The Conceptual Basis for Mediation Services

Gio Wiederhold and Michael Genesereth
Stanford University
May 1996, revised October 1996 and September 1997

---

mail to Issue editor

Prof. Dr. Michael Papazoglou
Director INFOLAB,
Tilburg University
INFOLAB, Room B304
P.O. Box 90153
5000 LE Tilburg
The Netherlands
E-mail : mikep@kub.nl
Telephone: +31 13 466-2349 , Telefax : +31 13 466-3069

Attachments:
8 figures (color)
Biographical sketches of authors
Black-and-white glossy photographs

The Conceptual Basis for Mediation Services

Gio Wiederhold and Michael Genesereth
Stanford University

Abstract

Mediator modules comprise a layer of intelligent middleware services in information systems, linking data resources and application programs. Earlier programs that led to the concept of mediation were either constructed to support specific applications or provided extended services from databases. Intelligent mediators are being built now by careful domain knowledge acquisition and hand crafting the required code.

In this paper we present the conceptual underpinning for automating the mediation process. Automation does not extend to fully automatic code generation, since additional knowledge is necessary to provide added value. The generation concept is based on the extraction of a hierarchical domain model out of the general network representing the available resources. Associated with the method are domain ontologies. Ontologies list the terms used by the models, and document their relationships. These terms provide the semantic foundation needed to perform the generation.

This paper is prescriptive, rather than a research report. The objective of publishing it now is to gain conceptual coherence among the projects that have adopted a multi-layer information architecture, while any implementations will differ greatly. Eventually having a limited number of interface standards will be essential, but these require a common base of understanding and technology.

1. Introduction

As information systems increase in scope, they depend to a greater extent on diverse, heterogeneous resources, as databases, knowledge bases, bibliographic files, web-based information, and computational facilities. These resources are typically developed and maintained autonomously. While their immediate applications tend to be inventory control, payroll, production control and the like, eventually the data become important to support high-level application, as planning and decision-making. Decision support applications are typically designed subsequently and independently. Planning support is synchronized with management objectives, and has to rely on existing sources, since it is rare that sufficient time is available to build planning systems and their data collection from scratch. Other sources of information for customer applications are digital libraries, geographic information systems, and simulations.

Dealing with many, diverse, and heterogeneous sources overwhelms high-level applications with excessive emphasis on irrelevant, but crucial details. Platform and operating system differences are commonly recognized, and are getting easier to manage. Differences in interface protocols, data description, level of abstraction, and the precise meaning of terms being used do not yet yield well to automation. Mediators provide relevant intermediary services for these problems, linking data resources and application programs [1]. Their function is to provide integrated information, without the need to integrate the base data resources. Specifically, the tasks required to carry out these functions are comprised of

1 Accessing and retrieving relevant data from multiple heterogeneous resources
2 Abstracting and transforming retrieved data into a common representation and semantics
3 Integrating the homogenized data according to matching keys
4 Reducing the integrated data by abstraction to increase the information density in the result to be transmitted.

as shown in Figure 1. Overall, this processing adds value by converting data to information. More detail is provided in Table 1, and some the services are described in more detail in the final section.

Figure 1 goes here

Figure 1. (#1 in paper) Transforming Data to Information, from Data and Simlation Resources to Decision-makers

One should note that in mediation effort is devoted to processing the results of the retrievals in addition to facilitating access, i.e., locating and gathering the data. Those access tasks are the current emphasis of much work in information systems, and are a necessary prerequisite to deal with distributed information, but cannot be the end for supporting decision-making. Well known examples of approaches to improve access are systems as KNOBOTS and the variety of WEBCRAWLERS[2], as well as multi-database systems[3].

Earlier programs that exhibited such functions and led to the concept of mediation were either constructed to support specific applications, or extended services from databases. A mediated architecture is conceptually comprised of three layers, as shown in Figure 2.

1 Customer applications, that require relevant information
2 Mediating modules, incorporating value added processing
3 Base resources, as databases and simulations

Mediators are being built now by careful domain knowledge acquisition and hand crafting the required code. Building mediators by hand is essential to validate the concept and establish the needed standards for the interfaces. Between the three layers there will be two major interfaces:

2 ⇔ 1 Mediators to applications
3 ⇔ 2 Base resources to mediation

In practice there will also be intermediate interfaces, since it is likely that within the mediation layer some sublayers exist as well. For intermediate interfaces any technology selected for the mediator-to-application interface is appropriate.

Figure 2 goes here

Figure 2. (#3 in paper) The Mediation Layer as a Central Trapezoid

We derive the shape of the mediator symbol from its intermediate role in the information processing hierarchy shown in Figure 2.

1.1 Architecture and its Support

Mediation is primarily an architectural concept. The precise implementation of a mediating module is less important than its ability to perform its function in a larger systems context. In *real* architecture the buildings can have big doors, small doors, wooden doors, doors with locks, etc. But all doors have to fit into the openings that are provided in the building. Using non-standard door openings increases the cost, reduces the flexibility to chose among types of doors, and will hinder moving large equipment in and out of rooms. Much of the effort in moving to sharable mediator architecture involves the recognition of interface standards, so that configurations, i.e., instances of the architecture, can be rapidly assembled. There is a strong linkage here to concepts as virtual enterprises [4]; mediation provides the required openess in the architecture.

For the base interface the many tools that are becoming available to serve the two-layer server-client model are appropriate, as distributed and augmented SQL and the interface tools for object-oriented access, as CORBA. At the application layer the interfaces need greater capabilities. Here agent languages, as the Knowledge Interchange Formalism (KIF) [5], for content and the Knowledge Query and Manipulation Language (KQML) for managing the needed multi-user, multi-representation, and multi-function operations are appropriate [6]. While KIF is defined to exchange knowledge in a first-order logic representation of rules, KQML can also handle other representations, as engineering objects represented in STEP (The ISO 10303 standard for Product Data Excange), mathematical equations, and even SQL tuples. Required is, of course, that the sender and the receiver agree on the chosen representation, as specified in the preamble of a KQML statement [7]. The sender and the receiver should also agree on the vocabulary and its structure, i.e., the *ontology*, specified in the preamble [8], to avoid terminological errors. We cannot expect that large-scale information system will have homogenous ontologies. A role of the mediator is to provide the knowledge that allows interoperation among semantically distinct ontologies by stitching them together along the needed seams. The architectural aspect of semantic interoperation are addressed in Section 8 of this paper.

