Spring 2002

Collaborative problem solving and program development model

Joanna DeFranco-Tommarello
New Jersey Institute of Technology

ABSTRACT

COLLABORATIVE PROBLEM SOLVING AND PROGRAM DEVELOPMENT MODEL

by
Joanna DeFranco-Tommarello

A model to enhance collaborative problem solving and program development is presented. The collaborative model is a detailed cognitive model that takes into consideration the cognitive and social activities that occur during collaborative problem solving and program development. The cognitive activities required for collaborative problem solving and program development are identified and integrated into a six-stage model. An extensive literature review in the associated fields is presented to show the need for the model described in this dissertation. In addition, a comprehensive study of tools to support collaboration during problem solving and program development was also performed as well as a critique of these tools.

A detailed statistical experiment to study the effect of this model on subjects collaboratively solving a software problem was designed and executed. The experiment included testing the collaborative problem solving and program development model with and without assistance from groupware tools. The subject teams each constructed a software design and this design was evaluated based on research hypotheses. This experiment produced results indicating the positive effect the Collaborative Model has on problem understanding and the quality of solution planning during collaborative problem solving and program development efforts.

COLLABORATIVE PROBLEM SOLVING AND PROGRAM DEVELOPMENT MODEL

by
Joanna DeFranco-Tommarello

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer and Information Science

College of Computing Sciences

May 2002

APPROVAL PAGE

COLLABORATIVE PROBLEM SOLVING AND PROGRAM DEVELOPMENT MODEL

Joanna DeFranco-Tommarello

Dr. Fadi P. Deek, Dissertation Advisor Date
Associate Professor of Information Systems, NJIT

Dr. James A. McHugh, Committee Member Date
Professor of Computer and Information Science, NJIT

Dr. S. Roxanne Hiltz, Committee Member Date
Distinguished Professor of Computer and Information Science, NJIT

Dr. Bartel A. Van de Walle, Committee Member Date
Assistant Professor of Computer and Information Science, NJIT

Dr. Cesar Perez, Committee Member Date
Assistant Professor, FDU

BIOGRAPHICAL SKETCH

Author: Joanna DeFranco-Tommarello

Degree: Doctor of Philosophy

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer and Information Science
  New Jersey Institute of Technology, Newark, NJ, 2002

- Masters of Science in Computer Engineering
  Villanova University, Villanova, PA, 1995

- Bachelor of Science in Electrical Engineering and Mathematics
  The Pennsylvania State University, University Park, PA, 1992

Major: Information Systems

Publications:

DeFranco-Tommarello, J., Deek, F., "Collaborative Software Development: A Discussion of Problem Solving Models and Groupware Technologies", HICCS-35, Jan 7-10 2002.

Deek, F., DeFranco-Tommarello, J., McHugh, J., "Requirements for Collaboration Technologies in Manufacturing", to appear in the International Journal of Computer Integrated Manufacturing, 2002.

DeFranco-Tommarello, J., Deek, F., McHugh, J., Van de Walle, B., "A Collaborative Problem Solving and Program Development Model", IC-SE '02, Las Vegas, 2002.

DeFranco-Tommarello, J., Deek, F., Hiltz, S. R., Keenan, J., Perez, C., "Collaborative Software Development: Experimental Results", Submitted to HICCS-36, 2003.

Dedicated to my father,

Joseph P. DeFranco,

for all of his advice and support, and especially teaching me the value of education.

ACKNOWLEDGMENT

I want to express my deepest gratitude to Dr. Fadi P. Deek, my advisor, for all of his guidance, support and constant encouragement during my academic career at NJIT. His work inspired me both to complete my Ph.D. at NJIT and to research his field of study. I also want to thank him for giving me the opportunity to teach full time as a Special Lecturer at NJIT. In addition, I want to express my sincere appreciation to my committee members, Dr. James McHugh, Dr. S. Roxanne Hiltz, Dr. Bartel Van de Walle, and Dr. Cesar Perez, for their outstanding guidance during this research.

I also appreciate the funding I received from NJ I-TOWER that gave me the opportunity to be both a teaching and research assistant during the 2000-2001 academic year.

Many thanks are extended to Dr. Julian P. Keenan for his expertise and feedback. I also want to thank Catherine L. Campbell and K. Steve Choi for their time and effort in judging the results of the experiment. I also appreciate the time of the many NJIT graduate students who participated in the experiment.

Certainly, without the support of my family and friends I could not have claimed this accomplishment. I want to thank my father, Joseph P. DeFranco, who encouraged me to earn a Ph.D. It is because of his advice both in my academic and engineering career that I got on the path where I wanted to be and never got off that path. I want to thank my mother, Anna F. DeFranco, for all of her support during my academic career. She always made sure I could focus on my goals by sharing her wisdom of life and the importance of family. A sincere thanks goes to my sister and friend, Judy Mastrocola, and my dear friends, Samantha, Kim, Dorothy, Ilene, Teri, Andrea, Janice, Patty, and Slick, for always helping me see the light at the end of the tunnel. I also appreciate the encouragement of my brothers, John and Joseph DeFranco, and especially for inspiring me to build a career in education.

Finally, I want to give my deepest thanks to my husband, Michael J. Tommarello, who always supported and encouraged me throughout my career and education. Without his love, help, and understanding I could not have made it through 14 years of college education. I look forward to focusing the rest of my life on him and our daughter, Michaela Anna.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

LIST OF TABLES
(Continued)

LIST OF TABLES
(Continued)

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

Problem solving is essential to software development. Indeed many of the basic processes that are the backbone of software development can be viewed as standard problem solving processes, ranging from requirements analysis, specification, and design to testing or verification (Deek, 1999; Deek, 1997). As software development has increased in complexity, an additional factor has grown in importance: collaboration. In fact, the increasing complexity of applications has necessitated the use of teams or groups to develop software because it is infeasible for individuals to develop large software systems with appropriate expediency or levels of quality.

The contemporary computing professional works in an environment where programs are thousands or millions of lines long, are often extensively modified and maintained rather than constructed, are manipulated in a tool-rich environment, and where work is usually a team effort (Mulder, Haines, Prey & Lidtke, 1995). Computer scientists are not well prepared for this contemporary environment according to Prey (1996) because their preparatory training usually focuses on the construction of small programs (programming-in-the-small) and provides little experience in complex software development. In contrast, the development of large systems in an efficient and timely manner requires a team effort, and the more complicated the problem, the larger the team needed to solve it. Another contributing factor to the need for team development is that domain-specific expertise tends to be localized and geographically distributed. Studies have shown that, particularly when such developers are dispersed, their success depends critically on their ability to use effective groupware (Nunamaker, 1999). Such factors have made collaboration in systems development a necessity, not merely a technically feasible option. The emergence of the World Wide Web has fortunately made geographically distributed collaborative systems technologically feasible in a way that was difficult or impossible until recently. The term *groupware* will be used to refer to the kind of software environments needed to support such a team, whose members collaborate over a network (Zwass, 1998). Groupware systems are intended to provide a team a shared workspace, despite being separated spatially and temporarily. Groupware or collaborative systems can be instrumental in alleviating the logistical difficulties that are associated with the application of distributed expertise. Indeed, the next generation of development processes is expected to focus on the effective integration of distributed expertise.

Experimental studies of both experienced programmers and novices have established the positive impact of collaboration. Wilson, Hoskin, and Nosek (1993) conducted a study to determine if experience with collaboration could benefit beginning programmers performing problem-solving/programming tasks. The experimental results provided positive support for the hypothesis that collaborative efforts could improve the problem-solving required in programming tasks. The experiment compared a control group of novice programmers, solving a software problem individually, with another group that allowed partners to communicate freely. The results demonstrated that even such simple collaboration enhanced the problem-solving performance of the novice programmer. The study also found evidence that an individual's ability had little overall effect on team performance, a phenomenon they claim occurred because the collaborative effort counterbalances individual deficiencies. The study also showed evidence that the collaboration provided the programmers confidence in the solution and enhanced their enjoyment of the problem solving process. Collaborative interactions appear to help beginning programmers analyze and model problems, and may also help them master the analytical skills required by such tasks (Wilson, Hoskin & Nosek, 1993). Other controlled experimental studies indicate it is worthwhile to integrate collaborative activities even at the early stages of problem solving and programming training (Sabin & Sabin, 1994). Experiments with experienced software engineers (Nosek, 1998) also demonstrate that collaboration improves the problem solving process. Indeed, all team projects evaluated in the study outperformed comparable individually implemented projects, while at the same time team members were more personally satisfied with their work and had greater confidence in their solutions.

The literature review in following chapter will focus on the research and development for collaborative or group problem solving in the area of software development with the objective of identifying important open issues and avenues for advancing both theory and practice. The review will examine collaborative problem solving and groupware in the software development domain, focusing on four areas: group problem solving, individual problem solving, groupware, and group psychology/sociology, including: group and individual problem solving models and tools, groupware systems, group cognition, and team dynamics. The contributions and outstanding issues in group problem solving and group software development, and the identification of the area of research that will represent an advance in the state of the art will be discussed. Subsequent to the review, a new collaborative problem solving and software development model, a system to best facilitate the model, and an experiment to test the proposed hypotheses are all presented.

CHAPTER 2

BACKGROUND LITERATURE

Problem solving is important in many fields and both domain-specific and generic problem solving methods have been developed over the years (Deek, 1997; Deek, Turoff, & McHugh, 1999). Most collaborative problem solving models are based on individual problem solving methods, so it is appropriate to begin our discussion with a consideration of individual problem solving prior to discussing the background literature on collaborative problem solving. Additional background literature that is included in this section is groupware systems, general groupware tools, and groupware tools specific for problem solving and software development.

2.1 Individual Problem Solving and Decision Making

Problem solving is central to software development, and a variety of domain-specific problem solving models for software development have been developed. The models are intended to support individuals in applying basic problem solving concepts in programming. They are intended to ameliorate recognized deficiencies in problem solving strategies and tactical knowledge as well as more widely recognized difficulties with the syntax, semantics, and pragmatics of programming language constructs (Deek, 1997; Deek & McHugh, 1999). Generally speaking, the comprehension of a problem requires the identification of the problem goal, givens, unknowns, conditions, constraints, and their relationships, and problem solution invariably requires some form of problem partitioning. Deek (1997) extensively reviews existing problem solving methodologies and develops a comprehensive problem solving model which integrates general problem solving methodology with program development tasks, and with the cognitive knowledge and skills needed at each stage of the process. The integrated model, called the Dual Common Model (DCM), identifies for each problem solving/program development task, the specific cognitive techniques required to accomplish that task. A brief overview of the problem solving tasks is as follows:

1. Formulating the problem: This stage leads to an organized representation of all relevant problem information: the goal, givens, unknowns, conditions and problem constraints.
2. Planning the solution: During this stage, the user identifies and evaluates or assesses alternative possible solutions, and also partitions the problem by refining the overall problem goal into sub-goals.
3. Designing the solution: This involves sequencing sub-goals, determining whether the sub-goals require further decomposition, establishing relationships among the various solution components and the associations between data and sub-goals.
4. Translation: At this stage, program development skills are used to translate the solution design into a coded solution.
5. Testing: At this stage the program is tested to verify that it meets the solution specifications.
6. Delivery: At this stage the solution and results are documented, presented or disseminated.

The Dual Common Model is heavily dependent on the classic work by the problem-solving mathematician and theorist George Polya (1945). Another seminal treatment of problem-solving was given by the Nobel prize-winner Herb Simon

(1960) who identified four components of a successful problem solving process:

1. Intelligence: The ability to recognize the existence of a problem, gather information pertinent to the problem, and produce an accurate definition of what the problem is.
2. Design: Generate possible alternative solutions, including preliminary solution plans for each.
3. Choice: Select and implement a suitable solution from the identified alternatives.
4. Implementation: Put choice into effect and produce the solution.

In later work, Simon explored a problem-space model of problem solving which viewed a solution as a sequence of transformations between partial problem solution states (Newell & Simon, 1972). Solving a problem consisted in identifying a set of operators that completed the transformation from an initial state (problem definition) to a final goal state (problem solution).

A domain-specific model of problem solving in the context of organizational operation is described in Barber (1984). This model, called office semantics, analyses organizational processes with the objective of understanding the problem solving processes underlying the physical and mental activities that occur in the execution of organizational tasks. The office semantics model distinguishes between organizational and application knowledge. Organizational knowledge refers to an organization's social structure, while application knowledge refers to an organizations products and processes. An instructive epistemological emphasis of office semantics is that the way in which a problem is solved is highly dependent on how information/knowledge about the problem is presented to the problem solver. It follows that to present this knowledge effectively, one should understand how individuals think about problems, as well as what it means to solve a problem. A problem solving methodology known to be effective in the domain of medical applications is **Problem-Based Learning (PBL)**. This approach was introduced in the context of software engineering by McCracken and Waters (1999). They emphasized self-directed instruction in problem solving skills using large, not well-defined problems, with the objective of promoting an understanding not only of product-related issues but more especially of process-related principles.

Each of these problem solving models requires a decision about which one of a set of alternative solutions should be implemented. Such decision processes were studied by Mintzberg (1979) who identified the following typical stages:

1. Recognition: Recognize the need to initiate a decision process.
2. Diagnosis: Assess the situation.
3. Search: Find ready-made solutions.
4. Design: Develop custom-made solutions.
5. Screen: Evaluate the ready-made solutions.
6. Evaluation-choice: Choose a solution.
7. Authorization: If the problem solving occurs in an organizational context, then obtain approval for the decision, which may be from individuals who not explicitly involved in the problem-solving process itself.

Though decision-making is closely related to problem solving, distinctions must be made. Problem solving is a process which advances analytically from a current problem state to a desired goal state, while decision-making tends to emphasize a more synthetic approach whereby a desired goal is reached by a selection process that chooses from one of a set of possible, and perhaps pre-existing, alternative solutions (Huitt, 1992). One should also observe that there is a difference between problem-solving-like decision-making steps, and the process whereby those steps are achieved (Finnegan & O'Mahony, 1996).

2.2 Collaborative Problem Solving Models

A group that develops a plan for designing a system that will solve an existing problem is by definition engaging in collaborative problem solving. Collaborative groups appear able to deal with complex tasks more effectively than individuals, partly because groups automatically have a broader range of skills and abilities than individuals (Finnegan & O'Mahony, 1996). Despite this, studies indicate that group problem solving is intrinsically more complex than individual problem solving (XXX Finnegan & O'Mahony, 1996). It can introduce difficulties that are specifically group-related, such as an interaction environment that inhibits the free expression of ideas (Hoffman, 1965), participation biases, conflicts caused by interpersonal difficulties, or complications arising from the structure of the group. Overall, however, the benefits of collaboration in problem solving far outweigh its disadvantages (Hohmann, 1997). For example, one notable benefit is the ancillary improvement of human capital effected by collaboration, because the individuals involved in a group learn from the skills and abilities of the other group members (Prey, 1996). The need to articulate designs, critiques, and arguments to other group members also hones an individual's technical, critical, and interpersonal skills (Guzdial et.al, 1996).

A collaborative problem solving *model* is an explicit methodology used to facilitate collaborative problem solving. Such a model when complete will include generic problem-solving steps, domain-specific tasks, and requisite cognitive skills, but also the communication and coordination activities required by a collaborative environment. The collaborative problem solving method may be similar to an individual problem solving method. Indeed, in his important work on group software development, Hohmann (1997) observes that collaborative problem solving can be done using the very same problem solving methods that are used by individuals. Hohmann claims that while it is important for a group to explicitly choose and follow a problem solving method, and while group members should be familiar with the selected method, nonetheless, the method itself does not need to be designed specifically for group problem solving. Despite this laissez-faire approach to the chosen problem-solving method, Hohmann observes that the way in which a team will appropriate such a method in a collaborative environment, will differ substantially different from the way in which an individual will apply the same method.

The main distinguishing characteristic that differentiates group from individual problem solving in Hohmann's model is the decisive role of communications in group problem solving. His collaborative model identifies several group-oriented processes including, distributing or assigning of tasks to individuals, coordinating team outcomes, and integrating solution components by subgroups. Partitioning tasks and subsequently coordinating these subdivided activities determines the communication requirements of the group. Various kinds of required communication processes can be distinguished, such as, each individual's self-understanding of the problem to be solved must be shared with other group members.

Hohmann emphasizes that the collaborative model should account for the possibly dynamic nature of group membership and its impact on communications, because group communications are affected every time the group membership changes. There are standard mechanisms for coordination and control, defined by the Software Institute Systems Engineering Capability Maturity Model (Hohmann, 1997), that support software development teams including: configuration management, outcome reviews, status monitoring, and in the case of more extended models of collaboration: customer interaction, method management, cost management and release packaging The following activities required by these mechanisms are worth noting in some detail because their social aspects are relevant to collaborative development (Hohmann, 1997):

1. Configuration Management: The process whereby outcomes: data models, requirements documents, etc are identified and agreed upon by the group members.
2. Outcome Reviews: Controlled reviews of the outcomes identified in configuration management, by a subgroup of the collaborative group, possibly supplemented by extra-group members. This is a critical component of the development feedback loop, and typically entails additional outcomes such as documentation of any required changes.
3. Status Monitoring: An ongoing step involving both the collaborative group and external management to ensure the project is on schedule and to determine actions in case of schedule slippage.
4. Customer Interaction: Communication and feedback between customer and developer is initiated.
5. Management of the Use of the Method: Adapting a pre-defined, off-the-shelf methodology to the problem at hand.

6. Managing Development Costs: Cost management is not only a management function, but collaboratively involves all group members, each of whom can contribute their own individual expertise to cost management.
7. Release Packaging: This entails collaboration between multiple groups. It involves the activities required on system completion in order to bring a product to the customer, and is required to ensure customer satisfaction.

An important aspect of group communications is how individuals interact in a group. The properties of a group that affect such communications and the impact of such interactions on group effectiveness can be examined using a cognitively-based analysis. Zhang (1998) describes such a framework for group problem solving that emphasizes how tasks are distributed across the individual group members, and interprets or views the group's understanding of a problem as a distributed cognitive representation system. The cognitive process of problem decomposition, for example, can be considered not only in terms of its relation to the problem itself, but also explicitly considered in relation to how a problem can best be partitioned with respect to the ultimate assignment of the partitioned tasks to individuals in the group. Further research that emphasizes the implications of coordination for group communication is (Kies, Williges, & Rosson, 1998). Similarly, (Ellis, Gibbs & Rein, 1991) underscore that the communication and collaboration of a group is enhanced if its activities are effectively coordinated.

Group problem solving can be interpreted in terms of cooperative decision making, a classic model of which was adumbrated by Simon (1997) requiring:

1. Plan Development: A plan of behavior for the group is developed for all the members of a group, not merely a set of individual plans for each member.

2. Plan Communication: This plan is effectively communicated to each group member.
3. Behavior Modification: Each individual group member must commit to the plan in the sense of agreeing to permit their behavior to be guided by the plan.

The objective of this kind of contractually agreed upon group plan is to relieve individual members of the group of the task from uncertainties in predicting the behavior of other members.

A logical/qualitative method for facilitating joint decision making and alleviating conflict resolution was developed in Wong (1994). The approach is applicable to the kind of cooperation required of software engineers on a development project. Wong's model has three stages: identification, processing, and negotiation. The identification stage entails first identifying a decision agenda using priority-ordered criteria, then identifying the agents concerned with each criterion, where the term agent refers to the person or system responsible for a problem solving step. Competing alternatives are identified and the relationships among the alternatives are determined. The processing stage develops a set of so-called preference expressions for each criterion in the decision agenda. These preference expressions are merely ordering relations for pairs of alternatives. The alternatives are then rank ordered to determine a recommended solution. A final negotiation stage then follows where the agents negotiate conflicts.

A model of group problem solving formulated by empirically observing group decision making behavior in environments which lay outside scientific/engineering/software development contexts was developed by Finnegan and

O'Mahony (1996). This behaviorally-based model empirically recognized the same kind of problem-solving processes that have been systematically and explicitly articulated for engineering contexts. Groups progressed from an initial problem realization to a solution choice by a process dominated by communication of information and group collaboration, and needed significant levels of coordination and control throughout the decision making process. The initial, problem realization stage was typically initiated by a specialized group or by organization management. The next stage, planning, required coordination of subgroups. A subsequent information search stage was followed by group discussion of the information discovered about the problem. Subsequently, alternatives were identified and evaluated, and a preferred alternative selected, followed by validating, marketing or selling of the alternative to other groups, and ultimately implementation of the selected solution. The process is iterative, adapting to new requirements as they arise, reminiscent of user-centered software design in which a design is tested and redesigned through multiple iterations (Kies, Williges & Rosson, 1998).

Collaborative problem solving has been addressed because it is at the very foundation of collaborative software development: when developing software an individual is designing a solution to a problem. Collaborative problem solving models are at an initial stage of development. Researchers have developed various models but their testing and implementation has not been extensive. Their utilization at both the industrial and the education level has been limited. Table 2.1 highlights the collaborative models considered, all of which had the same objective–solving a problem or making a decision collaboratively.

Simon (1997) and Hohmann's (1997) models will be used as points of reference. Simon's (1960) influence in the field of problem solving has been seminal (Deek, 1997; Hohmann, 1997). Though Simon considered only collaborative decision making, the similarities with collaborative problem solving (Huitt, 1992) makes his collaborative decision making model an important point of reference for the collaborative problem solving models that have been described. Hohmann's (1997) model, on the other hand, is a useful point of reference because it is relatively comprehensive and closely related to some of Simon's most influential work. A composite list of attributes from Hohmann's (1997) and Simon's (1960) models follows.

- Identification of tasks. The objective of this task is to correctly identify the components of the problem, which can happen only if the problem solution strategy is fully understood. This step also helps confirm the group's understanding of the problem. Hohmann (1997) indicates that the individuals who comprise the team are best prepared to accomplish this task, as opposed to an external agent.
- Distribution of tasks. The components of the problem solution should be distributed among the individual group members. This can be done by election, elimination or direct assignment (Hohmann, 1997). If group members are aware of each other's skill, distribution by direct assignment may be straightforward. If election is used, it is important for each group member to be explicit about the component they would like to execute to ensure successful implementation. Assignment by elimination occurs when there is only a single component left and no one yet assigned to the component.

- Coordinating Outcomes. Coordination is an on going task in which each member of the group should participate. Groups have been found to need a great deal of control and coordination to enable members to collaborate effectively (Finnegan & O'Mahony, 1996). Such coordination optimizes the likelihood that the group will work harmoniously towards the goal. Many of the tasks of problem solving need to be coordinated: distributing tasks, integration of the sub problems, design discussions etc so coordination is essential for successful development (Hohmann, 1997).
- Integrating Solutions. Since the components identified in the solution planning stage need to be integrated, an integration plan needs to be developed, beginning with the order of the solution integration.
- Plan Development. The development plan of behavior is an integrated plan for all the members of the group, not just a set of individual plans for each member (Simon, 1997).
- Communication Plan. The development plan needs to be communicated to each member (Simon, 1997).
- Behavior Modification. Individual members must be willing to allow their behavior to be guided by the plan (Simon, 1997).

Table 2.1 Summary of Collaborative Problem Solving (PS) and Decision Making (DM) Models

Model Name or Author	Id Tasks	Distribution of Tasks	Coordinating Outcomes	Integrating Solutions	Plan Development	Communication Plan	Behavior Modification
Hohmann (PS)	All group members contribute.	Election, elimination and direct assignment.	Implements status ongoing monitoring and formal reviews.	Develop integration plan	Identify and distribute tasks.	Make sure entire group understands problem.	Group should develop and follow norms.
Simon (DM)					Develop plan for entire group.	Make sure entire group understands problem.	Group should develop and follow norms.
Finnegan and OMahony (DM)			Planning Stage. Coordination plan developed.		Planning Stage. Process completed by a special group.	Established in Planning stage.	
Wong (DM)	Id important criteria.		Conflict resolution (negotiation) scheme.		Id competing alternatives	Conflict resolution	
Zhang (PS)	Explicit decomposition of tasks.	Decomposition into individual representations.				Emphasis on the interactions among individual representations.	
PBL (PS)			Meet with group facilitator once a week.				

2.3 General Groupware Theory

Groupware refers to software systems intended to support group interaction and collaborative teamwork. These systems range from electronic meeting rooms to workflow systems, such as a strategic information system used in project planning and implementation. As previously indicated, the globalization of business implies that team members are often geographically dispersed, and a dramatic growth in telecommuting and off-site consulting has accelerated the need for dispersed meetings (Nunamaker, 1999). Groupware is an enabling technology for such environments and reflects a fundamental change in emphasis from using computers to solve problems to using computers to facilitate human interaction (Ellis, Gibbs & Rein, 1991). The principle functions of groupware are information sharing, document authoring, messaging, computer conferencing, group calendars, project management, and support for team building (Zwass, 1998). Team building and project management tools provide coordination support and facilitate collaboration and communication by information sharing/document authoring and messaging/conferencing tools. A simple diagrammatic view is shown in Figure 2.1.

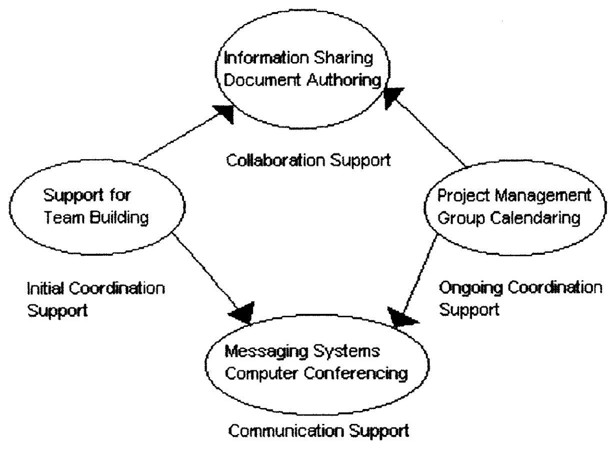


Figure 2.1 Workgroup Support Offered by Groupware.

Groupware systems can be characterized as synchronous, asynchronous, or a combination of both. Synchronous groupware systems run in real time and support group communication and collaboration using such techniques as instant messaging. An example is an electronic meeting system used for brainstorming. In *asynchronous* systems, users access stored messages or send messages to be viewed at a later time, such as e-mail. An example of a system with both asynchronous and synchronous features is a system with a message board and a chat feature. An analysis of how synchronous and asynchronous approaches effect communication behavior differently was done by Hiltz and Turoff (1985). For example, in asynchronous systems, communications tend to be lengthy with multiple, simultaneous discussion threads, in contrast to synchronous systems where participants tend to focus on a single topic at a time. A different kind of distinction between synchronous and asynchronous systems is given by Huang and Mak (2001) who differentiate between systems not just on the basis of the temporal characteristics of communications, but on how tasks and information are shared. They define asynchronous systems as groupware systems where individually allocated tasks and decisions are done separately and not shared until they are completed. In contrast, synchronous systems provide a completely shared workspace, continually accessible to all users, where work products are built and critiqued in a shared manner, with minimal work separation, and subsequently integrated via joint team decisions.

A model of groupware that has frequently been used to represent both synchronous and asynchronous systems is the Computer Supported Cooperative Work (CSCW) framework, developed by Dix and Beale (1996). The model distinguishes between participants in the collaborative process and the collaborative work artifacts, and explicitly emphasizes the need to develop a joint understanding of the problem by the participants. A well-known type of groupware is the Group Decision Support System (GDSS) which is used to facilitate such group processes as brainstorming, reaching consensus by voting, surveying experts, and negotiation: where parties resolve conflicting interests by communication (Zwass, 1998). For example, brainstorming in a GDSS entails a group of networked participants addressing a problem, with participants generating and posting their ideas synchronously, then voting on the ideas using the system in real-time. This not only saves time because of the parallel processing it allows, but also permits more ideas to be presented then in a traditional face-to-face meeting. Certain affective and behavioral side effects are also minimized. For example, individuals cannot talk over one another and since self-consciousness is less of an issue in such an environment, individuals are more inclined to present ideas. Thus, GDSS support organized human parallel processing, allow broader input, and promote more representative participation and discussion than in a typical face-to-face environment (Zwass, 1998).

An interesting general framework for collaborative systems was considered in Huang and Mak (2001) who distinguished between Computer Supported Collaborative Work (CSCW) as opposed to workflow management. In CSCW all interactions are between human members. In workflow management, on the other hand, the process agents are either human or software agents. Architecture for such a system of communicating agents in a cooperative planning environment is described in Fuliang and Wu (1999) and includes software agents dedicated to such tasks as addressing domain-level conflict recognition and resolution.

2.3.1 Organizational Factors in Collaborative Environments

The organizational, motivational, political and economic factors that are central to group activity are rarely explicitly addressed in the design of collaborative systems. Grudin (1994) identifies eight impediments to the development and use of groupware systems. These difficulties and possible resolutions follow:

1. *Perceived disparity*: The perceived disparity is between the effort required in collaborative environments versus the benefits that are perceived to accrue from their use.
2. *Critical mass problem*: A collaborative tool may not be used because it does not appear to be to the advantage of any single individual in the prospective collaborative group.

3. *Disruption of social processes*: Collaborative environments tend to level the playing field, violating in-place social hierarchies.
4. *Exception handling*: Groupware systems may not be flexible enough to accommodate the exception handling and improvisation required by most group activity.
5. *Unobtrusive accessibility*: The most frequently used features should be readily accessible, not hidden by being integrated with less frequently used features.
6. *Difficulty of evaluation*: Groupware is more difficult to evaluate then systems used by individuals because they are not affected by the backgrounds or personalities of other group members. Lab situations cannot reliably capture complex but important social, motivational, economic, and political dynamics of groups.
7. *Failure of intuition*: Systems are intuitively developed based on the needs of a subset of the users or based on experience from developing single user applications. Developers fail to recognize that groupware applications require participation from a range of users.
8. *Adoption or organizational integration process*: As with any system, *organizational integration* is critical, and careful efforts must be made to ensure groupware is accepted on an organizational wide basis.

All of these represent serious challenges to the successful design of groupware. The perceived disparity and critical mass problems can be addressed by educating potential users to the advantages of the systems. Equally critical is enhancing *self-efficacy*, defined in cognitive theory as the individuals' belief that they are capable of effectively using such technologies (Compeau, Higgins, & Huff, 1999). Nidamarthi et al. (2001) make related observations by underscoring that the role of collaborative environments should be to supplement rather than replace existing methods for communication, which is related to the requirement that collaborative environments should not add an undo technological burden on team members. Nidamarthi et al. (2001) additionally emphasize that the media used for collaborative communication should not "destabilize" existing effective traditional means of communication, such as simple pencil and paper calculations, a criterion which has important implications for the specification of the technological implementation of collaborative systems. Regarding social process disruption, it is worth observing that computerized collaborative environments tend to reflect implicit, built-in design assumptions which may well conflict with existing roles or responsibilities defined by an organization (Siemieniuch, Sinclair, & Vaughan, 1999). Since groups dynamically adapt to collaborative environments, the collaborative framework and environment must also be flexible enough to allow groups to develop their own norms for interaction (Majchrzak, Rice, Malhotra, King, & Ba, 2000). Flexible exception handling is especially relevant to the present thesis because improvisation and group cognition go hand and hand. Cognition is the process behind knowledge creation. Since cognitive processes, problem-solving methodologies notwithstanding, are often unstructured and spontaneous, a group-cognitive-model/groupware environment must balance the need to structure group activities with the need to support improvisation.

Cognitive effects of collaboration in asynchronous environments were examined by Dufner et. al. (1999) who observed positive effects on participants of having time to reflect on problems during asynchronous communication. Furthermore, more alternatives could be identified and explored because asynchronous meetings took place over extended periods of time. This research stressed the importance of coordination to keep group members focused on the task at hand. To facilitate such coordination, group process structures and methods should be included in groupware tools. Even simple voting or meeting agenda tools assist in facilitating coordination among group members during the decision making process (Dufner, 1994).

Distributed learning is another benefit observed for collaborative environments (Tinzmann, 1990), including knowledge sharing between experts and novices and peer-oriented knowledge sharing among novices. Collaboration also facilitates heterogeneous grouping, enabling weaker participants to learn from the more experienced. Nonetheless, it remains essential to have a mediating agent for intervention, such as when a group seemed blocked or misdirected.

Fundamental requirements for collaborative systems were identified by.Hahn, Jarke, and Rose (1990). In particular, it is important to recognize group development as an organized social process consisting of interactions between group members. The system support for group interactions must accommodate customary collaborative techniques such as negotiation, commitment, and responsibility contracts. The system must reflect the social protocols that underlie group communication in terms of strategies and policies for argument exchange, contract assignment, decision making, etc. The system should also provide tools that support the domain knowledge of the underlying project, such as, in the case of software engineering languages for specification, design, implementation, and date modeling. It has been observed that one of the benefits of collaboration is the learning that occurs during the process. Learning processes are typically socially distributed, extending beyond individual cognition to include features both of the social environment and the domain (Jarvela, 1999).

2.3.2 Groupware Theory for Software Development

Groupware reflects a change in emphasis from using the computer to solve problems to using the computer to facilitate human interaction (Ellis, Gibbs & Rein, 1991). It is a relatively new technology that is steadily earning a prominent place in today's workforce. Seminal early work by Turoff (1984) documented that groupware systems such as the pioneering EIES system were very useful in enabling software developers to manage development in environments characterized by frequently changing design features. Turoff observed that such systems allowed developers to keep one another informed about what did or did not work, and were particularly beneficial at the problem solving stage of software development. *Software development groupware* refers to any system that allows a group of software developers to design software, collaboratively, in the same workspace. The increasing complexity of software applications makes such systems a practical necessity.

A key application of groupware technology is supporting the decentralization of software development tools (Hahn, Jarke, & Rose, 1990), though available software development groupware often pays rather more much attention to syntactical than to critical semantic issues, such as documenting a change in a specification or software but not documenting the reason for the change. In order to make groupware effective, issues of both collaboration and software development should be addressed. For example, software development groupware should integrate the submodels of software development (requirement analysis, work package planning, programming, etc.) under a composite formal model of software project management to ensure that transitions between submodels are under formal control. Of course, groupware is not only beneficial to geographically distributed workers, but also to local developers engaged in team problem solving, because groupware can assist decision making, such as through voting, keep track of software requirements, and provide supplemental means for effective communication.

The information needs of developers can be analyzed in an a priori manner, or by empirically observing the actual observed information requirements of practicing development teams. Herbsleb and Kuwana (1998) performed just such an empirical study to determine the kind of information software development teams required by videotaping development team meetings and analyzing the resulting meeting minutes. The types of questions the developers asked were used by the experimenters to identify and categorize the team's information needs. Five specific areas where developers needed assistance emerged (Herbsleb & Kuwana, 1998):

1. Understanding the problem domain: General methods for understanding problem domains ranged from domain analysis and modeling, to mutual or

self-guided instruction on the domain, to observing how other more experienced developers viewed the domain. This kind of domain understanding is a persistent issue at every development stage from requirements analysis to design.

2. Exploring detailed design: Computer-Supported Meeting Environments (CSME) are needed to effectively support collaborative detailed design, and CASE tools are required to support representation of the design.
3. Tracking: Systems should support design traceability, which refers to the ability to conveniently answer questions about specific functionalities.
4. Providing user scenarios: Typical user case scenarios are beneficial both at requirements definition and detailed design, and synchronous or asynchronous collaborative methods can be used to share such user case knowledge.
5. Functional definitions and interfaces: Systems should provide assistance in tracking and sharing functional definitions of modules and interfaces.

Though developers may perceive these as obvious requirements, it is nonetheless important to highlight their significance. For example, while (4) may seem trivial, it is germaine to a leading cause of runaway projects: misunderstanding changing software requirements. Runaway projects are defined by (Mahaney & Lederer, 1999) as ones that significantly exceed the original budget estimate while at the same time possibly providing significantly less than the originally intended functionality. They identify the leading cause of such projects as ineffective handling of dynamically changing requirements. Each of these five areas where support is needed can be aided by groupware systems that facilitate effective communications and information sharing.

Collaboration and management address common issues such as scheduling meetings, task allocation, and negotiation, and also share common difficulties such as interpersonal conflict management. Brereton et. al. (2000) used the similarity between collaborative and management activities in to analyze collaboration in distributed groups in university environments, where they focused on the effectiveness of different kinds of computer-mediated technologies in facilitating collaboration. The collaborative technologies they examined included audio, whiteboard, chat, and video. Whiteboard was used for synchronous annotation of shared documents. Audio, video and associated chat capabilities were used for general communication. Their evaluation found all these modalities to be useful.

2.4 Group Cognition

Fundamental to understanding software development is a consideration of the kind of thought processes developers use (Stacy & Macmillan, 1995). The cognitive processes required in individual problem solving and program development were extensively examined in (Deek, 1997) where they were formalized in the Dual Common Model presented there. Since the cognitive activities that occur in a group are even more varied and complex than those in individuals (Hohmann, 1997), it is still more essential to understand the role of such processes in group problem solving. Their complexity is compounded because one is faced not merely with individual cognitive activities, but with the interplay of cognitive activities among individuals.

In the context of software development, one particularly interesting cognitive effect is cognitive bias. Cognitive bias refers to the propensity of individuals to be consistent and predictable in their behavior with respect to the kind of *errors* they make. Such biases operate at both the individual and the group level.

Various techniques have been proposed to reduce cognitive biases (Stacy and Macmillian, 1995) including obvious improvements such as using empirical as opposed to intuitive analysis, as well as less obvious strategies such as systematically seeking what is called *dis-confirmatory information*, or systematically recasting guidelines as trade-offs. For example, intuitive approaches, which refer to immediate cognition not reinforced by an explicit process, leave greater room for error than method-based or empirical approaches. Seeking dis-confirmatory information refers to the practice of asking negatively phrased empirical questions such as "How will I know if the feature does not work?" or "How will I know if this is not the cause of the problem? Finally, recasting one-sided guidelines as two-sided trade-offs refers to the practice, in the context of software engineering, of interpreting or applying guidelines by always applying and evaluating tradeoffs. These techniques reduce cognitive bias by moderating the cognitive impact of previous experiences, which cognition tends to bring to mind first, even though the previous experience may be irrelevant or invalid in the current situation.

A theory of group cognition developed by Nosek (1998) views group cognition as a coordinated, distributed cognitive process, the objective of which is to create a shared, distributed understanding of a problem at a team level. Though the importance of this area of research is increasing, information technologies are frequently not designed with the requirements of such teams in mind and suffer accordingly. Nosek calls the cognitive actions and interchanges that occur during collaborative problem solving *group sensemaking*. Nosek's model identifies three conditions required to create this kind of knowledge in a problem solving group:

distributed knowledge, distributed cognition, and coordinated cognitive processing among group members. Proper coordination of cognitive processing allows the members of a group to have comparable knowledge of a problem area. This research represents an attempt to determine the role of these effects in group collaboration.

2.5 Group Sociology, Psychology and Dynamics

Social science models have emerged as an important tool in understanding the impact of computers in the work environment. Sociological and psychological factors are especially significant in collaborative software development because of the intense interactions among software development group members. Since computer systems affect the social conditions of work groups, it is appropriate to incorporate social science models and methods into the development practices of software developers (Anderson, 1991). Anderson's work is based on Bion's model for human behavior in groups (Grinberg, Sor, & de Bianchedi, 1977) and indicates that understanding the behavior and actions of groups requires an awareness of the relevant psychological models because of the insights that can derive from a psychodynamic perspective. Anderson emphasizes several implications of Bion's model related to social group issues that developers of groupware should pay attention to in order to produce effective groupware. These include: viewing groups as an interdependent collection of individuals and viewing the behavior of individual group members as manifestations of the group culture. He characterizes groups as operating in either *work-group* mode or *basic-assumption* mode. The basic-assumption mode undermines effective problem solving, while the work-group mode is cooperative and uses rational methods to approach its task.

An important but underutilized tool of social science that is useful when designing Group Support Systems (GSS) for meetings is *socio-emotional theory*. Socio-emotional theory examines the impact of factors like group conflict, or group/individual satisfaction on the outcome of meetings or processes. GSS research has generally focused on task outcomes, leaving socio-emotional outcomes largely unexplored (Kelly & Bostrom, 1995). Thus, the success or failure of groups has been analyzed based on the objective contents of their outcomes with little reference to the underlying emotions that constitute the driving force of individuals and groups. Socio-emotional theories emphasize that if a poor decision may be due to socio-emotional issues, such as a lack of motivation or commitment by the group.

The research areas just mentioned are significant because the type of work that systems and software engineers do is almost by definition dependent on the expertise of a large number of individuals. Quintas (1993) lists several reasons why the social aspects of systems engineering and software development must be addressed. First of all, software engineering is a labor-intensive activity, so communication processes and social interactions within the developer community are of critical importance. Understanding why social activity is so prevalent in software development will be useful in developing models that will help us better understand the intricate interactions between developers during collaboration efforts. Secondly, software development is a *bridging process* that links areas of specialized and diverse expertise from the domain of the IT professional to the domain of the customer or user. Finally, the development of IT systems is itself a social and historical phenomenon, so it is appropriate to understand the social processes involved in the development, application and diffusion of IT.

Adaptive Structuration Theory (AST) indicates that the outcomes of meetings is a reflection of the way in which groups appropriate and modify structures implicitly or explicitly present in the meeting process, and that this in turn is significantly affected by a meeting facilitator. Group problem solving has many features in common with meetings, whence the relevance of such research. An analysis of the impact of a meeting facilitator in terms of socio-emotional factors in a GSS environment was done by (Kelly and Bostrom, 1999). The factors considered were: fidelity to procedures, group attitude, and group level of either conflict or consensus. The presence of a meeting facilitator tends to assist in the success resolution of these processes. The role of a facilitator is to foster a positive environment by appropriately selecting and facilitating the use of a structure that matches the group's task. A related area of research is *human factors*, defined as the study of the relation between the work environment of individuals and human behavior, with the objective of designing tools and systems that enhance the productivity of individuals and increase their job satisfaction (Thomas, 1984). Classic human factors like ease of use should be incorporated in a group problem solving model. For example, enhancing satisfaction increases motivation and indirectly product quality.

Though group interaction during collaboration has a critical impact on the problem solving process, the initial *composition* of the research group is at least as important. On the one hand, research in social psychology suggests that the presence of incompatible *personality types* tends to complicate collaboration in a general managerial environment (Polack-Wahl, 1999), thus suggesting that collaboration can be inhibited in the presence of cultural and *gender diversity* during collaborative software design. The historical prevalence of males in engineering fields accentuates gender-specific issues, but cultural, national, and racial diversity may also be factors. The same authors underscore that early exposure to diversity in collaborative projects prepares software professionals for the work environment and ameliorates complications rising from the diversity of group composition. Of course, the underlying assumption is that a group is constructed of technically competent and complementary individuals who jointly have enough knowledge to accomplish the job. Groups should also be given the opportunity to provide feedback about the collaboration. The authors also underscore the psychological, not merely the operational benefits, of assigning each member of a group specific tasks. The point is that even though there is a global group responsibility for a project, nonetheless such individual assignments psychologically tend to minimize displays of ego, aside from there advantages in terms of individual specialization or the ability to partition the work-load.

2.6 Group or Team Structure

The formation of a software development team is as important as the problem solving methodology and collaborative technologies a team uses. A team lacking a proper profile of members is unlikely to be successful even using the best collaborative problem solving model. Shneiderman (1980) presented a well-known *taxonomy* of software development teams into *conventional*, *ego-less*, and *chief programmer* teams, defined as follows: a conventional team has a senior member who directs the remaining junior team members, the ego-less team emphasizes cooperation versus competition, and any work developed is considered as the property of the group rather than merely that of the individual developer, so that the success or failure of a project is viewed as a collaborative effort, and a chief programmer team is built around well-defined roles - like a surgical team where specific roles and responsibilities are defined from the outset: surgeon, nurse, anesthesiologist, etc.

Each type of team has its advantages and disadvantages, so choosing the type appropriate for a given project and the available pool of team members can have a decisive impact on the outcome. Factors like the skill and work ethic of members have to be weighed prior to team formation, with a proper balance important. For example, members may be self-motivated, task motivated, or interaction motivated. Having too many task-oriented individuals on a team may inhibit effective group communications (Sommerville, 1996), while having the right composition of personality types increases group *cohesiveness*. Sommerville identifies a variety of advantages of *group* cohesiveness. First, a group quality standard can be developed, and quality standards determined by the group are more likely to be followed then standards imposed upon the group. Second, cohesion allows team members to work closely together, thereby learning from one other. Third, members of a cohesive team become more familiar with one another's work, promoting continuity and consistency when a member leaves the group. There are also some disadvantages associated with cohesive groups, such as resistance to changes in leadership and *groupthink* (Sommerville, 1996). Thus, it is generally better to select a new leader from within the group because introducing an outside leader as a replacement for a current leader commonly decreases productivity. *Groupthink* (Janis, 1982) occurs when the desire by group members for unanimity overrides their need to evaluate alternatives objectively. Particularly under the pressure of deadlines, cohesive groups can exhibit cognitive biases that preclude selecting the most effective solution. The risk of groupthink can be minimized by creating meeting environments where members feel free to criticize decisions or by introducing a third party to evaluate group decisions.

This collaborative problem solving section of this chapter has examined a variety of areas that are important in collaborative software development: problem solving and decision making, both individual and collaborative, general groupware systems theory - and for software development, group cognition, group sociology, group psychology and group dynamics. The issue is critical because professional software developers are expected to be able to engage in software projects as member of groups, hence groupware should be available to facilitate this effort. A variety of groupware systems for brainstorming, decision making, as well as general communication have been successful, but systems that assist group software development are at an preliminary stage of development. The next chapter reviews some of these systems.

