

Informing System Design Through Organizational Learning

Gerhard Fischer,¹ Stefanie Lindstaedt,^{1,2} Jonathan Ostwald,^{1,2} Kurt Schneider,¹ and Jay Smith¹

¹Center for LifeLong Learning and Design
University of Colorado at Boulder
Boulder, CO 80309-0430, USA

²NYNEX Science & Technology (S&T)
500 Westchester Avenue
White Plains, NY 10604, USA

{gerhard, stefanie, ostwald, ktschnei, smithjt}@cs.colorado.edu

Abstract: This paper describes a cycle in which organizational learning and work serve each other. Experience in a real-world software development project has led us to identify several key challenges for informing software design through organizational learning. A discussion of these challenges presents steps taken toward addressing them, as well as barriers remaining. The Group Interactive Memory Manager (GIMMe) system is presented along with early results from use of the system in software design projects.

Keywords: organizational learning, group memory, systems design, collaborative work

1 Introduction

Many organizations view learning as a stand-alone activity separate from daily work. Workers are trained away from their workplace, and then are assumed to perform their jobs according to the procedures learned during training. We argue for an integrated view of organizational learning [Senge 1990; Watkins 1993], in which workers learn as they do work [Fischer 1991], and the organization learns from the experiences of workers.

In the context of our work, *organizational learning* focuses on recording knowledge gained through experience (in the short term), and subsequently making that knowledge available to others when it is relevant to their particular task (in the long term) [Fischer 1992]. A central component of organizational learning is a repository for storing knowledge – a group memory [Terveen 1993]. However, the mere presence of a group memory system does not ensure that an organization will learn. For sustained organizational learning, two seemingly disparate goals must be served simultaneously: (1) group memories must serve work by making stored information relevant to the task at hand, and (2) group memories must be extended and updated as they are used to support work practices.

This paper considers organizational learning in the context of software design. "An organizational learning approach to software development uses an organization's accumulated knowledge of the development process and application domains as the basis for design" [Henninger 1995]. This perspective of software design highlights the relationship between the knowledge needs and products of a specific project, and the accumulated products of past projects that are stored in a group memory. When a group or organization works in the same domain for a while, organizational learning from past experiences becomes intriguing: many questions arise repeatedly, previous projects may have uncovered knowledge that is applicable to several projects, and newcomers as well as stakeholders who may have forgotten must be familiarized with the domain and design knowledge.

The next section describes a real-world software design project within a large organization. This project illustrates key challenges for organizational learning in the context of software design. The following section presents an organizational learning cycle as our integrated approach for meeting these challenges. We then describe the Group Interactive Memory Manager (GIMMe) system that has been developed to address specific portions of the organizational learning cycle. Early results from projects that employ GIMMe are described, and further research directions are articulated.

2 Challenges for Informing Software Design Through Organizational Learning

This section briefly recounts an industrial software development project that illustrates key challenges for organizational learning in the context of software design.

While working with an in-house software development group at NYNEX, we became involved in a problematic development project. Prior to our involvement, the user group (telephone service provisioners) had contracted with an outside development contractor to produce a new system. The development strategy followed a classic contract development model in which the user group first produced a request for proposals that described what they wanted in their new system. The development company then sent about 10 people to talk to representatives of the user group. The representatives of the user and development groups negotiated the system requirements and produced a requirements specification. The purpose of the specifications document was to “freeze” the requirements, and to serve as a contract that described the responsibilities of the development group. The contractors then assigned about 100 programmers to the project and began to design and implement a system based on the specifications document. After two years, the project was behind schedule and over budget, and the contractor had yet to show prototypes or any other evidence of progress to the users.

At this point we became involved in the project. Together with the user group, we decided the biggest problem the project suffered was the lack of interaction between the user group and software designers. This lack of interaction had two dimensions. First, the users did not feel that the designers understood the problems of the application domain. The users felt their knowledge was relevant to the project and that the specifications document did not reflect this knowledge. Second, the development process did not provide for ongoing interaction among stakeholders. When the project encountered difficulties, the obvious solution of discussing the problems was not available because the project was set up as though the specification document would be a static contract between designers and users.

