Keynote presenation at CIKM 92, Baltimore MD, November 1992

Intelligent Integration of Diverse Information

Gio Wiederhold

DARPA
on leave from
Stanford University

Abstract

We will present the architecture of an intermediate processing layer for large information systems. Tasks assigned to that layer include managing access to information resources as databases, knowledge bases, sensor-based processing systems, message processing systems, and systems that project into the future, as simulations can do. To convert the data into information, it has to be abstracted, correlated, fused, and labeled with certainty factors.

Mediation requires domain-specific knowledge and we expect that there will be many specialized mediators maintained by experts in their fields. The role of artificial intelligence technology is critical here. An application will call upon as many mediators as the number of domains it addresses. Each mediator will utilize as many data resources as needed for its specific view.

An important motivation is the structuring of knowledge maintenance, and, by extension, the maintenance of large information processing systems as well. We will argue that longevity of large systems requires such an approach. The isolation from changes that an architecture with an intelligent layer provides is likely to be crucial for systems with many applications and data sources. We will cite some examples from a prototyping effort to insert this technology in a large, real and demanding application which has been initiated.

1. Introduction

Today there is a dichotomy among classes of computer software systems. There are a number of very large, carefully constructed systems to provide operational assistance to regular recurring tasks. Typical of these are payroll systems, inventory management systems, command and control systems, hospital systems, reservation systems, and other systems which focus on resource management of one type or another. These systems are long-lived, and require steady maintenance. Their staff focuses on continuity and reliability.

At the same time there are smaller systems which aid in less voluminous, but often more challenging tasks. In this category we place decision aids, design support, and such. Here innovation is prized, and may be traded against reliability. Maintenance becomes at least in part the responsibility of the user. Most intelligent, knowledge-based systems fall into this category.

The dichotomy of these two classes is frustrating, especially where the decision-making systems need access to the data resources in the operational systems. When response time is not an issue, useful data is extracted, massaged, and fed to the decision-making systems. Once external file transfer was common to bridge the gap. Today, network-based file transfer is common. Database access languages, as SQL, have been adapted to provide access to data by programs as well as directly to users [Date:89]. Extensions to such languages provide access with closure [Ullman:89] and to stored objects [Kent:90]. Experiments to provide access to knowledge sources are in progress as well [Neches:91]. However, the inherent differences in the operational paradigms of the two classes still cause a significant impedance mismatch. While performance problems are being mitigated by having better hardware, issues of control, management, and maintenance remain an issue.

This research was supported by DARPA N39-84-C211 the National Library of Medicine (LM04836), and NSF IRI-9007753.

2. Architectural alternatives

We will summarize three classes of architectural solutions, and focus on one which promises to provide both the capability to insert intelligent systems technology at a service level, and break the maintenance bottleneck that, as often reported, consumes 80% of corporate software costs.

2.1 Integrated systems

The first, and simplest, solution is to move to tightly integrated systems, where the decision aids become an integral part of the operational systems. Expanding the capabilities, for instance, of database management systems is one current direction [Stonebraker:91]. However, many decision aids require access to multiple databases. Integrating multiple operational systems into one, for the sake of the applications they serve, creates ever larger systems. Such large systems become extremely rigid, since any change is likely to affect many other applications and resources in turn. Although these systems, if large, are likely to be physically distributed, they do require central management and change authority.

An example of a large integrated system is the World-Wide Military Command and Control System (WWM CCS). A CCS keeps, essentially, an inventory of military resources. It was created in response to the problems that arose from having, prior to its establishment, 11 distinct CCSs. Planning for integration began in around 1966, using the best database technology of its time (GE-Honeywell IDS) and integration was achieved about 10 years later. WWMCCS now supports hundreds of applications and has exceeded its earliest specifications. However, it is viewed by many as a millstone on the military neck. No significant improvements are possible because of its size and the many interdependencies among its parts. A re-engineering effort, initiated in 1982, has had only minor impacts, and has been rescaled several times. At most, it is now a stable data module for newer applications that can extract data.

2.2 Federated Systems

A logical alternative is to avoid such a tight integration. Federated systems are based on a loose coupling of applications and data resources [Heimbigner:85]. Since the data resources remain autonomous, applications in such a scheme provide interfaces to the data resources they need. Legacy databases may be equipped with wrappers to permit their incorporation into networks and enable them to become servers.

