MEDIATOR: INFORMATION AND KNOWLEDGE MANAGEMENT FOR THE VIRTUAL FACTORY

Brian R Gaines and Douglas H Norrie

Knowledge Science Institute and Division of Manufacturing Engineering
University of Calgary
Calgary, Alberta, Canada T2N 1N4
gaines@cpsc.ucalgary.ca, norrie@enme.ucalgary.ca

Abstract

Mediator is an open architecture information and knowledge management system designed to provide a flexible technology to support the management of complex manufacturing activities throughout the product life cycle. A heterogeneous environment is assumed in which the sub-systems are geographically dispersed and involve different application packages, not necessarily designed to work together, multiple platforms, protocols and forms of user interface. The function of Mediator is to provide a knowledge support system for all those involved in the manufacturing process from requirements through design, engineering, production, to maintenance and recycling. It is designed to facilitate communication, compliance with constraints including physical restrictions and legal obligations, and to generally represent knowledge *about* any activity or sub-system relevant to the manufacturing process.

The Mediator architecture is a distributed client, distributed server design, in which multiple users can collaborate synchronously or asynchronously through processes running anywhere on the network. Much of the user interaction is based on an open architecture visual language tool for representing conceptual schema in a way that is comprehensible to users, and can be easily tailored to different application domains. Mediator supports semantic networks that can be used to represent rules, procedures and constraints with formal, operational semantics such that an inference system can be used to give advice and check for constraint violations.

The representation is domain independent and the system can reason with legal constraints and corporate procedures as well as with design constraints. Less formal concept maps are also supported where the role of the system is primarily information retrieval rather than reasoning, and the formal and informal systems can be combined. Procedures can be triggered through the user interface to the visual language, and the procedures can include other applications or a general-purpose 'agent' system that supports active integration between applications and between sites. Data translation and knowledge interchange are supported through such agents.

INTRODUCTION

In modern manufacturing environments the 'shop floor' is no longer a well-defined location but rather a virtual space in which related manufacturing operations may be widely distributed geographically and yet closed linked conceptually in terms of dependencies and material, information and knowledge flows (Gaines, 1992). In particular, the flows relating to the physical manufacturing operations are only one part of a complex flow of information and knowledge concerned with underlying requirements, the logic of design decisions, the availability of inventory, contractual obligations relating to customers, suppliers, employees and government agencies, and so on. Information and knowledge technologies have come to play major roles in supporting and managing these flows, but the heterogeneity of the many different sub-systems involved continues to make the total management of manufacturing a complex problem. In general, manufacturing systems tend to function reasonably well under the routine conditions for which they have been developed but become problematic when situations arise that deviate from these conditions. Under abnormal conditions the information available in the system may be irrelevant or misleading, and background knowledge that would be valuable in rectifying the situation is likely to be absent.

Mediator is an open architecture information and knowledge management system designed to provide a flexible technology to support the management of complex manufacturing environments. A heterogeneous environment is assumed in which the sub-systems are geographically dispersed and involve different application packages, not necessarily designed to work together, multiple platforms, protocols and forms of user interface. The function of Mediator is to provide a knowledge support system for the managers and system operators involved in running a virtual factory. It is designed to facilitate communication, compliance with constraints including physical restrictions and legal obligations, and to generally represent knowledge *about* any activity or sub-system relevant to the manufacturing process.

The Mediator design and prototype is one outcome of the *knowledge systematization* technical work package of test case 7, GNOSIS, of the international Intelligent Manufacturing Systems (IMS) pre-competitive research program. GNOSIS involves over 100 participants in 31 industry and university organizations in 14 countries, with the objective of developing a *post mass production manufacturing paradigm* involving *reconfigurable artifacts*. The knowledge systematization work stream has been concerned with the modeling and management of information and knowledge flows throughout the complete product life cycle from initial needs through design, engineering, production, to reuse and recycling. A study using knowledge acquisition tools to define and analyze major policy and technical issues in GNOSIS itself is being reported at AAAI94 (Gaines and Shaw, 1994). This paper focuses on how some of the concepts developed in GNOSIS have been implemented in Mediator as a general knowledge systematization tool.

MEDIATOR FUNCTIONALITY

Mediator is a system designed to coordinate the overall manufacturing process over the complete product life cycle from requirements to recycling. In particular, it supports the integration of computer-based system design, production engineering and production sub-systems into a mutually collaborative framework for integrated design and manufacture. It provides facilities for recording and tracing decisions taken at each stage of the product life cycle, particularly the dependencies between knowledge, decisions, datasets, and so on.

