Argumentative Agents as Catalysts of Collaboration in Design

Raymond McCall and Erik Johnson
Sundance Laboratory for Computing in Design and Planning
College of Architecture and Planning, University of Colorado, Denver
Boulder, CO 80309-0314
mccall@spot.colorado.edu

ABSTRACT

Since the 1970s we have created hypertext systems supporting Rittel's argumentative approach to design. Our efforts aim at improving design by encouraging argumentative—i.e., reasoned—discourse during projects. Despite the intrinsically group-oriented character of the argumentative approach, all of our past prototypes were single-user systems. The project reported on here is the first in which we aim at supporting argumentation in group projects. To do this, we augmented our PHIDIAS hyperCAD system to shows how *argumentative agents* can initiate and sustain productive collaboration in design. These agents catalyze collaboration among designers working at different times and/or places by 1) detecting overlaps in the concerns of different participants in a design process, including conflict and support relationships, 2) notifying these people of these overlapping concerns, and 3) enabling asynchronous communication among these people to deal collaboratively with the overlaps. We call these agents *argumentative* because they represent different personal and professional viewpoints in design and because they promote argumentative discourse among designers about various issues. In addition to identifying and dealing with crucial problems of coordination and collaboration, argumentative agents enable the capture of important design rationale in the form of communication among project participants about these crucial problems.

BACKGROUND

Since the mid-1970s we have created hypertext systems supporting Rittel's argumentative approach to design (Rittel 1972). The goal of our efforts is to improve design by encouraging argumentative—i.e., reasoned—discourse during design projects. The argumentative approach is oriented toward group design but can also be used by the individual designer.

The first systems we created were hypertext systems that facilitated the creation and inspection of design rationale for individual projects (McCall 1979)(McCall, Mistrik, and Schuler 1981). At the beginning of a design project such a system would have little or no data; it would get filled up with rationale as the project proceeded. Our later systems came pre-loaded with domain-oriented databases of rationale and were coupled with CAD graphics (McCall et al. 1990). In recent years we have extended this work in a system we call PHIDIAS. PHIDIAS is domain-oriented and implements CAD graphics, hypermedia navigation, and knowledge-based computation using a common hypermedia substrate to create a kind of system we call *intelligent hyperCAD* (McCall, Bennett and Johnson 1994).

Despite the intrinsically group-oriented character of the argumentative approach, all of our past prototypes were single-user systems. The project reported on here is the first in which we have sought to support the argumentative approach in group design projects. This project thus represents for us a return to a foundational theme in argumentative design.

One of the motivations for our recent emphasis on collaborative design has been to overcome the difficulties that all researchers on design rationale (DR) have had in capturing the rationale that designers actually use. This problem is well known in the DR community, but very little mention of it is to be found in the literature—apparently because people do not tend to write papers about their failures. While designers generally resist attempts to get them to document their rationale, they are not reluctant to engage in argumentative discourse with fellow project

members. Ironically, they are not even averse to conducting much of this discourse by e-mail, thus in effect documenting their design rationale.

E-mail does, however, have serious drawbacks as a vehicle for communication in collaborative design—and thus as a means for documenting design rationale. For one thing, the communication it currently supports takes place outside of the work context and without any way of explicitly referring to the graphical solution configurations that are so often the subjects of design discussion. Nevertheless, the partial success of e-mail as a vehicle for argumentative communication among designers suggests that by solving the problem of supporting collaborative design, it might also be possible to solve the previously-unsolved problem of capturing design rationale.

THE PROBLEM OF SUPPORTING COLLABORATIVE DESIGN

The design of many if not most architectural projects of any size involves collaboration of a team of people including designers, engineers, and others. We shall refer to such a group as a *design team*, even though some of its members do not have the title designer. While much of the heroic mythology of design centers around the solitary designer, much of the most important work of architectural practice is done by skillful collaboration by such teams.

Much of the work of teams is done by individuals working separately. Coordination and collaboration amongst these individuals is accomplished in meetings of various kinds—i.e., by face-to-face interaction. In meetings, individuals—whether they are designers, engineers, or others—often discover crucial but unforeseen relationships between their own work and the work of other team members. Sometimes these relationships are conflicts; sometimes they reinforcements; sometimes they are unexploited opportunities for synergies. When meetings are infrequent, such discovered relationships are often problematic. The later conflicts and potential synergies are discovered, the more costly and impractical it is to improve the quality of the project by doing something about them. It would be ideal if team members could be alerted to the existence of potential conflicts and synergies as soon as they arise.

