Project Management for the 21st Century: Supporting Collaborative Design through Risk Analysis

Jamie L. Smith Shawn A. Bohner D. Scott McCrickard
Center for Human-Computer Interaction and Department of Computer Science
Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0106 USA
540-231-7409

jls05@vt.edu sbohner@vt.edu mccricks@cs.vt.edu

ABSTRACT

Managing software project teams is a complex task further complicated by a continued increase in the size and complexity of software-intensive systems and the distribution of project teams. Given limited project resources, distributed teams require appropriate team processes and adequate tool support to help them remain focused on the most critical design tasks, thereby structuring the design process and improving team coordination. However, existing project management tools typically fall short. Software project management as a discipline is not unlike human-computer interaction (HCI) in that both combine technical concerns with human psychological concerns. Both could benefit from a more systematic approach to applying theory to practice. One proposed approach to the science of design involves constructing a record of design rationale by leveraging design knowledge from previous projects. Extending the reuse paradigm from product-related knowledge to process-related knowledge could improve software project management by helping teams to externalize and maintain a physical record of their design process. A risk management model could help teams to prioritize design knowledge, allowing them to focus their effort on key design tasks.

Categories and Subject Descriptors

K.6.1 [**Management of Computing and Information Systems**]: Project and People Management

General Terms

Management, Design, Human Factors

Keywords

Project management, distributed teams, collaboration, risk analysis, risk management, notification systems, design knowledge reuse

1. INTRODUCTION

As the size and complexity of software-intensive systems continues to increase, it has become 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 [14]. An increase in the use of distributed teams, which, unlike traditional, co-located project teams, have the added difficulty of collaborating across the boundaries of space and time, has further complicated the issue of project management. 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 [15]. However, most existing collaborative tools do not adequately support the needs of distributed project teams.

Each member of a team adds a distinct set of knowledge and experience to the design process. As the project evolves, each member will develop distinct ideas and opinions concerning project goals, task priority, and other key decisions. Poorly coordinated teams do not communicate or make team decisions effectively. The members of a poorly coordinated team focus on individual tasks and have little awareness of the activities and perspectives of their teammates or of how the pieces of the project fit together. Unable to work as a cohesive unit, these teams find it difficult to focus on overall project goals.

In contrast, a well-coordinated team remains focused on the project as a whole. All members of a well-coordinated team not only share the same knowledge, but also *know* that they share the same knowledge [13]. Consequently, this team spends 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 [9]. By maintaining a “big picture” view of the project, this team can focus on the tasks that will help them to accomplish key project goals.

One key (and often overlooked) aspect of a design project is risk management. All projects have risks; some are more probable, influential, or costly than others. 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).

A risky action or event involves an associated loss, an element of uncertainty or chance, and a choice to be made [6]. Each member of a group of project stakeholders may hold a different opinion concerning the loss associated with a certain risk, the level of uncertainty involved, or the choice that should be made. To manage risk effectively, teams must discuss potential problems, agree on the priority of key risks, assign responsibility for risk mitigation, and monitor progress throughout the project. Although some traditional, co-located teams can manage risks with written documentation or the use of a risk database, distributed teams require adequate collaborative tools to support communication and to aid in the identification and management of project risks.

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 tools and proposes a strategy for improving collaboration by facilitating risk management.

2. RELATED WORK

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, tools used by distributed teams must support effective collaboration without adding excessive overhead. An effective collaborative environment must inject elements of project management, including activity awareness, task allocation, and risk management, directly into the design process.

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 [12]. 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.

SOPPTS [18], for example, is a task-oriented project management system for student software engineering teams. At the start of a project, teams produce a list of 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 can see which tasks have been completed, whether each task was completed on time, and if certain tasks, or team members, have fallen behind schedule.

Public task assignments reduce misunderstandings about who is responsible for completing which tasks and can 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.

TeamSCOPE [15] provides teams with a shared file repository, dedicated message boards for each shared file, and a detailed activity history, thus improving both communication and awareness among the members of a distributed team. At login, team members are presented with an overview of awareness data, including a summary of recent activity. The activity summary lists 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 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 [10], which supports the synchronization and documentation of team meetings, organizes information presented during a meeting into a timeline. 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 exploits our ability to organize past experiences into a sequence of episodes. Organizing information into a timeline aids team members in maintaining an overall view of the project and retrieving more detailed information as needed.

