Value-added Mediation in Large-Scale Information Systems

Gio Wiederhold
Computer Science Department, Stanford University
Stanford CA 94305, U.S.A., gio@cs.stanford.edu

Keywords
Information system architecture, mediation, client-server, objects, domain models, value-added services

Abstract
Many information-processing tasks can be part of multiple customer applications, as summarizing stock prices, integrating catolog information from several companies in the same line of business, predicting the weather, and checking on transportation resources. We assign such sharable services to an active middleware layer, interposed between clients and servers. We define domain-specific mediator modules to populate this layer.

Such mediating services must be of value to the customers, so that it will benefit their client applications to access mediators rather than the server sources directly. Several categories of value can be considered: improvement in access and coverage, improvement of content, and delegation of maintenance. We will define criteria for mediating modules: ownership by party who assumes responsibility for the rseults of the services, domain-specificity to delimit the scope of such a responsibility, and, of course, conformance with interface standards that enable interoperation.

Applications that benefit from mediation include planning and other decision-making tasks that require information from diverse resources, as databases, reference systems, data obtained from sensors, and analyses for projecting trends into the future. The sources are often autonomous, some of them are public, and will typically be heterogenous. The heterogeneities include representation, scope, level of abstraction, and context.

Mediated results are intented to be composable by higher level application programs, so that the applications can solve problems involving multiple subtasks and multiple domains. Mediation requires domain expertise, and mediators are hence domain-specific. Mediators need a machine-friendly interface to support their client applications. This interface must provide good communication, while encapsulating the mediation tasks, so that the complexity of the composed system is not much greater than that

of the individual subtasks. However, questions of effectiveness and efficiency do arise, and must be dealt with by exploiting the processing and storage capabilities of modern hardware.

The corresponding architecture is best viewed as a generalization of a server-client model. The partitioning into layers and by domain enhances maintainability: the applications software staff can concentrate on functional improvements, the data resource managers on operational issues, and the technical maintenance is concentrated in the mediator layer. The concept is network-oriented, and mediating services are best provided over the network by domain specialists. With suitable payment mechanisms, mediating services can be performed by independent entrepreneurs.

1 INTRODUCTION

As information systems are increasing in scope, they depend on many diverse, heterogeneous resources. These resources are typically developed and maintained autonomously of the majority of the applications that use their results. Motivating applications for establishing a database tend to be operational, as inventory control, payroll, production workflow, and help desks [WiederholdCC:1994]. Other services available over networks include textual information, geographic data, and simulation results, as weather forecasting.

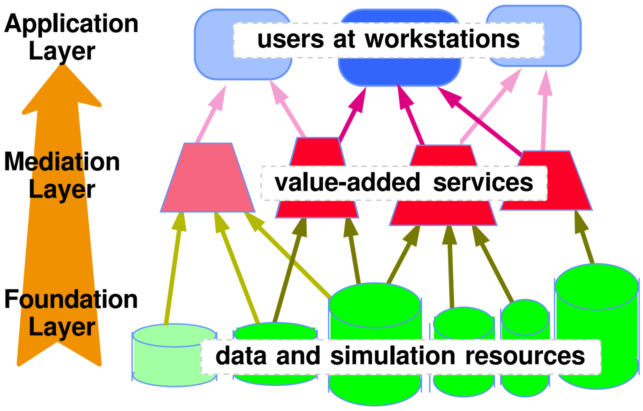

Figure 1. Transforming Data to Information

When the disparate data collections can be combined, they can support high-level applications, as decision-making and planning. Decision support applications are typically designed independently and subsequent to the establishment of operational databases. Computer-based planning support, i.e., developing and assessing a chain of candidate decisions into the future, is a management task which is still poorly supported in practice. It is rare that sufficient time is available to build data collection for planning from scratch, so that planning systems have to rely on existing sources. Figure 1 sketches the architectural layering of the data and information services.

Dealing with many, diverse, and heterogeneous sources overwhelms decision-making support applications. The problem is due to the need to deal in detail with many irrelevant, but crucial details. Mediators provide intermediary services, linking data resources and application programs [Wiederhold:1992C]. Their function is to provide integrated information, without the need to integrate the data resources. Specifically, the tasks required to carry out these functions are comprised of

1 Locating, accessing, and retrieving relevant data from multiple heterogeneous resources;

2 Abstracting and transforming retrieved data to a common model and level so that they can be integrated;

3 Integrating the transformed 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 2. Overall, this processing adds value by converting data to information. It is this aspect that this presentation will focus on. More detail is provided in Table 1.

> Definition: A mediator is a software module that exploits encoded knowledge about certain sets or subsets of data to create information for a higher layer of applications. It should be small and simple, so that it can be maintained by one expert, or at most, a small and coherent group of experts [Wiederhold:1992C] .

Tasks of Type 1 above, i.e., locating and retrieving the data are the current emphasis of much work in information systems. Access is a necessary prerequisite to deal with distributed information. Well known examples of approaches to improve access are systems are KNOBOTS, facilitators and the variety of WEBCRAWLERS, as well as multi-database systems [GenserethSS:1995], [Hurson:1993].

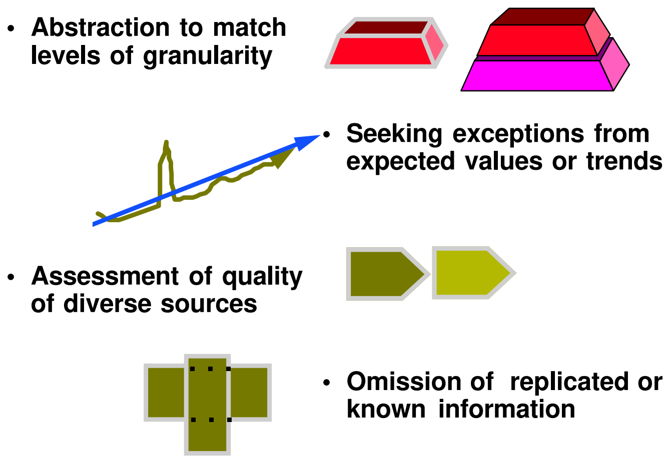

Figure 2. Some Processing Features of Mediators

In mediation the focus is on processing the results of the retrievals (task Types 2-4 above) rather than improving retrieval (task Type 1), i.e., dealing with the overload on the customer that effective retrieval can create. The added value of mediation is due to intelligent processing. The investment in building mediators is justified by the expectation that mediation services can support multiple applications. The value-added provided by mediation warrants the interposition of a distinct architectural layer, as was sketched in Figure 1.