The situation is worse with respect to collaboration amongst people who are not present at meetings. For example, in a large office there might be a designer in a different project whose knowledge is crucial for some aspect of your project. If you do not know already know this, how can you find out? There is currently no practical way unless someone tells you.

The basic problem here is that team members too often do not know who they should be collaborating with at any given time. They might usefully collaborate with other designers whose work is affected by their decisions—or vice versa. They might find it useful to collaborate with designers who have worked on similar problems. And of course, there are various experts who may have specialized knowledge and analytical skills that could help them in their work. The central problem is that designers typically do not know when, with whom, and on what they should collaborate.

We are at the beginning of a revolution in network computing centering around the rapidly growing and evolving use of the World Wide Web. This revolution will create tremendous new opportunities for collaboration. In principle, it will soon be possible for design to be done by huge and ad hoc groups whose members are distributed throughout the world. In fact, however, there is a fundamental obstacle to making use of these opportunities: designers' lack of knowledge of the needs and opportunities for collaboration.

OBJECTIVE

We set out to create a prototype demonstrating the power of a new software technology that we call *argumentative agents* to initiate and sustain productive collaboration in design. In particular, our intent was to demonstrate how such agents can catalyze useful collaboration among designers who work on a given project at different times and/or places by enabling them to learn about needs and opportunities for useful collaboration.

Argumentative agents are new types of software agents that

1. detect overlaps in the concerns of different participants in a design process—i.e., designers and other experts—including conflict and support relationships,
2. notify these participants of these overlapping concerns, and

3. enable and support sustained communication among these people to deal collaboratively with the overlaps. For examples, they might facilitate argumentative discussion aimed at settling differences of opinion over some issue of building form.

We call these agents *argumentative* because they represent different personal and professional viewpoints in design and because they promote argumentative—i.e., reasoned—discourse among participants in a design process. Participants would create such agents using PHIDIAS and place them in an "team memory" consisting of collections of domain-based information managed by our system. (PHIDIAS already has the ability to manage a class hierarchy of such domain-based information.) The particular domain in which a given agent is placed—and the associated level of generality—determine what projects that agent can be applied to.

We devised three types of argumentative agents that can work either separately or in concert. We call these *advocates*, *reporters* and *scouts*.

Advocates

These agents act as "spokespersons" for their authors' points of view. They are knowledge-based critics that can "look over the shoulders" of designers as they work and critique their partially formed graphical solutions. Advocates can spot configurations that violate principles of design or conflict with specific design decisions that their authors believe should be maintained. These agents then provide access to the textual rationale underlying the critique (McCall, Fischer and Morch 1990).

Reporters

These agents have "assignments" to watch for and report on activity *in given sections* of the system's information store. Typically, they watch for real or potential conflicts with statements or decisions made by a given person—e.g., their author. For example, they might watch for attempts by others to change graphic decisions made by the given person. They might also watch for comments made by others on a statement made by that person. If they detect such events, they would report them back to the given person—for example, reporting the text of an argument made against a decision or the alteration of a graphical configuration originally created by the given person. They would also report these events to any others who had indicated an interest in hearing such news. These additional interested parties we call "subscribers." These people indicate their interest by a formal process of "subscribing" to the "news"—i.e., events—that reporters detect.

Reporters have a crucial additional function: providing a means for multi-party communication among project participants. They do this by communicating comments in more than one direction. Thus, if a given person replies to a comment made by another person, that reply can be carried back to the author of that comment—and also to any others who have "subscribed" to such comments. In this way, reporters constitute the central vehicle for asynchronous communication within a design team. This communication resembles e-mail, but provides crucial additional functionality. For one thing, the comments are presented *in the context of and relevant to actual design tasks*. In addition, the reply (thread) structure is automatically recorded in the argumentative hyperdocument and can be retrieved whenever that task is undertaken or examined—thus automatically capturing design rationale and indexing it by the relevant task.

Reporters are especially powerful in combination with advocates, because advocates can critique a designer in such a way as to provoke a reaction from that designer. Such a reaction—e.g., rejecting a criticism, arguing against it, asking questions about it—can then be reported back to the author of the critic and any "subscribers." Further responses by anyone in this group to this report will in turn be sent to all other group members for further comment—and so forth. In this way, a group collaboration on specific design issues is initiated, sustained and documented in the hyperdocument as rationale for present and future design needs.

