The Carnot Heterogeneous Database Project: Implemented Applications

Munindar P. Singh, Philip E. Cannata,[1] Michael N. Huhns,[2] Nigel Jacobs,[3] Tomasz Ksiezyk,[4] Kayliang Ong,[5] Amit P. Sheth,[6] Christine Tomlinson,[7] and Darrell Woelk[8]

Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA

singh@ncsu.edu

Abstract

The Carnot project was an ambitious research project in heterogeneous databases. It integrated a variety of techniques to address a wide range of problems in achieving interoperation in heterogeneous environments. Here we describe some of the major implemented applications of this project. These applications concern (a) accessing a legacy scientific database, (b) automating a workflow involving legacy systems, (c) cleaning data, and (d) retrieving semantically appropriate information from structured databases in response to text queries. These applications support scientific decision support, business process management, data integrity enhancement, and analytical decision support, respectively. They demonstrate Carnot's capabilities for (a) heterogeneous query processing, (b) relaxed transaction and workflow management, (c) knowledge discovery, and (d) heterogeneous resource model integration.

[1]IBM, Austin, TX.
[2]Department of Electrical & Computer Engg., University of South Carolina, Columbia, SC.
[3]Independent consultant, Austin, TX.
[4]MCC, Austin, TX.
[5]Trilogy Corporation, Austin, TX.
[6]Department of Computer Science, University of Georgia, Athens, GA.
[7]Rosette Webworks, Austin, TX.
[8]Rosette Webworks, Austin, TX.

Introduction

Even as database technology has made significant inroads into real applications, nontrivial problems in information automation still remain. All too often, enterprise information systems consist of a diverse mix of applications, files, and databases that are each individually essential, but do not cohere well as a whole. Many of these systems were not designed as such, but have just evolved to keep up with new needs and technologies. This has resulted in a mix of operational systems that collectively manage huge amounts of data. This data is frequently critical to an enterprise but also redundant and inconsistent. It takes significant human effort to analyze, clean up, and turn this data into the information that is necessary to manage the enterprise.

The need to access diverse information systems in a logically coherent manner translates into a number of technical challenges. These include

- Interoperability, despite heterogeneity with respect to
  - communication protocols
  - database connection protocols
  - query languages
  - logical schema access
  - application semantics
- Distribution of resources
- Autonomy of resources in terms of
  - metadata and schemas
  - legacy applications and closed transactions.

The Carnot architecture provides a flexible framework for addressing the challenges. It presupposes an open, standards-based, distributed computational approach. In order to meet the above challenges, the Carnot philosophy recognizes the importance of mediated access to passive and active resources such as databases, knowledge sources, and applications. Carnot includes a facility for specifying constraints among resources. These constraints can implement a variety of data and application integration concepts, including transparent access to data at the conceptual level and strategies to maintain or restore consistency in the face of various contingencies.

The above situation is well-recognized [5, 19, 32]. Like the Carnot project, a number of research projects have addressed this problem by developing toolkits that enable interoperation to varying extents. These are excellently reviewed and tabulated in [22], so we shall not discuss them in detail here. Instead, we shall concentrate on the implemented applications of the Carnot project that showcase its key technical features. Of course, several research projects build prototypes, as did Carnot, but the applications we discuss are more significant in that they were realized in *customer* organizations. These applications cover a wide range of business needs and technical problems. We shall strive to highlight the technical—and sometimes the business—insights behind these applications. More detailed information on the Carnot architecture, and on some its modules can be found in the literature, notably [2, 6, 14, 17, 29, 33, 34].

Section 1 gives an overview of the Carnot architecture, with just enough detail to understand the applications. Section 2 describes the Carnot approach to legacy system access at the Eastman Chemical Company. Section 3 describes an exercise in workflow automation at Ameritech. Some details of this application are available in [27], but we include a brief discussion here, because it illustrates an interesting capability of Carnot. Section 4 describes a data purification effort at Bellcore. Section 5 describes our approach to coherently accessing and fusing structured and unstructured data, which was implemented at the U.S. Department of Defense. Section 6 summarizes the key lessons from Carnot, including a historical overview and a list of external Carnot publications.

1 Overview of the Carnot Architecture

Carnot is composed of the following five major layers of services: semantic, distribution, support, communication, and access. The Carnot execution environment is a software component called the *Extensible Services Switch (ESS)*. The ESS is a distributed interpreter that provides access to communication, information, and application resources at a site [30]. The ESS is constructed in Rosette, an actor language enhanced with object-oriented mechanisms [1, 10]. Rosette and its interpreter were developed over five years of research on parallel algorithms and control of distributed applications. They were enhanced during the Carnot project. Rosette includes some useful facilities that enable its use in initiating and coordinating activities at distributed sites.

- Remote evaluation: it is easy to evaluate expressions at a remote site, and the expressions can be sent along with as much of the environment as the programmer chooses.
- Treespaces: these are related to the Linda tuplespaces, and enable the structuring of a namespace to allow symbolic, pattern-directed communications among different sites [7].
- Lightweight threads and concurrent execution: each actor in Rosette has a lightweight thread. These can be spawned off naturally, and used to realize concurrency.

The above features make Rosette an effective infrastructure for coordinating information resources and transactions. Although now (circa 1996) a number of products, e.g., Java, are available that have many of the above features, a few years ago Rosette was quite exceptional for a system of its size and complexity.

A consequence of the above properties is that the ESS can exist as a single process at each host (this is the typical configuration). This process contains actors (or threads) for each computation in which it is engaged. The ESSs at different hosts communicate with each other and can invoke local operations based on their interactions. The operations can be invoked either through specialized actors within the ESS, or by spawning off separate operating system processes.

Consequently, the ESS can invoke operations at multiple databases concurrently, and manage the gathering and combination of results. The ESS enables the integration of new facilities and services, and the configuration of a desired system in the field. By this we mean that starting with a vanilla ESS, one can easily add the necessary functionality, for example, modules to interface with a database management system of one's choosing.

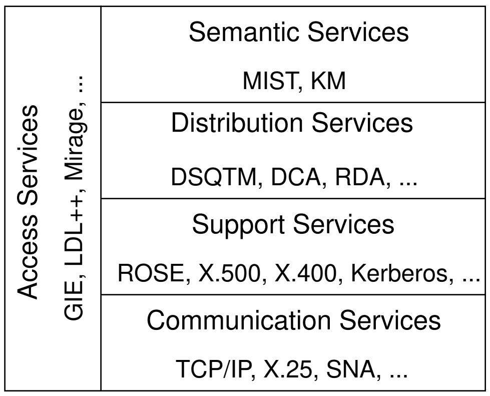

Figure 1: The Basic Carnot Layers

Figure 1 shows the five layers of Carnot schematically—four of the layers are stacked one on top of the next; the access services layer applies to each of the other four. Our main focus here is on distribution and semantic services; we only briefly describe the other services here. Physically, all of the necessary communication and support services are instantiated as part of the same ESS. Some of the distribution services functionality, written in Rosette, is also instantiated in the ESS. The semantic services tools remain separate modules, with the exception of a knowledge representation tool that was written in Rosette, although it is typically not loaded into the specific ESS that contains DSQTM functionality. The access services are also separate—a graphical interface toolkit was integrated with the ESS, but of the applications described below, it was used only in the one at Ameritech.