Earlier programs that exhibited such intermediary functions and led to the concept of mediation were either constructed to support specific applications, or extended services from databases [Wiederhold:1992C].

2 STATUS

Mediation is an architecture which extends the client-server model and provides scalability. The sketches in Figure 3 show the development from (a) client-server systems in a setting that assumed compatibility, i.e., dealing in a single domain, often dominated by local data, to (b) the addition of wrappers to deal with legacy data and an integrator to combine resources, (c) the sharing and associated generalization of the integrators, to (d) domain-specific services performed by independent mediators on the network that are not bound to the client nor to the server. Finally, applications that require information from multiple domains will use multiple mediators (e).

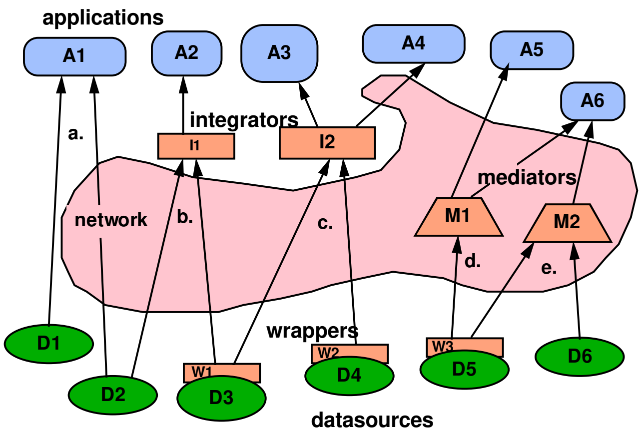

Figure 3. Evolution of network services from Client-Server to Mediation

Mediators are being built now by careful domain knowledge acquisition and handcrafting the required code. Building mediators by hand is essential to validate the concept and establish the needed standards for the interfaces. Since the mediator approach is conceptually comprised of three layers, as shown in Figure 4, there will be two major interfaces:

a Mediators to applications;
b Base resources to mediation.

In practice there will also be intermediate interfaces, since it is likely that within the layer a number of sublayers exist as well. For these the protocols chosen for interface a. are also appropriate.

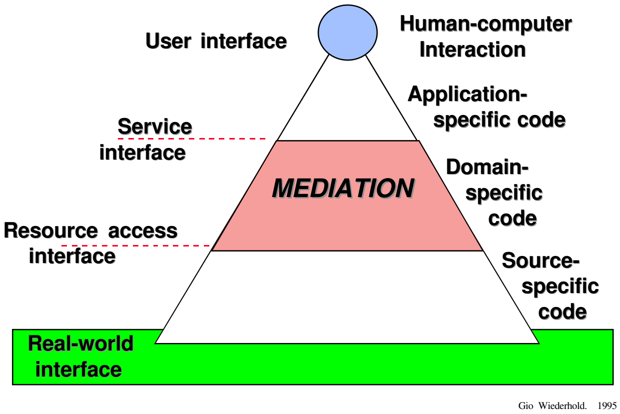

Figure 4. The Mediation Layer and its Interfaces

2.1 Interfaces

For the base interface (b.) the many tools that are becoming available to serve the two-layer server-client model are appropriate, as distributed and augmented SQL [Date:1993] and the interface tools for object-oriented access, as CORBA [OMG:1991]. At the application layer (a.) the interfaces need greater capabilities. Here agent languages as KIF for content and KQML for managing the kinds of multi-user, multi-representation, and multi-function operations are appropriate [GeneserethK:1994], [Patil:1992]. While KIF is defined to exchange knowledge in a first-order logic representations of rules, KQML also can handle other representations, as objects [MMDF:1994], mathematical equations [Fikes:1991], and even tuples. The requirement is, of course, that the sender and the receiver agree on the chosen representation, as specified in the preamble of a KQML statement. The sender and the receiver must also agree on the vocabulary and its structure, i.e., the *ontology*, specified in the preamble. This aspect will be the focus of Section 6 of this paper.

2.2 The user interface

Mediation is simplified by delegating the complexities of the customer interface to the application program. Code to deal with the variety of graphical user interfaces (GUI), interface devices (screens, 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, as represented by a KQML application program interface (API) block. A limited amount of GUI will be needed to interact with the owner of the mediator for its maintenance, as described in Sect. 5.4. Trading internal code for an API recapitulates an earlier paradigm shift. We recall that in the 60's 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 DBMS 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.

3 CREATING MEDIATORS

In this paper we present the conceptual underpinning for automating the mediation process. Automation does not mean fully automatic code generation, since knowledge is necessary to provide the added value that offsets the costs of having the 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 [Wiederhold:1992I]. At that time the more general vision presented in this paper had not yet been formulated.

Value-added services in a mediator include combinations of:

1 Selection of likely resources using precomputed information [Gravano:1994]
2 Invocation of wrappers to deal with legacy sources [Franchitti:1994]
3 Selection of relevant source material [Neches:1993]
4 Optimization of access strategies to provide high performance [King:1984]
5 Imposition of security filters to guard private data [Billelo:1995]
6 Resolution of domain terminology mismatches [DeMichiel:1989]
7 Resolution of ontology differences [Wiederhold:1994J,95M]
8 Resolution of scope of domain mismatches [Wiederhold:1992I]
9 Inter- or extrapolation to match differences in temporal data [Wiederhold:1993]
10 Reduction of historical data to limited snapshots [Shahar:1993]
11 Abstraction to match levels for integration [Chawathe:1994]
12 Integration of diverse material based on matching of candidate join keys [Li:1994]
13 Omission of replicated information [Brin:1995]
14 Assessment of quality of material from diverse sources
15 Pruning of data ranked low in quality or relevance [Salton:1990]
16 Omission of information already known according to the user model
17 Statistical summarization into higher level categories of the user model [deZegher:1988]
18 Reporting exceptions from expected values or trends
19 Triggering of actions due to exceptions from expected values [Buchmann:1995]
20 Transformation of material to make presentation effective for the customer
21 Adaptation to the bandwidth, device, and media capabilities of the customer
22 Transmission of resulting information and meta-information to the client [GM:1995]

We show only reference per topic, and chose those we are familiar with; no comprehensiveness is implied.

Table 1: Services to be provided in mediators.

All these tasks, and related tasks specific to specialized domains as finance, scheduling, etc., are now available in some or other existing applications, but are rarely identified as composable and reusable tasks. Placing them into mediators enables their reuse and partitions information processing tasks by specialty. Whenever these tasks or their combinations can provide enough added value to overcome the cost of adding a layer in the information processing flow, having identifiable and maintainable service modules provides long-term management benefits. Some of these services will be 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 in [Wiederhold:1995D].