Mediation is simplified by delegating the complexities of the customer interface to the application program. Code to deal with the variety of graphical user interface (GUI) devices, as windows, pop-up, roll-down, scroll, cut-and-paste, drag-and-drop, speech, etc. often occupies more than 70% of application programs. Mediators and the invoking applications only need a machine-friendly interface bewtween them, as represented by a KQML application program interface (API). A GUI interface will be needed to interact with the owner of the mediator for mediator maintenance.

Trading internal code for an API recapitulates an earlier paradigm shift. We recall that before database management systems existed, the programming of file operations took a major amount of effort and competence. The acceptance of the database paradigm removed that code from applications and delegated it to database management systems (DBMS). The DBMSs now provides all the really difficult and specialized code associated with managing files, as backup, recovery, integrity, and internal consistency management. The DBMS-based resource and its application share a model, as represented in the schema, and this concept will be a basis for the mediation paradigm presented here.

2. Services

Mediators provide information services to customer applications. In this paper we present the conceptual underpinning for automating the creation of mediator software. Facilitation provides assistance in locating and understanding resource capabilities. Automation in mediator construction focuses on matching identified data resources to customer needs. Human intervention is expected when creating mediators, since knowledge is necessary to provide added value to the customer. This value offsets the costs of having the additional architectural layer implied by formal mediation. In an earlier paper we presented some of the tools from the field of artificial intelligence that are of help in mediation [9]. At that time the architectural vision presented in this paper had not yet been formulated.

A range of generic value-added services is listed in Table 1. Greater benefits accrue when services are specialized for domains as finance, logistics, activity scheduling, etc., but the application scope will be narrower. Such services are now available in some extended resources or associated with existing applications, but are rarely identified as composable and reusable tasks. Many more citations of service examples suitable for mediators are possible, since improving information for customers is pervasive. Placing services into mediators within this architecture enables their reuse, and partitions information processing tasks by functionality and domain.

2.1 Value

To warrant implementation of a service as a distinct module there must be sufficiently much added value to overcome the cost of adding a layer and its interfaces in the information processing flow. But the benefits and costs to be considered are only partially related to performance. Having identifiable and maintainable service modules provides significant long-term management benefits. Today many free services are available on the Internet, but we see already the limitations when free also absolves providers of responsibility to maintain quality. In the long run many of these services are best provided by independent enterprises over the networks, or their programs can be leased to provide these services at customer sites, as detailed for digital libraries [10]. The cost of maintaining such services must always be offset by their value [11].

Value-added services in a mediator include combinations of:

1 Rapid determination of likely resources using indexes extracted earlier
2 Invocation of wrappers to deal with legacy sources
3 Selection of likely relevant source material
4 Optimization of access strategies to provide small response times or low cost
5 Imposition of security filters to guard private data
6 Resolution of domain terminology and ontology differences.
7 Resolution of scope mismatches
8 Interpolation or extrapolation to match differences in temporal data
9 Reduction of historical data to limited snapshots
10 Abstraction to bring material to matching levels of granularity for integration
11 Integration of material from diverse source domains based on join keys
12 Omission of replicated information
13 Assessment of quality of material from diverse sources
14 Pruning of data ranked low in quality or relevance
15 Omission of information already known from the customer model

16	Statistical summarization into higher level objects, as defined in the customer model
17	Relaxation of search terms to satisfy query expectations
18	Reporting deviations from expected values or trends
19	Triggering of actions due to exceptions from expected values or trends
20	Transformation of material to make presentation effective for the customer
21	Adaptation to the bandwidth and media capabilities of the customer
22	Sending the information and meta-information to the customer application

Table 1: Services to be provided in mediators.

2.2 Assigning responsibilities

When these tasks are performed in a two-layer, client-server, architecture they are either built into customer applications or database services. When built into applications it becomes difficult for their maintainers to keep up with the variety and change in resources that occur over time and over customer needs. For instance, when data reduction is bound to the databases, the abstractions are often not those that are most relevant to the customer, and integration from multiple sources is unlikely.

A service is associated with authority and responsibility in a defined domain. To circumscribe domains semantically we consider ontologies, since services without consistent ontologies will be incomprehensible to the customer. Service ontologies are unlikely to match available resources. For instance, database resources are modeled only in part by their schemas.

Models

The mediation paradigm depends on models; models of the resources and models of the customer needs. Since models can have a wide variety of flavors, this section focuses on making the model requirements for mediation more specific. Three well-accepted concepts combine to create the resource model:

- Schema integration
- Connecting schema entities with relationships
- Reducing the volume of required knowledge by using views.

Each of these concepts needs some formalization, so that mediation and the generation of mediating programs can become reliable and predictable.

3. Resource Models

Each resource can be described by its own model. The modeling technologies differ greatly, and their integration is a challenge for mediation.

3.1 Relations and their structure

In a relational database the schema describes the base classes of the model. During the database design phase an Entity-Relationship model may have been used. We use our particular form of that model, the *structural model*, which uses relations and connections among them. The connections also have algebraic properties, which can support the transformations that enable integration [12]. Figure 3 presents a simple model of a health care setting. In practice, even in a single hospital the entities depicted will reside in multiple databases. Certainly, if the model is extended to cover multiple clinics, and internal and external laboratories, there will be a distributed operation, and some of the resources will actually be autonomous and heterogeneous. Connections will, of course, extend over distributed resources as well [13].

Relational schema integration and evolution is a well developed sub-specialty in the database arena, and we will here benefit by modeling an integrated schema without trying to integrate the actual databases [14].

Figure 3 goes here

Figure 3. (#4 in paper) Networked Data Resources in a Health Care Setting

3.2 Object-structured Data

Object technology enables applications to use an infrastructure which aggregates detail into meaningful units in many important domains. When the underlying resources are modeled as objects, then some of the connections, the ones within the objects, are encapsulated, so that they are hidden from the customer. Internally, objects have explicit hierarchical linkages since the class definitions that control them are based on hierarchies. The use of multiple ownership connections, implying multiple inheritance within an object, is rare. For encapsulated objects those linkages are not directly accessible or transformable, although it is likely that methods will exist within the objects that allow fetching of all implied data. An interface language, as IDL for OMG CORBA or ISL for Xerox ILU provides access to the descriptive meta-data.

Since the granularity of objects is limited there is still a considerable scope for modeling referencing connections among objects. Figure 3 presents objects of fairly large granularity. Outside of the objects one finds the more arbitrary, i.e., non-hierarchical and multiple ownership linkages.

Some large-scale hierarchies are being constructed in the object world with great pain and compromise. We don't believe that very large hierarchical objects are viable since conflicts of viewpoints are bound to arise. A reasonable upper limit for an object class seems to be at a 200 element granularity. The assumption that one hierarchical viewpoint is right for all occasions is demonstrably false [15]. Forcing unsuitable representations onto processing programs increase cost of finding and executing solutions; it is well known in mathematics that finding the right representation for a problem is 80% of the effort, and the same holds true in computing. However, computers can transform general, well-defined representations into specific ones.