The communication services implement and integrate various communication platforms that provide functionality up to the application layer of the ISO OSI reference model [23]. Examples of such platforms include ISO OSI session and presentation layer protocols running on top of TCP/IP, ISO TP4 with CLNP, X.25, or SNA. The support services implement ISO OSI Association Control (ACSE), ISO OSI Remote Operations (ROSE), CCITT Directory Service (X.500), CCITT Message Handling System (X.400), and the Kerberos authentication service.

The distribution services provide directory services, and manage logical data access from multiple heterogeneous sources, as well as information consistency and integrity across such sources. This layer adds workflow and relaxed transaction processing capabilities with flexible transaction primitives that enable the specification and execution of workflows.

The *Distributed Semantic Query and Transaction Manager (DSQTM)*, which physically resides inside an ESS process, uses the data dictionaries to produce the scripts that distribute the queries to other ESSs or databases as needed, and collect and process results, e.g., to automatically perform joins or make any necessary domain value translations (value maps), where necessary. The DSQTM includes actors that embody the protocols for accessing various DBMSs and other resources. These include Oracle, Sybase, Ingres, Objectivity, and Verity (Topic). It also has an implementation of the Relational Data Access (RDA) standard, which sought to provide a protocol for accessing databases through an open transaction model (with primitives to open and close connections, and begin and rollback or commit transactions). RDA is no longer commercially supported, although similar ideas are supported in the ODBC standard.

The distribution layer also includes the *Distributed Communicating Agent (DCA)* facility. DCA is a tool that supports the modular construction and interoperation of agents. DCA supports human (user interface) and computational (knowledge-based expert system) agents, as well as databases. Each expert system agent, called a RAD agent, can perform forward and backward reasoning, and includes a frame system with multiple inheritance, distributed truth-maintenance [11], and contradiction resolution. The ESS manages communications among the agents: actors in the ESS serve as communication aides, one for each agent, and forward messages through the ESS treespace.

Lastly, distribution services include the *Declarative Resource Constraint Base (DRCB)*. The DRCB is an extended data catalog that captures interresource dependencies, consistency requirements, contingency strategies, and organizational rules. The interresource dependencies are expressed as mappings in a dictionary. The other aspects are realized through a knowledge-based agent.

The semantic services consist of a suite of tools for enterprise modeling, model integration, data cleaning, and knowledge discovery. The *Model Integration and Semantics Tool (MIST)* is used to generate mappings and consistency constraints, which form the basis for semantic mediation among information resources in the distribution services. MIST relies on a *common ontology* to assist a sophisticated user or DBA to perform model integration [13]. An ontology is a representation of the concepts and their relationships that characterize a given domain of interest [9]. Database schemas, even from the same application domain, implicitly involve various distinct concepts, which makes it difficult to relate them. However, by relating different database schemas to a common ontology, we can semantically relate the schemas with each other, and thereby enable interoperation of the underlying databases. MIST can work with a common ontology expressed in Cyc, or in Carnot's own knowledge representation tools called KRBL, which is described next. Cyc is a knowledge base being built—first as an MCC project and later as a separate company—to contain all of the "commonsense" knowledge that underlies any specialized domain, and can thus provide a basis for relating databases [20]. Carnot was one of the pioneers of an ontology-based approach to interoperation.

The knowledge-based RAD agents are used to capture the consistency constraints to help maintain the coherence of applications executing across autonomous information resources. The agents use models of each other and of the resources local to them so as to communicate and cooperate effectively. Resource models may be the schemas of databases, frame systems of knowledge bases, or process models of business operations. These enable relaxed, distributed transactions to execute concurrently across heterogeneous databases that previously had incompatible semantics. Thus the appearance and effect of homogeneity among heterogeneous resources is obtained.

The semantic services also included the *Knowledge Representation Base Language (KRBL)* as a tool for representing and accessing ontologies. This tool has a simple frame-based representation of knowledge. It has certain important features, such as the ability to represent n-ary relations, metaclasses, and metarelations, which are essential for representing higher-level knowledge. KRBL was written in Rosette, but supported a generic functional interface to a knowledge base through which it could be accessed from any language, specifically Lisp and C++. The generic functional interface was a slight extension of the one introduced in [4].

Another semantic tool is *Knowledge Miner (KM)*, which is used for knowledge discovery. It includes symbolic inductive learning and statistical clustering techniques, which it combines with the LDL++ deductive database environment (to be described). The knowledge discovery methods infer patterns and regularities from information resources and check consistency between information and corresponding models [24, 25]. The discovery is guided by expectations about the nature of the information and its embedding in the application.

The access services provide mechanisms for personal and group interaction with Carnot services. Our user interface software includes a 2D and 3D model-based visualization and animation facilities, GIE and Mirage. Although these interfaces were used early in Carnot, we shifted toward simpler and more common frameworks such as Motif and HTML. The access services also include the *Logical Data Language (LDL++)*, which provides a Prolog-like rule-based language optimized for database access [21].