4 THE PROCESS OF MEDIATION

The software equivalent of interchange hubs are called mediators. We will summarize here the essential aspects of mediation, an activity which reduces data to information by applying knowledge about resources, search strategy, and user requirements. Mediation is an integrating concept, combining a number of current technologies to find and transform data, and making the resulting information available at hubs along the information highways. Mediation recognizes the autonomy and diversity of the data systems and information services that support the hubs, and the user applications that utilize them. The autonomy of the participants enables the overall system to grow, since new sources, new means of transport, and novel information processes can be inserted. Incremental growth only requires that a few mediating hubs be adapted to link the new facilities into the traffic network. As the new facilities become more popular, further mediators will be adapted to take advantage of them and the business they represent. Those users that need the new sources will use the adapted mediators; users that don't care remain unaffected, as indicated in Figure 5. The crucial benefit is, that now not all changes associated with an upgrade have to be synchronized.

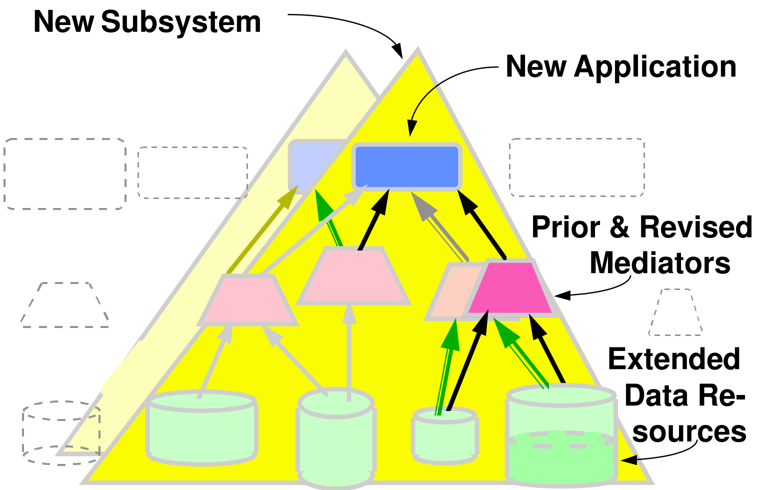

Figure 5. Incremental system growth through mediator reuse

Sometimes much processing is needed to abstract and merge data, so that specialized services may be utilized by a mediator, just as transportation hubs encourage specialized factories to provide value-added services.

The implementation of mediators may vary a great deal. If knowledge-based processing is crucial, we may find mediators 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. Many of our current mediators have been coded in C or C++.

Projects using mediators today have included manufacturing systems, as for the design of a new fighter aircraft at Lockheed Aeronautical Systems and gimbals for antenna positioning on spacecraft at Lockheed Space Systems. Access to data for healthcare services, managed at the University of Texas in Arlington, is slated to use mediator technology. Early applications have been in integrating information for flexible military command systems. The technology for these projects is supplied by a variety of vendors, ISX Corporation serves as a contact point for these projects [ISX:1994]. For the F-22 Integrated Weapons Systems Database (IWSDDB) project access is provided to IBM IMS design data (via SYBASE), DEC VMS factory information, UNIX workstations, WAIS for text access, and PC databases. One phase is shown in Figure 6. IWSDB is developed in phases, since simultaneous integration of information from the more than 100 local databases plus about 200 databases at independent supplier sites is not feasible. Physical integration of the databases is clearly impossible.

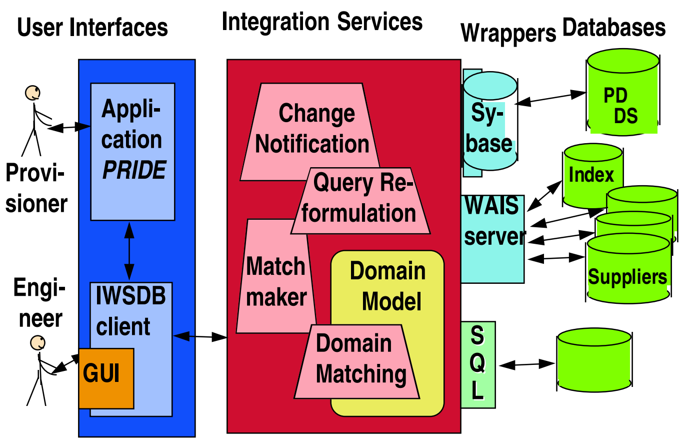

Figure 6. A phase of the F-22 IWSDB development

Since we assume that most mediation services are performed in autonomous computing nodes, the requirement for interoperation is the ability to communicate according to some standard conventions [Finin:1994]. Heterogeneities in computing platforms, operating systems, data representations, and message passing infrastructures are being overcome by concerted efforts in many communities. For mediation we must also consider the conventions used for terminology used to describe the objects in a domain.

4.1 Access

Netsurfing, using the World-Wide-Web [Kostner:1994] and MOSAIC tools is a common browsing technology today. The knowledge about access paths is embedded in referencing linkages. A number of subsidiary services have sprung up that collect such linkages periodically, as HARVEST [Bowman:1994], ALIWEB, NETCRAWLER, and FISH. The linkages are placed by subject into temporary databases, and represent a cached knowledge-base. Some systems also cache the actual text, or at least their abstracts, to reduce the work for successive retrievals. Today all these services are free, so that few resources are available to monitor quality. A recent experiment showed that all of them were quite incomplete, and at most provided initial entry points for a dedicated researcher. Most of the documents obtained do not carry copyright. If they do, copyright may well be violated by the use of these tools.

5 PARTITIONING

The amount of knowledge to perform all possible aspects of mediation is enormous. Modularization or partitioning is the divide-and-conquer tool used in science to deal with excess size and complexity. Here size and complexity go together, because as the number of concepts grows, their linkages, their distinctions based on context, and their subsumption relationships also grow, and that growth is more than linear.

In computing we can distinguish two types of partitioning which are orthogonal:

1 By top-down partitioning into distinct processes, linked to each other by input-output interfaces. This type of partitioning is commonly preached as the only solution in Software-Engineering texts [Gibbs:1994];

2 By structuring and then splitting the world of data and meta-data. Such a modularization is exemplified by the common data-engineering paradigm [WiederholdQ:1994].

The first choice is not available to us. In large systems there is no single decomposable objective, and hence no single modularizable process. There will be multiple applications, and they should be able to rearrange their processes in a way that is optimal for them [Wiederhold:1992D], [Flodin:1995].

Another partitioning is often used by technologists, which distinguish systems according to the technology used, say Artificial-intelligence versus Linear Programming. We do not want to prescribe any particular computing methods for mediators; each mediator should choose whatever methods and technology are best. Of course, interface constraints have to be satisfied, and having an excess of technologies employed will increase module maintenance overhead.