3.3 Text

Textual data is also an important information resource. We find text in messages, reports, papers, and books. Much of its processing has been distinct from database activities in the past. Now linking through texts using hyper-text and World-wide-web technologies enable shared processing. The linkages, and hence information integration, can join structured data to text. Furthermore, such links can point into multi-media objects, as images, video clips and speech segments, and multi-media objects can in turn have references to other objects. Such inter- and intra-document linkages are more ad-hoc than the structured connections seen in databases. Since these linkages cannot be expressed by a general model, some of the operations described later in this section will have to be deferred to execution time, and performed on the instances, rather than performed on the model. Since at this time we envisage model matching to be interpretive, the gap is not as wide as it would be if traditional data-processing and information systems had to be integrated.

3.4 The aggregate resource model

Taking all of these resources together, we find that the aggregated model is a general network, and could be represented as a hypergraph. Conceptual linkages or *connections* characterize the abstract resource model and actual or inferable linkages exist among its instances, the actual resources. Even when contributing resources and their ontologies were modeled by simple hierarchies, they contribute to an integrated network. The structure is symbolically depicted as the hexagon in Figure 4.

4. Customer models

In general, the concept of having a user model is daunting. We will be more specific and speak of a customer model, since our information systems have many flavors of users [10]. Such a customer model focuses on a task set and a domain of interest. The same user may assume a different customer role at other times, and is then represented by a distinct model. Furthermore, we limit ourselves to concepts that can be machine-processable, since the customer's actions are always transmitted via the customer's workstation to the mediating service. We do not look into the dark recesses of the human mind.

We now have a crucial hypothesis:

Customer models are hierarchical.

Unfortunately, it is impossible to prove this hypothesis formally. All instances of user models cannot be enumerated, and if we define customer models formally, it would create a tautology in our proof. Resorting to informal reasoning we consider that people, when faced with complex tasks, categorize the processes and objects to be dealt with, so that they can apply a divide-and-conquer paradigm. Good categorizations are taxonomies with two attributes: disjointness, i.e., no object belongs to more than one category, and completeness, i.e., all objects can be classified.

The widespread acceptance by customers of object models, which are also hierarchical in nature, argues for the hypotheses that customer models can be hierarchical, and hence manageable within this paradigm. Once the customer is comfortable with a taxonomy, processing provided by the supporting mediators, as inheritance, selection, and aggregation, should also follow the same model. A system where the customer's taxonomy matches the provided services will display deep 'user-friendliness'.

While we can create systems of objects with multiple inheritance, say a person being both in the profession category of accountant and in the sports category of skydiver, these categorizations belong in our approach to distinct customer models. We support this distinction below by introducing domains. Having a hierarchical customer model driving a mediation process does not inhibit user applications from integrating results from multiple mediators, and hence accommodating incompatible domains. Such high-level integrations tend to be pragmatic, since it is difficult to apply formal comparative metrics among such incompatible domains. For instance, two aspects of an sports figure, competence in sports, aggregated from scores and games, and pay scale, aggregated from base pay, bonuses, and endorsement income, can be pragmatically compared and analyzed, but a formal comparison is not justifiable. Figure 7 denotes the two levels of integration; in this section we focus on the methods within the mediators.

The need for conceptual clarity also supports the use of hierarchies. In mediators simplicity is essential. This rule holds, of course, for any software you want to work reliably. A mediating module carries out tasks to serve a customer, and a desirable aspect of such a computer-based agent is that it has a model which permits the customer to understand its capabilities. Such a customer is initially the programmer building the application. If the application requires flexibility, a mediator should also make its internal model available, so that its capabilities can be fully exploited. If unavailable capabilities are needed, the customer will have to negotiate with the service provider, a common situation today when dealing with databases.

In database technology, a view relation, defined by a single view expression, is also essentially a hierarchy. Although the result will be represented as a table, each join in the view expression has defined a relationship. The attribute named by `WHERE` clause of a join along a relationship defines the superior element. In general, a view relation is not in normal form. Any subsequent normalization will expose its hierarchical structure. Database views have been deemed adequate for major database applications, further bolstering our hypothesis.

Figure 4 goes here

Figure 4. (#5 in paper) A Customer Hierarchy in a Resource Network

5. Model Matching

The principal tasks in mediation are the identification of relevant resources for the customer model and the subsequent retrieval and reduction of the relevant data for a customer inquiry. We focus here on the initial process, matching the customer needs to the resources.

Given the resource network and the customer-model hierarchy, the matching process is based on mapping of the customer's model hierarchy onto the integrated resource network model. Since the mapping is part of the design process, the steps can be performed manually, by the database or view designer and the eventual owner of the mediator.

As the design information, i.e., the models and the ontologies that define the terms and their structures is being committed to machine-processable representations some automation can ensue. In Section 5.3 we present work on facilitators, which automate resource identification and adaptation. Major benefits of automation can accrue in long-term maintenance, enabled by the adaptability of mediation [16]. The creation of new functionality in mediation is a creative effort, and is effectively performed by programmers or knowledge-engineers with high-level tools. Figure 4 shows the concepts of a customer hierarchy in a resource network.

5.1 Mapping the hierarchy

Matching of a customer model to the resource model requires that the terminology used to describe objects and their attributes matches. Limiting mediation to closed-world domains is a precondition. Interoperation among domains is addressed in Section 8. The following steps are needed to satisfy the customer model in respect to the available resources:

1 Locating the pivotal linkage: the root node of the customer's hierarchical model has to be found in the general resource network. If we assume that the root node identifies an object class, then the search can be restricted to entities of the resource model. Say that the problem is a broken engine, and the pivot is the topic term 'engine'. Many instances of the term will exist in the network, it should be narrowed to an intersection of, say, 'truck' and 'repair'. Joint processing requires that the

domain ontologies match, otherwise there is a chance that an object in one model has been characterized as an attribute in another model. For instance, a six-cylinder engine can be an attribute of car in a detailed automotive catalog resource, but may be an object in the garage model. When matching attributes are located, a transformation of the resource representation has to be initiated. These transforms have been considered when integrating databases, but not, as far as we know, applied dynamically. If any match of the root fails, objects lower in the customer hierarchy should be searched for, and the root becomes an object for aggregation only.