Within federated systems the question of data dictionary management is paramount. Should there be one central data dictionary, or should each application maintain its own subset? In the former case, we again have a critical central node, which requires constant management as supporting subsystems change [Hsiao:92]. A reasonable estimate for autonomous operational systems is that, on the average, their schemas change about once every two years to accommodate new requirements and technology. Very large resource systems will change less often, being subject to the rigidity of integrated systems, while responsive and well maintained resources change more frequently. When the number of resource systems (r in Figure 1.) becomes large, maintenance of the dictionary and adapting the affected applications (d) again leads to large costs.

Since the resources in a federated system are autonomous, the actual occurrence of maintenance needs is not predictable. When maintenance is needed, experts must be gathered that understand the meaning of the change for the applications. After a change, resources will be unavailable for some time. Simple changes may be accommodated by automated mapping tools, as provided in SYSTEM R [CAKLMPSSSWY:81]. Automated approaches are triggered by schema changes. They assume a stable overall model, only minor schema perturbations can be handled. Insidious are semantic changes in resource data that do not lead to schema changes, say a decision to include value-added taxes in prices (an actual occurence in Switzerland last year).

Large federated applications are hence subject to the same maintenance costs as integrated systems, only their autonomous components are unaffected. The desire for autonomy also makes that updates to the resource systems are typically not provided. Any changes are submitted as independent transactions, and can be rejected by the autonomous systems [Ceri:91].

The second case, where the dictionary is kept application by application, avoids the rigidity somewhat. The total maintenance load is actually increased, because one resource change will affect multiple applications. Resource autonomy will still induce periodic unavailability of data, until maintenance has been performed. This maintenance is now the responsibility of application specialists, rather than central resources. Local staff are likely to be more responsive, but will also resent their being removed from their primary function to repair technical problems not of their own making.

2.3 Mediated Systems

We now attempt to resolve these problems with an architecture which places the technical management of integration in a separate layer. We propose a three-layer architecture of private applications, mediating information servers, and information resources which may be

	result $\longrightarrow$ decision making		cardinality
Layer 3	Independent **applications**	— managed on workstations	a
	- - - - network services to information servers - - - -		
Layer 2	Multiple **mediators**	— managed by domain specialists	m
	- - - - network services to data servers - - - - -		
Layer 1	Multiple **data resources**	— managed in server systems	r
	input $\longleftarrow$ real-world changes and simulations		

Figure 1: The three layers of this architecture.

databases, knowledgebases, sensor processing programs which report the current state, and simulations for projecting into the future.

Origins of the mediation concept are many [Wiederhold:92]. They include database views [Dayal:83], wrappers for legacy systems [Drew:92] database translators, object-generation systems [Barsalou:91], and active dictionaries [Weldon:85] [Rosenthal:89]. All these technologies remain valid, and can be subsumed in the mediation concept.

Mediators produce a level of information for applications similar to that now produced for end-users of freestanding AI programs. This transition is conceptually identical to the transformation of user-oriented database access, as exemplified by the original SEQUEL and QUEL database languages to the embedded SQL, which now dominates database use. The lack of a simple algebra makes the acquisition task harder, so that a KQML has to have more options for control of inference, errors, etc., than SQL provides.

There will be multiple mediators to effect a domain-specific partitioning. A mediator is staffed for its technical maintenance, and lives on its own network node, although it may be copied for greater performance to nodes closer to the resources or applications. Each mediator attempts to serve as many application as are in need of its domain information. Each mediator will access relevant views of as many data resources as are needed to produce the information in its domain.

A specific goal is that mediators, like most effective artificial intelligence programs, remain simple and constrained to a single ontology [Gruber:91]. Tasks requiring intelligence in the mediating layer encompass selection, focusing, pruning, fusion, ranking, mismatch processing, and data reduction. There are also control tasks associated with effective resource management. Results of mediation are to be composable by higher level applications, which also have to solve problems involving multiple heterogeneous subtasks.