We hence generalize the data and meta-data splitting approach, scaling it up to *domains*. We define a domain to be a subset of knowledge in the world that can be managed by a single person, or at least by a small coherent group. No compromises should be necessary to define terms, and no committee effort should be required. It is best if there is an organization willing to take responsibility for a domain, say the

Society of Pathologists for medical diagnoses or the Automotive Engineering Society for problems with cars [Gruber:1994].

Within a domain the terms are then wholly unambiguous, both in definition and scope. Considering the differences between, say, *employees* in *Personnel* and on the *Payroll*, we need to distinguish those two domains. We show a simple example of two domains in Fig. 15, and note that outside of the domain we should label the terms *Shoesales.model*, *Shoesales.supplier*, etc. We observe that a post-implementation partitioning was required in CYC, when the system grew to a size that was no longer maintainable [Lenat:1990]. They defined *micro-theories* to group related concepts in their general common-sense knowledge base.

Since the number of domains is large we cannot foresee a central solution to that problem, and hence will have to deal with subsets of the domains, and bring those subsets together, as and when needed.

5.1 Domain-specific Partitioning

Large enterprises can only operate when they are partitioned into departments, with specialized authorities and responsibilities. We follow the same paradigm in mediation, and expect, in fact, that mediator partitioning will follow organizational boundaries. The mediator paradigm returns power from the computing manager to the domain specialists, as sketched in Figure 7.

Figure 7. Domain Specialization

Specialization makes knowledge maintenance feasible; an expert can focus on one's own domain, without having to consider the different constraints imposed by handling unrelated domains, say, fish versus bicycles [Tracz:1995]. Differing domains may be best served by different programming paradigms, say finance versus engineering. Other subsets of domains may use similar technology, but differ in the concepts and structure of their knowledge representations, say electronic versus civil engineering. Section 6 of this paper will deal with the latter issue, namely the management of diverse *ontologies*.

5.2 Distribution

Mediation is achieved by software. A mediator transforms data available on the network to make it more suitable and relevant to the consumer. This software function can be carried out on the computer where the mediator was developed, or can be assigned to other computers on the network. Since software is easy to copy over the digital highways, mediators can be rapidly replicated. Software is typically much smaller than the data it processes, so that it is easy to install at any location where compatible computers are available. The ease with which software-based factories can change location implies a much more flexible configuration of the network than seen in traditional manufacturing. A supplier who fails to deliver sufficient added value can be rapidly replaced.

Replication can also enhance the technical performance of mediators. By locating the mediator near the data source, communication lead can be reduced, since the resulting information is typically smaller. If the processing algorithms used in mediation are costly, the software can be moved to a high performance processor on the network, again improving response time.

If demand for a particular type of mediator is strong, replicates can be distributed to additional sites in the network. Since now the communication links will shorten, response time is enhanced as well.

5.3 Caching

At a hub one often finds hotels for people and warehouses for goods. These are necessary because the variety of transport mechanisms cannot be perfectly synchronized, and some temporary storage is needed. The equivalent function in the digital systems is intermediate storage or caching of data. In a mediator only copies of selected and processed information, likely to be used later, needs to be stored. Any stored data must be well identified with name, type, source, date-of-validity, and mediation performed, so that it can be appropriately used. If source data has to be cached, it may be wise just to keep a reference to the source in the mediator, and acquire it from the source databases when required [Roussopoulos:1991].

5.4 Maintenance

Mediation adds value to the data by applying the expert 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 be a candidate for replacement by a competitor.

Existing users 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 now upgrades are not constrained by the effect on the existing community of users. The maintainer will, of course, try to keep the number of versions of a mediator service modest. The charges for old mediators may increase, to encourage applications that depend on old versions to upgrade.

Providing maintenance service for applications in a world that is constantly changing is a major, maybe the major, added value in mediation. The cost to applications of dealing with changes because their resources are altered is very high. Change must occur in resource databases whenever anyone of the applications needs new data, but costs are imposed on possibly hundreds of other customers.

It is important that new systems we develop are designed with maintenance in mind, else we will follow the example of the traditional software industry where 60 to 95% of overall cost is in maintenance, and yet the users complain that the services are always out-of-date.

5.5 Selection and review

Today we count on intermediaries to help locate high-quality and relevant information. Reviewers and editors filter much material prior to publication, and indexers, librarians and post-publication reviews provide additional services [Wiederhold:1995D]. The added value of review, managed by publishers, is not easy to manage when much material is placed by the authors on the WWW. Subscribers to publications will be lost if the equivalent material is available to anyone by integrating documents according to a list of current contents, which is easy to place on the Internet, since most papers will be on the net as well.

Forward-looking publishers are trying to resolve some of this issues, but the conflict between free dissemination of information and reimbursement for value provided is great [Maurer:1994]. An intermediary agent, taking the role of a publisher, sustained by a mediator, might be able to filter material with respect to quality and accessors, taking into account privileges gained by subscription or payment of fees.

5.6 Views and objects

Designers of relational databases have used *views* to define subset of source databases that are relevant to a particular application. For a mediator, the view would be determined by the scope of the domain. Rather than creating a single, unnormalized relation representing the view, as is done by default in relational systems, it is better to create objects classes corresponding to concepts relevant for the domain [Barsalou:1990]. When the semantics of such objects are carefully managed, the view-update problem can be resolved effectively, permitting update as well as retrieval of domain-specific information. Objects are always hierarchically structured, making them easy to navigate and understand.

5.7 Summarization

The volume of material on the World-Wide networks is huge and growing. A significant added-value service is provided by summarization. A recent examples is the generation of tables describing the status of publically traded companies based on the K-10 reports they submit to the regulators [Krishnan:1994]. Seeking for exceptions is an even more effective summarization technique, since now only information that needs follow-up is transmitted to the user. An example here is the warnings issued by some tax-preparation programs that a person's tax deductions is outside of the normal range and likely to trigger special consideration by the income tax people.

In the design of a mediator the added-value must be balanced with its complexity. We are still gathering experience about this trade-off, as illustrated in Figure 8. Two dimensions must be considered:

1 The amount of processing provided in information generation
2 The breadth of the domain to be covered

In both caeses adapatability is a concern, because information requirements will continue to change. The technology to build and maintain mediators is a concern. Their manageable size provides opportunities to apply modern programming technology [Papakonstantinou:1995].

