Toward Introducing Notification Technology into Distributed Project Teams

Jamie L. Smith, Shawn A. Bohner, and D. Scott McCrickard
Center for Human-Computer Interaction and Department of Computer Science
Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106 USA
jls05@vt.edu, sbohner@vt.edu, mccricks@cs.vt.edu

Abstract

Software development can be thought of as the evolution of abstract requirements into a concrete software system. The evolution, achieved through a successive series of elaborations and refinements, is inherently a complex process. The complexity is not merely in the product – the complexity of managing the evolution of the product throughout the life of a project is proportional to the size and complexity of the product. Managing software project teams is a complex task complicated by the distribution of project teams. Distributed collaboration requires adequate support for team coordination and risk management. Human-computer interaction (HCI) combines technical concerns with human psychological concerns. Similarly, software project managers confront complexities in both of these dimensions. In this paper, we report on a collaborative design environment called LINK-UP being developed to guide the design process and facilitate reuse. We examine some web-based project management tools in terms of their support for collaborative design and virtual team processes. We present a notification system approach to making the design record a natural by-product of the design process and a lever for knowledge reuse. Knowledge from past projects could then be used to automatically notify project teams of potential problems and suggest possible solutions.

1. Introduction

Researchers define complexity by citing the ‘‘phase transitions’’ that occur in the physical world between highly ordered and highly disordered systems [9]. For example, water possesses complex physical properties that lie on a spectrum between the highly ordered state of ice crystals and the highly disordered movements of steam molecules.

Corning states that complexity often implies the following attributes [9]:

1. a complex phenomenon consists of many parts,
2. many relationships/interactions exist among the parts; and
3. the parts produce combined effects (synergies) that are often novel, unexpected, even surprising.

This fluidity analogically describes the modern software project – the intangible, malleable product, produced by teams of diversely skilled people distributed across time and space, dealing with many uncertainties. Responding to complexity problems like this often requires effective mental models combined with automation that seeks to relieve the participant from repetitive details and bring into focus the relevant decisions that must be made [5].

Complexity and integration issues frequently dominate modern computing. As the size and complexity of software-intensive systems continues to increase, it becomes difficult for one individual to achieve a full understanding of all aspects of a system design. The knowledge and expertise necessary for successful design is typically distributed among a group of individuals who must share their knowledge, coordinate their efforts, and resolve conflicting perspectives to solve a given problem. Consequently, individuals rely on effective teamwork, sound management, and adequate tool support in the design of complex, interactive systems.

Software development teams are plagued by management problems that result in missed deadlines, budget overruns, and canceled projects, and effective management remains an open problem as development teams struggle to keep pace with changing technology. Today, successful organizations are those capable of adapting to the ever-changing trends of global competition and responding quickly and effectively to evolving customer needs [17].

Globalization has further complicated the issue of project management by popularizing the use of organizationally and geographically distributed teams. Unlike traditional teams that share a physical workspace, distributed teams have the added difficulty of collaborating across the boundaries of space and time.

Expected to compete with traditional teams in terms of quality and efficiency, distributed teams rely heavily on information technology to support many of the communicative and collaborative processes that traditional teams take for granted [17].

1.1. Toward a Solution

Effective collaboration requires team coordination and risk management. Coordination involves establishing team cohesion, maintaining awareness about the activities and perspectives of everyone on the team, and managing dependencies among project-related activities. Risk management involves identifying and prioritizing potential problems and monitoring, mitigating, and controlling those risks throughout the life of the project. To accomplish these goals, the members of a team must maintain a shared understanding of all project-related knowledge, which should include knowledge about the past (what happened in previous projects), the present (what is happening in the current project), and the future (what could go wrong as the project progresses) [12].

All members of a well-coordinated team not only share the same knowledge, but also know that they share the same knowledge [15]. Consequently, these teams spend less time discussing process-related issues of how goals should be accomplished and more time discussing product-related issues of what goals should be accomplished [11]. Existing collaborative systems support activity awareness to varying degrees through the use of notification systems, which display information in the users' periphery without unwanted interruptions to their primary tasks [14]. Notification systems typically support awareness by signaling isolated events, such as the arrival of an email. However, notification systems are also useful in monitoring the evolution of long-term collaborative activities. Notification systems can provide a plethora of awareness data to the members of a distributed team without distracting them from their primary tasks; however, most collaborative systems do not take full advantage of these benefits.

