Collaborative Software Development: A Discussion of Problem Solving Models and Groupware Technologies

Joanna DeFranco-Tommarello
joannadt@cis.njit.edu
New Jersey Institute of Technology
3331 Rt.38
Mt. Laurel NJ, 08054

Fadi P. Deek
Fadi.deek@njit.edu
New Jersey Institute of Technology
University Heights
Newark, NJ 07102
(973) 596--2997

Abstract

Teamwork is always challenging. Adding the complication of problem solving and software design only amplifies this challenge. The challenges of developing software as a team can be reduced by using groupware to coordinate and communicate the intricate details involved in the process. This study set out to determine if in fact there are tools available to assist in the collaborative problem solving and software development process. It must be understood at the outset of this paper that problem solving is at the heart of software development. Without amplifying the collaborative problem solving steps required for developing an effective and efficient solution, a much less then accurate solution will result. This paper provides a review of collaborative problem solving techniques and groupware in the software development domain covering both methodology and technology. There are three areas of focus in this paper. Specifically, collaborative problem solving and decision making; groupware theory and tools; and group cognition and psychology will be explored. The review is then followed by a careful analysis of the collaboration models and tools as well as their potential impact on software development. This paper ends with a discussion of our future work that will include developing a tool, which incorporates a problem solving model and collaborative structures for the software development domain.

1. Introduction

The old adage "two heads are better than one" has been realized by many, especially in the software development domain. With the rapid increase of software complexity and e-commerce, applications and web sites must be developed as efficiently and timely as possible. Some software can contain millions of lines of code. The objective is to have these large applications developed by several people. Collaboration in the area of software development is therefore a necessity rather than just a benefit of technology.

When developers are geographically separated, their communication success may depend on utilizing effective groupware [21]. Groupware is any software tool that supports teams whose members work collaboratively on interconnected personal workstations [35]. Groupware systems are made up of many components ranging from the simple to the very complex. These systems can successfully assist a team to work in a shared workspace toward a common goal. Physical space and time may separate team members using such systems. Well-integrated groupware systems promote effective communication, coordination and collaboration within an organization [1]. Groupware and collaborative software development remain relatively new research areas and additional benefits can be achieved. Next, collaborative problem solving/decision making, groupware theory/tools, and group cognition/psychology will be reviewed.

2. Collaborative problem Solving and Decision Making Methodology

The contemporary computing professional must work in an environment where programs are thousands or millions of lines long, are often extensively modified and maintained rather than

0-7695-1435-9/02 $17.00 (c) 2002 IEEE

IEEE COMPUTER SOCIETY

constructed, are manipulated in a tool-rich environment, where work is usually a team effort, and where the form of a solution has profound impact on future cost and performance [18]. These large systems need to be maintained and may have new or changing requirements, and thus require group collaboration. Collaboration has been shown in experiments to improve the problem solving processes of teams [20] [33]. Nosek (1998) found all teams outperformed the individual programmers, enjoyed the problem-solving process more, and had greater confidence in their solutions. Another collaborative experiment performed in introductory computer science classes, showed the collaborative learning group had more improvement (pre-test to post-test) than did the control group and rated the course somewhat higher [27]. These results could suggest that it might be worthwhile to integrate collaborative activities early into the computing curriculum, where problem solving and programming is first taught.

Problem solving is at the heart of software development. While learning to develop software, beginning programmers often encounter difficulty understanding basic problem solving concepts. Their deficiencies are in problem solving strategies and tactical knowledge; misconceptions about syntax, semantics, and pragmatics of language constructs; and ineffective pedagogy of programming instruction [4]. Collaborative problem solving is an activity where groups of two or more people develop a plan for the design of a complex system that will solve an existing problem. A collaborative problem-solving model is a procedure that is followed in order to facilitate the collaborative problem solving process. When solving problems as a group, some find the process to be far more complicated and chaotic than individual problem solving [7]. Another barrier to effective collaboration is the existence of those conditions that prevent the free expression of ideas in a group [10]. For example, conditions listed by Hoffman (1965): participation biases, personal characteristics and group structure. However, the benefits of collaboration during problem solving far outweigh the process inefficiencies [11]. Groups appear to be able to deal with complex tasks more effectively than individuals simply because groups have a larger range of skills and abilities [7]. Another benefit could be learning, or at least be exposed to the skills and abilities of the other group members [24]. For example, learners working in groups need to articulate designs, critiques, and arguments to other group members encouraging the kind of reflection that leads to learning [8].