3. Partitioning

The mediated architecture assumes a *horizontal* partitioning, where complex, domain-specific technical processing is allocated to a mediating layer. The mediators will draw upon resources from a voluminous base layer containing private and public resources. Applications will have to invoke multiple mediators, as sketched in Figure 1. We have described a demonstration system in [Wiederhold:90]. A large prototype is now being developed, which takes detailed manufacturing data on a new aircraft to produce information for global design and verification tasks. Initial applications planned are variability reduction of products, to reduce scrap and rework, and navigation to the multi-contractor design databases.

3.1 Resources

To clarify the tasks that are needed in the mediator layer we summarize the types of resources which are available. Many of the resources are likely to be massive. They must use representations that are effective for algorithmic approaches. Typical are databases, large bodies of text, and simulations of physical processes using spatial or four-dimensional (space and time) representations. The resources, however, are active engines, to be invoked by modules in other layers. The processing in the resource modules is rarely very complex, but has to deal effectively with large volumes. Where efficiency is a concern algorithmic processes will often be preferred. However, the results of this basic algorithmic processing are rarely directly suitable for high-level decision-making.

3.2 Mediators

The power of knowledge derives from its application and reapplication to the changing world. The resource modules provide data about these instances and events, so that the intelligent programs are relieved of the task of managing volumes of detail. Out of that volume they must extract manageable amounts of selected, focused, but possibly complex results. We use the term *Mediation* to cover a wide variety of functions that enhance stored data prior to their use in an application.