Collaborative systems also overlook the importance of risk management. To effectively manage risk, teams must understand potential problems at the start of the project so that they can attempt not only to avoid common pitfalls but also to identify signs that trouble may be mounting. Teams that do not address project risks often end up over or under-scoping their projects, developing good solutions to the wrong problems, and performing large amounts of rework late in the project timeline [4]. Collaborative systems could draw on knowledge from past projects to warn teams of common mistakes; however, existing tools rarely reuse knowledge or aid teams in explicitly monitoring risk.

As system complexity and team distribution continue to increase, the task of developing tools to support effective collaboration is becoming more important and more difficult to accomplish. This paper examines the strengths and weaknesses of several existing project management tool technologies and proposes a strategy for improving these tools by better supporting team coordination and facilitating risk management through the use of notification systems and the reuse of process knowledge.

2. Web-Based Team Technology

Techniques for managing distributed teams have not been fully explored; however, it is generally accepted that distributed teams cannot be managed using traditional paradigms [3]. Regardless of the management techniques applied, distributed teams require adequate tools to support them in maintaining team coordination and managing risk. Existing project management tools typically fall short in one or both of these categories. A number of web-based research tools have been developed for use by various project teams. Some of the more recent tools target student software engineering teams, demonstrating the relevance of improving project management for partially, as well as fully, distributed teams. Student teams, although not always geographically distributed, rarely share a physical workspace, and thus, often experience coordination problems similar to those of fully distributed teams.

SOPPTS [22], for example, is a task-oriented project management system for student software engineering teams. At the start of a project, teams produce a list project tasks and assign subsets of those tasks to each team member. Team members are then responsible for updating the system as progress is made on each task. Consequently, all team members and the project manager (or the course instructor) can see which tasks have been completed, whether each task was complete on time, and if certain tasks, or team members, have fallen behind schedule.

Public task assignments reduce misunderstanding about who is responsible for completing which tasks and also add an element of peer pressure. Team members are rewarded for completing their assigned tasks on time and pressured by their teammates when progress is slacking. The web-based nature of the system facilitates geographic distribution; however, the system is only beneficial when used regularly by everyone on the team. The amount of overhead it adds to a project in terms of consistently updating progress on individually assigned tasks can distract team members from other project tasks and actually hinder progress. Consequently, use of the system typically diminishes as a project progresses.

JReflex [21] supports project management in student software engineering teams by monitoring team collaboration and evolution of the overall project design. The system integrates project management and development tools without adding overhead by using

CVS records to monitor changes to shared project files in object-oriented development projects. Information about file updates (which files, when, from where, and by whom) is used to visualize the evolution of the project design and to make inferences about team collaboration. As a result, teams can monitor and reflect on their own processes and make changes accordingly. Course instructors can also monitor team processes, notice problems, and provide feedback to the team so that they can fix problems before it is too late.

The data collected by JReflex is archived in an experience repository, allowing instructors to reference past projects to illustrate common pitfalls that students should avoid and to showcase exceptional projects from which students can model their work [21]. However, the potential for reuse is limited. Through reverse engineering, JReflex creates a design model from existing code. Although this model identifies design evolution, it does so only at the OO class level, providing no indication of changes to user requirements or of the rationale behind design decisions.

Similar research tools are also applicable beyond the academic environment. TeamSCOPE [18], for example, targets a broad range of teams. The system improves asynchronous coordination within distributed teams by providing them with a shared file repository, dedicated message boards for each shared file, and a detailed activity history. At login, team members are presented with an overview of awareness data, including recently posted messages and an activity summary. The message boards allow team members to discuss issues related to a particular file and organize those discussions both by topic and by time. The activity summary lists the most recent activities in reverse chronological order and allows team members to filter activities based on the type of event (e.g. posted messages and file or calendar updates) or the context of the event (e.g. activities related to files in a specific folder). As a result, users can monitor their teammates' recent relevant activities (i.e. those activities that relate to their own current tasks) without being inundated with information about all recent project activities.