The basic Mediator technology is an open architecture visual language tool for representing conceptual schema in a way that is comprehensible to users, and can be easily tailored to different application domains. Mediator supports semantic networks that can be used to represent rules, procedures and constraints with formal, operational semantics such that an inference system can be used to give advice and check for constraint violations. The representation is domain independent and the system can reason with legal constraints and

corporate procedures as well as with design constraints. Less formal concept maps are also supported where the role of the system is primarily information retrieval rather than reasoning, and the formal and informal systems can be combined.

Procedures can be triggered through the user interface to the visual language, and the procedures can include other applications or a general-purpose 'agent' system that supports active integration between applications and between sites (Kwok and Norrie, 1993b). Data translation and knowledge interchange are supported through such agents.

It is assumed that many of the systems to be integrated are geographically dispersed and have been developed separately without any coordination. The problem is one of heterogeneous integration to provide a powerful and usable combined system that:

i) Enhances the functionality of each of the individual sub-systems
ii) Offers users additional capabilities beyond those of the individual sub-systems
iii) Provides users with an overall coordination system that is readily comprehensible.

It is a fundamental design requirement that the basic Mediator technology should be minimal, modular, open-architecture, cross-platform, and easy to enhance, maintain and use. Mediator is only justified to the extent that it is resource-effective in improving the coordination of complex situations, and would be counter-productive if it introduced new problems and complexities.

MEDIATOR ARCHITECTURE

The Mediator architecture is a distributed client, distributed server design, in which multiple users can collaborate synchronously or asynchronously through processes running anywhere on the network. A heterogeneous environment is assumed in which there are multiple protocols and multiple forms of user interface.

Figure 1 shows the basic architecture. At the center is a collaborating and geographically dispersed user community. Beneath this is shown the computational infrastructure to support the collaboration. The community interacts with the system through a variety of forms of user interface, typically graphics, text, visual languages and hypermedia. They access a variety of 'applications' defined as packages of functionality providing defined services. The functionality is made operational by initiating processes which may run anywhere on the network; that is, remote procedure calls are expected to be common. Inter-process communications is provided through local and wide area networks supporting a range of generic and proprietary knowledge and data interchange formats.

On the left of Figure 1, these four layers are shown instantiated in terms of conventional software packages designed independently of mediator: as standard interfaces, standard applications, application-specific processes and application-specific communications. It is assumed that such standard applications can play a role in a Mediator-coordinated systems, minimally by the application and its datasets being registered in Mediator with its application windows open on the Mediator desktop, and maximally by Mediator controlling its inputs, outputs and operations by job-control scripts. Existing application software is assumed to play a major role in the operation of Mediator, and one can conceptualize this role by noting that the Mediator system will know a lot about such applications while they will know virtually nothing about Mediator.

At the top of Figure 1, the four layers are shown instantiated in terms of core Mediator-specific technology, what might be called a Mediator-shell since the majority of the software is not specific to any particular application of Mediator, but provides general collaboration and integration facilities. The primary user interface to Mediator is through visual languages allowing the general representation of semantic systems through graphical representations. This will be supplemented by hypertext and hypermedia as appropriate. The application layer uses particular instances of these visual languages with application-specific semantics and a

visual appearance designed to be natural and easy to use in the context of the specific application. The process layer supports 'agents' as light-weight processes triggered by user interaction with the visual languages, and also triggered by other agents (Kwok and Norrie, 1993a). The communication layer supports object-oriented protocols for knowledge and data interchange such as STEP, Ontolingua and KQML (Finin, Weber, Wiederhold, Genesereth, Fritzson, McKay, McGuire, Shapiro and Beck, 1992). It also supports messages in arbitrary formats as appropriate to communication with other applications not designed with Mediator in mind.

On the right of Figure 1, the four layers are shown instantiated in terms of separately designed applications that are 'Mediator-aware' to some extent, for example in using the Mediator interface technology, applications, agents or protocols as part of their normal operation. The Mediator shell technology is being designed to be highly modular and readily integrated in whole or in part with existing applications. Thus, in an engineering domain, one might envision a Mediator graphic interface and data interchange protocols being integrated with functional design tools such as HUT's Δ (Gui and Mäntylä, 1993) or University of Tokyo's SYSFUND (Ishii, Tomiyama and Yoshikawa, 1993).