2 Now relationships in the resource must be matched to those emanating from the root pivot of the customer object. There are again potential mismatches in semantics. Relationship representations can be value- or reference- based. The customer's model may be simpler than the resource model, requiring the match process to combine intermediate linkages. For instance, the tire of the customer's truck can be linked to the wheel of the truck, since there is a dependency if various types of wheels are available. But the customer's model may not have considered this choice.

∗ Now further entities and relationships can be matched until the leaf nodes of the customer's model are reached.

+ Entities and relationships beyond the customer's model can be collected, until the leaf nodes of the resource model are reached. These nodes may not be reported in the result, but contain data for aggregation into the customer's model.

Unmatched links and objects of the customer's model will be reported to the builder of the mediator, who must decide how to deal with the problem. If more resources are legitimately needed to satisfy the customer, then they must be obtained from other resources and the internal resource model correspondingly be augmented. In practice, resource cost or accessibility makes the matching of all of an ideal customer model pragmatically infeasible. At other times the customer's model is out of scope for the service domain. In both cases the application has to be informed by the mediator of the mismatch of expectation to available resources.

5.2 Automation

Automation of the process of building mediators means developing tools that support the mapping and the creation of physical representations to perform the tasks. Search engines can help in locating resources. Obtaining their ontologies facilitates formalization of their models [8]. Object structures help in organizing their components and dealing with their instances [17]. Identifying intersections of model ontologies provides focus to integration. Presentation tools that deal with large objects can help in delivery to the customer. Knowledge-base management tools will help the maintainer of a mediator in keeping the services up-to-date. We hence envisage today primarily a toolbox approach to autoamtion. How far automation can proceed eventually depends on the acceptance of common architectures and representations, [6] since the task of automation appears to be too large for any single research effort.

5.3 Improving the Mapping

Matching the customer's objects to the resource objects, like any design process, may require some iterations. A customer or domain specialist may have ideas on substitute resources. When dealing with structured databases the breadth of the resources is typically well defined, since their schemas establish a closed world [17]. To determine their scope, i.e., their coverage of object instances requires again analysis or expertise. For instance, nowhere was it documented that a certain Navy database about ships we used did not include fishing boats. In many contexts that omission does not matter, but when trying to stem smuggling it does.

When dealing with unstructured information sources, as browsing through bibliographies in the world-wide-web, no schema exists. No clear definition of scope is likely and resources must be selected based on content analysis [19]. The most likely sources are selected, but no guarantee can be given that relevant instances will be found at query time.

The instantiated customer's model may also become too large. Potential resources found may actually be redundant. The designer of the mediator has the option to omit resources, or add conditions on their use. A control rule, for instance, can state that only if actual retrieval from a primary source is incomplete, will the secondary source be consulted [20]. Some level of detail may not be needed. If some summaries exist already in the resources they will not have to be recomputed.

5.4 Facilitation

Automation of mediating tasks is obviously desirable, even though lack of automation does not invalidate the concepts. Facilitation provides automation, primarily in resource identification and data format conversion [21].

The vision underlying facilitators is one in which any system (software or hardware) can interoperate with any other system, without the intervention of human users or their programmers. This level of automation depends on having very thorough ontologies to describe the resources, since now no human is interposed between the customer's application and the resources. Human input will still define the ontologies of the resources, but the task of matching now occurs at query processing time.

By definition, facilitators are required to accept runtime submissions of

1 meta-data about information resources
2 logical statements relating disparate concepts (usually definitions)
3 format descriptions for the data items being accessed

Facilitators are expected to use this meta information in converting, translating, or routing of data and information.

For example, when a new resource becomes available, the facilitator will expect a machine-processable description of the new resource, which must be coherent with the existing description used by the facilitator. Then the facilitator can integrate the new resource, or elide an existing one, and the consumer can be immediately served with the new resource.

For the same situation the party responsible for a mediator, its owner, is to be informed of the new resource and its capabilities. If there is facilitator between the resouced the mediator, the facilitor can issue the notification. The knowledge that defines the value-added function in the mediator has to be augmented to include the new capability. The augmentation is best done by the owner, perhaps aided by an automated process if the semantic difference of current and new resource capabilities is modest. A new version of the mediator, with augmented capabilities is then made available. Existing applications will be informed, but the owner also has a responsibility to continue serving current applications, with the results that remain compatible.

For added value-services a facilitator may call upon a mediator as a resource. Then the required meta-information is provided by the owner of the mediator, moving the hard task of matching ontologies one layer away from the base data, potentially simplifying adoption of facilitators in the architecture. To achieve this level of interoperation, compatible interface languages must be employed [5]. Both mediators and facilitators can use wrappers to access non-conforming resources.

Figure 5 goes here

Figure 5. (#7 in paper). Systems with Facilitators and Mediators

In summary, a facilitator can be viewed as an instant mediator to the extent that automation allows. A mediator, in turn, can be viewed as a petrified facilitator, requiring a human mason to restructure its role in a system configuration when resources, customer requirements, or the mediating knowledge changes.

6. Mediation Services

A major task for an effective mediation service is the reduction of data volume to be shipped to users' applications, while maintaining its information content. More information embedded in less data increases the information density. Having a high information density deals with the complaint that customers voice now: that there is *information overload.* Reducing data transmission volume to the customer's workstation also reduces communication delays and costs. The principal tool for data reduction is abstraction, either by summarization or by exception seeking. Both functions depend on having a simple, hierarchical model of the customer's needs. We now discuss how the simple model can be exploited.

6.1 Summarization

Summarization provides aggregated data, classified along the dimension specified by the customer. Summarization is a common task now mainly performed by people who support high-level decision-makers. Today, they will, of course, process the data with the help of computer programs, but that involves always explicit programming of their view of the customer's model. For example, cost data collected in a factory are at a level of detail which records the activity of every worker and every machine with respect to every task. For the payroll domain the worker's efforts are aggregated to daily hours, and then processed with data which determine overtime rates, and then further aggregated to weekly totals, which determine pay checks. At the pay level benefits are added, taxes are computed, and contributions are withheld.

The same source data from the factory floor can be aggregated according to a different customer model to arrive at costs per product, and to this aggregation the allocation of product development costs is added to arrive at the base costs which eventually can determine sales prices and profits.

Other examples are summarization of medical records, of sales trends by product-type, of population dynamics etc. The final presentation is often graphical [22], but aggregations are performed prior to the graphical presentation [23].

6.2 Exception seeking

An alternate, even more effective abstraction is just seeking and reporting exceptions. Here only results that differ significantly from the customer's expectation are presented, for instance, abnormal clinical findings or an unexpected drop in sales for a product line. The need for a customer model is obvious here. A change in a patient's weight by 10% over a short time is typically a cause for concern; putting absolute limits on patients' weights would lead to useless exceptions, even if patients were categorized by age, height, and gender.