Although general system features expand TeamSCOPE's application to a broad range of teams, they also limit the tool's usefulness for teams within any specific domain. The system can monitor changes to any shared document; however, no real insight can be gained with respect to how those changes affect the project as a whole. Furthermore, the organization of the activity history into a list of recent events hinders a team's ability to see the project as a whole and allows team members to get lost in the details of current tasks.

TeamSpace [13] provides a shared workspace for document and task management and supports the synchronization and documentation of team meetings. Information presented during a meeting is organized into a timeline and archived for future reference. Key events, such as a team member arriving, leaving, presenting important information, or making a decision, are recorded using descriptive icons. These icons can then be filtered by type or selected to access further details.

Team meetings are only one type of event that can then be included on a full project timeline along with deadlines and other project milestones. Structuring process-related knowledge according to the common dimension of time takes advantage of our ability to organize past experiences into an episodic memory. Based on well-known processes, such as the phases of a development cycle, or temporal markers, such as before or after the team acquired a new manager, team members can reconstruct a sequence of events. Organizing information into a timeline aids team members in maintaining an overall view of the project and retrieving more detailed information as needed.

Most project management tools have limited support for collaboration. They merely model the project activities, sometimes integrate with other tools (e.g., cost estimation tools), and provide effective project reports for managers. Those that do provide collaboration support are largely repositories for project information accessed across the web. The tools presented here go the next step, with collaboration/notification being the centerpiece to improving performance. However, these tools continue to overlook some vital aspects of project management, such as risk management. Few existing tools attempt to explicitly facilitate risk management and, to our knowledge, none attempt to automate risk awareness through the use of notification systems and by leveraging knowledge from previous projects. The concept of process knowledge reuse has been explored with limited success. Our approach to improving process knowledge reuse stems from work in human-computer interaction (HCI) in the area of design knowledge reuse [7, 8, 20], as discussed in the next section.

3. Leveraging Knowledge Reuse

It is generally accepted that system developers can reduce development time and cut costs on a larger scale by incorporating reuse as early as possible in the development cycle [10, 20]. Consequently, new problems do not have to be solved from scratch. Archiving and reusing knowledge about a design product or a design process can help to ensure that effective ideas are remembered and that mistakes are made only once.

3.1. Design Knowledge Reuse

Project planning activities frequently call upon software engineers' design abilities to coordinate many resources for the ultimate goal of a successful project delivery. Design is a creative task that benefits from considerable

knowledge reuse. The concept of design knowledge reuse is of particular interest in HCI, which is concerned with designing interfaces to interactive systems that allow users to accomplish their goals. A key aim of HCI is to inject rationale from psychology, sociology, and other social sciences into the design process so that usability problems can be detected and diagnosed early.

As system complexity increases, so too does the complexity of system interfaces. Successful design increasingly requires a well-constructed, well-trained, and well-managed team that follows a systematic approach to applying scientific knowledge to design practice. This ''engineering approach'' to HCI must complement current software engineering paradigms yet involve the analysis of design rationale to ensure that socio-technical systems are designed with the user in mind [7, 20].

Carroll's method for claims analysis [7] outlines a systematic design approach that contributes to these goals. Claims explicitly state the positive and negative tradeoffs of a particular design feature. Delivered in informal, natural language, claims encourage designers and other system stakeholders to debate design tradeoffs, with the goal of mitigating the downsides of each claim while maintaining or strengthening the upsides. In this way, a set of claims is compiled that exemplifies the rationale behind a system design [7].

Claims represent reusable design rationale grounded in theory from the social sciences. Although claims are tied to a specific context of use, the underlying design rationale can be reused in subsequent projects. To facilitate design knowledge reuse, claims must be abstracted, classified, and stored in a knowledge repository for future application within a new design context [19].

3.2. Process Knowledge Reuse

If teams can leverage knowledge from previous projects to improve their design product, they should also be able to leverage knowledge from previous projects to improve their design process. Basili's Experience Factory [2] was developed to facilitate process improvement in software development by structuring, classifying, and storing packaged experiences from previous projects for reuse. Experiences include both product and process-related knowledge and are input into a repository as artifacts, models, or lessons learned. Experiences are then tailored to meet the needs of a specific project and supplied on demand in the form of models, tools, or baselines.