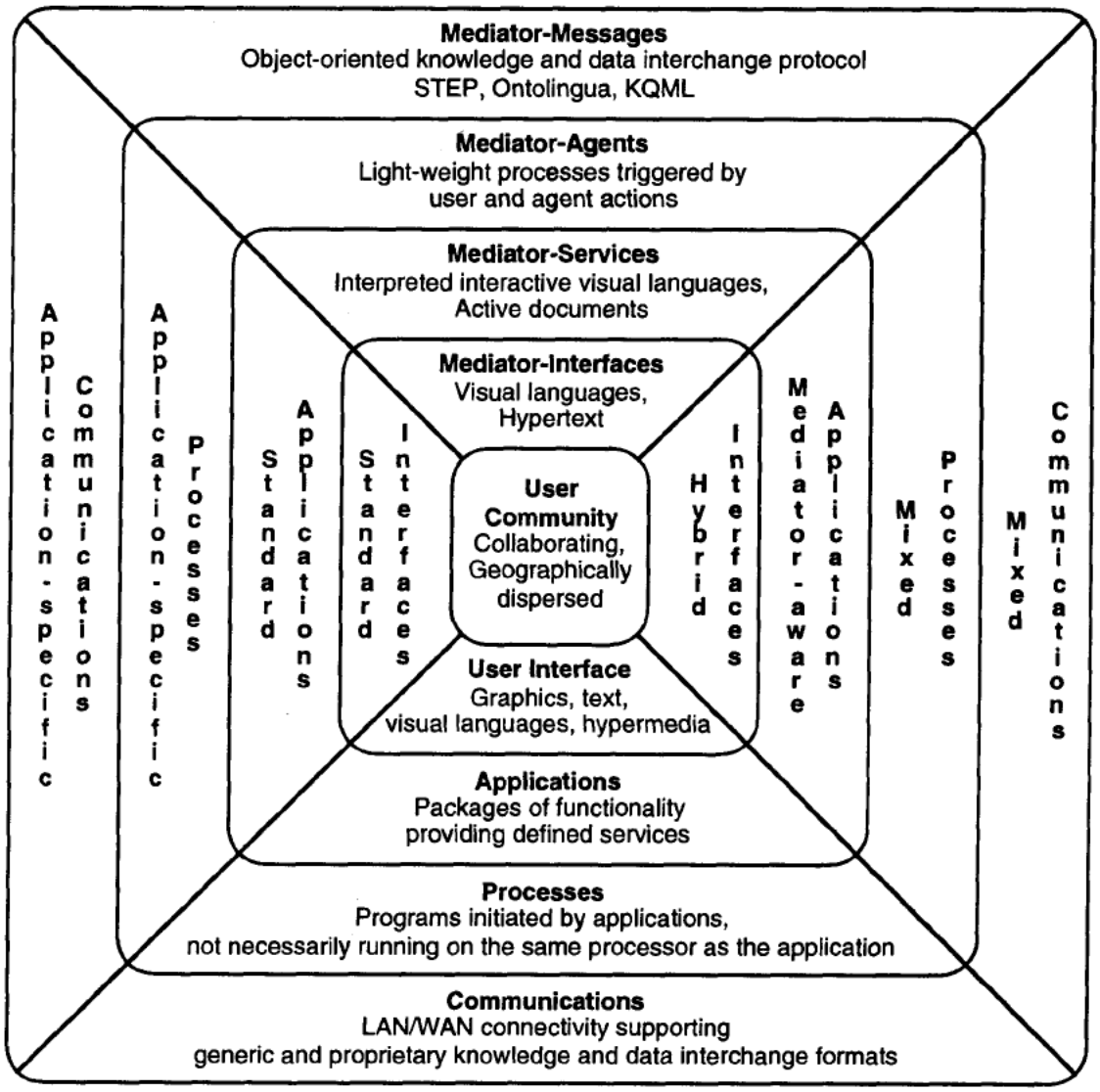

Figure 1 Mediator architecture

MEDIATOR VISUAL LANGUAGE

The primary user interface to Mediator is through an open architecture visual language system developed to support graphic interaction with computers on a wide range of topics including knowledge representation, concept mapping, Petrinets, bond graphs, and so on. The system was developed as an alternative to textual interfaces to take advantage of modern graphic workstations through a simple and natural visual language of great generality. The generality is achieved because the underlying data structure is a sorted digraph which can represent typed binary relations and hence virtually any mathematical structure. The provision of simple graphic expression for equality, inequality and subgraph operations allow complex conditions on graphs to be represented in a natural way. An attractive and comprehensible user interface is provided through the capability to reflect the type structure in node decorations such as fonts, shading and color.

