8-15-1997

Collaborative Molecules: A Component-Based Architecture for GSS

Robert O. Briggs
University of Arizona, bob_briggs@cmi.arizona.edu

Daniel D. Mittleman
University of Arizona, dmittleman@cmi.arizona.edu

E. Santanen
University of Arizona, santanen@bpaosf.bpa.arizona.edu

Collaborative Molecules: A Component-Based Architecture for GSS

Robert O. Briggs (bob_briggs@cmi.arizona.edu),
Daniel D. Mittleman(dmittleman@cmi.arizona.edu),
E. Santanen (santanen@bpaosf.bpa.arizona.edu), and
Duffy Gillman (duffy_gillman@cmi.arizona.edu)
Center for Management of Information
University of Arizona, Tucson, Arizona 85721

Introduction

Over the past decade a great deal of field and laboratory research has shown that groups using Group Support Systems (GSS) technology can work faster, create higher quality products, and generate higher commitment to results than do teams using conventional methods, (Connolly, Jessup, & Valacich, 1990; Gallupe, et al., 1992; Nunamaker et Al, 1996-97). However, the architectures of existing GSS tools may tend to impose unnecessary limits on the teams that use them because a) components of the tools are set in pre-determined relationships, b) they are embedded in closed environments, and c) they typically run on a single platform. This paper proposes an alternative architecture for GSS tools which may provide substantially increased flexibility and power for teams, while substantially reducing the development cycle for GSS researchers.

Theory

Team Theory of Group Productivity posits (among other things) that when team members collaborate they must divide their limited attention resources among three processes: communication, deliberation, and information access (Briggs, 1994, Briggs, Reinig, Shepherd, Yen, & Nunamaker, 1997). Collaborative technology may be used to reduce the cognitive load of any or all of these processes. It may reduce the cognitive load of communication, for example, by offering simultaneous or parallel channels. It may reduce the cost of deliberation by structuring and focusing group thinking on just the piece of the process that is at hand. It may reduce the cost of information access by providing permanent records of information that would otherwise be ephemeral, or by allowing many experts to opine simultaneously in a structured mind-dump.

Limits of Existing Architectures

The tools in the GSS toolkit are composed of some combination of collaborative components-like lists and texts-in fixed juxtaposition with one another. While these tools are demonstrably effective, they are generic, supporting important deliberative processes generally used by groups, but not necessarily supporting the specific processes that might be desirable for a particular group. The group would have to adapt their process to the tools available, rather than vice versa. Programming a new tool for the needs of each team each day would be prohibitively expensive, and far too time-consuming to be effective. Additionally, most GSS tools are embedded in closed, or proprietary environments, require the users to learn multiple software tools, and have poor data integration facilities (Dennis, Pootheri, & Natarajan, 1996). To take advantage of any component of the GSS, the user must load the entire environment before opening one of the existing tool sets to begin work. The collaborative object created in this manner has no existence outside the environment unless its contents are converted to a different format, say a word processing document or a web page; however, once the conversion takes place the object loses its collaborative nature. Keeping the collaborative objects bounded within their proprietary environment eliminates a number of knotty programming problems. However, it also precludes the team from embedding stand-alone collaborative objects in other places, such as web pages, presentations, documents, or maps.

A Solution: Component-Based GSS.

In this paper we offer a new conceptual architecture for GSS technology. With this new architecture we propose to break GSS tools up into fundamental collaborative components, allowing teams to create and juxtapose these components in any order their needs dictate. Just as chemical elements combine to form many varied and useful molecules, so we propose to let teams assemble these fundamental collaborative components on-the-fly into tools that meet the specific demands of their work. We have therefore call our architecture a "Collaborative Molecules" approach to GSS.

We propose to allow these collaborative objects or "molecules" to stand alone as functional collaborative objects outside the environment in which they were created. By taking this approach, we expect to vastly increase the flexibility of support a GSS can provide to a group, and to substantially reduce GSS tool development cycles. The approach should make it easier to cope with a coming generation of as yet unanticipated group needs and group technologies.

The Collaborative Molecules Architecture

In keeping with our molecular metaphor, our architecture offers three levels of collaborative objects: particles, elements, and molecules.

Consider first the collaborative elements. All GSS tools are made up of just a few kinds of fundamental collaborative components such as a tree, text, graphic, picture, and matrix. Each of these fundamental components is analogous to an element from the periodic table, and so we refer to them as collaborative elements. Each kind of collaborative element has attributes and methods that distinguish it from the others.

Just as chemical molecules are assembled from more basic elements, under our architecture GSS tools will be assembled from the collaborative elements. Any kind of element may serve as the starting of a molecule. For example, a team may elect to begin work by building a shared tree element. At some point they may elect to attach shared text editors to each node of the tree. On the other hand, a team may elect instead to start their molecule with a shared text at the root. Thus, collaborative tools would be created on-the-fly to match the form and complexity of the situation at hand.