The concept of reusing process-related knowledge is a natural extension of the reuse paradigm; however, the packaging of reusable experiences is too coarse. Reusing an experience is similar to reusing a generic software process model that has been adapted for a specific project. In this way, attempting to reuse an experience is like attempting to reuse an entire claim set. Although this level of reuse might be possible, it is not a suitable starting point. Experiences must first be broken down into structured chunks of process knowledge appropriate for storage, retrieval, and reuse in a variety of projects across domains.

One potential strategy for process knowledge reuse is to decompose projects into a set of risks. All projects have risks, and many risks occur consistently in projects across domains. Although the probability, priority, and impact of a particular risk will vary with each project, the general risk statement and possible mitigation strategies will be applicable to many different projects.

Reusing process knowledge in the form of project risks is only one possible approach to process knowledge reuse. Additional work is needed to investigate the optimal size and structure of a risk item and to determine the feasibility and usefulness of archiving risk-related knowledge for reuse.

Project management, like HCI, is a complex discipline in need of a more systematic approach, and the effective reuse of process knowledge could be an initial step toward the science of software project management. The success of these proposed methods of reuse for both design knowledge and process knowledge relies on the development of reuse repositories and effective tool support. Tools are needed first to evaluate the practicability of these methods and then to facilitate learning and promote practical acceptance of these methods in academia and industry. The ongoing development of such tools is discussed in the next section.

4. Managing Teams in LINK-UP

In support of the science of design, a suite of web based tools, called LINK-UP [8], is being developed to guide designers through the process of designing a notification system. LINK-UP facilitates the use, validation, and improvement of the claims analysis method by supporting the construction of a claims analysis record during the design process. The system connects to a design knowledge repository, allowing teams to leverage knowledge from previous design efforts by searching for reusable claims relevant to their current project. Throughout the design process, designers also extend this knowledge repository by updating existing claims and creating new ones [8].

Two key goals of the LINK-UP system are to promote practical acceptance of the claims analysis method and to facilitate learning through applied project work in undergraduate and graduate HCI courses. However, to achieve industrial and academic acceptance, LINK-UP must adequately support collaborative design efforts. Computer-aided design tools, like LINK-UP, typically guide the design process and facilitate management of product-related knowledge; however, few tools support users in documenting and reflecting on process-related

knowledge [20]. Given the growing complexity of system design and the increased distribution of project teams, collaborative design tools must aid teams in maintaining effective coordination and managing project-related risks. The addition of a project management tool to LINK-UP has the potential to support both team coordination and risk management while extending the reuse paradigm to include process-related knowledge. The tool should improve the system's overall usability for distributed teams by better supporting collaborative team processes. Additionally, the tool should encourage the evolution of project management techniques for distributed teams as process knowledge is organized, archived, and improved through reuse.

An effective project management tool should benefit distributed project teams by supporting team coordination through the externalization and maintenance of a collective team memory and by facilitating risk management through reuse of knowledge from previous projects.

4.1. Maintaining a Team Memory

With an increase in system complexity comes the need for effective knowledge management to promote efficiency and coordination in project teams. Information technology plays a key role in organizing, storing, and retrieving large amounts of knowledge and in allowing organizations to take advantage of the knowledge reuse paradigm [10]. However, knowledge management is more than simply storing documents in a searchable repository. It involves acquiring, sharing, and integrating knowledge from multiple perspectives into a shared understanding of a given problem and its intended solution [1].

Distributed cognition [16] stresses that the true power of human intelligence is captured only through the interaction of minds. Consequently, team members should not assume shared understanding of knowledge regarding design rationale, experience from previous projects, or task dependencies. To facilitate learning through shared knowledge and synthesis of competing perspectives, distributed knowledge must be externalized and recorded, creating a physical record of the team's mental efforts in the form of a collective team memory [1]. A team memory should contain all knowledge related not only to the design product, such as design rationale, but also to the design process, such as team roles, responsibilities, contributions, and progress. This knowledge can be collected and maintained through the use of adequate communication and awareness mechanisms.

Team members need to maintain a ''big picture'' view of their project while ensuring that all members of the team have access to the same project-related knowledge. They need not only to remember how the design has evolved throughout the life of a project, but also to notice and understand recent changes that teammates have made to the design. With this knowledge, team members should possess a better understanding of project tasks, dependencies, and risks as the design progresses and evolves.