User interaction with Mediator takes place through the creation of statements in the visual language, and through interaction with existing statements through popup menus whose content is specific to node type. The actual action initiated is context-sensitive: to the node selected for the popup, to nodes linked to it, and to other nodes preselected by clicking on them in the graph. This allows both simple and complex activities to be initiated by simple and comprehensible user actions. For example, Figure 2 shows a shop floor information access concept map being used to access bill of materials information for a particular product. The general concept map tool allows the style and functionality of such windows to be tailored to a particular class of situations. Individual maps can then be generated automatically or manually for particular situations, in this case a product with two possible line layouts. The user interface is also open architecture and context specific, and can be programmed through a wide range of possible dialogs. In this case a popup menu gives access to the bill of materials, possibly through another concept map, or by initiating action with the appropriate database.

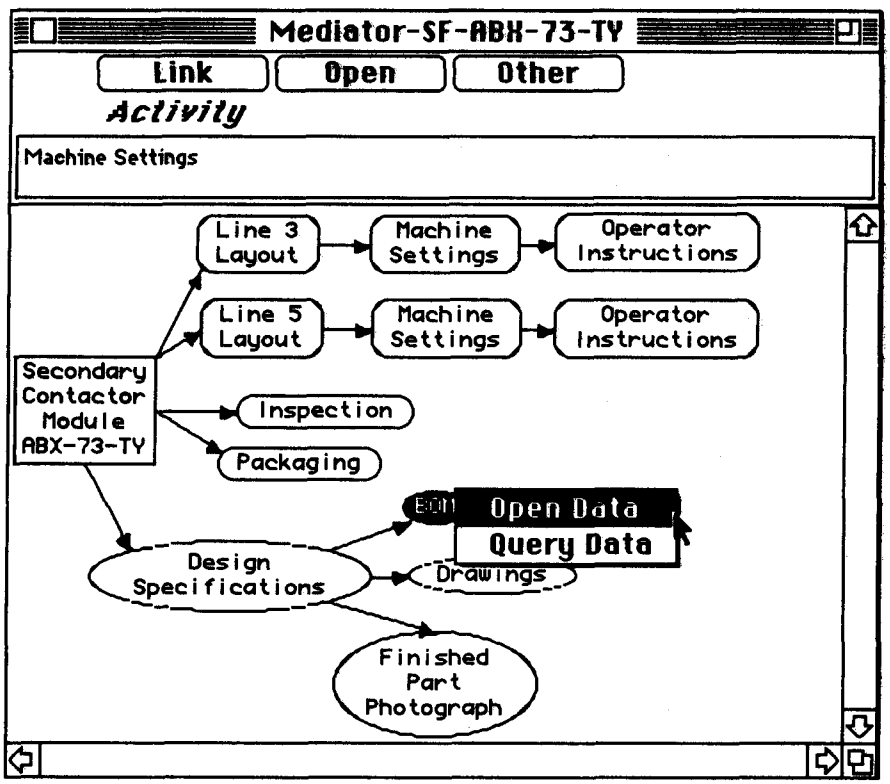

Figure 2 Interacting with a shop floor information map

The visual language provides a very general and easily customizable interface to the underlying knowledge and data structures, and processes operating on them such as specific applications and general agents, including communications with, and remote procedure calls to, other systems. The language may be used as a 'wrapper' to existing applications available only in binary form, and it may also be used as an embedded component of other applications available in source form. A computer-supported cooperative work approach has been adopted from the outset so that maps may be shared across local and wide area networks and used for distributed project coordination.

INTEGRATION WITH OTHER SYSTEMS

The primary objective of the Mediator development is to provide a 'shell' technology for coordinating collaborative manufacturing projects across networks. However, the open architecture, modular, networked, distributed client, distributed server technology underlying the Mediator implementation is well-suited to vertical application with applications as shown on the right of Figure 1. The development of 'Mediator-aware' applications is attractive not only in increasing the functionality of Mediator, but also in allowing rapid prototyping of new applications. The effort required to develop good quality user interfaces is a major component of resource usage in most application development, and the use of the generic Mediator interfaces and protocols can increase the speed, and decrease the cost, of specialist system development.

One reason a generic visual language was chosen as a general-purpose interface is because graphical representations are used in many branches of science and engineering, and the Mediator interface is to be able to emulate many such representations through a single software system. For example, a research group focusing on optimal scheduling and rapid shop-floor reconfiguration could develop algorithms that interface to Mediator for purposes of user interaction, and run on a high-performance server anywhere on the network. The existing Mediator graphic representation facilities could be used to provide an iconic interface giving a geometric, or more abstract, representation of the configuration and material flows. However, the primary project resources could be applied to the development of improved scheduling and reconfiguration algorithms and not diverted to user interface aspects of the system which, while very important, are not the primary focus of the research. For example, Figure 3 shows a bond graph from HUT's Δ functional design system (Gui and Mäntylä, 1993) represented in the visual language, and Figure 4 shows a partial behavioral model from University of Tokyo's SYSFUND functional design system (Ishii et al., 1993) similarly represented. The generic Mediator data structures can be imported from, and exported to, the specific systems through simple converters.