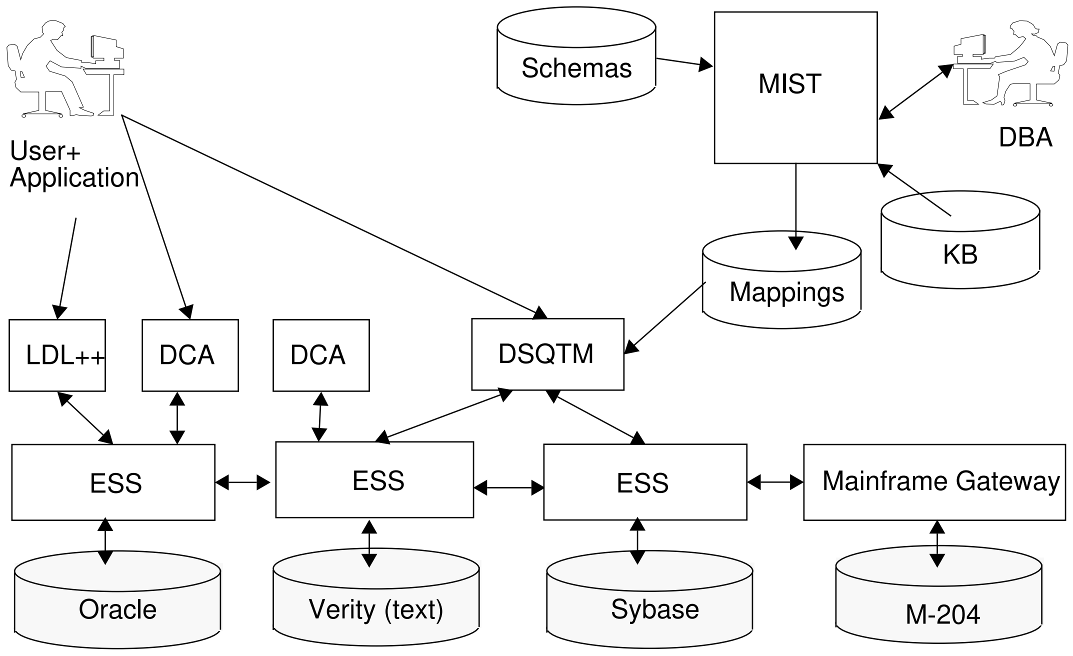

Figure 2: Schematic Carnot Configuration

Figure 2 shows some example Carnot configurations with the major components and their interrelationships. We merge the application program with the user icon. The shaded drums refer to external databases or filestores. The unshaded (and slightly smaller) drums refer to representations that Carnot reads and reasons about, or produces. The "mainframe gateway" refers to software provided by different mainframe manufacturers to interface their mainframes with Unix-based workstations. In order to simplify the system design, the ESS uses gateways to attach mainframe databases. We now turn to specific Carnot applications, where this basic configuration is instantiated in different ways.

2 Legacy System Access

This application involves accessing data from a legacy database for scientific decision support at Eastman Chemical Company (ECC). As one would expect from a large company that has been in business for a number of years, ECC maintains information about chemical research, development, manufacturing, marketing, and customer contacts. This information, some of it going back 35 years, is contained in several large, incompatible databases.

Historically, queries to these databases have been very difficult, often requiring an expert to assist in retrieving information and taking days to weeks to satisfy a single query. This delay proves particularly expensive in the case of the research chemical database, which contains about 100 tables of information about experiments conducted by ECC chemists. Chemists are called upon to determine the properties of different compounds, or to find compounds that meet different requirements. If the information produced in previous experiments cannot be found, the chemists are forced to redo their experiments. Redoing an experiment is not only time-consuming, but can cost up to several thousand dollars.

We briefly describe the domain in order to better motivate our approach. The domain consists of chemical compounds, which are identified by unique names and defined as compositions of other chemicals. Roughly about 100 different chemical and physical tests are performed on different compounds to measure their properties of interest to various applications (the details of these tests are proprietary and, in any case, not of interest to our readers). The `composition` table represents the main entity; there is a table for each experiment recording (few to several) values in different columns. Since the number of chemicals in a compound is not limited, the ECC database designers used a flattening representation in which each compound may be represented through a set of tuples in the `composition` table, each tuple carrying the identifier of the compound and the given chemical and its amount. The above database is primarily of interest to scientists; another database contains information of interest to marketers, and uses a different key, but we shall not discuss in any detail that here.

Several of the Carnot technology components were used to implement a solution as shown in Figure 3. The system is in operation and is undergoing further enhancements. When the system is setup, a knowledge base is created containing representations of all the logical views of interest. In addition dictionaries are created that relate the logical views to the physical tables and columns in the Polymer Research (PDRS) database.

Our implementation includes two user interfaces, one that is forms-based and uses LDL++, and the other a natural language one provided by MCC's Knowledge-Based Natural Language project (KBNL) [3]. LDL++ supports

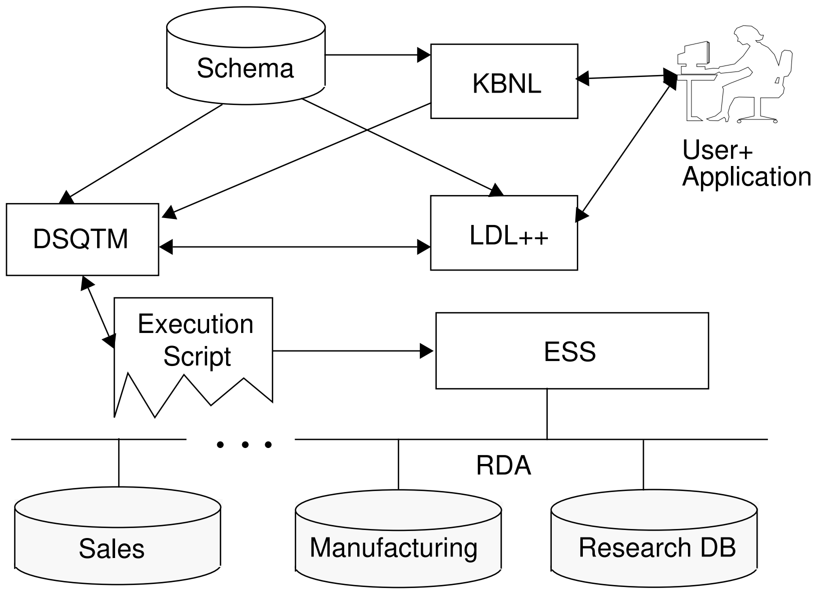

Figure 3: Architecture for Legacy System Access

the formulation of complex queries as logical rules. Briefly, the LDL++ compiler gathers the rules for each query, generates compact SQL statements, and dispatches them to the database server via the ESS. We show some sanitized examples to give a flavor of the steps involved. LDL++ represents `composition` and the experimental result tables as predicates:

```
composition(Id: int, Code: string, quantity: float)
```

One of the interfaces is a form by which scientists can find the results of tests on specified compounds. A technically more interesting query is to find compounds that satisfy some range conditions on the chemicals that compose them. For example, one might ask to see all compounds that contain 5-10% of `A` and 50-67% of `B`. This yields the following list of constraints:

```
[compositionC(Id,'A',range(5.0,10.0)),
 compositionC(Id,'B',range(50.0,67.0))]
```

From the input range values, the LDL++ application performs some reasoning using domain knowledge to validate the constraints and possibly to augment them. If successfully validated, the augmented constraints are converted to SQL and executed on the database.

```
SELECT DISTINCT TB_COMPOSIT_1.PEL_ID
FROM TB_COMPOSIT TB_COMPOSIT_1 , TB_COMPOSIT TB_COMPOSIT_2
WHERE TB_COMPOSIT_1.PEL_ID  = TB_COMPOSIT.PEL_ID
      AND TB_COMPOSIT_1.AMO_CODE =  'A'
      AND TB_COMPOSIT_1.AMOUNT >= 5 AND TB_COMPOSIT_1.AMOUNT <= 10
      AND TB_COMPOSIT_2.AMO_CODE =  'B'
      AND TB_COMPOSIT_2.AMOUNT >= 50 AND TB_COMPOSIT_2.AMOUNT <= 67
```