Simon (1997) described three-steps model for coordination when cooperatively making decisions that relieves each member of the group the task of anticipating the behavior of the others as a basis for the member's own. In addition to making decisions coordination is needed in software development. To develop successful software, tight coordination among the various efforts involved in the software development cycle is essential [17]. Coordination also enables each group member to adopt the decision made by the group. Finnegan and O'Mahony (1996) developed a model of group problem solving by observing group decision-making and group supporting technology in organizations. While they felt the steps involved in the problem solving process are important, the means by which the group progresses from problem realization to solution is through the communication of information and ideas, as well as the collaboration of group members. Wong (1994) developed a logical, qualitative group problem-solving scheme for making joint decisions and to promote conflict resolution. Cooperative decision making, according to Wong, considers the problem of cooperation. For example, cooperation between software engineers building a complex software application or e-commerce web site. In addition to the theoretical models used in the group problem solving effort, there are communication issues that need to be addressed. Coordinating collaborative activity requires communication among the parties and communication needs to be effective for the problem solving effort to be successful [15]. Zhang (1998) addressed group properties and group effectiveness using a purely cognitive perspective. In other words, the research did not only examine the properties of individuals in the group but how the individuals interacted with one another.

3. Groupware Theory and Technology

Groupware or Virtual Teaming Systems (VTS) has changed the way office work is viewed. The globalization of business means that team members are often found at different locations [21] [22]. Nunamaker observed that the steady growth in telecommuting and the use of off-site consultants has increased the occurrence of dispersed meetings. The principle functions of groupware, according to Zwass (1998), are information sharing, document authoring, messaging systems, computer conferences, group calendars, project management, and support for team building.

IEEE COMPUTER SOCIETY
Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

Groupware technology should be applied to the decentralization of software development tools [9]. In order to make groupware effective and efficient in the collaborative software development process, the issues of both collaboration and software development must be understood. Hahn, Jarke and Rose (1990) also suggest how cooperative work in software development should be shifted: recognize the development of the group as an organized social process and negotiation facilities must be available. From this theory they developed the following four requirements for computer aided group work in software projects: support of negotiations and responsibility contracts; account for social protocols that underlie group communication; proper tool support and properly controlled tool integration mechanisms must consider domain knowledge of the underlying software project, working procedures and languages used for specification, design and implementation; and single modeling efforts have to be combined i.e., requirement analysis, work package planning, programming, etc.

Following is a review of tools that facilitate collaborative problem solving and software development. This review will also be selective rather than exhaustive, and is intended only to illustrate some of the variety of available systems. Some of the applications reviewed were not exclusively developed to facilitate software development, nonetheless are being used for such purposes.

The first example of a collaborative tool is Groove [32]. Groove provides strong tools for collaboration. The Groove application resides on each client's machine and the network is used as a pipe between the clients. It encompasses many collaborative activities such as live voice over the Internet, instant messaging, text based chat, file sharing (text, pictures, presentations), web browsing, drawing, interactive brainstorming, games, and threaded discussions. It also has coordination tools that keep track of meeting action items, agenda, and schedules. Each member can be in different tools at the same time or the users can choose to navigate together to work in the same tool at the same time.

Lotus Notes is a customizable client for e-mail, calendaring, group scheduling, Web access, and information management. The latest revision of Lotus Notes is called *Notes R5*. Notes R5 is an integrated, Web-like environment that provides users with quicker access to and better management of many types of information including e-mail, calendar of appointments, personal contacts and to-dos and Web pages.

GroupSystems is an application consisting of a collection of electronic meeting tools. GroupSystems is based on extensive research by Professor Jay F. Nunamaker at the University of Arizona. To collaborate with GroupSystems, it needs to be installed on a shared server, which is then used by team members on connected workstations. The GroupSystems tools support many aspects of collaboration. The main tools include activities such as: brainstorming, categorizer, topic commenter, group outliner, alternative analysis, voting and survey. This system can be used both synchronously and asynchronously.