To address these problems, we adopted a participatory and evolutionary development approach. The goals were to activate the domain knowledge of the users and, to keep them involved throughout the project. To achieve these goals we opened up the process to users and made communication the driving force for software design by using diagrams and prototypes to ground discussions between stakeholders. During discussions designers took notes, which were then added to a group memory along with the artifacts themselves. The group memory was intended to be inspectable and understandable by all stakeholders. It was built as a hypertext document using the EVA software design environment [Ostwald 1996]. EVA integrates a powerful programming environment with a hypertext environment, allowing prototypes, documentation, comments, and suggestions to be stored in a single hypertext structure. This new approach was successful in engaging the users in the design process (for details see [Ostwald 1996]). The group memory grew to contain more than 400 hypertext records and several prototypes.

However, two new challenges were encountered. First, we had difficulty in capturing activated knowledge that was produced in the course of communicating about the design, even after we had assigned the exclusive role of scribe to one designer. Second, we could see that success in capturing all the knowledge produced by our participatory and evolutionary approach would result in a huge group memory. Making use of this memory later in the project or in a subsequent project would require that the stored information was made relevant to the design task at hand.

In summary, the key challenges identified for informing software design through organizational learning are: (1) activating relevant knowledge, (2) keeping users involved and interested, (3) capturing activated knowledge, and (4) making stored information relevant to the design task at hand. The next section discusses these challenges and how they can be integrated into an organizational learning cycle.

3 Toward an Integrated Approach: An Organizational Learning Cycle

Our approach to informing software design through organizational learning is to realize a continuous cycle in which individual projects serve group memory while group memory serves individual projects. Projects serve the group memory by adding new knowledge that is produced in the course of doing design, such as innovative design products and practices, rationale, and communication. Group memory serves projects by providing relevant knowledge when it is needed, such as solutions to similar problems, design principles, or advice. This

section describes a theoretical organizational learning cycle that addresses the challenges identified in the previous section.

The organizational learning cycle involves three of the four challenges (activating knowledge, capturing activated knowledge, and making stored information relevant to the task at hand), which form a feedback cycle of interdependent activities [see Fig. 1]. Breaking the cycle at any given point will have an adverse impact on all of the activities. The remaining challenge (keeping users involved and interested) is the driving force for the organizational learning cycle. Keeping users involved and interested drives the cycle by continuously activating new knowledge, which can then be captured and used to activate more knowledge. The components of the organizational learning cycle are now described in more detail.

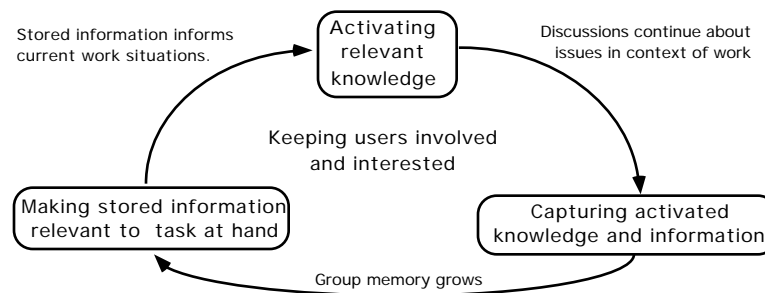


Figure 1: An Organizational Learning Cycle. Challenges (boxes), processes (arrows) and their interrelations form a feedback loop driven by interactions between designers and users.

Keeping Users Involved and Interested. Interaction between users and designers is the driving force of the organizational learning cycle, making it an evolutionary and participatory approach. Contract-based approaches are not conducive to organizational learning because they fail to exploit the learning opportunities that software design presents. These approaches can push projects to write elaborate specifications of poorly-understood user interfaces, before the problems to be addressed by the systems are well understood [Boehm 1988]. Once the specification is frozen, it is assumed that no further interaction between designers and users need take place. By failing to keep users involved and interested throughout the development process, these approaches may cut off communication before design issues are understood. Besides leading to design errors, cutting off communication is also detrimental to organizational learning because knowledge potentially useful to the organization is never activated.