Mediators have an active role, their function can-

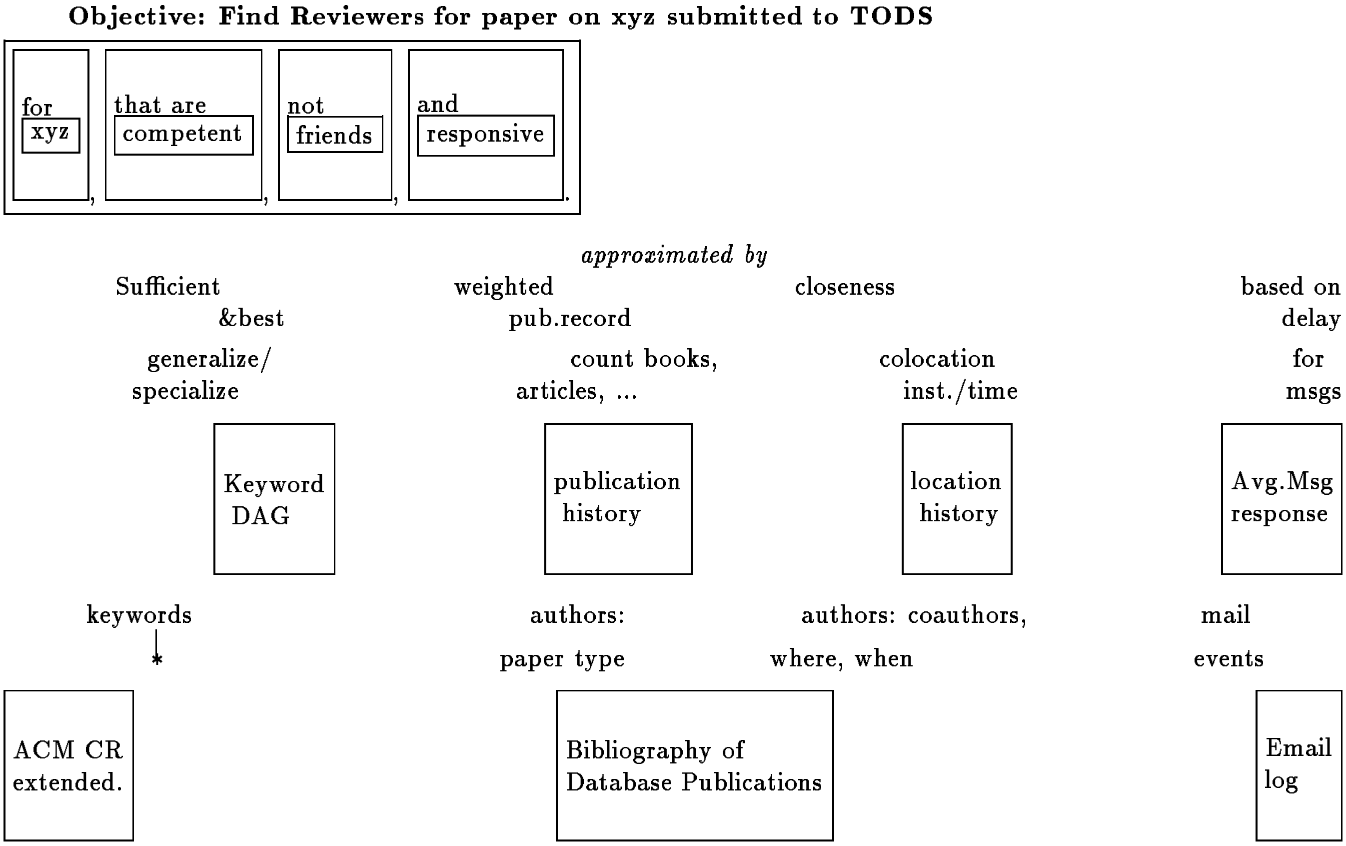

Figure 2: An example with mediators

not be achieved by having simple interface specifications or standards. Mediators may also store intermediate results, but then must manage the consistency of such caches versus the base data. Mediators also adapt to changes in the base resources. Since the changes relevant to a mediator are constrained to be domain-specific views, it becomes feasible to assign this maintenance to a technical domain specialist. Maintenance by committee is avoided.

If applications require information not presented by an existing or, a new mediator is created if changing an existing one would affect existing active applications. Often such new mediators can be built by reuse of existing ones.

To be effectively used by applications, mediators must be clear. Clarity can be achieved by:

1 Simple and well-defined ontologies

2 Formal processing methods over the data corresponding to these ontologies

We see them profiting from the more formal directions of artificial intelligence research.

3.3 End-user applications

We envisage end-users invoking applications at their workstations. Applications obtain information by dealing with the abstractions supported by the mediators. Different tasks will require a different set of such mediators, but in each application the number of domains, and hence the number of mediators to be accessed, remains manageable. If mediators are effective, the application maintainer can focus on functional enhancements. The application functions may use artificial intelligence technology as well. Since they do not provide information to other systems, the constraints placed on mediators in regard to formality and consistency can be relaxed. The applications can be pragmatic, to provide maximal benefits to the end-user. Since they delegate much of the base tasks to mediators, they can be much smaller, enhancing their maintainability.

A language will be needed to provide flexibility in the interaction between the end-user's workstation and the mediators. We discussed the partitioning of artificial intelligence paradigms into pragmatics (at the user-workstation layer) and the formal infrastructure (in the mediation layer) in [Wiederhold:90].

A simple example of mediation is shown in Fig.2. One objective, finding suitable reviewers for a journal, is broken into four phrases, and each of these phrases is processed by a mediator. Each mediator has a distinct ontology: an internal vocabulary, a data structure, and a processing style. The data are obtained from three sources; one source, the annotated bibliography, is used three times in different views. The only ontological term they need to share is that of a reviewer.

3.4 Vertical partitioning.

At each of those layers will be a vertical partitioning. There will be multiple user applications, each using various configurations of mediators. Each of the mediators in turn will use distinct views over one or several base resources. The vertical partitioning creates a hierarchical binding only for one application instance, but not for the overall system structure. Just as databases today are justified by a diversity of shared usage, mediators should be sharable. Note that today's expert systems are rarely modular and sharable, so that their development and maintenance cost is harder to amortize.

For instance, the mediation module which can deal with inflation adjustment can be used by many applications. The mediator which understands postal codes and town names can be used by the post office, express delivery services, and corporate mail rooms.

We foresee here an incentive for a variety of specialists to develop powerful, but generally useful mediators, that can be used by multiple customers. Placing one's knowledge into a mediator can be more rapidly effective, and perhaps more rewarding, than writing a book on the topic.

3.5 Interfaces

We have assigned modules functionally to three layers. Only the application modules need to interact with end-users. Mediator modules need machine-friendly interfaces to support the applications, acquisition interfaces to obtain information from the base resources — which are often pre-existing, so that some of these interfaces have to adapt to legacy systems, and an expert's interface for the management of the knowledge they contain.