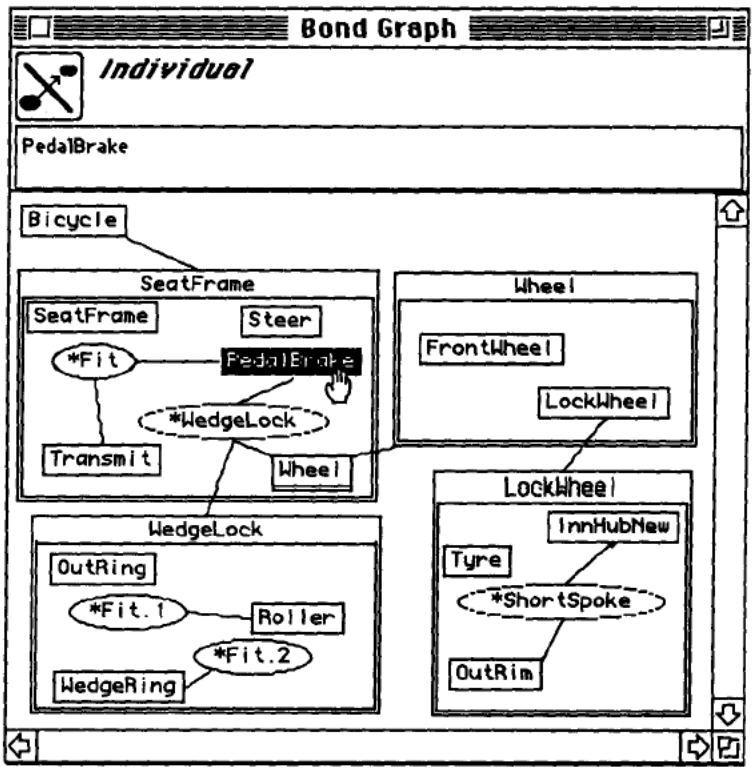

Figure 3 A Δ bond graph in Mediator

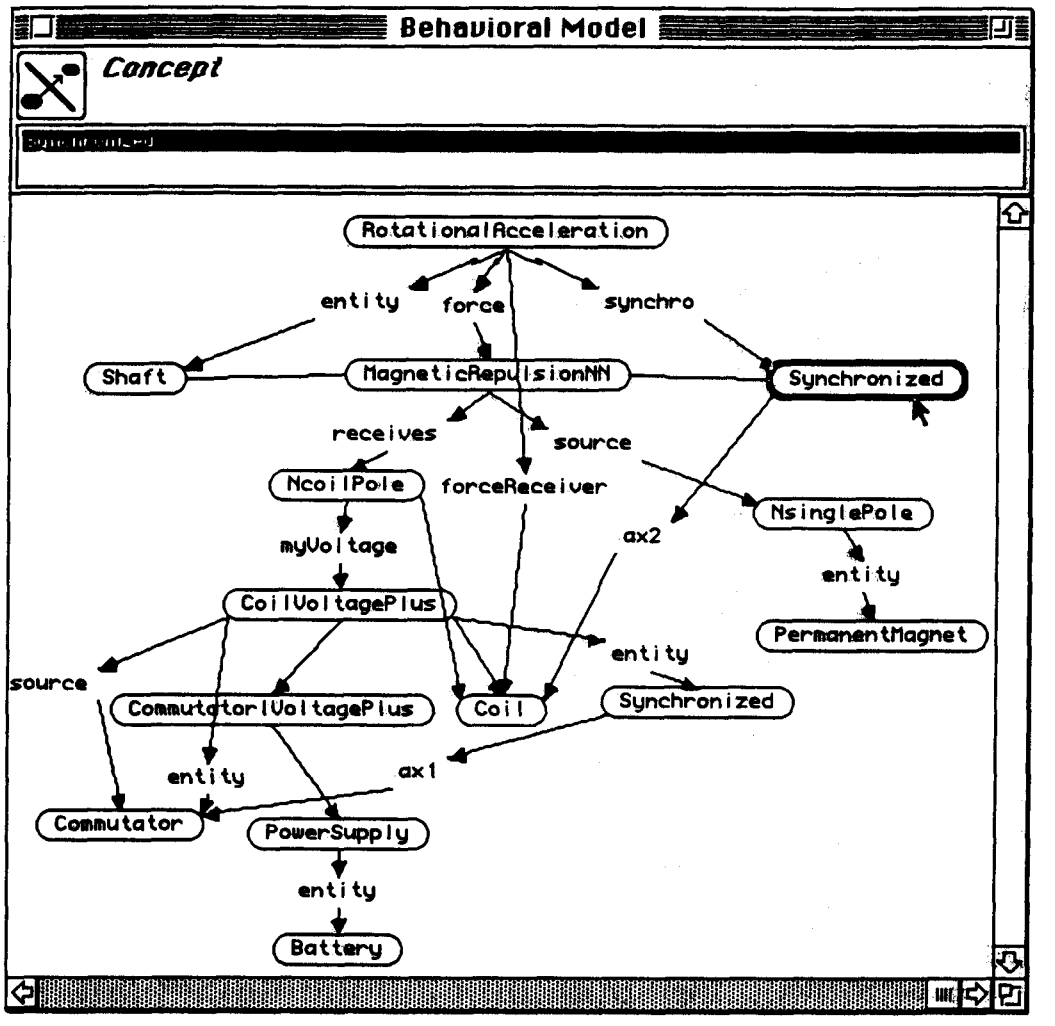

Figure 4 A SYSFUND behavioral model in Mediator

A semantic network visual language for Mediator is already available (Gaines, 1991b) that compiles into a KL-ONE-like knowledge representation system (Gaines, 1991a; Gaines, 1993a) and may be used to represent computational knowledge structures (Gaines, 1993b). This has been used to operationalize a corporation's procedures manuals (Kremer, 1991) in a way that is directly applicable, for example, to the regulatory aspects of shop floor management.

MEDIATOR NETWORK PROTOCOLS

Figure 5 shows the way in which Mediator operates over a network. A server agent at a site manages a knowledge base consisting of a heterogeneous set of files from different applications. Concept maps are used to represent the files and relations between them. Files may be opened from the maps in the appropriate applications. Since the maps and hypermedia