Software development is an opportunity for organizational learning if those with the relevant knowledge are kept continuously involved. There is growing evidence that system requirements are not so much analytically specified as they are collaboratively evolved through an iterative process of consultation between users and designers [CSTB 1990]. This interactive process is necessary because users can initially know their requirements only vaguely at best. New requirements emerge during development because they cannot be identified until portions of the system have been designed or implemented. Users need to experience technological possibilities before they can understand their requirements in a new system.

Activating Relevant Knowledge. A prevalent problem facing software design projects is the so-called "thin spread of application knowledge" [Curtis 1988], in which software designers have too little knowledge about the application domain to make informed decisions. A major factor contributing to this problem is that much application domain knowledge is tacitly [Polanyi 1966] understood by domain workers. This tacit knowledge is essentially taken for granted because of its familiarity, and can be difficult to articulate. Activating knowledge means to make knowledge explicit so it can be shared by others. Interactive artifacts, such as prototypes, can activate tacit knowledge by creating breakdowns [Fischer 1994; Winograd 1986] in tacit understandings. Breakdowns are caused when prototypes fail to act as expected, making users aware of what they formerly took for granted. This activated knowledge can help both users and designers to better understand the application domain and what the new system should be like.

In the first phase of the project described above, we found that approaches that depend solely on textual documents can fail to activate domain knowledge. In the second phase, we found informal diagrams and prototypes to be effective for engaging users in constructive discussions. We also found that the knowledge

these discussions activated could be difficult to capture. This difficulty leads to the third challenge for organizational learning in software design: capturing activated knowledge.

Capturing Activated Knowledge. If software development is to contribute to organizational learning, activated knowledge must be captured and stored in a group memory. Feeding information into a group memory for later use by others is a long-term investment in the future of the organization. However, the perceived short-term value of this effort can be marginal for the current work task. Grudin [1988], in his cost/benefit analysis, pointed out that approaches for organizational learning are doomed to fail when they are seen by workers as requiring extra work without providing any perceived benefit from this extra work.

Our experiences suggest that an integrated approach, intertwining design work with the collection of information, can decrease the perceived cost of contributing to a group memory. In the second phase of the project this cost was negligible, since the design products were created for the immediate needs of the project and then simply collected in the memory. The contents of the group memory in the second phase of the development project indeed grew quite rapidly. However, the contents of group memory were all contributed by designers. Thus, while we succeeded in lowering the perceived cost of contributing to group memory, a bottleneck formed at the point where information representing the user's comments was entered into the group memory. A remaining challenge to capturing activated knowledge is to move from seeing the group memory as a holder of organizational knowledge to seeing the group memory as a medium of communication through which users and designers "speak for themselves."

Making Stored Information Relevant to the Design Task at Hand. To inform software design, group memories must provide their users with the information they need when it is needed. Group memories are not useful simply because they store a great deal of information. They are useful when they make stored information relevant to the task at hand. The information needs of development projects cannot be completely anticipated in advance. Instead, information needs occur in the context of trying to accomplish a particular task.

We found that integrating documentation and prototypes, as supported in the EVA system, enabled rich structures for linking related information in group memory. For example, we often drew sketches and mockups of interfaces and discussed them before building a prototype. The communication from these discussions (in the form of textual notes) was entered into the system and linked to the sketches by designers. Then, when the prototype was built, it, too, was linked with the other design products, forming a structure of related information. During discussions, these structures made the stored information relevant to the task at hand by creating a "neighborhood" of related information that could be easily accessed.

A shortcoming of such static structures is that when they become very large, browsing becomes infeasible and relevant information becomes difficult to locate. Advanced retrieval and delivery mechanisms can complement browsing techniques to help locate pieces of knowledge relevant to a specific current design task or to an ongoing design discussion. These mechanisms can complement static structures through semantic text analysis [Dumais 1988], by exploiting domain-specific terminology [Furnas 1987], and by exploiting the context of the current design task [Fischer 1993].