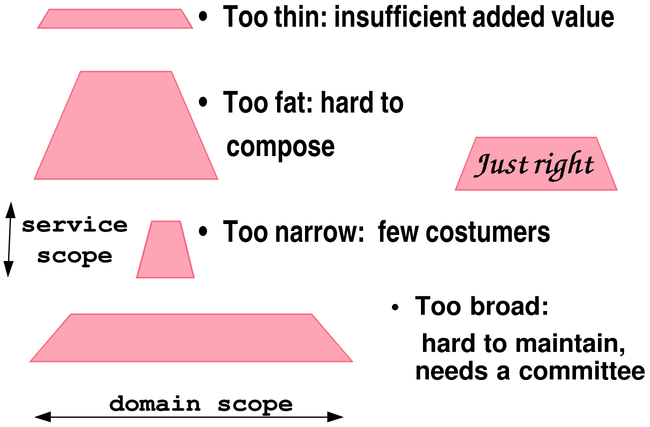

Figure 8. Fat versus Thin Mediators

5.8 Knowledge-based computing

A mediating module carries out tasks to serve a user. A desirable aspect of such a computer-based servant is that it has a model to *understand* what is needed, can be directed by the user, and, a prerequisite for direction, can explain to the user the current information structure it uses. Database systems explain themselves by exposing their schema. In mediation the hierarchical model of the domain task and its references to the resources accessed have to be made visible. While understanding the intent of user is very hard, when the tasks are well organized and suitably limited, the effort to provide an adequate interface can become quite manageable. The information to support understanding must be formalized into a machine-processable model. It is crucial to have a simple model of the user's application structure, and of the available resources in the domain. The structure is typically hierarchical. The linkage between the user and the model is represented by the domain *ontology*, as defined in Figure 9.

The model of the application, representing the current view of the user as it pertains to the domain of the mediator, consists of a listing of concepts, and relationships among them, the *ontology*. A mediator will manage access and processing of data corresponding to the concepts its domain. The concepts are typically hierarchically structured, following the divide-and-conquer paradigm used for scientific problem-solving. The relationships can be seen as candidate paths among the concepts, while the nodes are typically represented by information that can be obtained from the resources. The linkages among the resources form a network. Complementing the ontology are the processing paradigms used in that domain. Over time processing requirements and some linkages will change, requiring mediator maintenance.

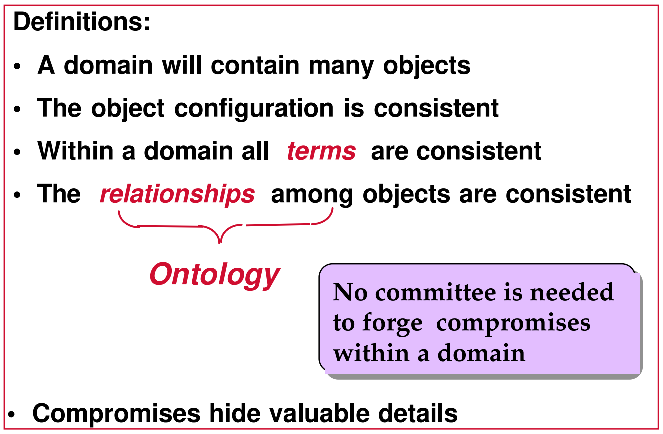

Figure 9. Domains and Ontologies

5.9 Revenue Collection

We assume an economic model for mediation. Unless services are reimbursable, it is unlikely that they will be maintained very long. The traditional method for obtaining software has been purchase, and those purchases typically embody restrictions on replication and secondary distribution, closely following the copyright protection model [Denning:1995].

More satisfactory approaches to revenue collection are being developed today [Cox:1994]. The user of a mediator should be able pay for services received incrementally, i.e., with each use. Incremental charges encourage trial use, since the financial investment will be small. Since a mediator can live remotely from the user's application workstation, the charges are easily collected, and fraud, common today in software acquisition, becomes much more difficult.

The creator and maintainer can then receive payments proportionally to its use. Since the fee per use will be small, it is essential that a direct and automatic electronic payment mechanism be used.

6 ONTOLOGIES

To permit information from distinct sources to be accessed by a mediator, there must be agreement on the terminology in the shared area. It is not sufficient to match information on the basis of words, because words differ in meaning in different contexts. For each domain an ontology must be defined: a vocabulary of terms and a specification of their relationships. With each term a definition is needed, both an informal annotation, perhaps with some examples, and a more formal specification of relationships, as *part-of*, *owned-by*, *referenced-by*, *subclass-of*, etc., which cite other terms in the domain [WiederholdQ:1994]. The entire structure forms a semantic net. Isolated terms should be avoided.

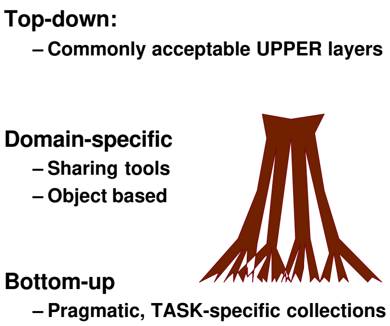

Figure 10. Establishing Ontologies

6.1 Sources

Initial sources for ontologies are defined vocabularies for a domain, perhaps represented by a textbook on the topic, thesauri that have been created to help indexers create bibliographies and help researchers locate references [Humphreys:1992]. Smaller, but practical and verifiable ontologies can be obtained from database schemas. In the original design document for a database additional information may be available, as the relationships among represented concepts and constraints imposed on the attributes. The ontology can capture this knowledge for reuse.

Standardization efforts have contributed much effort towards ontologies, since the proper definition of a standard depends on getting all subsidiary terms precisely defined [Morris:1994]. Some large ontologies have been created for knowledge-based processing, since it becomes essential that the human understanding of the processes is faithfully transmitted to the computer. Within a domain terms are generaaly well-undrrstood, outside of a domain meanings can differ drastically, as shown in Figure 11.

Only recently have the results of gathering terminologies and structuring their relationships been recognized as a valuable effort in its own right, deserving of attention. The work of Tom Gruber traces that development [Gruber:1993]. Most current candidate ontologies are hidden, since they are part of the infrastructure for some effort in communicating among people, or between people and systems they share.