The KBNL system takes English inputs and, after appropriate interactions with the user, produces a high-level SQL query, and hands it over to Carnot for processing. Example English inputs include (i) `Find polymers with elongation of 3.3 and creep of 4.4`, (ii) `What is the creep for polymer A?` and so on. KBNL interacts with the user to disambiguate their information request to the level of the conceptual schema, e.g., to determine that the user cares about `chemical resistance break elongation` and `adhesive creep at time 50`, making the query effectively be:

```
Find polymers with chemical resistance break elongation of 3.3 and
adhesive creep at time 50 of 4.4
```

KBNL then produces an SQL query, involving logical tables and columns, and forwards it to Carnot:

```
SELECT *
FROM POLYMERS
WHERE POLYMERS.PEL_ID = CHEMICAL_RESISTANCE.PEL_ID
AND POLYMERS.PEL_ID = ADHESIVE_CREEP.PEL_ID
AND CHEMICAL_RESISTANCE.ELONGATION_AT_BREAK = 3.3
AND ADHESIVE_CREEP.TIME_050 = 4.4
```

The above SQL query is received and processed by the DSQTM. The DSQTM generates and the executes the following low-level SQL query:

```
SELECT DISTINCT TB_COMPOSIT.PEL_ID
FROM TB_COMPOSIT , TB_CHEM_RES , TB_ADH_CREP
WHERE TB_CHEM_RES.BRK_ELNG  =  3.3
      AND TB_COMPOSIT.PEL_ID  =  TB_CHEM_RES.PEL_ID
      AND TB_COMPOSIT.PEL_ID  =  TB_ADH_CREP.PEL_ID
      AND TB_ADH_CREP.TIME_050 = 4.4
```

In order to produce the above query, the DSQTM uses knowledge about KBNL's logical view, and its mapping to the physical view. The interesting part of this knowledge is in the form of articulation axioms. In this application, most of the axioms give a straight one-to-one mapping. However, some axioms encode that the objects of the logical view are represented as split across multiple rows in the physical table–these are the *id convention* axioms below. These axioms are used in a queries similar to the one we showed above using LDL++, so we shall not discuss them again.

```
# Mapping KBNL and PDRS views to common (ECC) view
KBNL ADHESIVE_CREEP TIME_050 <==> ECC_ADHESIVE_CREEP TIME_050
KBNL ADHESIVE_CREEP TIME_150 <==> ECC_ADHESIVE_CREEP TIME_150
KBNL CHEMICAL_RESISTANCE ELONGATION_AT_BREAK <==>
                   ECC_CHEMICAL_RESISTANCE ELONGATION_AT_BREAK

KBNL ADHESIVE_CREEP POLYMERS <==> ECC ECC_ID_CONVENTION ECC_NAME
KBNL CHEMICAL_RESISTANCE POLYMERS <==> ECC ECC_ID_CONVENTION ECC_NAME
PDRS TB_ADH_CREP PEL_ID <==> ECC ECC_ID_CONVENTION PEL_NAME
PDRS TB_CHEM_RES PEL_ID <==> ECC ECC_ID_CONVENTION PEL_NAME


# ECC_NAME is represented as multiple columns in TB_COMPOSIT
PDRS TB_COMPOSIT PEL_ID  <==> ECC_ID_CONVENTION ECC_NAME
PDRS TB_COMPOSIT AMP_CODE <==> ECC_ID_CONVENTION ECC_NAME
PDRS TB_COMPOSIT AMOUNT   <==> ECC_ID_CONVENTION ECC_NAME
PDRS TB_COMPOSIT AMO_CODE <==> ECC_ID_CONVENTION ECC_NAME

# TB_COMPOSIT joins with every table that has a PEL_NAME column
PDRS TB_COMPOSIT PEL_ID  <==> ECC_ID_CONVENTION PEL_NAME
```

Our implementation at ECC involved a commercial implementation of the RDA protocol, which enables access to backend nonrelational and older relational databases.

For the above architectural framework to be effective, the logical view of the database taken by the KBNL project must agree with the logical view supported by the DSQTM. Since we were dealing with about 100 tables, many involving concepts that were arcane to us, we formulated a shared representation of the database schemas. We defined a number of scripts to map these shared schemas into the knowledge base used by KBNL to understand and disambiguate English queries, by LDL++ to form predicate descriptions, and by the DSQTM to build its dictionaries. The success of the above approach notwithstanding, there is an acute need for commercial products to manage dictionaries and views, as well as design rationales for the same.

3 Workflow Automation

Workflows, especially database-oriented workflows, have emerged as the leading paradigm for structuring complex computations in heterogeneous information environments [8]. Carnot was one of the pioneers in workflow management from a database perspective (as opposed to the more traditional organizational or groupware perspective). Workflows are important wherever there are complex, long-running, flexible, interactive flows of information and control. As a rule, the service industry has been the prime ground for deploying workflow technology. The telecommunications industry, particularly its customer service component, is no exception.

Carnot was applied to the *service provisioning* activities of Ameritech, one of the Bell companies and a sponsor of Bellcore, in turn a sponsor of Carnot. Service provisioning refers to the task of connecting a customer to the system—assigning a connection to them, making sure the physical infrastructure exists, and updating the various databases. Interestingly, this task could take up to two weeks, involving tens of operations on over a dozen operational (legacy) systems.

Our aim was to prototype a system through which the throughput and delay of the service provisioning activity could be improved. To maximize the impact, a simple, but important, service was chosen for prototyping. However, we realized that a significant effort—several person-months—was required simply to identify the information and control flows taking place in the organization. A large fraction of this effort was expended by our customers, because they alone knew their operational details, although we assisted in trying to understand and debug their specification.

Our system consists of four DCA agents—a user interface agent and three RAD expert system agents. The user agent assists the user in ensuring that the service request is valid. When it is completed, it sends a message to the scheduling agent. The scheduling agent determines a workflow schedule—initially, this is the normal case of the execution of the activity. The schedule processing agent executes the tasks in this schedule by invoking operations on the back-end systems concurrently. Some of these operations require significant protocol conversion, e.g., in generating messages that can be sent via custom interfaces to legacy systems or mainframes. Exception conditions are captured declaratively in the schedule repair agent. These conditions determine when the composite activity should be aborted and when different component transactions should be retried or compensated. Additional details of this application are available in [27].