The following section describes a new tool under development that addresses the challenges exposed in the second phase of the project. The challenges are (1) to capture activated knowledge in the form of communication from users, and (2) to provide additional mechanisms to make stored information relevant to the task at hand.

4 A Supportive Group Memory System as Catalyst: GIMMe

As discussed in section 2, the EVA approach succeeded in addressing two of the four challenges in the feedback loop: keeping users involved and interested and activating relevant knowledge. However it showed difficulties in meeting the other two challenges: capturing activated knowledge and making the stored information relevant to the task at hand. In the following we introduce a group memory system designed to address the last two challenges.

GIMMe is a web-based group memory system. It helps project teams to capture, store, organize, share, and retrieve electronic mail conversations. Mail sent to a specific group alias is automatically added to an information space and categorized according to its subject line. Group members can access the information space via the Internet. We implemented four retrieval mechanisms to this information space: (1) browsing in reverse

chronological order, (2) browsing according to project-specific categories, (3) retrieval by free form text queries, and (4) information delivery of related mail messages. GIMMe supports users in creating, rearranging, or deleting categories and the mail belonging to them. GIMMe is designed to allow groups to create and negotiate domain conventions over time, as well as to evolve categories that reflect the structure and vocabulary of the application domain. (For a more detailed description of GIMMe, see <http://www.cs.colorado.edu/~stefanie>.)

GIMMe uses the Latent Semantic Indexing (LSI) algorithm [Dumais 1988] to support free text queries over the group memory. A user of LSI searching for a document does not have to use the same (key)words the document uses. Since LSI is based on co-occurrence of words, it can retrieve a related document even when this document does not contain a crucial keyword but uses related words. Thus, GIMMe helps users to overcome the vocabulary problem in information retrieval [Furnas 1987]; users can write a query in their own words and LSI can detect relevant information even if it is phrased differently. Neither authors nor a librarian has to index contributions, as the algorithm works on the full text. This reduces the overhead required to add information to the group memory. The scenario in Figure 2 illustrates the power of LSI. A new group member (Jay) can retrieve a stored conversation between two senior group members (Kurt and Jonathan) by simply posting an informal question to the system.