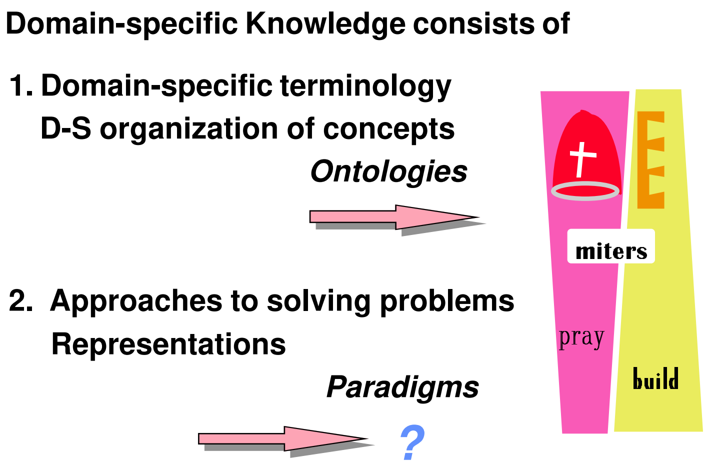

Figure 11. Knowledge, Ontologies, and Paradigms

6.2 Ontology integration

Domain differences make a once-and-for-all integration of distinct domain infeasible. A dynamic merging capability is essential if we wish to achieve associative access to multiple domains, since the transformations required to achieve optimization must maintain the correct semantics. We have experience with that issue in object-databases, since objects tend to represent a hierarchical view while the underlying world of data allows arbitrary composition via the relational algebra [Wiederhold:1986]. For instance, the PENGUIN system constructs objects as needed out of relational databases, given a structural model of connecting references [Barsalou:1991].

There are several approaches to dealing with building composed ontologies from domains that have ontological differences:

1 Aggregate the terms from all relevant ontologies, give them to a committee, and ask them to prepare definitions that are acceptable to all. When the definitions are completely documented, release the document and expect that all participants will adjust their usage to conform to the definitions [Morris:1994]

2 Assume that terms match, and when mismatches are discovered, make the terms distinct, typically by prefixing them with source or domain identifier. This is the approach used by UMLS [Humphreys:1992]; all the sources are labeled to make such distinctions easy, and by CYC, where micro theories can encapsulate differences LG:1990]. Over time, the processes of sharing of information encouraged by the availability of the joint ontology will cause convergence of meanings, although coherence can never be assured.

3 Assume that term never mean the same thing unless explicitly instructed. Such instructions, encoded as *matching rules*, form a knowledge-base to be managed by collaborators from two or more domains [Wiederhold:1995M]. No restrictions are imposed on the evolution of local terms within a domain. Terms that are covered by matching rules form a new, second layer abstract ontology. Higher abstract layers

can be defined recursively, leaving unneeded abstract terms local in their abstract layer.

We focus here on the third alternative.

7 DOMAIN-SPECIFICITY VS. DOMAIN INTEROPERATION

We achieve scalability of information systems in this approach by the ontological partitioning. We enable composition over the parts by having a knowledge-based algebra, with operators as shown in Figure 12. The individuals chartered with defining and maintaining the knowledge need more breadth than those that maintain domain-specific ontologies, but do not need the same depth of knowledge.

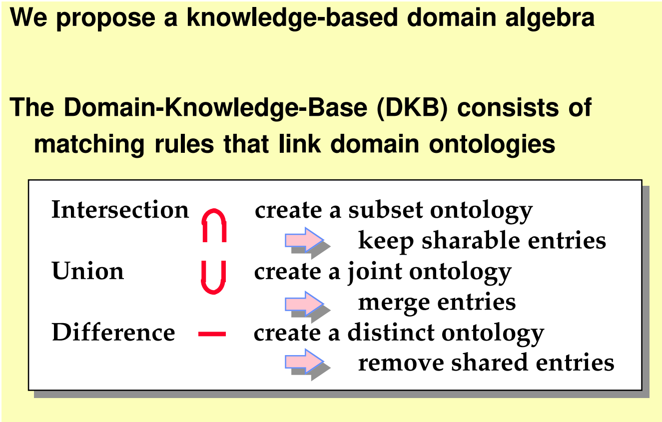

Figure 12. A Proposed Domain Algebra

Constraining terms to a domain is useful, since most ambiguities disappear when the domain context is known. When a term is used outside of its context, it is labeled with its source domain, for example: *Carpentry.miter* versus *Religion.miter*. These domains are largely disjoint, and terminology can be maintained independently. It is actually likely that the domain *Religion* will be subdivided into hundreds of autonomous domains, since it is likely that no agreement can be reached for many crucial terms. The term *miter* need only occur in a few of those subdomains.

The pessimistic principle in domain-partitioned mediation is that terms do *not* match unless there are matching rules in the DKB. The relative autonomy of the local source terms provides scalability. The layered structure actually adopts for its information structuring the successful domain management strategy used by the INTERNET distributed naming conventions [Kahn:1987].

With the conservative assumptions embedded in the *DKB-model*, the risk is that, because of having insufficiently many matching rules, too little information will be retrieved. By assigning the task of creating matching rules to multiple specific expert groups, we expect that high quality operations over data from distinct, but overlapping, domains can be created at a reasonable cost. To evolve these systems effectively, feedback loops must exist that permit users to suggest new candidate matching rules, or to modify existing ones. Having small, distributed groups to maintain the partitioned *DKB-models* will help ensure responsive maintenance of the domain knowledge.

Note that we stress maintenance throughout. Keeping ontologies small enables semantic agreements among the people using them with little lag time. If maintenance takes too long, some terms will change in meaning during the time it takes to agree.

7.1 A DKB algebra

Keeping all terms disjoint disables the domain interoperation we seek. Automatic reasoning about ontologies requires a formal approach to transformation and their equivalences, eventually enabling optimization of access. We have to develop a set of operations that permits us to match and merge ontologies: a *Domain-Knowledge-Base Algebra* over ontologies. Only a few operations seem to be needed in the DKB algebra, namely Intersection, Union, Difference, and Map [Wiederhold:1995M]. All these operation require associated matching rules.

Intersection of two or more base ontologies provides the shared ontology needed for semantic integration of two domains. Unions, typically taken over shared ontologies obtained by intersections, permits the construction of sparse but effective enterprise ontologies. Difference permits the definition of local subsets which can be maintained autonomously. No negation is provided to avoid the problem of having infinite sets. Mapping operators provide for domain interoperation where definitions or structures of ontologies differ.

The DKB algebra can provide a basis for interrogating multiple domains and their databases which are semantically disjoint, but where a shared knowledge-base has been established. This process mirrors the approach used in CARNOT, where a knowledge base is used to create *articulation axioms* for joining of data [Collet:1991]. CARNOT employs a large and broad CYC knowledge base, which makes the optimistic default assumption that terms that are spelled the same match, unless told otherwise. This means that irrelevant retrievals can occur, so that in practice CARNOT system applications must limit the depth of search.