The success of this application derives from its taking into account an entire run through the system, not just focusing on some of the pieces. The expert system technology that we used here was key to rapid prototyping, but is by no means conceptually essential in a production version. However, capturing execution conditions is a major challenge for conventional programming, and is facilitated by a rule-based language that includes support for maintaining dependencies among various decisions. How to handle exceptions elegantly and

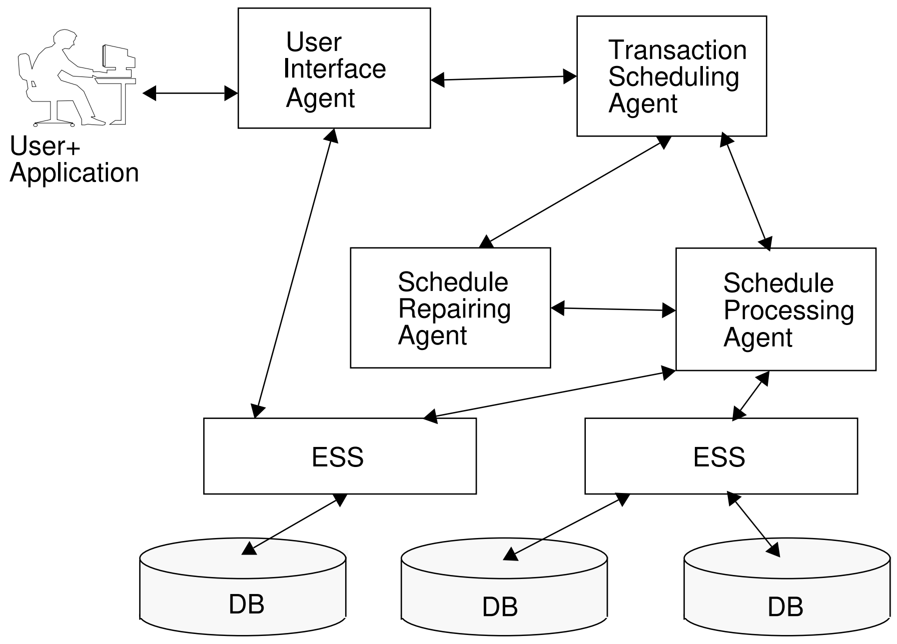

Figure 4: Architecture for Workflow Automation

efficiently remains a limitation of current workflow products and research [18].

4 Data Purification

Data is considered a vital company asset and the quality of data plays a critical role in the quality and efficiency of the operations. Unfortunately, huge corporate data resources are often plagued with errors. Data is often either inconsistent, incorrect, incomplete, or not current. Poor data quality can be attributed to many reasons, including flawed data acquisition, erroneous creation, flawed updates, lack of integrity enforcement across multiple databases, process reengineering, and corporate reorganization.

Bellcore, the research arm of the seven regional telephone companies, began a data quality project through an application partnership with the Carnot project. The two main purposes were data validation and cleaning of large databases of telephone-related information. Poor data quality causes two major problems for the telephone companies. First, it impedes workflow automation because of more frequent and unnecessary exceptions during processing. Second, it makes it harder to provide good customer service due to the inconsistency and incompleteness of data [26]. In order to perform data validation

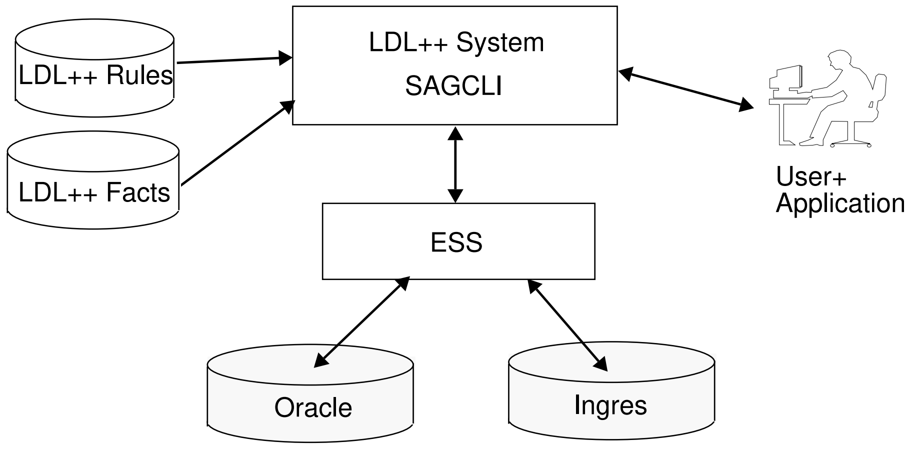

Figure 5: Architecture for Data Purification

and cleaning, we determined that three capabilities are required:

- *Database Access.* Telephone companies have a wide range of heterogeneous databases, which must be accessed before any data can be validated and cleaned.
- *Specification of Complex Validation Rules and Queries.* Data is validated based on the rules that either define whether the given data is correct or is suspected to be incorrect. Processing each rule may require complex operations such as joining, selection, and aggregation.
- *Rapid Refinement of Validation Specification.* The system verifies correctness and cleanliness of data with respect to specifications of what is valid. Typically, it takes several iterations to determine the right specification.

LDL++ and ESS were identified as the core Carnot technologies that fulfill the above requirements. As shown in Figure 5, the ESS provides access to the Oracle database. LDL++'s declarative nature facilitates specification of the validation conditions as a set of LDL++ rules. A graphical user interface built at Bellcore—using a commercial product, Galaxy—allows users to dispatch different validation queries, review the results and take corrective actions. As for ECC, LDL++ generates SQL, which is executed via the ESS. The user is prompted to verify if any apparent discrepancy detected by the system is indeed an error and how it might be fixed.

The database is validated for different types of constraints. These include the following (to preserve proprietary information, these have been heavily sanitized):

- *Domain Value Constraints*—columns values must be from a certain range.

```
channel(length,range(0.0,10.0))
channel(conductivity,range(50.0,117.0))
```

- *Quantitative Constraints*—values in different columns must satisfy certain numerical constraints. We give two examples below. Violations of the first constraint—if a channel connects to a piece of equipment, then the channel must terminate at that equipment—are obviously errors. Conversely, data that satisfies the second constraint—if a loop links to a cable, then the start location of the loop equals the end location of the cable—is also likely to be erroneous.

```
link(channel,equipment) => equal(endLoc(channel), loc(equipment))
link(loop,cable) => equal(startLoc(loop), endLoc(cable))
```

- *Uniqueness Constraints*—some column values must be unique, e.g., a cable only links to one loop.

```
link(loop1,cable), link(loop2,cable) => equal(loop1,loop2)
```

- *Referential Integrity Constraints*—the existence of a value in one place may presuppose its existence elsewhere in the system, possibly in another database.

```
link(loop,cable) => loopInfo(loop, \_, \_)
```

5 Combining Structured Data and Text

The U.S. Department of Defense (DoD) has several text as well as traditional structured databases, which must be used in concert. We considered a problem where the *Verity Topic* text retrieval system is used. A Topic query takes the form of a weighted tree of concepts including target words and phrases. Verity processes the user's query and presents a ranked list of matching documents. The query trees can be saved and reused or refined at a later date.