Rose, a Rational product, is an application that assists in visually modeling software to be developed. The models produced from using the Rose application could potentially identify requirements and communicate information, identify component interactions and relationships. Rose supports teamwork by the integration of most available version control applications. This enables each team member to operate in a private workspace that contains an individual view of the project. Changes made by team members are made available to other team members by using a version-control system. Rational asserts that visual modeling improves communication across the team through the use of a common graphical language.

Another Rational product, Requisite Pro, is a requirements management tool. This application assists in the integration of user requirements into the development process. The Requisite Pro environment has two main elements: *Tool Palette*, and *View Workplace*. The Tool Palette can be used to access the requirements database and the View Workplace in addition to creating requirements, updating their attributes and producing high-level reports. The View Workplace is used to view the database of requirements.

We-Met is a collaborative graphical editor tool that allows both asynchronous and synchronous modes of user interaction [25]. Users work asynchronously, then synchronously broadcast their work to the group. The advantage of having both synchronous and asynchronous features is that late users can enter a group that has already started, catching up by reviewing messages that occurred before s/he joined.

Another example is a cooperative design environment called Design Collaboration Support System (DCSS) [16]. This system focuses on design conflict detection and resolution, which is a critical

IEEE COMPUTER SOCIETY
Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

component of the cooperative design process. DCSS allows design agents to express design actions, assists in detecting design conflicts, and suggests potential resolution to the design conflicts detected. This type of conflict resolution is called domain level conflict, and refers to inconsistencies in a design. It differs from collaborative conflict, where interpersonal issues may be involved.

Computer-Supported Cooperative Training (CSCT) is a synchronous collaborative system designed to enable geographically separated users to work together on a large programming project [31]. The primary goal of the system is to allow beginning programmers to collaborate when designing software. Requirement elicitation is the context used to teach collaboration, the objective being to develop a requirements document for a software problem. There are four shared tools: Procedural activity, to establish operating procedures via a voting system; problem definition, to specify an agreed or problem statement; criteria establishment, to enter criteria for requirements; and solution activity, to establish a priority for requirements via the voting tool.

The Evolving Artifact (EVA) is a collaborative tool that supports software development. Developers use this system to understand problems and develop solutions. This is accomplished by constructing and refining so called design representations [23]. This system uses a hypertext environment where users can view and interact with prototypes and to document their comments, the belief being that access to a combination of prototypes and documentation increases problem understanding.

CoNeX [9] is another example of a tool for collaborative software development. It emphasizes integration of the semantics of the software development domain with aspects of group work, on social strategies to negotiate problems by argumentation, and on assigning responsibilities for task fulfillment by way of contracting. CoNeX contains three tools: the argument editor for negotiating, the contract manager to document dialog, and the conference system for informal messaging. Users can also browse a knowledge base to trace software project history.

To assist in the group problem solving effort, an asynchronous groupware tool, Web-CCAT, is proposed [6]. This tool assists in collaborative work among geographically distributed users. Web-CCAT consists of project management software, computer aided software engineering (CASE) tools, and GSS (Group Support System) tools. The goal of this tool is to provide a more enriched environment than a face-to-face meeting.

SOLVEIT [3] is a tool that facilitates problem solving in the context of software development. This tool is primarily used by individual programmers, but does support some aspects of collaboration in software development, such as solution integration and component testing. SOLVEIT provides a set of tools that are associated with a six-stage process for problem solving and program development.

4. Group Cognition and Psychology

Group Cognition and Psychology need to be considered when developing any type of groupware or VTS. A groupware system should use automation to accelerate collaborative thinking, consensus building, decision making, and planning [21] [22]. Developers' thought processes are a fundamental area of concern [30]. Cognitive processes are a major attribute of individual problem solving and program development [3]. Group problem solving also needs to take into consider the human cognitive activities involved in problem solving. Cognitive activities within a group are different and more involved than the cognitive activities of one individual solving a problem alone [11]. This is because one must consider not only the cognitive activities of an individual, but also the cognition activities that result from group interaction.

