) is an `instanceOf` these types and belongs to a context characterizing that schema. The slot `dBSchemaMt`, defined for `DatabaseSchema`, is used to express the relationship between an instance of a schema and its context. Information about the usage of a resource and the functionalities it provides (DDL, DML, transactions) are represented similarly, i.e., using frames such as `RelationalService`, `ERService`, `RelationalDDLType`, and `ERTransactionType`.

## 4.2 Matching

How articulation axioms are generated and how different schemas are integrated depends crucially on the process of matching them. For resource integration, the problem of matching is: given a (Cyc) representation for a concept, find its corresponding concept in the global schema. There are several factors that affect this phase: there may be a mismatch between the local and global schemas in the depth of knowledge representing a concept, and there may be mismatches between the structures used to encode the knowledge. For example, a concept in Cyc can be represented as either a collection or an attribute [7, pp. 339ff].

If the global schema's knowledge is more than or equivalent to that of the local schema's for some concept, then the interactive matching process described in this section will find the relevant portion of the global schema's knowledge. This knowledge will be in one of Cyc's two forms for concept representation. If the global schema has less knowledge than the local schema, then knowledge will be added to the global schema until its knowledge equals or exceeds that in the local schema; otherwise, the global schema would be unable to model the semantics of the resource. The added knowledge refines the global schema.

### 4.2.1 Lexical Heuristics

Finding correspondences between concepts in the local and global schemas is a subgraph-matching problem. We base subgraph matching on a simple string matching between the names or synonyms of frames representing the database schema and the names or synonyms of frames in the global schema. Matching begins by finding associations between attribute/link definitions and existing slots in the global schema. After a few matches have been identified, either by exact string matches or by a user indicating the correct match out of a set of candidate matches, possible matches for the remaining schema concepts are greatly constrained. Conversely, after integrating an entity or object, possible matches for its attributes are greatly constrained.

### 4.2.2 Heuristics Based on Rules

While DBs are passive, KBSs are active. Intuitively, the rules of a KBS capture the operational semantics of the representations in the underlying KB; e.g., a `broker` is a broker because he (or it) participates in certain ways in certain financial transactions. Rules can thus be exploited in different ways in the matching phase.

Using heuristics based on rules for matching limits the set of possible matches significantly and improves the chances of obtaining a match that preserves the important semantic properties of the application. It also reduces the amount of input required from human designers, thereby making it simpler for them to guide the process of integration. Thus the added complexity of dealing with a KBS instead of a DB has immediate pay-backs in terms of the effort required for integration and the quality of the solution. In this abstract we describe the simplest ways in which rules can be used.

**Syntactic type checking:** As an abstract example, consider a rule `If foo(?x ?y) bar(?y ?z) then baz(?x ?y ?z)`. By inspection, we can determine constraints such as the following on potential mappings of the predicates `foo`, `bar`, and `baz`. The type of the first argument of `foo` is compatible with the type of the first argument of `baz`: this is

because if the rule ever fires, `?x` must be bound to something. Similarly, the types of the second argument of `foo` and of the first argument of `bar` must be compatible. This precludes matches in which `foo` and `bar` are matched with entities in Cyc that are incompatible in the appropriate sense.

**Constraints on domains and ranges:** While syntactic type checking is a useful heuristic, it is too cautious and can often be improved. The improvement is based on the intuition that rules are usually written for the entire concepts involved, not for any arbitrary subclasses. That is, the restrictions on the firing of a rule are explicit in its antecedent. This means that, in terms of the preceding example, the type of `foo` must not just be compatible with the type of the first argument of `baz`, it must be a subclass of the latter.