Many business decisions are motivated by changes in customer demand. Simple tabulations do not tell the full story. Sale amounts are affected by exchange rates and promotions. Factory sales are buffered by inventories. Many products are affected by the weather and regional preferences. Only when these have been taken into account by specialists is it useful to use the information for production planning and investment decisions.

6.3 Caching

Caching stores subsets of the persistent data inside a mediator. A large cache, like warehouse middleware, improves performance, but it is unwise to augment mediators with responsibilities for persistent storage. Since not all resources maintain historical data a mediator can cache past data to enable summarization over time [24]. Predicting cache requirements implies an understanding about the levels and scope of summaries and the need for temporal validity. Such meta-data is rarely available today. Much ongoing research in data-warehousing focuses on the technology of retention and summary use and maintenance. We can expect useful results from warehousing projects that will leverage customer needs encoded within mediators.

While processing routines for deriving historical information on trends and evolution from data are common, their invocation today requires human interaction [25]. Dealing automatically with the heterogeneity of temporal representations requires adoption of standards and adequate algebras.

6.4 Guiding the processing flow

As a byproduct of matching the models, sequences of high-level operations can be generated, similar to automatic code generation during proof procedures [26], [27]. The operations identified during model matching are collected for subsequent elaboration. The functional granularity and completeness will differ substantially from the codes created in automatic programming research. Rather than lines of code, model and pattern matching will generate invocations to standard package interfaces. The inputs and results for these invocations are attached to the model.

Processing packages have to be linked to those invocations to perform the intended functions over the data. These packages will be selected by the designer of the mediator to carry out the function that the mediator is to perform, since only the framework of the information flow can be generated during model matching, and not the internal code within the functions. This level of added-value mediation cannot be created automatically until we have programming languages at a much higher level of abstraction. Abstraction for processing paradigms is much more elusive than abstraction over information models [28].

In any case, in mediation full automation is less crucial than having reliable guidelines for the structuring of the mediating program, since we do not see removing human participation from the creation process, but rather providing supporting consistent access to the proper data and proper paths for the results.

7. Maintenance

Mediation adds value to the data by applying the knowledge of the expert who has created the mediator. Mediators should also be maintained by those experts, so that the mediators remain effective in a constantly changing world. As soon as an improved mediator is developed it can be advertised over the network, both to existing subscribers as well as to potential new clients. A poorly maintained mediator will lose value over time and become a candidate for replacement by a competitor. A responsive mediator maintainer will create versions as new or existing customers develop new needs. Other customers can continue to use the old mediator version, and not be disturbed until they decide that their application needs the upgrade. This flexibility is crucial, since with versioned mediators upgrades are not constrained by the effect on the existing community of customers. The maintainer will, of course, try to keep the number of versions of a mediator service modest. The prices charged for services provided by obsolescent mediators may be increased, to encourage applications that depend on old versions to upgrade.

Automation is likely to play an important role in long-term maintenance. When changes are needed in the customer model or the resource model, the ability to re-execute the matching process, while invoking the same processing functions that defined the role of a mediator can rapidly regenerate up-to-date services. Today, maintenance of software occupies the dominant fraction of programming resources of many enterprises. Without tools to provide some automation the situation is likely to be worse in large-scale heterogeneous information systems because

1 Information should be novel, requiring constant source updates

2 Customers' needs legitimately change as they learn more about the domain they are exploring

2 Autonomous resources cannot afford to stay stable to make life easy for existing customers. They must advance in order to serve new customers with better, perhaps more detailed, data.

Failure to adapt to these changes will rapidly reduce the value of information systems. A small amount of formalization and automation of the process of programming the value-added services in mediators is likely to have a large leverage in keeping maintenace costs tolerable and benefits high. Support tools and modules are needed to build the required wrappers and transformations for mediators.

Enabling the customer to alter the customer model as needed can keep the information system useful and viable. A prerequisite for customer interaction is a clear presentation of the current customer model. To indicate what changes can be accommodated requires browsing in the resource model and discovering what alternative attributes and categories are available

7.1 Facilitation versus Mediator Generation

Notice that tools to automatically generate mediators are not facilitators, because they still require human interaction and assumption of responsibility. Any such interaction also disables true dynamic interaction of consumers and resources. Version maintenance is also not an issue in facilitation. Programs capable of generating mediators, routers, translators, or wrappers need formal specifications. In some cases, these generators may work automatically, in some cases interactively with humans.

8. Domain Partitioning

We present mediation as the principal means to resolve problems of semantic interoperation. However, mediation will be needed in many topics, and we cannot expect that a single, general mediator can cover all topics of interest to any application. We can expect even less that a single group of individuals can develop and maintain such a general mediator. We also expect that most applications need to combine more than one topic, and hence, to need support from multiple mediators, as indicated in Figure 6. Different applications will use different configurations of mediators. For instance, a production planner needs production cost estimates and product demand information. The sales manager needs the demand information, perhaps at a lower level of granularity, and inventory data.

Figure 6 goes here

Figure 6. (#8 in paper) Resource, Mediators, and Applications in the Architecture

We hence expect to have many mediators, each focusing on their own domain. A domain should be limited by its ability to maintain internal consistency. The consistency requirement means that the owners and maintainers have identical semantics for all terms. This requirement translates to a number of technical constraints. For instance, it is best if the terms used within a domain can be formally enumerated, so that we can be specific about the scope of a domain.

The issues of domain specificity and interoperation have been addressed in [11]. We have also proposed a knowledge-based algebra to enable formalization and, in due time, optimization of interactions among domains. Domain technology inherits many of its concepts from database view models. Notice that views also help in making large resources understandable to the user and enable optimization of access.

Figure 7 goes here

Figure 7. (#6 in paper) Integration at Two Levels

9. Status

No system exists today that performs the set of tasks envisaged above. However, we do have partial examples. The PENGUIN system creates hierarchical objects classes from a structural database network model [29] and is now used to support SunSoft's open systems DCE. We cited examples of code generation earlier. If that technology can be applied to mediator generation, a considerable increase in scale and significance of that technology may ensue. Facilitation employs local ontologies, because few resources provide them today.

Since we assume that most mediation services are performed in autonomous computing nodes, a requirement for interoperation is the ability to communicate according to some standard conventions [5]. To enable a greater variety of communication actions, participants, and representations the KQML approach has been developed [30]. Automatic access procedures to heterogeneous resources are created in CARNOT, and the results are automatically joined at their *articulation* points, where labels match, although not reduced [31]. A hierarchical model of terms was used to automatically control the volume of topic citations to be returned in support of the tasks of an editor searching for referees [32].