Scouts

These agents watch for and report on the creation of new information *anywhere* in the system as long as the information fits a description created by a participant in a design process—e.g., a particular designer. The descriptions might deal with the content of the new information, e.g., any new texts containing the string "airlock" or having the keyword "pressure suit"—the latter including texts described by synonyms of "pressure suit" or by more specific (narrower) terms that inherit from "pressure suit." They might in addition detect structural patterns, e.g., reporting on those new *answers to lunar habitat issues* that are described by the keyword "wardroom." Like advocates, scouts have the ability to use reporters to sustain multi-party discussions among groups of "subscribers"; they can thus initiate, sustain, and document group collaboration.

IMPLEMENTATION APPROACH

Our basic approach to implementing argumentative agents was to enhance two existing knowledge-based mechanisms of PHIDIAS and enable them to interact. We then exploited the synergy resulting from this interaction to create argumentative agents. This new functionality was thus a simple extension of PHIDIAS that realizes the inherent potential of previously existing functionality. Because of its deep integration of knowledge-based computation, vector-graphics and hypermedia, PHIDIAS provided an excellent foundation for implementing argumentative agents. One of the two mechanisms was LINQ (Language for Inference, Navigation and Queries), our hypermedia language. The other was virtual copying, our mechanism for inheritance of hypermedia networks.

LINQ is a functional language bearing some resemblance to FP and FL (Backus 1978). It uses an operator algebra with function-level definitions; and it implements inference through recursive evaluation of logical expressions. LINQ operates exclusively on data stored in a persistent hypermedia network.

Needed Augmentation of Virtual Copying

Although the previous version of our mechanism for virtual copying enabled inheritance of whole or partial hypermedia networks, it failed to provide a basic kind of functionality required by argumentative agents: support for creation of user-extensible class hierarchies of nodes—i.e., hierarchies of node types. This functionality is essential for the creation of the generalized critics and on-added predicates used by "advocates." It also enables agents to serve as "reporters," observing and reporting on activities of various users of the system, each of whom will have a user-specific inheritance context.

Needed Augmentation of LINQ

Once the virtual copying had been extended to support user-extensible class hierarchies of nodes, corresponding changes had to be made to LINQ so that it could exploit these capabilities. First, we made it possible to use "is-a" in LINQ expressions. Second, we made it possible to use the names of node classes in LINQ. Third, we made it possible for the system to take inheritance into account when evaluating LINQ expressions—enabling it to "know," for example, that a *chair* is a type of *furniture* and that a *computer* is a type of *equipment*. Fourth, we enabled agents to be inherited, so that, for example, agents defined for *equipment* would apply automatically to *computers* unless overridden at the level of computers. Thus, for example, "advocates" for *furniture* placement would also apply to *chair* placement. Finally, we enabled LINQ to use knowledge from one inheritance context, yet display the results of expression evaluation in a different inheritance context. Thus, an event in one user context could be detected by LINQ and a message about the event displayed in a different user context. This last piece of functionality provided the central mechanism for communication among multiple users through use of "reporters."

To implement the type of agents we call "scouts," some further extensions were needed to LINQ that did not directly relate to its integration with virtual copying. For one thing, LINQ needed capabilities for content-based retrieval using freetext search and keyword indexing. Early predecessors of LINQ had such capabilities (McCall 1989). As a consequence, it was straightforward for us to implement this functionality in PHIDIAS. The retrieval

capabilities we needed for the agents were quite simple; so no complicated algorithms—e.g., for query optimization—were needed to obtain reasonable performance.

Making PHIDIAS a Multi-user System

A variety of modifications were needed to make PHIDIAS a multi-user system. These included controlling both authoring privileges and system access. Our approach was to assign or deny authoring privileges to each given user in each inheritance context—with authoring privileges themselves being inheritable. This approach was in turn based on the virtual copying mechanism.

Additional modifications were needed to give each user a separate, personal inheritance context for his or her use of the system. We used our existing inheritance mechanism for this. We first of all created a node in the system for each user. From each such node we used our pre-existing mechanism for switching contexts when a link is traversed to link the users to the various types of information that they were dealing with. Finally, we arranged it so that when each user logs on, a "home screen" for that user is displayed. This screen lists various information relevant for that user's work, including any new "mail" messages from argumentative agents.