Structuring design rationale in the form of claims presents the opportunity to improve activity awareness by providing greater insight into the nature of a design change. An update to an arbitrary, user-defined document in a system, such as TeamSCOPE [19], provides little insight into the scope of the update or the effect it might have on the project as a whole. While JReflex visualizes detailed changes at the implementation level, it provides no insight into the rationale behind those changes. Visualizing changes within a claim set could help to solve these problems. An update to a single claim or to a subset of claims narrows the context of a design change, providing a better indication of what aspects of the design are changing and to what extent. Additionally, given the structure of a claim, changes to a design can be monitored more meticulously. For example, adding a new upside to a claim might indicate a minor update, while modifying the feature text and several design tradeoffs might indicate a more significant design change.

If a physical team memory is to be beneficial to project teams, it must be easy to maintain and use. The collection, organization, and archiving of project-related knowledge should be a natural by-product of the design process that adds minimal overhead to the project. Additionally, a team memory must be organized and presented to the team in such a way that team members can quickly notice and understand changes and potential problems and easily retrieve further details when necessary. The use of timeline visualization allows team members to organize all project-related knowledge along the common dimension of time. As the project evolves over time, the visualization should reflect those changes, notifying the team of important updates and maintaining a physical record of all project activities.

4.2. Facilitating Risk Management

The knowledge contained within a team memory should not be discarded at the end of a project. Instead, it should be archived as a physical project record for reuse. Subsequent design teams can then examine design processes, discover problems, and consider solutions from previous projects, acquiring advice from multiple perspectives before making a decision that is tailored to their specific project. In this way, the team memory serves as a case study for a given project, with the benefits of automatic generation and integration within a reuse repository.

To reduce project overhead, the process of locating and comparing similar projects could be automated in LINK-UP. When a new project is created, the system could retrieve records from similar past projects, based on

parameters such as the size of the team, the project timeline, and the type and scope of the system being designed. Of the similar projects archived for reuse, those that were completed successfully could be used to create a template to guide plans for the new project. From this template and from additional knowledge stored in the past project records, teams could learn important lessons, such as what internal deadlines they might want to set for themselves or as what point in the timeline their design should begin to stabilize.

If the knowledge contained within project records could be decomposed into reusable project risks, this process knowledge could then be used to automatically warn students of common pitfalls. For example, sizeable deviation from the average number of claims used in successful similar projects might indicate a poorly scoped design. Design changes made late in the project timeline might indicate late scrap and rework, which often results in missed deadlines. LINK-UP could alert teams when the majority of work seems to take place just before a deadline or when particular team members are not contributing to the project. Notification of these and other potential problems should improve risk awareness in project teams. To facilitate risk management, LINK-UP should notify teams of potential problems, highlight records from similar past projects that experienced the same problems, and continue to monitor the likelihood and impact of risks throughout the course of the project.

Exposure to multiple successful and unsuccessful past projects would allow teams to compare processes and make informed decisions based on the needs of their specific project. Teams could study concrete examples of processes that succeeded and processes that failed, examining reliable methods and common problems that occurred in projects of similar structure. By viewing knowledge from past project records at the start of their project, teams should gain a better initial understanding of the processes and the risks involved in design work. By having access to past project records throughout the life of their project, teams could not only compare progress and design evolution, but also discover and compare multiple solutions to problems when they occur. Teams who are aware of potential problems at the start of the project, and who monitor changes in the impact and likelihood of those risks throughout the life of the project, will be better prepared to manage those risks and complete their project successfully

5. Management Support Strategy

Effective project management requires a systematic process and supporting tools that facilitate team coordination and leverage knowledge from previous projects to identify potential problems and propose viable solutions. By maintaining a physical process record throughout the life of a project and by reusing knowledge archived from the records of past projects, teams could simultaneously and automatically reuse process knowledge and contribute further process knowledge for reuse. To accomplish these goals, tools to support project management in distributed teams should adhere to the following guidelines:

- **Coordinate team through activity awareness –** Aiding distributed teams in the externalization and maintenance of a collective team memory will help team members to remain aware of the activities and perspectives of everyone on their team. A team memory should include knowledge related to progress, individual contributions, decision rationale, and design evolution. The creation and maintenance of a team memory should be a natural by-product of the design process, adding minimal overhead to the project while helping teams to coordinate tasks and dependencies.