A number of applications have been developed using mediator technology [33]. Current applications have focused on manufacturing, where design and prototype production data have been combined. This environment is being extended with services to handle engineering change management. A specific application has been the selection and validation of gimbals for antenna positioning on spacecraft at Lockheed-Martin Space Systems. The gathering and integration of military intelligence data is an obvious application. Other areas being developed now are in healthcare management and plant safety and environmental cleanup. An interesting project is in the collection and integration of satellite data for land-use planning.

The implementation of mediators varies greatly. If knowledge-based processing is crucial, many mediators are programmed in languages as LISP. If optimization is crucial to processing, the mediators may depend on packages written in FORTRAN. Maintenance would be enhanced by using declarative approaches that could be understood and modified by end-users. However, most current mediators have been coded in the C and C++ languages. Object concepts are being generalized to improve the granularity of the representation of information [17]. Most communication is supported by commercial standards, as CORBA, although within CORBA objects information as defined to facilitate semantic interoperation as specified in KQML may be wrapped. HTML and JAVA are becoming increasingly important for delivery to customers, although they do not provide convenient multi-domain integration at the application level.

Figure 13 goes here

Figure 8. (#9 in paper) Fat versus thin mediators

A number of contractors now have the capability to rapidly build the required application interfaces and implement the architecture. The number of platforms and languages varies, and there is some discussion on **fat** versus `thin` mediators. To minimize the divergence of the mediator technology and its tools there ARPA has sponsored workshops to develop a shared reference architecture. The shared architecture document for the effort is currently being maintained at George Mason University.

10. Conclusion

Information technology is serving us well in specific domains, although we have remained dependent on specialist model designers and programmers for the implementation. Mediation is an architecture intended promote reuse and scalability, so that sources from many domains can contribute services and information to the end-user applications. The layered structure adopts for information structuring the partitioned domain management strategy used by the INTERNET distributed naming conventions [34]. Software as well as artificial intelligence technologies have been hard to scale when domains grew large or became diverse.

We presented a concept for generation of mediators which is based on the extraction of a hierarchical domain model out the general network. The result provides an object model for the using application, and hence a much simpler world-view then is represented by the underlying, heterogeneous resources. Mediators represent responsible, predictable and stable services. To warrant the use of mediators, there should be significant value-added processing.

Data reduction, exception search, dealing with uncertainty among heterogeneous resources, and ranking of results are examples. The owner of the mediator assumes responsibility for the correctness of such processing. Facilitators provide automation and rapid response to changing situation. Automation depends on consistency in the ontologies which describe resource capabilities and customer needs.

A partitioning into domains creates a desirable autonomy to reduce the cost of maintenance. The focus on maintenance distinguishes the mediated approach from many other proposals, which attempt to design optimal systems. In large systems the major costs are in integration and maintenance, rather than in achieving initial functionality and optimality. Integration in mediation can proceed at multiple levels of abstraction, avoiding the centralization that hinders progress in data exploitation of data from diverse sources.

Dealing with the intersection among ontologies has not been widely explored. Work in the Carnot project, using the large CyC knowledge base, has recognized the existence of articulation points where resources must be linked [30]. We are working on formalizing the management of such intersections, as well as unions and intersection operations and expect to be able to report on early results soon.

Mediators and facilitators inhabit the central layer of a three-layer architecture. Facilitators and mediators can interact with each other and resources within an information system.

From the primary distinction made in Section 5.3 we can derive the following strengths and weaknesses for mediation and facilitation in the main areas addressed by the DARPA Intelligent Integration of Information (I3) program

function	Mediators	Facilitators
Control and Routing	determined by human design	dynamic and responsive
Content Manipulation	cricial to add value	per rule technology
Format Conversion	via wrapper invocation	dynamic, per descriptions
Run-time flexibility	no model changes	high
Efficiency	high with human involvement	requires more meta-information
Mutual usage	facilitation in design phase	mediators for services

Table 2. Distinction of Mediators and Facilitators.

Tools are needed to support the development and maintenance of mediators. To have effective tools a common formal paradigm is needed. We cited some early tools used to support the initial mediation projects. The emerging standards have been made public on the networks, to avoid proprietary dominance. Rapid development is possible by exploiting the same modern networks that make the architecture itself feasible. With network access to standards and the potential consumers, whoever build the best tools or provides the best services will gain the deserved advantage.

Acknowledgment
This research direction would not have been feasible without the support of many colleagues in the DARPA Intelligent Integration of Information (I3) program, and, of course, from DARPA itself. Recent workshops sponsored by Bob Neches and David Gunning of DARPA, and chaired by Bill Mark, now at National Semiconductor, Tom Gruber, now at IntraSpec, and others have contributed valuable ideas. This paper is based on a presentation made May 1995 at the COOPIS meeting in Vienna, Austria and benefited from feedback received there. An unknown reviewer and David Maluf of Stanford University provided important input to clarify the presentation. A videotape on Mediation Technology was prepared with Stephen Cross of Carnegie-Mellon University and Charles Channell of ISX Corporation and is available from IEEE Educational Videotapes, Picataway NJ.

References
We cite here only a few of the references which have contributed to the concepts described here. Many are available on the world-wide web and can be found via our web pages starting at http://csd.stanford.edu.

[**1**] Gio Wiederhold:"Mediators in the Architecture of Future Information Systems"; *IEEE Computer*, March 1992, pp.38-49.

[**2**] Tim Berners-Lee et al.: "World-Wide Web: the Information Universe"; *Electronic Networking: Research, Applications, And Policy*, Vol.2 No.1, Spring 1992, pages 52–58.

[**3**] A.R.Hurson, M.W.Bright, S.H.Pakzad (ed.s): *Multi-database Systems: An Advanced Solution for Global Information Sharing*; IEEE Press, 1993.

[**4**] Howard Rheingold: *The Virtual Community: Homesteading on the Electronic Frontier*; Addison Wesley, 1993.

[**5**] Michael Genesereth and Steven Ketchpel: "Software Agents"; *Comm. ACM*, Vol.37 No.7, July 1994, pp.48-53,147.

[**6**] Ramesh S. Patil et al.: The DARPA Knowledge Sharing Effort: Progress Report; in Nebel, Rich and Swartout, eds., *Principles of Knowledge Representation and Reasoning*; pp.777–788, Morgan Kaufmann, 1992.

[**7**] Robert Neches et al.: "Enabling Technology for Knowledge Sharing"; *AI Magazine*, Vol.12 No.3, pp.37–56, 1993.