Stacy and Macmillian (1995) have studied cognitive biases in the area software engineering. Cognitive biases are the areas where humans consistently and predictably make errors. Their findings suggest the following three recommendations to eliminate common failure among software engineers: always opt for empirical investigation over intuition, seek disconfirmatory information, and recast one-sided guidelines as two-sided trade-offs. Nosek (1998) also explored this research area and developed a theory of group cognition. Group cognition is explained as a "combination of distributed and coordinated cognition that directly affects the creation/recreation of distributed and similar knowledge within a team". Nosek also discusses three items needed to create "reasonable knowledge" within a problem-solving group: distributed knowledge, distributed cognition, and coordinated cognitive processing amongst the group members.

0-7695-1435-9/02 $17.00 (c) 2002 IEEE

IEEE COMPUTER SOCIETY

5. Analysis

This section provides an analysis of the models and tools presented earlier. The models will be evaluated to determine the kind of collaborative skills and knowledge they require for the tasks of group problem solving. The tools will then be evaluated to examine their support for collaborative problem solving and, where appropriate, their relevance to software development.

5.1. Methodology

Finnegan and O'Mahony's (1996) model includes features that benefit and emphasize group activities such as group decision making, but does not provide for activity coordination. Coordination in software development is where a group of developers agree to a common definition of what they are building, share information, and integrate their activities [17]. Conflict resolution mechanism is also missing in this model. This would involve the negotiation of activities when determining solution alternatives. Setting specific guidelines for team interaction in collaborative problem solving is important. When group activities are not properly channeled and coordinated, the interaction among the members could affect the iterative process of solution finding and thus negatively affect the solution.

Wong's (1994) qualitative group problem solving model, on the other hand, does focus on conflict resolution. There are definite negotiation attributes to this methodology. But this method is also lacking a framework for coordination of activities between the group members as well as stressing the iterative process involved in problem solving. Team interaction is not taken into consideration either in this model.

Hohmann's (1997) theory of collaborative problem solving is one that focuses on communication and collaboration of the process. This method, for example, recognizes that there is a need to account for the fact that group communication changes every time a member is added to the team. However, this model does not provide for an explicit group problem solving process since Hohmann indicates that the individual problem solving models can be used. Conflict resolution is also not apparent in this model.

Zhang (1998) indicates that collaborative problem solving should not have explicit steps and thus the methodology highlights four steps that must be included when collaboratively solving problems. These steps are similar to what other models already include, such as breaking the problem down into individual representations. However, coordination among group members is also not emphasized here. Lack of coordination could lead to failure in integrating the modules developed by each team member [17]. Additionally, all of the reviewed models emphasize the importance of developing and monitoring the group dynamics. The next section examines the tools we have reviewed.

5.2. Technology

Groove's features provide a basic framework that *assists* in the kind of coordinated problem solving efforts required in software development. However, collaborative software development requires specific problem solving support including features or guidelines that enhance group cognitive activities for collaborative solving problems, and explicit guidance and tools for the problem solving and program development process, which Groove was not intended to explicitly support. For example, the brainstorming tool provided in Groove allows users to rank ideas, but does not provide an explicit mechanism for voting or other methods or tools for facilitation consensus. Nonetheless Groove does support group performance one of the major aspects of the problem solving process, including ways to identify the tasks of the proposed solution, distributing tasks, and coordinating, communicating and modifying the solution.

Notes is an application that assists in coordinating a group's efforts but does not provide tools for real time or asynchronous collaboration activities. The coordinating features only assist in the problem solving and design efforts of software development. More collaboration tools are necessary to successfully collaborate when problem solving in the area of software development. As previously mentioned, collaborative software development requires specific problem solving support including features or guidelines that enhance group cognitive activities for collaboratively solving problems. In addition, explicit guidance and tools are needed for the group problem solving and program development process, which Lotus Notes R5 was not intended to explicitly support. Additional tools to facilitate activities such as brainstorming, voting, task identification, task distribution, etc are needed.