Just as elements are made up of sub-atomic particles, so each collaborative element would be composed of smaller parts: particles. A tree, for example, would have nodes, a graphic would be composed of shapes, and a matrix would be a compound composed of lists and cells. Each item on a list, each node on a tree, each cell in a matrix would be a child to some element. Further, each particle would function as a link to other elements in the collaborative molecule. Thus *each particle has an element as a parent, and each element has a particle as a parent.*

Under this architecture, collaborative molecules will be hierarchical structures. The starting point for any collaborative molecule will be a generic root particle. This one root particle will link to some set of elements that constitute the top layer of the tree. Each particle of the top layer may link to child elements, which may in turn link to other child elements, and so on until the groups needs are met.

Three Tier Software Architecture

The Collaborative Molecules project will exist within the context of a three-tiered software architecture which will provide the maximum flexibility of arrangement and future growth potential. The tiered components consist of an interface object for user interaction, a server object for group management and coordination, and finally, a persistent database object. Each of these components are to be coded as individual component modules so that one module with increased functionality may be substituted for another without effecting the existing functioning objects.

The interface object will be the GUI that the users interact with during the collaborative sessions. It will be divided into a menu bar, a tool bar, and a graphical viewing area, each providing a way to search for, retrieve, and display specific elements and molecules, interact with the server, and add to the current molecule.

The server object will perform the tasks associated with administrating the collaborative session in two modes: a central server, and an intermediate server. Collaboration with a central server should be fine for small groups of users, while introducing an intermediate server can be useful in situations where a great number of clients who are working in geographically dispersed clusters are participating in the collaboration effort. This will limit the number of replications the central server will be called upon to make in larger sessions, and will provide a base for continued work should some communication lines fail.

Finally, the database will exist in three levels: a server database, an intermediate database, and a local cache. The server database, through its tight coupling with the central server, will act as the source of coherency for the entire collaborative effort by maintaining the master record for each element. Like the server database, the intermediate database will be tightly coupled to the intermediate server to support geographically dispersed groups; each client will interact with their intermediate database, which will in turn interact with the server database. In addition to the intermediate database, each client machine will maintain a local caching system, to speed switching between elements and molecules on the client side.

In the event that some communication lines cannot be maintained between the central and intermediate servers/databases on a continuous basis, the increased need for coupling the intermediate database to the intermediate server is emphasized so that groups of individuals who have been disconnected from the central server can continue to be productive on a local level, and synchronize their work with the others at a later time.

A final consideration is the network architecture. When collaborating over the internet, the data requirements of large groups working with graphic objects will be significantly different from smaller groups working with predominantly text objects. Thus, two communication modes are required: continuous high-speed access, and intermittent low-speed or "keyhole" access. It will thus be important for the servers to track the location of each participant within the current molecule to allow for optimization of low bandwidth transmissions such as background transmission and data compression of elements and molecules proximate to where the user is currently working.

Conclusion

Through our continued efforts, we expect to vastly increase the flexibility of support a GSS can provide to a group, ease the role of the group facilitator, and to substantially reduce GSS tool development cycles. The approach should make it easier to cope with a coming generation of as yet unanticipated group needs and group technologies.

References:

Dennis, Alan, George, Joey F., Jessup, Len M., Nunamaker, Jay F. Jr., and Vogel, Douglas R. "Information Technology to Support Electronic Meetings," *Management Information Systems Quarterly*, December 1988, p. 591-624.

Dennis, Alan, Pootheri, Sridar, and Nataranjan, Vijaya, "TCBWorks: A First Generation Web-Groupware System," *Proceedings of the Thirtieth Annual Hawaii International conference on Systems Science*, 1997, p. 167-177.

Gallupe, R.B., Dennis, A.R., Cooper, W.H., Valacich, J.S, Bastianutti, L.M, & Nunamaker Jr., J.F. Electronic brainstorming and group size. *Academy of Management Journal*, 35(2), 1992, 350-369.

Nunamaker, Jay F., Chen, Minder, Purdin, Titus D.M. "Systems Development in Information Systems Research," *Journal of Management Information Systems*, Winter 1990-1991, vol 7, No. 3, p 89-106.

Nunamaker, Jay F.; Dennis, Alan R.; Valacich, Joseph S., Vogel, Douglas R. "Information Technology for Negotiating Groups: Generating Options for Mutual Gain," *Management Science*, October 1991, vol 37, no 10, p1325-1346.

Nunamaker, J., Briggs, R., Mittleman, D., Vogel, D., & Balthazard, P., "Lessons from a Dozen Years of Group Support Systems Research: A Discussion of Lab and Field Findings," *Journal of Management Information Systems* /Winter 1996-97, 13(3), in press.