7.2 Implementation coherence

In order to keep projects working in different domains, on different tasks, and on different platforms from diverging, ARPA managers convened two workshops. A reference architecture document and its base vocabulary was initially established during the I3 (Intelligent Integration of Information) Architecture Meeting in Boulder CO, 8 Nov.1994, organized by Roger King of the University of Colorado and refined during the I3 architecture meeting 22 Jan.1995, organized by Mike Genesereth. of Stanford University. It is based on contributions by many I3 participants [Genesereth:1995].

8 CONCLUSION

Information technology is serving us well in specific domains, although customers have remained dependent on specialist model designers and programmers for the implementation of their systems. Object technology has lessened our dependence on specialists by being able to use an infrastructure which aggregates detail into meaningful units in many important domains.

Mediation is a technology that is intended to scale systems so that sources from many domains can contribute services and information to the end-user applications. When the breadth of information system grows beyond coherent domains, knowledge about their intersections should be incorporated. To profit from such knowledge we propose a knowledge-based information algebra. The tasks of collecting and maintaining the matching rules to support such algebras can be naturally partitioned among specialists and collaborating integrators. Integration can proceed at multiple levels of abstraction, avoiding the centralization that hinders progress in exploitation of data from diverse sources. The two prime levels of integration are depicted in Figure 13. The are characterized by a formal service infrastructure supporting pragmatic applications. The combination provides scalability as well as flexibility in application development.

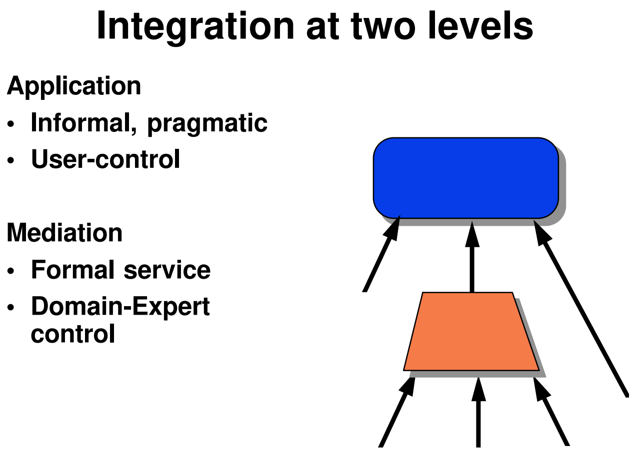

Figure 13. Integration at Two Levels

Tools are needed to support such development, but to have effective tools a common formal structure is needed. Both software and artificial intelligence technologies has been hard to scale when domains grew large or became diverse. The technology we describe can provide the needed formalism for scaling, by building on concepts demonstrated in relational algebras, formal management of semantics, and the incorporation of ontological concepts as a foundation for the management of the required knowledge bases.

Recognition of domain autonomy is an important enabler for local progress and adaptation to maintain relevance. At the same time, we wish to have a capability for rapid incorporation of new domains by innovative applications. Support of positive change, namely convergence of mutual understanding, is a major motivation for formal management of ontologies. Unless the current state and extent of mismatch can be well defined, we will not be able to note the improvements being made as people, by interacting via these information structures, achieve increasing levels of consensus.

A number of projects are now using the mediator concept, and some standards, as KQML, are being promulgated for the interfaces that are required [Finin:1994]. A series of prototype efforts support the U.S. Air Force concept of an Integrated Weapons Systems Database [ISX:1994].

ACKNOWLEDGMENT

This research direction would not have been feasible without the support of many colleagues in the ARPA Intelligent Integration of Information program, and of course from ARPA itself. Recent workshops sponsored by Bob Neches and David Gunning of ARPA, and chaired by Bill Mark of Lockheed, Tom Gruber of EITech, and others have contributed valuable ideas. Work in progress on these topics can be located via the WWW node http://www-ksl.stanford.edu/knowledge-sharing .

REFERENCES

Since much of this work is recent, the list provides handles into the World-Wide-Web, where much of the current information appears, especially when it deals with applying the network's capabilities to today's systems.

Thierry Barsalou and Gio Wiederhold (1990): "Complex Objects For Relational Databases"; Computer Aided Design, Vol. 22 No.8, Buttersworth, Great Britain, October 1990, pages 458-468.

T. Barsalou, N. Siambela, A. Keller, and G. Wiederhold (1991): "Updating Relational Databases through Object-Based Views"; *ACM SIGMOD Conf. on the Management of Data 91*, Boulder CO, May 1991.

Michel Billelo, XiaoLei Qian, and Gio Wiederhold (1995): The TIHI Project; http://www-db.stanford.edu/pub/gio/TIHI/TIHI.html.

C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber and Michael F. Schwartz (1994): "The HARVEST Information Discovery and Access System"; *Proc. Second International World Wide Web Conference*, Chicago IL, October 1994, pages 763–771. Also ftp://ftp.cs.colorado.edu/pub/cs/techreports/schwartz/ Harvest.Conf.ps.Z

Sergey Brin, James Davis, and Hector Garcia-Molina (1995): "Copy Detection Mechanisms for Digital Documents"; Proceedings Sigmod 95, ACM, May 1995, pages 398–409.

Alex P. Buchmann, J. Zimmermann, Jose A. Blakeley, and David L. Wells (1995): "Building an Integrated Active OODBMS: Requirements, Architecture, and Design Decisions"; Proceedings Data Engineering Conference 11, Taipei, Taiwan, IEEE, pages 117-128.

Sudarshan Chawathe et al (1994): "The TSIMMIS Project: Integration of Heterogeneous Information Sources'; Proceedings 1994 IPSJ Conference, IPSJ, Japan.

C. Collet, M. Huhns, and W-M. Shen (1991): "Resource Integration Using a Large Knowledge Base in CARNOT"; *IEEE Computer*, Vol.24 No.12, Dec.1991.

Brad Cox (1994): "Superdistribution"; *Wired*, Idea Fortes section, Sep. 1994; see also http://gopher.gmu.edu/ bcox/CEM/OOCEM.html, George Mason Un., Fairfax VA.

C.J. Date and Hugh Darwen (1993): *A Guide to the SQL Standard, 3rd ed*; Addison Wesley.

Peter Denning and Mark Mendelbaum (1995): "The ACM Copyright Policy"; Comm. of the ACM; April 1995.

Linda DeMichiel (1989): "Performing Operations over Mismatched Domains"; *IEEE Data Engineering Conference* 5, Feb.1989; *IEEE Transactions on Knowledge and Data Engineering*, Vol.1 No.4, Dec. 1989.

Isabelle deZegher-Geets, Andrew G. Freeman, Michael G. Walker, Robert L. Blum, and Gio Wiederhold (1988): "Summarization and Display of On-line Medical Records"; *M.D. Computing*, Vol.5 No.3, March 1988, pages 38–46.