2.7 Groupware Systems

A number of environments have been developed that provide some of the basic elements of groupware functionality. This section will review some of these tools, systems, and environments for collaboration, collaborative problem solving, and collaborative software development. The discussion will begin with overviews of some typical groupware systems, and then proceed to a review of systems specifically designed for collaborative problem solving and/or software development. Finally, the focus will be a detailed discussion of the features of several important collaborative systems including Lotus Notes, Groove, and Rational Rose.

2.7.1 CyberCollaboratory

The CyberCollaboratory (Dufner et. al., 2002) is an asynchronous group support environment. The CyberCollaboratory contains the following tools to support collaborative work: GDSS, Chat, and a Group Discussion and Document Production.

The GDSS environment contains an Electronic Brainstorming tool, an Idea Organizer tool, and Voting Methods. A Facilitation tool is also available for the team member who has been designated as the group leader. In the Group Discussion and Document Production environment, the group leader will set up a category and then ask group members to read and reply to questions and upload their revisions or modifications for the document.

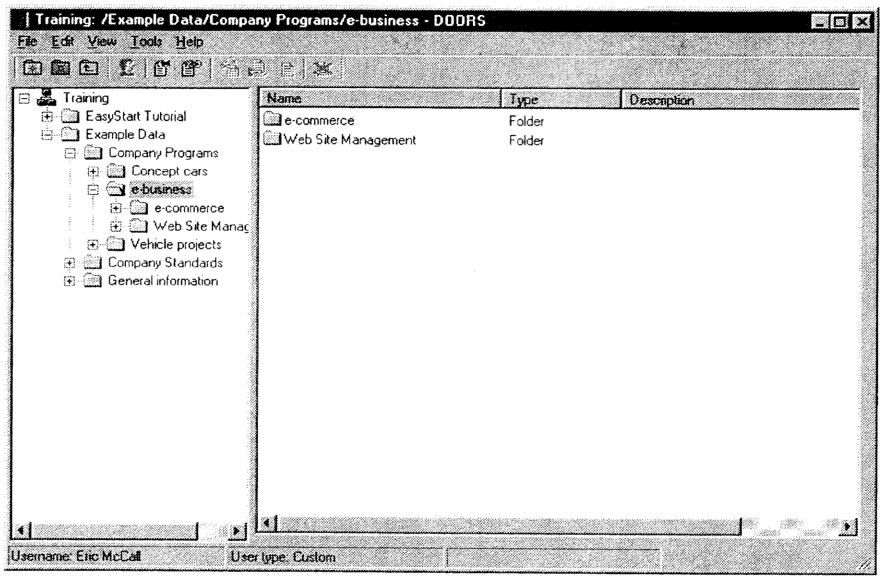
The CyberCollaboratory is a good tool to assist in some of the problem solving and programming tasks of software development. The main focus of the tool is decision support, which is a key task in problem solving but certainly not the only task.

2.7.2 Doors

Telelogic's Doors product is a requirements management system. Doors is able to capture, link, trace, analyze, and manage information to keep a project compliant with its specific requirements during its lifecycle. Doors has multiple tools that give the user multiples ways to access information. This feature benefits the needs of the different roles involved in developing software such as managers, developers, and end-users.

An Example Solution

Doors is a client/server application. Each user or client needs to log into the system located on a local server. Once the user has successfully logged in the main screen, shown in Figure 2.2, they have access to all of the projects.



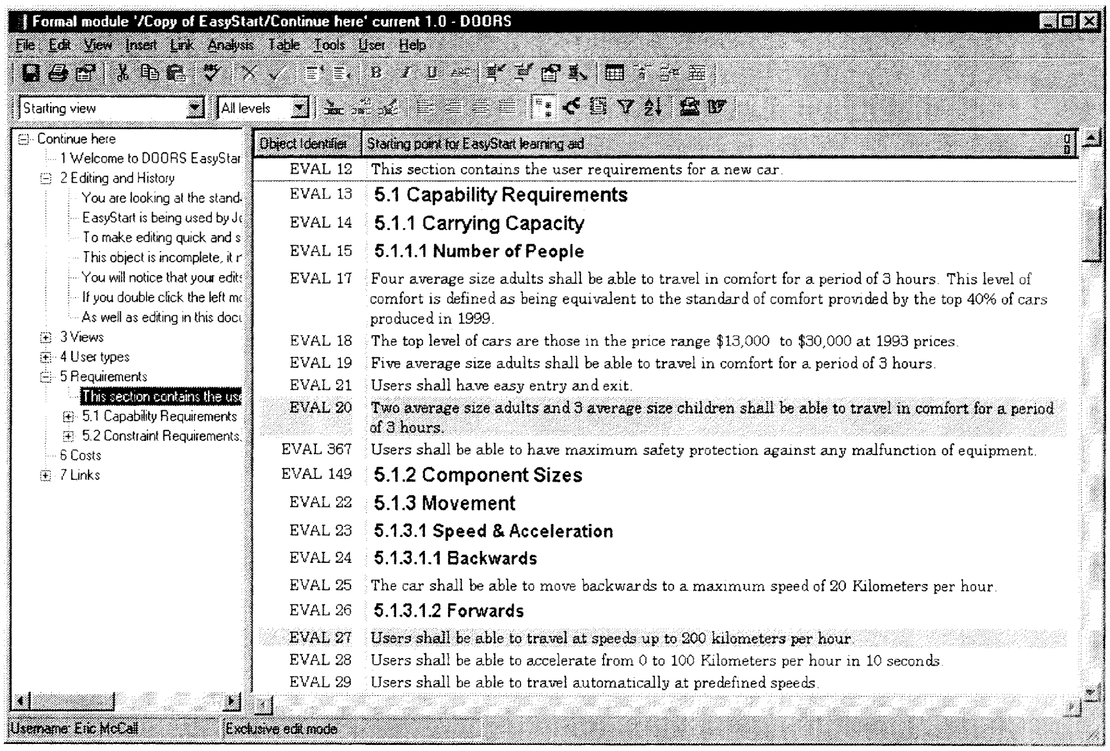
Name	Type	Description
e-commerce	Folder	
Web Site Management	Folder	

Username: Eric McCall

User type: Custom

Figure 2.2 Main Screen.

By opening up a project folder and clicking on one of the formal modules, a new window like the one shown in Figure 2.3 will open containing access to the selected project files.


Formal module '/Copy of EasyStart/Continue here' current 1.0 - DOORS
File Edit View Insert Link Analysis Table Tools User Help
Starting view
All levels
Username: Eric McCall
Exclusive edit mode

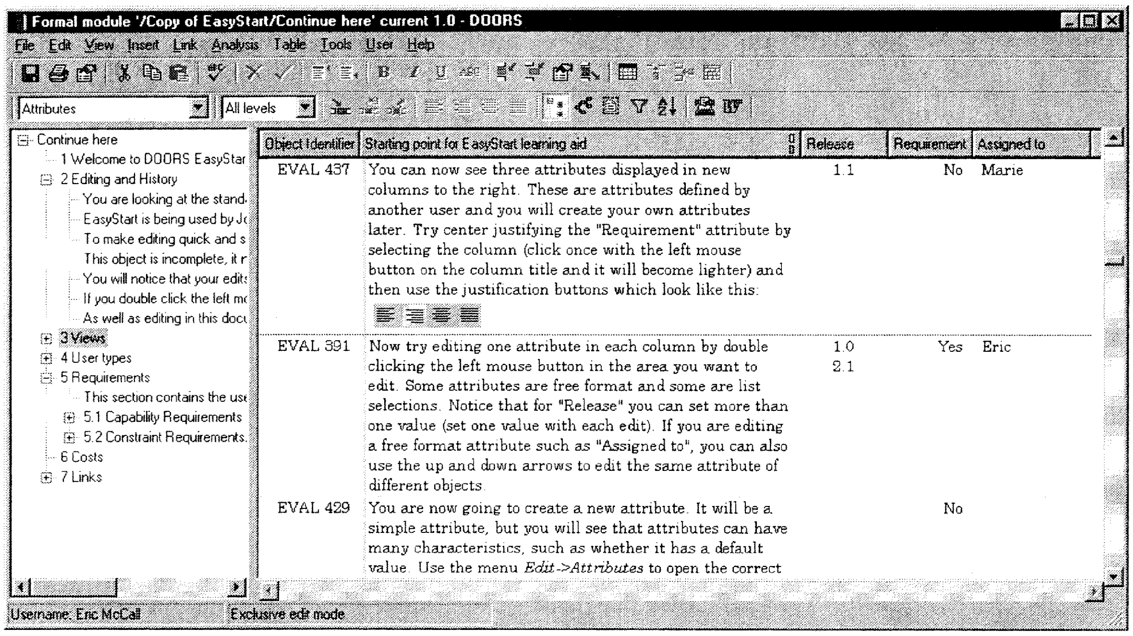
Continue here
- 1 Welcome to DOORS EasyStar
- 2 Editing and History
  - You are looking at the stand
  - EasyStart is being used by J
  - To make editing quick and s
  - This object is incomplete, it r
  - You will notice that your edit
  - If you double click the left m
  - As well as editing in this doc
- 3 Views
- 4 User types
- 5 Requirements
  - This section contains the us
  - 5.1 Capability Requirements
  - 5.2 Constraint Requirements
- 6 Costs
- 7 Links

Object Identifier	Starting point for EasyStart learning aid
EVAL 12	This section contains the user requirements for a new car.
EVAL 13	**5.1 Capability Requirements**
EVAL 14	**5.1.1 Carrying Capacity**
EVAL 15	**5.1.1.1 Number of People**
EVAL 17	Four average size adults shall be able to travel in comfort for a period of 3 hours. This level of comfort is defined as being equivalent to the standard of comfort provided by the top 40% of cars produced in 1999.
EVAL 18	The top level of cars are those in the price range $13,000 to $30,000 at 1993 prices.
EVAL 19	Five average size adults shall be able to travel in comfort for a period of 3 hours.
EVAL 21	Users shall have easy entry and exit.
EVAL 20	Two average size adults and 3 average size children shall be able to travel in comfort for a period of 3 hours.
EVAL 367	Users shall be able to have maximum safety protection against any malfunction of equipment.
EVAL 149	**5.1.2 Component Sizes**
EVAL 22	**5.1.3 Movement**
EVAL 23	**5.1.3.1 Speed & Acceleration**
EVAL 24	**5.1.3.1.1 Backwards**
EVAL 25	The car shall be able to move backwards to a maximum speed of 20 Kilometers per hour.
EVAL 26	**5.1.3.1.2 Forwards**
EVAL 27	Users shall be able to travel at speeds up to 200 kilometers per hour.
EVAL 28	Users shall be able to accelerate from 0 to 100 Kilometers per hour in 10 seconds.
EVAL 29	Users shall be able to travel automatically at predefined speeds.

Figure 2.3 Project Window.

In the formal module window, the user can create different views. A view is the way the information is displayed on the screen. Depending on the users needs, different views are more appropriate. Each project initially starts out with only the standard view, which only contains one column for identification of each object. After different views are created and saved to the project, they can be applied by choosing them from the views selector located on the far left of the second line of the toolbar also shown in Figure 2.3 above. The following discusses a few possible views.

In an *attribute view*, shown in Figure 2.4, three different attributes are displayed: *release*, *requirements*, and *assigned to*. The data of each attribute can be modified, and new attributes can be added to the view using the attribute choice under the edit menu. Once a new attribute is created, the user can create another view to display the new attribute.


Formal module '/Copy of EasyStart/Continue here' current 1.0 - DOORS
File Edit View Insert Link Analysis Table Tools User Help
Attributes
All levels
Username: Eric McCall
Exclusive edit mode

- Continue here
  - 1 Welcome to DOORS EasyStar
  - 2 Editing and History
    - You are looking at the stand
    - EasyStart is being used by J
    - To make editing quick and s
    - This object is incomplete, it r
    - You will notice that your edit
    - If you double click the left m
    - As well as editing in this doc
  - 3 Views
  - 4 User types
  - 5 Requirements
    - This section contains the us
    - 5.1 Capability Requirements
    - 5.2 Constraint Requirements
  - 6 Costs
  - 7 Links

Object Identifier	Starting point for EasyStart learning aid	Release	Requirement	Assigned to
EVAL 437	You can now see three attributes displayed in new columns to the right. These are attributes defined by another user and you will create your own attributes later. Try center justifying the "Requirement" attribute by selecting the column (click once with the left mouse button on the column title and it will become lighter) and then use the justification buttons which look like this:	1.1	No	Marie
EVAL 391	Now try editing one attribute in each column by double clicking the left mouse button in the area you want to edit. Some attributes are free format and some are list selections. Notice that for "Release" you can set more than one value (set one value with each edit). If you are editing a free format attribute such as "Assigned to", you can also use the up and down arrows to edit the same attribute of different objects.	1.0 2.1	Yes	Eric
EVAL 429	You are now going to create a new attribute. It will be a simple attribute, but you will see that attributes can have many characteristics, such as whether it has a default value. Use the menu *Edit->Attributes* to open the correct		No	

Figure 2.4 Example Attribute View.

Another example view is a *hierarchies view*. The objects can be sorted and be viewed graphically by switching to graphics mode; each object is displayed as a box that contains one attribute, as shown in Figure 2.5.

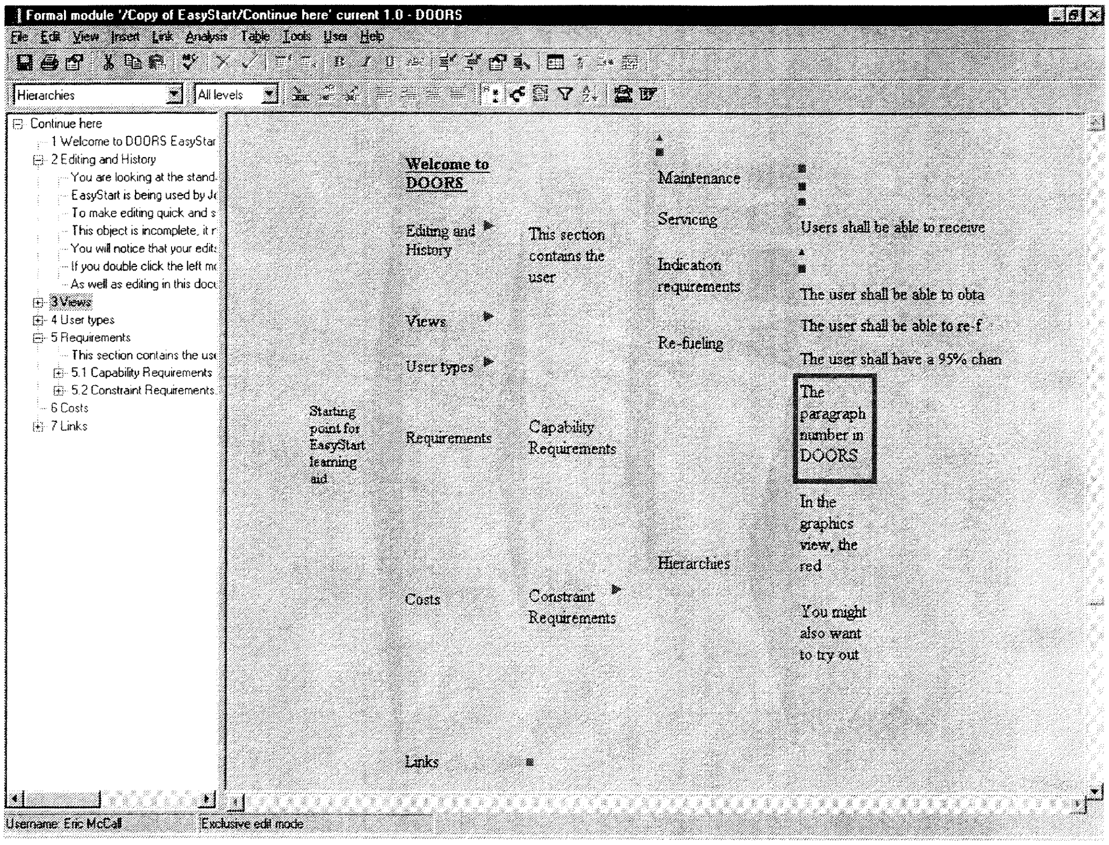

Formal module '/Copy of EasyStart/Continue here' current 1.0 - DOORS
File Edit View Insert Link Analysis Table Tools User Help
Hierarchies
All levels
Continue here
1 Welcome to DOORS EasyStar
2 Editing and History
You are looking at the stand
EasyStart is being used by Jo
To make editing quick and s
This object is incomplete, it r
You will notice that your edit
If you double click the left m
As well as editing in this doc
3 Views
4 User types
5 Requirements
This section contains the us
5.1 Capability Requirements
5.2 Constraint Requirements
6 Costs
7 Links
Starting point for EasyStart learning aid
Welcome to DOORS
Editing and History
Views
User types
Requirements
Costs
Links
This section contains the user
Capability Requirements
Constraint Requirements
Maintenance
Servicing
Indication requirements
Re-fueling
Hierarchies
Users shall be able to receive
The user shall be able to obta
The user shall be able to re-f
The user shall have a 95% chan
The paragraph number in DOORS
In the graphics view, the red
You might also want to try out
Username: Eric McCall
Exclusive edit mode

Figure 2.5 Example Hierarchies View.

The Doors product has three different edit modes for collaboration: *read-only*, *exclusive*, and *shareable*. In read-only the user can only look at the module, but not edit it. In exclusive mode one user can edit the module but all other users can only read it. In shareable mode while one user is editing one section of the module, another user can edit another section. In addition, the first user would have to lock the section of the module they want to edit to stop another user from editing it. The other users will still be able to read the data just not modify it.

Doors Technology Analysis

Doors is an asynchronous requirement management groupware product. Its main goal is to organize and communicate project information during and following a project's lifecycle. The collaborative feature is similar to other requirement management tools where only one user at a time can modify a project file.

A feature that stands out in Doors is that the data can be organized depending on the needs of the user. For example, the sales person doesn't need to see the same level of detail as a software engineer so a hierarchical view of the project can be displayed.

2.7.3 Groove

Groove (Udell, Asthagiri, & Tuvell, 2001) is a peer-to-peer groupware system that provides small groups of collaborators (2-25) the ability to share documents, messages, applications, and application-specific data related to group projects in a secure way. This system provides security via a virtual private network (VPN) and synchronization where offline changes are synchronized when users reconnect.

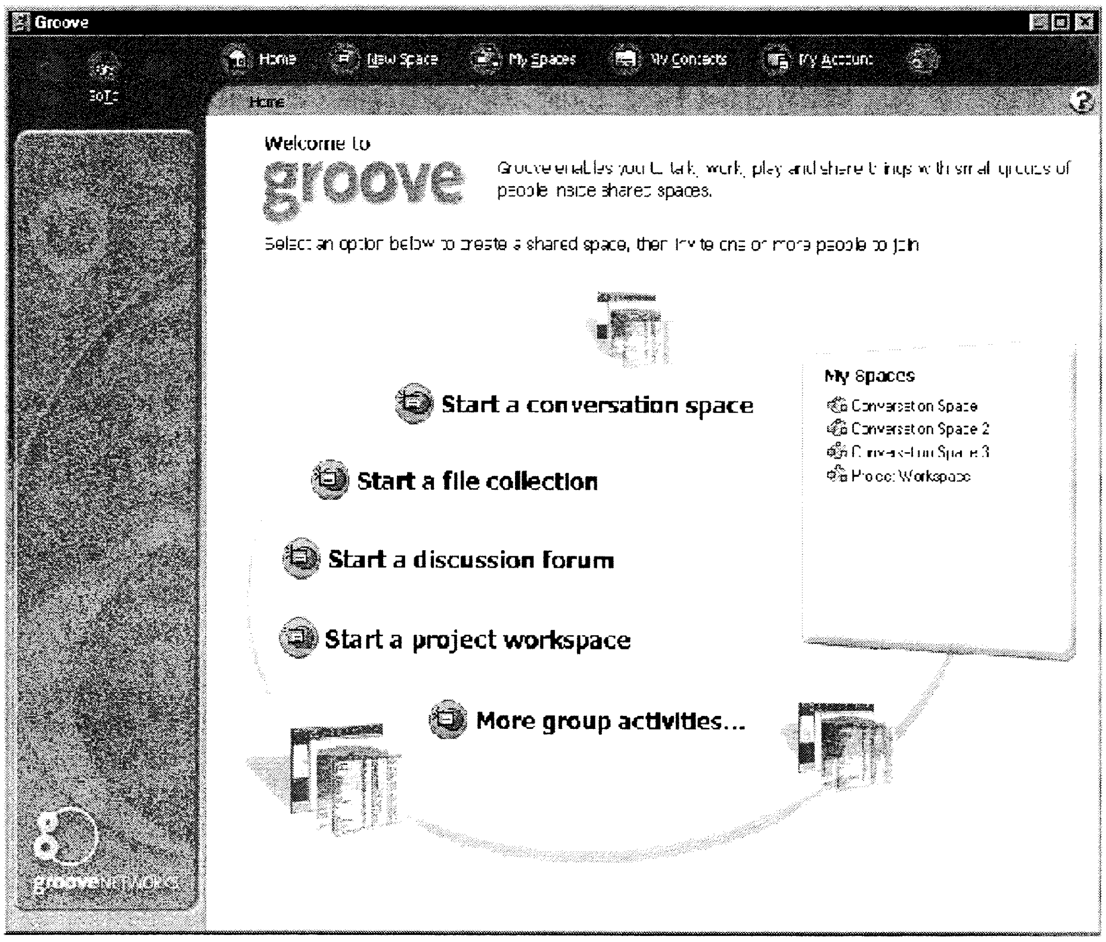
Groove provides strong tools for collaboration. The Groove application resides on each client's machine and the network is used as a pipe between the clients. It encompasses many activities such as live voice over the Internet, instant messaging, text based chat, file sharing (text, pictures, presentations), web browsing, drawing, 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.

The Challenge of Groove

In this brief example Groove will be used in the context of developing software. The basis of the analysis will be how well the system assists the attributes of Hohmann's (1997) and Simon's (1960) collaborative problem solving and decision making models. The analysis criteria are identification of tasks, distribution of tasks, coordinating outcomes, integrating solutions, plan development, communication plan, and behavior modification.

An Example Solution

Login enables Groove members to contact and collaborate with each other. Once a user is logged into Groove's start up window, shown in Figure 2.6, they can choose between five different types of workspaces to create or they can choose to return to an already created space from a previous work session. For example, when choosing to create a *conversation space,* the initial default tools include a note pad, a Web browser, a file manager, and a drawing pad. The user can also choose to return to one of their previous sessions. Figure 2.6 shows four sessions that were created previously to choose from.

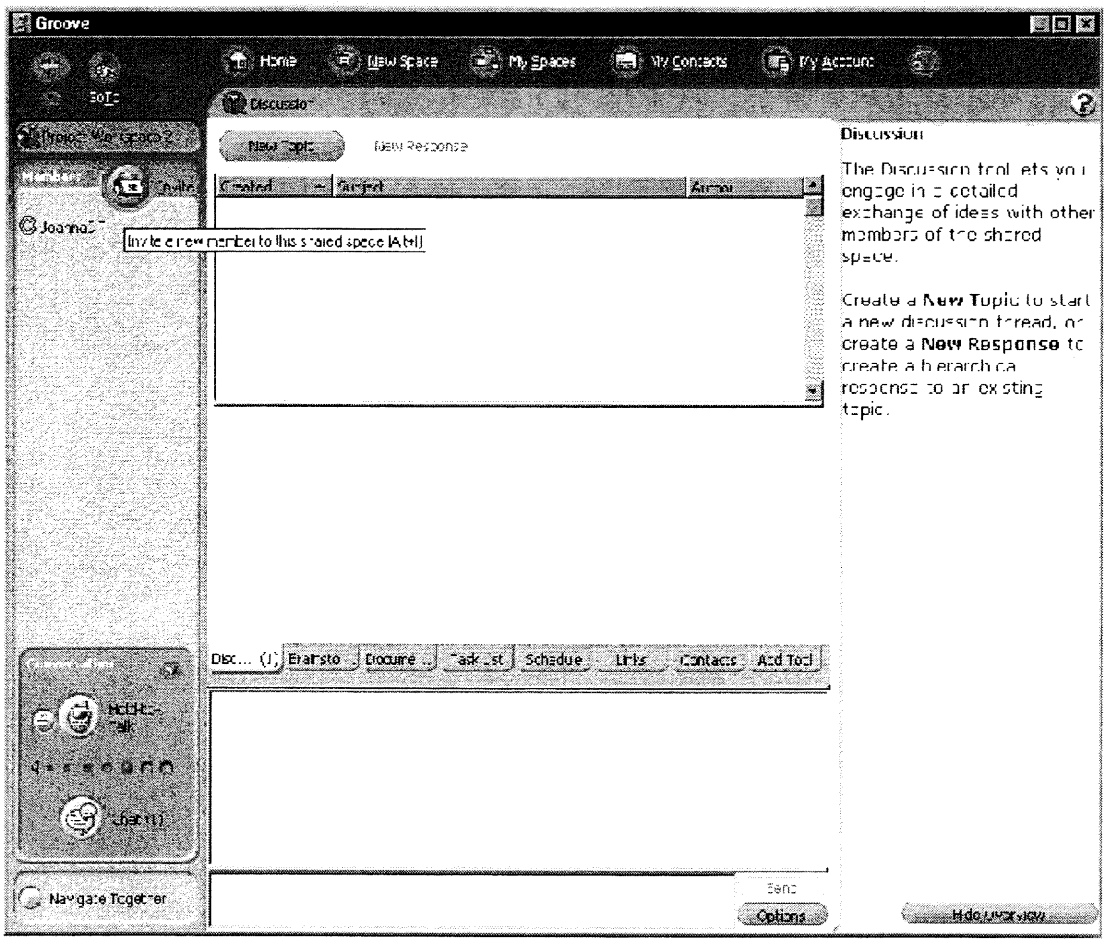


Groove enables you to talk, work, play and share things with small groups of people inside shared spaces.

Select an option below to create a shared space, then invite one or more people to join

Figure 2.6 Groove's Start-up Window.

This example uses Groove to assist the software development process. Therefore, a *project workspace* is created as shown in Figure 2.7. The following default tools are included in the initial space: a discussion board, brainstorming tool, document list, task list, schedule tool, Web browser, and contact list.



Discussion

The Discussion tool lets you engage in a detailed exchange of ideas with other members of the shared space.

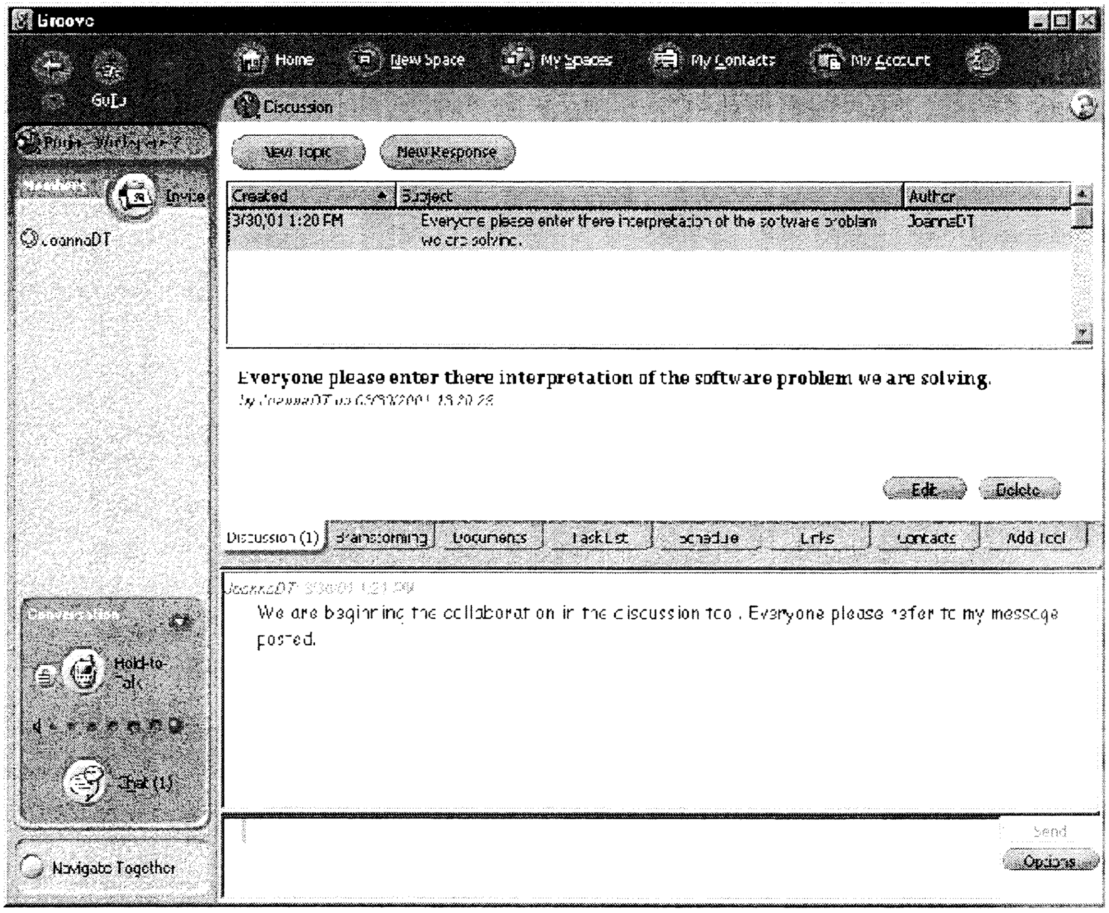
Create a **New Topic** to start a new discussion thread, or create a **New Response** to create a hierarchical response to an existing topic.

Figure 2.7 Initial project workspace window.

The *discussion tool* enables users to exchange detailed ideas with the other project members. The *brainstorming tool* allows the team to outline ideas in a hierarchical structure. The *document tool* allows users to share files. Any type of file can be transferred right form Windows Explorer into the document tool. The *Task tool* is a more organized version of the brainstorming tool. Ideas from the brainstorming tool can be organized with a framework created by using the task tool. The *Schedule tool* is where the group can store the goals and deadlines of the group. The *Links tool* allows the group to browse the web together. The last default tool in the project work space is the *Contacts tool* that enables the group to store the identity and characteristics of each member on the team for easy messaging access. Each of the tools described can be utilized by switching between various tabs. Also, additional tools can be added to the shared space buy choosing the *add tool* tab. For example, other tools that may be necessary when developing software are possibly the *sketchpad tool* or the *pictures tool*. After the project workspace is created, invitations are sent out to others to join the workspace either by instant messaging or e-mail by clicking on the red invite button also shown in Figure 2.7.

The shared link that connects to the workspace is actually sent and the user connects to it to accept an invitation. This is implemented by a combination of XML metadata describing what tools are in the space and the actual data that would be in the shared space at that point in time. Each workspace is stored locally. If a user is connected to the Internet, each of the shared space members sees each other's edits as they are made. If the user is not connected, they can still write or edit in any tool in the same shared space. When the user connects later, their edits get relayed and synchronized in the shared space so other members can see them; at the same time, the document they see gets updated with edits other members have made while they were working unconnected. The same synchronization occurs with the workspace tools. If one user changes the environment by adding a tool or a file the change is reflected in everyone's workspace. This will be either an instant change or a synchronized change when an off-line user reconnects with the group.

In this case study, the Groove is used to demonstrate the problem solving steps related to the software development process. To begin the project, the person who in leading the collaboration could use the *conversation tool* in the lower left hand corner of the screen, shown in Figure 2.8, to communicate which tool the group is using to collaborate in the beginning of the project. The conversation tool is similar to a walky-talky device. The person talking keeps the icon pressed until they are finished talking. Another option for synchronous communication is utilizing the chat feature. Figure 2.8 illustrates this in the bottom window to the right of the conversation tool. The chat message indicates that the group should be participating in the discussion tool where the group leader has posted a message to start the discussion.



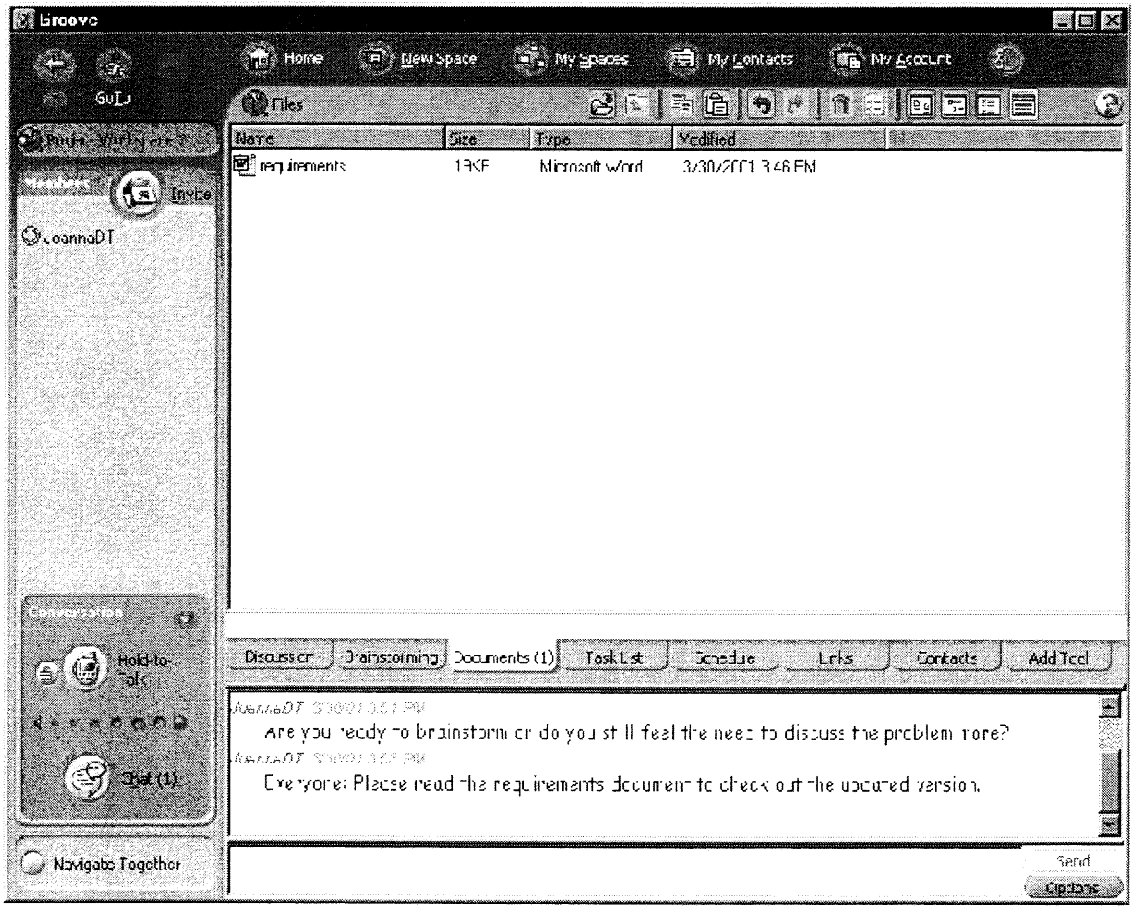
Created	Subject	Author
3/30/01 1:20 PM	Everyone please enter there interpretation of the software problem we are solving.	JoannaDT

Everyone please enter there interpretation of the software problem we are solving.
By JoannaDT on 03/30/2001 13:20:23

JoannaDT 3/30/01 1:21 PM
We are beginning the collaboration in the discussion tool. Everyone please refer to my message posted.

Figure 2.8 Example of discussion, conversation, and chat tools.

Figure 2.9 shows the documents tool. In this example, there is a single document called requirements.doc that can be accessed and modified by any member. Members should use this tool for sharing any kind of requirements or user documents developed during the course of development. Documents can be dragged right from Windows Explorer.

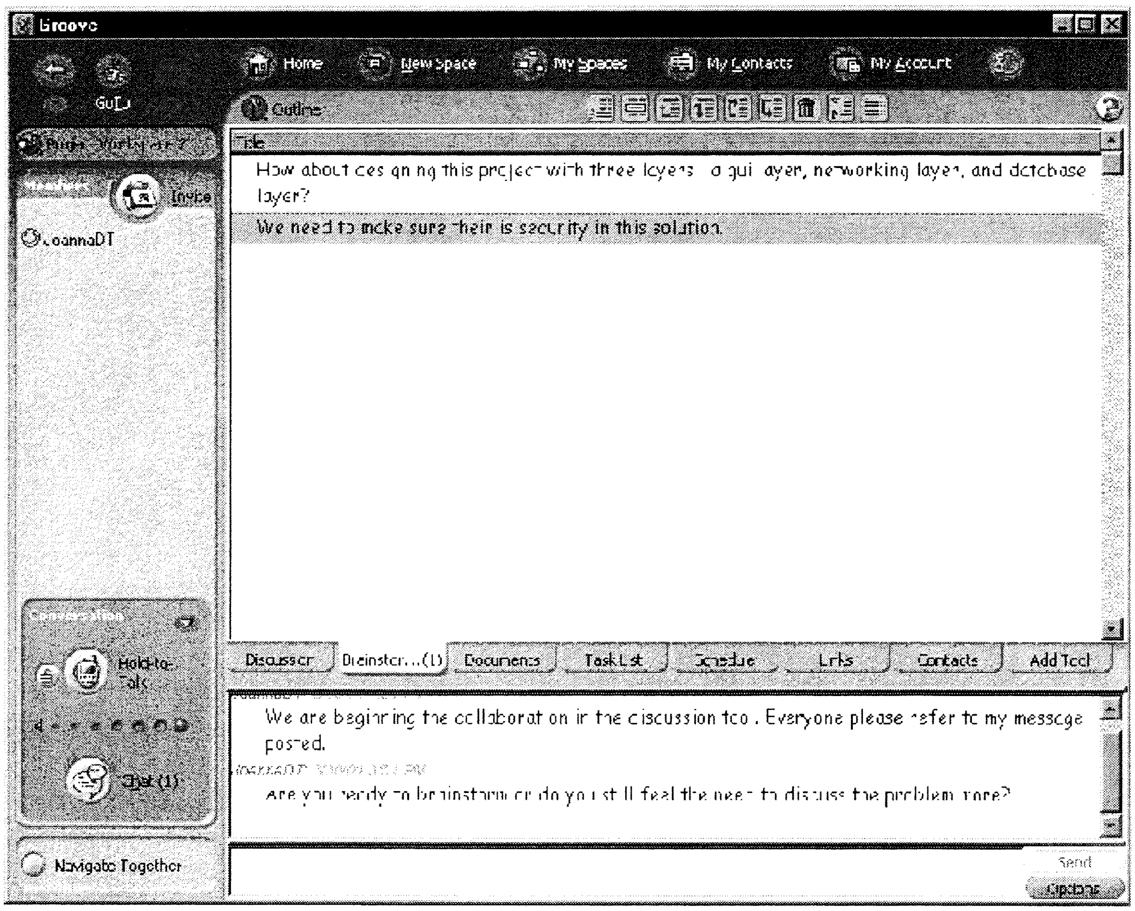


Name	Size	Type	Modified
requirements	13KB	Microsoft Word	3/30/2001 3:46 PM

JoannaDT
Are you ready to brainstorm or do you still feel the need to discuss the problem more?
JoannaDT
Everyone: Please read the requirements document to check out the updated version.

Figure 2.9 Documents Tool.

Once the discussion session is over and the group has read and understood the requirements document, the group needs to brainstorm for possible solutions to the problem using the brainstorming tool. By switching tabs to the brainstorming tool, shown in Figure 2.10, the group brainstorms some solution ideas.



How about designing this project with three layers a gui layer, networking layer, and database layer?

We need to make sure their is security in this solution.

We are beginning the collaboration in the discussion tool. Everyone please refer to my message posted.

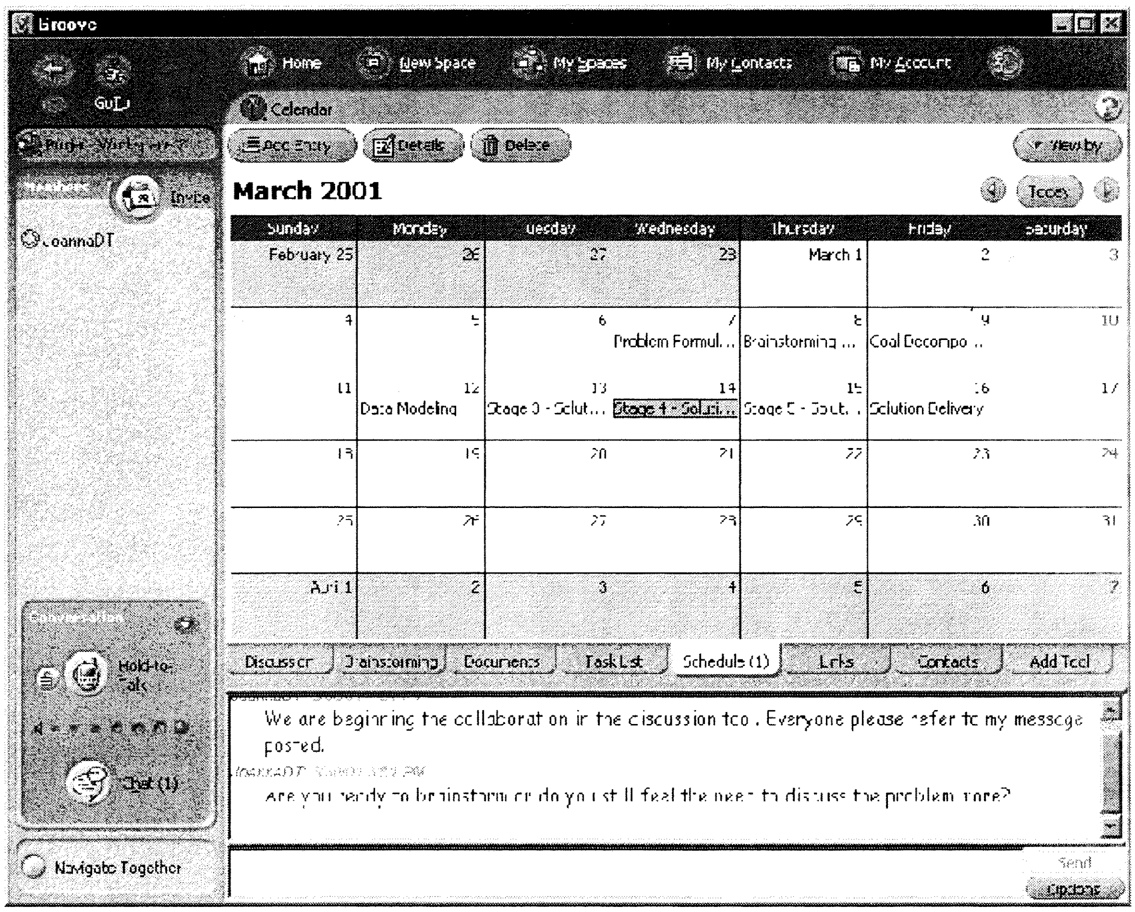
Are you ready to brainstorm or do you still feel the need to discuss the problem more?

Figure 2.10 Brainstorming Session.

For example, if the group needed information on Java to see if that is the language that would be the most effective and efficient for their development purposes, they could surf the Web using the Links tool.

Once the problem has been discussed and solutions have been brainstormed, the group needs to distribute the tasks to each of the members. The group members could either post their own task assignment or the group leader could post the assignments and have group members post a message indicating if they are not satisfied with their assignment.

To keep the group on track in meeting the goals of the project, a schedule is kept in the schedule tool shown in Figure 2.11. A monthly schedule is shown. There are options to view the schedule as either weekly, daily or as a work month.



March 2001

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
February 25	26	27	28	March 1	2	3
4	5	6	7 Problem Formul...	8 Brainstorming ...	9 Goal Decompo ..	10
11	12 Data Modeling	13 Stage 3 - Solut...	14 Stage 4 - Soluti...	15 Stage 5 - Solut...	16 Solution Delivery	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
April 1	2	3	4	5	6	7

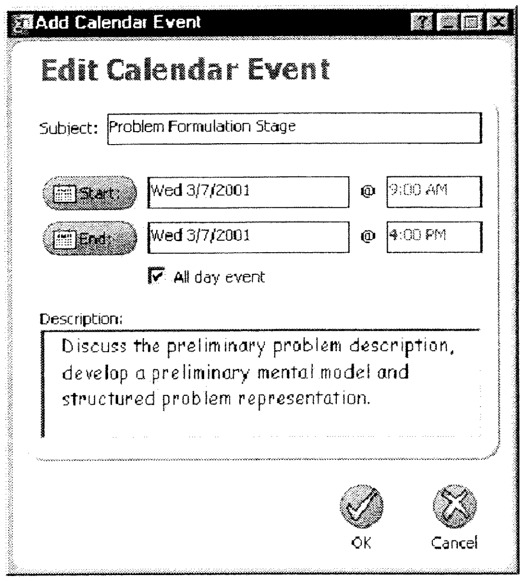
JoannaDT

We are beginning the collaboration in the discussion tool. Everyone please refer to my message posted.

Are you ready to brainstorm or do you still feel the need to discuss the problem more?

Figure 2.11 Schedule Tool.

Figure 2.12 shows the details when a user clicked on March 7th. Any team member can add to the schedule or to the details of particular days events.



Subject: Problem Formulation Stage

Wed 3/7/2001 @ 9:00 AM

Wed 3/7/2001 @ 4:00 PM

✓ All day event

Description:

Discuss the preliminary problem description, develop a preliminary mental model and structured problem representation.

Figure 2.12 Event details occurring on March 7th.

Groove Technology Analysis

Groove's features have the ability to *assist* in the coordination and problem solving effort of software development. However, software development needs more then synchronization, security features, file sharing, and asynchronous/synchronous messaging. Collaborative software development requires specific problem solving direction that includes efforts to enhance group cognitive activities while collaboratively solving problems, which this system does not incorporate.