documents of Mediator are also files, the system can be used to support large-scale linked knowledge structures. Client agents at remote sites connect to server agents across the network and allow files to be accessed remotely in the same way as they are locally. A write token for each file is passed around the network allowing collaborative development of knowledge structures. The data structures for the visual languages are very compact allowing real-time updates with network data rates as low as 1 Kbyte/sec.

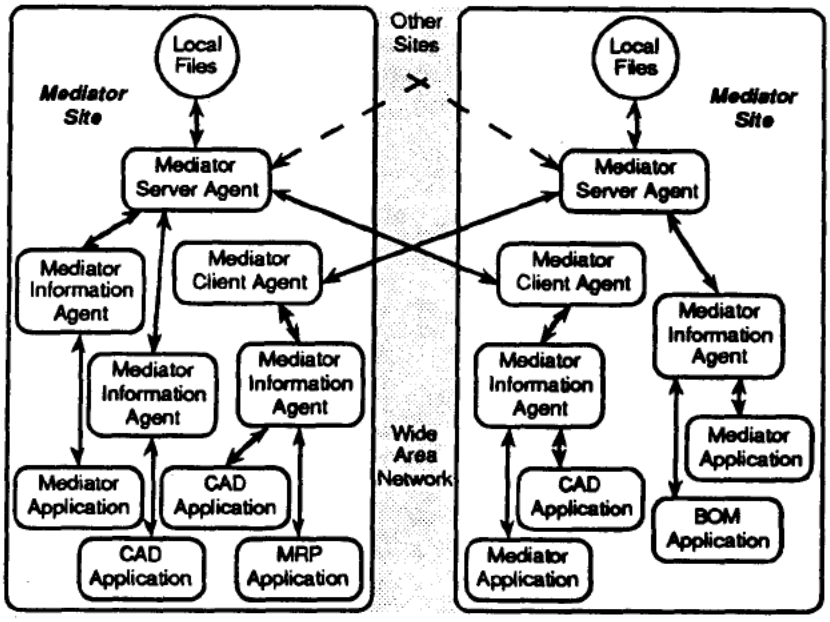

Figure 5 Mediator operation over a network

Figure 6 left shows a screen dump of a Mediator server agent managing files for three projects at one location. Figure 6 right shows a Mediator client agent at a remote location having linked to the server agent. The cursor in the right screen dump is over a node in the concept map and has changed to a button shape to indicate that the associated dataset will be opened if the mouse button is clicked.