THE IMPLEMENTED FUNCTIONALITY

Advocates and Reporters in PHIDIAS

Figure 1 shows how an advocate appears to a designer whose work is being critiqued by someone else. (The project information shown in the figures concerns the design of a lunar habitat.) The designer, Patrick, had just placed a computer work center into the wardroom area. The advocate was created by another designer, named Erik, to "lobby for" his belief that work centers should not be placed in wardrooms. Erik's advocate critiqued Patrick when he violated that principle. Note that the dialog box containing the advocate's critique also provides both 1) a button that enables Patrick to view the rationale that Erik gave for this principle and 2) a button that enables Patrick to "argue" this matter with Erik in asynchronous, textual form.

Figure 2 shows Erik's "home screen" when he later logs on. The system notifies him that someone (Patrick) has chosen to argue against his advocate's advice about not placing workstations in the wardroom. In particular, Erik has received the message, "Somebody has argued against your advocate." The author of this message is "Erik's advocate"; in other words, this is a message that Erik is sent by his advocate whenever a critiqued person argues back. Below this in is the English-like LINQ expression of the advocate's rule (principle) which is subject of disagreement. This rule specifies that if the wardroom that the designer is working on physically contains a work center, then the system is to display the message, "The wardroom should not contain a work center." Below this LINQ expression is the argument that the designer (Patrick) has given against the advocate's critique: "Due to tight quarters, we need to be open to the use of multi-purpose spaces, which includes putting workstations in the wardroom." Below this is the name of the author, Patrick.

Figure 2 also shows that Erik has responded to Patrick with an argument further supporting his belief about work centers not belonging in the wardroom by saying, "In order to preserve the 'off-duty' character of the wardroom, it should not contain a workspace." The system automatically sends this comment back to Patrick using a reporter agent associated with the advocate. If Patrick responds to Erik, that response will in turn be sent to Erik using reporter functionality. This shows how advocates can trigger—i.e., catalyze—a collaborative dialog which is then supported by reporters.

Scouts and Reporters in PHIDIAS

Figure 3 shows Erik's "home screen" with messages from a scout and a reporter that he created. Erik had posted the scout to be on the lookout for the creation of any text containing the term "claustrophobia." He had posted the reporter to watch for a change by any other designer to Erik's decision about chair placement in the lunar habi-

tat. Accompanying the reporter's message is a vector graphic representation of the altered configuration of chairs—with the chair in question highlighted.