For example, the brainstorming tool does allow the users to move around the different ideas for ranking purposes but there is no way to vote on items without everyone physically typing in their choice and them someone else tallying the votes to come up with the solution.

The Groove system does facilitate a group's performance of the major aspects of the group problem solving process. For example, there is a way to identify the tasks of the proposed solution, the tasks can easily be distributed, and the solution can easily be coordinated, communicated and modified. However, the entire problem solving and software development process using this system needs to be heavily guided through its various stages. An inexperienced software developer may skip important stages of the collaborative process.

2.7.4 Group Systems

GroupSystems is a groupware application with many useful collaborative tools. GroupSystems, must be installed on a shared server which is then used by team members on connected workstations. The GroupSystems tools support many aspects of collaboration including brainstorming, list building, information gathering, voting, organizing, prioritizing, and consensus building. The system can be used both synchronously and asynchronously.

An Overview of the GroupSystems Tools

This section presents a brief discussion of the main tools available in GroupSystems. A discussion and analysis of how these tools facilitate the collaborative problem solving and software development process will be given following the tool presentation.

Following a successful login, the main GroupSystems screen, shown in Figure 2.13, appears. The main screen defaults to the *people* tool. The GroupSystems tools are shown in the large lower right window of the main screen. The people tool displays which users have access to the active project folder. It also has a *sign in* feature where team members can record their names and other information about themselves.

All of the main tools available in the GroupSystems application are listed in the icons shown across the top of the main window, i.e. agenda, people, whiteboard, handouts, opinion, reports, briefcase, log, find, and folder list. The tool icons show no matter which tool is active.

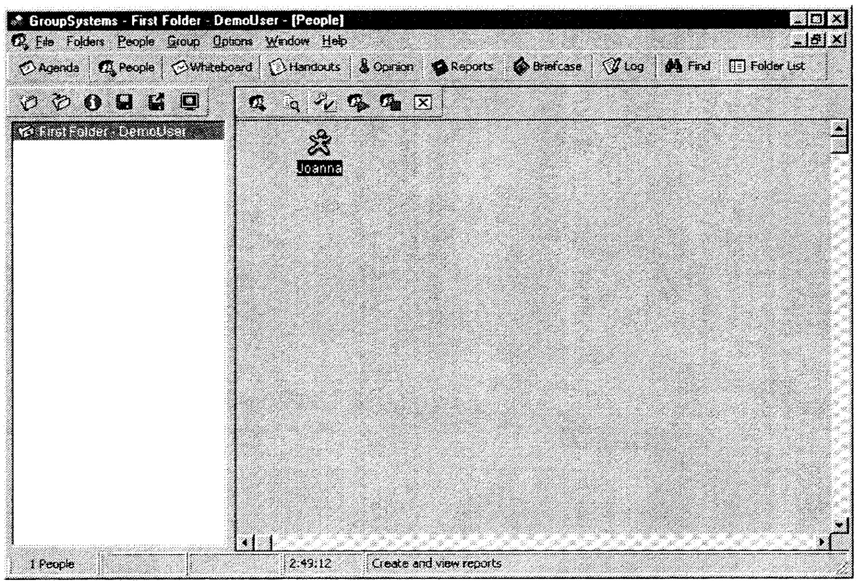
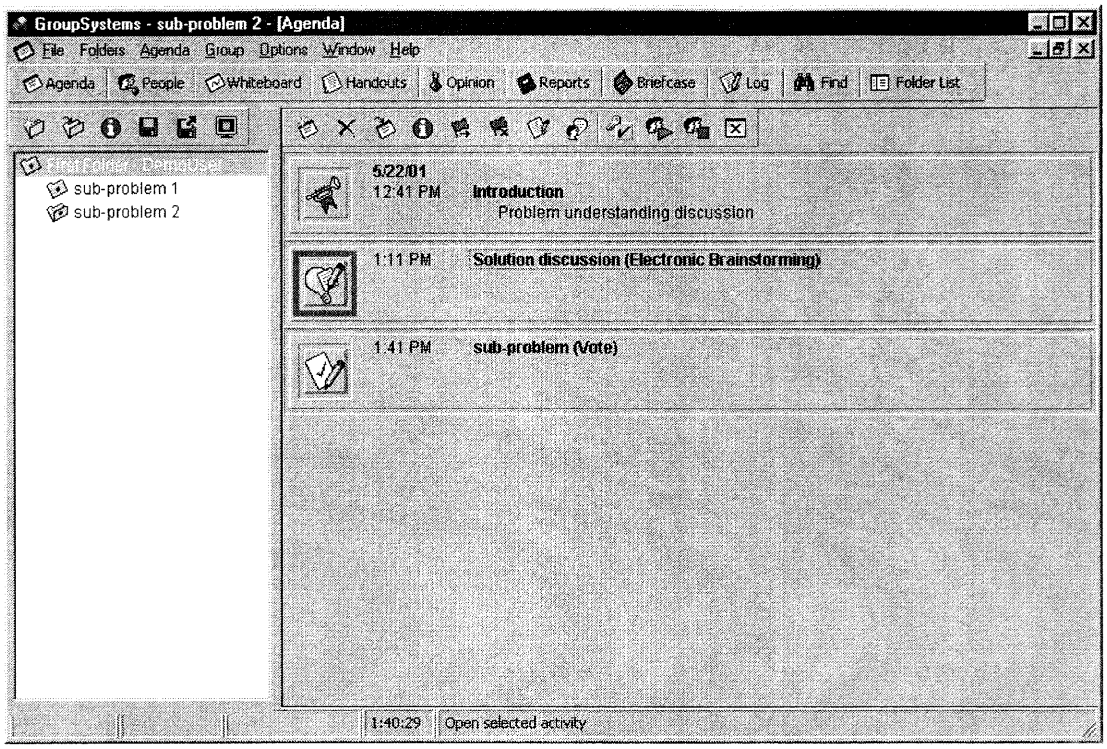

GroupSystems - First Folder - DemoUser - [People]
File Folders People Group Options Window Help
Agenda People Whiteboard Handouts Opinion Reports Briefcase Log Find Folder List
First Folder - DemoUser
Joanna
1 People
2:49:12
Create and view reports

Figure 2.13 Main GroupSystems Screen.

Clicking on the agenda icon activates the *Agenda* tool (Figure 2.14). Activities in the agenda tool can be added, deleted, or opened; opening an activity changes the tool window to the selected activity. Figure 2.15 shows the brainstorming tool activated by clicking the solution discussion activity in the agenda window.

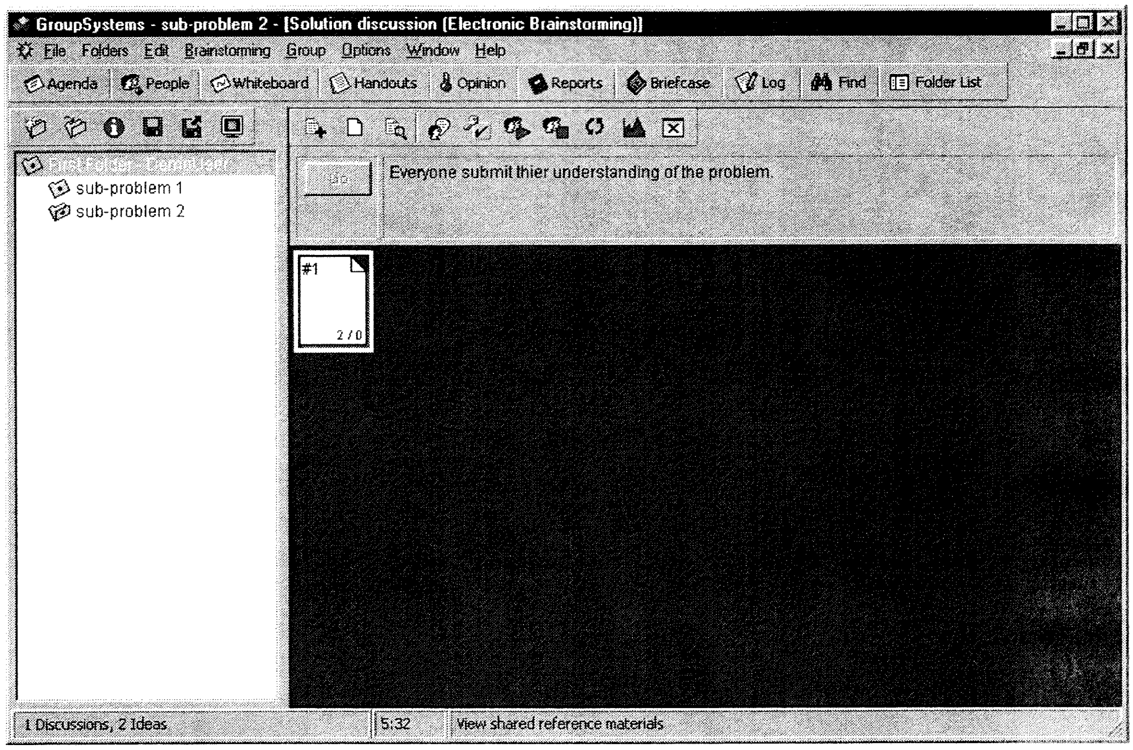

GroupSystems - sub-problem 2 - [Agenda]
File Folders Agenda Group Options Window Help
Agenda People Whiteboard Handouts Opinion Reports Briefcase Log Find Folder List
FirstFolder - DemoUser
sub-problem 1
sub-problem 2

5/22/01
12:41 PM Introduction
Problem understanding discussion
1:11 PM Solution discussion (Electronic Brainstorming)
1:41 PM sub-problem (Vote)

1:40:29 Open selected activity

Figure 2.14 Agenda Screen.

The initiator of the brainstorming session inserts a topic to initiate the discussion. In this case, the topic is "Everyone submit their understanding of the problem". A discussion sheet, also shown in Figure 2.15, was added. Team members can double click on the discussion sheet to add their ideas.

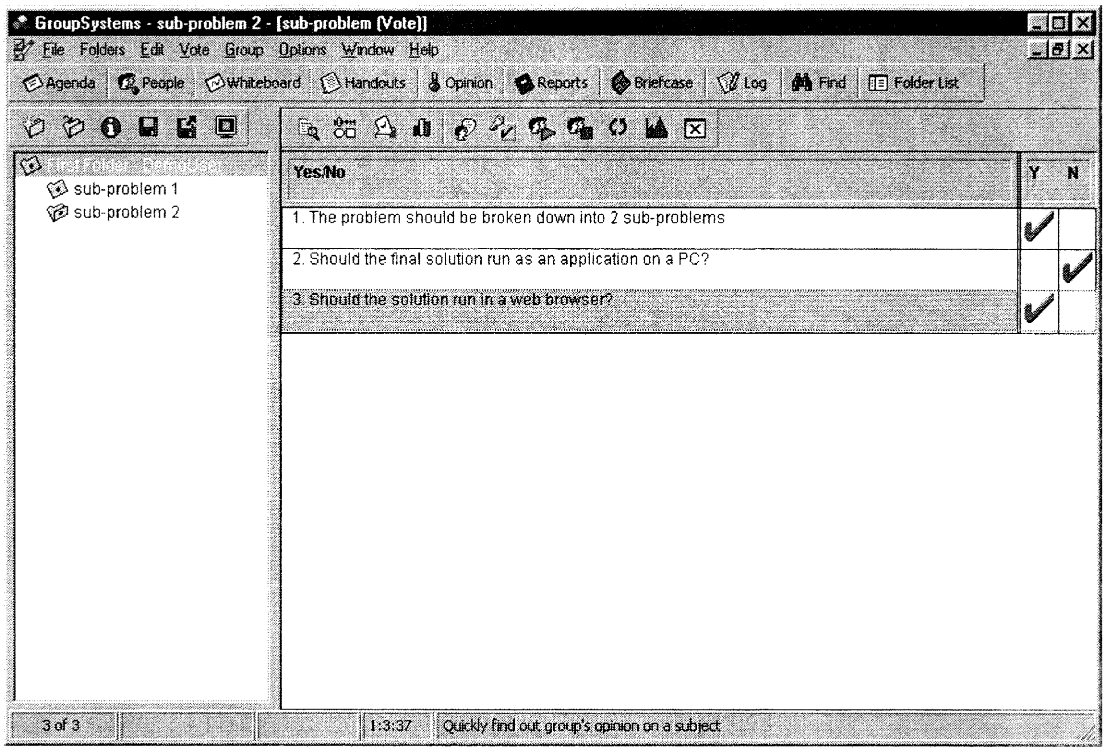

GroupSystems - sub-problem 2 - [Solution discussion (Electronic Brainstorming)]
File Folders Edit Brainstorming Group Options Window Help
Agenda People Whiteboard Handouts Opinion Reports Briefcase Log Find Folder List
sub-problem 1
sub-problem 2
Go
#1
2 / 0

Everyone submit thier understanding of the problem.

1 Discussions, 2 Ideas | 5:32 | View shared reference materials

Figure 2.15 Brainstorming Tool.

Figure 2.16 shows an example voting activity, which can be useful to organize ideas, developed in a brainstorming activity or simply to provide a basis for a group decision. Under the *vote* menu item, that will appear once a vote is in session, the leader of the voting activity can add voting items by choosing the *List-Buliding* menu item and entering a statement for the group to vote on. By clicking on the second icon in the voting activity box, the leader can choose the voting method for the ballot. Voting methods include: rank order, 10-point scale, multiple selection, yes/no, true/false, agree/disagree (5-point), agree disagree (4-point) or a custom method. Once all the voting is complete, the results can be viewed by clicking on the results icons.


GroupSystems - sub-problem 2 - [sub-problem (Vote)]
File Folders Edit Vote Group Options Window Help
Agenda People Whiteboard Handouts Opinion Reports Briefcase Log Find Folder List
First Folder - DemoUser
sub-problem 1
sub-problem 2
3 of 3
1:3:37

Yes/No	Y	N
1. The problem should be broken down into 2 sub-problems	✓	
2. Should the final solution run as an application on a PC?		✓
3. Should the solution run in a web browser?	✓	

Quickly find out group's opinion on a subject

Figure 2.16 Voting Example.

Figure 2.17 shows the *Whiteboard* tool that emulates a traditional physical whiteboard. Team members can draw and edit images on the whiteboard with the various tool icons located on the left of the tool. There are ten available pages for drawing. The users can print or save them to a file upon completion.

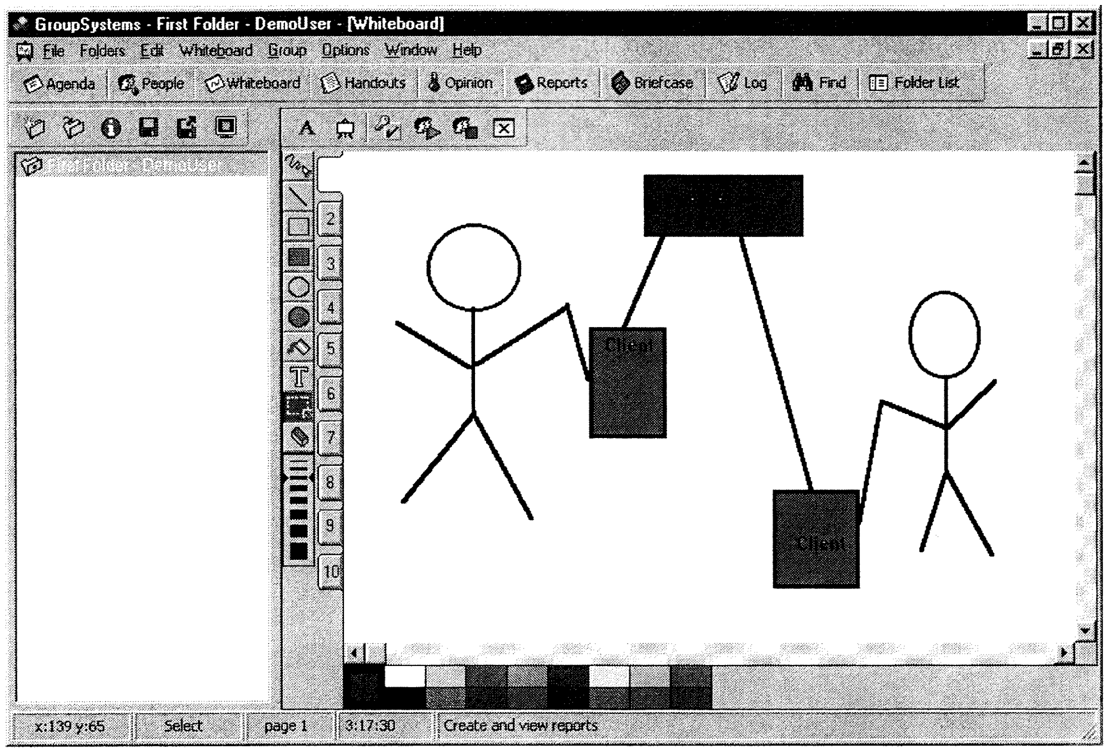

GroupSystems - First Folder - DemoUser - [Whiteboard]
File Folders Edit Whiteboard Group Options Window Help
Agenda People Whiteboard Handouts Opinion Reports Briefcase Log Find Folder List
First Folder - DemoUser
Client
Client
x:139 y:65
Select
page 1
3:17:30
Create and view reports

Figure 2.17 Whiteboard Tool.

The *Handout* tool, shown in Figure 2.18, is essentially a document storage area for the team to view up to 100 reference files. This tool gives access to reports, multimedia files, mission statements, spreadsheet projections, or visual materials such as graphs, charts, and diagrams. All files are read only. When a team member double clicks a document, the application associated with the document is automatically opened.

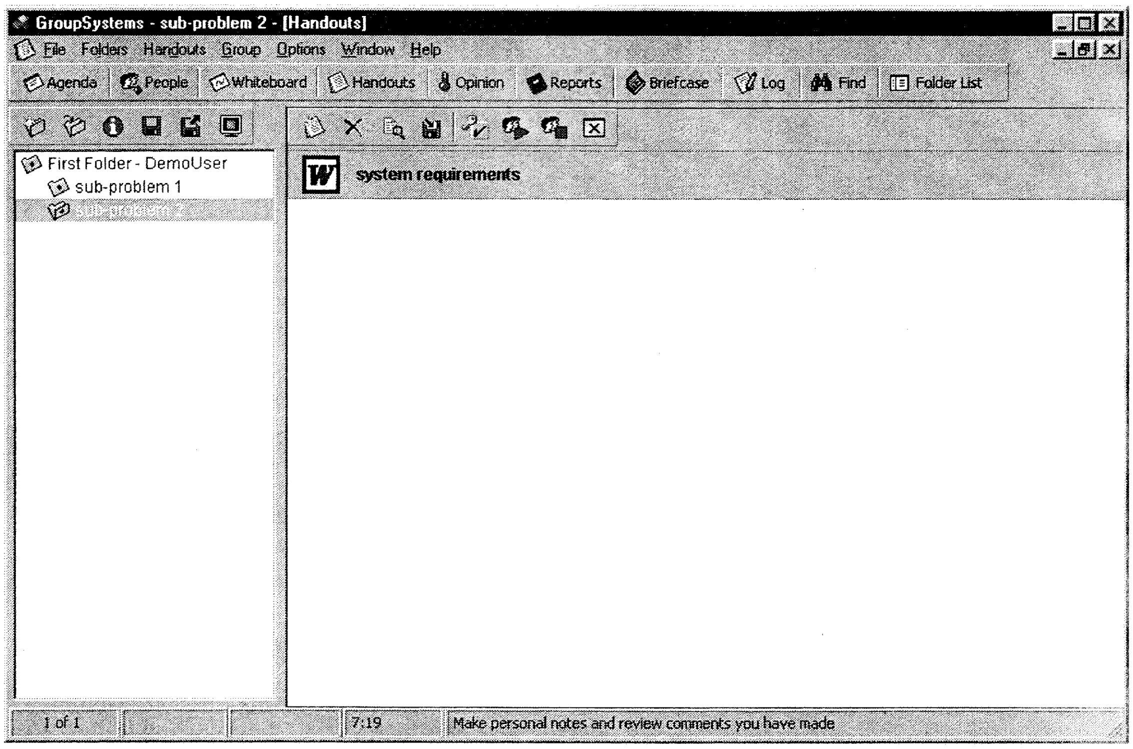
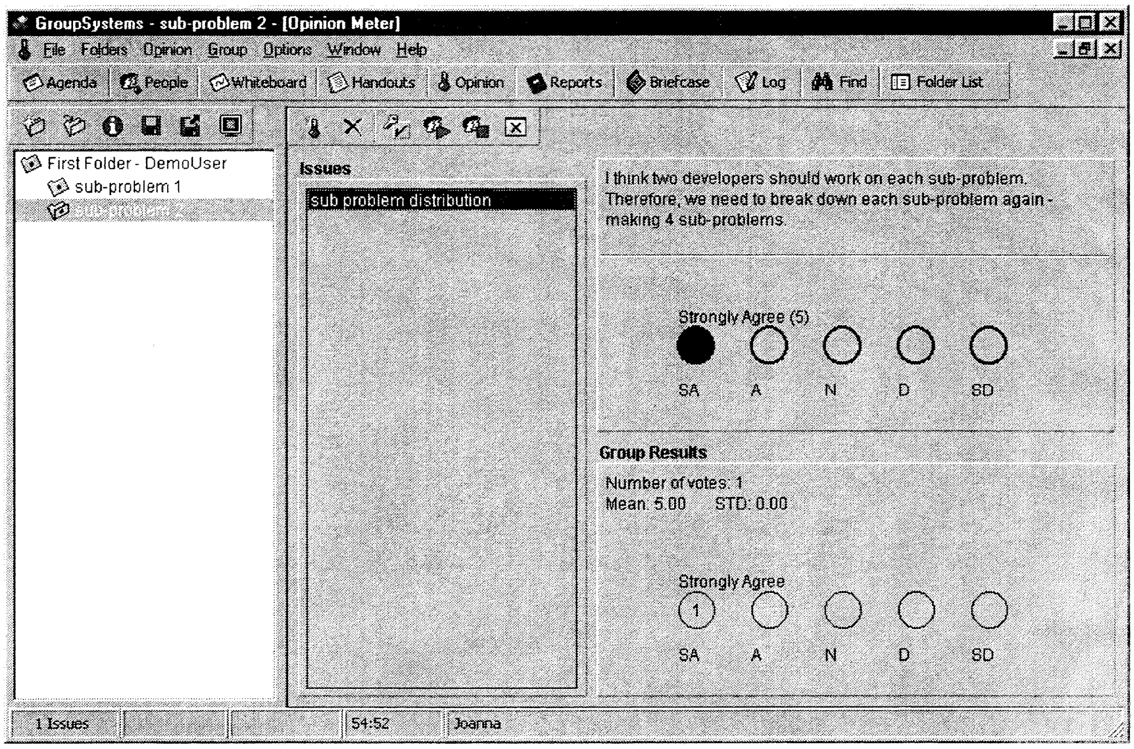

GroupSystems - sub-problem 2 - [Handouts]
File Folders Handouts Group Options Window Help
Agenda People Whiteboard Handouts Opinion Reports Briefcase Log Find Folder List
First Folder - DemoUser
sub-problem 1
sub-problem 2
system requirements
1 of 1
7:19
Make personal notes and review comments you have made

Figure 2.18 Handouts Tool.

The *Opinion* tool is essentially a faster, simpler voting tool. This tool provides a flexible and informal means of gauging the opinions of the team. For example, Figure 2.19 shows how a team member initiates/proposes a vote on breaking a problem solution by breaking it into four sub problems instead of the planned two sub problems. The team member who submitted the opinion can choose from three different voting schemes: yes/no, agree/disagree, and a 10-point scale.


GroupSystems - sub-problem 2 - [Opinion Meter]
File Folders Opinion Group Options Window Help
Agenda People Whiteboard Handouts Opinion Reports Briefcase Log Find Folder List
First Folder - DemoUser
sub-problem 1
sub-problem 2
Issues
SA A N D SD
Group Results
1 Issues
54:52
Joanna

sub problem distribution

I think two developers should work on each sub-problem. Therefore, we need to break down each sub-problem again - making 4 sub-problems.

Strongly Agree (5)

Number of votes: 1
Mean: 5.00 STD: 0.00

Strongly Agree

Figure 2.19 Opinion Tool.

The *Report* tool is similar to the handouts tool in that the team can store project related reports in this tool. The difference between the report tool and the handout tool is that it enables the creation of aggregate report files containing data from several activities and resources in the folder. The *Briefcase* tool, shown in Figure 2.20, allows easy access to commonly used utilities such as: calculator, clipboard and notepad etc. This tool can be personalized by added other commonly used applications such as e-mail or a word processor.

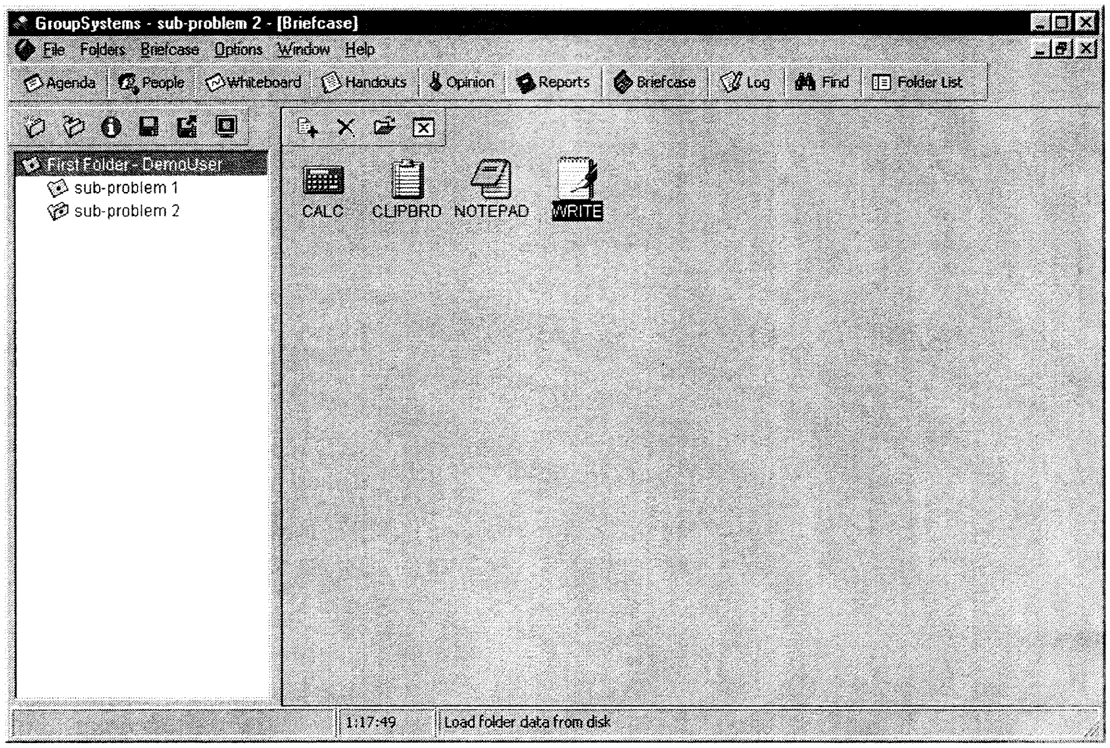
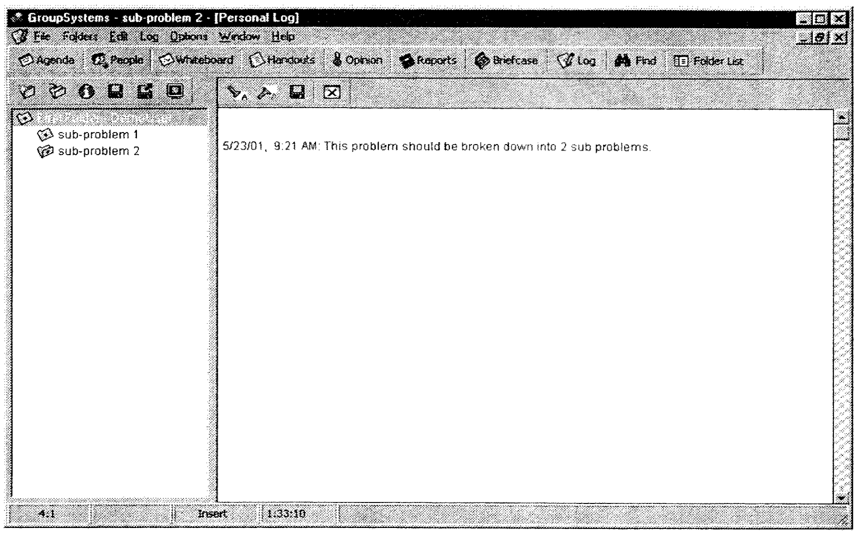


Figure 2.20 Briefcase Tool.

The *Personal Log* tool, shown in Figure 2.21, is where team members can keep track of anything. For example, if workstations are shared, each user can keep track of what they modified in the project. In addition to manual logging, automatic logging can be enabled. If automatic logging is enabled, each comment made in the brainstorming tool will be stored.


GroupSystems - sub-problem 2 - [Personal Log]
File Folders Edit Log Options Window Help
Agenda People Whiteboard Handouts Opinion Reports Briefcase Log Find Folder List
sub-problem 1
sub-problem 2
4:1 Insert 1:33:10

5/23/01, 9:21 AM: This problem should be broken down into 2 sub problems.

Figure 2.21 Log Tool.

The last tool is *Find.* The Find tool can help search through many folders to find a particular activity. This can be useful if a team member is invited to join a project when collaboration is already in progress.

GroupSystems Technology Analysis

GroupSystems, (GroupSystems.com, 2001), is an excellent collaborative tool. Its main goal is to support collaborative knowledge activities like strategic planning and risk assessment. Although this tool was not directly intended for software development, it has most of the features required to collaboratively solve a software problem. For example, it has a brainstorming tool designed 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 no step-by-step process to guide the team through the problem solving and software development process, but all of the tools needed to facilitate each aspect of collaborative problem solving and software development process is available in the GroupSystems tool. The only groupware 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 missing.

2.7.5 Lotus Notes

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

An Illustrative Example

The initial Notes R5 screen is shown in Figure 2.22. It provides convenient access to all Notes features: e-mail, calendar, address book, and to-do list.

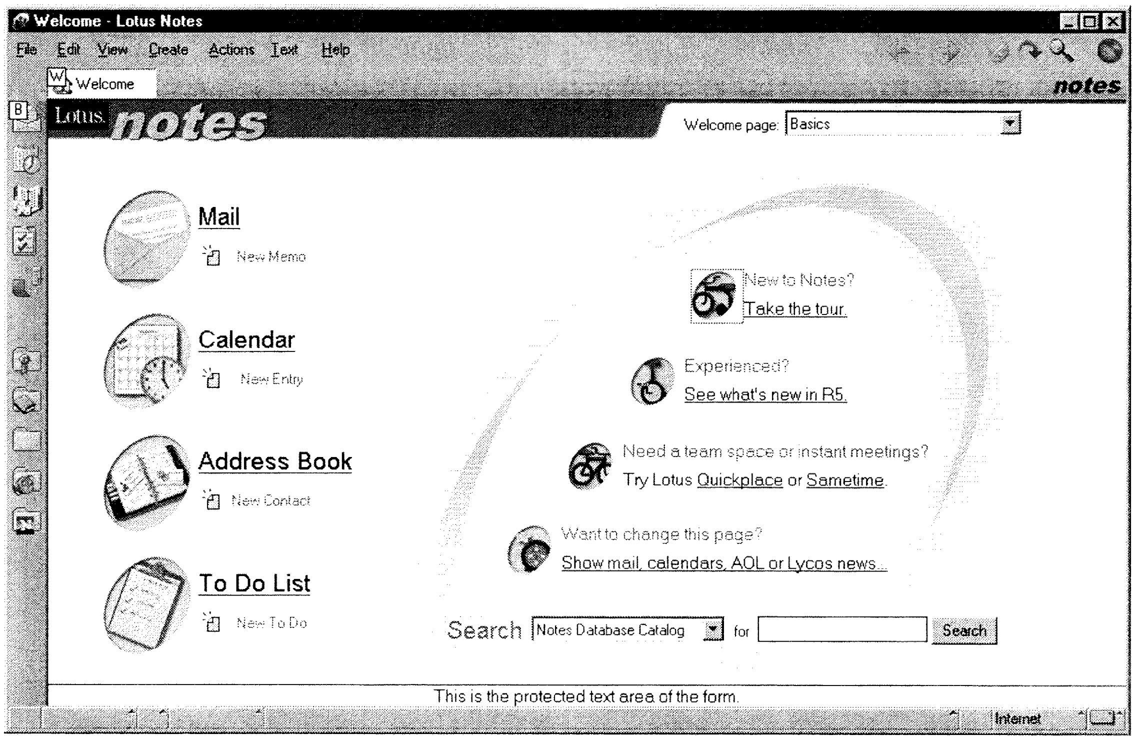
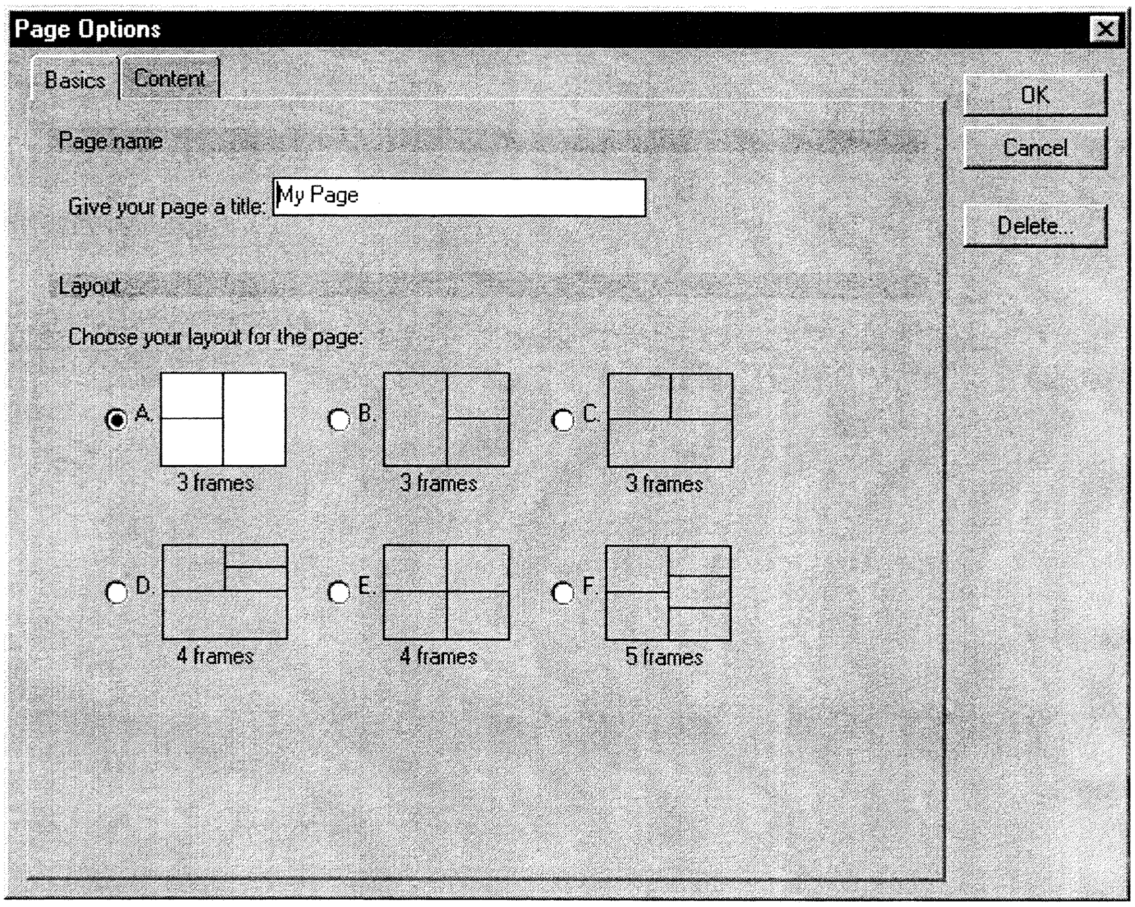

Welcome - Lotus Notes
File Edit View Create Actions Text Help
Welcome
Lotus notes
Welcome page: Basics
Mail
New Memo
Calendar
New Entry
Address Book
New Contact
To Do List
New To Do
New to Notes?
Take the tour.
Experienced?
See what's new in R5.
Need a team space or instant meetings?
Try Lotus Quickplace or Sametime.
Want to change this page?
Show mail, calendars, AOL or Lycos news...
Search Notes Database Catalog for
Search
This is the protected text area of the form.
Internet

Figure 2.22 Notes R5 Welcome Page.

The Notes interface is partially modifiable, allowing the user, for example, to begin with different start up screens with different selections of links and a different look-and-feel. This modification functionality is accessed through the drop-down menu. One selects *create new page style* from the *Welcome page* drop down box in the upper right-hand corner of the screen in Figure 2.22. A *Page Options* window then appears as shown in Figure 2.23. The *Basics* tab in the Options window allows the user to choose the frame layout of the screen. The *Content* tab lets the user choose which items appear in the layout.



Give your page a title: My Page

[✓ A.] 3 frames
B. 3 frames
C. 3 frames
D. 4 frames
E. 4 frames
F. 5 frames

Figure 2.23 Page Options Window.

On the far left of the main Lotus Notes screen is a *bookmark bar* that contains Note's key features in addition to providing access to the user's browser bookmarks, a replica tool, and database access. The browser bookmarks are synchronized with the user's Netscape and Explorer browsers. Individual Notes users can access any website that is book marked in either of these browsers or a specific Web address can be typed in the address field shown in Figure 2.24 in the upper right hand corner of the screen.

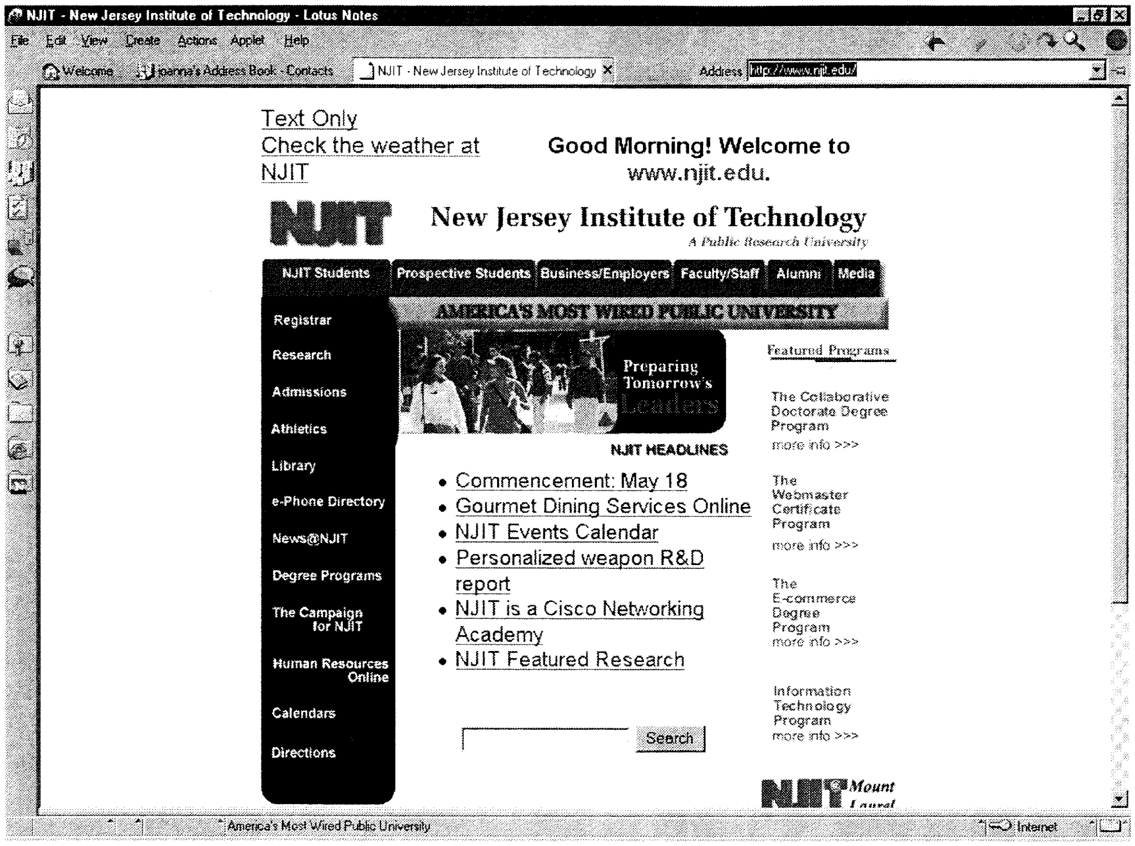
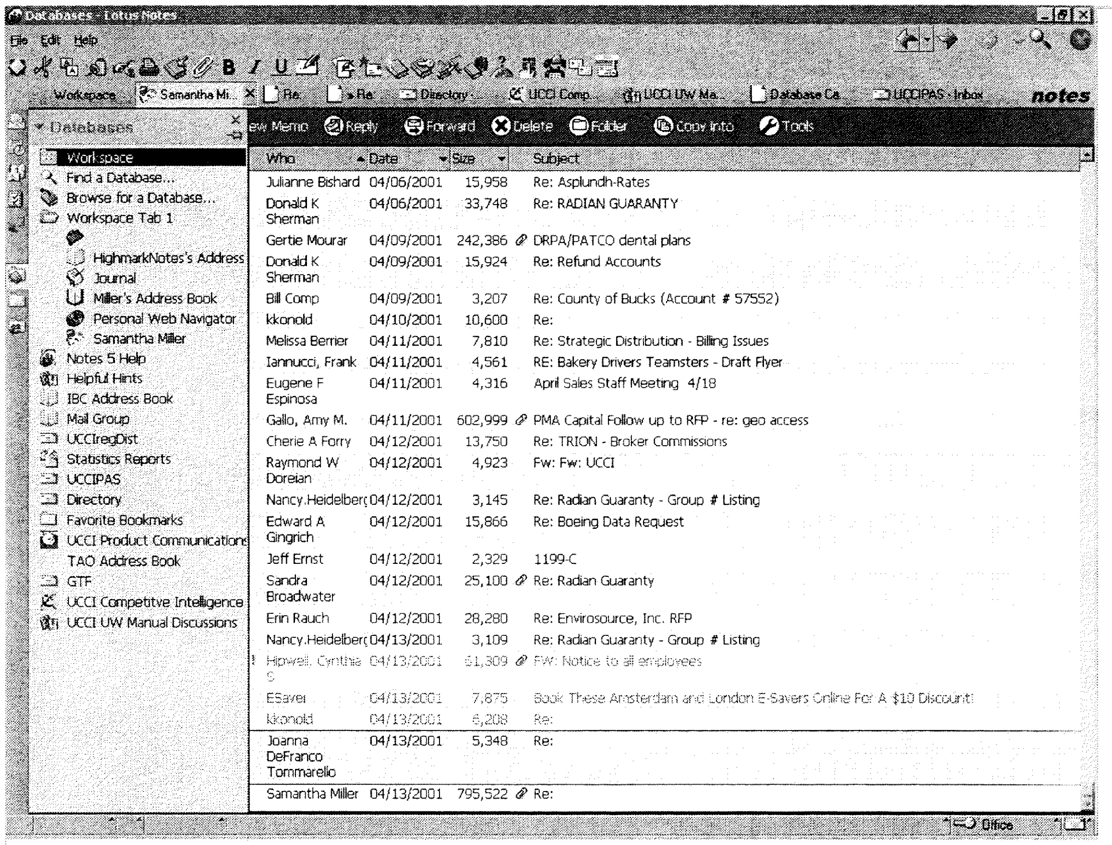

NJIT - New Jersey Institute of Technology - Lotus Notes
File Edit View Create Actions Applet Help
Welcome
Joanna's Address Book - Contacts
NJIT - New Jersey Institute of Technology
Address http://www.njit.edu/
Text Only
Check the weather at NJIT
Good Morning! Welcome to www.njit.edu.
New Jersey Institute of Technology
A Public Research University
NJIT Students
Prospective Students
Business/Employers
Faculty/Staff
Alumni
Media
AMERICA'S MOST WIRED PUBLIC UNIVERSITY
Registrar
Research
Admissions
Athletics
Library
e-Phone Directory
News@NJIT
Degree Programs
The Campaign for NJIT
Human Resources Online
Calendars
Directions
Preparing Tomorrow's Leaders
NJIT HEADLINES
Commencement: May 18
Gourmet Dining Services Online
NJIT Events Calendar
Personalized weapon R&D report
NJIT is a Cisco Networking Academy
NJIT Featured Research
Search
Featured Programs
The Collaborative Doctorate Degree Program
more info >>>
The Webmaster Certificate Program
more info >>>
The E-commerce Degree Program
more info >>>
Information Technology Program
more info >>>
NJIT Mount Laurel
America's Most Wired Public University
Internet

Figure 2.24 Notes Browser Tool.

The Notes replica tool allows a user to work with Notes off-line. For example a user can replicate their mail database so they have a current copy of their complete mail file with them when disconnected from the network. When a user eventually does connect to the mail server, Notes sends any pending outgoing mail to the server, gets new mail, and pushes any recent changes to their mail, made on the local replica, to the server

Most of the databases accessed from Notes are shared databases, stored on servers accessible to multiple Notes users. A user can view a list of accessible databases by merely depressing a database icon on the bookmark bar. The list of databases then appears in a panel as shown in Figure 2.25. Whatever Notes tool was previously being used continues to appear on the right side of the screen, in this case the mail In-box of the user.