As a concrete example, consider a KBS containing a rule involving `broker` and `client`, where the rule refers to the relation of `isAdvisorOf` holding between an instance of `broker` and an instance of `client`. Thus in the matching process, we consider the binary relation `isAdvisorOf`, and the concepts, `broker` and `client`. The class `broker` must be matched to a subclass of the domain of `isAdvisorOf` and `client` to a subclass of its range. Similarly, if the rule makes a reference to the advice being given to `client` and involves a relation that restricts it to be financial in nature, the match of `isAdvisorOf` would be constrained to be with a subrelation of `isFinancialAdvisorOf`; i.e., the match would be improved. This would improve the other matches too, by constraining `broker` to have financial accreditation and the right to work in a particular market.

**Capturing implicit restrictions:** Reasoning about the rules in a KBS can also yield constraints on potential matches that are not obvious from the (local) schemas to be integrated, but which are nevertheless important in the global schema. These concern features that are implicit in the original local schemas, but need to be considered explicitly in the global schema, because it might contain entities that do not have them.

Consider a KBS with a rule that selects hotels for handicapped persons by looking at the preferences of a given person and the features available in a given hotel. Assume that the KBS implicitly considers only hotels that are wheelchair accessible, because those are the only ones listed in its local KB. Suppose this KBS is to be integrated with a general DB containing entries for hotels in general. For this integration to be correct, the generated articulation axioms must related `hotel` in the local schema to `hotel` $\cap$ `accessible` in the global schema. When `persons` in the original KBS is matched with `handicappedPerson` in the global schema, the relation `acceptableHotel` is constrained to match with the subrelation of `acceptableHotel` in the global schema that applies to the given subclass of `person`. This forces `hotel` to be matched with `accessibleHotel`, as desired. As a result, the correct articulation axiom would be generated such that where the KBS queried its local KB for hotels, it would now query the integrated DB for hotels that are wheelchair accessible. The rule can then fire as before.

### 4.2.3 Concept Matching

Let $a_{ij}$, $j = 1, 2, \ldots, n$ denote the attributes of concept $E_i$ in a local schema. $E_i$ is the domain of the attributes, i.e., the entity, relationship, relation, class, or object for which the $a_{ij}$ are defined. Let $s_j$ be the global schema slot that corresponds to, or matches, $a_{ij}$.

**Observation 1** *The domain $C_j$ of slot $s_j$ is a generalization of the concept in the global schema that matches $E_i$.*

For example, the domain of the attributes `facility` and **phone** is the entity **AAAInfo**, whereas the domains of the corresponding Cyc slots **hasAmenities** and **phoneNumber** are the frames **HumanOccupiedStructure** and **Agent**, respectively. These are generalizations of Cyc's **Lodging**, which is the frame whose semantics most closely corresponds to **AAAInfo**.

As we match each of the attributes of $E_i$, we compute the common subdomain of the domains of their corresponding slots. The resulting common subdomains, although still generalizations of $E_i$, approximate it more and more closely.

**Observation 2** *The "best" match for $E_i$ is $\bigcap_{j=1}^{n} C_j$, the most general common subdomain (greatest lower bound in the generalization hierarchy) of the slot domains.*

In the above example, the most general common subdomain of **HumanOccupiedStructure** and **Agent** is **ServiceOrganization**, a generalization of **Lodging**. This would be suggested as the approximate match for **AAAInfo**. If no other attributes are matched, this would also be the *best* match that could be determined automatically for **AAAInfo**.

The greatest lower bound might not exist as a single frame in the global schema, however; it might be a set of frames. For example, the greatest lower bound would be the set {**HumanOccupiedStructure Agent**} if the frame **ServiceOrganization** did not exist. In such a case, a frame would be created in the Cyc knowledge base with the frames in the set listed as its generalizations.

### 4.3 Constructing Articulation Axioms

An articulation axiom is constructed for each match found. For example, the match between the relational attribute **phone** and the Cyc slot **phoneNumber** yields the axiom

*ist(Cyc phoneNumber(?L ?N))⟺ist(AAA phone(?L ?N))*

which means that the **phone** attribute definition determines the **phoneNumber** slot in the global schema, and vice versa. Articulation axioms are generated automatically by instantiating stored templates with the matches found.