[**8**] Thomas R. Gruber: "A Translation Approach to Portable Ontology Specifications"; *Knowledge Acquisition*, Vol.5 No. 2, pp.199–220, 1993

[**9**]Gio Wiederhold: "The Roles of Artificial Intelligence In Information Systems"; *Journal of Intelligent Information Systems*, Vol.1 No.1, 1992, pp.35–56.

[**10**] Gio Wiederhold: "Digital Libraries, Services, and Productivity"; *Comm. of the ACM*, April 1995, pages 85-96..

[**11**] Gio Wiederhold: "Value-added Mediation in Large-Scale Information Systems"; *Proceedings of the IFIP DS-6 Conference*, Atlanta, May 1995. To appear in Meersman(ed): *Database Application Sematics*, Chapman and Hall.

[**12**] Ramez El-Masri and Gio Wiederhold: "Data Model Integration Using the Structural Model"; *Proceedings 1979 ACM SIGMOD Conference*, pages 191–202.

[**13**] Gio Wiederhold, and Xiaolei Qian: "Database Engineering"; in J.J. Marciniak: *Encyclopedia of Software Engineering*, Vol.1, Wiley, NY, pages 269-282.

[**14**] Carlo Batini, M. Lenzerini, and Shamkant B. Navathe: "A Comparative Analysis of Methodologies for Data Base Schema Integration"; *ACM Computing Surveys*, Vol.18 No.4, Dec.1986, pp 323-364.

[**15**] Gio Wiederhold: "Views, Objects, and Databases"; *IEEE Computer*; Vol.19 No.12, December 1989, pp.37-44.

[**16**] Gio Wiederhold: ""Modeling for Software System Maintenance"; in Michael P. Papazoglou (ed.): *OOER'95: Object-Oriented and Entity Relationship Modeling*; Springer Lecture Notes in Computer Science, Vol. 1021, pages 1-20.

[**17**] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom: "Object Exchange Across Heterogeneous Information Sources"; *IEEE Data Enginering Conf.*, March 1995, pages 251-260.

[**18**] R. Reiter: "On Closed World Data Bases"; in Gallaire and Minker (eds): *Logic and Data Bases*, Plenum Press, 1978.

[**19**] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic: "The Effectiveness of GLOSS for the Text Database Discovery Problem"; *Proc. of the 1994 ACM SIGMOD Conference*, ACM Sigmod Record, Vol.23 No.2, May 1994, pp.126–137.

[**20**] Shailesh Agarwal et al.: "Flexible Relation: An Approach for Integrating Data from Multiple, Possibly Inconsistent Databases"; *IEEE Data Engineering Conference*, March 1995.

[**21**] Michael Genesereth: "An Agent-Based Framework for Software Interoperability"; to appear in *AI magazine*, 1995.

[**22**] Jock Mackinlay and Michael Genesereth: 'Intelligent Presentation: The Generation Problem for User Interfaces"; *Data- and Knowledge Engineering*, Vol.1 No.1, Jun.1985, pp.17–29.

[**23**] Isabelle deZegher-Geets et al.: "Summarization and Display of On-line Medical Records"; *M.D. Computing*, Vol.5 No.3, March 1988, pages 38–46.

[**24**] Nick Roussopoulos: "An Incremental Access Method For Viewcache: Concept, Algorithm, and Cost Analysis"; *ACM Transactions on Database Systems*, Sep. 1991, Vol.16 No.3, pp.535–563.

[**25**] Abdullah U. Tansel et al.: *Temporal Databases, Theory, Design and Implementation*; Benjamin Cummins Publishing, 1993.

[**26**] D.E. Cooke and A. Gates: "On the Development of a Method to Synthesize Programs from Requirements Specifications"; *Int. J.on Software Eng. and Knowledge Eng.*, Vol.1 No.1, March 1991, pp.21–38.

[**27**] David R. Smith: "Constructing Specification Morphisms", *Journal of Symbolic Computation*, Special Issue on Automatic Programming, Vol.16 No.5-6, 1993, pp.571-606.

[**28**]Gio Wiederhold, Peter Wegner, and Stefano Ceri: "Towards Megaprogramming"; *Comm. ACM*, Vol.35 No.11, November 1992, pp.89-99.

[**29**] Thierry Barsalou et al.: "Updating Relational Databases through Object-Based Views"; *ACM SIGMOD Conf. on the Management of Data 91*, Boulder CO, May 1991, pages 248-257.

[**30**] Tim Finin et al.: "KQML as an Agent Communication Language"; *The Proceedings of the Third International Conference on Information and Knowledge Management* (CIKM'94), ACM Press, November 1994.

[**31**]Christine Collet, Michael N. Huhns, and Wei-Min Shen: "Resource Integration Using a Large Knowledge Base in CARNOT"; *IEEE Computer*, Vol.24 No.12, Dec.1991, pp.55-62.

[**32**] Surajit Chaudhuri: "Generalization and a Framework for Query Modification"; *Proc. IEEE CS Intl. Conf. on Data Engineering 6*, Feb. 1990.

[**33**]Gio Wiederhold: "Intelligent Integration of Information"; foreword for a special issue of *Journal of Intelligent Information Systems*, Vol.6 Nos.2/3, June 1996, pp.93-97..

[**34**]Robert E. Kahn: "Networks for Advanced Computing"; *Scientific American*, Vol 257 No.5; Oct.1987, pp.136-143.

Gio Wiederhold
is a professor of Computer Science at Stanford University. He also holds courtesy appointments in Medicine and Electrical Engineering. Since 1976 he has supervised 27 PhD theses in these departments. During 1991-1994 Gio was on leave performing program management at DARPA, initiating programs in Intelligent Integration of Information and Object Bases. His research focuses on the application and development of knowledge-based techniques for information systems, and on large scale software systems in general.

Wiederhold has published several books and more than 250 papers and reports on computing and medicine. He is on a number of editorial boards, focusing on electronic publication. Wiederhold received a degree in Aeronautical Engineering in Holland in 1957 and a PhD in Medical Information Science from the University of California at San Francisco in 1976. He has been elected fellow of the ACMI, the IEEE, and ACM. His home page is http://www-db.stanford.edu/people/gio.html. Gio Wiederhold can be reached at the Computer Science Department, Gates 4A, Stanford 94305-9040 or as Gio@cs.stanford.edu.