Databases commonly have interfaces that can be used by other computing modules, with some standards (as SQL), that were unfortunately specified before sufficient experience was gathered in composite systems. Simulations, sensor-based processing, and knowledge-based systems have less well established interfaces, although work in this area is in progress [Neches:91].

All these components are positioned along a *data highway*, provided by modern communication technology. The interactions of the components will be primarily constrained by logical and informational limitations, and not by physical linkages.

The partitioned architecture is well suited to the modern multi-processor networked environment. This model of computation assumes a resource-rich environment, as envisaged in [HPCC:91]. There is no global optimization over all the modules, outside of the adaptation caused by applications rejecting use of overloaded modules. Any optimization is oriented towards application performance, the critical issue in large-scale software.

4. The Research Tasks

Defining an architecture which places the technical management of integration in a separate layer is intended to give integration issues a focus of their own. To support intelligent integration technology we must also draw upon a scientific base, an *integration science*. Many of the elements needed for an integration exist in the arsenal of artificial intelligence and related computer science. Their evaluation and development will be critically affected by placing them within larger systems science objectives, rather than as stand-alone efforts.

Critical tasks are abstraction and fusion of data [Wiederhold:92I] . Abstraction requires formal models for its automation. Measures of significance and utility will drive the interpretation of those models. When abstraction levels match, fusion can occur [Cohen:87]. However since abstractions will rarely match, mismatches must be recognized and resolved [DeMichiel:89]. Reporting such mismatches must consider the uncertainty due to mismatches. Further uncertainty is created when the mediator assesses the reliability of the base resources [Lee:92] or when aggregations do not match the models [Rundensteiner:89].

The critical element missing today in such an architecture is the high-level language needed to compose applications from large scale modules, although we see some research results in application composition from large objects being relevant.

To create a library of such modules we can initially wrap existing programs to serve within this architecture. The interface code to make them user-friendly can be omitted, greatly reducing their complexity. We foresee that the input-output code will move to the end-users' workstations, which will then communicate with the root nodes of the application tasks [WWC:92], [Green:91].

5. Conclusion

The architecture provides a basis for the insertion of intelligent services into large systems. The essential differences for maintenance of base resources, technical support, and pragmatic user applications are recognized and assigned to their own levels. As interface capabilities improve it will become easier to improve and replace modules, and in that manner assure long-lived effectiveness of the overall systems. Legacy resources can be integrated at a relatively low cost, enabling their immediate exploitation. As actual interoperation becomes a way of life, the motivation to come to a consensus regarding interface standards and domain ontologies increases. But even with standards we must assure that we have systems where growth and change is supported at least as strongly as current cost-effectiveness and reliability.

References

[Barsalou:91] Thierry Barsalou, Niki Siambela, Arthur M. Keller, and Gio Wiederhold: "Updating Relational Databases through Object-Based Views,"; *ACM SIGMOD*, Denver, CO, May 1991.

[CAKLMPSSSWY:81] D.D. Chamberlin, M.M. Astrahan, W.F. King, R.A. Lorie, J.W. Mehl,J.W, T.G. Price, M. Schkolnick, P.G. Selinger, D.R. Slutz, B.W. Wade, and A. Yost: "Support for Repetitive Transactions and Ad Hoc Queries in SYSTEM R"; *ACM TODS*, Vol.6 No.1, Mar.1981, pp.70–94.

[Ceri:91] Stefano Ceri, Maurice A.W. Houtsma, Arthur M. Keller, Pierangela Samarati: "A Classification of Update Methods for Replicated Databases," Stanford University, Computer Science Technical Report STAN-CS-91-1392, October 1991.

[Cohen:87] P.R. Cohen, G. Shafer, and P.P. Shenoy: "Modifiable Combining Functions"; *Uncertainty in Artificial Intelligence*; Proc.of an AAAI Workshop, Jul.1987.

[Date:89] C.J. Date: "A guide to the SQL Standard"; Addison-Wesley, 1989.

[Dayal:83] U. Dayal and H.Y. Hwang: "View Definition and Generalization for Database Integration in MULTIBASE: A System for Heterogeneous Databases"; *IEEE Trans. Software Eng.*, Vol.SE-10 No.6, Nov.1983, pp.628–645.