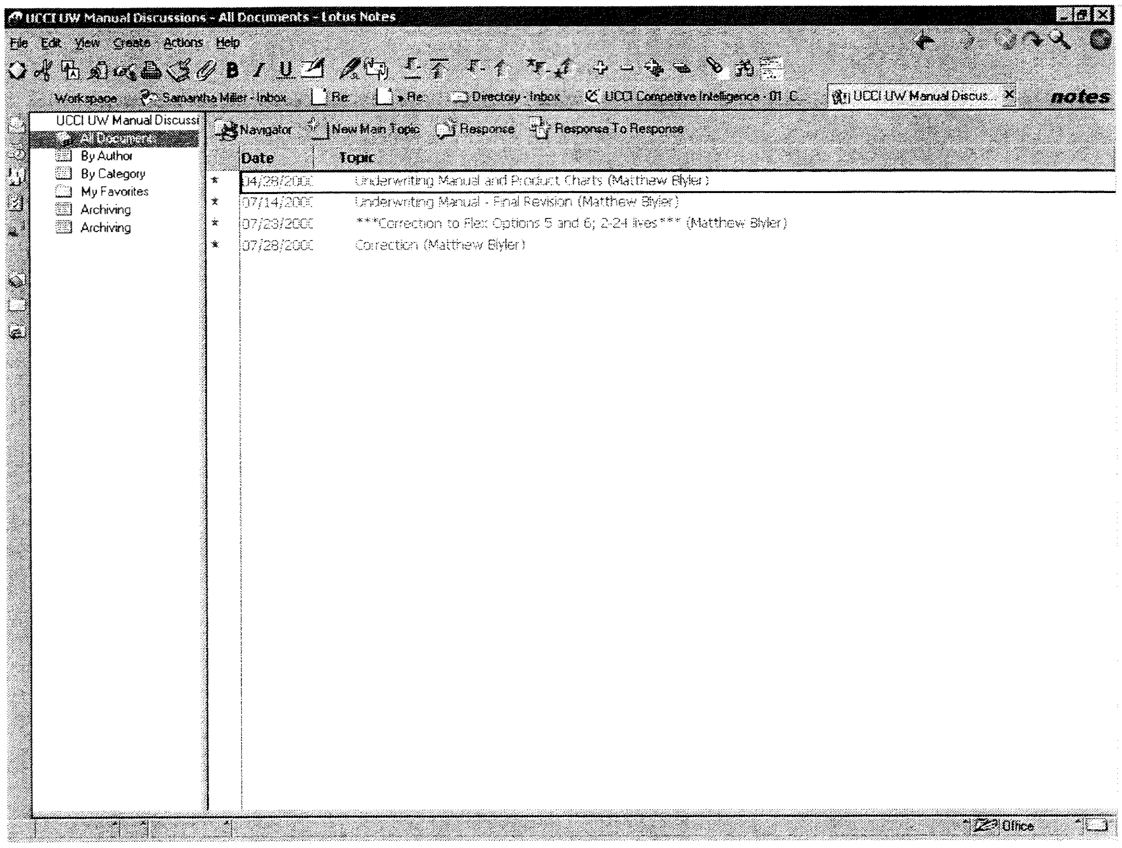


- Workspace
- Find a Database...
- Browse for a Database...
- Workspace Tab 1
  - HighmarkNotes's Address
  - Journal
  - Miller's Address Book
  - Personal Web Navigator
  - Samantha Miller
- Notes 5 Help
- Helpful Hints
- IBC Address Book
- Mail Group
- UCCIregDist
- Statistics Reports
- UCCIPAS
- Directory
- Favorite Bookmarks
- UCCI Product Communications
- TAO Address Book
- GTF
- UCCI Competitve Intelligence
- UCCI UW Manual Discussions

Who	Date	Size		Subject
Julianne Bishard	04/06/2001	15,958		Re: Asplundh-Rates
Donald K Sherman	04/06/2001	33,748		Re: RADIAN GUARANTY
Gertie Mourar	04/09/2001	242,386	📎	DRPA/PATCO dental plans
Donald K Sherman	04/09/2001	15,924		Re: Refund Accounts
Bill Comp	04/09/2001	3,207		Re: County of Bucks (Account # 57552)
kkonold	04/10/2001	10,600		Re:
Melissa Berrier	04/11/2001	7,810		Re: Strategic Distribution - Billing Issues
Iannucci, Frank	04/11/2001	4,561		RE: Bakery Drivers Teamsters - Draft Flyer
Eugene F Espinosa	04/11/2001	4,316		April Sales Staff Meeting 4/18
Gallo, Amy M.	04/11/2001	602,999	📎	PMA Capital Follow up to RFP - re: geo access
Cherie A Forry	04/12/2001	13,750		Re: TRION - Broker Commissions
Raymond W Doreian	04/12/2001	4,923		Fw: Fw: UCCI
Nancy.Heidelber(	04/12/2001	3,145		Re: Radian Guaranty - Group # Listing
Edward A Gingrich	04/12/2001	15,866		Re: Boeing Data Request
Jeff Ernst	04/12/2001	2,329		1199-C
Sandra Broadwater	04/12/2001	25,100	📎	Re: Radian Guaranty
Erin Rauch	04/12/2001	28,280		Re: Envirosource, Inc. RFP
Nancy.Heidelber(	04/13/2001	3,109		Re: Radian Guaranty - Group # Listing
Hipwell, Cynthia S	04/13/2001	61,309	📎	FW: Notice to all employees
ESaver	04/13/2001	7,875		Book These Amsterdam and London E-Savers Online For A $10 Discount!
kkonold	04/13/2001	6,208		Re:
Joanna DeFranco Tommarello	04/13/2001	5,348		Re:
Samantha Miller	04/13/2001	795,522	📎	Re:

Figure 2.25 Data Base Panel.

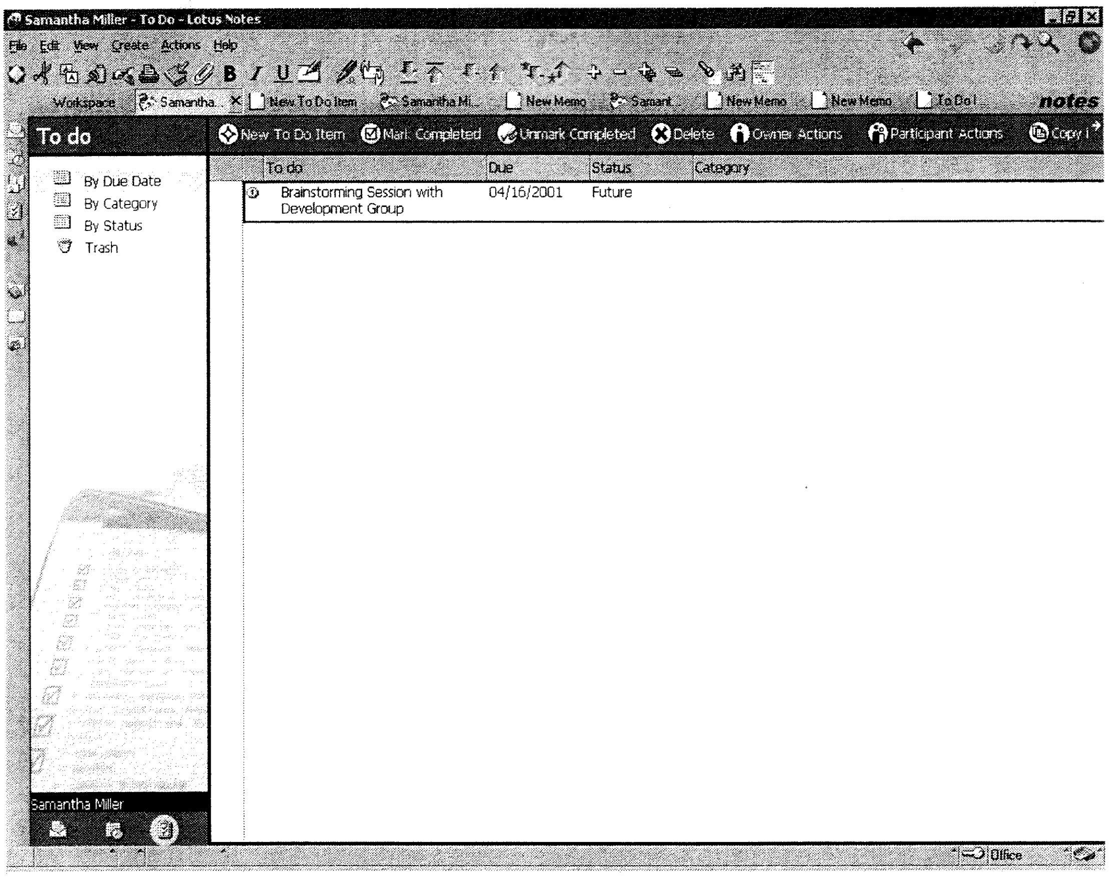
When the user selects a specific database, the contents of the database then appear in the window located to the right of the databases listing. Figure 2.26 illustrates this with an example database called *UCCI UW Manual Discussions*.



	Date	Topic
*	04/28/2000	Underwriting Manual and Product Charts (Matthew Blyler)
*	07/14/2000	Underwriting Manual - Final Revision (Matthew Blyler)
*	07/23/2000	***Correction to Flex Options 5 and 6; 2-24 lives*** (Matthew Blyler)
*	07/28/2000	Correction (Matthew Blyler)

Figure 2.26 A database content example.

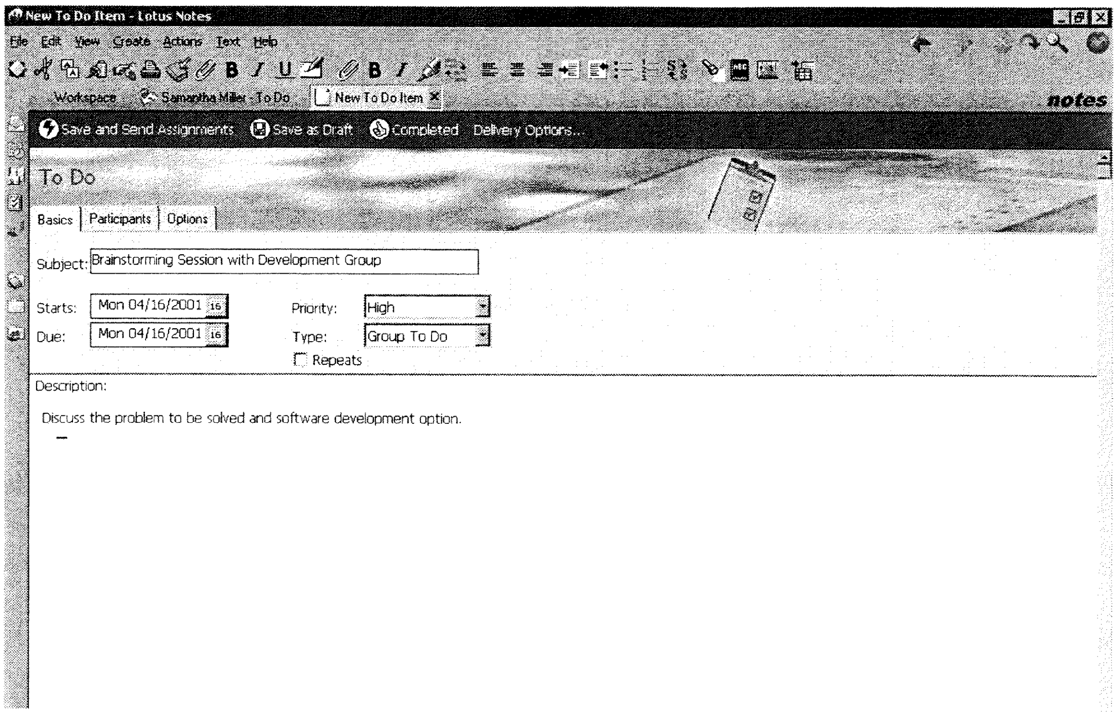
The to-do tool, shown in Figure 2.27, is accessed by depressing its icon on the bookmark bar. The list of actions that a user can select from are located above the to-do list, i.e. New To-do Item, Mark Completed, etc. The to-do list contains agenda items for the group project.



To do	Due	Status	Category
Brainstorming Session with Development Group	04/16/2001	Future	

Figure 2.27 To-Do Tool example.

The details of a *To Do* item can be read by double clicking on the item so a window appears as shown in Figure 2.28.



Subject: Brainstorming Session with Development Group

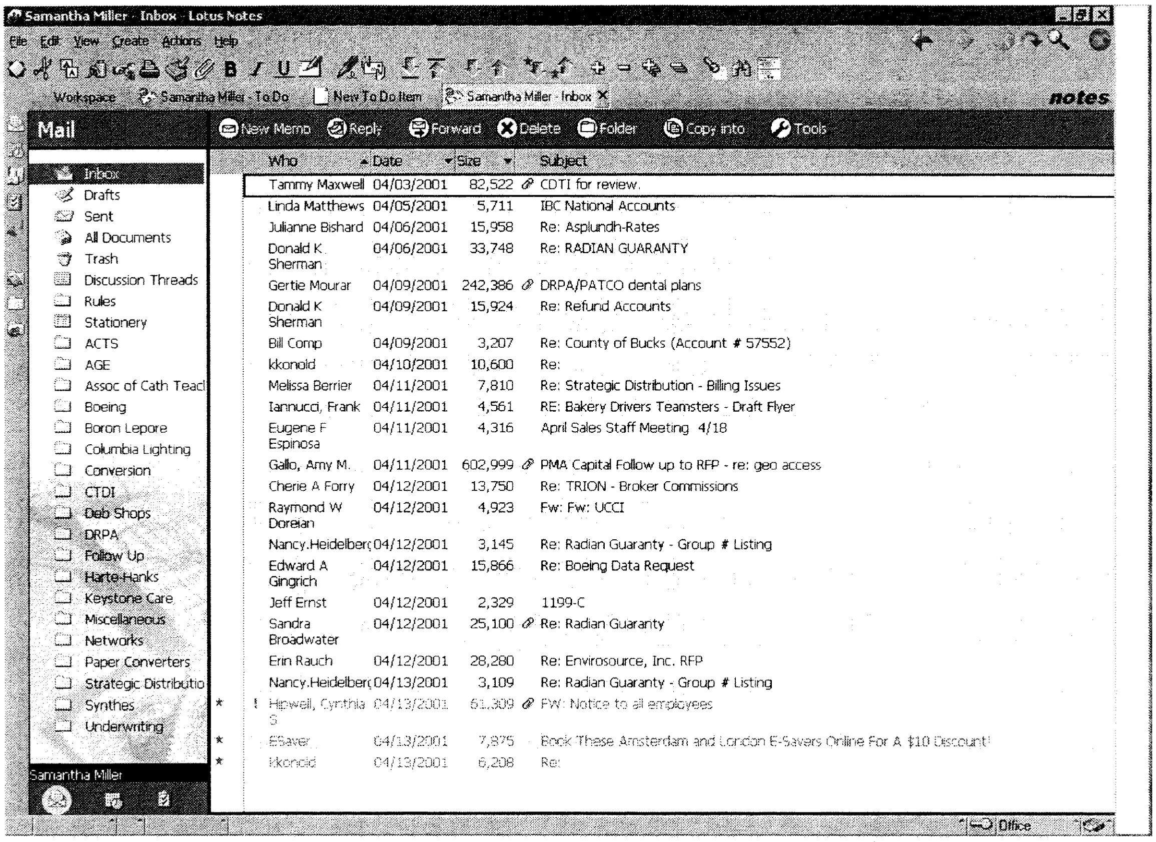
Starts: Mon 04/16/2001
Due: Mon 04/16/2001
Priority: High
Type: Group To Do
Repeats

Description:

Discuss the problem to be solved and software development option.

Figure 2.28 To-Do item detail.

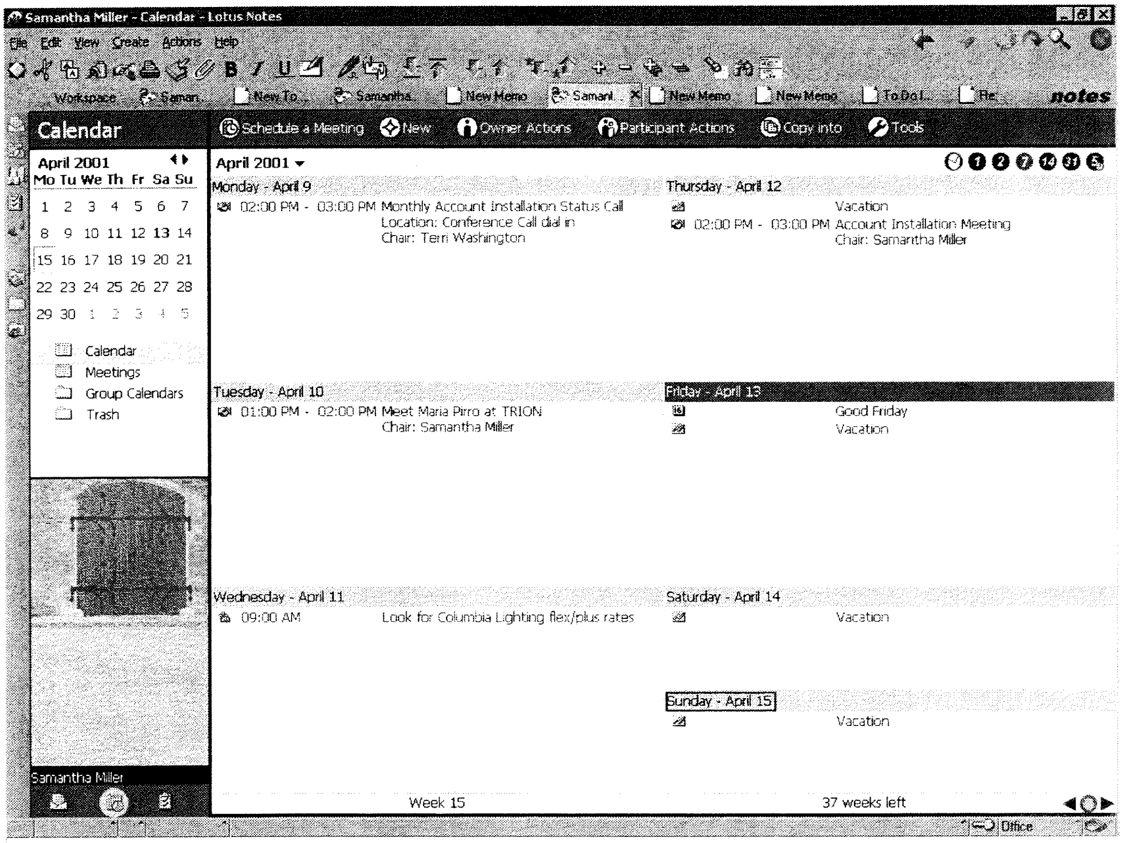
The Mail tool shown in Figure 2.29 is like other electronic mail tools, providing the ability to create, reply to, or forward messages, attach files, etc. The user effects these actions by clicking on an icon in an *action bar* above the mail inbox.



		Who	Date	Size		Subject
		Tammy Maxwell	04/03/2001	82,522	📎	CDTI for review.
		Linda Matthews	04/05/2001	5,711		IBC National Accounts
		Julianne Bishard	04/06/2001	15,958		Re: Asplundh-Rates
		Donald K Sherman	04/06/2001	33,748		Re: RADIAN GUARANTY
		Gertie Mourar	04/09/2001	242,386	📎	DRPA/PATCO dental plans
		Donald K Sherman	04/09/2001	15,924		Re: Refund Accounts
		Bill Comp	04/09/2001	3,207		Re: County of Bucks (Account # 57552)
		kkonold	04/10/2001	10,600		Re:
		Melissa Berrier	04/11/2001	7,810		Re: Strategic Distribution - Billing Issues
		Iannucci, Frank	04/11/2001	4,561		RE: Bakery Drivers Teamsters - Draft Flyer
		Eugene F Espinosa	04/11/2001	4,316		April Sales Staff Meeting 4/18
		Gallo, Amy M.	04/11/2001	602,999	📎	PMA Capital Follow up to RFP - re: geo access
		Cherie A Forry	04/12/2001	13,750		Re: TRION - Broker Commissions
		Raymond W Doreian	04/12/2001	4,923		Fw: Fw: UCCI
		Nancy.Heidelber(	04/12/2001	3,145		Re: Radian Guaranty - Group # Listing
		Edward A Gingrich	04/12/2001	15,866		Re: Boeing Data Request
		Jeff Ernst	04/12/2001	2,329		1199-C
		Sandra Broadwater	04/12/2001	25,100	📎	Re: Radian Guaranty
		Erin Rauch	04/12/2001	28,280		Re: Envirosource, Inc. RFP
		Nancy.Heidelber(	04/13/2001	3,109		Re: Radian Guaranty - Group # Listing
*	!	Hipwell, Cynthia S	04/13/2001	61,309	📎	FW: Notice to all employees
*		ESaver	04/13/2001	7,875		Book These Amsterdam and London E-Savers Online For A $10 Discount!
*		kkonold	04/13/2001	6,208		Re:

Figure 2.29 Mail Tool.

The Notes calendar, shown in Figure 2.30, allows users to organize their activities such as meetings. Figure 2.30 shows a 7-day view of the calendar, and 1, 2, 7, 14, or 31-day views of their schedule are accessed by clicking the number in the upper right of the screen. The calendar is scrollable. Invitations to meeting to other users are created by clicking on the *schedule a meeting* action item above the calendar. If a group is working jointly on a project, a user can view the calendars of other group members, a facility which is intended to simplify scheduling meetings by allowing users to determine open times of group members. The calendaring is relatively automatic. If an individual schedules a group or subgroup for an activity at a particular time, the system automatically verifies that the schedule is compatible the calendars of all the designated members, and updates each participant's individual calendars accordingly.



April 2001						
Mo	Tu	We	Th	Fr	Sa	Su
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

April 2001

Monday - April 9
02:00 PM - 03:00 PM Monthly Account Installation Status Call
Location: Conference Call dial in
Chair: Terri Washington

Tuesday - April 10
01:00 PM - 02:00 PM Meet Maria Pirro at TRION
Chair: Samantha Miller

Wednesday - April 11
09:00 AM Look for Columbia Lighting flex/plus rates

Thursday - April 12
Vacation
02:00 PM - 03:00 PM Account Installation Meeting
Chair: Samantha Miller

Friday - April 13
Good Friday
Vacation

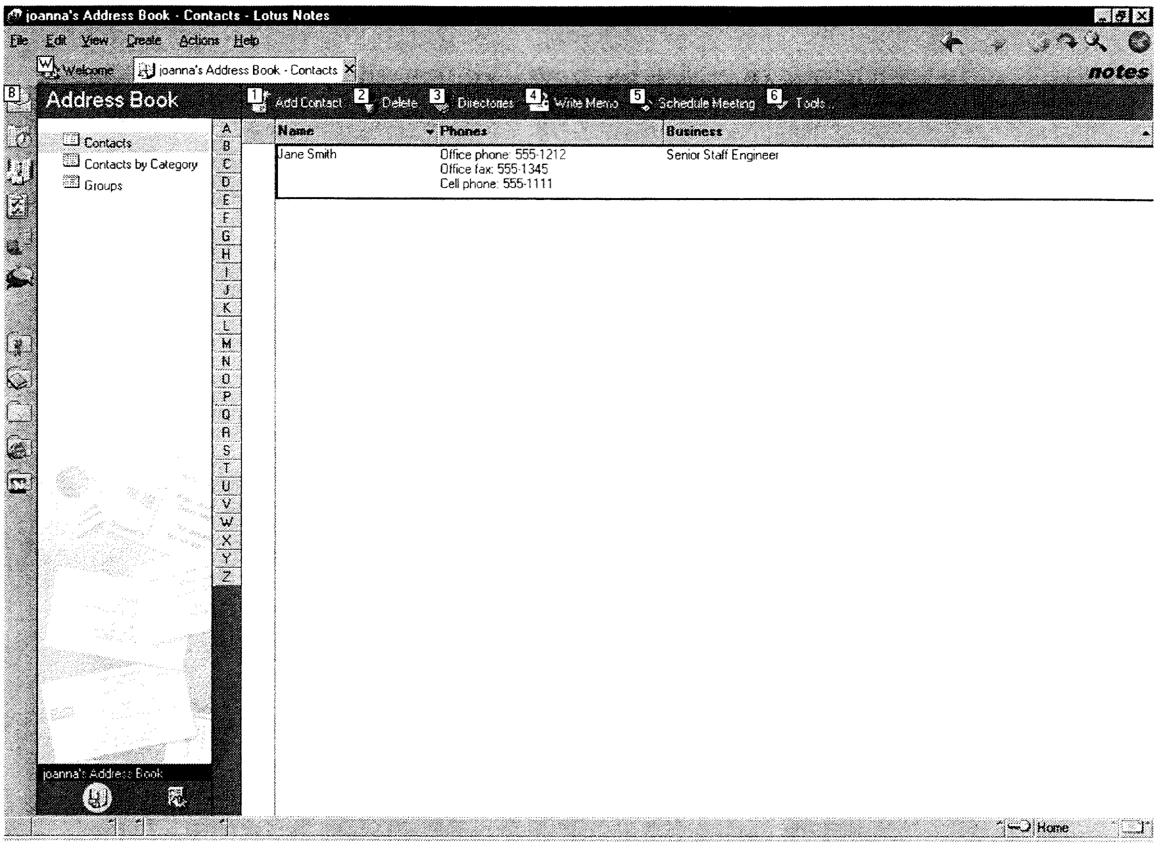
Saturday - April 14
Vacation

Sunday - April 15
Vacation

Week 15 37 weeks left

Figure 2.30 Calendar Tool.

The address book shown in Figure 2.31 is used to manage and track contacts. The Action bar above the contact window allows users to add/delete contacts, send e-mail, schedule meetings, etc.


joanna's Address Book - Contacts - Lotus Notes
File Edit View Create Actions Help
Welcome
joanna's Address Book - Contacts
notes
Address Book
Add Contact
Delete
Directories
Write Memo
Schedule Meeting
Tools
Contacts
Contacts by Category
Groups
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
joanna's Address Book
Home

Name	Phones	Business
Jane Smith	Office phone: 555-1212 Office fax: 555-1345 Cell phone: 555-1111	Senior Staff Engineer

Figure 2.31 Address book.

Lotus Notes Analysis

Lotus Notes provides general-purpose assistance for coordinating a group's efforts, but it does not provide specific tools for real time or asynchronous collaboration activities. Certainly, additional collaboration tools are necessary for collaborating in the context of software development. 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.

Lotus also has two other more extensive colaborative tools for group collaboration: Sametime and QuickPlace. Sametime is a client-server environment. A Sametime server downloaded by the user group manages the environment; a separate client provides the local functionality for each user. *Sametime* provides a forum in which group can communicate in real time via instant messaging. Sametime lets each user know who is currently available for instant messaging. It allows team members to collaborate with other members on the same local network with a variety of tools including instant audio messaging and video, shared whiteboard, shared documents/drawings/presentations, text-based chat or instant messaging. These features are similar to those provided by the Groove application to be discussed subsequently.

QuickPlace is web-oriented environment, unlike the private network, client-server environment provided by Sametime. Quickplace provides a text-based chatroom environment which is enhanced to facilitate sharing and organizing of ideas, content, and tasks for a project. Quickplace also provides the ability to create rich text documents directly from a Web browser, file sharing, and e-mail. A shared file can only be modified by one user at a time. The software also provides a document revision history and facilitiesies checking documents in or out. There is also off-line access that will synchronize any modified data when you reconnect.

2.7.6 Microsoft Visual SourceSafe

Microsoft's Visual SourceSafe is a collaborative tool needed during the implementation phase of the problem solving and programming process. It stores current source code so team members can easily re-create previous versions and maintain an audit trail for any file. This tool can also reconcile file changes from multiple users and prevent accidental code overwriting with check out file locking, visual merge, and difference reporting.

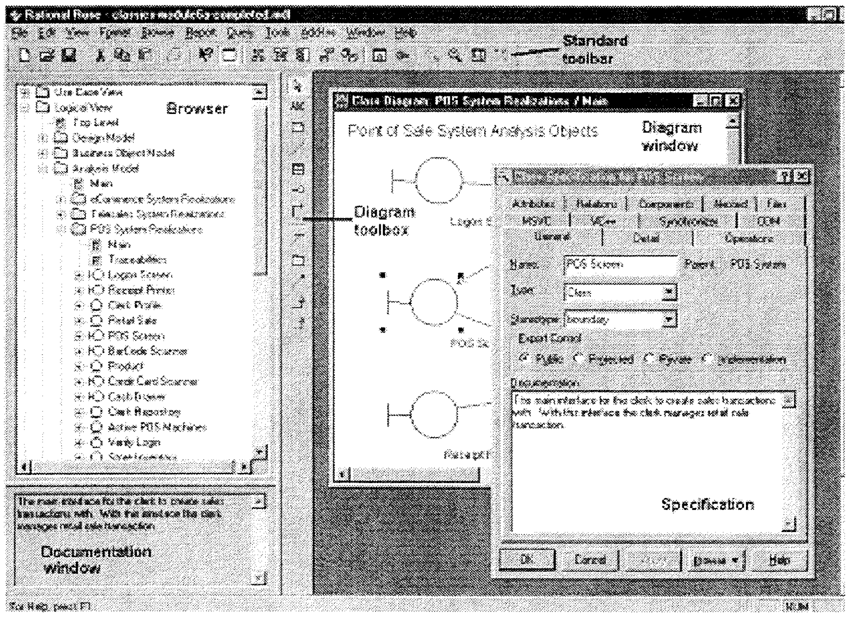
2.7.7 Rational Rose

Rose is an individually oriented up-to-date, object-oriented software development application that assists in visually modeling software, but is not intended as a collaborative tool. The models produced from using the Rose application can identify requirements and communicate information, identify component interactions and relationships.

Rose is compatible with 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 claims that visual modeling improves communication across the team through the use of a common graphical language.

A Detailed Review

Figure 2.32 shows the Rose graphical user interface. The main features are the standard toolbar, diagram toolbox, diagram window, browser, documentation window, and specifications. The *standard toolbar* located at the top of the GUI is always visible. The *diagram toolbox*, shown vertically in Figure 2.32, changes based on the active diagram displayed. The *diagram window* enables the user to display, create, and modify project related diagrams. Multiple diagrams can be open at one time. The *browser* allows the user to view the names and icons representing diagrams and model elements. Selecting 'documentation' from the view menu opens the *documentation window*. It allows the user to create a self-documenting model, which in turn generates self-documenting code. *Specifications* are dialog boxes that allow you to set or change model element properties. Changes made to a model element either through the specification or directly on the icon are automatically updated throughout the model.



The main interface for the clerk to create sales transactions with. With this interface the clerk manages retail sale transaction.

Figure 2.32 Rose GUI.

There are four different views to show the different aspects of the working model: use case, logical, component, and deployment. The *use case view*, shown in

Figure 2.33, helps the user to understand and use the system. Essentially, this view gives a graphical representation of the completed system.

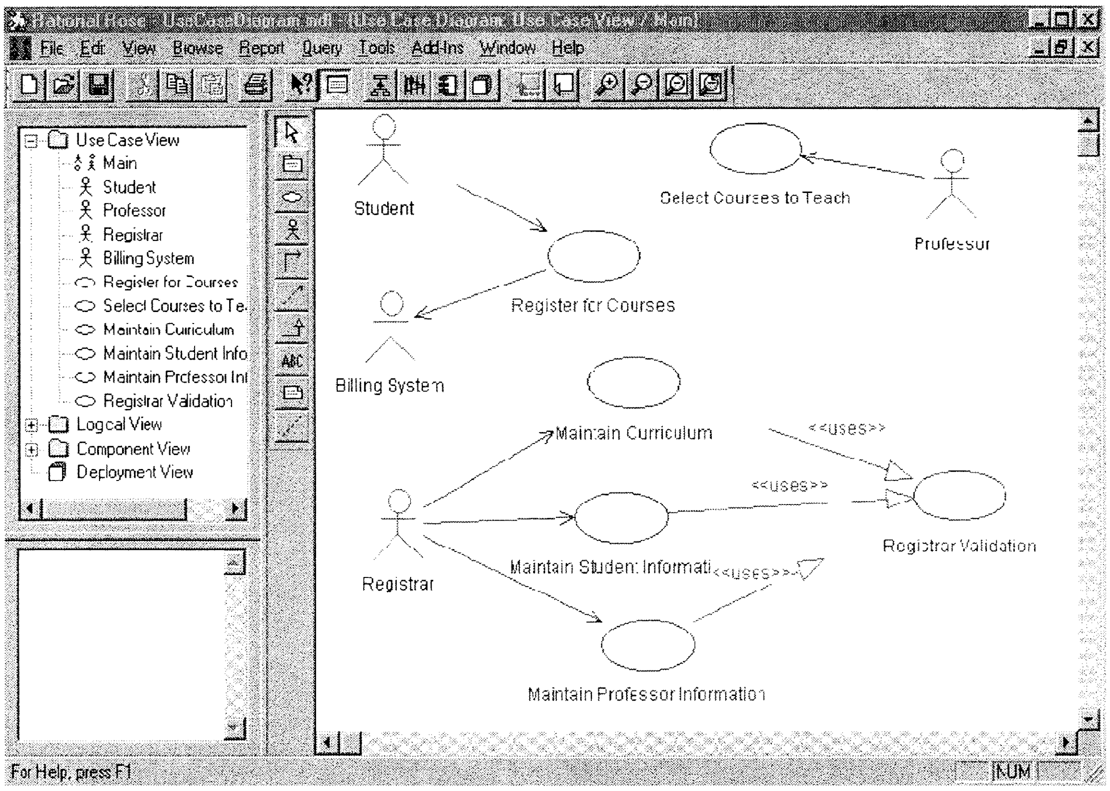


Figure 2.33 Use Case View example.

The *logical view* addresses the functional requirements of the system. For example, the class diagram shown in Figure 2.34 depicts the project classes for a university. The diagram also shows the logical relationships between classes, methods inside classes, and class variables.

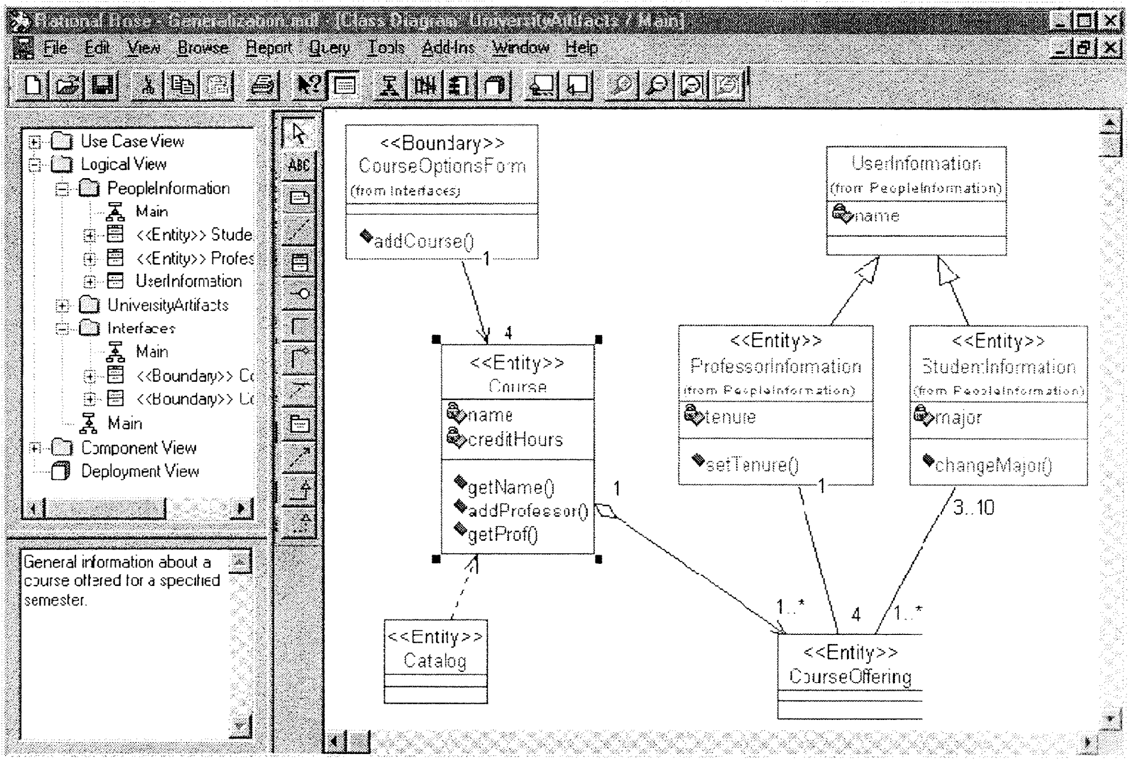


Figure 2.34 Logical View Example.

The *component view* addresses the software organization of the system with component diagrams. As shown in Figure 2.35 below, the component view displays information regarding the software, executable, and library components of the system.

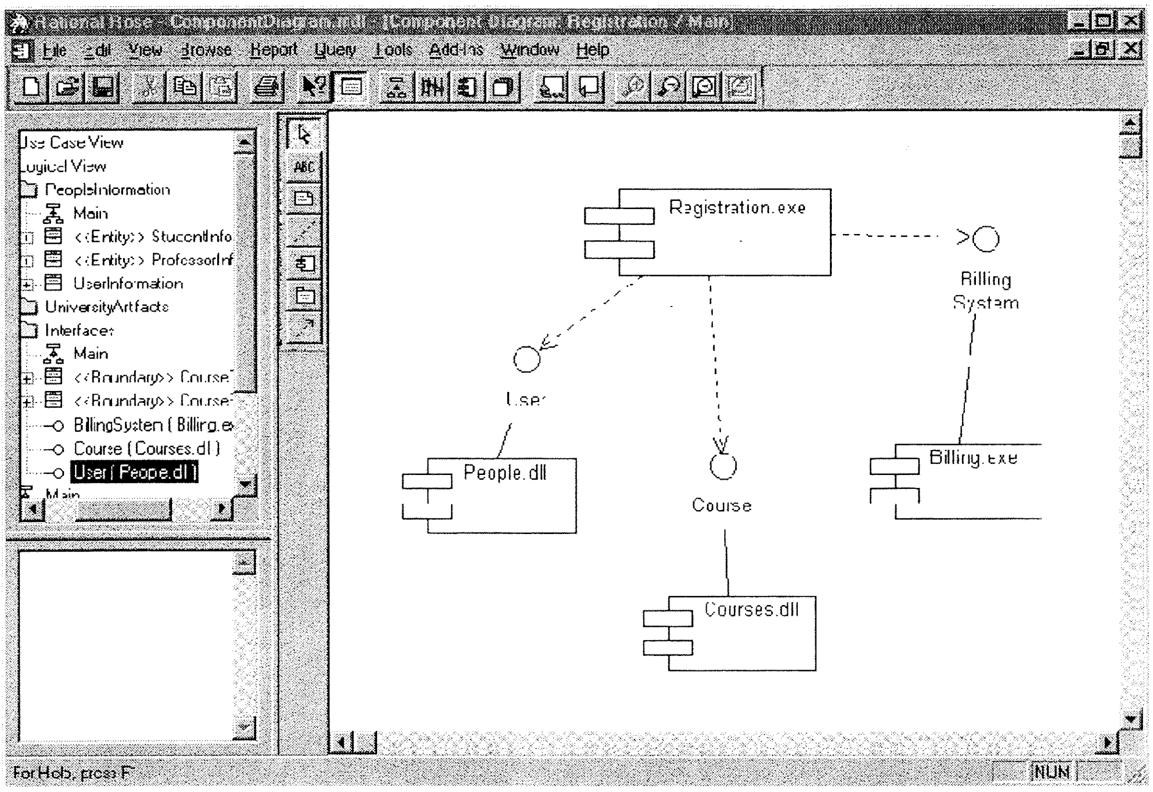


Figure 2.35 Component View Example.

The *deployment view*, shown in Figure 2.36, depicts the mapping of processes to hardware. This is useful when there is a distributed architecture involved in the system design, i.e. applications and servers in different physical locations.

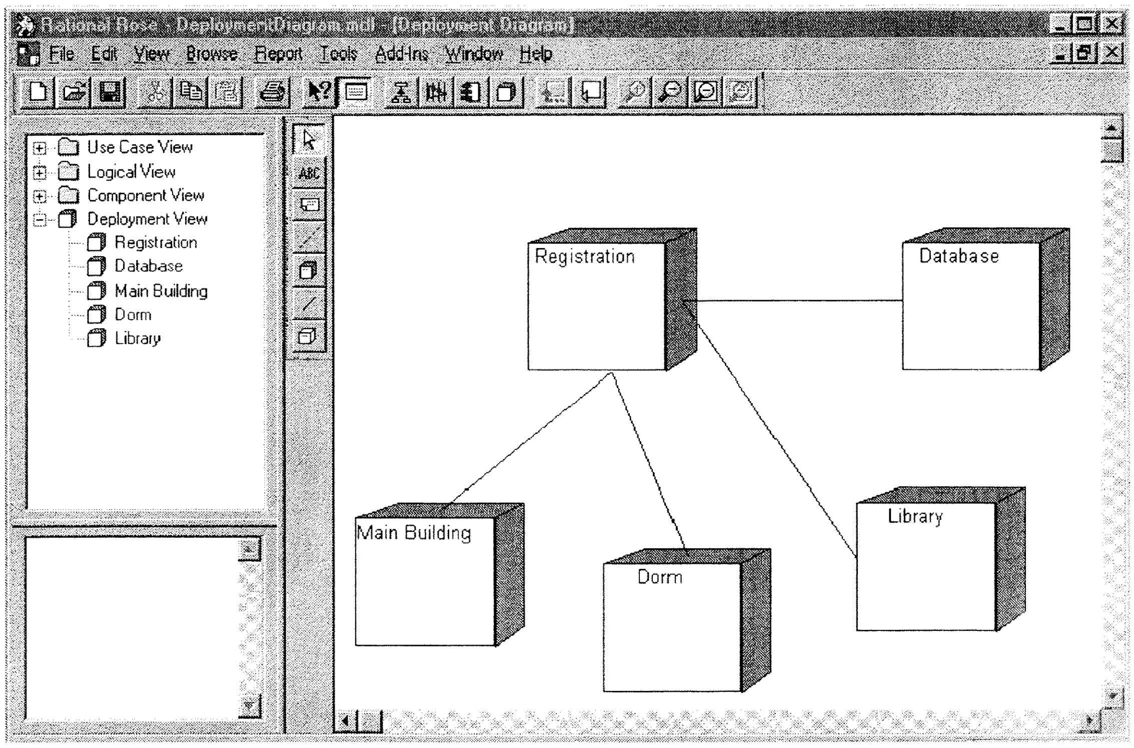


Figure 2.36 Deployment View Example.

All of the views presented are graphical representations of the elements in the developing system. These representations give multiple perspectives useful during the development stages of the system.

Rational Rose Technology Analysis

Rose is a good development tool with an emphasis on visual modeling. Rational claims that such visual modeling shortens the development life cycle, increases productivity, and improves software quality and team communication. Rose is not a specifically collaborative tool, but team members sharing various documents developed with the tool could very well increase team communication. In other words, team communication could be increased because of using a common modeling language. However, there are no build in provisions to communicate via the tool. There is no messaging, brainstorming or voting tools integrated into Rose. Rose is primarily a design tool. It is assumed that the initial problem solving stage is complete when Rose is used. Rose helps model solutions prior to the implementation stage of development.

2.7.7.1 Rational Requisite Pro Case Study. Requisite Pro is a requirements management tool, which 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.

A Detailed Discussion of Requisite Pro

Requisite Pro initializes with two windows: the tool palette and the view workplace. Figure 2.37 shows the tool palette, which is the main interface for accessing and working with Requisite Pro projects. The buttons on the tool palette window provide quick access to project information, requirements information, and views. The Tool Palette menu commands are also used to perform requirements management operations in a Requisite Pro project.

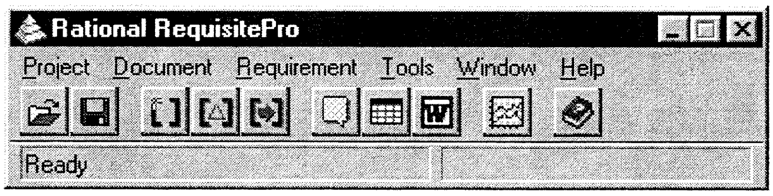


Figure 2.37 Tool Palette Window.

The main function of the View Workplace, shown in Figure 2.38, is to analyze and print requirements. It is also used to view the database of requirements created in different formats.