GroupSystems is an excellent collaborative tool. Its main goal is to support mission-critical collaborative knowledge activities like strategic

IEEE COMPUTER SOCIETY

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

planning and risk assessment. Although this tool was not specifically intended for software development it has most of the features required to collaboratively solve a software problem. For example, it has a brainstorming tool that is organized to keep the team focused on the brainstorming topic. There are also ample voting tools to make the many group decisions required when solving a problem. Obviously there is not a step-by-step process to guide the team through specific problem solving and software development tasks but all of the tools needed to facilitate each aspect of collaborative problem solving and software development process is certainly available in the GroupSystems tool. Comparing it to other systems reviewed, the only features GroupSystems doesn't have are the chat feature, calendar, web browsing, and a task list. However, the features contained in GroupSystems have much more of an impact on the collaborative problem solving and software development process then the features it is missing.

Rose is a good development tool that has a main focus of visual modeling. Rational claims that visual modeling shortens the development life cycle as well as increases productivity, improves software quality and team communication. Team communication could very well be increased but only by team members sharing various documents developed with the tool. In other words, team communication is increased because of using a common modeling language. There is no provision to communicate via the tool. Hence there are no messaging, brainstorming, voting tools integrated into the Rose tool. Therefore, it is already assumed that the initial problem solving stage is complete when Rose is used. Rose is a tool to model solutions prior to the implementation stage of development.

Requisite Pro is an asynchronous groupware application for requirement management with the exception that only one person at a time can modify the requirement documents. In other words, the user is forced to either have exclusive rights to modify entire project or just read only access. The feature that positively balances the exclusive rights problem is the discussion tool. All users can post messages at any time. Requirement management is a very important task when developing software. Changing requirements during project development has been known to cause unsuccessful results such as "run away projects". Managing requirements can help to minimize these kinds of problems. Requisite pro is an extensive application to manage project requirements. Using this application as part of a collaborative software development model can only increase its success.

All the collaborative tools presented provide asynchronous communication capabilities but only some include synchronous communication features. Both modes have their positive points. The asynchronous mode allows group members to join a problem solving session at various times, either by design or necessity, and because of archived entries be able to develop an understanding and familiarity with the progress. Synchronous communication allows real time discussion, which could expedite the problem solving process. Of course, having both modes seems most beneficial. We-Met is one tool supporting synchronous interactions but supports only a limited range of collaborative features: meeting discussions, brainstorming, and collaboration archiving. Though it has been used as a problem solving tools, it does not provide explicit problem solving facilities. DCSS, on the other hand, provides an interesting problem solving feature that assists in detecting design conflicts. But the system does not support conflict resolution among group members, such as might occur during consideration of design alternatives. Like We-Met, DCSS provides no overall problem solving framework and neither was developed to support software development tasks.

Some of the tools we reviewed provide specific support for software development. CSCT is a collaborative tool that features synchronous communication, with the main objective being to facilitate requirement elicitation for software development. This tool does not provide any other facilities to support the rest of the software process. In addition, the absence of an asynchronous communication features prohibits any group member to catch up if they enter the process late. The main objective of this software development tool is teaching object-oriented design concepts. Problem solving facilities are not included.

EVA is an asynchronous group tool that has a primary feature for sharing software prototypes and the development process documentation. Even though, this tool does not assist in solving problems or developing software, it provides suggestions based on a repository of designs. CoNeX is a very rich environment for software development. This system integrates the semantics of the software development domain with aspects of group work and social strategies to negotiate problems. However, CoNeX only provides limited

0-7695-1435-9/02 $17.00 (c) 2002 IEEE

IEEE COMPUTER SOCIETY

problem solving support, i.e. it only considers group interactions not specific methodology. Web-CCAT is a tool that aids in developing software by providing CASE (computer aided software engineering) tools but is another example of an application not focusing on the problem-solving component of software development. SOLVEIT, is primarily a tool for individual problem solving and program development. This system is based on a specific model for problem solving and provides stage-specific tools that support both problem solving activities and software development activities.