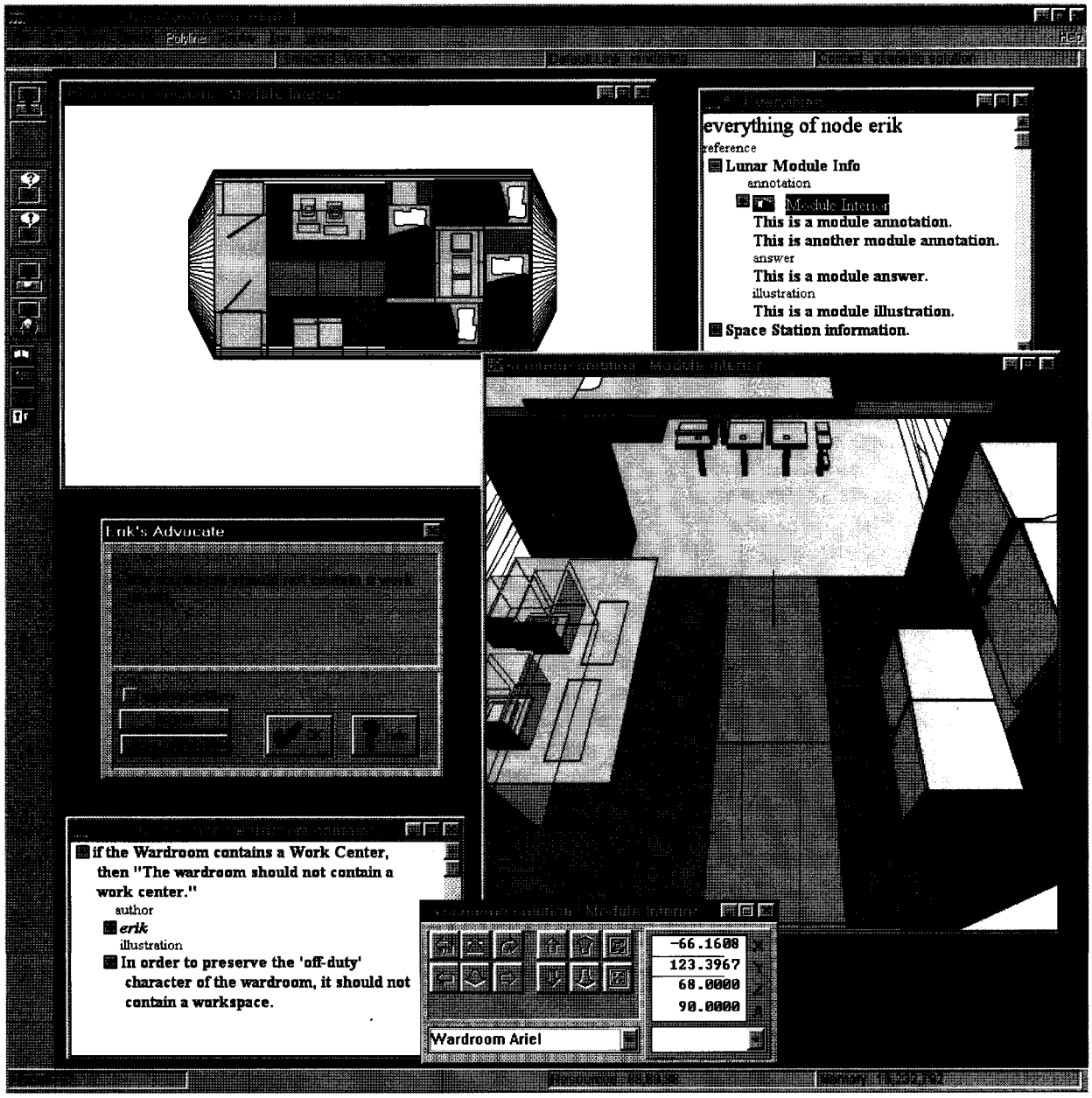

Figure 1. An advocate created by one participant critiques another participant's work.

Figure 4 shows how Erik can expand the scout's message by clicking the mouse on the "plus button" shown to the left of the message. As a result of this mouse click, the system displays the text that the scout found that contains the word, "claustrophobia." Erik has chosen to argue with this statement and his argument is shown below it. The author of the original statement has in turn responded with an argument against Erik's argument. And Erik has replied with a further argument in support of his original argument.