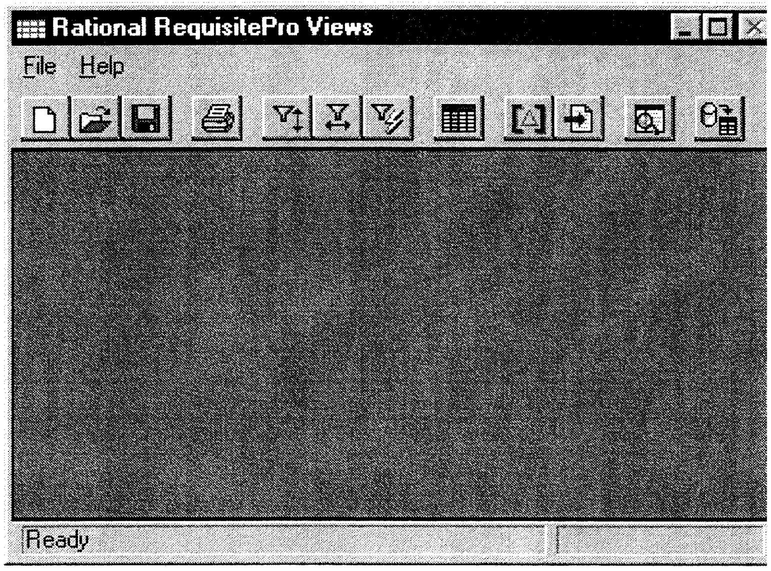
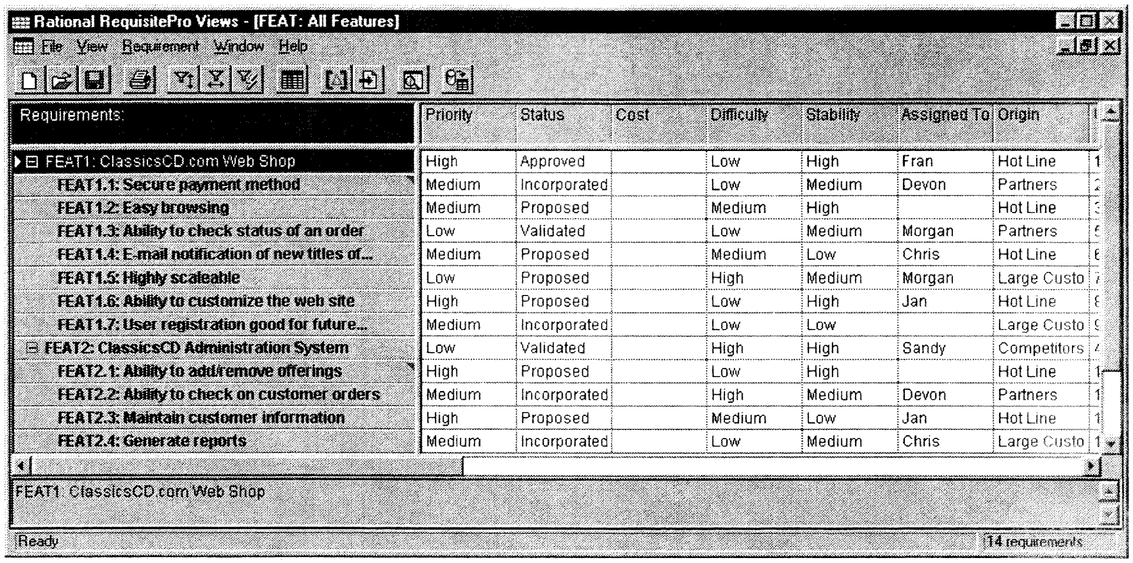


Figure 2.38 View Workplace Window.

The three formats, *attribute matrix*, *traceability matrix*, and *traceability tree* are used to display requirement attributes and relationships. The attribute matrix, displays all requirements and their attributes for a specified requirement type. The spreadsheet-like view, shown in Figure 2.39, displays requirements in rows and the attributes that describe the requirements in columns. You can add or change values in the attribute fields.


Rational RequisitePro Views - [FEAT: All Features]
File View Requirement Window Help

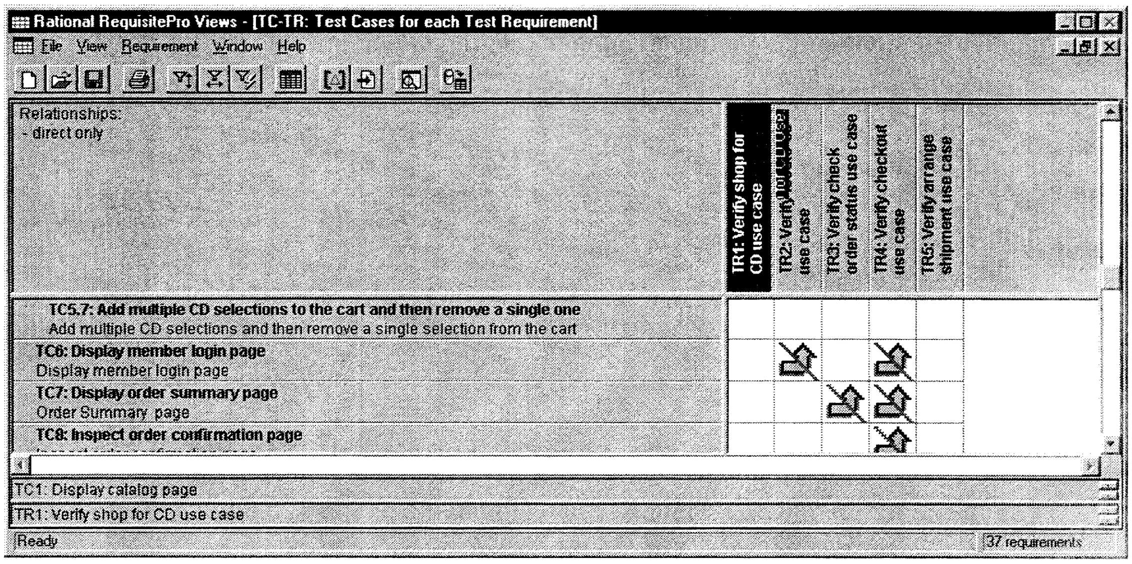
Requirements:	Priority	Status	Cost	Difficulty	Stability	Assigned To	Origin	I
FEAT1: ClassicsCD.com Web Shop	High	Approved		Low	High	Fran	Hot Line	1
FEAT1.1: Secure payment method	Medium	Incorporated		Low	Medium	Devon	Partners	2
FEAT1.2: Easy browsing	Medium	Proposed		Medium	High		Hot Line	3
FEAT1.3: Ability to check status of an order	Low	Validated		Low	Medium	Morgan	Partners	5
FEAT1.4: E-mail notification of new titles of...	Medium	Proposed		Medium	Low	Chris	Hot Line	6
FEAT1.5: Highly scaleable	Low	Proposed		High	Medium	Morgan	Large Custo	7
FEAT1.6: Ability to customize the web site	High	Proposed		Low	High	Jan	Hot Line	8
FEAT1.7: User registration good for future...	Medium	Incorporated		Low	Low		Large Custo	9
FEAT2: ClassicsCD Administration System	Low	Validated		High	High	Sandy	Competitors	4
FEAT2.1: Ability to add/remove offerings	High	Proposed		Low	High		Hot Line	1
FEAT2.2: Ability to check on customer orders	Medium	Incorporated		High	Medium	Devon	Partners	1
FEAT2.3: Maintain customer information	High	Proposed		Medium	Low	Jan	Hot Line	1
FEAT2.4: Generate reports	Medium	Incorporated		Low	Medium	Chris	Large Custo	1

FEAT1: ClassicsCD.com Web Shop

Ready 14 requirements

Figure 2.39 Attribute Matrix Example.

The traceability matrix, shown in Figure 2.40, illustrates the relationships between requirements of the same or different types. The Traceability Matrix displays the requirements of one requirement type as its rows, and the requirements of another (or the same) requirement type as its columns. This matrix is used to create, modify, and delete traceability relationships and view indirect relationships.


Rational RequisitePro Views - [TC-TR: Test Cases for each Test Requirement]
File View Requirement Window Help

Relationships:
- direct only

	TR1: Verify shop for CD use case	TR2: Verify [illegible] use case	TR3: Verify check order status use case	TR4: Verify checkout use case	TR5: Verify arrange shipment use case
TC5.7: Add multiple CD selections to the cart and then remove a single one Add multiple CD selections and then remove a single selection from the cart					
TC6: Display member login page Display member login page		↗		↗	
TC7: Display order summary page Order Summary page			↗	↗	
TC8: Inspect order confirmation page				↗	

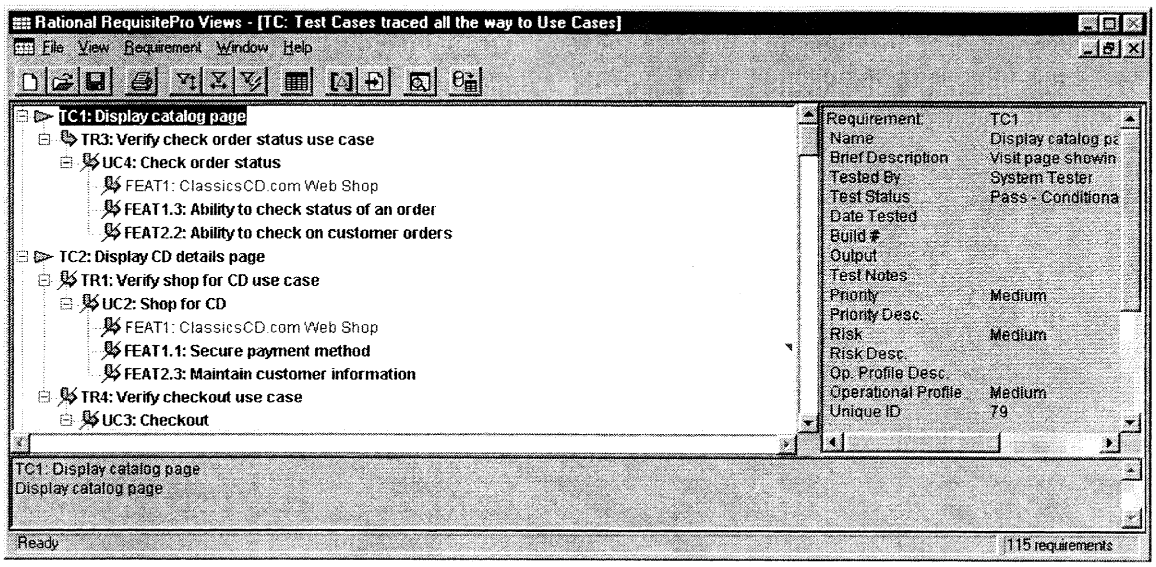
TC1: Display catalog page

TR1: Verify shop for CD use case

Ready 37 requirements

Figure 2.40 Traceability Matrix Example.

The traceability tree, shown in Figure 2.41, displays all requirements of one type and all requirements related to/from requirements of the root type. It provides a graphical view of relationships to or from root requirements of a specific requirement type. The tag, name, and attributes of the selected requirement are displayed in the attribute pane.


Rational RequisitePro Views - [TC: Test Cases traced all the way to Use Cases]
File View Requirement Window Help
Ready
115 requirements

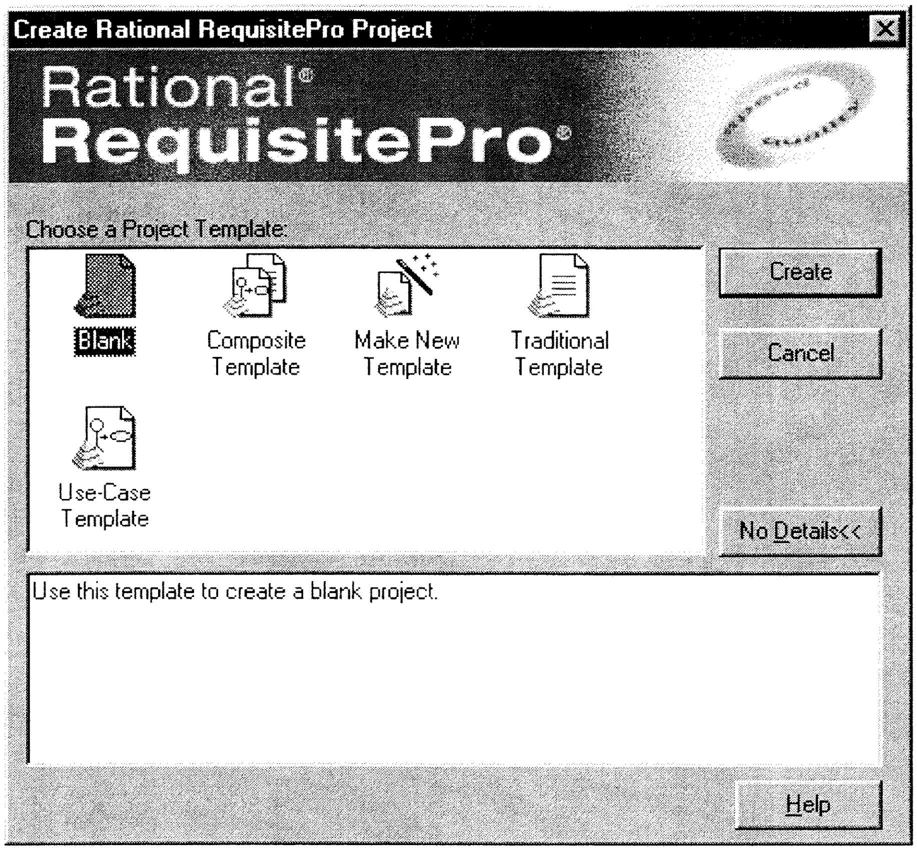
```
TC1: Display catalog page
  TR3: Verify check order status use case
    UC4: Check order status
      FEAT1: ClassicsCD.com Web Shop
      FEAT1.3: Ability to check status of an order
      FEAT2.2: Ability to check on customer orders
TC2: Display CD details page
  TR1: Verify shop for CD use case
    UC2: Shop for CD
      FEAT1: ClassicsCD.com Web Shop
      FEAT1.1: Secure payment method
      FEAT2.3: Maintain customer information
  TR4: Verify checkout use case
    UC3: Checkout
```

Requirement	TC1
Name	Display catalog pa
Brief Description	Visit page showin
Tested By	System Tester
Test Status	Pass - Conditiona
Date Tested	
Build #	
Output	
Test Notes	
Priority	Medium
Priority Desc.	
Risk	Medium
Risk Desc.	
Op. Profile Desc.	
Operational Profile	Medium
Unique ID	79

TC1: Display catalog page
Display catalog page

Figure 2.41 Traceability Tree Example.

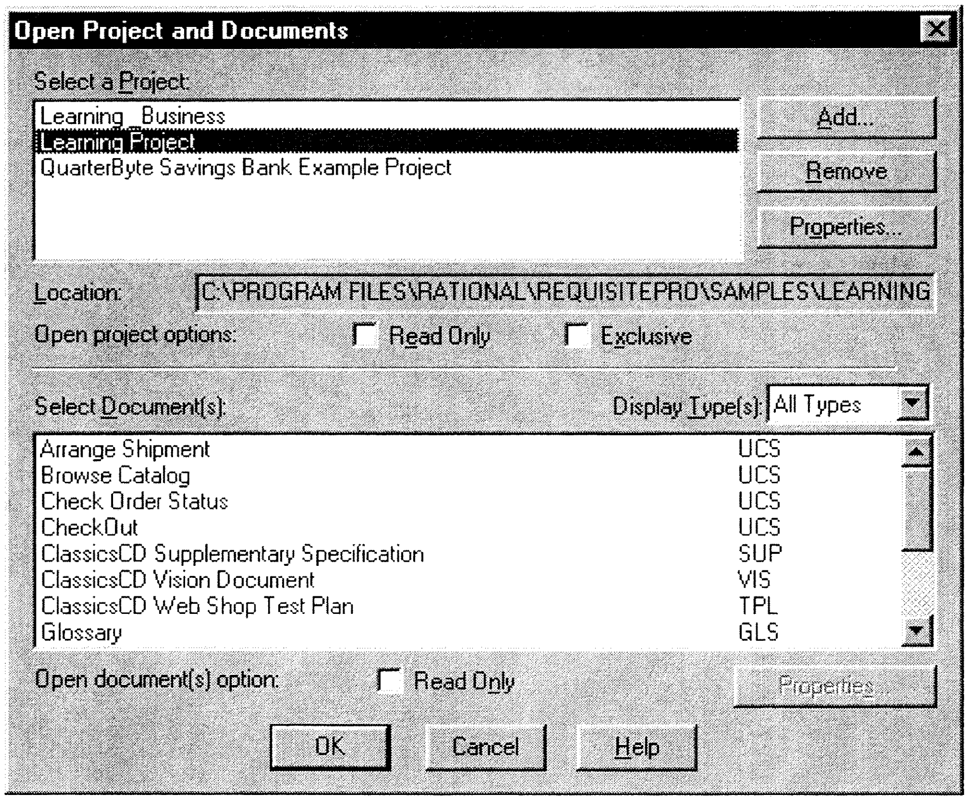
The first task a user needs to do is create a new project. Figure 2.42 shows the New Project window that appears following the selection of *New* under the *Project* drop down menu. The user can choose from a blank project or one of templates provided. The templates differ from traditional declarative requirements approach and follow a use case approach. Use cases are methodologies used in system analysis to identify, clarify and organize system requirements. A use case can be thought of as a collection of possible scenarios related to a particular goal. Each template choice creates a directory for the project with the necessary requirement files.



Use this template to create a blank project.

Figure 2.42 New Project Window.

Before any of the developers modify the project created, the project options must be viewed because only one person at a time can modify the requirement documents of a project. To be able to modify any of the project documents, the user must open the project file from the tool palette, shown in Figure 2.43, and check the *Exclusive* box. The window can also be used for opening requirement files.



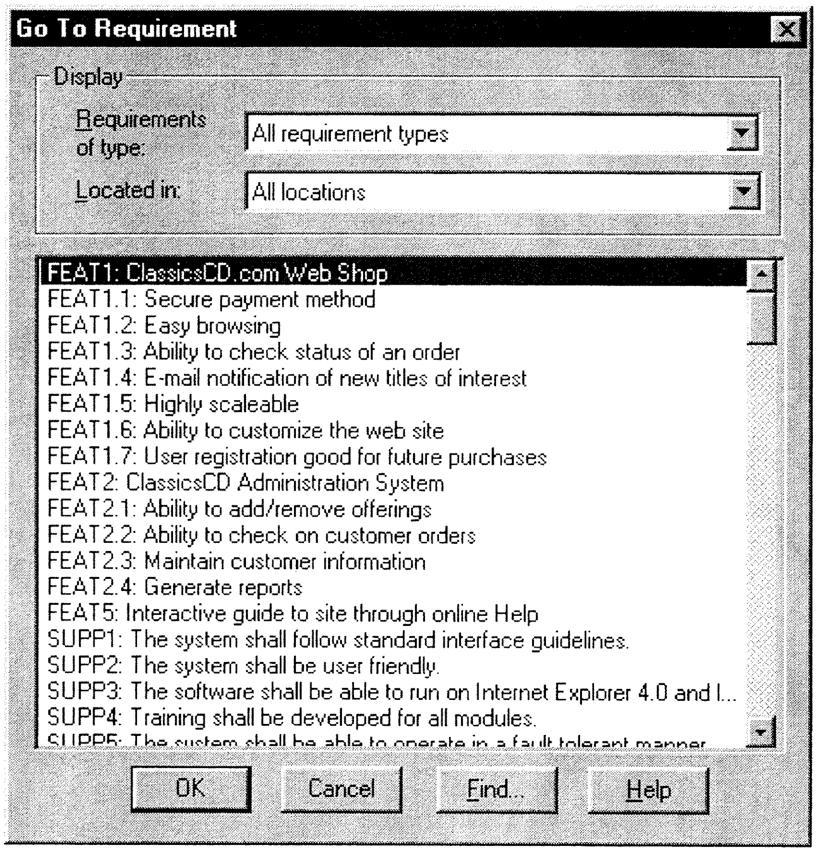
Learning _Business
Learning Project
QuarterByte Savings Bank Example Project

Location: C:\PROGRAM FILES\RATIONAL\REQUISITEPRO\SAMPLES\LEARNING

Arrange Shipment	UCS
Browse Catalog	UCS
Check Order Status	UCS
CheckOut	UCS
ClassicsCD Supplementary Specification	SUP
ClassicsCD Vision Document	VIS
ClassicsCD Web Shop Test Plan	TPL
Glossary	GLS

Figure 2.43 Project Contents Window.

Specific project requirements can be found by clicking on the *goto requirement button* on the tool palette. A requirements selection window will appear, as shown in Figure 2.44, where any of the projects requirements can be selected. Choosing a requirement will automatically open up the file which contains that requirements information.



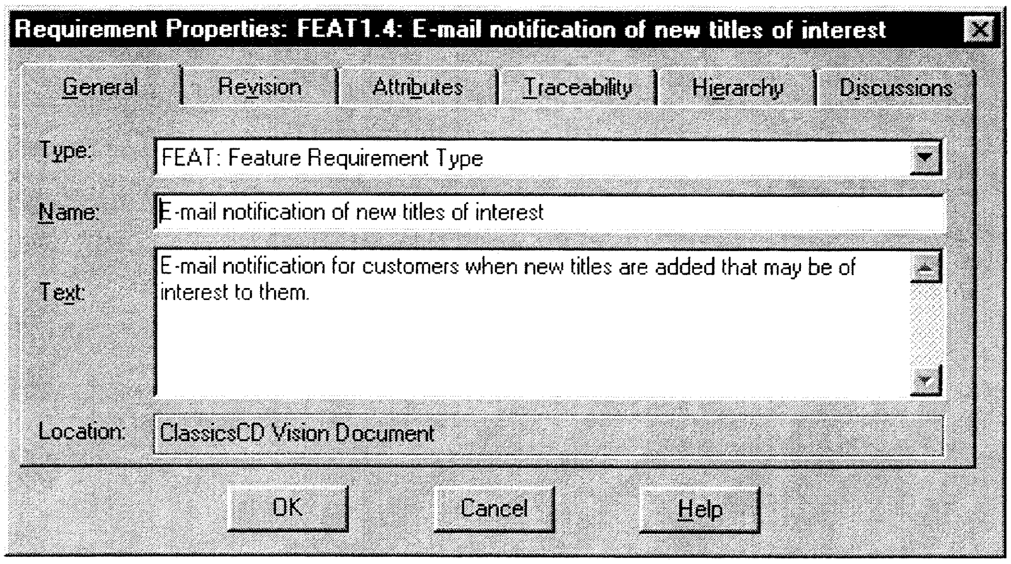
All requirement types

All locations

FEAT1: ClassicsCD.com Web Shop
FEAT1.1: Secure payment method
FEAT1.2: Easy browsing
FEAT1.3: Ability to check status of an order
FEAT1.4: E-mail notification of new titles of interest
FEAT1.5: Highly scaleable
FEAT1.6: Ability to customize the web site
FEAT1.7: User registration good for future purchases
FEAT2: ClassicsCD Administration System
FEAT2.1: Ability to add/remove offerings
FEAT2.2: Ability to check on customer orders
FEAT2.3: Maintain customer information
FEAT2.4: Generate reports
FEAT5: Interactive guide to site through online Help
SUPP1: The system shall follow standard interface guidelines.
SUPP2: The system shall be user friendly.
SUPP3: The software shall be able to run on Internet Explorer 4.0 and l...
SUPP4: Training shall be developed for all modules.
SUPP5: The system shall be able to operate in a fault tolerant manner

Figure 2.44 Requirement Selection Window.

Once a specific requirement is chosen, its properties, shown in Figure 2.45, can be viewed by clicking on the properties icon on the tool palette. Requirement properties include: revision dates, attributes (priority, status, etc.), traceability (list of actions performed on the requirement), hierarchy (shows any dependent requirements) and related discussions.


Requirement Properties: FEAT1.4: E-mail notification of new titles of interest
General
Revision
Attributes
Traceability
Hierarchy
Discussions
OK
Cancel
Help

Type: FEAT: Feature Requirement Type

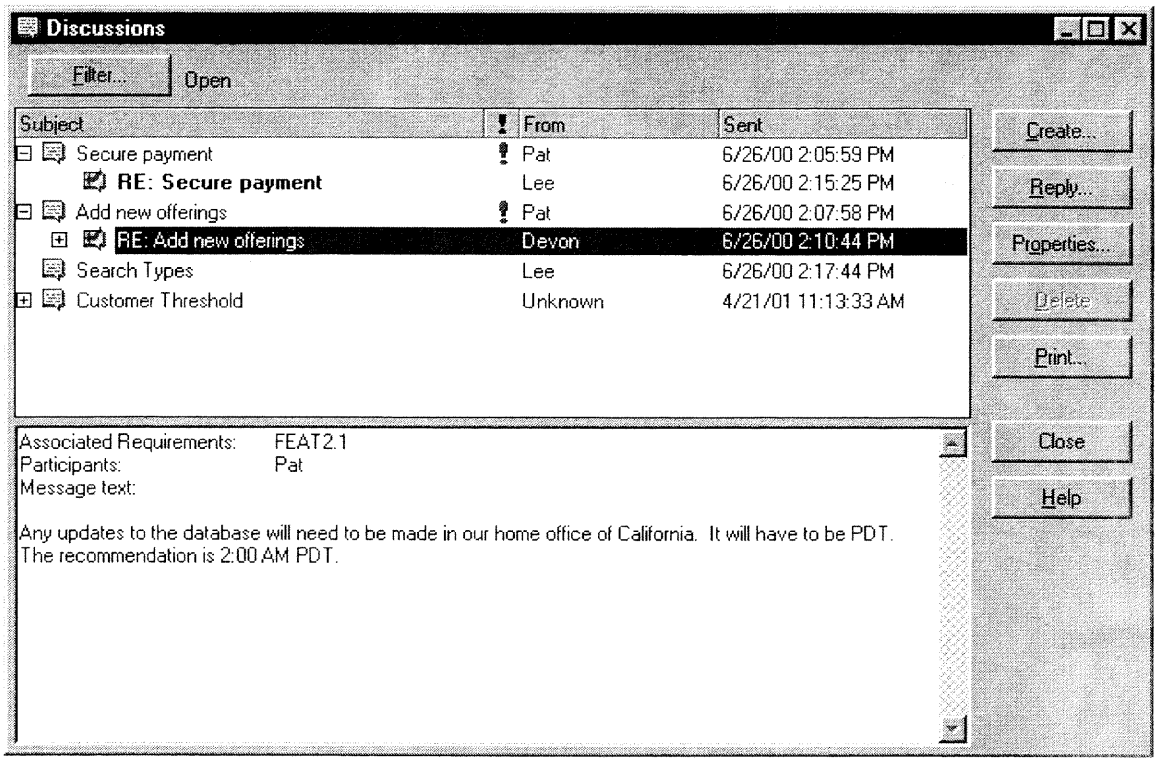
Name: E-mail notification of new titles of interest

Text: E-mail notification for customers when new titles are added that may be of interest to them.

Location: ClassicsCD Vision Document

Figure 2.45 Requirement Properties Window.

Requisite Pro also contains an asynchronous discussion tool shown in Figure 2.46. This tool is activated from the Tool Palette. Users can also view the properties of the messages posted by clicking on the properties button. Properties include: message attributes (time, date, priority), discussion participants, and requirements under discussion.



Subject	!	From	Sent
Secure payment	!	Pat	6/26/00 2:05:59 PM
RE: Secure payment		Lee	6/26/00 2:15:25 PM
Add new offerings	!	Pat	6/26/00 2:07:58 PM
RE: Add new offerings		Devon	6/26/00 2:10:44 PM
Search Types		Lee	6/26/00 2:17:44 PM
Customer Threshold		Unknown	4/21/01 11:13:33 AM

Associated Requirements:	FEAT2.1
Participants:	Pat
Message text:	

Any updates to the database will need to be made in our home office of California. It will have to be PDT. The recommendation is 2:00 AM PDT.

Figure 2.46 Discussion Window.

Requisite Pro Technology Analysis

Requirement management is an important task when developing software. Requisite Pro essentially provides 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.

Requisite pro represents an extensive application to manage project requirements and as such is a key component of a collaborative software development model.

2.7.7.2 Rational Unified Process Case Study. The Rational Software Company offers a multitude of software development lifecycle products. There are also products that assist in the development, testing and documentation of software applications. The next example will illustrate of the Rational Unified Process (RUP) application. In contrast to Rose, The RUP is a groupware product.

RUP is a generic web-based software engineering process that provides a framework to assign and manage *tasks* and *responsibilities* within a development organization. RUP attempts to enhance team productivity by delivering what Rational calls “software best practices” to all team members.

Figure 2.47 shows the dimensions of the Rational Unified Process: a horizontal axis represents the lifecycle of the process; a vertical axis represents the core process workflow. The graph shows how the emphasis of the workflows varies over time. In the early stages, most of the time is spent on requirements; and in later stages, more time is spent on configuration and change management.

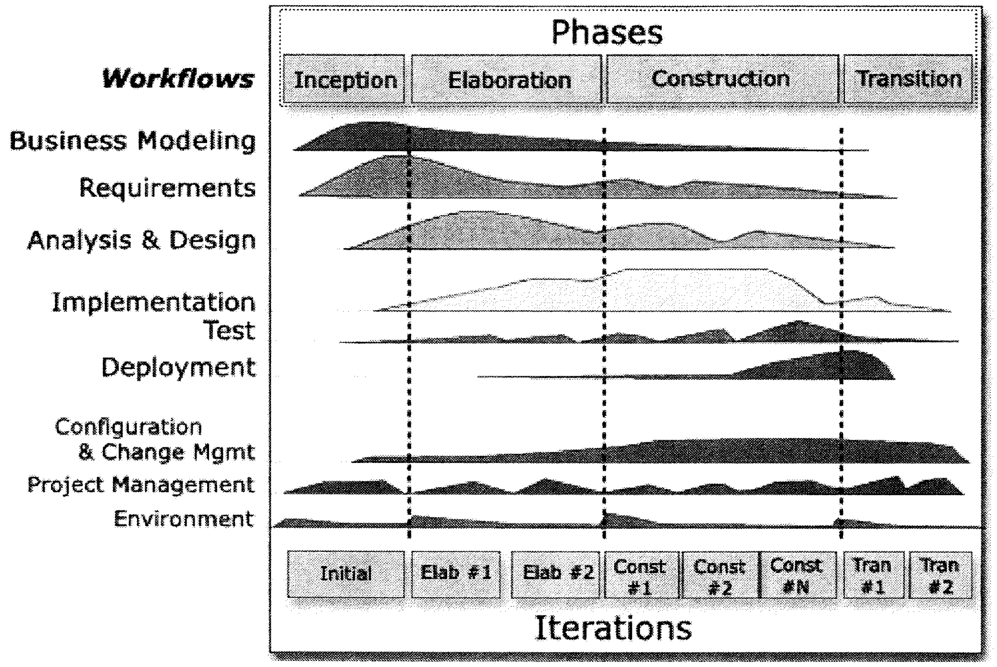
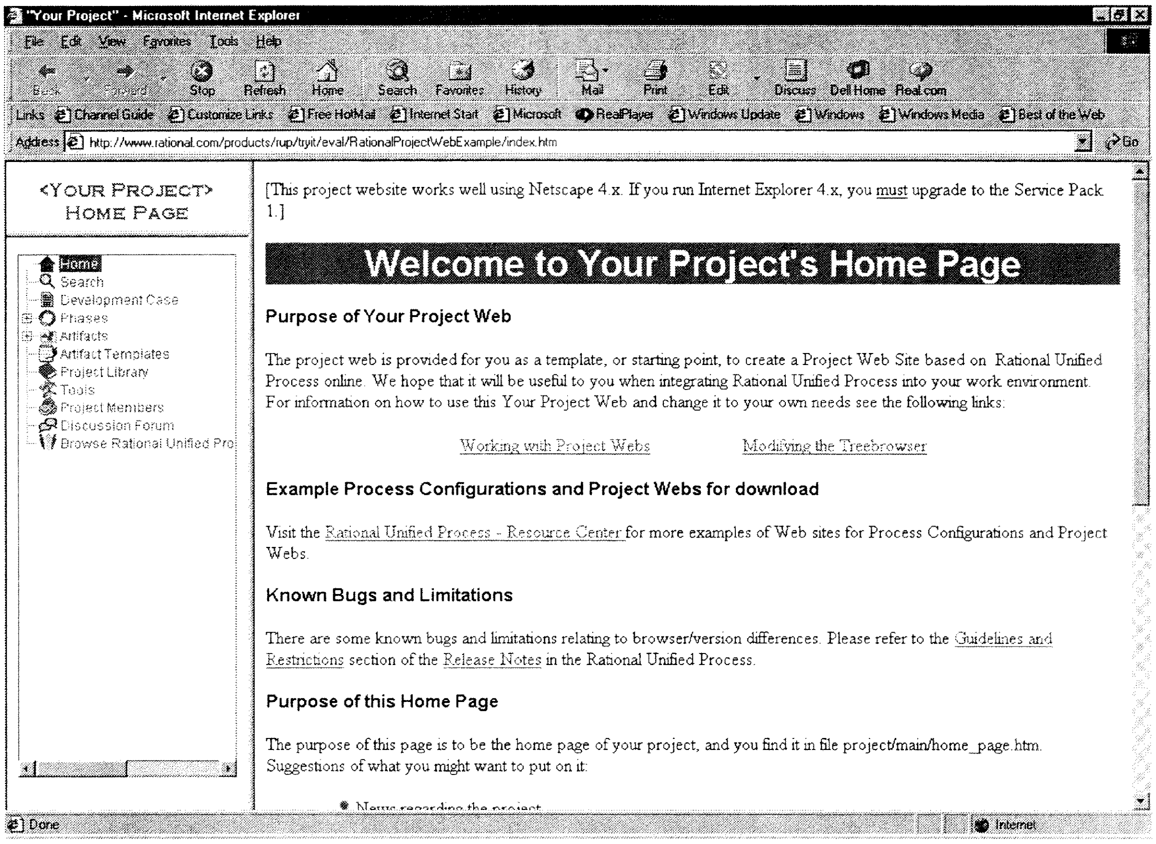


Figure 2.47 RUP architecture.

An Example Solution

A template web site is the starting point for any project and shown in Figure 2.48. The template helps a team maintain project information, facilitates project team communication, and initiates a web-enabled central project "knowledge" repository.


"Your Project" - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Dell Home Real.com
Links Channel Guide Customize Links Free HotMail Internet Start Microsoft RealPlayer Windows Update Windows Windows Media Best of the Web
Address http://www.rational.com/products/rup/tryit/eval/RationalProjectWebExample/index.htm
Go
<YOUR PROJECT> HOME PAGE
Home
Search
Development Case
Phases
Artifacts
Artifact Templates
Project Library
Tools
Project Members
Discussion Forum
Browse Rational Unified Pro
Done
Internet

[This project website works well using Netscape 4.x. If you run Internet Explorer 4.x, you must upgrade to the Service Pack 1.]

Welcome to Your Project's Home Page

Purpose of Your Project Web

The project web is provided for you as a template, or starting point, to create a Project Web Site based on Rational Unified Process online. We hope that it will be useful to you when integrating Rational Unified Process into your work environment. For information on how to use this Your Project Web and change it to your own needs see the following links:

Working with Project Webs   Modifying the Treebrowser

Example Process Configurations and Project Webs for download

Visit the Rational Unified Process - Resource Center for more examples of Web sites for Process Configurations and Project Webs.

Known Bugs and Limitations

There are some known bugs and limitations relating to browser/version differences. Please refer to the Guidelines and Restrictions section of the Release Notes in the Rational Unified Process.

Purpose of this Home Page

The purpose of this page is to be the home page of your project, and you find it in file project/main/home_page.htm. Suggestions of what you might want to put on it:

- News regarding the project

Figure 2.48 Main web page of example project.

The tree structure, shown in the far left of Figure 2.48, can be modified to meet the needs of the project. It defaults to the following items: development case, phases, artifacts, artifact template, project library, tools, directory of project members, discussion forum and an over view of RUP.

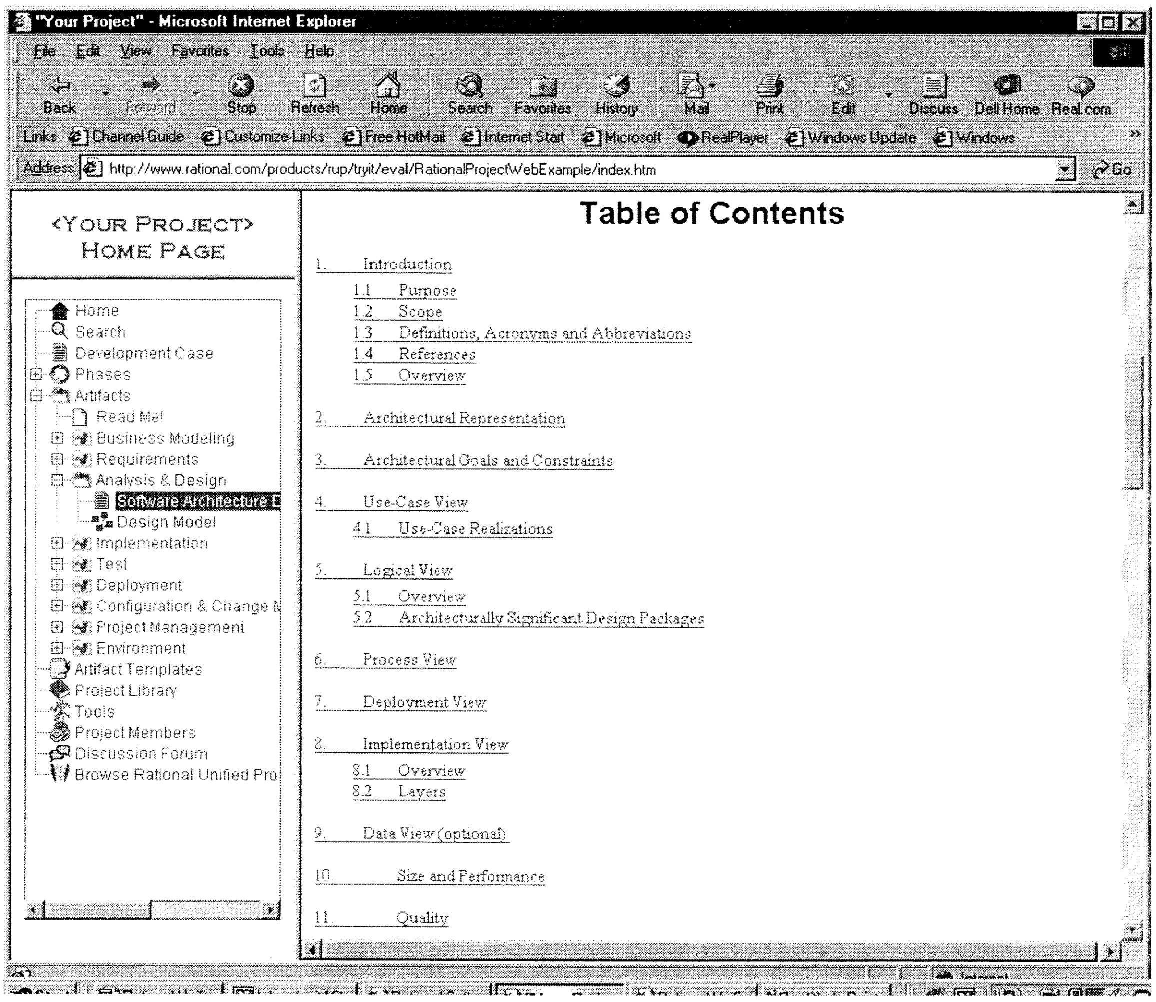
The *development case* link provides a template to briefly describe descriptions of the project milestones and their purpose. All projects should start by defining a "development case", which is viewed as a high-level project plan that describes the artifacts that will be produced in this project and the level of formality.

A *Phases* link can be modified to describe what will occur during the various RUP phases, i.e. inception, elaboration, construction, and translation, as depicted in

Figure 2.47 above. The *inception* phase's main goal is to ensure the project is worth doing and feasible. This phase consists of the following activities: formulating the scope of the project, planning and preparing a business case, feasibility analysis, preparing the environment for the project. The goal of the *elaboration* phase is to develop a solid base for the design and implementation effort in the construction phase. Elaboration activities include: defining, validating and baselining the software architecture, refining the vision, creating and baselining detailed iteration plans for the construction phase, refining the development case and putting in place the development environment, and refining the architecture and selecting components. The *construction* phase's main goal is to clarify the remaining requirements and complete the development of the system based upon the baselined architecture. Activities for this phase include: resource management, control and process optimization, complete component development and testing against the defined evaluation criteria, and assessment of product releases against acceptance criteria for the vision. The *translation* consists of several iterations to make sure it is what the end users requested. This is accomplished by executing deployment plans, finalizing end-user support material, testing the deliverable product at the development site, creating a product release, getting user feedback, fine-tuning the product based on feedback, making the product available to end users.

The *artifacts* link is where all of the documents related to the project can be stored. Templates are provided for all types of project related documents. For example, Figure 2.49 shows the table of contents of the software architecture artifact. Each item in the table of contents is a link to that section in the document.

Each section has explicit information regarding what information should be contained in that section.


"Your Project" - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Dell Home Real.com
Links Channel Guide Customize Links Free HotMail Internet Start Microsoft RealPlayer Windows Update Windows
Address http://www.rational.com/products/rup/tryit/eval/RationalProjectWebExample/index.htm
Go

<YOUR PROJECT> HOME PAGE

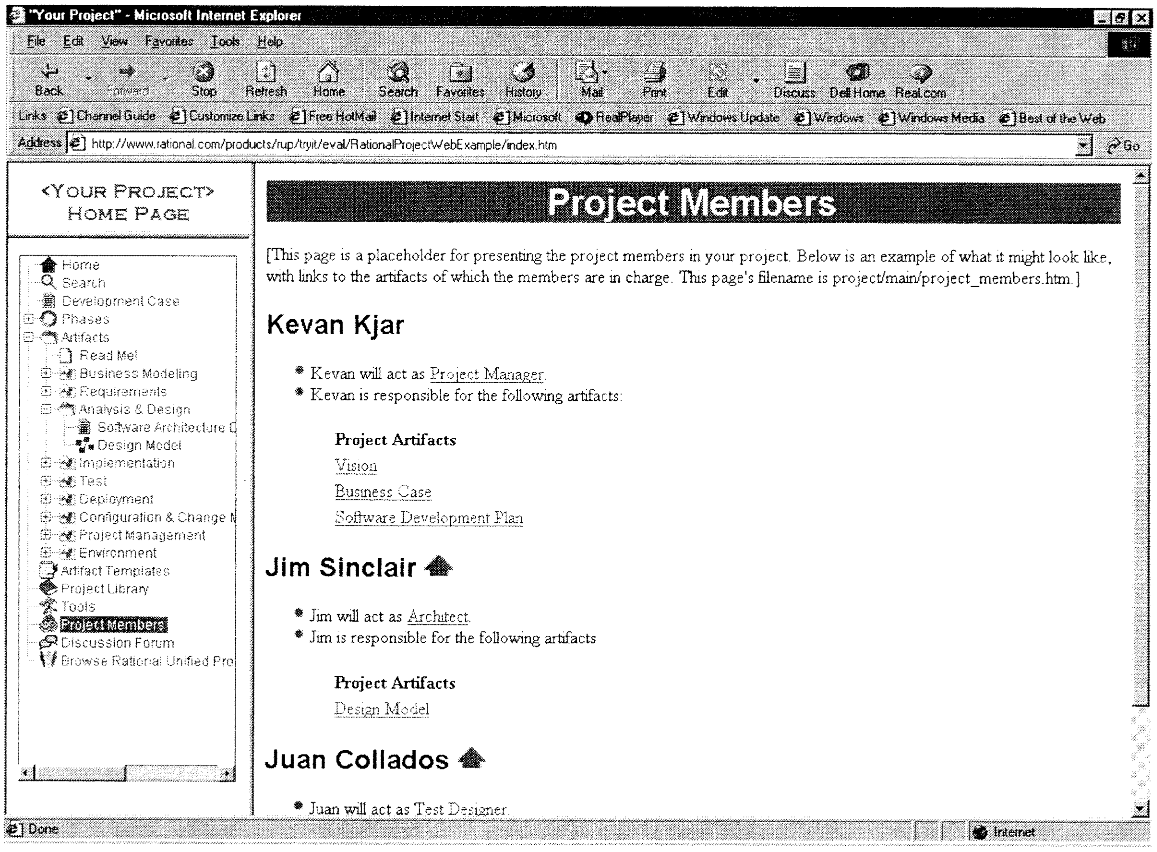
- Home
- Search
- Development Case
- Phases
- Artifacts
  - Read Me!
  - Business Modeling
  - Requirements
  - Analysis & Design
    - Software Architecture D
    - Design Model
  - Implementation
  - Test
  - Deployment
  - Configuration & Change M
  - Project Management
  - Environment
- Artifact Templates
- Project Library
- Tools
- Project Members
- Discussion Forum
- Browse Rational Unified Pro

Table of Contents

Figure 2.49 Software Architecture Artifact Template.

The *project library* link is a place to store any project related papers or articles that would be useful to the team members such as white papers, experience reports on tools or techniques, market surveys, and interview material from requirements capturing. The *tools* link is a placeholder for collecting information on any tools you are using in your project. The information could include links to online manuals or tutorials.

The *project member's* link is where the team members are listed along with their role and responsibilities. A major factor in the successful software development is the model or process followed during development. An efficient process can save development and debugging time. One key aspect of the Rational Unified Process is that it maps the people on a development team to specific roles. One physical person can have the responsibility of many roles. Figure 2.50 shows is an example of a *project member's* link when using the RUP web model.


"Your Project" - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss Dell Home Real.com
Links Channel Guide Customize Links Free HotMail Internet Start Microsoft RealPlayer Windows Update Windows Windows Media Best of the Web
Address http://www.rational.com/products/rup/tryit/eval/RationalProjectWebExample/index.htm
Go
<YOUR PROJECT> HOME PAGE
Home
Search
Development Case
Phases
Artifacts
Read Me!
Business Modeling
Requirements
Analysis & Design
Software Architecture D
Design Model
Implementation
Test
Deployment
Configuration & Change M
Project Management
Environment
Artifact Templates
Project Library
Tools
Project Members
Discussion Forum
Browse Rational Unified Pro
Done
Internet

Project Members

[This page is a placeholder for presenting the project members in your project. Below is an example of what it might look like, with links to the artifacts of which the members are in charge. This page's filename is project/main/project_members.htm.]