6. Discussion and Conclusion

One could infer based on the benefits of collaboration in general that collaborative problem solving and software development would improve the software development process. In addition, experimentation has shown that developers valued other people as their most used source of help when developing software [17]. Collaboration during software development activities would assist in bringing products to market faster and may also assist software developers in creating solutions of more complex problems with a lower frequency of errors and at a faster rate. However, our review indicates that there is more emphasis on the technology for collaboration but less focus on the methodology and coordination activities. Current models need to consider the psychology and sociology associated with collaborative problem solving. For example, none of the software development groupware tools presented here has taken into consideration such issues as group cognition in collaboration. Jones and Marsh (1997) suggest that this is because a majority of groupware designers are technologists who have both the experience and tools to develop new and effective hardware and software, but these same designers do not have the expertise in social protocols to provide the support necessary in groupware systems. Group cohesion is another aspect of group psychology and sociology that needs to be considered.

Human Computer Interaction (HCI) practitioners, whose work combines psychology, social sciences, computer science and technology, have been addressing such concerns [2]. Shneiderman (1980) indicates that understanding of human factors, skills and capacity can improve the design of effective computer systems by applying the techniques and methods of cognitive, social, personnel and industrial psychology. Shneiderman also indicates that although focusing on psychological and social issues may increase design time and ultimately cost, the design quality will be improved. There is a need to apply software psychology to the design of collaborative software development.

7. Future Work

We are currently in the process of designing a collaborative problem-solving model for the software development domain, which will take into consideration not only the cognitive and collaborative processes of a group working together, but also the group dynamics that occur in teamwork. This model will define a structure for collaboration by identifying the "who", "how", "what", and "when" in collaboration. These items will be translated into both the modality of the collaboration and the dynamics of the group [5]. Figure 1 gives a hierarchical view of this collaborative problem solving structure. Modality of collaboration consists of possible interactions modes, i.e. asynchronous messaging or synchronous meetings. Group dynamics refers to the processes that define collaboration: negotiation, scheduling, coordination, integration etc. The side effects that occur from group dynamics are: cognitive bias, conflict resolution, group cohesion, distributed learning, etc. Collaborative Administration consists of: task initiation, delegation of functions, subcomponent integration, on going evaluation, etc., and the management of side-affects.

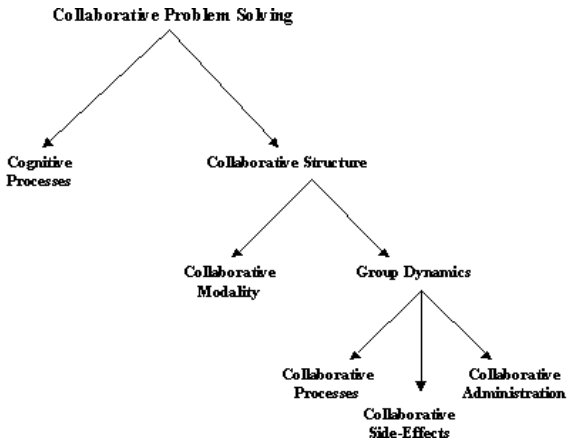

Figure 1. Collaborative Problem Solving Hierarchy

IEEE COMPUTER SOCIETY

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

A tool will also be developed to support a problem solving model as well as collaborative structures including modality, group dynamics, and collaborative processes (e.g. negotiation, coordination etc.). In addition, a statistical experiment will be performed to determine the effectiveness of the tool and model when collaboratively solving problems and developing software.

The planned experiment consists of two conditions using graduate computer and information science students enrolled in an Object Oriented Programming course at the New Jersey Institute of Technology. The experimental condition will have access to groupware systems providing collaborative communication tools. The control condition will only have access to email. Both conditions will have access to the collaborative model described. The subjects will be formed into teams of three and four people. Half of the teams will be part of the control condition and half will be part of the experimental condition.

The collaborative model will have specific tasks outlined for each stage of problem solving and software development. The model has six stages that progress from problem formulation to solution delivery. Each group will be asked to solve the same software problem. They will be asked to only complete the first two stages: problem formulation and solution planning. Each stage is further broken down into tasks. The output of each group will be evaluated by trained experts to determine the effect of the groupware tools on the following dependent variables: problem understanding, planning skills, decision quality, number of alternatives, solution creativity, and solution quality. A post-task questionnaire will be used to determine the outcome on the solution satisfaction and discussion and brainstorming satisfaction dependent variables. Group size will be an additional independent variable to test overall team performance and team satisfaction.

