Mediation in Information Systems

GIO WIEDERHOLD

Department of Computer Science, Stanford University, Stanford, California
⟨gio@db.stanford.edu⟩.

Information systems can be defined as computing systems that provide information for their customers, often in a decision-support role. They are a large and increasing fraction of modern computing, slowly eclipsing the effort expended on massive data processing, cycle-intensive scientific computing, and critical real-time systems. They are also quite visible, since they have a large number of direct users. However, they do not stand alone.

Planning and other decision-making tasks require information from diverse sources such as databases, reference systems, data obtained in real time from sensors, and analyses projecting trends over the totality. The databases are mainly part of operational systems, built to satisfy local objectives. Useful data are available from public services, both free and for-profit. Processing of sensor inputs and simulations requires the technologies of scientific computing and produces results in widely different formats. Most of these resources are autonomous with respect to decision-making tasks. Autonomy, in turn, leads to heterogeneity, including representation, scope, level of abstraction, and context.

Rapid construction of information systems is being aided today by the many tools provided by vendors of client-server systems. A furious race is under way to establish standards for such *middleware*, since adoption of standards is expected to direct much future business into one camp of vendors or another. Some vendors openly push for certain standards, others stay on the fence, and those that have sufficient resources promise to be compatible with everybody. Candidate middleware interface conventions go by such names as CORBA (Common Object Request Broker Architecture), DOE (Distributed Objects Everywhere), ILU (Inter-Language Unification), KQML (Knowledge Query and Manipulation Language), OLE (Object Linking and Embedding), OpenDoc (Open Document Exchange), PDES (Product Data Exchange using STEP), and many others. Descriptions of these interfaces are found primarily in the commercial literature; some can be located by surfing the Internet starting at the references given at the end of this paper. The major element they have in common is that they all promise to replace SQL.

An example of a composed decision-support system is one that helps a transportation planner for heavy equipment. The task of, say, shipping a desalinization plant to a country in the Near East requires access to airfreight schedules and capacities, port and warehouse information, weather predictions, personnel for assembly, and the like. The sales office needs some of that information for pricing, and the exact configuration to be shipped also depends on some of that information, as well on data about local climate conditions. Very little code will be written in developing such software,

but much effort will be expended in accessing and integrating the resources so that reliable support can be provided.

The academic community has largely stayed outside this hubbub. However, as these standards solidify, academic directions will also be affected. For instance, software-building processes are now taught based on top-down concepts, but these will become much less relevant when practitioners build application systems rapidly by bottom-up composition. Even now substantial object libraries are becoming available that encapsulate the beautiful data structures we teach our students.

The basic client-server model also has its weaknesses. The available standards are simplistic and provide little support for heterogeneity and asynchronous operation. Large systems built in client-server fashion are as hard to maintain as large homogeneous systems. For instance, a change required by a customer in a client application can require changes in one or more server databases, and making those changes can again affect an untold number of other clients. This coupling problem closely mimics the problems encountered with large systems everywhere. The frustration that about 80 percent of software costs are in maintenance is not addressed by making systems faster to build.

A logical evolution of client-server architecture is provided by *mediated architectures*. In mediation, an extra software layer is inserted between the client and the server. This layer breaks the coupling. The intermediate modules, *mediators*, bring source information into a common form. Mediators are domain-specialized so that their maintenance can be focused. Applications requiring information from multiple domains, say budget and inventory, would use two mediators, corresponding to the allocation of responsibility in the enterprise. The mediators, in turn, can each access several databases or other resources. Such resources can, of course, be shared by multiple mediators when their contents overlap multiple domains. A mediator may have to use multiple standards to access its resources but can present a single interface to the client.

Because complexity of relationships is an important factor in system maintenance, the regular structure of mediation reduces the cost of maintenance growth to a linear function of the number of modules from the OB-squared effort in an arbitrarily connected network. However, the number of modules is larger, so benefits are only expected in systems that have more than thirty modules. The ease of composition brought about by modern software technology makes the building of such large systems more convenient and their existence more likely.

The mediators themselves should also remain simple. Domain specialization avoids the tendency to have committees and their compromises drive software specification. Information-processing tasks in mediators include accessing of appropriate resources, data selection, format conversion, bringing data to common abstraction levels, matching and integration of information from distinct sources, and preparing information and descriptive metainformation for shipping to the customer's workstations, including focusing, pruning, and summarizing. The general objective is to provide as much useful information in as compact and relevant an information package as feasible. The actual display and multimedia services belong in the user's workstations, as do any domain-spanning analyses.

Mediated architecture is also attractive to academia. Large, realistic databases are hard to manage in an academic setting, and too many of our exercises work with the simple employee-manager-job model and a dozen entries. Building effective user interfaces is also an area where industry has excelled and, with a few exceptions, little progress can be expected from the scale of effort and background available in academia. Mediators, being mainly code, are easier subjects for academic research.

Mediating modules do raise many questions related to their generation,

their maintenance in the face of change, and the innovation and power of their value-added services. Today most mediators are handcrafted. We look forward to the day when formal knowledge, in the form of resource schemas, user models, and functional specification, can be used to generate mediators or at least help in their maintenance. Questions of efficiency and allocation in networks will be challenging. Optimization requires the development of algebras that permit the manipulation of access to heterogeneous resources. High-level functional descriptions can evolve to a new language level. There is a rich mine of problems here, and linkages to industrial resources and needs are plentiful. A number of projects are now using the mediator concept, and some standards are being propagated for the interfaces that are required.

References. Mediation is an active and recent topic. Convenient access to ongoing work is gained by surfing the Internet, starting at http://isx.com/pub/I3. An early reference, with many backward pointers, is Gio Wiederhold: "Mediators in the Architecture of Future Information Systems," *IEEE Computer 25*, 3 (March 1992), 38–49. Technologies for interfacing intelligent mediators are described in Michael Genesereth and Steven Ketchpel: "Software Agents," *Commun. ACM 37*, 7 (July 1994), 48–53, 147. Several related papers appeared in Nabil Adam (Ed.): *Proceedings of the Third International Conference on Information and Knowledge Management* (Gaithersburg, MD, Nov. 1994), ACM, New York, and others will appear in Leo Mark (Ed.): *Database Application Semantics*, IFIP WG2.6, Sixth Working Conference on Database Semantics (DS-6), Atlanta, Ga., May 1995, Chapman and Hall, London (to appear). Papers on heterogeneous databases are found in A. Elmagarmid, M. Rusinkiewicz, and A. Sheth, *Heterogeneous Distributed Databases*, Morgan Kaufmann, San Mateo, Calif., 1995. Some opportunities for new mediating services are described in Gio Wiederhold: "Digital Libraries and Productivity," *Commun. ACM 38*, 4 (April 1995), 85–96. A special issue, dedicated to mediation, of the *J. Intelligent Inf. Syst.* is due to appear in May 1996.