Kevan Kjar

- Kevan will act as Project Manager.
- Kevan is responsible for the following artifacts:

Project Artifacts
Vision
Business Case
Software Development Plan

Jim Sinclair

- Jim will act as Architect.
- Jim is responsible for the following artifacts

Project Artifacts
Design Model

Juan Collados

- Juan will act as Test Designer.

Figure 2.50 Example Project Members link.

The Roles include analysts, developers, testers, and managers. The example project shown in Figure 2.50 utilizes the project manager, architect, and test designer. The project manager role allocates resources, shapes priorities, coordinates interactions with customers and users, and generally keeps the project team focused on the right goal. The project manager also establishes a set of practices that ensure the integrity and quality of project artifacts. The software architect role leads and coordinates technical activities and artifacts throughout the project. The software architect also establishes the overall structure for each architectural view: the decomposition of the view, the grouping of elements, and the interfaces between these major groupings. Therefore, in contrast to the other roles, the software architect's view is one of breadth as opposed to one of depth. The test designer is the principal role in testing and is responsible for planning, designing, implementing, and evaluating the test, including: generating the test plan and test model, implementing the test procedures, evaluating test coverage, results, and effectiveness, and generating the test evaluation summary

The last two items in the RUP project web site are the *discussion forum,* which is where links to sub forums can be found, and the overview of the RUP link. The overview brings up a separate window with links to useful information regarding the RUP.

RUP Technology Analysis

The RUP attempts to unify the entire software development team by providing each team member with one knowledge base, modeling language and view of how to develop software.

The RUP does coordinate many activities of the software development process. However, the process does miss some key factors when collaborating during software development. For example, the web site is a good forum for coordination of project documents but there are no facilities for brainstorming, voting, negotiation, conflict resolution etc. RUP does have a discussion forum placeholder in the website but it is up to the team to develop or integrate a message board or chat room.

2.7.8 Together

Together, (TogetherSoft Corporation, 2000), is a team software development platform for building software solutions. It focuses on synchronizing models, code and documents created while developing software. It is possible to integrate Together with most leading version control applications. Together can run on a server where all users access the server installation's global configuration settings that merge with individual settings at runtime, or it can run on individual workstations where users share a centralized set of global configuration properties that merge with individual settings at runtime.

Together has several features that enable teams to use it not only for modeling and documentation, but for actual implementation coding as well. Features include: an editor for multiple languages, a debugger and compiler for Java, code generation, syntax highlighting, auto-indent etc. Together's integrated debugger enables you do work in conjunction with the Editor. The debugger also features multi-threads, watches, and breakpoints.

Together Technology Analysis

Together provides support for multi-user implementation, but it does not provide tools for collaboration during the design stages of development such as problem formulation. The support for multi-user development is a result of the team using the same language, diagrams and "building block" components. Essentially the team members share a development space, which requires a group to integrate a version control application with the Together application. Individual team members can "check out" files from version control to see what other team members have worked on. There is no brainstorming or synchronous exchange of ideas or any organized task tracking tool.

2.7.9 WikiWeb

WikiWeb is a collaboration tool that operates through a web browser. The tool is essentially a server that hosts a website which can be modified and instantly published. In other words, web pages are automatically created and linked to one another. The main features, in addition to instant web publishing, are file sharing, page change notifications via e-mail, controlled user access and privileges, page indexing, and full text search.

WikiWeb Details

WikiWeb reads and navigates like a standard web page. WikiWeb pages each have an *edit* link. Users can *save* changes, which are then instantly published, and easily create links to new pages including pictures, files, and e-mail addresses. This tool can be useful in brainstorming.

WikiWeb Technology Analysis

WikiWeb is ideal for a collaborative project that is does not need to follow a stringent framework. It could be adapted to include a framework. It is an interesting collaboration tool, but the group leader needs to set up the website to coordinate the collaborative efforts.

2.7.10 Summary of Collaborative Tools

Table 2.2 is a comparative analysis of each tool reviewed in this section.

Table 2.2 Summary of Groupware Tools

	Groove	CyberCol-Collaboratory	Visual Source Safe	Req-Pro	Doors	Group System	Lotus Notes	Together	Wiki-Web	Rose	RUP
Brainstorm Tool	✓	✓				✓					
Voting Tool		✓				✓					
Document Storage Tool	✓	✓				✓	✓	✓	✓		✓
Idea Org. Tool		✓									
Member Contact Tool	✓					✓	✓				✓
Task List Tool	✓					✓	✓				
Scheduler Tool (Calender)	✓						✓				
Message Board/ Discussion Board Tool/ Asynch. Message Tool	✓	✓							✓		
Chat/Sync. Message Tool	✓	✓									
Link Tool	✓										
E-mail Tool											
Project Manage/ Req. Manage Tool				✓	✓						
Code Database Tool			✓								

2.8 General Groupware Tools

Representative general groupware systems not specifically designed for problem-solving or collaborative software development include: EIES, Virtual-U, Learning Space, WebCT, Co-Mentor, Colloquia, and TopClass. A synopsis of each of these systems follows.

The Electronic Information Exchange System **(EIES)** developed at NJIT is a groupware system originally intended for computerized conferencing, but later enhanced to support distance learning. EIES provides a number of interesting features, including hierarchically organized discussion threads. One objective of later versions of EIES was to *improve,* rather than merely attempt to emulate, the activities of a face-to-face classroom (Turoff, 1995). For example, EIES provided sophisticated question and response capabilities, as well as a variety of group learning tools including mechanisms to track student participation in group activities.

Virtual-U provides a framework for designing, delivering and managing courseware (Harasim, 1999) and integrates conferencing, chat, and performance evaluation tools using an underlying web environment. The system features e-mail, file exchange, multimedia applications, an announcements area, asynchronous discussion groups, as well as a detailed help system that supports course design.

An asynchronous, distributed learning environment, based on Lotus Notes, is provided by **Learning Space**. The system supports scheduling, provides a course material database, threaded discussions, user profiles, and a mechanism for user feedback, as well as standard features such as e-mail, announcements, file exchange, chat, whiteboard, and video conferencing.

WebCT (World Wide Web Course Tool) is a system that facilitates developing web-based courses (Goldberg, 1997). It provides a conferencing system, chat, progress tracking, an announcements area, file exchange, e-mail, timed quizzes, asynchronous discussions, whiteboard, and search capabilities. The system provides course designers with the ability to modify the look and feel of courses they implement on the system.

An on-line course environment that allows participants to collaborate by providing synchronous and asynchronous discussion capabilities, e-mail, file exchange, databases of previous work, and an announcements board is provided by **CoMentor** (Hepplestone, 2000). The objective of the system is to supplement existing off-line courses rather than provide self-contained on-line courses.

Colloquia (or Learning Landscapes) is a software system that supports group work. Colloquia is distributed, that is each individual receives a private copy, unlike most client server collaborative systems, allowing users to work independently off-line. Colloquia provides both asynchronous group communication as well as person-to-person conversation facilities and file exchange.

2.9 Groupware Tools for Problem-Solving and Software Development

The focus of this section is on tools or systems that support problem solving or software development tasks, and specifically groupware tools that facilitate the kind of communication and coordination needed when a group collaborates on problem solving or software development. Such environments differ from environments that support individually-based problem solving/software development activities.

The collaborative characteristics of each tool will be identified. The number of tools available is small since the technology needed to appropriately implement computer-aided collaborative environments is relatively new. A selective illustrative review follows.

2.9.1 Collaborative Problem Solving Tools

Group problem solving occurs in many formats, a particularly useful one being the face-to-face group meeting. On-line versions of meetings are usually called *conferences*. Thus, the tools for collaboration in meetings and groupware that assists problem solving in a group setting will be considered. One interesting fundamental presentation requirement of such meeting tools is that they provide a coordinated interface for all participants (Stefik et. al., 1987), an abstraction frequently to by the acronym *WYSIWIS*: "What <u>You</u> See Is What <u>I</u> See." The same researchers also claim that for a meeting tool to be effective, every member of the group must be able to view the work of the other group members. In addition to providing such coordinated interfaces or views, meeting tools can also provide capabilities for brainstorming, organizing, and meeting evaluation. Descriptions of various such meeting or conference tools follows.

A simple example of an early conferencing tool that allows both asynchronous and synchronous modes of user interaction is **We-Met**, a basic collaborative graphical editor (Rhyne, 1992). The conference participants work independently and asynchronously, then synchronously broadcast their work to the group. One elementary advantage of having both synchronous and asynchronous capabilities is that participants can enter a group after work has begun and catch up by reviewing messages that occurred before they joined. The We-Met system is *non-anonymous*: users are explicitly associated with their work. Furthermore, a history of all individual work actions is maintained.

An example of a cooperative design environment that focuses on detection and resolution of design conflicts, a critical issue in any cooperative design process, is the **Design Collaboration Support System** DCSS (Klein, 1994). DCSS's major features are: it facilitates design agents in expressing their design actions; it assists in detecting design conflicts; and it suggests potential resolution to design conflicts detected. This kind of conflict resolution is called *domain level conflict* and refers to inconsistencies in design criteria. It of course differs from the phenomenon of *collaborative conflict*, which refers to interpersonal conflict between members of a collaborative group.

An example of an asynchronous instructional conferencing system designed to assist for engineering students is **CaMILE** (Collaborative and Multimedia Interactive Learning Environments) (Guzdial et.al, 1996). Typical problems addressed by Camille are small-scale engineering design problems such as might occur in a college-level structural design course. The system prompts users to discuss alternative approaches to designs already suggested by other group members. The system also links to related multi-media documents, allowing users to share access to a design database and resources. Projects developed under the system are archived in a database and accessible to subsequent groups. Another rudimentary domain-specific system designed to enhance problem-solving and diagnostic reasoning skills in elementary biology is **Biology Sleuth**, developed by Denning and Smith (1995). The system comprises a database of domain-specific facts and a graduated instructional environment, but uses little computer-mediated collaboration.

A web-based system that supports collaborative information sharing for document development and cooperative work is the Basic Support for Cooperative Work project (**BSCW**) (Bentley et. al., 1995). The system is asynchronous and supplies an application infrastructure that runs on a basic web server.

2.9.2 Collaborative Software Development Environments

A preliminary point of reference is **SOLVEIT** (Deek, 1997), an individually oriented environment designed to develop problem solving skills in the domain of elementary software development. SOLVEIT has a strong methodological foundation, being systematically based on an underlying cognitive model for both problem-solving and program development, the Dual Common Model (refer to section 1.1). An individual using SOLVEIT begins with a preliminary verbal problem description and is then interactively guided through a sequence of problem-solving stages, ranging from preliminary problem clarification and modeling to final testing of a software implementation.

A synchronous collaborative system intended to allow geographically dispersed participants to work jointly on a large programming project is provided by **Computer-Supported Cooperative Training** (CSCT) (Swigger, Brazile & Shin, 1995). The primary objective of the system is to train novices how to collaborate when designing software. The context used to develop collaborative skills is requirement elicitation, the outcome being a requirements document for a software problem. There are four shared tools: a Procedural Activity which is used to establish operating procedures via a voting system; a Problem Definition Activity which is used to specify an agreed-on problem statement; a Criteria Establishment Activity which is used to develop criteria for requirements; and a Solution Activity which is used to prioritize requirements via the voting tool. Each of these tools can be used at any point during the collaboration to identify the software requirements of the problem.

A web-based collaborative, software development system is provided by the **Karell++ Collaborative Laboratory**. This system has both synchronous and asynchronous collaboration capabilities, and enables participants to collaborate in real time. The specific purpose of the system is to develop the participants' skills in object-oriented programming techniques. The system provides a shared development environment for programs development, written in a domain-specific language Karell++ designed for rudimentary robotic applications (Rossi, 1999). Participants design programs that simulate robots using component-based program elements, and do program verification on a graphical simulator.

Another collaborative tool designed to support software development is **EVA**, an acronym for "Evolving Artifact". This system allows developers and end-users to iteratively understand design problems and to develop solutions. The system is based on the construction and refinement of so called design representations (Ostwald, 1995). EVA utilizes a hypertext environment that lets end users view and interact with design prototypes that they can annotate with comments, the underlying expectation being that access to a rich set of prototypes and documents deepens problem understanding.

A more sophisticated environment for collaborative work, flexible enough to adapt to both different software development processes as well as to very different collaborative domains is provided by **Conversation Builder** (Kaplan et. al., 1992). The Conversation Builder environment emphasizes the collaborative character of work activities, allows individual tasks to be multitasked, recognizes that simultaneous activities are typically dependent on each other, and that tasks usually have associated actions to perform. In addition to providing an architecture that supports these characteristics, the system also provides messaging capabilities, version management, negotiation activities, shared data modules, and the ability to dynamically interconnect activities.

A collaborative environment for software development that emphasizes integrating the semantics of the software development domain with the characteristics of group work, supporting strategies for negotiating problems via social-based argumentation, and using contracts as a way to ensure responsibility for task fulfillment called **CoNeX** was developed by Hahn, Jarke, and Rose (1990). CoNeX includes an argument editor for negotiation, a contract manager for document dialog, and a conferencing system for informal messaging. Users can also browse a knowledge base to trace the project history. The tool is reminiscent of **Wong's** (1994) model (section 1.2) for joint decision making and conflict resolution.

An asynchronous groupware tool for problem solving that assists in collaborative work among geographically distributed participants called **Web-CCAT** was developed and tested at the University of Illinois (Dufner, Kwon, & Hadidi, 1999). The tool consists of project management software, GDSS tools, and computer-aided software engineering CASE tools, CASE being the generic term for tools that aid software engineers develop and maintain software. The goal of the tool is to provide a more enriched environment than face-to-face meetings.

2.9.3 Summary of Problem Solving and Software Development Tools

Some characteristics of the collaborative problem solving and software development tools reviewed are summarized below in Table 2.3 using the Simon (1997) and Hohmann (1997) models utilized in Table 2.1. The Table identifies whether a tool facilitates any of the activities identified in the models reviewed in Table 2.1. The headings for Table 2.3 are discussed in section 2.2, though the behavior modification column has been deleted.

Table 2.3 Summary of Collaborative Problem Solving Tools

Tool Name	Id Tasks	Distribution of Tasks	Coordinating Outcomes	Integrating Solutions	Plan Development	Communication Plan
WeMet						Asynchronous and Synchronous messaging capabilities.
DCSS			Conflict resolution feature.			Asynchronous messaging capabilities.
CaMILE					Contains a database of designs to assist in developing a solution plan.	Asynchronous messaging capabilities.
Biology Sleuth						Asynchronous messaging capabilities.
BSCW					Contains a database of designs to assist in developing a solution plan.	Asynchronous messaging capabilities.

Table 2.4 summarizes the software development tools, again in terms of the Simon (1997) and Hohmann (1997) models. An additional “Tool Type” heading has been added to indicate whether the tool is intended for individual or for collaborative development.

Table 2.4 Summary of Software Development Tools

Tool Name	Tool Type	Id Tasks	Distribute Tasks	Coordinating Outcomes	Integrating Solutions	Plan Development	Communication Plan
SOLVEIT	Individual	Assists user in breaking problem into sub-problems and tasks.	N/A	N/A	N/A	N/A	Asynchronous messaging capabilities.
CSCT	Group			Negotiation Voting Capabilities			Synchronous messaging capabilities.
Karell++	Group						Asynchronous and synchronous messaging capabilities.
EVA	Group					Contains a database of designs to assist in developing a solution plan.	Asynchronous messaging capabilities.
CB	Group						Synchronous messaging capabilities.
CoNeX	Group			Negotiation Voting Capabilities		Contains a database of designs to assist in developing a solution plan.	Asynchronous and synchronous messaging capabilities.
Web-CCAT	Group						Asynchronous messaging capabilities.

The following section will analyze and critique existing tools and methodologies to help identify open areas for future research in the software engineering related collaborative problem solving and software development.

2.10 Critique of Existing Approaches

The purpose of this section is to identify deficiencies and/or open areas of current theory and tools in collaborative problem solving and software development. The results of this evaluation will be to help delineate an appropriate direction for advancing the state of the art.

2.10.1 Functional Weaknesses and Practical Deficiencies

This section will highlight the functional weaknesses and practical deficiencies of the models and tools presented in the previous chapter. The models will be evaluated to determine if they take into consideration the skills needed to solve problems collaboratively. The tools will be evaluated to determine if they address the needs of collaborative software development and collaborative problem solving.

2.10.1.1 Models. Table 2.1 summarized collaborative problem solving and decision making models, categorizing the qualities a successful model should possess. The first model summarized is Hohmann's (1997) that focused on the communication and collaborative aspects of the problem solving process. The model recognizes the dynamic character of group communications and that it must account for the fact that group communication changes whenever its membership changes. It does not focus on the group character of the problem solving process, assuming instead that individual problem solving models suffice. The model also appears to not address conflict resolution.

Simon (1997) presents a collaborative decision making model. The model stresses the need for group-level understanding of the problem and for the development of a group plan. Each of these objectives facilitates [and requires] coordination. The model does not address the distribution of tasks to individual team members, a requirement that is critical in complex problems.

The model of **Finnegan and O'Mahony** (1996) has a number of features that support group activities, but it does not appear to adequately address coordination activity. For example, it lacks activities that partition a problem and determine the assignment of team members to work on particular parts of the partitioned problem, such as provision of a group leader for coordination. The model also does not address conflict resolution, which would be particularly relevant in the context of this model, which does address the negotiation required when determining alternative solutions. Guidelines for team interaction should also be included in such a collaborative problem solving model. Their omission is significant because if group activities and cognition are not properly channeled, inappropriate interaction among members can undermine the steps involved in the iterative process proposed in the model.

Wong's (1994) group problem solving model focuses on conflict resolution, and includes explicit negotiation attributes as part of the methodology. However, the method does not include a framework for coordination of activities between group members, or stress the iterative character of problem solving. The model also does not address issues of team interaction.

The model of **Zhang** (1998) reflects his supposition that collaborative problem solving should not entail an explicit sequence of steps. Zhang's methodology identifies four elements that should be included when collaboratively solving a problem:

1. Consider the individual group members as a distributed representation system.
2. Explicitly, decompose the problem into individual tasks for each group member.
3. Identify the individual task interactions.
4. Emphasis on interactions among individual group members and their tasks.

The elements, like some of the previously described models, include partitioning the problem into individual representations. The model does not address coordination among group members.

The **PBL** method does explicitly emphasize the need for a communication and coordination framework, but doe not address detailed guidance or direction of the participants. It would be advantageous to include a more explicit step-by-step process. The method has had some success in software engineering instruction (McCracken and Waters, 1999), though the experiment did identify a lack of focus on tasks by participants. The proposed remedy was to provide monitoring of group discussions with the objective of explicitly increasing coordination.

Despite their limitations, these methodologies do highlight both the usefulness of group problem solving, as well as the need for participants to invest time to understand group dynamics. Merely grouping participants and expecting them to develop an application collaboratively does not imply they will effectively solve problems as a group, or that the group will appropriately address either conflict resolution or consensus building.

2.10.1.2 Tools. Two types of tools have been considered for collaborative problem solving and for collaborative software development. Since problem solving is at the heart of software development, any collaborative tool for software development should incorporate problem solving methods.

Collaborative Problem Solving Tools

Refer to 2.2 for differences among the collaborative problem solving tools.

We-Met supports only a limited range of collaborative features: meeting discussions, brainstorming, and a history of the collaboration. Though it can be used for problem solving, it does not provide an explicit problem solving framework. All the collaborative tools depicted have asynchronous communication capabilities; some like We-Met have synchronous capabilities as well. Each mode has its advantages. The asynchronous mode allows group members to enter the problem solving session at their convenience by supplying all members a conference history. The synchronous communication allows real time discussion which expedites the problem solving process.

DCSS has a particularly interesting problem solving feature: namely, it can assist in detecting design conflicts. If a group is collaborating on a design and chooses an alternative design, the system helps detect problem with the alternative design. However, while supporting recognizing internal conflicts in a proposed design, the tool does not support interpersonal conflict resolution among group members as might occur during collaborative consideration of design alternatives. Like We-Met, DCSS provides no overall problem solving framework.

The **CaMILE** system provides an asynchronous problem solving tool that allows a user access to a database of resources, such as examples of designs provided by other users. The objective is to provide participants with previous cases to promote an understanding of decisions made by others in similar situations. The system lacks an overall problem solving framework. **Biology Sleuth** is not groupware in the true sense, but its methodology does require each member of a team to perform problem solving steps and to discuss their hypotheses with a group. Thus the tool uses a problem solving framework, but lacks true computer-mediated collaborative functionality.

The main purpose of **BSCW** is document storage and sharing. The system runs on a Web Server and so is available the internet with consequent advantages (Bentley et. al., 1995) such as platform, network and operating system transparency; integration with end-user environments and application programs, a consistent user interface across platforms; an application programmer interface to incorporate additional functionality; and ease of deployment facilitating rapid system prototyping. However, the system only provides asynchronous communication and does not include a specific problem solving methodology.

Collaborative Software Development Tools

Table 2.3 summarized the software development tools presented in this paper. This section considers the functional and practical deficiencies associated with these tools.

The first tool considered was **SOLVEIT** which was only to facilitate individual software development based on a cognitively-explicit, problem solving methodology, unlike other tools in Table 2.3. Some software development groupware also addresses problem solving, but none seems to explicitly address the very complex process by which groups solve problems.

CSCT is a collaborative tool that supports synchronous communication. The system is primarily designed to facilitate software requirements elicitation. This tool does not address the rest of the problem solving process. Its omission of an asynchronous communication capability is an obvious drawback.

The **Karell++** groupware tool enables both synchronous and asynchronous communication, but does not address problem solving. **EVA** also does not assist in problem solving or software development but provides a repository of design ideas that could assist during the design process. In other words **EVA** is an asynchronous group tool that has its only focus on sharing prototypes and developer documentation and comments.

CB is a system that has many features to allow collaboration and any other observed activities of group software development. However, there is no particular problem solving model associated with the application. The system boasts its "flexibility", but the collaborative process needs much more to produce an effective and efficient result.

CoNeX has many collaborative features a software development tool should encompass. It is however missing an important feature of a specific problem solving methodology to assist in the software development process. Therefore, the many facets of group cognition are not taken into consideration.

Web-CCAT is a tool that aids in developing software by providing CASE tools but is another example of an application not focusing on the problem solving portion of software development. Today's CASE tools minimize their support of the soft aspects of software development (Jarzabek and Huang, 1998). Soft aspects are defined as creativity understanding, idea generation, analogy, goals, emotions etc. Therefore, these tools do not assist in the problem solving effort. There are no means in the CASE tools available today for any freedom to express ideas or stimulate intuition.

The EVA system and the CSCT system are very useful in developing initial software requirements collaboratively. Using EVA most likely limits the possibility for "run away" projects by eliminating the change of user requirements early on in project development. This system seems to be beneficial for the initial development stages of a project. The iterative development process seems to be necessary for collaborative development. It emphasized the discovery of new goals, the role of prototyping and evaluation, and the importance of involving diverse stakeholders – including users (Carroll, 1997). The functional weakness is that there is no problem solving support in any of these collaborative software development tools. The implementation of the early problem solving steps should happen well before the elicitation of software requirements.

Out of the groupware tools presented in this paper CoNeX, does address the social needs of software development, but does not incorporate group cognition and psychology into its methodology. Group cognition should not be minimized in the problem solving process. Brereton (2000) found that during group projects it seemed that their problems were group cohesion, an aspect of group cognition, which to those writers was minor, but in a larger more realistic project lack of group cohesion for one reason or another could mean a very unsuccessful project. Group cognition will be discussed in further detail later in this paper.

2.10.2 Absence of Psychological and Sociological Collaborative Issues

It has been noted previously that current groupware tools are primarily technically oriented and do not explicitly address or provide mechanisms to address the social interaction issues that arise in group contexts. For example, none of the software development groupware tools considered in our review takes the psychological and sociological issues of collaborative problem solving into consideration. This phenomenon may be a side-effect of the personal characteristics of the software developers. Indeed, Jones and Marsh (1997) suggest the absence of attention to the social phenomena in collaboration occurs because most groupware designers are technologists whose basic experience lies in developing technologies. The designers lack expertise in the social protocols that play such a pervasive role in groupware systems.

A recent collaborative software engineering case study [in the 13th Conference on Software Education and Training] illustrates the kind of problems that can occur. In this instance, the case study exhibited difficulties with group cohesion, an important phenomenon of group psychology and sociology that must be considered in collaboration, as well as with individual commitment. The technology used in the study provided a diverse collection of media to for communication and collaboration including video, audio, and chat, but there were still collaboration problems. Communication difficulties involved in problem solving groups make the group process much more difficult than the individual process. However, effective communication mechanisms are only necessary but not sufficient to successfully address successful group problem solving. In the case study, the collaborative framework lacked a formal group problem solving methodology that could have possibly eliminated cohesion issues. An explicit problem solving methodology combined with appropriate communication tools would have enhanced group cohesion. The authors of the case study attributed the collaborative difficulties to an absence of group commitment together with a too steep learning curve required to use the system's technologies. Commitment was apparently a problem because group members were inadequately associated with or focused on their particular role in the project: they did not have a sense of ownership. With respect to group cohesion, it is possible that requiring each member to contribute comments to each problem solving stage of the process could enhance the group cohesion.

With the cultural ubiquity of computers, individuals both technical and non-technical now interact with computers. Studies in Human Computer Interaction (HCI), a combination of psychology, social sciences, computer science and technology, have led to some elegant successes that enhance the utilization of computers: direct manipulation interfaces, user-interface management systems, and task-oriented help and instruction. Key issues that have been identified include: how can iterative development be supported and improved; how resources for iterative development can be managed to optimize cost-benefit; and how the scope and richness of user cognitive models can be extended? The research (Carroll, 1997) considers how the technical lessons learned in an iterative development process can be accumulated and exploited

The goal of software psychology is to improve the human use of computers by making computers easier to use. A premier researcher in the area of software psychology is Shneiderman (1980) who emphasizes that the techniques of experimental psychology and an understanding of human skills can be used to improve the design and thereby the impact of computer systems. Some of the relevant techniques in experimental psychology include: the analysis of cognitive and perceptual processes, methods of social, personnel and industrial psychology; and psycholinguistics. Shneiderman contends that though attending to psychological issues may increase the design time and cost, overall design quality will be significantly improved. The tools and concepts of experimental psychology can also be imported and applied to understanding how groups collaborate to develop software. One application of software psychology to collaborative software development is in terms of group communication. Many factors effect such communication: the size of the group, its structure, and the status and personalities of group members (Sommerville, 1996).

2.11 Summary

The ultimate objective of software development groupware is to improve the software development process. To date, such applications have emphasized process and technologies, rather than people (McGuire & Randall, 1998). There are corresponding limitations in current systems, and so corresponding opportunities for improvement. As a prominent researcher in this area has observed: these limitations result from "not understanding the unique demands this class of software imposes on developers and users" (Grudin, 1994). The objective is to turn these defects into research areas.

One significant limitation of current research on collaborative problem solving is that current collaborative problem solving *models do not adequately or explicitly address the characteristics and requirements of group cognition.* Another serious limitation is the limited application of *psychology and sociology in collaborative problem solving models.* Significantly, the *tools* available for collaborative software development *do not explicitly incorporate a problem solving methodology.* Current collaborative software development tools also reflect a highly centralized view of project planning and fail to adequately incorporate domain-specific knowledge of the software domain (Hahn, Jarke & Rose, 1990). The challenge for groupware systems is to design a system that equals or exceeds the effectiveness of face-to-face interactions. Hence, groupware systems should focus on facilitating group interaction, with an emphasis on the core areas of communication, collaboration, and coordination. Group software development should be tightly coupled with group problem solving, and the group problem solving model should be coupled with a group cognition model which will both facilitate cooperation and foster mutual dependence among group members, enhancing cohesion. These models will also provide a conceptual architecture for facilitating communication and coordination among group members.

It is clear that collaborative software development/problem solving will expedite the software development process, enabling faster more cost-effective delivery of product, and more reliable development of complex systems. Suitable collaborative environments can also enhance academic preparation in problem solving and software development. Collaboration with co-workers, or group problem solving, is an expected competency for contemporary software developers because of the complexity and rapidity of application development, as surveys of IT professionals confirm (McGuire & Randall, 1998). Thus, individuals should be exposed to group problem solving models and methodologies at an early stage in their career to prepare them for successful participation in the workforce. Suitable groupware is an effective way to accomplish this, and will prepare individuals for the team-oriented technology-based contemporary workforce. This will train individuals in group dynamics and process related issues, rather than the current product- related focus (McCracken & Waters, 1999).

The overall conclusion of this review of the literature presented is that there are significant opportunities to advance the state of the art by combining views and issues from collaborative problem solving, psychology, sociology and collaborative software development. *Our general objective will be to create a collaborative problem solving model that takes into consideration the problem-solving cognitive* *processes of a collaborative software development group and that addresses the psychological and sociological factors in teamwork.* The model will explicitly address the communication, collaboration and coordination requirements of a group.

Indeed cooperative learning based on interaction and communication will strengthen academic inquiry (Rossetti & Nembhard, 1998). Rossetti and Nembhard found by implementing five basic elements of cooperative learning students actually improved their thinking and problem-solving skills in engineering classes. These elements include: positive interdependence (success of a group depends upon the success of every individual in the group), face-to-face promotive interaction (explaining to each other findings), collaborative skills (group members should understand active listening methods and conflict management), group processing (self-evaluate group success).

Another aspect of the model could be assistance in object oriented software development while problem solving. Problem solving and software development are processes where user requirements are found and transformed into a software application. Traditionally software is developed using structured methodology witch is a top down or bottom up design. When hardware costs rapidly decreased while increasing in functionality, software applications became large and increased in complexity. These sparked the development of object-oriented methodology. Some benefits of using object-oriented methodology versus a structured methodology include the following: code reuse, data abstraction, information hiding, and no adequate means of dealing with concurrency.

The most challenging component of this research will involve tying together theories involving group problem solving and software development while keeping in mind cognition, psychology and sociology of collaboration.

CHAPTER 3

COLLABORATIVE PROBLEM SOLVING AND PROGRAM DEVELOPMENT MODEL

The Collaborative Problem solving and Software Development Process is a detailed cognitive model that takes into consideration the cognitive and social activities that occur during problem solving and program development. Specifically, the structure consists of six stages where each stage is decomposed into three phases. Each phase is a complete sub-process encompassing each of the major collaborative aspects of problem solving and programming. Such aspects include: collaborative modality and group dynamics, where group dynamics breaks down into collaborative processes, side effects and administration. The collaborative modality is the pattern reflecting the nature of the cognitive requirements the group follows to complete a specific phase of the model. The group dynamics takes an in depth look at the specific collaborative and administrative tasks needed to complete the phase while also pointing out possible collaborative side effects and preventative solutions to the side effects.

This model is based on the Dual Common Model for Problem Solving and Program Development (Deek, 1997) that focuses on the individual cognitive aspects of problem solving and programming. The Dual Common Model has shown to improve the problem solving and programming skills of the individual programmer (Deek & McHugh, 2002; Deek et al., 1999). The Collaborative Problem Solving and Programming Model is hypothesized in this proposal to also improve these cognitive skills of a group while taking into consideration the psychological and sociological issues that are present in group work.

3.1 STAGE 1 - Problem Formulation

The *Problem Formulation* stage of the theoretical model incorporates three phases: Preliminary Problem Description, Preliminary Mental Model, and Structured Problem Representation. Following is a detailed description of each individual phase.

3.1.1 Preliminary Problem Description

The first step in problem solving is to create a basic description of the problem to be solved which is called preliminary problem description or intact problem. An intact problem is a problem presented to the group where the whole problem description is given at one time (Dennis et. al., 1996). The key *cognitive processes* and representation techniques for identifying the information needed are described in Deek (1997). The cognitive processes include problem clarification via techniques such as verbalization. Supporting techniques include various kinds of descriptions: verbal, written, symbolic, graphic, or a combination (Greeno, Collins, Resnick 1996; Mitroff & Turoff, 1973). This is a key step because describing a problem effectively and identifying its facts compensates for such cognitive side effects as overlooking known information or introducing unnecessary constraints (Rubinstein 1975; Anderson 1983). In individual problem solving, the domain knowledge and critical abilities of a single person are drawn upon to create the problem description. In a collaborative environment, a collection of expertise is involved and a collective product is agreed upon. The application of multiple experts makes it less likely that known information will be overlooked then in the individual case. The application of multiple critical abilities makes it less likely that superfluous constraints will be introduced.

The *collaborative structure* in this preliminary problem description phase is determined, with respect to its modality by the typical characteristics of a startup process, which include organizational initiation, knowledge base creation, and operational agreements. With respect to collaborative group dynamics, initial scheduling issues dominate the processes, side effects, and administration. The most critical side effect is cognitive bias because the preliminary problem description is the point of departure for the rest of the project. The primary administrative tasks are project initiation and preliminary delegation responsibilities.

The *collaborative modality* should naturally reflect the cognitive requirements of preliminary problem description. This phase requires uninhibited and highly interactive exchange of ideas (brainstorming), cross-fertilizing by the various domain experts (Adelson & Soloway, 1985) represented by the group, and the counter balancing of the critical abilities of group members. In other words the group composition will be discovered in this phase. Group composition includes the group membership characteristics or demographics. The characteristics include whether members are peers or whether a hierarchical order exists (Nunamaker et al., 1991). Peer groups are where the members are not significantly differentiated by differences in status, power relationships, or task expertise. A knowledge base of the individual group members' domains of expertise must also be built to support the delegation of tasks and roles in the stages of the collaborative problem solving and program development process. This knowledge base will highlight the group heterogeneity. Group heterogeneity is the degree to which members of a group possess different types of abilities, status, dispositional qualities or motivation (Steiner, 1972). The knowledge base may also reveal each individual cognitive style preference. Cognitive style refers to the kind of cognitive processes an individual prefers to use to solve problems. Typical types include adaptors, where problems are solved within existing frameworks, versus innovation, where newly invented external structures are preferred (Hohmann, 1997).

Predominately, the c*ollaborative processes* at this phase are negotiation and scheduling with coordination, integration, and acceptance playing secondary roles. By definition, the preliminary problem description phase proceeds from a loosely defined problem statement to an agreed upon reference description of the problem. Reaching a consensus agreement starting from a incomplete problem statement, by its nature involves not only a extensive exchange of ideas but also potentially extensive *negotiation* over disputed points. While such differences may lead to conflicts requiring conflict resolution, one expects the majority of such differences to be resolved in a manner which does not rise to the level of conflict, as is implied by negotiation. *Scheduling* at this phase involves identifying the reference deadline for the overall project and for the conclusion of the preliminary problem description phase. Scheduling is a particularly important process at this stage since the project is being initiated and so there is no prior collective understanding of the parameters of scheduling, such as deadlines and frequency of interactions. The deadlines may of course be modified subject to the external constraints under which the problem is being solved. The granularity of the schedule has to be identified, such as that the participants are expected to update themselves regarding the state of the collaboration daily. Benchmarks must be established, such as preliminary response to the problem statement are due by the second day, critique of responses due by third day, integration of preliminary responses and critique by fourth day, etc. Iterations must fit in with the reference deadline for the phase. *Coordination* issues are secondary at this phase because coordination presumes division of labor into sub-problems, which must then be coordinated. Since sub-problems have not been identified at this point, there is a minimal need to coordinate groups delegated to solve such sub-problems. Rudimentary coordination is handled by the deadlines defined in the scheduling process; but the integration process entails more significant coordination. *Integration* at this phase consists of integrating the group members' contributions to the description and critiques of other members' contributions, which are then combined to form the successive versions of the preliminary problem description. Compatible contributions can be integrated in a straightforward manner, but incompatible contributions require group resolution implemented via negotiation. *Acceptance* of the preliminary problem description version as the reference version for the next iteration is done through consensus, and repeated until an agreed upon final preliminary problem description is arrived at, subject to scheduling constraints.

The main c*ollaborative side effect* is cognitive bias. *Cognitive bias* refers to the propensity of individuals to be consistent and predictable in their cognitive behavior with respect to the kind of errors they make. It presents a particular risk at this stage since the preliminary problem description is the starting point of the rest of the problem solving process. On the positive side, collaboration will tend to reduce the risk of individual cognitive bias because of the mutually counter balancing effects of multiple domain expertise and also multiple critical abilities. On the other hand, the collaborative modality chosen should include measures to mitigate "group think" such as options for anonymous communications. Procedures for *conflict resolution* are critical in this phase of the collaborative problem solving process because there is a potential for a higher level of conflict since the preliminary problem description phase is less structured than subsequent phases and the group members may not be unfamiliar with each other. The required highly dynamic interchange and integration of ideas lends itself to conflict. The implication of unresolved conflicts at this point resulting either from disagreement over the creation of the problem description or interpersonal conflict are more damaging since this phase represents the point of departure for the whole project. Thus, it is particularly important that conflicts are identified and resolved. A significant collaborative side effect at this phase is *distributed learning* precisely because the problem description is a highly joint product involving the application of multiple domain expertise and skills. Exposure to these domains presents considerable opportunity for distributed learning. There is a need and an opportunity to establish *group cohesion* at this phase to reduce the potential for conflict, previously noted to be more likely here. Cohesive groups tend to exhibit higher levels of communication overall, as well as higher task-related and non-task related communication (Dennis et. al., 1990). Many of these side effects may be a result of organizational behavior norms. These norms are preexisting behavioral characteristics of a group (Dennis et. al., 1991).

Collaborative administration is dominated in this phase by initiation and delegation. The collaborative leader *initiates* the project by providing the base problem statement, identifying the scheduling parameters for the overall product, and the local scheduling parameters for this phase: preliminary deadline and granularity of response. The collaborative leader can serve as the initial project *coordinator*, but in subsequent phases as the process become more complex may delegate a separate agent to perform this task. The collaborative leader can act as the integrator, but may delegate this task also on the basis of domain expertise.

3.1.2 Preliminary Mental Model

Following the development of the problem description, a preliminary mental model will be developed. A mental model is an internal representation of a problem that focuses on certain aspects of a situation at the expense of other aspects, which it ignores (Hohmann, 1997). The incompleteness of a model entails that multiple models should be used in problem solving, including but not limited to functional models that identify system functions and interfaces, information models that identify the flow and storage of data in a system, and state models that describe the states a system can be in as well as the legal state transitions for the system (Hohmann, 1997). The goal of this step is to develop a model that can be used as a solid foundation for the following steps of the problem solving process. The essential *cognitive processes* during this step include verbalization and inquiry questions activities (Deek, 1997).

These problem understanding activities have a permanent effect on the rest of the problem solving process (Volkema, 1983) and result in an initial mental model of the problem to be solved.

Verbalization is where the problem question as well as the meaning of the problem terminology and facts are understood (Charles et al., 1987). The inquiry questions activity is where the problem is understood and important information is found within its description (Polya 1945; Mitroff & Turoff 1973; Lauer, Peacock, Graesser 1992), which oblige the problem solver to explicitly identify what is known about the problem, what needs to be discovered, what should be done, and how it should be done (Stepien et. al., 1993). Problem understanding, central to successful problem solving (Lyles & Mitroff, 1980), is the primary beneficiary of this technique. Inquiry questions can be effected by asking questions which provoke the problem solver to examine the problem closely and uncover its goal, givens, unknowns, conditions, constraints or any additional requirements for understanding and solving the problem (Polya 1945; Rubinstein 1975).

In individual problem solving, written notes and diagrams are indicators of an individual verbalizing the problem. In collaborative problem solving this is a more complex process because each member needs to verbalize to the group their understanding of the problem. Not only is it important for the individual to communicate effectively, the group also needs to listen and make sure each member has the correct understanding of the problem. It is extremely important that each member of the team has the same understanding of the problem to be solved.

A common understanding of the problem will undoubtedly increase the effectiveness and efficiency of the solution developed by the team.

The *collaborative structure* in the preliminary mental model phase as far as modality consists of a formal message knowledge base and voting capabilities. The group dynamics regarding group processes consist of formal communication, negotiation sessions, and coordination. The collaborative side effects are mainly concerned with the group being hasty with developing this model. Since the preliminary model will serve as the basis to the other phases of the solution it is important that it is correct. Collaborative administration in this phase consists of facilitating group discussions to create the base model and voting sessions to determine group consensus.

The *collaborative modality*, when developing a preliminary mental model, incorporates tools that can continue the brainstorming activities that occurred during the development of the preliminary problem description. The difference in this phase of problem formulation is that a more formal method of communication needs to take place in order to keep track of the preliminary mental model of the problem. A message board or activity log could be of use where everyone posts their understanding of the problem and each member of the team would be required to vote on the correct model descriptions of the problem. The main purpose of an activity log is to keep a summary of digital records during team member interactions (Dennis et. al., 1991).

Collaborative processes that occur during this phase assist in the team members understanding the agreed upon description of the current problem.

Initially team members will *pull* and *push* information from the preliminary problem description. Pulling is where the user initiates and information access or interaction. Pushing is where other team members will initiate interaction with other team members such as e-mailing or instant messaging. Pushing initiates the ever so important collaborative process of *communication.* It is important that team members individually communicate their understanding of the problem description to each other. This will ensure that everyone is solving the same problem and eliminate errors as the problem solving progresses to further stages. The mode of communication is an *idea generation activity* (Gallupe et. al., 1992) where team members create their own idea of the problem being solved. This stage also requires *negotiation.* This will occur while the group is verbalizing their understanding of the problem. Each member will be voting on correct problem verbalizations creating a need for negotiation if there are conflicting problem understandings. By negotiating, the team will come to an agreement on the problem verbalization and team goals making it easier to make group decisions in later stages of the problem solving process (Ware, 1992). The result will be the dominant group preference representing the majority preference of the group (Dennis, 1996). Each member's verbalization will have to be *coordinated.* This will be accomplished by arranging an organized message board available for each team member to post their verbalization of the problem. The voting/negotiating will also need to be coordinated.

A collaborative *side-effect* while developing a preliminary mental model is the *adoption barrier* of the groupware used during communication. Since acceptance is required by all individuals who use the system, groupware requires greater care to achieve organizational and marketplace acceptance (Grudin, 1994). Another main side effect during this phase is *eagerness*. Often when a group assembles to solve a problem they have a tendency to begin offering solutions to the immediately perceived problem rather than exploring the facts to define the real problem (Hoffman, 1965). This needs to be avoided. A synchronized understanding of the problem will result in a more effective solution. Team members may have a tendency to begin *free riding* where they rely on others to achieve this task without their own contribution limiting their understanding of the problem to be solved (Dennis & Valacich, 1993). Another side-effect is *consensus building*, where by discussing the problem model and sharing the contributions of all the team members an increase of their sense of participation and individual ownership of the joint solution will result (Constantine, 1990). The opposite of consensus building is *conflict,* which may occur when discussing problem understanding. The group should be skilled at confronting their differences. They should adopt the philosophy that disagreements are healthy if they are worked through in pursuit of a better more effective solution (Ware, 1992). If communication is problematic, *media richness* should be considered where emphasis is placed on the communication medium of the group. For example, voice communications may be preferred in situations where relating to others is critical (Whitworth et al., 2000). On the other hand, media richness from digital communications may be more precise increasing positive communication (Nunamaker et. al., 1991). Another way to minimize conflict is to put together a group that works well together. Determination of skill, work ethics, and interpersonal interaction style should be determined about each team member before the team is formulated. For example, a team member could have a *reclusive style of interaction* where they avoid information exposure and solicitation (Hohmann, 1997). Where exposure is there is a willingness to inform others of information and solicitation is where information is actively sought out from others. Some developers are also self-motivated, some are task motivated, and some are interaction motivated. Having too many people in the group who are task-oriented may inhibit the effectiveness of the group's communications, (Sommerville, 1996). Having the right personalities composing the group will contribute to the group's cohesiveness and will produce a synergistic environment. A final side effect of this activity is called *cognitive synchronization*, which occurs when participants make certain they share a common representation of a given subject (Robillard & Robillard, 2000). Nosek (1998) calls this process *group sensemaking* where the group interprets the situation to produce the sense that a shared meaning exists. It is extremely important that the group focuses on synchronization/sensemaking before planning the solution.

The main *collaborative administration* during this phase is when the team leader facilitates the problem understanding process. The team leader should lead the group by explicitly representing the goal of the group (Whitworth et. al., 2000). This would include, initiating the group discussion, whether it is on-line or face-to-face every team member is required to participate and vote on the correct verbalizations of the problem.

3.1.3 Structured Problem Representation

The structured problem representation phase of collaborative problem solving has a goal for the team to identify and organize any relevant information to the problem. The *cognitive process* in this phase requires a more structured approach then previous phases of the model. This more structured approach is a formal information elicitation method to the problem description in order to extract and organize meaningful information in an appropriately structured model for use in the next stages of the process (Simon 1969; Benbasat & Taylor 1982). The goal, givens, unknowns, conditions and constraints are extracted from the problem description and organized by category. Such formal elicitation and documentation of information is essential for identifying, retrieving, and utilizing information in problem solving (Anderson 1983; Rubinstein 1975; Miller 1956).

In individual problem solving, a developer will proceed through this task by refining the preliminary problem description, transforming the problem statement into an organized *knowledge base* that will then evolve during the remaining stages of the process. In collaborative problem solving, the knowledge base will be contributed to by all group members. The group should encompass a *logically large group size* where the expertise of members does not overlap to ensure the knowledge base has enough information to solve the problem (Dennis et. al., 1991). Every team member by either contributing facts to the knowledge base or commenting on the correctness of the facts presented will have contributed all relevant information as well as eliminating all non-essential facts. This activity will not only ensure that every group member will have a reinforced understanding of the goal to be achieved but it will also reduce equivocality and ambiguity. Equivocality and ambiguity are reduced by the mere organization of related project information (Hohmann, 1997).