Mike Genesereth
is a professor of Computer Science at Stanford University. Professor Genesereth's research centers on the study of "logical systems," i.e. computer systems able to process information in the form of equations, constraints, negations, disjunction, rules, and so forth. Research topics include knowledge representation, automated reasoning, agent architecture and collaboration. Applications include automated design and control of autonomous systems (such as robots, automated factories, and autonomous vehicles) and automated interoperation among heterogeneous software systems (and related technology needed for the creation of an integrated information infrastructure). He is the current director of the Stanford Center for Information Technology. He can be reached at the Computer Science Department, Gates 2A, Stanford 94305-9020 or as Genesereth@cs.stanford.edu.

References not retained for IEEE Expert Publication

[Catell:94] R.Cattell (eds): The Object Database Standard: ODMG (3 9version 1.1); Morgan Kaufman, May 1994

[Chu:92] W.W. Chu and Q. Chen, "Neighborhood and Associative Query Answering,"; *Journal of Intelligent Information System*, Vol.1 No.3/4, 1992, pp.355–382.

[Cox:94] Brad Cox: The Coalition On Electronic Markets; http://www.site.gmu.edu/ bcox/CEM/00CEM.html

[DD:93] C.J. Date and Hugh Darwen: *A Guide to the SQL Standard, 3rd ed*; Addison Wesley, June 1993.

[Dunstan:89] James E. Dunstan (ed.): *Knobots in the Real World*; Report of Workshop on the Protection of Intellectual Property Rights in a Digital Library System, Corp. for Nat. Res. Initiatives, Reston VA., May 1989.

[Elmagarmid:90] Ahmed Elmagarmid et al: "A Multidatabase Transaction Model for Interbase"; *Proceedings of the Conference on Very Large Databases* 16, Morgan Kaufman pubs. 1990.

[FCGB:91] R. Fikes, M. Cutkosky, T.R. Gruber, and J.V. Baalen: *Knowledge Sharing Technology Project Overview*; Knowledge Systems Laboratory, KSL-91-71, November 1991.

[FFMM:94] Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire: "KQML as an Agent Communication Language"; to appear in *The Proceedings of the Third International Conference on Information and Knowledge Management* (CIKM'94), ACM Press, November 1994; http://www.cs.umbc.edu/kqml.

[Fisher:94] David Fisher: *Adanced Technology Program, Component-Based Software* (94-06); National Institute of Standards and Technolgy, U.S. Dep.of Commerce. 1994.

[Franchitti:94] J.C. Franchitti, R. King, and O. Boucelma: "A Toolkit to Support Scalable Persistent Base Infrastructures"; in *Proceedings of the Sixth International Workshop on Persistent Object Systems*, Tarascon, France, September 1994, Springer-Verlag LNCS.

[G:92] Michael Genesereth: "An Agent-Based Framework for Software Interoperability"; Procedings ARPA Software Engineering Conference, Los Angeles 1992, Meridian Corporation, Arlington VA, pp.359-366; to appear in AI magazine, 1995.

[GSS:94] Michael R. Genesereth, Narinder P. Singh and Mustafa A. Syed: "A Distributed and Anonymous Knowledge Sharing Approach to Software Interoperation"; *Proc. Int.Symp. on Fifth Generation Comp Systems*, ICOT, Tokyo, Japan, Vol.W3, Dec.1994, pp.125-139.

[ISX:94] ISX Corporation: Intelligent Integration of Information (I3); http://isx.com/pub/I3, Westlake Village CA, 1994

[King:95] Roger King et al: *Reference Architecture, Intelligent Integration of Information Program*; http://www.cs.colorado.edu/ dbgroup/i3-ref-arch.html, University of Colorado, Sept.1995.

[Kerschberg:95] Larry Kerschberg et al.: *Reference Architecture, Intelligent Integration of Information Program*; http://isse.gmu.edu/I3_Arch/index.html, Dec.1995.

[Kieburtz:94] Richard B. Kieburtz: "Generating Software from Specifications"; *Proceedings of the Monterey Workshop on Formal Methods*, Luqi (ed.), U.S. Naval Post Graduate School, Sept. 1994, pp.122–128.

[Koster:94] Martijn Koster: *Rules for Web Robots*; Nexor, 1994, http://web.nexor.co.uk/users/mak/doc/robots/robots.html

[OMG:91] Object Management Group: *The Common Object Request Broker: Architecture and Specification*; OMG Document 91.12.1, OMG and X/Open, distributed by QED-Wiley, Wellesley MA, 1991.

[PFPMFGR:92] R.S. Patil, R.E. Fikes, P.F. Patel-Schneider, D. Mckay, T. Finin, T.R. Gruber, and R. Neches: The DARPA Knowledge Sharing Effort: Progress Report; in Nebel, Rich and Swartout, eds., *Principles of Knowledge Representation and Reasoning*; pp.777–788, Morgan Kaufmann, 1992.

[RDCLPS:94] B. Reinwald, S. Dessloch, M. Carey, T. Lehman, H. Pirahesh and V. Srinivasan: "Making Real Data Persistent: Initial Experiences with SMRC"; *Proc. Int'l Workshop on Persistent Object Systems*, Tarascon, France, pp.194–208, Sept.1994.

[Salton:90] Gerard Salton: "Full Text Information Processing Using the Smart System"; *IEEE CS Database Engineering Bulletin*, March 1990, Vol.13 No.1.

[STEP:92] International Organization for Standardization: *ISO 10303 Industrial Automation Systems and Integration — Product Representation and Exchange — Overview and Fundamental Principles*; Draft standard ISO

[Tourtier:94] Paul-Andre Tourtier: "A Flexible, Facilitator-based Cooperation Framework"; Proc. Int.Symp. on Fifth Generation Comp Systems, ICOT, Tokyo, Japan, Vol.W3, Dec.1994, pages 101-110.

[Ware:93] Willis H. Ware: *The New Faces of Privacy*; Rand Corporation, report P-7831, Santa Monica, 1993.

[WCC:94] Gio Wiederhold, Stephen Cross, Charles Channell: *Information Integration*; IEEE Educational Videotape, 2 hours, October 1994, Robert Kahrman, sponsor. IEEE, Picataway NJ.

[W:93] Wiederhold, Gio: "Intelligent Integration of Information"; *ACM-SIGMOD 93*, Washington DC, May 1993, pages 434-437.

[W:94A] Gio Wiederhold: "An Algebra for Ontology Composition"; *Proceedings of 1994 Monterey Workshop on Formal Methods*, Sept 1994, U.S. Naval Postgraduate School, Monterey CA, pages 56-61.

[W:94I] Gio Wiederhold: "Interoperation, Mediation, and Ontologies"; *Proceedings International Symposium on Fith Generation Computer Systems (FGCS)*, Workshop on Heterogeneous Cooperative Knowledge-Bases, Vol.W3, pp. 33–48, ICOT, Tokyo, Japan, Dec. 1994.