None of the tools discussed here explicitly incorporate risk management, either exclusively or in conjunction with other project management capabilities. However, a few tools, such as SoftRisk [11], are making strides toward risk automation. SoftRisk aids software developers in risk identification, prioritization, and monitoring throughout an iterative project lifecycle. Based on responses from a set of checklists and questionnaires, potential risks are identified and assigned a risk exposure value. Risk exposure is determined by both the probability that the risk will become a problem during the life of the project and the impact that the risk will have on the project if it does become a problem. Risks are then prioritized according to their risk exposure values. The tool visually monitors changes in risk priority throughout the life of the project.

SoftRisk was developed for use with any size or type of software project; thus, the specific benefit for any single domain is limited.

Elements of the checklists and questionnaires used in identifying and estimating risks are necessarily general. However, despite these limitations, the underlying concepts that drive SoftRisk are fundamentally important. Applied to a more specific domain and integrated within a collaborative design environment, a risk management tool such as SoftRisk could provide significant benefit in terms of project management.

Most project management tools, like those discussed here, support team coordination to varying degrees through activity awareness. However, these tools do not fully support team members in maintaining a "big picture" view of project goals and in focusing their efforts on the key tasks that will help them to achieve those goals. Consequently, teams often take an ad hoc approach to design. Integrating key elements from each of the tools discussed here – specifically, task allocation, activity history, and timeline visualization, along with risk management capabilities similar to those found in SoftRisk, could significantly improve the collaborative design process and further efforts to transform design into a scientific discipline.

3. TOWARD A SCIENCE OF DESIGN

Human-computer interaction (HCI) as a discipline is concerned with designing interfaces to interactive systems that allow users to accomplish their goals. A key aim of HCI is to inject knowledge from psychology, sociology, and other relevant disciplines 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 [5, 17].

To gain acceptance into current software engineering practices, the science of design must facilitate reuse. Within the software domain, reuse has seen considerable success in the form of reusable code modules and object-oriented pattern libraries. However, it is generally accepted that system developers can reduce development time and cut costs on a larger scale by incorporating reuse at an earlier stage of the development process [8, 17]. Consequently, new problems do not have to be solved from scratch. In the context of a design project, knowledge can be related to the product being designed or to the process by which the design team accomplishes its goals. Archiving and reusing knowledge about a design product and a design process can help to ensure that effective ideas are remembered and that mistakes are made only once.

3.1 Design product knowledge

Carroll's method for *claims analysis* [4] outlines a systematic approach to design. *Claims* summarize particular aspects of design rationale, explicitly stating the positive and negative tradeoffs of a 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, the compilation of a sufficient set of claims exemplifies the rationale behind a system design [5].

Claims represent design rationale grounded in scientific theory or experimental evidence. Although claims are tied to a specific context of use, the underlying design knowledge 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 [16].

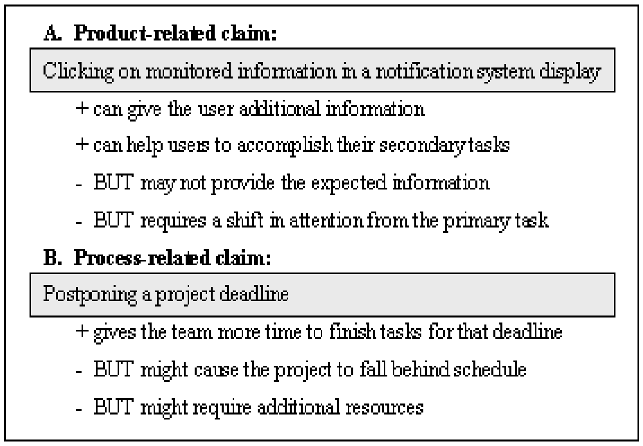
3.2 Design process knowledge

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] attempts to facilitate process improvement in software development by structuring, classifying, and storing packaged *experiences* from previous projects for reuse. Experiences, which include both product and process-related knowledge, are input into a repository in various forms, including as artifacts, models, and 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 analogous 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, for example, as claims, which can then be stored, retrieved, and reused in a variety of projects across domains.