The DoD also has many structured databases containing important historical information. These databases are often maintained by individual staff members, and have evolved without organizational standards, employing different designs, database software, and hardware platforms. Ideally, the users should corroborate their findings from documents with information from these databases. But this requires them to learn about different schemas, to master

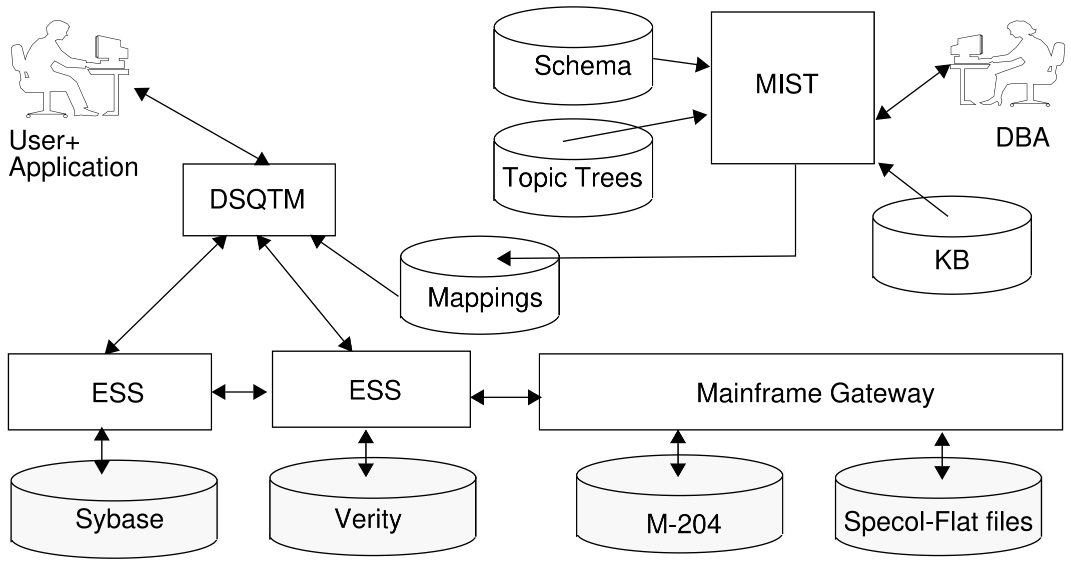

Figure 6: Architecture for Combined Structured Data and Text Access

multiple database query languages, and to be able to interpret the data that is returned.

Carnot was applied to this case as follows. As shown in Figure 6, at compile-time, unstructured (textual) and structured (relational) data sources were integrated via a common conceptual model [14]. At run-time, database queries were managed by a distributed network of database agents [30].

A simplified Topic tree is shown in the top left of Figure 7. This tree captures the concept of a MiG29, a Soviet/Russian fighter plane, through terms related to it. The concepts are structured as a tree to assist in modularizing them, but there is no semantics other than that the concepts of the child nodes are associated with the concept of the parent node. Intuitively, an article, e.g., a newswire report, might be about MiG29s if it mentions enough of the terms related to MiG29. (This is a simplified description of the assumption implicitly made by Verity and other text retrieval systems.) The top right of Figure 7 shows a simplistic ontology with concepts pertaining to weapons, fighter aircraft, and the human designers of weapons. Regular lines represent generalization/specialization links; dashed lines represent attributes of classes; wavy lines represent named relationships among classes. The bottom right shows two database tables with information about fighter aircraft, and about persons, respectively. There might be an entry for MiG29 in the first table, and one for its designer, Mikoyan, in the second table. The Topic tree and database tables are related through the ontology.

First, a Topic concept tree parser is used to create internal representations

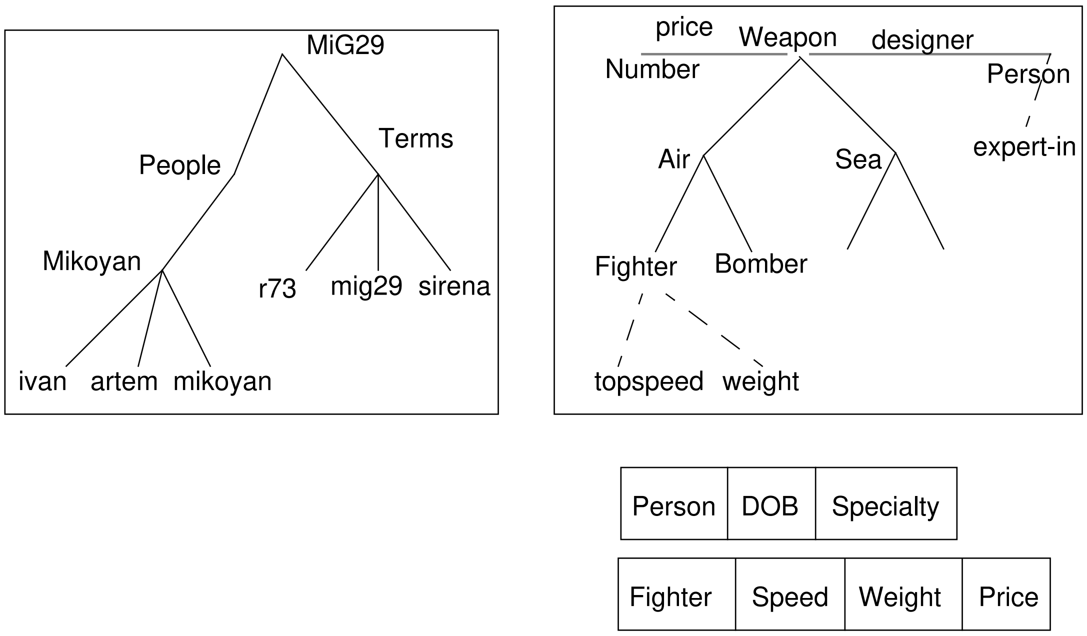

Figure 7: Relating Topic Trees and Database Schemas

of Topic trees that can be used by MIST (this process is referred to as *preintegration*). Second, MIST is then used to map each concept in the trees to corresponding concepts in a common ontology. This step provides a semantic interpretation for each tree, and identifies concepts from different trees having the same meaning. Relational database schemas are also mapped to the same common ontology. The two sets of mappings yield executable linkages among the Topic trees and structured databases. The above task is performed by sophisticated users or resource administrators.