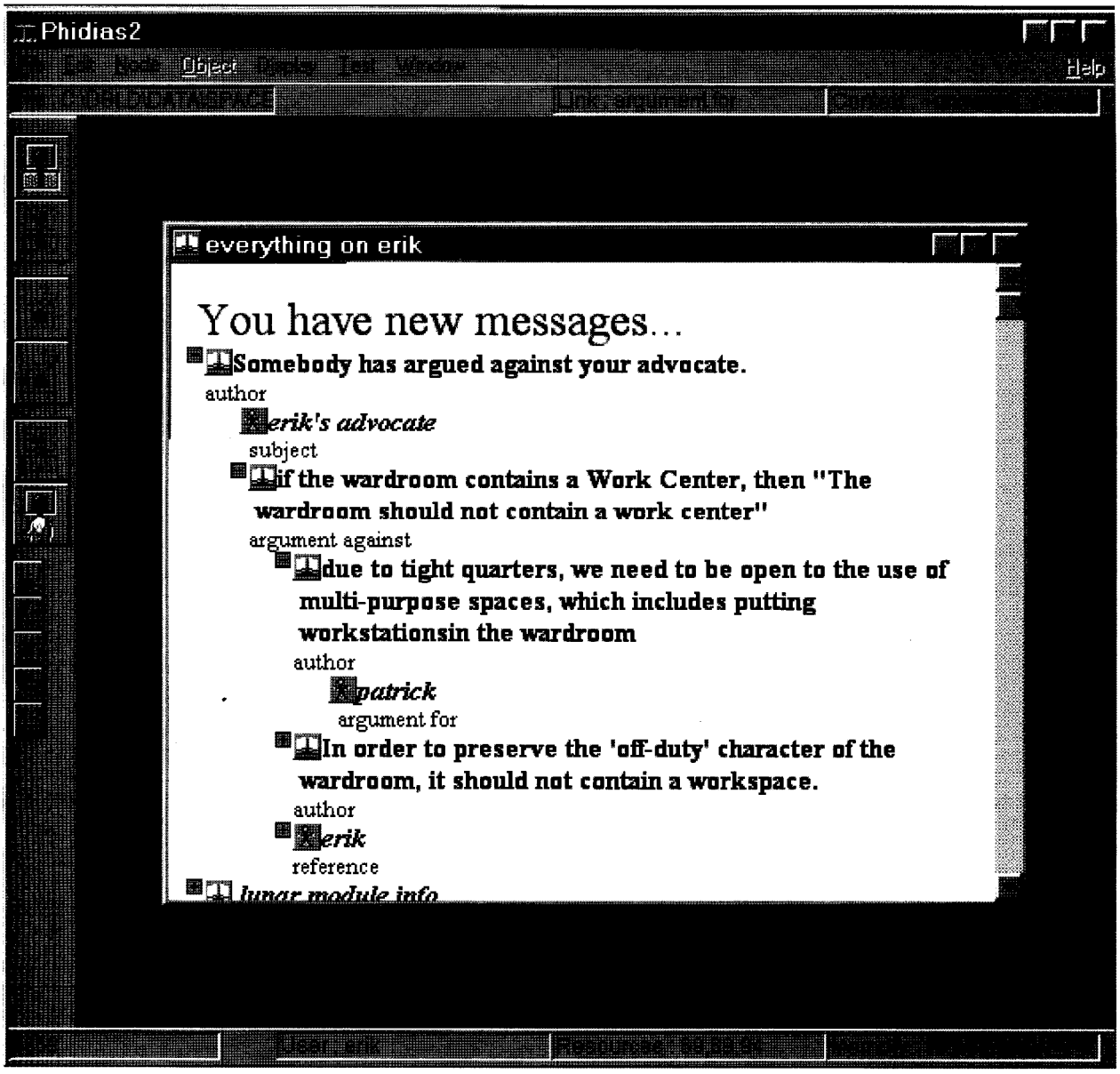

Figure 2. An advocate and reporters catalyze and sustain an argumentative discussion between two project participants.

FUTURE WORK AND CONCLUSION

The creation and use of argumentative agents raises a number of issues that we will need to address in future projects. One is the need for substantial support in helping users to create and post agents in the system. Although the English-like syntax of the LINQ language makes critic rules easy to understand, it does not by itself provide adequate support for users to create agents. For this, a variety of additional aids will be needed. These might include libraries of examples of agents, templates for creating agents, a syntax debugger, and built-in tutorials.

Another issue is that of managing and otherwise regulating agent use. Agents should not be used to spy on or bully other team members—though both of these might be possible unless care is taken to prevent them from happening. Something might also need to be done to prevent agents from overly disrupting the flow of the design

process. For example, it might be necessary to limit the number and type of agents that can display or send messages in any given situation.