- **Organize process knowledge using time-based visualization techniques –** Organizing the team memory intuitively will allow teams to monitor, reflect on, and improve their processes throughout the course of a project, while visualizing process-related knowledge according to time takes advantage of episodic memory. An effective activity timeline should allow team members to quickly understand design changes, notice potential problems, and retrieve more detailed information on demand. The activity timeline should help team members to maintain a ‘‘big picture’’ view of the project as it evolves over time.

- **Archive process knowledge as a physical project record for reuse –** Maintaining the team memory throughout the life of a project and archiving project records for reuse will allow future teams to study the designs of similar projects, model their processes from those of successful teams, and avoid common mistakes. Additionally, decomposing project records into reusable chunks of process knowledge and classifying that knowledge for storage in a repository is an important step in improving process knowledge reuse.

- **Facilitate risk management by leveraging knowledge from previous projects –** Automatically presenting new teams, at the start of their project, with records from similar past projects will help team members to understand project expectations and risks. Records for both successful and unsuccessful similar projects should be retrieved based on relevant project parameters, such as team size and project domain and scope. Leveraging process knowledge from similar projects to maintain a visible list of

project risks and alert teams to an increase in the probability of potential problems will help teams to become and remain aware of risks throughout the life of a project.

6. Conclusion

Software project management is increasingly complex. As systems grow, team size and distribution rise, and the manifold complexities increasingly overwhelm us, we must rely more and more on our ability to share knowledge, coordinate efforts, and synthesize diverse and conflicting perspectives in the design of software-intensive systems.

In this paper, we examined what it will take to apply notification systems to support project management. Motivated to support project management and expand process knowledge reuse, we discussed how we could apply the LINK-UP technology to a number of project management efforts to evaluate the effects on usability and performance. As our effort unfolds, we hope to show that visualizing changes within a claim set can improve activity awareness and that leveraging knowledge from past projects can improve risk management. These improvements, in turn, should have a positive effect on team performance.

The knowledge gained through team collaboration should not be forfeited at the end of a project, but instead, archived for reuse. Reusing knowledge from previous projects to support risk management is an initial step in the development of a systematic approach to process knowledge reuse. By archiving project records and extracting chunks of knowledge in the form of project risks, we can begin to investigate the optimal size and structure of process knowledge to be incorporated into LINK-UP's reuse repository.

Supporting project management in LINK-UP is an important step toward improving project management for distributed teams and toward extending the reuse paradigm to include not only project-related knowledge, but also process-related knowledge. The result could help to bridge the gap between software engineering and HCI by contributing to the state-of-the-art in collaborative teamwork, software project management, and reuse.

7. Acknowledgements

We wish to thank Ali Ndiwalana, Shahtab Wahid, and Jason Lee for their careful review and constructive comments on this work.

We also thank the Virginia Tech ASPIRES program, in part, for funding this research.

8. References

[1] Arias, Ernesto, Eden, Hal, Fischer, Gerhard, Gorman, Andrew, and Scharff, Eric. ‘‘Transcending the individual human mind – creating shared understanding through collaborative design.’’ ACM Transactions on Computer-Human Interaction (TOCHI), Vol. 7, No. 1, March 2000, p. 84 - 113.

[2] Basili V. R.: "The Experience Factory: packaging software experiences." In Proceedings of the NASA Goddard Space Flight Center's 14th Annual Software Engineering Workshop, 1989.

[3] Beise, Catherine M. ‘‘Employees and impact on work: IT Project Management and Virtual Teams.’’ In Proceedings of the 2004 SIGMIS conference on Computer personnel research: Careers, culture, and ethics in a networked environment, April 2004, p. 129-133.

[4] Boehm, Barry, and Port, Daniel. ‘‘Educating Software Engineering Students to Manage Risk.’’ In Proceedings of the 23rd International Conference on Software Engineering, May 2001, p. 591-600.