The expected outcome is that the dependent variables discussed earlier will be increased by the use of groupware tools coupled with the collaborative model. We expect to see an increase in solution quality and creativity from the experimental group as compared to the control group. We also expect those participating in the experimental condition to have a more satisfying experience as compared to those participating in the control condition. Therefore, based on the results of this experiment we expect to contribute a process for collaboration during software development that will include a cognitive model and tools to produce more creative and effective software.

8. Acknowledgments

This work was supported by New Jersey Information-Technology Opportunities for the Workforce, Education and Research (**NJ I-TOWER**), a New Jersey Institute of Technology (NJIT) project funded by the New Jersey Commission on Higher Education "High Technology Workforce Excellence Grant" initiative, award # 01-801020-02.

9. References

[1] Aannestad, B., Hooper, J., "The Future of Groupware in the Interactive Workplace", HRMagazine, Vol. 12, Issue 11, November 1997, pp. 37-41.

[2] Carroll, J., "Human-computer interaction: psychology as a science of design", International Journal of Human Computer Studies, Volume 46, pp. 501-522, April 1997.

[3] Deek, F.P., An Integrated Environment For Problem Solving and Program Development, Unpublished Ph.D. Dissertation, New Jersey Institute of Technology, 1997.

[4] Deek, F.P., Turoff, M., McHugh, J., "A Common Model for Problem Solving and Program Development", Journal of the IEEE Transactions on Education, Volume 42, Number 4., pp. 331-336, November 1999.

[5] Deek, F.P., DeFranco-Tommarello, J., McHugh, J., "A Model for a Collaborative Technologies in Manufacturing", In Review for the International Journal of Computer Integrated Manufacturing, September 2001.

[6] Dufner, D., Kwon, O., Hadidi, R., "WEB-CCAT: A Collaborative Learning Environment For Geographically Distributed Information Technology Students and Working Professionals", Communications of the Association for Information Systems, Vol. 1, Article 12, March 1999, Available [Online]: http://cais.isworld.org/articles/1-12/article.htm [26 November 2000].

[7] Finnegan, P., O'Mahony, L., "Group Problem Solving and Decision Making: an Investigation of the Process and the Supporting Technology", Journal of Information Technology, Vol. 11, Num. 3, September 1996, pp. 211-221.

[8] Guzdial, M., Kolodner, J., Hmelo, C., Narayanan, H., Carlson, D., Rappin, N., Hubscher, R., Turns, J., Newstetter, W., "Computer Support for the Learning through Complex Problem Solving", Communications of the ACM, Vol. 39, No. 4, 1996, pp. 43-45.

IEEE COMPUTER SOCIETY

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

[9] Hahn, U., Jarke, M., Rose, T., "Group Work In Software Projects: Integrated Conceptual Models and Collaboration Tools", Proceedings of the IFIP WG8.4 Conference on Multi-user Interfaces and Applications. Heraklion, Greece, September 1990, Amsterdam: North-Holland, pp. 83-101.

[10] Hoffman, L. R., "Group Problem Solving" Chapter from Advances in Experimental Social Psychology, Edited by Leonard Berkowitz, Academic Press, New York, 1965.

[11] Hohmann, L., Journey of the Software Professional, Prentice Hall PTR, New Jersey, 1997.

[12] Jones, S., Marsh, S., "Human-Computer-Human Interaction: Trust in CSCW", SIGCHI Bulletin, Volume 29, Number 3, July 1997.

[13] Kaplan, S., Tolone, W., Carroll, A., Bogia, D., Bignoli, C., "Supporting Collaborative Software Development with ConversationBuilder", ACM-SDE, Virginia, pp. 11-20, December 1992.

[14] Kelly, G., Bostrom, R., "Facilitating the Socio-Emotional Dimension in Group Support Systems Environment", SIGCPR '95, Nashville, TN.