The *collaborative structure* modality aspect in this phase is based on a message board to coordinate and record the facts and problem decomposition decided on by the team. The group dynamics consist of activities associated with the team producing facts associated with the problem. There may be several brainstorming sessions that have some general collaborative side effects such as groupthink. The administrative tasks can help lessened the collaborative side effects by facilitating productive discussions.

The *collaborative modality* requires organizational skills that are also necessary to not only coordinate the breakdown of components and to associate facts with each identified component, but also to coordinate each team member's contribution to this task. The same format as used when discussing and voting on the preliminary mental model can be used for developing a structured problem representation. It is important to keep track of the group memory because it is not only an artifact of the team but an essential component for successful team functioning (Constantine, 1990). All of the facts with their component associations posted by each team member and recorded on the message board will also be voted upon to come up with a final listing of facts associated with the problem. The final listing of facts will be the *common information* that will be a reference for the upcoming discussions (Dennis, 1996).

When the team is developing a structured problem representation, there are many *collaborative processes* that will occur. These processes are *tightly coupled* in that they require the input from the entire group (Olson & Olson, 2000). The first of which is a *brainstorming* activity. The team members will be brainstorming to come up with a list of facts associated with the problem at hand. The contributions as a result of the brainstorming activity need to be *coordinated* in order to organize the contributed facts. Organization will facilitate the *voting* activity that occurs at the end of the brainstorming session. *Negotiating* is another collaborative activity that may also occur during the voting activity to solve any conflicts.

The main *Collaborative side-effect* that occurs in this phase is *groupthink*. Groupthink is a possibility when voting is involved because it could occur when team members stop critically examining decisions (Hohmann, 1997). It is important the team is reminded that the right balance between quality and time/cost effectiveness always is the priority to minimize groupthink. Just as in the last phase of Problem Formulation, *cognitive synchronization* will also be a side-effect when participants make certain a common representation of the problem is shared. While the team members judge each other's contributions to the problem representation, *cognitive evaluation* will occur.

Just as in the last phase, the *collaborative administration* that occurs is a structured problem representation discussion, requiring every member to participate and vote on the correct facts associated with the problem.

3.2 STAGE 2 - Solution Planning

The *Solution Planning* stage of the theoretical model incorporates three phases: Strategy Discovery, Goal Decomposition, and Data Modeling. Following is a detailed description of each individual phase.

3.2.1 Strategy Discovery

The goal of the strategy discovery stage is to devise a preliminary plan to solve the problem. This is necessary before the additional problem transformations that take place during the subsequent design and translation stages. The main *cognitive process* is planning, which requires using general problem solving strategies to assess solution alternatives and produce a plan for the problem (Deek, 1997). The pre-existing knowledge, beliefs and information about the problem afford an understanding of the problem's requirements, enabling a software developer to plan a preliminary course of action and to devise a potential solution for the problem (Butler & Winne, 1995). In individual problem solving, this is accomplished by drawing upon a single person's creativeness and their knowledge base of previously solved problems. During collaboration, the team has access to more creativity and a larger knowledge base of experience.

The *collaborative structure* in the strategy discovery phase in terms of modality is similar to the preliminary problem description phase in that it contains a database. The difference in this database is that it keeps track of all the proposed ideas and opinions of the group discovered after the problem was understood. It is important for a group to not only have an organized area to promote group discussion but to also have a reference that will keep track of why a particular idea was either accepted or discarded in terms of modality. The group dynamics in this stage focuses on communication and coordination to create an environment in which different ideas can freely be presented and discussed. As with any collaborative phase, there are side effects that need to be monitored and managed.

The *collaborative modality* in this stage reflects the cognitive requirements of discovering a solution strategy. These requirements are all related to the brainstorming activity that occurs during the exchange solution ideas. This phase is mainly concerned with *information provision.* The information available to a collaborative group can range from a well-defined package of information to information provided from a variety of sources and modalities (Dennis et. al., 1991). In this phase, all of the information provided to the team members will be organized in one place. This will occur by a database of all proposed ideas and evaluations being created during the brainstorming process for future referencing and the final alternative selection. The database will essentially serve as the group memory (Dennis et. al., 1997). Groupware that can support the memory of a group is called *organizational memory.* It provides an archive or repository that stores details of the group's interactions. A solution cannot be chosen until every idea is presented and evaluated. Just as a group has a tendency to offer solutions before understanding a problem, choosing a solution before evaluating all proposed solutions needs to be avoided. Keeping track of all ideas and evaluations in a database will make the final selection process more accurate. Before a vote of the strategy, a *categorizer* would be useful. A categorizer is software that group's comments exchanged during a team discussion into categories on the basis of key words (Nunamaker, 1999).

Collaborative processes in this stage turn the initial set of facts associated with the problem into a solution strategy. In order for a team of developers to produce the solution strategy, a few collaborative processes need to occur. The first collaborative process is *communication.* In order for the communication process to be a success, each team member's solution needs to be verbalized and understood by the other team members for it to have a fair evaluation. This can either occur as face-to-face communication or *publication style communication* where the group members do not necessarily know each other and communicate by broadcasting information to the entire group. Team members initiating an elaboration activity can assist the communication process. Elaboration activities should occur when any group member proposes a new solution to the problem under study (Robillard, 2000). All of the proposed solutions and evaluations also need to be *coordinated* in order for the next collaborative process of *negotiation and voting* to occur. Using both *process templates* and *process structure* can facilitate coordination. Process templates can simplify the use of a group tool during activities such as brainstorming and voting etc. (Nunamaker et. al., 1996). Process structures are rules for directing the pattern, sequencing, or content of communications between group members. These structures include techniques such as dialectical inquiry, where subgroups argue for different alternatives, or devil's advocacy, where one subgroup acts as the foil to dispute a solution proposed by another subgroup (Dennis et. al., 1997). After voting on the alternative solution, an *acceptance* of this group decision will need to happen.

The principle positive *collaborative side effect* in this stage is *distributed learning*. It is a side effect in that all of the ideas presented, although may not be efficient solutions for the particular problem at hand, will now be a part of each individual's knowledge base of solutions. This is because each team member is exposed to the knowledge and ideas of the various team members. The principle negative side effect is *cognitive inertia* where a team member may stick to a single thematic line of thought and not explore all of the alternatives (Dennis et. al., 1990). Another side effect that can occur during strategy discovery is *initial preference inertia* where team members may maintain their initial attitude on a solution they suggested and be very closed minded towards any other ideas (Dennis, 1996). *Conflict* will possibly occur if participants argue about using a particular solution presented (Robillard & Robillard, 2000) or may occur due to the initial preference inertia. This type of *collaborative conflict* is referring to a combination of interpersonal conflict between members of a collaborative group and domain level collaborative conflict. By performing a *cognitive evaluation* activity, other team members could judge the value or give their opinion on a particular solution possibly minimizing the any conflict that is occurring. If the conflict is not resolved, team members may resort to reducing their needs for communication (Galegher & Kraut, 1992), which will have a detrimental effect on the project success. Another option is to establish norms. *Norms* are implicit or explicit agreements made by members of a group to minimize disorder concerning what should or should not be done and when. Norms and roles define expectations about what is considered appropriate just as methods facilitate expectations (Steiner, 1972).

There is a lot of *collaborative administration* that needs to occur during this phase of solution planning. The collaborative leader will need to *initiate* the solution brainstorming process and also provide an organized forum for the evaluation of the proposed solutions. The collaborative leader should also assess the progress of the solution evaluation process and make sure each team member is contributing.

3.2.2 Goal Decomposition

This phase is concerned mainly with goal decomposition, which is the subdivision aspect of the problem solving process. Organization and sequencing and any further subdivision are done in the next stage. Goal decomposition is the process of refining the goal into subgoals and subgoals into smaller subgoals. The intent is to break the problem into smaller problems that are more easily solved. The key *cognitive process* is a commonly used technique based on identifying, organizing and sequencing subgoals called *Divide-and-conquer* (Deek, 1997). It is accomplished by using a step-wise refinement technique.

In individual problem solving, goal decomposition is performed with the intention of a single person solving each subgoal. In collaborative problem solving the problem to be solved is much more complex, requiring two stages of decomposition. The first stage the problem is transformed into divisible tasks that are accomplished by different individuals on the team. These tasks are discretionary in which they can be distributed in any manner the team deems necessary (Steiner, 1972). The expertise of each group member is evaluated to determine the best fit for solving each of the sub goals. Individual team members on their assigned subgoal or task perform the second decomposition similar to what is described in individual problem solving.

The *collaborative structure* in this stage as far as the modality aspect of this model is mainly concerned with organizing the project by modifying or adding to the database the various subgoals and assignments of each team member. Group dynamics in this phase focuses on the delegation of individual work, highlighting the need for negotiation and conflict resolutions skills. Assigning individual responsibility or not breaking down the project efficiently could have the most side effects to the project. The administrative task primarily is facilitating the delegation of individual responsibilities.

The *collaborative modality* when decomposing goals requires the exchange of ideas to determine how the problem should be broken down into its subgoals. This is much like the brainstorming session used in formulating the problem. However, the starting point of this brainstorming session is the output from the problem formulation stage, i.e. the refined problem description. Weather the team is meeting face to face or not a *group support system* (GSS) will help facilitate the meeting by supplementing or replacing verbal communications with computer-mediated communications, thereby providing parallelism, anonymity and group memory of the discussion (Dennis et. al., 1997).

The knowledge base of group member expertise that was created in the problem formulation stage will be used to determine which sub-goal best suits the skills and knowledge of a particular developer. An electronic voting system is needed to help put closure to the distribution process. Team members can vote to determine which task they will be assigned. And finally a *workflow application* that manages the flow of tasks among workers such as routing and queuing tasks will be used.

Collaborative processes in the goal decomposition phase occur while transforming the refined problem description into various subgoals, one for each team member to accomplish. *Task clarity* and *task matching* will be the main processes of this phase. The members of a group may have different understanding of a task, and none of these understanding may correctly match the actual problem represented by a situation (Steiner, 1972). *Cognitive evaluation* will occur during the goal breakdown process assisting in the clarification task. This may ensue some *negotiation* on how the main goal should be broken down. Task matching will identify who is to perform a subtask and how the matching is done (Steiner, 1972). This can be done by election, elimination and direct assignment (Hohmann, 1997). The group members should be aware of each other's skills, by the skills knowledge base making the distribution straightforward (direct assignment). If election is preferred, each group member should be vocal about which component they would like to solve to ensure successful implementation. Elimination occurs if there is only one component left and there is someone that is not yet assigned a component. Task matching in collaborative problem solving is similar to load balancing. *Load balancing* is used in parallel processing when tasks are assigned to different processors in order to try to balance processor utilization; effective and efficient task matching balances the team's utilization. *Acceptance* of the distribution is a major factor in the individual's success in solving the problem. If an individual assignment is not acceptable to an individual, *negotiation* will take place again. Finally, the collaborative leader will *coordinate* the individual assignments and schedule review meetings. Coordination is the act of managing mutually dependent activities performed in order to achieve a specific goal (Malone & Crowston, 1990). It is an on going activity in which each member of the group needs to take part. Groups have been found to need a great deal of control and coordination to enable members to collaborate effectively (Finnegan & O'Mahony, 1996). The coordination process is present to keep everyone harmoniously working towards the same goal.

Cognitive or social entrainment is a *collaborative side effect* of particular risk when collaboration occurs. While discussing the goal breakdown, group members tend to become mutually entrained to one another (McGrath, 1990). This fact stresses the importance of having a *highly cohesive* group because highly cohesive groups may tend to be more productive in their thinking and discussions. *Conflict resolution* is critical at this stage because the goal breakdown will be a factor in determining each member's assignment. Conflict resolution will also be a factor when distributing each assignment because it is important for each member to feel confident that they can accomplish their specific goal. This kind of conflict is a result of the team attempting to make a decision on who is working on each part of the problem. Group values guide behavior that is helpful when the group is making decisions. Hohmann (1997) discusses two types of values, interpersonal and functional. Interpersonal values effect how the team members interact. This type of value consists of honesty, integrity, respect, an inclination for direct communication, a dislike for hidden agendas, and an understanding for compromise. Functional values pertain to having pride in your work. For example, developing code that is understandable to other group members. Group members should seek group members with similar values, since values are not something easily modified. These values could then be reinforced during the collaboration. *Cognitive overload* (Fussell, 1998) is also a possible side effect in this phase when a task assigned to a team member is too demanding and not fitting of their skill set. It is important to be aware of every team member's skills so the team's resources are used efficiently.

Collaborative administration leads this phase with *initiating* the problem discussion. The result is the agreed upon and refined problem statement. The collaborative leader also will *identify* and *coordinate* the individual assignments making sure the role of each team member is defined. The *role* is an implicit or explicit agreement made by members of a group that specifies who does every task (Steiner, 1972). The leader much also determine the *self-efficacy* of each individual. This refers to the confidence an individual has that they can solve any problems required to complete a task (Hohmann, 1997). *Scheduling* the review dates for the individual solutions must also be administered.

3.2.3 Data Modeling

The data modeling phase is where the data structures are developed (Deek, 1997). The data must have *persistence* where it is useful beyond the original application or interaction that generated it. Logged and categorized groupware interactions or data structures that are archived in a data repository are an important potential source of persistent data from an organizational standpoint. The *cognitive process* to develop the data structures involves refining the data from givens and unknowns, which were already identified through the elicitation technique of problem formulation. Having outlined a plan and a strategy to implement it, an accurate organization of information suited for manipulation is the last phase of the solution planning stage. Facts acquired during problem formulation may be incomplete or imprecise, but are used as the basis for a comprehensive analysis and refinement of data requirements. The relationships between the problem's givens, unknowns, and the various solution components are established in the solution design stage.

In a collaborative environment, the data and facts recorded in the problem formulation stage by the entire team needs to be associated with each team member's subcomponent. These data structures also need to be accessible by all team members in a global database. Much like individual problem solving, organization of the information associated with each subcomponent will be manipulated. Although each individual is working on their own data model for their subcomponent, the other team members are available for consulting.

The *collaborative structure* modality deals with the characteristics of developing open communication. The group dynamics aspects of the structure are not as apparent since the team members at this point are working individually. However, the team members need to be aware of potential side effects, processes, and administration during the limited collaboration.

The *collaborative modality* in this phase focuses on the future integration of the solution. To accomplish this task each team member needs access to the progress and design of other subgoals. To maintain open communication, each team member will manage a message board dedicated to the progress and design of their subcomponent. This will include storing data structures associated with each team member's component. At this point, only data structures will be the focus but it will serve as a base for the more involved design in subsequent phases of problem solving.

Collaborative processes in this phase are intermittent. The *communication* will be regular in the sense that each member is updating their message board but responding to messages will be an *ad hoc collaborative activity* (Herbsleb & Kuwana, 1998). Team members will store *meta-information* about the data such as interactions with other components worked on or owned by other team members (Romano et. al, 1998). Meta-information will encourage communication and participation from other team members. Team members should be encouraged to respond in that they have a vested interest in assisting or at least communicating with their fellow team mates because the tasks being worked on by each individual are additive. *Additive tasks* are when the group's success is a summative combination of the outputs contributed by all the team members (Steiner, 1972).

There most likely will also be *scheduling* constraints that each team member would need to adhere to. The collaborative leader will need to put strict time limits on the individual work so to not have the rest of the group waiting for one team member to complete the data modeling task.

The main positive collaborative *side effect* in this stage is *distributed learning.* This is a benefit observed in collaborative environments including knowledge sharing between experts and novices and peer-oriented knowledge sharing among novices (Tinzmann, 1990). *Process losses* and *process loss side effects* are also possible due to working mostly on an individual level in this phase (Dennis, 1996). Process losses are inefficiencies associated with the intrinsic characteristics of a process, or factors that decrease performance. In this case group members working on the data structures alone will lose the benefits of collaboration.

Another possible side effect is *conflict resolution.* Maintaining communication and group cohesion during the limited collaboration may encourage less conflict. Group cohesion, also known as *group unity*, is not based on interpersonal attraction alone (Whitworth et. al., 2000). How the group resolves conflicts will also have an effect on the cohesion of the group. There is also possibility for *cognitive resource limitation* side effect. This side effect is a natural limitation when a human engaging in one cognitive activity limits their ability to simultaneously engage in another (Dennis, 1996). Therefore, while a team member is heavily involved in data modeling their own component, it will be difficult to communicate with team members about other aspects of the solution.

Each team member accomplishes *collaborative administration.* Each team member is administering a message board for their component of the solution. Everyone is responsible for updating their message board with status reports and design documents. Each team member is also responsible for acknowledging any comments or discussions from other team members on their message board. This phase is essentially the first time group members are working on their own component. The group leader needs to manage *goal congruence* where the personal interests of the group members stay compatible and aligned with the goal of the group (Nunamaker et. al., 1996). The collaborative leader will also manage the schedule and monitor status updates.

3.3 STAGE 3- Solution Design

The *Solution Design* stage of the theoretical model incorporates three phases: Organization and Refinement, Function/Data Specification, and Logic Specification. Following is a detailed description of each individual phase.

3.3.1 Organization and Refinement

The goal of organizing and refining the problem is to ultimately have the problem broken down in to well-defined tasks. The programming activity is a sequence of design decisions for decomposing tasks (or goals) into subtasks (or subgoals), and maintains that the level of decomposition will effect the ease or difficulty with which a solution will be implemented, adapted or changed (Deek, 1997). The initial problem statement defined in problem formulation was transformed into the preliminary solution decomposition in the planning stage. That preliminary decomposition is now further resolved into a more refined hierarchy of among solution components, each of which is assigned a preliminary function statement. Since the decomposition proceeds in an iterative top-down manner, the subgoals and their inter-relationships may require reorganization and resequencing upon further analysis. Therefore, the *cognitive processes* include the constant refinement of the subgoals until each is a well-defined task by using a visual representation of the problem decomposition and hierarchy between subgoals.

Organization and refinement is completed on an individual basis whether the entire problem is being solved individually or by collaborating. The difference when collaborating is that team members need to make sure their well-defined tasks don't overlap and are compatible with everyone else's task designs.

The *collaborative structure* in this organization and refinement phase has a collaborative modality that reflects the nature of organizing and refining the individual goals of each team member. The group dynamics deals mostly with the communication facet of this process, in particular team members needs to document their status and keep up with other team member progress.

The *collaborative modality* reflects the need for each group member to balance multiple tasks of strategically organizing and refining their component into well-defined tasks, while keeping in tune with the progress of the other component designs. This will require synthesis of *common information.* Common information is information known to members of a collaborative group prior to accomplishing this task (Dennis, 1996). Each developer will synthesize the information output from the solution planning stage with the problem understanding from the problem formulation stage. Synthesis also occurs when the developer transforms the general understanding of their component to a particular designed solution. The component assigned to each team member, which was broken down into subcomponents in the previous stage needs to be decomposed into single tasks, which will activate additional cognitive activities such as organization and specification. The individual message boards build for each group member's component design should be continuously maintained and updated reflecting the organization and refinement of their component. Each group member should frequently check each other's message boards to ensure no overlap of design tasks or incompatibility. *Individually breakdown into well defined tasks*

Collaborative processes in this phase, just as the last phase, are rather intermittent. However, this does not diminish the value of the *group processes* that occur in this phase. The basic processes of a group consist of identifying subtasks that are distributed among members who then coordinate their activities and finally integrate their work to solve the original problem (Hohmann, 1997). As past phases highlighted identifying subtasks and the distribution of them, this phase highlights the beginning of coordinating activities among group members. The way to accomplish coordination is establishing *group protocols*. Group protocols are mutually agreed on ways of interacting (Ellis, Gibbs, Rein, 1991). Both technological and social protocols are necessary. An example of a technological protocol is where each member would *asynchronously communicate*, by posting messages and responding to other team members' design message boards. This type of communication is beneficial since participants can interact without communicating at the same time. An example of a social protocol is the team adhering to the *scheduling* constraints to keep the project moving towards completion. A way to encourage group protocols is to incorporate an *incentive structure* into the protocol. However, collaboratively sharing/seeking information with others requires an appropriate organizational incentive structure (Olson & Olson, 2000). You don't want to cloud the purpose of the protocol with incentives.

The *Collaborative side effects* in this phase revolve around communication. C*onflict resolution* again, being extremely critical to maintain *communication* and *group cohesion* during the limited collaboration. Conflict can be resolved by having *information influence*. Information influence is support for an opinion derived from the quality of the information, rather then from social factors such as the status or number of supporters for that position. The type of communication in this phase will create an *awareness* of what other team members are doing (Fussell et. al., 1998). A possible side effect that may come with awareness is *increased information volume* or *cognitive overload* (Fussell et. al., 1998). Team members will need to judge for themselves which information is pertinent to their sub-solution. Adding passive information to already large amounts of project information could consume too much of the teams resources (Fussell et. al., 1998).

The *Collaborative administration* in this phase is actually performed by each team member. Each team member will administer a message board for their component of the solution. Everyone is also responsible for updating their message board with status reports and design documents and for acknowledging any comments or discussions from other team members on their message board. The collaborative leader will manage the schedule and monitor status updates.

3.3.2 Function/Data Specification

This phase of the model focuses on the functions and data flows between the modules refined in the previous phases of the model. The function of each module is a statement of the goal of the module. Data flows between modules are identified using information gathered during the formulation and planning stages. The data gathered from the previous stages needs to be refined. This is accomplished with a data table that specifies the precise data flow between the models, indicating the source, destination and type of data. Typically, intermediate data elements may have to be introduced, reflecting the decomposition process. The result is a data dictionary table that includes the data element names, type, description, the associated goal and the direction of the data flow.

Collaborating during this phase requires not only data flow between the modules of but also the data flow between each team member's sub-problem. This adds to the already complex process of function/data specification when individually solving a problem. The success of this phase is critical for smooth integration performed in stage five of the problem solving process. A well-planned data flow between each team members sub-problem will also eliminate or at least reduce the amount of debugging to be performed in stage six. The main *cognitive process* that occurs in this phase is synthesis (Deek, 1997). Synthesis is where the modules of each team member are developed into a coherent whole by rearranging and establishing relationships.

The *collaborative structure* of function and data specification focuses on creating open communication and organizing function and data requirements of the problem solution. These modal requirements are accomplished with collaborative tools such as message boards and chat rooms. The group dynamics in this phase include processes that promote cognitive synchronization and eliminate conflict and cognitive bias. Administratively, the group leader will guide the team toward a positive outcome in this phase.

The *collaborative modality* in this phase must provide a forum for open communication and discussion to facilitate the synthesis of each team member's component. An efficient synthesized design of function and data specifications is critical for later debugging and integration stages. Interactive communication needs to occur along with documented outcomes. Thus, specifying the data specification responsibilities for each team member's component is key. This is accomplished by adding documentation to the project database specifying the data required by each team member's sub-problem. Facilities for open communication will also be necessary such as a chat room or an asynchronous messaging tool. To ensure successful communication, the tools used by the team members should have *common ground factors*. These factors are environmental characteristics that facilitate establishing a shared collaborative experience (Olson & Olson, 2000). These characteristics include factors that enhance cueing such as audibility, contemporality (immediate receipt of messages), simultaneity (all participants can send/receive messages simultaneously), and factors that enhance message quality such as reviewability (messages can be reviewed afterwards) and revisability (messages that can be revised before sending) (Olson & Olson, 2000). In addition, the asynchronous messing tool will be one of the most frequently used tools and should have unobtrusive accessibility where it is readily accessible and not obscured by being integrated with less frequently used system features (Grudin 1994).

The most important *collaborative process* in this phase is *communication.* It is imperative that the data needs of each team member's sub-problem be expressed and understood to ensure *cognitive synchronization.* This will eliminate any misunderstanding of the data requirements of each sub-problem. Team members will also need to *negotiate* responsibilities imposed by the needs of other sub-problems. It is important that the data requirements are designed in the most efficient manner. At the completion of any negotiations, *acceptance* will occur and the responsibilities will be documented. Team members should also individually organize their own sub problem decomposition to determine the most efficient way to solve it. Team members could create a selection if more than one sequence is possible as well as simultaneity of subtasks and communicate with other team members about finalizing a sequence that would be most beneficial to the team (Steiner, 1972).

As with any negotiations, *conflict* is a possible *side effect.* Resolution should be immediate to avoid wasting time. *Group cohesion* is another side effect in that the more each developer discusses their sub-problem and design the more cohesive the group will become. *Cognitive bias* is also at risk if the data requirements for each sub-problem are not well thought out and understood by all team members.

The group leader will perform the *collaborative administration* in this phase. The collaborative leader will need to perform *status monitoring.* This is an ongoing activity to ensure a project is on schedule and to determine actions in case of schedule slippage. The leader will concentrate on monitoring the data requirement responsibilities and document them in the project database.

3.3.3 Logic Specification

Once modular decomposition and data flow are completed, the process proceeds by specifying the algorithmic logic for each module. This requires the use of algorithmic pseudocode to identify the data structures, control structures, and operators. This pseudocode generated at this stage can serve as the basis for code translation to a target programming language. The *cognitive process* that occur in this phase of the solution design include cognitive strategies which demonstrate the ability to carry out the transformation of previously developed plan for a solution into an actual solution (Gagne, 1985).

Logic specification is accomplished individually whether this is a collaborative or an individual project. The difference with a collaborative project is a pseudocode review. It is important to review the logic for each sub-problem before time is wasted translating inaccurate pseudocode to a programming language. This could be accomplished with a brief face-to-face or electronic meeting.

The *collaborative structure* in this phase includes tools to assist the pseudocode review as far as the collaborative modality aspect of the structure is concerned. The group dynamics occur more casually since this is a relatively individual phase of the problem solving model. Side effects such as cognitive overload can be minimized by the collaborative administration aspect of this phase.

The *collaborative modality* in this phase is accomplished with electronic meeting tools assisting the pseudocode review. Support for electronic meetings include brainstorming tools, asynchronous messaging tools, document databases, chat, synchronous audio and video etc. In essence these environments support *computer-mediated interactions* (CMI). CMI refers to group environments where all interactions between the members of the group are through computer communications only, as distinguished from computer supported face-to-face interactions (Whitworth et. al., 2000). The main quality of an electronic meeting tool is WYSIWIS. This stands for "What You See Is What I See" This is sometimes considered a fundamental presentation requirement of meeting tools, namely that they provide a coordinated interface for all participants (Stefik et. al., 1987). However, it is important that team members not *destabilize* other means of communication such as simple pencil and paper calculations Nidamarthi et. al., 2001) .

The *collaborative processes* are intermittent because the logic specification is mostly accomplished individually until the pseudocode *review* is performed either face-to-face or using CMI. If using CMI, the review can be done via *depersonalization* where it is anonymous allowing more objective evaluation and better error detection to the extent that persons are separated from the contribution being critiqued (Nunamaker et. al., 1991). It is also important that each team member *communicates* all *unique information* pertaining to his or her component. Unique information is information that is known to only a single member of a collaborative group prior to group discussions (Dennis , 1996). This information may effect another subcomponent being developed by another team member.

The main *collaborative side effect* is *evaluation apprehension*. Team members may not want to express their logic specification ideas or important project information because they fear criticism and evaluation (Dennis & Valacich, 1993). Anonymous computer-mediated communications are intended to reduce evaluation apprehension especially in the presence of status differences or pressures to conform (Dennis et. al., 1997). Presentation of *partial information* may have a detrimental effect on the outcome of the solution. Partial information is information that is only known to a subset of members of the collaborative group (Dennis, 1996).

Another side effect would come from the casual communication is *distributed learning.* Team members are exposed to enlarged knowledge base of skills and expertise by collaborating. Another positive side effect is *cognitive synchronization.* Cognitive synchronization refers to the kind of cognitive process that occurs when group members synchronize because they have a shared representation of a problem or solution. In this phase by having a successful pseudo-code review, all project information will be known by all group members causing group members to synchronize cognitively. This will facilitate the effectiveness of each team member's sub-component solution.

The basic c*ollaborative administration* that occurs in this phase is the supervision of the pseudocode review meeting. This meeting will reassure that every team member's sub-component is compatible with the other sub-components developed.

3.4 STAGE 4 - Solution Translation

The *Solution Translation* stage of the theoretical model incorporates three phases: Implementation, Integration, and Diagnosis. Following is a detailed description of each individual phase.

3.4.1 Implementation

Implementation involves transforming the detailed design into instructions suited for compilation and execution. Data modeled in earlier stages is transformed into type definitions, declarations, parameter statements, etc. Pseudocode is converted to syntax specifying operators to manipulate and transform the data to produce the desired results. Preliminary documentation of the individual instructions and the modules is done at this time. The key *cognitive process* that occurs in this phase is the application of knowledge. The knowledge acquisition component is concerned primarily with determining relevant language features and integrating previously identified partial solutions. At this point, all of these details of each sub-problem are in line with the other sub-problem designs because of previous collaboration tasks. Therefore, this work can be accomplished individually. The other team members are available for support if any questions should arise during the implementation.

The *collaborative structure* when implementing the solution translation deals primarily with the awareness aspect collaborative modality. The group dynamics in this phase are nominal because of the nature of this phase. However, communication will continue to be a major collaborative process and scheduling is side effect that needs to be monitored.

The *collaborative modality* in this phase is related to the main cognitive requirement, which is awareness. The team members should have awareness of each team member's implementation plan. For example, if everyone is developing their code in Java they should all use the same Java compiler to ease integration steps that will follow. There should be a posted document stating the project requirements as far as language, compiler, schedules, etc.

Collaborative processes are minimal in this phase because it involves tasks that can be completed by each individual without much collaboration. This is a result of all the effective collaboration that happened in previous stages of collaborating. The lines of *communication*, both asynchronous and synchronous, should still be kept open for any issues that may come up. Team members could also use *direct style communication* where members know each other and communicate directly, as opposed to publication style communication where messages are broadcasted.

A *collaborative side effect* in this phase is not staying on *schedule*. The collaborative leader needs to acquire implementation status from each team member on a regular basis.

The *Collaborative administration* in this phase as with the previous phase that is accomplished individually, the collaborative leader needs to enforce the scheduled time allocation for the implementation phase. Since the team members are doing this individually, this phase is more difficult to monitor but can be easily accomplished with required status reports. A simple e-mail to the collaborative leader would suffice. If there are any issues, the collaborative leader could call a meeting with effected team members. It is imperative that the team stays on schedule so as to not waste valuable man-hours waiting to move to the integration phase.

3.4.2 Integration

In addition to writing new code, program implementation can also involve code reuse, which involves integrating existing code used previously. It is classic problem solving strategy to recall similar problems that have been solved previously; this correlates with current thinking in the software design methodology where the strategy is to develop new software systems by, as much as possible, integrating previously written code. This is also called *opportunistic design* where each developer solves key parts of a problem by mentally scanning their personal cognitive library of solution plans until they identify a plan that matches the problem at hand (Hohmann, 1997). The *cognitive process* occurring in this phase is synthesis. Synthesis transpires in two ways: first, with respect to the integration of existing software components into the solution, and secondly, with respect to the piecemeal integration of the modules under development.

When individually solving problems, the integration phase includes only code reuse and individual integration. Adding the collaborative factor to the integration phase adds an additional task: the integration of the sub-problems of each team member. This additional integration task will cause the integration phase and the diagnosis phase that follows the integration phase to be iterative. Therefore, the team members will first perform the integration phase and the diagnosis phase individually then when every team member has individually completed these two phases, the team will revisit these phases together doing the additional collaborative tasks.

The *collaborative structure* has a main focus of coordinating the integration efforts. A source code tool would fulfill the modality need for this phases structure. The group dynamics will be modified slightly since there will be a central person leading the coordination efforts with assistance from each of the other team members. This person will lead all collaborative administration of this phase and is most at risk for the collaborative side effects of this phase.

The *collaborative modality* of this phase accentuates the synthesis cognitive requirement. A tool for a team member to not only integrate their work but also enable integration for the entire team is necessary. This tool must provide a secure source code database, which will promote both code reuse and the overall project integration. Source code tools are usually designed to handle huge repositories of source code and project data, which is ideal for a team project. A *networked application* that is able to distribute files over a network could also be useful. For example, an application such as FTP (file transfer protocol) might be needed to transfer files from one machine to another. A *prioritizing tool* (Dennis et. al., 1997) is also necessary to assist in deciding the sequence each component should be integrated.

The most significant *collaborative process* that occurs in this phase is the *identification* of the person leading the integration effort. This person needs to have the most *domain expertise* in order to integrate the entire project without difficulty. Integration will also require *coordination* and *scheduling* processes. The leader of this effort may choose to coordinate the integration in a way where one team member's component is integrated at a time making the next phase (diagnosis) easier.

Scheduling each component's integration is necessary to keep the project moving smoothly.

The main *collaborative side-effect* is *process bias*. Process bias is a cognitive bias in favor of certain kind s of process in decision making. Example process biases include: preference for overly length processes, excessive preference for group processes, excessive analogizing, and over simplification (Hohmann, 1997). The team has to not only agree on the process that will be used to integrate the project but also monitor the integration progress.

Another side-effect is *cognitive overload* of the integration leader. If the person chosen for that role does not have the proper skill set they could experience cognitive overload. If there is a situation where there is an integration leader with a limited skill set the team members could lessen the overload by communicating effectively with the integration leader to assist in efficient coordination and integration efforts.

The integration leader predominantly handles the *collaborative administration* of this phase. That person will also coordinate the integration effort with scheduling any necessary deadlines for integration and the forthcoming diagnosis phase.

3.4.3 Diagnosis

Debugging is the process of diagnosis that refers to the identification and remediation of possible errors. Debugging is a process also supported by existing development tools in most programming environments. Possible errors could include: syntax errors, run-time errors, and logical errors. The programming environment detects the first two types of errors. Syntax errors are found when the program is compiled. Modern programming environments provide sophisticated error reporting and debugging utilities to assist with syntax problems. Run-time errors are detected and reported during program execution. Logical errors, on the other hand, are not detected by the system and may be a challenge for the developer to correct. These types of errors are usually handled in the testing stage. The main *cognitive process* of this phase includes applying language syntax and problem domain knowledge during the diagnostic analysis of errors.

Just as in the integration phase of this stage, diagnosis has an additional task when developers are collaborating. Once each team member has completely debugged their sub-problem solution they will return to the integration phase where each of their sub-solutions will be joined together. Following the joining of all sub-solutions, the team will return to the diagnosis phase to debug the entire solution, thus completing the iteration of the integration and diagnosis phases. Debugging the entire solution will be much more complex in the second iteration in that their will be mostly run-time errors to detect.

The *collaborative structure* in this phase is mainly concerned with a bug database as far as the collaborative modality is concerned. The group dynamics deal with integration, scheduling processes while the side effect focuses on conflict. The main administration tasks in this phase are assessment and delegation.

The *collaborative modality* during diagnosis focuses on the characteristics of run-time errors during the second iteration of the debugging process. During the second iteration of this phase, the person leading the integration efforts will most likely be doing the debugging. This person may encounter run-time errors that need to be solved by the team member who designed the component that may have the problem. Therefore, a database containing the list of bugs and who is being assigned to look into the bug needs to be created. A notification protocol will also be followed. For example, this protocol could be accomplished through an e-mail application where the recipient would be required to respond with weather they are going to fix the bug or not. Instant messaging is also a possibility if the recipient is available.

The main c*ollaborative process* in this phase will take place during the integration and testing of the each team member's component. This task should be accomplished by *scheduling* the integration of one component at a time. This will facilitate the diagnosis process. *Communication* is also the key to success during this phase. Each team member will need to be aware of possible run-time error bugs they may need to investigate. Details of the problem will be communicated via the bug database or *synchronous communication.* Participants when communicating synchronously interact at the same time, in contrast to asynchronous communication such as the bug database. Synchronous groupware systems run in real time and support group communication and collaboration using such techniques as instant messaging. *Task complexity* will also vary due to the different types of possible bugs, i.e. logical errors being more difficult to detect then syntax errors. The group will need to work together to fix more complex problems during this phase.

The main *collaborative side-effect* in this phase is possible *conflict* during diagnosis. If the integration leader experiences problems during debugging he/she is going to need to diagnose the problem by essentially pointing the finger at one of the team member's components. It is important that each team member goes into this phase with an open mind and accept any problems that may arise. If the previous stages are completed correctly, there should be minimal problems eliminating any conflict.

The integration leader will manage *collaborative administration* in this phase. That person will need to *assess* any problems that comes up. The assessment will involve *delegating* the problem to one of the team members for further assessment.

3.5 STAGE 5- Solution Testing

The *Solution Testing* stage of the theoretical model incorporates three phases: Critical Analysis, Revision, and Evaluation. Following is a detailed description of each individual phase.

3.5.1 Critical Analysis

The goal of the critical analysis phase is to develop test cases that verify the goals have been met by the program developed by the team. In other words, the test cases will determine that the problem has in fact been solved. This requires an examination of the original goal, requirements, and specifications of the problem. The problem requirements, data modeling and design specifications are used to define test cases for each goal and subgoals. Code-based tests are also designed on the basis of the control variables in the program. Developing test data to use as input for the program under verification is the first task in this stage. The objective is to design and apply a testing strategy that uncovers all program errors.

The main *cognitive process* at this phase is analysis. The team needs to analyze the original goal and solution requirements in order to develop appropriate test cases. In collaborative problem solving, the solution is much larger thus the need for more elaborate test cases. Initially, team members will create a test case for their component. Once the results for the individual components are correct, a test case will be developed collaboratively to test the integrated solution. It is important to test the individual components first to facilitate the analysis of the large integrated solution.

The *collaborative structure* modality consists of fundamental collaborative tools to reach the goal of testing the solution for correctness. The group dynamics focuses on accepting results, scheduling testing, and avoiding cognitive bias. Each team member will have a hand in the collaborative administration of initializing testing and assessing the results.

The *collaborative modality* tools needed to critically analyze the solution consist of fundamental collaborative tools. Most of the collaboration will occur when the team is creating the test case for the integrated solution. Initially, the team will use the tools to assist in any difficulties encountered while testing their individual components. Then the team will need a forum to discuss the best way to test the integrated solution thus brainstorming tools; voting tools, documentation storage tools will all be useful during this task.

The main *collaborative process* that occurs while critically analyzing the integrated solution is *task division.* Task division is how the group will divide the critical analysis task into smaller subtasks. This may be externally specified by the environment or determined by the group (Steiner, 1972). The critical analysis task is developing test cases to analyze each component of the solution. In order to completely test the solution, a *rational model of behavior* needs to be addressed. This is where the decisions of a group, in this case the test cases need to be decided on, are intentionally rational choices, as opposed to politically motivated decisions that are not optimal from the viewpoint of an organization (Dennis et. al., 1991). Another collaborative process is *scheduling.* The integration leader will create a testing schedule in order to keep pending deadlines. The team will also have *acceptance* of any results that come from the testing. Team members will *negotiate* problem results as well as proposed solutions to incorrect testing results.

A *collaborative side-effect* that is the largest hazard to critical analysis is *cognitive bias.* Team members that develop test data for their own component may cognitively bias the data to always produce correct results. In order to choose data that will challenge the outcome of the solution, team members should not critically analyze their own component. *Outcome bias*, a type of cognitive bias, is also a possible side effect. Outcome bias is when certain kinds of outcomes from decision making is favored. In this phase, the outcome bias is an aversion to thorough testing of a solution (Hohmann, 1997). *Cognitive simplification* is a possible side effect that is similar to outcome bias. Cognitive simplification is that it is the tendency to use easily available data and make conclusions after only using a small sample (Nosek, 1998). A positive side effect is possible *process gains* that increase the performance of the group (Nunamaker et. al., 1991). A processes gain is possible synergies that occur while the group is deciding on test cases and test procedures. In order to maximize the synergies occurring within the group, team members need to be given *airtime*. This refers to the need in a non-computer mediated environment for group discussions to partition speaking time among members because they can't access the floor simultaneously (Dennis et. al., 1991).

The *Collaborative administration* that takes place during this phase is *initiating* the analysis process as well as *assessing* the testing results. Each team member should both initialize the analysis of their component as well assess the results. The integration leader will initialize and assess the critical analysis of the integrated solution.

3.5.2 Revision

The outcome of the critical analysis phase may produce incorrect results. Either the whole solution or some part of it does not match the purpose intended when the plan was created. It may be necessary to reorganize or retrace the solution path, returning to previous phases of the problem solving process. The solution may require changes to a program affecting its logic, language constructs used, or data representation. In the case where errors are found, the program will require modification and perhaps revision either in the code or the design. This requires that the location of the error be determined, their cause established, and corrective measures taken. The main *cognitive process* involved when revising a solution is self-critical attitude (Gagne

1985), which demonstrates the ability to critically assess one's own thought processes as well as ones' own intellectual creations. When collaboratively solving a problem, the group will be assessing critically not only their own components but also the team's integrated solution, which in effect will be assessing other team member's work.

The modality of the *collaborative structure* will focus on previous tools created to complete the other phases of this model. The group dynamics will focus on activities that will help the team communicate in order to pinpoint the solution problem.

The *collaborative modality* will first need to distribute the revision task to the appropriate team member. A *workflow application* similar to what was used during the component distribution would be useful during the distribution of the particular revision task. The actual revision will require the particular developer or team member to look back on the entire problem solving process. They will need to review all of the databases that were created to store the outcomes of the previous problem solving phases. That information coupled with the incorrect results from the critical analysis should give the developer insight on an appropriate revision.

Collaborative processes during the revision phase are processes that can assist in locating the origination of problems discovered in the critical analysis phase. This will require the activities such as *communication* and *brainstorming* between group members. If a particular problem cannot be broken down for several team members to solve, it is called a *unitary task*. These types of problems correspond mathematically to tasks that cannot be usefully paralyzed, such as computing the square root of a number (Steiner, 1972). Choosing the team member to solve a unitary task or problem will initiate a *negotiating* process. Incurring *credibility of information source* can facilitate the negotiating process. This is a key factor in acceptance of information (Dennis, 1996).

During the task of pinpointing the origination of the incorrect results, the team may encounter some c*ollaborative side effects*. The main side effect is *domain level conflict*. This type of conflict refers to inconsistencies in design criteria. It differs from collaborative conflict that refers to interpersonal conflict between members of a collaborative group. However, domain level conflict can turn into collaborative conflict. If the group is assessing the revision either on-line or face-to-face *flaming* may result. Flaming is uninhibited, angry negative criticism (Nunamaker et al., 1991). During the revision task, *confirmatory bias* is a prospective side effect. This type of bias is where the group has a cognitive tendency to seek and observe evidence that verifies or confirms a viewpoint. This bias can be reduced by explicitly searching for errors in the design rather than an attempt to verify the error (Stacy & Macmillian, 1995).

Assembly effect is also a possibility due to the consequences of group composition (Steiner, 1972). Group assembly may result in more effective solution revisions. Another positive side effect is *cognitive stability*. Cognitive stability is the tendency of a group to resist changing the focus of a discussion due to social inhibitions against repeatedly changing the focus of a discussion (Steiner, 1972). The importance of the revision phase warrants this type of focus.

The team leader will manage the *collaborative administration* during the revision phase. That person will need to *initiate* the revision process by determining which team members need to revise their component. *Assessing* the results from the critical analysis phase and reviewing the entire project can accomplish this. The leader will need to also *delegate* responsibility to various team members.

3.5.3 Evaluation

The final phase of solution testing is evaluation. Evaluation requires assessing the completed solution and monitoring the thinking process. Techniques used to monitor thinking are called cognitive strategies, or metacognition. Metacognition is the main *cognitive process* in the evaluation phase because it is concerned with monitoring the thinking process and evaluating the solution (Sternberg, 1985).

Individual observations made as the solution evolves, called internal feedback, provide grounds for reassessing the problem's need and the solution. Internal feedback is an important progress indicator of the problem solving process and is triggered as a result of the problem solver's own comprehension of the project's progress. External feedback, such as comments provided by other team members, can either confirm or conflict with the developer's strategy, thereby also causing reassessment and adjustment.

Collaborative problem solving differs in individual problem solving in that the focus is on the external feedback. It is important to include the entire team in the evaluation process since collaborative problem solving is essentially a *vertical division of tasks* were the team members are depending on each other's performance

(Steiner, 1972). This is opposed to a *horizontal division of tasks* where the division of labor in which each member of a group performs all aspects of a task. This is a less efficient way of dividing tasks (Steiner, 1972).

The main characteristics of the *collaborative structure* when evaluating the collaborative solution are communication tools and processes that promote convergence and team exposure. The side effects to be avoided are critical mass problems and situations where team members are not able to take suggestions from each other.

The *collaborative modality* in this phase is any tool that can assist in the communication process. Fundamentally, the tool should provide *collaborative convergence* (Romano et. al., 1998) where the team can come to a conclusion or decision on a critical issue using a collaborative environment. For example, asynchronous and synchronous messaging tools such as chat, e-mail, and/or a discussion board. In addition a voting tool would be used to obtain the teams final consensus.

The primary c*ollaborative process* that will occur while evaluating is c*onvergence* as described above. In addition to converging to a decision the team will develop a shared understanding of the solution in order to be able to accurately evaluate the solution's accuracy. The shared understanding occurs by way of *information exchange* between all of the team members (Dennis, 1996). The goal is to have a *consensus change* after the information exchange, which is the difference between the post-group consensus and the pre-group discussion consensus (Dennis, 1996). While communicating to assist in the convergence process the group will gain a more detailed *exposure* (Hohman, 1997) to the components of the solution.

The most serious *collaborative side-effects* is the *critical mass/prisoner's dilemma problem* (Grudin, 1994). Some team members may not see the importance of the evaluation process leaving a gap in this phase of solution testing thus the team will not be able to reach critical mass of the groupware system that is being used for communication. A non-participating team member may feel their component works and the evaluation process will not benefit them so they possibly will abandon the project before every aspect of the solution is evaluated. Another side effect that may occur is *A priori preferences* (Dennis, 1996). During evaluation, there may be some criticism of a particular component design. The team member who worked on that design may focus only on information supporting their opinion thus, ignoring anything contradictory. *Normative influence* is also a problem during evaluation. This type of influence could occur during a discussion and the support for an opinion is derived from secondary factors such as the status of participants. Normative influence also refers to the tendency of individual's to defer to what they perceive as the group opinion without the need for group pressure, coercion or persuasion. Another side effect related to normative influence is *group polarization*. This refers to the alleged tendency of groups to adopt more extreme positions or decisions than individuals (Whitworth et al., 2000).