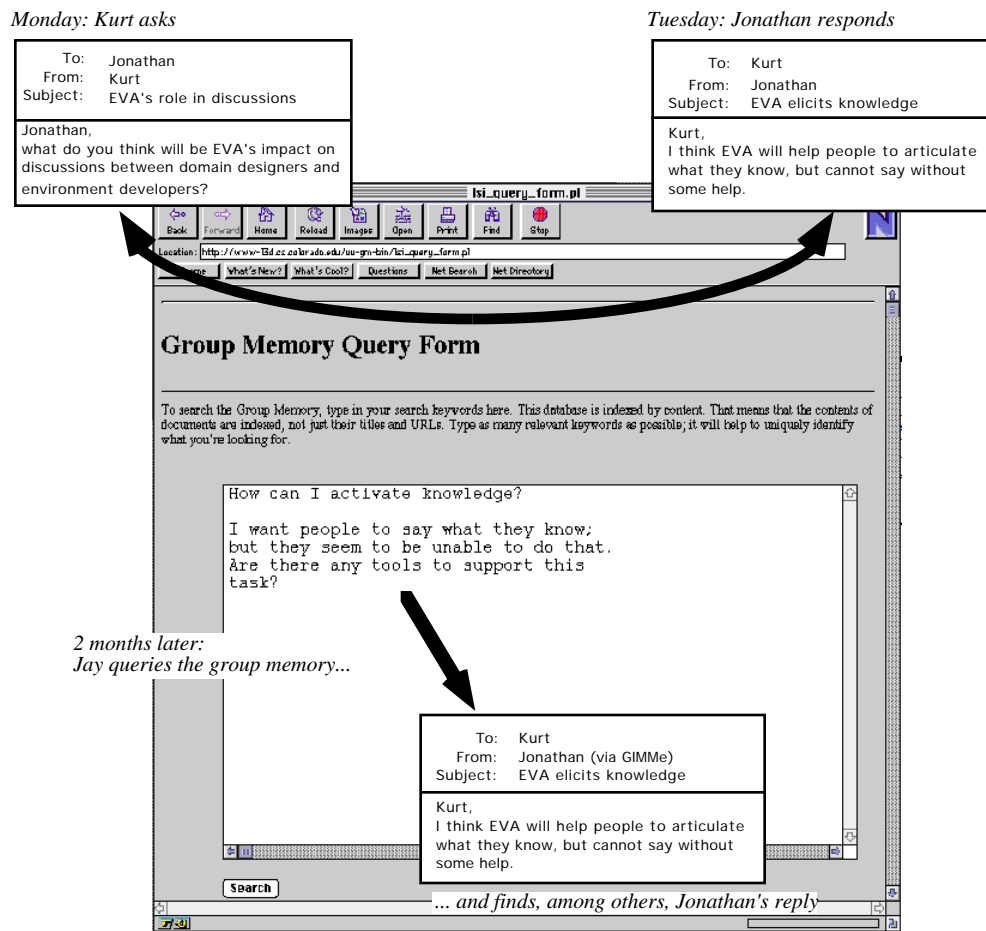


Figure 2: Latent Semantic Indexing in GIMMe. This algorithm helps users to find information in group memory even when they do not know exactly what to ask for.

In addition to the different retrieval mechanisms described, GIMMe also provides an information delivery [Fischer 1993] mechanism that responds to new entries. This mechanism compares new mail to the mail already in the group memory. If a stored mail item is rated as extremely similar to the new one (e.g., LSI rating > 0.9) and if this mail is older than a specified time interval (e.g., 3 months) it is automatically sent to the author of the new mail. This mechanism makes group members aware of what information is in the group memory and

encourages them to discuss it with the authors of the old entries. Group members can control the intrusiveness of this feature by disabling it at any time.

5 Use Experiences

GIMMe is currently employed by several project and research groups in NYNEX S&T and in research projects at CU Boulder (see Table 1 for installation data for three groups at NYNEX S&T). We seeded GIMMe with the mail messages that existed prior to GIMMe's introduction. The different groups at NYNEX have been using GIMMe for periods of time ranging from four to seven months.

Table 1: Information about different groups at NYNEX S&T at the time of the GIMMe installation.

Project	Number of members in project team	Number of e-mail messages already present at time of GIMMe introduction	Was there a group mail alias before GIMMe was introduced?
#1	32	96	yes
#2	50	31	no
#3	8	520	yes

GIMMe automatically collects usage data. We also interviewed members of the groups about how they used GIMMe, what kind of problems they solved using GIMMe, and what they liked or disliked about GIMMe. We analyzed typical quotes from NYNEX employees to identify to what extent GIMMe meets the discussed challenges in the feedback loop in a real-world setting.

Capturing and Structuring Activated Knowledge. GIMMe supports the effortless, automatic collection of communication data and provides low-overhead mechanisms to structure it. Interviews revealed that not only did GIMMe produce no additional overhead; it even reduced the overhead team members had to deal with *before* GIMMe was introduced:

“Once I saw that all the mail is stored in GIMMe, I stopped saving it locally.”

Most team members no longer save mail locally. With GIMMe, team members do not have to decide each time a message arrives if and where to save it.

“There was too much mail. Now I just look into GIMMe once a week to keep up with what is happening.”

In group 3 there are several subgroups working on different parts of the project. Before the introduction of GIMMe everyone always sent all the mail to the group 3 mail alias even when it concerned only a subgroup – just to be sure everyone knew what was going on. With the introduction of GIMMe, people started to send mail more selectively to the subgroups (and the group memory). The group felt this was an advantage, as during normal work people are less distracted by mail not directly related to their work. Everyone is still able to access all mail messages in the group memory.