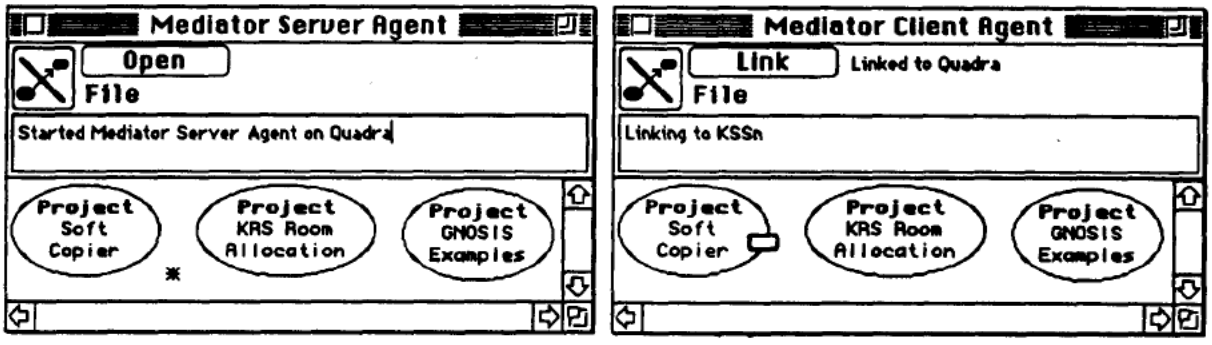

Figure 6 Client agent linked to server agent across a network

Figure 7 shows the dataset opened when the button is clicked. It is another concept map, but this time embedded in a document, giving the project structure from requirements through design to operational models, including a video demonstrating the prototype system in operation. The embedded concept map remains active. If the user selects this video through the "Open File" option in the popup menu as shown, or through the button cursor as shown above, then the video will open and play in a QuickTime viewer. The popup menu also allows the user to check on the application program appropriate to the file since it is being fetched from a remote site, and the local site may not have the appropriate application available. The "fetch" operation may be virtual since the system caches files transparently to avoid unnecessary overheads in transporting large files over slow networks. A lookahead facility between collaborating sites allows the caches to be set up in advance if it is likely that files will be accessed remotely.

All of the Mediator knowledge structures may also be embedded in active, printable documents (Gaines and Shaw, 1993), so that reports, manuals, etc, are readily generated.