Now when a user issues a Topic query (either chosen from an existing library or constructed using existing concepts), the DSQTM uses the above mappings to produce a corresponding set of SQL queries. The original query is executed on the text engine; the SQL queries are executed on the structured databases. The DSQTM fuses the results. One effective way to fuse these results in our application was to produce a hypertext document from the "articles" returned by the test search. The hypertext essentially links the selected words, e.g., "MiG29," with results of queries from the structured databases providing additional information to the user, e.g., the price or top speed of the aircraft. We adopted the HTML standard for these documents, which enables using commercial WWW browsers.

Thus, the user is provided relevant data from the structured databases, but shielded from their uninteresting details. The mappings are constructed by expert users, but since the mappings are reusable, their effort is leveraged to facilitate the work of other expert and naive users.

6 Conclusions

The above applications include some of the major and common business needs that heterogeneous database systems must address:

- Getting to the data with a high-level view
- Coordinating transactions across systems
- Data cleaning
- Fusing traditional data with nonstandard data.

It is instructive to see how Carnot addressed these different problems through a uniform framework. The problem of accessing data through high-level views was well-known before Carnot, although few commercial solutions are available even today. The growth of the workflow and data cleaning industries over the past few years is phenomenal, but these problems were just barely being understood when Carnot implemented the corresponding applications. It is now well-known that nontraditional, unstructured data such as text is extremely common in most practical applications of computing. Yet, a few years ago, unstructured data was not given its due importance within the database community.

Carnot contributed a number of interesting ideas to database research. These include

- Development of powerful tools to perform resource integration, even among resources of different models.
- Use of intelligent agent technology to coordinate transactions and workflows, in particular encapsulating exception conditions for workflows.
- Use of actor technology with scripting languages to coordinate distributed activities (at a lower level of abstraction than the intelligent agent techniques).
- Use of deductive database technology to provide natural access to data, especially for data intensive applications such as knowledge discovery and data purification.

More practically, we learned a number of important lessons which, though obvious, merit emphasis:

- Standards are extremely valuable in increasing the effectiveness of one's effort. Even standards that later prove to be dead-ends are better than no standards, because they force one to define component interfaces more cleanly.

- A lot of time and effort goes into debugging the requirements of the applications.
- Even low-level tools such as Perl can be quite effective in managing complexity, provided one does not expect too much in the way of abstractions.
- There is some data cleaning required in almost every application.
- Establishing correspondences between schemas and ontologies is nontrivial, but often worth the effort in understanding the requirements—it is a form of data cleaning applied at the schema level.

Carnot as a research project addressed some of the most challenging problems in making heterogeneous information systems function effectively. It possibly contributed in giving those problems greater visibility and importance. Carnot involved developing new theories, prototyping them locally, and finally implementing and deploying them at customer sites. By having essentially the same group of people engaged in the applications as developed the given techniques, we were able to reduce the time lag in technology transfer. However, success is determined by more than just the soundness of an approach or the technical acumen of the scientists involved. It requires the presence of champions of the technology in the sponsoring organizations, who are willing to assign the human and infrastructural resources necessary to identify their most interesting problems, understand them with sufficient technical detail to solve them, and finally to carry them out.

This was not an easy task, especially in end-user organizations, who would have greatly preferred (and rightly so) to obtain some commercial off-the-shelf (COTS) software. However, when commercial solutions were not forthcoming, these organizations were willing to step forward to take the lead. We emphasize that, despite the "success" of the above applications, they are still prototypes, albeit carried out at a remote organization. A lot more effort is required before these applications can be considered commercial quality.

Historical Remarks

The Carnot project was a descendant of the Object-Oriented & Distributed Systems (OODS) and the Reasoning Architectures (RA) projects at MCC. From its inception, Carnot sought to marry artificial intelligence, distributed computing, and database techniques to address the problems of heterogeneous and distributed information systems. The Carnot staff had a similar range in training and professional background.

The Carnot project was a consortial research project, with sponsorship from organizations that included (a) software vendors, (b) large end-users of software, and (c) consultancies. For this reason, the Carnot project developed *application partnerships* as a formal method of interacting with sponsors. These were in addition to the more traditional periodic technology transfer workshops and project reviews. The partnerships were designed to link the Carnot research staff with developers at the sponsor organizations so as to deploy the Carnot technologies in real applications. The goals of these partnerships were threefold. First, they would provide real-world demonstrations of (and serve as test-beds for validation of) the Carnot technologies, thereby assisting in refining research directions. Second, each end-user participant would receive a prototype solution to one of their urgent problems, which would help refine their understanding of their problems. Third, the partnerships would provide market development for the vendor participants, thus facilitating commercialization of the Carnot technology.

There were a number of serious attempts made for the commercialization of Carnot. These failed due to a variety of business reasons. However, the Carnot sponsors did give Carnot among the best technical reviews of the MCC projects. Although Carnot has officially ended, its intellectual property became the substrate for the *InfoSleuth* project [36]. Work is continuing, and some of the Carnot components may yet be commercialized.

Carnot Publications

At the advice of a referee, we include references to the external Carnot publications, classified as follows.

- *Conceptual aspects and architecture:* [6, 34]
- *Extensible services switch:* [31, 30]
- *Resource integration:* [15, 35, 12, 13, 16, 14, 17]
- *Relaxed transaction processing:* [2, 33, 29, 27, 28]
- *Knowledge discovery:* [24, 25].

The above papers discuss the various components of Carnot as they evolved. There was obviously significant research effort in the relaxed transaction processing and resource integration components. Almost all of this work was prototyped within Carnot, although not all of it was deployed in real applications during Carnot's lifetime. As remarked above, this work has been inherited by Carnot's successor project, *InfoSleuth.* Further, most of the Carnot research staff have taken up positions elsewhere, and are pursuing many of the research goals that first attracted them to Carnot. Thus, the legacy of Carnot lives on.

7 Acknowledgments

We are indebted to the Carnot sponsors for supporting our efforts financially and morally, even through difficult times. We are also indebted to the anonymous referees for their careful reading of this paper, and comments that have greatly improved it. Munindar Singh was partially supported by the NCSU College of Engineering, and the National Science Foundation under grants IRI-9529179 and IRI-9624425.

References

[1] Gul A. Agha. *Actors.* MIT Press, Cambridge, MA, 1986.

[2] Paul C. Attie, Munindar P. Singh, Amit P. Sheth, and Marek Rusinkiewicz. Specifying and enforcing intertask dependencies. In *Proceedings of the 19th VLDB Conference*, pages 134–145, August 1993.