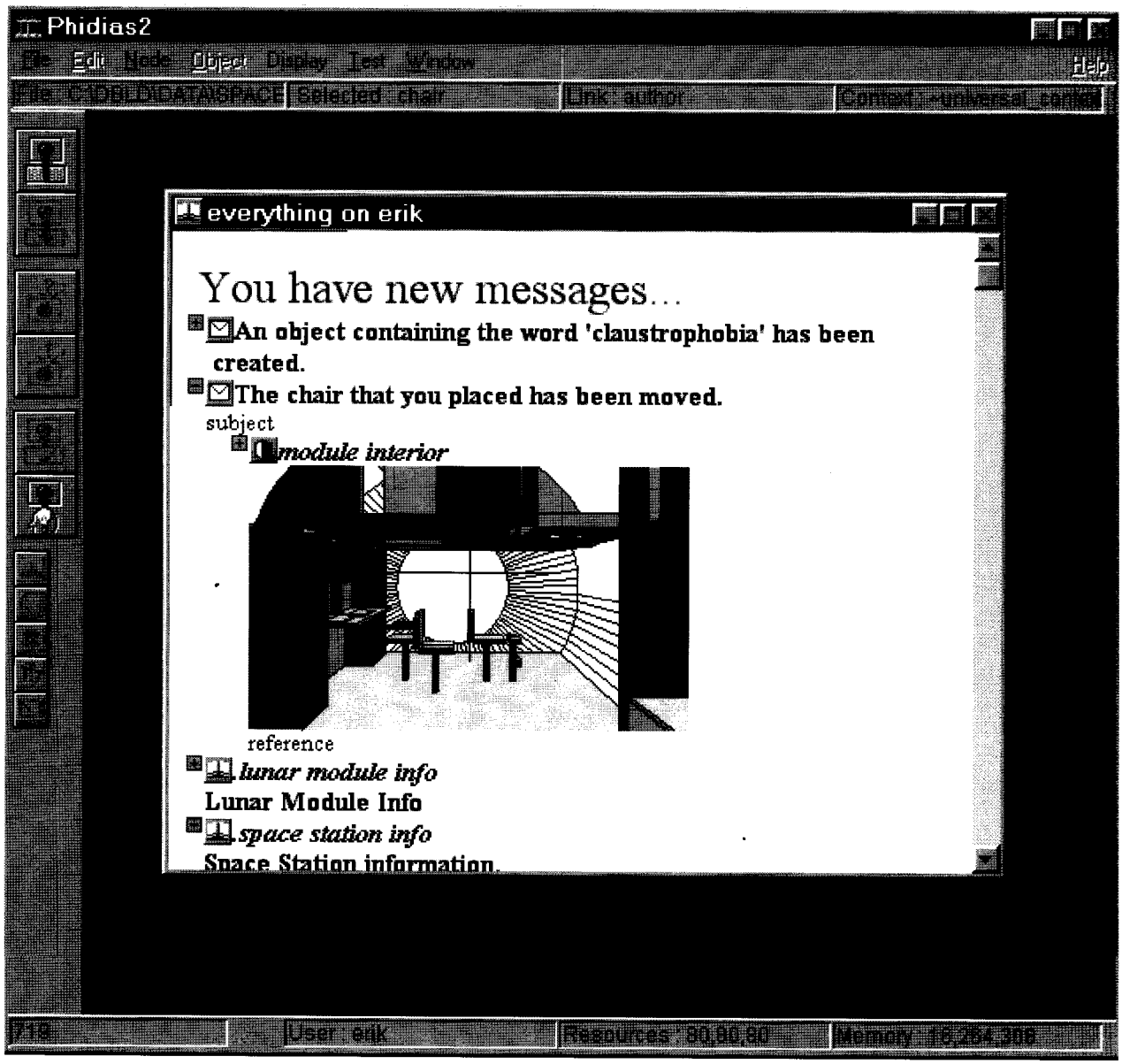

Figure 3. A project participant receives messages from a scout and a reporter.

Despite the need to address these and other issues, we believe that our current prototype demonstrates the feasibility and suggests the potential value of argumentative agents in collaborative design. They can help designers to become aware—and aware early—of needs and opportunities for coordination and collaboration in team-based design. Such agents could therefore play a crucial role in exploiting the potential of the ongoing networked computing revolution to support design by teams, especially by geographically distributed teams.