3.3 Extending the reuse paradigm

Successful knowledge reuse relies on the appropriate definition of both product and process-related knowledge as well as an analysis of how the two types of knowledge can be juxtaposed to improve the design process. One possible strategy is to decompose design projects into a set of product-related claims and a set of process-related claims and to treat the downsides of these claims as project risks. In this way, the claims analysis process adds structure to the design process and incorporates an element of risk management at no additional cost. Moreover, both types of claims can be archived for reuse.



A. Product-related claim:

Clicking on monitored information in a notification system display

+ can give the user additional information
+ can help users to accomplish their secondary tasks
- BUT may not provide the expected information
- BUT requires a shift in attention from the primary task

B. Process-related claim:

Postponing a project deadline

+ gives the team more time to finish tasks for that deadline
- BUT might cause the project to fall behind schedule
- BUT might require additional resources

Figure 1. Comparison of product and process-related claims.

Consider, for example, the two claims in Figure 1. Claim A relates to the design product – a notification system. The two downsides of the claim represent potential problems that directly affect the quality of the product. Claim B, on the other hand, relates to the process by which the notification system is designed. The two downsides of this claim might also affect the quality of the resulting product, but in a more indirect manner. For example, if additional project resources were required but unattainable, then postponing a deadline could result in the cancellation of the project.

A third claim, about accelerating the project schedule to stay on schedule for the next deadline, could be added to the set to mitigate the Claim B downside about the project falling behind schedule. However, the accelerated schedule might still require additional resources; therefore, a fourth claim would be needed to mitigate that risk.

3.4 Managing design risks

A system design might involve dozens of claims. With limited time and resources, along with the inherent nature of design, a team cannot expect to mitigate all of the risks associated with their project. Consequently, the team must prioritize their claims and, at any given time, focus on the most critical project risks.

A claim often has multiple downsides, or risks, each of which can be assigned a specific weight. The priority of a claim can then be determined by the sum of the weights of its downsides. The weight of a downside is a combination of the probability that the given downside will become an actual problem in the design and the impact that the downside will have on the design if it does become a problem. Any number of factors could be used to determine the probability and impact values for a specific risk, and these factors will differ for process and product-related risks. We will first focus on a risk model for product-related claims about notification systems since significant work has been done in defining claims within this domain.

Claim upsides and downsides are grounded in either scientific theory or experimental evidence. A claim might initially consist of only one validating source; however, as the claim is reused and revalidated within other projects, its reference list might grow to include multiple sources from different domains. The impact value for a particular claim downside should take into consideration the number of validating sources, the similarity of the source domains to the new, untested domain, and the trustworthiness of each source. It should also consider the results of any evaluation of the claim within the context of the current design project, taking into account problems that have been shown to exist with the current system when analytically evaluated by HCI experts or empirically evaluated by potential users. The probability value for a claim downside should consider the statistical power of any evaluation the claim has undergone, reflecting a level of confidence that the evaluation yielded correct results. The probability value should also reflect the degree to which the downside has been mitigated in the design since a mitigated risk is less likely to become a problem.

Project management, like HCI, is a complex discipline in need of a more systematic approach, and effective risk management could be a step in furthering both the science of design and the science of software project management. The success of these proposed methods for risk management and knowledge reuse relies on the development of adequate 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 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 [7], is being developed to guide designers through a usability engineering process for the design of notification systems. LINK-UP facilitates the use, validation, and improvement of the claims analysis method by supporting the actual construction of a claims analysis record during the design process. The system is tied 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 [7].

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 [17]. Given the growing complexity of system design, the increased distribution of project teams, and the push to complete projects in less time with fewer resources, collaborative design tools must aid teams in focusing their effort on the key design tasks to achieve project goals.

Incorporating a risk management model within LINK-UP has the potential to improve performance in design teams by helping them to focus their efforts on key design tasks, thereby structuring the design process and improving team coordination. 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 and the extension of the reuse paradigm as the risk management model is applied to product-related knowledge and later extended to include process-related claims.