Making Stored Information Relevant to the Design Task at Hand. GIMMe supports easy and effective retrieval of stored information relevant to the task at hand. Users are able to query in their own words and find relevant information, even if it uses another vocabulary.

“Without LSI I would have had to ask other people in the group to find a mail message since I was not familiar with the terminology used.”

New group members benefit from LSI. A new team member was brought up to speed by GIMMe. He learned about the group, the project, and the terminology without having to bother other team members [see Fig. 2].

By making old communications easily accessible, GIMMe supports consideration of previous arguments, and saves groups from revisiting design decisions.

“Somebody in the group challenged a design decision which was made weeks ago. Using LSI I found the old discussion about this and forwarded it to this person, saying: ‘OK, first read this, then we talk again.’ The result was that we stuck to the old decision and did not waste more time on discussing it.”

The effectiveness of the stored information is increased with the new information-delivery mechanism described in Section 4. At the time of the described incident this feature of GIMMe was not yet implemented, so the person had to take the initiative and search for the old conversation.

Related Approaches. Design rationale approaches attempt to preserve the reasoning processes in a design project. Several approaches to recording design rationale in a group memory have been reported [Moran 1996]. However, there are few reported cases in which design rationale approaches have met expectations. Early design rationale systems such as gIBIS [Conklin 1988] required designers to perform a nontrivial amount of cognitive and manual labor to collect and organize design rationale. Our approach also attempts to capture information from a design project, but we lower the cost of producing this information by trying to capture naturally occurring communication between stakeholders.

Terveen [1993] and Ackerman [1994] describe group memory systems that are able to answer questions about technical issues specific to an organization. They support the key process of making stored information relevant to the task at hand, but they are not geared toward activating application domain knowledge from users.

Planned GIMMe Extensions. Our experience with GIMMe so far is encouraging because GIMMe is being used as a medium for communication within groups. We have learned that such a system can change the way a group communicates. The next step is to enable GIMMe to manage other kinds of group artifacts in addition to electronic mail. From work with EVA we learned that rich group memory structures that link diagrams and notes with prototypes adds context and meaning to the individual pieces. When GIMMe can manage computational objects such as prototypes, it will enable sharing design products as well as sharing communication among group members - an essential step toward integrating organizational learning and work. When GIMMe addresses each challenge in the organizational learning cycle, we can gain further insight into how group memories evolve and how they can sustain evolution as they are used to inform many software design projects [Fischer 1992]. This effort will require continual collaboration with industrial partners.

7 Conclusions

This paper has argued that organizational learning should be an integral part of work. The integration of organizational learning and work was examined in the context of software design. An approach for informing software design through organizational learning was presented, and the following key challenges were identified: (1) activating relevant knowledge, (2) keeping users involved and interested, (3) capturing activated knowledge, and (4) making stored information relevant to the design task at hand. We discussed two systems that each provide support for a subset of these four challenges. The EVA system was found to be successful in activating knowledge and keeping users involved and interested. However, it is unable to capture activated knowledge. The GIMMe system, by being embedded in a group’s work, unobtrusively collects and structures electronic mail conversations. Early results from the use of GIMMe indicate that the system has changed work practices by reducing the overhead associated with using conventional electronic mail systems. Our next step is to extend the GIMMe system to provide support similar to that provided by EVA. By doing so, we hope to close the organizational learning cycle.

8 References

- [Ackerman 1994] M.S. Ackerman, "Augmenting the Organizational Memory: A Field Study of Answer Garden," *The Conference on Computer Supported Collaborative Work (CSCW'94)*, ACM Press, 1994, pp. 243-252.
- [Boehm 1988] B.W. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, Vol. 21, No. 5, 1988, pp. 61-72.
- [Conklin 1988] J. Conklin and M. Begeman, "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," *Transactions of Office Information Systems*, Vol. 6, No. 4, 1988, pp. 303-331.
- [CSTB 1990] Computer Science and Technology Board, "Scaling Up: A Research Agenda for Software Engineering," *Communications of the ACM*, Vol. 33, No. 3, 1990, pp. 281-293.