[DeMichiel:89] Linda DeMichiel: "Performing Operations over Mismatched Domains"; *IEEE Transactions on Knowledge and Data Engineering*, Vol.1 No.4, Dec. 1989.

[Drew:92] Pam Drew, Roger King, and Dennis Hemibigner: "A Toolkit for the Incremental Implementation of Heterogeneous Database Management Systems"; *VLDB Journal*, Volume 1, Issue 2, 1992.

[Green:91] M. Green and R. Jacob: "Software Architectures and Metaphors for Non-WIMP User-Interfaces"; Computer Graphics, Vol.25 No.3, Jul.1991, pp.119–235.

[Gruber:91] T. R. Gruber: "The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases"; in Allen, J.A., Fikes, R., and Sandewall, E. (Eds) *Principles of Knowledge Representation and Reasoning*, Morgan-Kaufmann, 1991.

[Heimbigner:85] D. Heimbigner and D. McLeod: "A Federated Architecture for Information Management"; ACM TOIS, vol.3 No3, 1985.

[Hsiao:92] D. Hsiao: "Tutorial on Federated Databases and Systems"; *The VLDB Journal*, Vol.1 no.1, July 1992, pp.127–179.

[HPCC:91] Committee on Physical, Mathematical, and Engineering Sciences: "Grand Challenges: High Performance Computing and Communications" Supplement to the President's FY 1992 Budget, 1991.

[Kent:90] William Kent: "Important Features of Iris OSQL"; *Object-Oriented Task Group Workshop*, Atlantic City, May 1990 (NISTIR 4503).

[Lee:92] S. Lee: "Imprecise and Uncertain Information in Databases: An Evidential Approach"; *IEEE DE 8*, Tempe, AZ, Feb 1992.

[Lewis:91] J.W. Lewis: "Wrappers: integration Utilities and Services for the DICE Architecture"; in *Proc.of the 2nd nat.Symp. on Concurrent Engineering*; Cocurrent Eng.Res.Ctr., Morgantown WV, 1991, pp.445-457.

[Neches:91] Robert Neches, Richard Fikes, Tim Finin, Tom Gruber, Ramesh Patil, Ted Senator, and William R. Swartout: "Enabling Technologies for Knowledge Sharing"; AI Magazine, Vol.12 no.3, Fall 1991.

[Rosenthal:89] A. Rosenthal, U.S. Chakravarthy, B. Blaustein, and J. Blakely: "Situation Monitoring for Active Databases"; *VLDB 15*, Amsterdam, Aug. 1989.

[Rundensteiner:89] E.A. Rundensteiner, and L.Bic,: "Aggregates in Possibilistic Databases"; *VLDB 15*, Amsterdam, Aug. 1989.

[Stonebraker:91] M. Stonebraker and G. Kemnitz: "The POSTGRES Next-Generation Database Management System"; Comm.ACM Vol.34

No.10, Oct. 1991, pp.110-120.

[Ullman:89] Jeffrey Ullman: "Principles of Database and Knowledge-Base Systems, Vol. II: The New Technologies"; CSP, New York, 1989.

[Weldon:85] Jay-Louise Weldon: "The Case for Active Data Dictionaries: Lessons from the Microcomputer World"; *Inf.Sys.Manag.*, Vol.2 No.3, 1985, pp.42–45.

[Wiederhold:90] G. Wiederhold, P. Rathmann, T. Barsalou, B-S. Lee, and D. Quass: "Partitioning and Combining Knowledge"; *Information Systems*, Vol.15 no.1, 1990, pp. 61–72.

[Wiederhold:92C] Gio Wiederhold: "Mediators in the Architecture of Future Information Systems"; *IEEE Computer*, March 1992.

[Wiederhold:92I] Gio Wiederhold: "The Roles of Artificial Intelligence in Information Systems"; *Journal of Intelligent Systems*; Vol.1 No.1, 1992.

[WWC:92] Gio Wiederhold, Peter Wegner, and Stefano Ceri: "Towards Megaprogramming"; *Comm.ACM*, Nov. 1992.