## 5 Discussion

We described an experiment in integrating information models. In our approach, the integration of resource schemas is based on articulation axioms defined between two contexts: the context of a resource schema and a global schema context provided by the Cyc knowledge base. Our methodology is based on the following principles:

- Existing data should not have to migrate or be modified to achieve integration.
- Existing applications should not have to be modified due to integration.
- Users should not have to adopt a new language for communicating with the resultant integrated system, unless they are accessing new types of information.

- Resources should be able to be integrated independently, and the mappings that result should not have to change when additional resources are integrated.

The above principles are incorporated in an integration tool for assisting an administrator in integrating a resource and a transaction tool for providing users and applications with access to the integrated resources. The integration tool uses an extensive set of semantic properties to represent an information resource declaratively within the global schema and to construct bidirectional mappings between the resource and the global schema. The mappings are used by the transaction tool to translate queries and updates written against any local schema into the appropriate form for each information resource. These tools constitute part of the semantic services of Carnot [3], under development at MCC. Carnot will enable development of open applications that can be tightly integrated with information stored on existing, closed systems. The semantic service layer of Carnot provides facilities to specify and maintain the semantics of an organization's integrated information resources.

Thus our approach to resource integration allows a user or application to use a familiar local schema, while still benefiting from newly added resources.

# References


[1] M. Ahlsen and P. Johannesson, "Contracts in Database Federations," in S. M. Deen, ed., *Cooperating Knowledge Based Systems 1990*, Springer-Verlag, London, 1991, pp. 293–310.

[2] O. P. Buneman, S. B. Davidson, and A. Watters, "Querying Independent Databases," *Information Sciences*, Vol. 52, Dec. 1990, pp. 1–34.

[3] P. E. Cannata, "The Irresistible Move towards Interoperable Database Systems," *First International Workshop on Interoperability in Multidatabase Systems*, Kyoto, Japan, April 7-9, 1991.

[4] C. Collet, M. N. Huhns, and W.-M. Shen, "Resource integration using a large knowledge base in Carnot," *IEEE Computer*, Vol. 24, No. 12, Dec. 1991, pp. 55–62.

[5] R. V. Guha, "Micro-theories and Contexts in Cyc Part I: Basic Issues," MCC Technical Report Number ACT-CYC-129-90, Microelectronics and Computer Technology Corporation, Austin, TX, June 1990.

[6] D. Heimbigner and D. McLeod, "A Federated Architecture for Information Management," *ACM Transactions on Office Information Systems*, Vol. 3, No. 3, July 1985, pp. 253–278.

[7] D. Lenat and R. V. Guha *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*, Addison-Wesley Publishing Company, Inc., Reading, MA, 1990.

[8] W. Litwin, L. Mark, and N. Roussopoulos, "Interoperability of Multiple Autonomous Databases," *ACM Computing Surveys*, Vol. 22, No. 3, Sept. 1990, pp. 267–296.

[9] S. B. Navathe, S. K. Gala, S. Geum, A. K. Kamath, A. Krishnaswamy, A. Savasere, and W. K. Whang, "Federated Architecture for Heterogeneous Information Systems," *NSF Workshop on Heterogeneous Databases*, Dec. 1989.

[10] A. P. Sheth and S. K. Gala, "Attribute Relationships: An Impediment in Automating Schema Integration," *NSF Workshop on Heterogeneous Databases*, Dec. 1989.

[11] A. P. Sheth and J. A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, Vol. 22, No. 3, Sept. 1990, pp. 183–236.

[12] J. de Souza, "SIS—A Schema Integration System," *Proc. Fifth British National Conference on Databases (BNCOD5)*, Cambridge University Press, 1986, pp. 167–185.

[13] R. Wang and S. E. Madnick, "Facilitating Connectivity in Composite Information Systems," *Data Base*, Fall 1989, pp. 38–46.