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

Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire (1994): "KQML as an Agent Communication Language"; *The Proceedings of the Third International Conference on Information and Knowledge Management* (CIKM'94), ACM Press, November 1994.

Staffan Flodin and Tore Risch (1995): "Processing Object-Oriented Queries with Invertible Late Bound Functions", *Proc. VLDB*, Morgan-Kaufman.

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

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

Michael R. Genesereth, Narinder P. Singh and Mustafa A. Syed (1994): "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.

Michael Genesereth and Roger King(eds.) (1995): *Reference Architecture, Intelligent Integration of Information*; Stanford University and University of Colorado http:/logic.stanford.edu/architecture/reference.html.

W. Wayt Gibbs (1994): "The Software Crisis"; *Scientific American*, Sep. 1994.

James Gosling and Henry McGilton (1995): *The* JAVA *Language Environment: a White Paper*; Sun Microsystems, Mountain View CA, http://java.sun.com/ whitePaper.javawhitepaper_1.html.

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

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

Thomas R. Gruber and Gregory Olsen (1994): "An Ontology for Engineering Mathematics"; in Doyle, Torasso, and Sandewall (eds.): *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, May 1994.

ISX Corporation (1994): Intelligent Integration of Information (I3); http://isx.com/pub/I3.

B.L. Humphreys and D.A.B. Lindberg (1992): "The Unified Medical Language Project: A Distributed Experiment in Improving Access to Biomedical Information"; *MEDINFO 92*, North-Holland, pp.1496–1500.

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

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

Jonathan J. King (1984): *Query Optimization by Semantic Reasoning*; Univ.of Michigan Press.

Martijn Koster (1994): *On Information Discovery*; http://web.nexor.co.uk/users/mak/doc/robots/robots.html

R. Krishnan and D.H. Steier (1994): "Adaptive Access to a Digital Library of Corporate Information"; presented at the *Digital Libraries Workshop*, Rutgers Un., Newark NJ, May 20-21, http://www.cs.umbc.edu/conferences/dl/ abstracts/krishnan.html

Wen-Syan Li and Chris Clifton (1994): "Semantic Integration in Heterogeneous Databases Using Neural Networks"; *Proc. of the 20th Conference on Very Large Data Bases (VLDB)*, Morgan-Kaufmann, pages 1–12.

D. Lenat, R.V. Guha, et al. (19*90): "CYC: Towards Programs with Common Sense"; *Communications of the ACM*, Vol.33 No.8, Aug.1990, pages 30–49.

Herrman Maurer (1994): "Advancing the Ideas of World Wide Web: Hyper-G"; *Proc. Distributed Multimedia Systems and Applications*, Honolulu, 1994, pages 201–203; http://iicm.tu-graz.ac.at/pub/Hyper-G.

KC Morris, Mary Mitchell, Christopher Dabrowski, and Elizabeth Fong (1994): "Database Management Systems in Engineering"; in the *Encyclopedia of Software Engineering*, pp.282–308, John Wiley and Sons; http://elib.cme.nist.gov/fasd/pubs/morris92c.ps.

R. Neches, R. Fikes, T. Finin, T.R. Gruber, R. Patil, T. Senator, and W.R. Swartout (1993): "Enabling Technology for Knowledge Sharing"; *AI Magazine*, Vol.12 No.3, pages 37–56.

Object Management Group (1991): *The Common object Request Broker: Architecture and Specification*; OMG, Framingham MA, OMG Document 91-12-1, Rev.1.2, Dec. 1991.

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

Yannis Papakonstantinou, Hector Garcia-Molina, and Jeffrey Ullman (1995): "MEDMAKER: A Mediation System based on Declarative Specification; submitted for publication; Extended version on http://www-db.stanford.edu/ pub/papakonstantinou/1995/medmaker-extended.ps .

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

Gerard Salton (1990): "Full Text Information Processing Using the SMART System"; *IEEE CS Database Engineering Bulletin*, March 1990, Vol.13 No.1.

Yuval Shahar and Mark A. Musen (1993): "RESUME: A Temporal-abstraction System For Patient Monitoring"; *Computers and Biomedical Research*, Vol.26 No.3, pages 255–273.

Will Tracz (1995): *Collected Overview Reports from the DSSA Project*; Loral Federal Systems, Owego NY.

Gio Wiederhold (1986): "Views, Objects, and Databases"; *IEEE Computer*, Vol.19 No.12, December 1986, pages 37–44.

Gio Wiederhold (1992C):"Mediators in the Architecture of Future Information Systems"; *IEEE Computer*, March 1992, pages 38–49.

Gio Wiederhold (1992D): "Model-free Optimization"; *DARPA Software Technology Conference*, April 1992, Meridien Corp., Arlington VA, pages 83–96.

Gio Wiederhold (1992I): "The Roles of Artficial Intelligence in Information Systems"; *Journal of Intelligent Information Systems*, Vol.1 No.1, pages 35–56.

Gio Wiederhold, Peter Wegner, and Stefano Ceri (1992WC): "Towards Megaprogramming"; *Comm. ACM*, November 1992, pages 89–99.

Gio Wiederhold, Sushil Jajodia, and Witold Litwin (1993): "Integrating Temporal Data in a Heterogenous Environment"; in Tansel, Clifford, Gadia, Jajodia, Segiv, Snodgrass: *Temporal Databases, Theory, Design and Implementation*; Benjamin Cummins Publishing, pages 563–579

Gio Wiederhold, Stephen Cross, Charles Channell (1994CC): *Productivity and Information Integration*; IEEE Educational Videotape, 2 hours, October 1994, Robert Kahrman, sponsor, IEEE, Piscataway NJ.

Gio Wiederhold (1994J): "Interoperation, Mediation, and Ontologies"; *Proceedings International Symposium on Fifth Generation Computer Systems (FGCS94)*, Workshop on Heterogeneous Cooperative Knowledge-Bases, Vol. W3, pages 33–48, ICOT, Tokyo, Japan, Dec. 1994.

Gio Wiederhold and Xiaolei Qian (1994Q): "Database Engineering"; in J.J. Marciniak: *Encyclopedia of Software Engineering*, Wiley-Interscience, Vol.1, pages 269–282.

Gio Wiederhold (1995D): Digital Libraries, and Productivity; *Comm. of the ACM*, Vol.38 No.4, April 1995, pages 85–96.

Gio Wiederhold (1995M): "Objects and Domains for Managing Medical Knowledge"; *Methods of Information in Medicine*, Schattauer Verlag, Vol.34, No.1/2, March 1995, pages 40–46.