[15] Kies, J., Williges, R., Rosson, M., "Coordinating Computer-Supported Cooperative Work: A Review of the Research Issues and Strategies", Journal of the American Society for Information Science, Volume 49, Number 9, 1998, pp. 776-791.

[16] Klein, M., Chapter 11 of Design Issues in CSCW – Edited by Dan Diaper And Colston Sanger, Springer-Verlag London Limited, Great Britain, 1994.

[17] Kraut R., Streeter, L., "Coordination in Software Development", Communications of the ACM, Volume 38, Number 3, March 1995.

[18] Mulder, M., Haines, J.E., Prey, J.C., Lidtke, D.K., "Collaborative Learning in Undergraduate Information Science Education", Papers of the 26th SISCSE technical symposium on Computer Science Education, 1995, pp. 400-401.

[19] Nosek, J., T., "Augmenting the Social Construction of Knowledge and Artifacts", Air Force Research Laboratory, Report Number AFRL-HE-WP-TR-1998-0082, February 1998.

[20] Nosek, J. "The Case for Collaborative Programming", Communications of the ACM, Vol. 41, No. 3, 1998, pp. 105-108.

[21] Nunamaker, J., "Collaborative Computing: The Next Millennium", Computer, Volume 32, Number 9, September 1999, pp. 66-71.

[22] Nunamaker, J., "The Case for Virtual Teaming Systems", IT Pro, September/October 1999, pp. 52 – 57.

[23] Ostwald, J., "Supporting Collaborative Design with Representations for Mutual Understanding", CHI' Companion, 1995, pp. 69-70.

[24] Prey, J.C., "Cooperative learning and closed laboratories in an undergraduate Computer Science curriculum", Proceeding of Integrating Technology into Computer Science education, June 1996, Spain, pp. 23-24.

[25] Rhyne, J., Wolf, C., "Tools for Supporting the Collaborative Process", Proceedings of the fifth annual ACM symposium on User interface software and technology, 1992, pp. 161-171.

[26] Rossi, A., "KPPCDL: An Internet Based Shared Environment for Introductory Programming Education", Proceedings of the 4th annual SIGCSE/SIGCUE on Innovation and technology in computer science education, 1999, pp.196.

[27] Sabin, R. E., Sabin, E., "Collaborative Learning in an Introductory Computer Science Course", SIGCSE symposium on Computer science education, 1994, pp. 304 – 308.

[28] Shneiderman, B., Software Psychology: Human Factors in Computer and Information Systems, Winthrop Publishers, Inc., Cambridge, Massachusetts, 1980.

[29] Simon, H. A., Administrative Behavior, Fourth Edition, The Free Press, New York, 1997.

[30] Stacy, W., Macmillian, J., "Cognitive Bias in Software Engineering", Communications of the ACM, Volume 39, Number 6, pp. 57-63, June 1995.

[31] Swigger, K., Brazile, R., Shin, D., "Teaching Computer Science Students How to Work Together", CSCL Conference Proceedings, October 1995, Available [Online]: http://www-cscl95.indiana.edu/cscl95/swigger.html [26 November 2000].

[32] Udell, J., Asthagiri, N., Tuvell, W., Peer-To-Peer: Harnessing the Power of Disruptive Technologies, O'Reilly & Associates, 2001.

[33] Wilson, J., Hoskin, N., Nosek, J., "The Benefits of Collaboration for Student Programmers", 24th SIGCSE Technical Symposium on Computer Science Education, February 1993, pp. 160-164.

Wong, Stephen, T.C., "Preference-Based Decision Making for Cooperative Knowledge-Based Systems", ACM Transactions on Information Systems, Volume 12, Number 4, October 1994, pp. 407-435.

IEEE COMPUTER SOCIETY

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE

[34] Zhang, J. "A Distributed Representation Approach to Group Problem Solving", Journal of the American Society for Information Science, Volume 49, Number 9, 1998, pp. 801-809.

[35] Zwass, V., Foundations of Information Systems, Irwin McGraw-Hill, Boston, Massachusetts, 1998.

IEEE COMPUTER SOCIETY

Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS-35'02)
0-7695-1435-9/02 $17.00 © 2002 IEEE