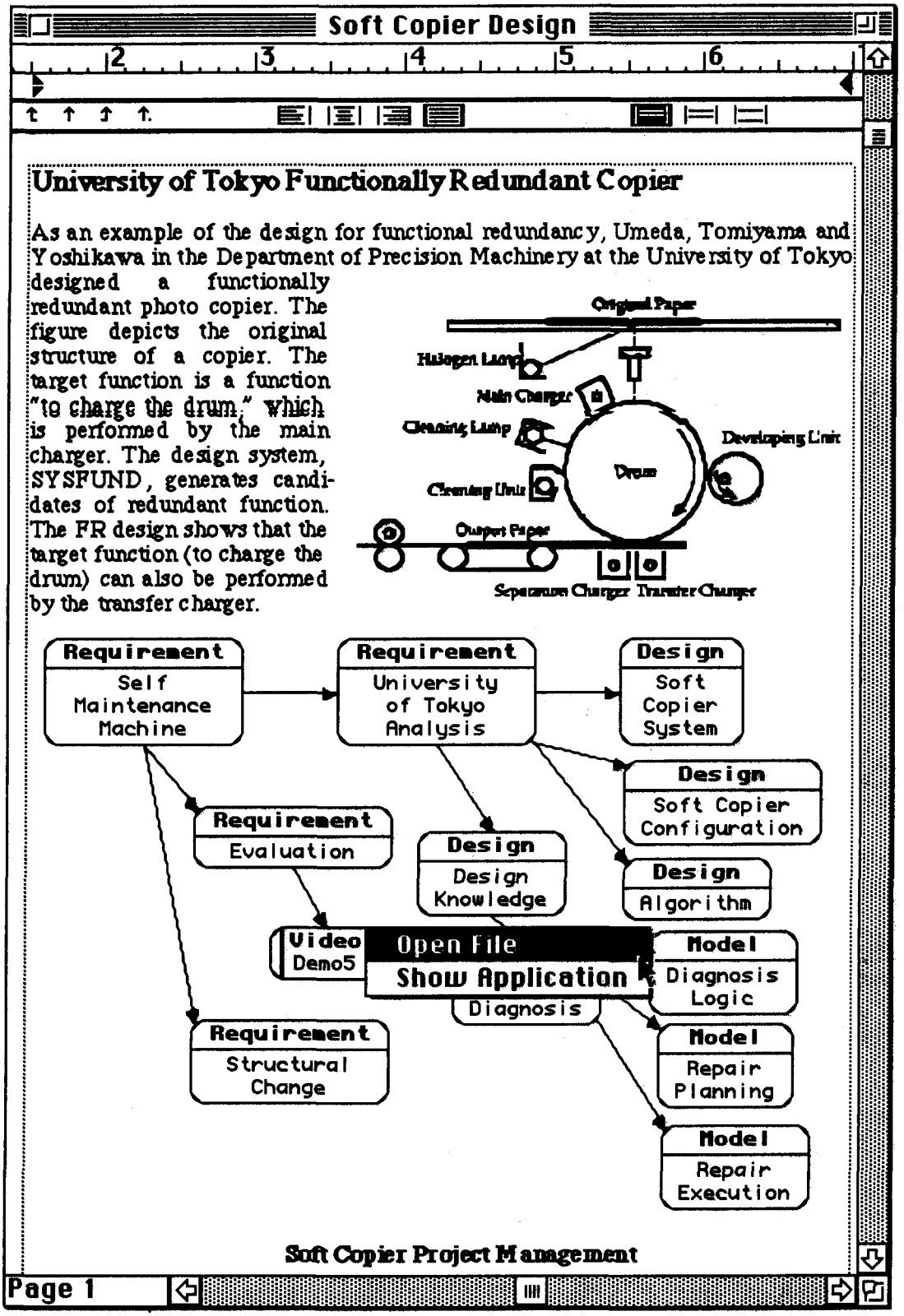

Figure 7 Project management concept map embedded in a document

CONCLUSIONS

An open architecture information and knowledge management system has been described which provides a flexible technology to support the management of complex manufacturing environments. The prototype implementation is Macintosh-based and operates over the Internet through TCP/IP protocols. An X11 implementation for Unix platforms is underway.

ACKNOWLEDGMENTS

This work was funded by Industry, Science and Technology, Canada, as part of the international IMS research program, and by the Natural Sciences and Engineering Research Council of Canada. We are grateful to many colleagues in IMS for their contributions to the Mediator project, in particular, Aldo Dagnino, Peihua Gu, Martti Mäntylä, Tetsuo Tomiyama and Moriki Toyama.

REFERENCES

Finin, T., Weber, J., Wiederhold, G., Genesereth, M., Fritzson, R., McKay, D., McGuire, J., Shapiro, S. and Beck, C. (1992). Specification of the KQML Agent-Communication Language. The DARPA Knowledge Sharing Initiative External Interfaces Working Group.

Gaines, B.R. (1991a). Empirical investigations of knowledge representation servers: Design issues and applications experience with KRS. **ACM SIGART Bulletin 2**(3) 45-56.

Gaines, B.R. (1991b). An interactive visual language for term subsumption visual languages. **IJCAI'91: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence**. pp.817-823. San Mateo, California, Morgan Kaufmann.

Gaines, B.R. (1992). Manufacturing in the knowledge economy. **Proceedings of ICOOMS'92: International Conference on Object-Oriented Manufacturing**. pp.19-36. Calgary, University of Calgary.

Gaines, B.R. (1993a). A class library implementation of a principled open architecture knowledge representation server with plug-in data types. **IJCAI'93: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence**. pp.504-509. San Mateo, California, Morgan Kaufmann.

Gaines, B.R. (1993b). Experience with a class library for organizational modeling and problem solving. **Integrated Computer-Aided Engineering 1**(2) 93-107.

Gaines, B.R. and Shaw, M.L.G. (1993). Open architecture multimedia documents. **Proceedings of ACM Multimedia 93**. pp.137-146.

Gaines, B.R. and Shaw, M.L.G. (1994). Using knowledge acquisition and representation tools to support scientific communities. **AAAI'94: Proceedings of the Twelfth National Conference on Artificial Intelligence**. pp.to appear. Menlo Park, California, AAAI Press/MIT Press.

Gui, J.-K. and Mäntylä, M. (1993). Assembly modeling on the basis of a mechanical design prototype. Laboratory for Information Processing Science, Helsinki University of Technology.

Ishii, M., Tomiyama, T. and Yoshikawa, H. (1993). A synthetic reasoning method for conceptual design. Department of Precision Mechanical Engineering, The University of Tokyo.

Kremer, R.C. (1991). Experience in applying KRS to an actual business problem. Boose, J.H. & Gaines, B.R., Ed. **Proceedings of the Sixth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop**. pp.11-1-11-12. Calgary, Canada, University of Calgary.

Kwok, A.D. and Norrie, D.H. (1993a). Integrating multiple reasoning in intelligent agent systems. **Integrated Computer-Aided Engineering 1**(2) 83-90.

Kwok, A.D. and Norrie, D.H. (1993b). Intelligent agent systems for manufacturing applications. **Journal of Intelligent Manufacturing** 285-293.