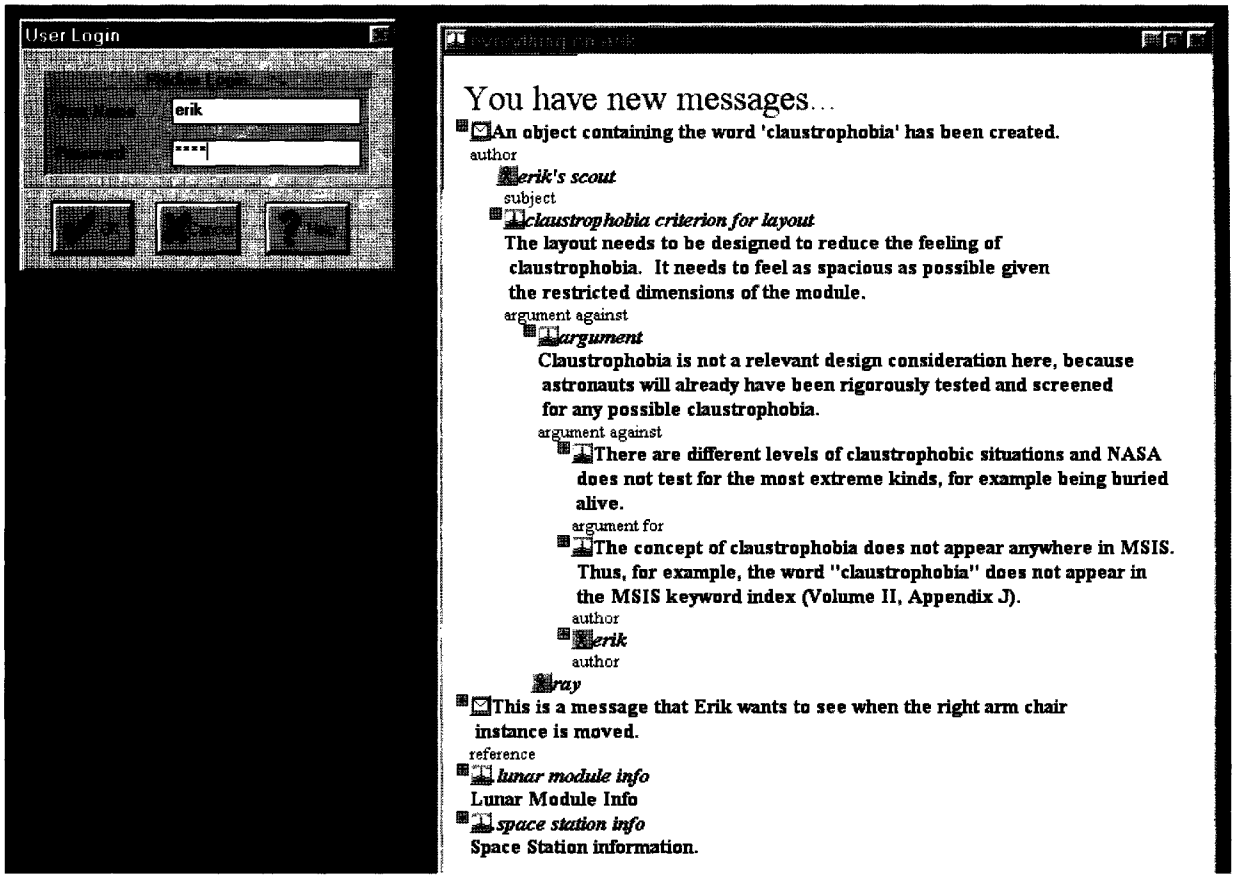

Figure 4. A scout and reporters catalyze and sustain an argumentative discussion between participants.

REFERENCES

Backus, J. Can Programming Be Liberated from the von Neumann Bottleneck: a Functional Style and Its Algebra of Programs, 1977. *Communications of the ACM*, Vol. 2, August, 1978, pp. 613-641.

McCall, R. *On the Structure and Use of Issue Systems in Design*. Doctoral dissertation, University of California, Berkeley (1978), University Microfilms, Ann Arbor, 1979.

McCall, R. MIKROPLIS: a Hypertext System for Design. *Design Studies*, Vol. 10, No. 4, Butterworth Scientific Ltd., Surrey, UK, 1989, pp. 228-238.

McCall, R.; Mistrik, I.; and Schuler, W. An Integrated Information and Communication System for Problem Solving: Basic Concepts. In *Data for Science and Technology: Proceedings of the Seventh International CODATA Conference*, Pergamon, 1981, pp. 107-115.

McCall, R.; Fischer, G.; and Morch, A. Supporting Reflection-in-Action in the Janus Design Environment. In *The Electronic Design Studio*, W. Mitchell, M. McCullough, and P.Purcell (eds.), MIT Press, 1990, pp. 247-259.

McCall, R.; Bennett, P.; Shipman, F.; and Wallace, N. "Integrating CAD with Dynamic Hypermedia," *Proceedings of 1990 European Conference on Hypermedia* (ECHT'90), INRIA, Paris, 1990, pp. 252-261.

McCall, R. ; Bennett, P.; and Johnson, E. An Overview of the PHIDIAS HyperCAD System. In *Reconnecting*, proceedings of the 1994 conference of the Association for Computer-Aided Design in Architecture (ACADIA'94), A. Harfmann and M. Fraser (eds.), Association for Computer-Aided Design in Architecture, 1994, pp. 63-76.

Rittel, H. On the Planning Crisis: Systems Analysis of the First and Second Generations. *Bedriftsokonomen*, No. 8, 1972, pp. 390-396.