[5] Bohner, S., ‘‘Extending Software Change Impact Analysis into COTS Components,’’ IEEE/NASA Software Engineering Workshop, December 2002.

[6] Carroll, J. M. ‘‘Making use: a design representation.’’ Communications of the ACM, Vol. 37, No. 12, December 1994, p. 29-35.

[7] Carroll, J.M. Making use: scenario-based design of human-computer interactions. The MIT Press, 2000.

[8] Chewar, C. M., Bachetti, Edwin, McCrickard, D, Scott and Booker, John. "Automating a Design Reuse Facility with Critical Parameters: Lessons Learned in Developing the LINKUP System." In Proceedings of the 2004 International Conference on Computer-Aided Design of User Interfaces, January 2004.

[9] Corning, P.A., ‘‘Complexity is Just a Word!’’, in Technological Forecasting and Social Change. 2001, Institute for the Study of Complex Systems: Palo Alto.

[10] Davenport, Thomas H, and Prusak, Laurence. Working knowledge: how organizations manage what they know. Harvard Business School Press, 1998.

[11] Fussell, Susan R., Kraut, Robert E., Lerch, F. Javier, Scherlis, William L., McNally, Matthew M., and Cadiz, Jonathan J. ‘‘Coordination, Overload and Team Performance: Effects of Team Communication Strategies.’’ Proceedings of the 1998 ACM conference on Computer supported cooperative work, November 1998, p. 275 - 284.

[12] George, B., Bohner, S., and Prieto-Diaz, R., ‘‘Software Information Leaks: A Complexity Perspective,’’ 9th IEEE International Conference on Engineering of Complex Computer Systems, Florence, Italy, April 2004.

[13] Geyer, Werner, Richter, Heather, Fuchs, Ludwin, Frauenhofer, Tom, Daijavad, Shahrokh, and Poltrock, Steven. ‘‘A Team Collaboration Space Supporting Capture and Access of Virtual Meetings.’’ In Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work, September 2001, p. 188–196.

[14] McCrickard, D. Scott, Chewar, C. M., Somervell, Jacob P., and Ndiwalana, Ali. "A Model for Notification Systems Evaluation--Assessing User Goals for Multitasking Activity." ACM Transactions on Computer-Human Interaction (TOCHI), Vol. 10, No. 4, December 2003, p. 312-338.

[15] Malone, Thomas W., and Crowston, Kevin. ‘‘The interdisciplinary study of coordination.’’ ACM Computing Surveys, Vol. 26 No. 1, March 1994, p. 88-119.

[16] Norman, D. A. Things That Make Us Smart: Defending Human Attributes in the Age of the Machine. Addison-Wesley Longman Publ. Co., Inc., Reading, MA, 1993.
[17] Powell, Anne, Piccoli, Gabriele, and Ives, Blake. ‘‘Virtual teams: a review of current literature and directions for future research.’’ The DATA BASE for Advances in Information Systems. Vol. 35, No. 1, Winter 2004, p. 6-36.
[18] Steinfield, Charles, Jang, Chyng-Yang, Pfaff, Ben. ‘‘Supporting virtual team collaboration: the TeamSCOPE system.’’ Proceedings of the international ACM SIGGROUP conference on Supporting group work, November 1999, p. 81-90.
[19] Sutcliffe, Alistair. ‘‘On the effective use and reuse of HCI knowledge.’’ ACM Transactions on Computer-Human Interaction, Vol. 7, No. 2, June 2000, p. 197-221.
[20] Walz, Diane B., Elam, Joyce J., and Curtis, Bill. ‘‘Inside a software design team: knowledge acquisition, sharing, and integration.’’ Communications of the ACM, Vol.36, No.10, October 1993, p. 63-77.
[21] Wong, Kenny, Blanchet, Warren, Liu, Ying, Schofield, Curtis, Stroulia, Eleni, and Xing, Zhenchang. ‘‘JReflex: toward supporting small software project teams.’’ In Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange, October 2003, p. 50-54.
[22] Zhang, Jeff, Zage, Dolores, and Zage, Wayne. ‘‘Improving project planning/tracking for student software engineering projects through SOPPTS.’’ In Proceedings of the 16th IEEE Conference on Software Engineering Education and Training, March 2003, p. 185 – 19.