[3] Jim Barnett, Kevin Knight, Inderjeet Mani, and Elaine Rich. Knowledge and natural language processing. *Communications of the ACM*, 33(8):50–71, August 1990.

[4] Jim Barnett, Juan Carlos Martinez, and Elaine Rich. A functional interface to a knowledge base: An NLP perspective. Technical Report ACT-NL-393-91, Microelectronics and Computer Technology Corporation, Austin, TX, 1991.

[5] Omran A. Bukhres and Ahmed K. Elmagarmid, editors. *Object-Oriented Multidatabase Systems: A Solution for Advanced Applications.* Prentice Hall, 1996.

[6] Philip E. Cannata. The irresistible move towards interoperable database systems. In *First International Workshop on Interoperability in Multidatabase Systems*, April 1991. Keynote address.

[7] Nicholas Carriero and David Gelernter. Linda in context. *Communications of the ACM*, 32(4):444–458, April 1989.

[8] Dimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–152, April 1995.

[9] Thomas R. Gruber. The role of a common ontology in achieving sharable, reusable knowledge bases. In *Proceedings of the Knowledge Representation and Reasoning Conference*, pages 601–602, 1991.

[10] C. Hewitt, P. Bishop, and R. Steiger. A universal modular actor formalism for Artificial Intelligence. In *IJCAI*, 1973.

[11] Michael Huhns and David M. Bridgeland. Multiagent truth maintenance. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1437–1445, 1991.

[12] Michael N. Huhns. Integrating information semantics for concurrent engineering. In *Proceedings of the First Workshop on Enabling Technologies for Concurrent Engineering*, April 1992.

[13] Michael N. Huhns, Christine Collet, and Wei-Min Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12):55–62, December 1991.

[14] Michael N. Huhns, Nigel Jacobs, Tomasz Ksiezyk, Wei-Min Shen, Munindar P. Singh, and Philip E. Cannata. Integrating enterprise information models in Carnot. In *International Conference on Intelligent and Cooperative Information Systems (CoopIS)*, May 1993.

[15] Michael N. Huhns, Nigel Jacobs, Tomasz Ksiezyk, Wei-Min Shen, Munindar P. Singh, and Philip E. Cannata. Enterprise information modeling and model integration in Carnot. In Charles J. Petrie, Jr., editor, *Enterprise Integration Modeling*, pages 290–299. MIT Press, 1992.

[16] Michael N. Huhns and Munindar P. Singh. The semantic integration of information models. In *Proceedings of the AAAI Workshop on Cooperation among Heterogeneous Intelligent Agents*, July 1992.

[17] Michael N. Huhns, Munindar P. Singh, Tomasz Ksiezyk, and Nigel Jacobs. Global information management via local autonomous agents. In *Proceedings of the 13th International Workshop on Distributed Artificial Intelligence*, August 1994.

[18] Mohan Kamath and Krithi Ramamritham. Bridging the gap between transaction management and workflow management. In *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems: State-of-the-art and Future Directions*, May 1996. `http:// optimus. cs.uga.edu:5080/ activities/NSF-workflow/ kamath.html`.

[19] Won Kim, editor. *Modern Database Systems: The Object Model, Interoperability, and Beyond.* ACM Press (Addison-Wesley), New York, NY, 1994. Reprinted with corrections, 1995.

[20] Douglas Lenat and R.V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc project.* Addison Wesley, Reading, MA, 1990.

[21] Shamim Naqvi and Shalom Tsur. *A Logical Language for Data and Knowledge Bases.* W.H. Freeman Publishers, New York, NY, 1989.

[22] Evaggelia Pitoura, Omran A. Bukhres, and Ahmed K. Elmagarmid. Object-oriented multidatabase systems: An overview. In *[5]*, chapter 10. 1996.

[23] Marshall T. Rose. *The Open Book*. Prentice Hall, 1990.

[24] Wei-Min Shen. Complementary discrimination learning with decision lists. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992.

[25] Wei-Min Shen, Bharat Mitbander, KayLiang Ong, and Carlo Zaniolo. Using metaqueries to integrate inductive learning and deductive database technology. In *Proceedings of the AAAI Workshop on Knowledge Discovery from Databases*, August 1994.

[26] Amit Sheth, Christopher Wood, and Vipul Kashyap. Q-Data: Using deductive database technology to improve data quality. In Raghu Ramakrishnan, editor, *Applications of Deductive Databases*. Kluwer Publishers, 1994.

[27] Munindar P. Singh and Michael N. Huhns. Automating workflows for service provisioning: Integrating AI and database technologies. *IEEE Expert*, 9(5), October 1994. Special issue on *The Best of CAIA'94* with selected papers from Proceedings of the 10th IEEE Conference on Artificial Intelligence for Applications, March 1994.

[28] Munindar P. Singh, L. Greg Meredith, Christine Tomlinson, and Paul C. Attie. An event algebra for specifying and scheduling workflows. In *Proceedings of the 4th International Conference on Database Systems for Advanced Applications*, April 1995.

[29] Christine Tomlinson, Paul Attie, Phil Cannata, Greg Meredith, Amit Sheth, Munindar Singh, and Darrell Woelk. Workflow support in Carnot. *Bulletin of the IEEE Technical Committee on Data Engineering*, 16(2):33–36, June 1993.

[30] Christine Tomlinson, Philip E. Cannata, Greg Meredith, and Darrell Woelk. The extensible services switch in Carnot. *IEEE Parallel and Distributed Technology*, pages 16–20, May 1993.

[31] Christine Tomlinson, Greg Lavender, Greg Meredith, Darrell Woelk, and Philip E. Cannata. The Carnot extensible services switch (ESS)—support for service execution. In Charles J. Petrie, Jr., editor, *Enterprise Integration Modeling*, pages 493–502. MIT Press, 1992.

[32] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, March 1992.

[33] Darrell Woelk, Paul Attie, Philip E. Cannata, Greg Meredith, Munindar P. Singh, and Christine Tomlinson. Task scheduling using intertask dependencies in Carnot. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 491–494, May 1993. Industrial track paper.

[34] Darrell Woelk, Philip Cannata, Michael Huhns, Nigel Jacobs, Tomasz Ksiezyk, Greg Lavender, Greg Meredith, Kayliang Ong, Wei-Min Shen, Munindar Singh, and Christine Tomlinson. Carnot prototype. In *[5]*, chapter 18. 1996.

[35] Darrell Woelk, Wei-Min Shen, Michael N. Huhns, and Philip E. Cannata. Model driven enterprise information management in Carnot. In Charles J. Petrie, Jr., editor, *Enterprise Integration Modeling*, pages 300–309. MIT Press, 1992.

[36] Darrell Woelk and Christine Tomlinson. Carnot and InfoSleuth: Database technology and the world wide web. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, May 1995. Industrial track paper.