- [Curtis 1988] B. Curtis, H. Krasner and N. Iscoe, "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM*, Vol. 31, No. 11, 1988, pp. 1268-1287.
- [Dumais 1988] S.T. Dumais, G.W. Furnas, T.K. Landauer, S. Deerwester and R. Harshman, "Using Latent Semantic Analysis to Improve Access to Textual Information," *Human Factors in Computing Systems, CHI'88 Conference Proceedings (Washington, D.C.)*, ACM Press, 1988, pp. 281-285.
- [Fischer 1991] G. Fischer, "Supporting on Demand with Design Environments," *Proceedings of the International Conference on the Learning Sciences 1991 (Evanston, IL)*, Association for the Advancement of Computing in Education, 1991, pp. 165-172.
- [Fischer 1992] G. Fischer, J. Grudin, A.C. Lemke, R. McCall, J. Ostwald, B.N. Reeves and F. Shipman, "Supporting Indirect, Collaborative Design with Integrated Knowledge-Based Design Environments," *HCI*, Vol. 7 (Special Issue on Computer Supported Cooperative Work), No. 3, 1992, pp. 281-314.
- [Fischer 1993] G. Fischer, K. Nakakoji, J. Ostwald, G. Stahl and T. Sumner, "Embedding Critics in Design Environments," *The Knowledge Engineering Review Journal*, Vol. 8, No. 4, 1993, pp. 285-307.
- [Fischer 1994] G. Fischer, "Turning Breakdowns into Opportunities for Creativity," *Knowledge-Based Systems, Special Issue on Creativity and Cognition*, Vol. 7, No. 4, 1994, pp. 221-232.
- [Furnas 1987] G.W. Furnas, T.K. Landauer, L.M. Gomez and S.T. Dumais, "The Vocabulary Problem in Human-System Communication," *Communications of the ACM*, Vol. 30, No. 11, 1987, pp. 964-971.
- [Grudin 1988] J. Grudin, "Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces," *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'88)*, 1988, pp. 85-93.
- [Henninger 1995] S. Henninger, K. Haynes and M. Reith, "A Framework for Developing Experience-Based Usability Guidelines," *Symposium on Designing Interactive Systems (DIS'95)*, ACM Press, 1995, pp. 43-53.
- [Moran 1996] T. Moran and J. Carroll, *Design Rationale: Concepts, Techniques, and Use*, Lawrence Erlbaum and Associates, Mahwah, NJ, 1996.
- [Ostwald 1996] J. Ostwald, "Knowledge Construction in Software Development: The Evolving Artifact Approach," University of Colorado at Boulder, Ph.D. dissertation, 1996.
- [Polanyi 1966] M. Polanyi, *The Tacit Dimension*, Doubleday, Garden City, NY, 1966.
- [Senge 1990] P.M. Senge, *The Fifth Discipline, The Art and Practice of the Learning Organization*, Doubleday, New York, NY, 1990.
- [Terveen 1993] L.G. Terveen, P.G. Selfridge and M.D. Long, "From Folklore to Living Design Memory," *Human Factors in Computing Systems, INTERCHI'93 Conference Proceedings*, 1993, pp. 15-22.
- [Watkins 1993] K.E. Watkins and V.J. Marsick, *Sculpting the Learning Organization -- Lessons in the Art and Science of Systemic Change*, Jossey-Bass, Inc, San Francisco, 1993.
- [Winograd 1986] T. Winograd and F. Flores, *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corporation, Norwood, NJ, 1986.

Acknowledgments

We thank Beatrix Zimmermann, the co-developer of GIMMe, and Bart Burns, the co-developer of EVA. We also thank the members of the Center for LifeLong Learning and Design at the University of Colorado for helpful discussions on these issues. This work is supported by NYNEX, ARPA No. N66001-94-C-6038, NSF IRI-9311839, and a DAAD Research Fellowship.