An effective risk management tool should benefit distributed project teams by helping them to focus their design efforts on key project issues, thereby:

1. Structuring the design process with key steps for multiple design iterations
2. Supporting team coordination by maintaining an external, collective team memory

4.1 Structuring the design process

LINK-UP guides design teams through the design process, from defining user requirements to performing an analytic or empirical evaluation of an initial design prototype. However, given the results of an evaluation, designers are left to navigate subsequent design iterations on their own. Designers are aware of certain inadequacies in the current design of their system; however, they are given little guidance in terms of how to resolve those issues.

The integration of a risk management model within LINK-UP could give teams the guidance they need during the redesign process. The results of an analytic or empirical evaluation show that a subset of the project claims are performing inadequately. Following an evaluation, LINK-UP could prioritize the claim set, based on a combination of stored data and team input, with higher priority given to those claims that need to be "repaired." Upon examining the prioritized list, teams will immediately know which risks are most critical for the current design iteration and the order in which claims should be addressed to resolve key design issues. Team members can then choose or assign specific claims that they will be responsible for mitigating. Mitigation might involve finding new claims to reuse or creating new claims to mitigate the most critical downsides of the highest priority claims.

The priority list will initially include only product-related claims; however, it could eventually be extended to include process-related claims as well, allowing teams to identify and manage problems with their design process in addition to their design product with minimal overhead. Once process-related claims are created and stored in the reuse library, it will become easier for teams to identify recurring risks in subsequent projects.

4.2 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 [8]. 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].

To facilitate 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.

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.

5. MANAGEMENT SUPPORT STRATEGY

Effective project management requires a systematic process and supporting tools that add structure to the design process and facilitate team coordination. To the greatest extent possible, project management tasks must be incorporated into the design process with minimal added overhead.

To accomplish these goals, tools to support project management in distributed design teams should adhere to the following guidelines:

- **Guide the design process with a risk management model**
  Guiding design teams through the steps of a redesign process will promote iterative design. Prioritizing project risks draws attention to the key problems in the current design that should be addressed in the next iteration. Teams can quickly determine and allocate key tasks for redesign. Consequently, teams can remain focused on the most critical aspects of the project.

- **Support team coordination 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, task assignments, 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 project-related knowledge using time-based visualization techniques**
  Organizing an intuitive team memory will allow teams to monitor, reflect on, and improve their processes throughout the course of a project, while visualizing project-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 product and process-related knowledge for reuse**
  Maintaining a team memory throughout the life of a project and archiving product and process-related knowledge for reuse will allow future teams to find valuable knowledge and identify common mistakes early in the design process. A growing repository of both product and process-related knowledge contributes to the science of design.

6. CONCLUSIONS AND FUTURE WORK

Software project management is an immature, but increasingly important discipline. As system complexity and team size and distribution continue to increase, we 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. Appropriate team processes and adequate tool support are critical to the success of software design. Furthermore, the knowledge gained through team collaboration should not be forfeited at the end of a project, but instead, archived for reuse.

Motivated by the goals of supporting project management, furthering the science of design, and expanding knowledge reuse in LINK-UP, we plan to incorporate a project management tool into the system, leveraging relevant ideas from the tools discussed in Section 2 and following the guidelines outlined above. The first step in this process is to fully define and evaluate an appropriate risk management model for product-related claims. This model should help design teams to focus on the key avenues for improvement within their project, thereby adding structure to the design process and improving team coordination.

Another critical task is to compare and contrast product and process-related knowledge and to define the structure of a process-related claim that compliments the existing product-related claim composition. A risk management model can then be developed for prioritizing process-related claims. Finally, the reuse paradigm and the LINK-UP system can be extended to include process-related knowledge. These improvements, in turn, should have a positive effect on team performance.

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 in the design of interactive software-intensive systems.

7. ACKNOWLEDGEMENTS

We 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] Carroll, J. M. “Making use: a design representation.” *Communications of the ACM*, Vol. 37, No. 12, December 1994, p. 29-35.

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

[6] Charette, Robert N. *Software Engineering Risk Analysis and Management*. Multiscience Press, Inc., 1989.

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

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

[9] 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.

[10] 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.

[11] Keshlaf, Ayad Ali, and Hashim, Khairuddin. “A Model and Prototype Tool to Manage Software Risks,” *Proceedings of the First Asia-Pacific Conference on Quality Software,* October 2000, p. 297-305.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.