The main *collaborative administration* task in this phase is *facilitation* (Dennis, Nunnamaker, & Vogel, 1991). The communication process needs to be facilitated by the team leader. The leader will have to manage and encourage participation in the evaluation process. The leader must keep in mind that evaluation is an additive task. An additive task (Steiner, 1972) is when the success of a collaborative task is based on the result of all team members participating. Success will be determined by the combined effort of the group. Team members will each lead the communication of their solution component evaluation.

3.6 STAGE 6 - Solution Delivery

The *Solution Delivery* stage of the theoretical model incorporates three phases: Documentation, Presentation, and Dissemination. Following is a detailed description of each individual phase.

3.6.1 Documentation

Documentation has significant consequences on the clarity and readability of the whole solution and is essential for comprehending and modifying programs (Tremblay & Bunt, 1989). The major *cognitive process* during documentation is synthesis. Synthesis requires the ability to produce a well-organized whole (Bloom, 1956).

In addition to the documentation generated during the earlier stages of problem solving, program documentation including comments and explanations are important to understand the approaches and techniques used to solve the problem. Maintenance, which may require modifying the existing program functionality or addition of new user requirements, would be difficult without adequate documentation. Other forms of documentation, such as help features or user manuals in the case of complex systems, are also essential for understanding system operations.

In collaborative problem solving, the amount of documentation is increased largely because the problem is larger then a problem solved by one individual. On one hand this is not a problem because team members could document their own component, however, the integration of the documentation needs to occur as well as documentation regarding the integration of each individual component.

The *collaborative structure* during the documentation phase incorporates processes that focus on the organization of the main project document. Organization requires communication and coordination of the documentation tasks as well as identifying a team member to lead the integration of documentation effort.

The *collaborative modality* needed for documentation is a tool that can monitor the activity of a document as well as limit modification rights to one team member at a time. In other words, all of the team members will be able to read the document but only one user will be able to modify it. Each team member should create their own documentation for their component and post it to the documentation tool. One team member will then integrate the component documents into one main document.

The main *collaborative processes* needed to create documentation for a collaborative solution is *coordination.* Each team member will be creating a document to describe the component solution they developed. These separate documents need to be coordinated in order to *integrate* them in the main solution document. *Symmetry of information* (DNV) also needs to occur so that each team member is using a similar format facilitating the integration of the documents.

A major *collaborative side-effect* that occurs is *information redundancy* (D). Since each team member will write about features and reasons for specific design decisions, some information my get repeated. This is not necessarily a negative side effect. Information may be repeated and the integrator of the documentation may choose to leave in redundant information to increase *information saliency* (DNV). An increase in saliency can decrease the possibility of important information being overlooked.

The main *collaborative administration* activities are *facilitation* and *identification*. The team member that will take charge of the document integration needs to be identified. This person will also facilitate the documentation activity of each component by the other team members.

3.6.2 Presentation

Once the solution has been tested and documented these results have to be organized as a report. The important *cognitive process* of this phase is the communication of verbal information as exhibited by the ability to formulate and organize a complete and coherent report (Gagne 1985).

In individual problem solving, a presentation can be done relatively easily with a written report. In the case a group projects, the results may need to be orally presented in addition to a well-organized report. The written report documents, in an orderly manner, the original problem description/requirements, solution plan/specification, and the coded/verified solution.

The *collaborative structure* of the presentation phase centers around providing a documentation tool with shared access and the team deciding on essentially how to use that tool most efficiently. Efficiency is measured on the basis of how quickly the entire written presentation can be integrated.

The *collaborative modality* when a group is planning a presentation needs to focus on standardizing the tools and style used during the presentation design. Team members will be developing a section of the presentation based on the tasks assigned to them, therefore, a standardized style will enhance the execution of the presentation. Finished presentations can be posted in the documentation tool for easy access by the presentation integrator.

The c*ollaborative process* is not only the *development* of the written project presentation but also the *execution* of the verbal presentation. Depending on the group proximity, the presentation can be completed by either one or all of the members. The team will also need to *vote* on the standardized style and tools to be used during the development of their individual portions of the presentation. During the actual presentation, the *media speed* should be considered while deciding on the method of presentation execution and tools to be used. The relative speeds of typing, reading, speaking, and listening effect the amount of information available to and processes by a group, thereby affecting process gains and losses (Nunamaker et al., 1991).

When presenting a large organized project the main *collaborative side-effect* is *synergy* (Dennis & Valacich, 1993). The group will build on each other's presentation making the whole of the presentation greater then the sum of the individual parts of the presentation. This will only occur is the *group proximity* (Nunamaker et al., 1991) is close enough for the presentation to be executed as a group. A negative side effect that could occur during a presentation is *production blocking*. This is blocking associated with mutually exclusive access to a resource. For example, in a verbal exchange only one person can speak at a time, so other participants are blocked in the meantime (Dennis, 1996). *Attenuation blocking* is also a possibility because it can occur during production blocking. This type of blocking is when a member of a group forgets or suppresses expression of an idea that could not have been expressed in a timely way because of production blocking (Nunamaker et al., 1991).

The *collaborative administration* of developing both a written and oral presentation focuses the decision of *standardizing* a tool and style of the presentation. The team leader can *facilitate* a vote and *identify* a presentation integrator of all the project components.

3.6.3 Dissemination

Most projects and vital project information needs to be disseminated to the appropriate community of interest. The most significant *cognitive process* is the performance component directing task organization (Sternberg, 1985). Individually this may not be a difficultly organized task, however, because of its size a group project will require much more organization of the information to disseminate as opposed to an individually worked on project.

The *collaborative structure* during dissemination deals mostly with the interaction between the team leader and the group requesting the solution to the original problem such as communication, conflict, externality, and decision-making.

The *collaborative modality* of the dissemination phase is more feasible and more important, by the availability of Internet technology. The project and the files can either be uploaded to an Internet site for easy downloading or depending on the size of the files can be compressed and e-mailed to the interested parties.

Collaborative processes of dissemination are at this point between the team leader and the group or company who initiated the problem that was solved. The collaboration focuses on the *communication* of the dissemination process. Where the process is concerned with the method in which the project will be disseminated. Communication is not an easy process because of *cognitive multi-threading* (Whitworth et. al., 2000). This refers to the notion that a single act of communication, ranging from the literal content of the message, sender context information such as about the state of mind of the communicator, and sender position such as an associated intended action. The Dissemination will be a *mass publication* where identical information is dispersed to a group of users. This could occur via FTP, e-mail, or regular mail.

The only *Collaborative side-effect* might be any *conflict* that may occur while deciding on a dissemination process. When the solution is actually used there may be either positive or negative *externality* depending on the how many people accept or use the product. When distributing the solution, *representativeness bias* needs to be considered. This is the cognitive tendency to expect the local characteristics of what seems o be a typical sample are general or global characteristics (Stacy & Macmillian, 1995). Therefore, the dissemination process needs to be approved by the receiver of the solution as well as the group working on the solution.

The team leader accomplishes *collaborative administration* that occurs in the dissemination phase. The team leader will actually *execute* the dissemination process agreed to by all interested parties.

CHAPTER 4

COLLABORATIVE PROBLEM SOLVING AND PROGRAM DEVELOPMENT SYSTEM

A framework for an integrated environment to support a group in all problem solving and program development stages including problem formulation, solution planning, solution design, solution translation, solution testing and solution delivery is proposed. This environment is based on the tools needed to support the Collaborative Cognitive Model for Collaborative Problem Solving and Program Development described in the previous chapter. The model takes into consideration the cognitive skills, psychology, and sociology and tasks that must be addressed by a team during collaborative problem solving and program development.

4.1 System Description

The complete collaborative problem solving and software development system is made up of four commercial applications. The combinations of these four applications, shown in table 4.1, supply all of the tools needed to satisfy the collaborative and cognitive activities during collaborative problem solving and software development.

The first application is Groove. Groove provides numerous tools to assist in collaboration. The Groove tools that will be utilized are as follows: a *documentation storage tool* is available to keep track of group decisions and solution plans, a *member contact tool* for easy access to team members, a *task list tool* assists in organizing the individual and group tasks of the team, a *scheduler tool* to organize meeting dates and details, a *message board* to facilitate asynchronous messaging, a *link tool* to assist in group Internet searching, and finally a *chat tool* for spontaneous communication.

CyberCollaboratory is primarily needed for its Group Decision Support System (GDSS) tools: a brainstorming tool assists team members in generating ideas for problem understanding and solution planning, an idea organizer tool, and a voting tool to facilitate group decisions.

The last two systems Rational Requisite Pro, a requirement management systems, and any Visual Source Safe, a source code management system, are essential in the overall system, however not needed until stage 3 of the collaborative model.

Table 4.1 Complete Collaborative System

	Groove	CyberCollaboratory	Rational Requisite Pro	Visual Source Safe
Brainstorming Tool	✓	✓		
Voting Tool		✓		
Document Storage Tool	✓	✓		
Idea Organizer Tool		✓		
Member Contact Tool	✓			
Task List Tool	✓			
Scheduler Tool (Calender)	✓			
Message Board /Discussion Board Tool/ Asynchronous Message Tool	✓	✓		
Chat/Synchronous Message Tool	✓	✓		
Link Tool	✓			
E-mail Tool				
Project Management/Requirements Management Tool			✓	
Code Database Tool				✓

In the following sections a description of how the tools presented above integrate with the specific tasks of the collaborative model is presented.

4.1.1 Tools for Problem Formulation

Brainstorming occurs in a few of the beginning stages of this model. In this stage the *brainstorming tool* will be used during the creation of the problem description and the refinement of the problem description. Following brainstorming the *idea organizer tool* is used to facilitate the use of the *voting tool* occurring next. This stage also utilized the *member contact tool* to create a database of the individual team members and their skills. The team will use the *asynchronous messaging tool* to keep track of the extracted facts from the refined problem. One person from the team will have the responsibility of documenting the resulting problem description and facts. This information is stored in the *document storage tool.* This stage as well as the following stages will all use the *scheduler tool* to schedule the occurrences of each of the tasks in this stage.

4.1.2 Tools for Solution Planning

Brainstorming, *idea organizing* and *voting* are a large part of this stage just as in the previous phase. Those tools are specifically needed for generating alternatives and selecting a solution strategy, and for breaking down the problem solution into major components. The major components need to be distributed to the team members. This can be accomplished using a *voting* method or team member volunteering. This distribution will be documented using the *task list tool.* Organizing the facts depicted from the last stage with the various problem components can be discussed using the *message board* and documented in the *document storage tool.* The *scheduler tool* is used for setting various deadlines in this stage.

4.1.3 Tools for Solution Design

Now that the components are distributed among the team members, the solution design is partially an individual task. The team will discuss the overall design but individual team members need to determine the breakdown of their own component. The collaborative tasks will use again, *brainstorming*, *idea organizing*, and voting. During a solution design brainstorming session, questions may arise where the *link tool* will come in handy. The team will be able to surf the Internet together. The *scheduler tool* and the *requirements management tool* will be used to set deadlines for the individual tasks of further component breakdown and algorithm logic specifications. *Asynchronous messaging* and *chat* are primarily utilized during the individual tasks for team member assistance.

4.1.4 Tools for Solution Translation

During solution translation each individual is composing the code from the algorithmic specifications from the last stage. If team member assistance is needed, *chat*, *e-mail*, and *asynchronous messaging* tools are available. The code will be stored in the *code database tool* for version control. The collaborative element of this stage is code integration. Initially, the team members will be integrating their own components and debugging. Then an integration and debugging *schedule* for the team components will bet determined. The code will be integrated from the *code database tool.* During the integration, debugging various problems may arise. These issues will be distributed among the team members using the *task list tool.* Problems should be documented using the document storage tool.

4.1.5 Tools for Solution Testing

The team needs to develop test data to determine the correctness and completeness of the solution. Individuals should determine test data for their own component but every team member should test the entire solution for effectiveness and efficiency problems. Solution problems will result in debugging tasks that will be distributed among the team members using the *task list tool.* Problems should be documented using the *document storage tool.* Team discussions will arise during the testing process possible to compare results. All of the communication tools, i.e. *chat*, *e-mail*, *messaging*, may be utilized.

4.1.6 Tools for Solution Delivery

Delivery of the solution will be a result of a presentation of the documentation collected during the problem solving and software development process and stored in the *documentation tool,* as well as delivering the actual software to the end-user.

CHAPTER 5

EXPERIMENTAL DESIGN

The impetus behind the development of a collaborative problem solving and software development model is improving the output and success of a group attempting to solve a problem with software. Most groupware systems have focused on the communication aspect of collaboration but not the coordination and cognitive issues that need to be addressed during problem solving and software development. Previous studies in this area have examined software requirement development with use of different modes of collaboration (Ocker, Hiltz, Turoff, Fjermestad, 1995; Ocker & Fjermestad 1998; Ocker et. al. 1998; Ocker, 2001) and use of a decision making model with modes of collaboration (Ocker, Hiltz, Turoff, Fjermestad, 1995). The decision making model is described as a structured approach with a sequence of 3 steps where the subjects were guided in generating alternatives alone then as a group evaluate each alternative and finally a group consensus is reached. The modes of collaboration are described as either computer conferencing alone, face-to-face or a combination of both.

The work presented in this experiment, both in terms of theoretical model and experimental design, considers a much larger aspect of the problem solving and software development process. Specifically, the focus is on the first two stages of problem solving and software development: Problem Formulation and Solution Planning. This model takes into consideration the cognitive processes of groups during these tasks. Ocker's (1995) research stated that using a "problem solving approach did not significantly impact creativity or quality" of the software requirements produced. It is this writer's opinion that there was not a significant impact because the structured method was not extensive enough to impact creativity or quality. Group Cognition was not considered.

In the problem formulation stage of the proposed model, the subjects were required to perform tasks that guide them through the phases of problem formulation: developing a preliminary problem description, a preliminary mental model and the development of a structured problem representation. During the solution planning stage, the subjects were guided to discover a solution strategy, goal decomposition and data modeling. Past research (Ocker, 1995) only covered 1 out of 6 phases that will be tested in the proposed experiment.

The experiment also utilized a collaborative system that provides all of the tools necessary to accomplish all of the tasks required to effectively and efficiently solve a problem. Specifically, these tools provide brainstorming, documentation storage, member contacts, task list, scheduler, message board, chat, synchronous messaging and a link tool. Past research utilizes a computer conferencing system to provide collaboration tools (Ocker, 2001). When Ocker (1995) tested the quality of the solution produced by the group using computer conferencing the quality was judged to be higher but not significantly so. This research hypothesized that by using the appropriate tools for the specific tasks outlined in the proposed model a significant difference will be apparent when judging quality. In this research, a plan has been developed to evaluate two stages of the model presented in the previous chapter using existing tools.

This chapter describes the evaluation plan of the collaborative problem solving and software development model and the effects of using it with specific groupware tools. An experiment to test and investigate hypotheses was conducted over two semesters. The hypotheses that were used in the evaluation, the subjects, the design, instrumentation and data collection methods are all presented.

5.1 Introduction

The groupware system that was used in this experiment is Groove. This groupware application provides the necessary tools that the subjects utilized to complete the tasks as outlined in the *problem formulation* and *solution planning stages* of the collaborative model.

This study is sought to verify the claims made regarding the collaborative problem solving and software development model and to investigate the impact resulting from using collaborative tools as a support structure for the model.

5.2 Task

The problem solving task for each group was to design a solution for a super market simulation program. Neither implementation nor coding was required, only the solution's design. This task was similar to other collaborative projects commonly assigned in graduate level object-oriented courses such as the course in which the subjects are enrolled. The final design of the supermarket should have included the different aspects of a supermarket designed using object-oriented concepts. The user of the simulation program should be able to input the customer frequency, the number of stockers re-stocking the shelves, and the number of cashiers working, where customer frequency is how often a customer will enter the store. The subjects were also required to determine any additional objects needed to simulate a supermarket and what functions all of the objects need to perform during simulation. The output of the design will be any statistical information from the different objects in the supermarket the subjects feel necessary.

The subjects had one week training. During that time, the subjects were instructed to vote on a process facilitator and a content facilitator and be able to familiarize themselves with the collaborative systems by working on a very simple problem. The *process facilitator* was responsible for initiating any activities noted in any day's tasks. The *content facilitator* was responsible for updating daily the output documents required for submission.

Following training, the subjects were given two weeks to complete the experimental task. The entire experiment lasted 3 weeks. The subjects were given a schedule to structure their time for the tasks and to allow time for documentation. The subjects were also given a post-task questionnaire that included questions used to measure subjects' perceptions regarding the task. The subject task list for each condition is located in Appendix C of this document. Documentation and the questionnaire are discussed in section 5.3.2. The subjects were also asked to participate in a debriefing session to discuss the task, the conditions, group member interactions, any time issues, any modifications needed in the experiment, and finally what they learned in the experiment. This session was conducted as an asynchronous messaging session in the subjects' course Web board conference.

5.2.1 Subjects

The subjects consisted of Computer and Information Science graduate students at the New Jersey Institute of Technology enrolled in Object Oriented Programming (CIS 601 & CIS 602). All students received course credit for their participation. Students were given an alternative task if they choose not to participate in the experiment. The alternative task was exactly the same task as given for the experiment. The alternative task is included in Appendix C of this document. Groups of four were randomly assembled for all groups whether experimental or alternative.

5.2.2 Independent variables

There were two independent variables: *tools* and *model* generating a 2X2 factorial design. Therefore, there are four conditions in this study:

(1) Access to Groove AND access to the Model

(2) Access to Groove AND no Model access

(3) E-mail AND access to the Model

(4) E-mail AND no Model access

5.2.3 Dependent variables and Data Collection

The data was obtained from multiple sources including: (1) subjects' post-test questionnaire, (2) subject performance on the given problem to be solved, (3) subjects' output documents on brainstorming sessions, alternative decisions, problem understanding and solution plan, and (4) subjects' e-mail communication. Table 5.1 outlines the dependent variables and their measurement details.

Table 5.1 Measurement Methods for Dependent Variables

Variable	**Measurement**
Problem Understanding	Output Document Analysis
Quality of Solution Planning	Output Document Analysis
Number of Alternatives	Output Document Analysis
Solution Creativity	Experts Solution Analysis
Solution Satisfaction	Ratings on Post-Task Questionnaire
Solution Quality	Experts Solution Analysis
Process Satisfaction	Ratings on Post-Task Questionnaire
Quality of E-mail Participation	E-mail Content Analysis
E-mail Message Pattern	E-mail Statistics
Process Conflict	E-mail Content Analysis

The reports/documentation required from the subjects were as follows:

1. Problem Formulation Document - This document contained the following information:
    a. The problem description their own words
    b. Any information known regarding the problem
2. Solution Plan Document - This document contained the following information:
    a. A strategy to accomplish a solution, i.e. any alternatives the teams devised with and the final alternative chosen by the team.
    b. An exact plan to accomplish the solution.
    c. Any facts associated with the plan

The post-task questionnaire, located in Appendix B, measures solution and process satisfaction as well as validating the experimental task. The questions were based on a questionnaire, also located in Appendix B, from the literature that also measured solution and process satisfaction (Ocker, Fjermestad, Hiltz, Turoff, Johnson, 1998). Table 5.2 shows the questions asked to measure solution satisfaction, Table 5.3 shows the questions to measure process satisfaction and Table 5.4 shows the task validation questions.

Table 5.2 Solution Satisfaction Questionnaire Items

1. I am very satisfied with the quality of my group's solution.
3. I am not confident in the group's final solution.
5. I am very committed to my group's final solution.
7. The final solution and formal reports do not reflect my inputs.
9. I feel I had an equal part in the correctness of the group's final solution.

Table 5.3 Process Satisfaction Questionnaire Items

2. My group's problem solving process was efficient.
4. My group's problem solving process was coordinated.
6. My group's problem solving process was unfair.
8. My group's problem solving process was confusing.
10. My group's problem solving process was satisfying.

Table 5.4 Task Validation Questionnaire Items

11. I felt the task was too difficult.
12. I understood the task.
13. I felt there wasn't enough time to complete the task.
14. I felt that everyone on my team understood the task.

Scale for questionnaire items 1,2,4,5,9,10,12,14

Strongly Agree			Undecided			Strongly Disagree
7	6	5	4	3	2	1

Scale for questionnaire items 3,6,7,8,11,13

Strongly Agree			Undecided			Strongly Disagree
1	2	3	4	5	6	7

5.2.3.1 Judging procedures. Trained expert judges were selected based on having academic and/or professional experience in software design. The judges met prior to evaluating the results for training and practice on report evaluation. The judges evaluated the solutions presented by each group from the problem solving/software development sessions. Specifically, the judges in analyzing the Problem Formulation document evaluated *problem understanding* on a separate 10-point scale. *Solution quality, creativity, quality of solution planning,* and *number of alternatives* were judged on a 10-point scale by evaluating the Solution Plan document. *Solution satisfaction* and *process satisfaction* were also judged from the post-task questionnaire given to the subjects immediately following completion of the experimental task. This questionnaire included measures to evaluate the experimental task.

Quality of participation and *process conflict* were measured by the content of the e-mails passed between the group members. Specifically, *quality of participation* was measured by comparing the number of social oriented messages and the number of task oriented messages. There was also consideration for messages that are both task and socially oriented and messages where the subjects are not following the directions as outlined in the task document. *Process conflict* was measured specifically by the quantity of disagreement and agreement messages passed between the group members. Finally, the *message pattern* variable was measured by comparing statistically the number of messages passed per day, the average number of lines per message and the average number of messages per group.

5.2.4 Hypotheses

The Hypotheses were designed to assess whether the tools in the collaborative environment and the Collaborative Model (CM) aid the subjects in their search for the solution, producing better results, enhancing their perception, attitude, and motivation, and in the development of skills and knowledge necessary for collaborative problem solving and software development. The hypotheses were designed to test the relationship between using the collaborative problem solving/program development model and various collaborative tools. The major assumption of the hypotheses was that subjects using the tools along with the CM would perform better on problem solving and software development tasks than subjects not using the system and the CM.

The tools used in this study specifically intend to facilitate problem formulation and solution planning and design tasks during collaboration. In addition, the CM outlines the tasks required during these problem solving stages. As reflected in the hypotheses formulated, the tools and CM usage are expected to directly effect the output of the teams. Table 5.5 shows the hypotheses used in this experiment.

Table 5.5 Hypotheses

H 1a. Teams working with the tools will produce **more creative solutions** than the teams working without tools.
H 1b. The teams having access to the Collaborative Model (CM) will produce **more creative solutions** than the teams working under the condition without the CM.
H 1c. When evaluating **solution creativity**, a positive synergistic effect will occur between the tools and the CM.
H 2a. The teams working with the tools will produce **higher quality solutions** than teams working without tools.
H 2b. Teams with access to the CM will produce **higher quality solutions** than the teams working under the condition without the CM.
H 2c. When evaluating **solution quality,** a positive synergistic effect will occur between the tools and the CM.
H 3a. **Solution Satisfaction** will be higher in the teams with tools than for teams working without tools.
H 3b. **Solution Satisfaction** will be higher in the teams having access to the CM than the teams working under the condition without the CM.
H 3c. When evaluating **solution satisfaction**, a positive synergistic effect will occur between the tools and the CM.
H 4a. The teams having access to the collaborative tools will show **superior understanding of the problem** as demonstrated by their ability to clearly and correctly state problems and extract problem facts better than teams without tool access.
H 4b. The teams having access to the CM will show **superior understanding of the problem** as demonstrated by their ability to clearly and correctly state problems and extract problem facts better teams without CM access.
H 4c. When evaluating **problem understanding,** a positive synergistic effect will occur between the tools and the CM.
H 5a. Teams working with the tools will **generate more alternatives** than those teams working without tools.
H 5b. The teams having access to the CM will **generate more alternatives** than the teams working under the condition without the CM.
H 5c. When evaluating the **number of alternative** generated, a positive synergistic effect will occur between the tools and the CM.
H 6a. The teams having access to the collaborative tools will show **higher quality solution planning** as demonstrated by their ability to provide detailed and clear plans, complete goal refinements and representation of facts better then the teams working under the condition without tool access.
H 6b. The teams having access to the CM will show **higher quality solution planning** as demonstrated by their ability to provide detailed and clear plans, complete goal refinements and representation of facts better then the teams working under the condition without the CM.
H 6c. When evaluating **solution planning quality**, a positive synergistic effect will occur between the tools and the CM.
H 7a. **Process Satisfaction** will be higher in the teams with tool access than that of the teams working without tool access.
H 7b. **Process Satisfaction** will be higher in the teams having access to the CM than the teams working under the condition without the CM access.
H 7c. When evaluating **process satisfaction**, a positive synergistic effect will occur between the tools and the CM.
H 8a. **Quality of e-mail participation** will be lower in teams with tool access than that of the teams working without tool access.
H 8b. **Quality of e-mail participation** will be higher in teams having access to the CM than the teams working under the condition without the CM.
H 8c. When evaluating **e-mail participation quality**, no interaction effect will occur between the tools and the CM.
H 9a. **E-mail Message Pattern** will be less complex in teams with tool access that that of the teams working without tool access.
H 9b. **E-mail Message Pattern** will be more complex in teams with CM access.
H 9c. When evaluating **e-mail message pattern** no interaction effect will occur between the tools and the CM.
H 10a. **Process Conflict** will be lower in teams with tool access that that of teams with tool access.
H 10b. **Process Conflict** will be higher in teams having access to the CM that that of teams working under the condition without the CM.
H 10c. When evaluating **process conflict** no interaction effect will occur between the tools and the CM.

5.2.4.1 Hypotheses Details. This section includes a detailed discussion of the twenty-one hypotheses. This will include the reasoning behind the hypotheses and any studies that may have influenced their selection. This section will also discuss the instruments for hypothesis verification.

Hypothesis 1 measures *solution creativity.* Based on the findings of a prior study where more creative solutions were produced by teams who had access to computer conferencing systems than those who only met face-to face (Ocker & Fjermestad, 1998), this experiment speculates that teams using the groupware tools will also produce more creative solutions than groups without tools. In addition to the tool influence, another study (Ocker et. al., 1995) concluded that a decision making model had no influence on the outcome of a group's solution. This proposal speculates that the use of the CM will produce more creative solutions due to the CM considering more extensive characteristics of the collaborative problem solving and software development process. This proposal also speculates that the combination of both the collaborative tools and the CM will produce significantly more creative solutions than conditions with only the CM or only tool access and most definitely the condition without either the tool or CM access.

Hypothesis 2 measures *solution quality.* Based on the speculations of Ocker et. al. (1995) that groups producing more creative solutions will also produce solutions with higher quality, this proposal also hypothesizes about solution quality. However, Ocker's (1995) results did not show a significant increase in solution quality when compared to solution creativity. This experiment speculates that quality will be significantly higher in groups with access to both the collaborative tools and the CM. This is based on the speculation that the CM will facilitate creativity thus improve quality. Therefore, solution quality will have a high correlation with solution creativity.

Hypothesis 3 measures *solution satisfaction.* This variable will measure the group member's opinion regarding the final group solution. The hypothesis is based on results from a collaborative experiment with experienced software engineers (Nosek, 1998), which, showed team members were more personally satisfied with their work and had greater confidence in their solutions then individual software engineers. In an experiment by Ocker et. al. (1998), there was no significant difference in solution satisfaction between the different collaborative modes of communication. However, there was not a model for the subjects to follow. It is speculated in this experiment that team members with access to groupware tools and the CM will have more satisfaction with their team's solution plan than team members without access to groupware tools and the CM.

Hypothesis 4 is measuring *Problem understanding.* This hypothesis is based on results from Deek (1997) which showed that individuals using a problem solving model which facilitated understanding the question, the meaning of the problem's terminology, and identifying relevant problem facts did better than individuals without the model. It is speculated that the same results will apply to a group where the teams exposed to the CM will understand the problem significantly better than teams without access to the CM which guides the teams through the problem understanding process.

Hypothesis 5 measures the *number of alternatives*. It is speculated that with the access to the collaborative tools the number of alternatives produced will be significantly higher than groups without access. This hypothesis is based on the fact that the addition of the collaborative brainstorming tool facilitates alternative generation; as such it is speculated that those groups having access to the tools will produce a significantly higher number of alternatives from which to choose their solution.

Hypothesis 6 measures *Quality of Solution Planning.* Based on results from Deek (1997) where individuals using a problem solving model showed higher quality solution planning as opposed to individuals without access to a problem solving model. This proposal speculates that significantly higher quality solution planning will result in groups that have access to the CM as opposed to groups without access. The subjects will be developing an appropriate strategy for a solution by following the tasks outlined in the CM. The subjects will be guided in considering various alternatives to achieve the goal of the problem by subdividing the goal into subgoals, and identifying the tasks needed to accomplish each subgoal.

Hypothesis 7 measures *process satisfaction.* Process satisfaction measures the satisfaction experienced by groups that accomplished the same task via different means. This measure is based on Ocker et. al. 1998 where process satisfaction was measured and there was not a significant difference between the modes of communication. This proposal speculates that there will be a significant increase in process satisfaction between those groups with access to the tools and the CM than the teams without access to either the collaborative tools and the CM. Ocker's experiment did not include a model for the groups to follow thus this proposal hypothesizes that since the CM facilitates problem solving process satisfaction will increase.

Hypothesis 8 measures *quality of participation.* Quality of participation will be analyzed by doing a content analysis of the e-mails passed between the groups. E-mails will be coded in 4 different categories: social, task oriented, both social and task oriented, and off-track e-mails (not following directions). The higher the number of task and task/social messages will increase the score of that group. It is speculated that groups with only e-mail access and the CM will have a higher outcome for this variable because the groups with CM access will have more motivation to communicate that groups without CM access.

Hypothesis 9 measures the *e-mail message pattern.* E-mail message pattern is accomplished by measuring the number of messages passed per day, the average number of lines per message, and the average number of messages passed per group member. Teams with only CM access are hypothesized to have a more complex e-mail message pattern than groups working under the other conditions. The CM will promote more frequent and detailed e-mail discussions.

Hypothesis 10 measures *process conflict.* Process conflict will be evaluated by determining the number of disagreement and agreement messages passed between group members. It is hypothesized that teams having access to the CM will have a higher amount of conflict than the other conditions. Because of the hypothesized more frequent and detailed e-mail discussions occurring there could be a higher probability of conflict.

Ten 2X2 ANOVAs will be performed on the data collected. To correct for possible type 1 errors, a step-wise Bonferonni procedure will be employed.

The highest F observed value will be examined at Alpha .007 or .05/7. The next highest F observed will be examined at .0083 or .05/6.

5.2.4.2 Theoretical Model of the Hypotheses. The hypothesized model shown in Figure 5.1 depicts the predicted relationships and how each variable combines to form a total model. The model shows that the uses of the collaborative tools and the collaborative model together have a direct effect on all variables. It is speculated that solution planning quality will have a high correlation with solution quality. Solution quality is also highly correlated to number of alternatives and problem understanding. Problem Understanding will also correlate highly with solution creativity. In addition, process satisfaction is speculated to have a high correlation with solution satisfaction. A Structural Model of the data collected will be performed to test the model for consistency with the data collected.

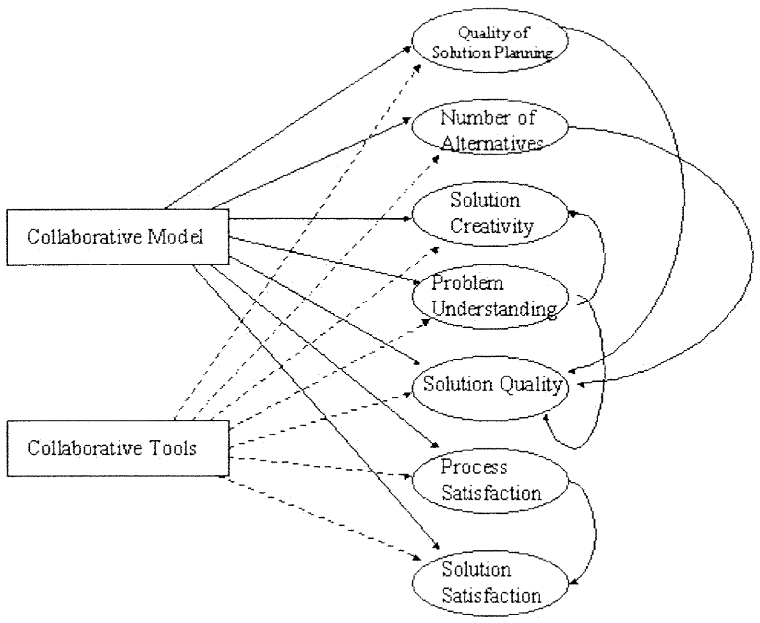


Figure 5.1 Theoretical Model

CHAPTER 6

EXPERIMENTAL RESULTS AND DATA ANALYSIS

An experiment to measure the effectiveness of the collaborative problem solving model coupled with groupware tools was conducted over two semesters. The analysis performed was compared against a set of hypotheses. Each experiment lasted 3 weeks including a 3-day training session. The subjects were randomly placed into groups of four; then each group was placed in one of the four conditions.

Data were collected from three main sources: group written documents rated by expert judges, a post task questionnaire presented at the end of the experiment, and the categorization of e-mails passed between group members. The analysis of this data, performed using a collection of statistical procedures, is presented in this chapter.

6.1 Descriptive Statistics

The experiment took place in the Fall 2001 and Spring 2002 semesters. The subject groups were placed into one of the four conditions:

1. Access to Groove AND access to the Model
2. Access to Groove AND no Model access
3. E-mail AND access to the Model
4. E-mail AND no Model access

There were 57 subjects who completed in the fall semester pilot study. These subjects were placed into groups of 4. For the fall semester there were 5 groups in condition 1, 4 groups in condition 2, 4 groups in condition 3 and 2 groups in condition 4. Originally, there were 4 groups in condition 4 but those subjects either dropped the course or opted for the alternate task.

There were 174 subjects who completed the spring semester experiment, equating to 12 groups in condition 1, 10 groups in condition 2, 11 groups in condition 3, and 11 groups in condition 4. None of the groups dropped out in the spring semester. All subjects were students of graduate C++ and JAVA courses.

6.2 Inter-rater Reliability

The two expert judges, who were blind to the experimental conditions, were each given the two group output documents to evaluate: Problem Understanding document and Solution Plan document. In the Problem Formulation document the subjects evaluated the problem by restating it in their own words and organized any information they knew regarding the problem. The judges were to evaluate the groups' *problem understanding* on a scale of 1 to 10 where 10 was the best score. By evaluating the Solution Plan document the judges evaluated separately *solution quality*, *solution creativity*, *quality of solution planning*, and *number of alternatives* using the same scale. The judges were also responsible for evaluating the *quality of e-mail participation* and *e-mail process conflict* dependent variables. The judges were each given a copy of every e-mail passed between the group members. To evaluate the quality of e-mail participation, they categorized the e-mails into 3 categories: task oriented, socially oriented, both task and socially oriented. To evaluate process conflict the judges categorized the e-mails as agreement or disagreement.

The results from both judges were evaluated to determine if they were trained properly and to determine the reliability and validity of their evaluations of the dependent variables. An inter-rater reliability check was performed with a bivariate Pearson 2-tailed test. It was found that there was a significant correlation at the .01 level between the two judges (r = .932, p < .01), shown in Table 6.1.

Table 6.1 Correlation Between Judges

Correlations

		RATERONE	RATERTWO
RATERONE	Pearson Correlation		
	Sig. (2-tailed)		
	N		
RATERTWO	Pearson Correlation	.932**	
	Sig. (2-tailed)	.000	
	N	135	

**. Correlation is significant at the 0.01 level (2-tailed).

The results of a Paired Samples T-Test was also performed to show no significant difference between the judges is shown in Table 6.2.

Table 6.2 T-Test for Judge's Significant Difference

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
					95% Confidence Interval of the Difference				
		Mean	Std. Deviation	Std. Error Mean	Lower	Upper			
Pair 1	RATERONE - RATERTWO	.3222	5.5780	.4801	-.6273	1.2717	.671	134	.503

6.3 Pilot Hypotheses Analysis

Thirteen 2X2 ANOVAs were performed on the pilot data collected as well as Factor Analysis on post-task questionnaire data and presented in this section. The pilot study, performed in the Fall 2001 semester, was the same as the experimental study performed in the Spring 2002 semester. There were not any problems discovered with the task given during the pilot study, therefore, nothing was changed for the spring semester experiment.

6.3.1 Problem Understanding Results

The problem understanding variable was measured by the judges' evaluation of the Problem Formulation document that was written by each group. Table 6.3 shows the results from an ANOVA evaluation of the problem understanding dependent variable.

Table 6.3 Problem Understanding Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 7.2 SD: 1.39	Mean: 6.8 SD: 1.14	7
NO TOOL	Mean: 7.13 SD: .75	Mean: 5.0 SD: 3.5	6.07
ALL	7.17	5.9	Grand Mean: 6.53

Tools:
F = 1.27 p = .284
Model:
F = 2.24 p = .162
Tools X Model:
F = 1.07 p = .322

These results show no significance. Hypothesis H 4a, H 4b, and H 4c were not supported.

6.3.2 Quality of Solution Planning Results

The quality of solution planning variable was measured by the judges' evaluation of the Solution Plan document that was written by each group. Table 6.4 shows the results from an ANOVA evaluation of quality of solution planning dependent variable.

Table 6.4 Quality of Solution Planning Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 6.65 SD: .38	Mean: 4.25 SD: 2.89	5.45
NO TOOL	Mean: 7.06 SD: 6.83	Mean: 5.6 SD: 1.59	6.33
ALL	6.86	4.93	Grand Mean: 5.89

Tools:
F = 1.0 p = .34
Model:
F = 4.61 p = .055
Tools X Model:
F = .29 p = .601

These results show no significance. Hypothesis H 6a, H 6b, and H 6c were not supported.

6.3.3 Number of Alternatives Results

The Number of Alternatives variable was measured by the judges' evaluation of the Solution Plan document that was written by each group. An ANOVA evaluation showed a significant difference in teams that did not have access to Groove compared to teams that did have access to Groove. The teams without access created a higher number of alternatives then teams that did have access. There was also a significant interaction between the tool and model. Table 6.5 shows the results from an ANOVA evaluation of the number of alternatives dependent variable.

Table 6.5 Number of Alternatives Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 1.5 SD: .5	Mean: 1.0 SD: .41	1.25
NO TOOL	Mean: 1.5 SD: .41	Mean: 2.5 SD: .71	2
ALL	1.5	1.75	Grand Mean: 1.63

Tools:
F = 8.25 p = .015
Model:
F = .917 p = .359
Tools X Model:
F = 8.25 p = .015

The interaction effect was further evaluated by doing a Post Hoc Bonferroni procedure to determine where the interactions lie. Six independent T-tests were performed. Zero of the six possible tests showed significance.

6.3.4 Solution Creativity Results

The Solution Creativity variable was measured by the judges' evaluation of the Solution Plan document that was written by each group. The results showed no significance with any of the independent variables as shown in Table 6.6. Therefore, hypotheses H1a, H1b, and H1c were not supported.

Table 6.6 Solution Creativity Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 7.85 SD: .78	Mean: 5.06 SD: 2.99	6.46
NO TOOL	Mean: 7.44 SD: .83	Mean: 7.75 SD: 1.77	7.6
ALL	7.65	6.41	Grand Mean: 7.03

Tools:
F = 1.38 p = .265
Model:
F = 1.63 p = .228
Tools X Model:
F = 2.56 p = .14

6.3.5 Solution Quality Results

The Solution Quality variable was measured by the judges' evaluation of the Solution Plan document that was written by each group. The results, shown in Table 6.7 showed no significance with any of the independent variables. Therefore, hypotheses H2a, H2b, and H2c were not supported.

Table 6.7 Solution Quality Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 7.2 SD: 1.04	Mean: 5.38 SD: 3.35	6.29
NO TOOL	Mean: 7.25 SD: .96	Mean: 7.5 SD: 1.41	7.38
ALL	7.23	6.44	Grand Mean: 6.83

Tools:
F = 1.02 p = .34
Model:
F = .532 p = .481
Tools X Model:
F = .924 p = .357

6.3.6 Task Oriented E-mail Message Results

The Task Oriented e-mail variable was evaluated by the judges' categorization of the e-mails passed between group members during the experiment. The e-mails were judged to be either socially oriented or task oriented. The results of the categorization were then evaluated by performing 2X2 ANOVAs on both variables. The results of the task oriented e-mails are shown in Table 6.8. The results of the socially oriented are discussed in section 6.3.7.

Table 6.8 Task E-mails Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 35.0 SD: 22.99	Mean: 15.0 SD: 7.4	25
NO TOOL	Mean: 59.75 SD: 74.72	Mean: 34.0 SD: 9.9	46.88
ALL	47.38	24.5	Grand Mean: 35.94

Tools:
F = .92 p = .36
Model:
F = 1.0 p = .34
Tools X Model:
F = .02 p = .90

These results showed that all three hypotheses, H 7a, H 7b, and H 7c were not supported.

6.3.7 Socially Oriented E-mail Message Results

The Socially Oriented e-mail variable was evaluated by the judges' categorization of the e-mails passed between group members during the experiment. The e-mails were judged to be either socially oriented or task oriented. The results of the categorization were then evaluated by performing 2X2 ANOVAs on both variables. The results of the socially oriented e-mails are shown in Table 6.9.

Table 6.9 Social E-Mail Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 2.5 SD: 2.89	Mean: 1.13 SD: 1.11	1.82
NO TOOL	Mean: 9.25 SD: 13.19	Mean: 1.25 SD: 1.06	5.25
ALL	5.88	1.19	Grand Mean: 3.53

Tools:
F = .77 p = .40
Model:
F = 1.44 p = .26
Tools X Model:
F = .72 p = .42

These results showed that all three hypotheses, H 8a, H 8b, and H 8c were not supported.

6.3.8 Average E-mail Messages Passed Per Day Results

E-mail Messages passed per day was evaluated by averaging the total number of e-mails that were passed by each group. This variable was evaluated by performing a 2X2 ANOVA. The results, indicating no significance are shown in Table 6.10. Hypotheses H 10a, H10 b, and H 10 c were not supported.

Table 6.10 Average E-mail Message Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 4.1 SD: 3.72	Mean: 1.55 SD: .98	2.83
NO TOOL	Mean: 11.48 SD: 15.25	Mean: 4.95 SD: .78	8.22
ALL	7.79	3.25	Grand Mean: 5.52

Tools:
F = 1.41 p = .26
Model:
F = 1.0 p = .34
Tools X Model:
F = .19 p = .67

6.3.9 Average Number of Lines Per E-mail Passed Per Group Results

Average number of lines per e-mail was evaluated by performing a 2X2 ANOVA. The results are shown in Table 6.11 indicate no significance. Hypotheses H 11a, H 11b, and H 11c were not supported.

Table 6.11 Average Number of E-mail Lines Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 6.37 SD: 4.29	Mean: 5.08 SD: 1.54	5.73
NO TOOL	Mean: 5.27 SD: 4.18	Mean: 9.31 SD: .58	7.29
ALL	5.82	7.2	Grand Mean: 6.51

Tools:
F = .67 p = .43
Model:
F = .52 p = .49
Tools X Model:
F = 1.95 p = .19

6.3.10 Total E-mail Messages Passed Per Group Results

Total number of e-mails was evaluated by performing a 2X2 ANOVA. The results shown in Table 6.12 indicate no significance. Hypotheses H 12 a, H 12 b and H 12 c were not supported.

Table 6.12 Total E-mail Messages Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 41 SD: 37.17	Mean: 15.5 SD: 9.85	28.25
NO TOOL	Mean: 114.75 SD: 152.53	Mean: 49.5 SD: 7.78	82.13
ALL	77.88	32.5	Grand Mean: 55.19

Tools:
F = 1.46 p = .26
Model:
F = 1.0 p = .34
Tools X Model:
F = .19 p = .67

6.3.11 E-mail Process Conflict Results

E-mail process conflict was evaluated by performing a 2X2 ANOVA. The results shown in Table 6.13 indicate no significance. Hypotheses H 9a, H 9b and H 9c were not supported.

Table 6.13 E-mail Process Conflict Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 1.1 SD: 1.67	Mean: 0 SD: 0	.55
NO TOOL	Mean: .63 SD: .63	Mean: 1.0 SD: 1.41	.82
ALL	.87	.5	Grand Mean: .69

Tools:
F = .18 p = .68
Model:
F = .34 p = .57
Tools X Model:
F = 1.39 p = .26

6.3.12 Questionnaire Evaluation

The post task questionnaire, shown in Appendix B, was evaluated via a Factor Analysis and Chronbach's Alpha to determine the factors; then an ANOVA was done on the resulting scales.

The results from the Factor Analysis, using a factor loading of .55 or greater and an Eigen value greater than one, were three scales. Scale one included questions 1, 3, 5, 8, 10, 12, and 14. It had an Eigen value of 6.5 explaining 46.4% of the variance un-rotated. Scale one, when rotated, has an Eigen value of 3.99 explaining 28.5% of the variance. Scale two included questions 2, 4, 6, 7, and 10. It had an Eigen value of 1.55 explaining 11.06% of the variance un-rotated. Scale two, when rotated, has an Eigen value of 2.76 explaining 19.74% of the variance. Scale three included questions 4, 11, 13, and 14. It had an Eigen value of 1.24 explaining 8.88% of the variance. Scale three, when rotated, has an Eigen value of 2.54 explaining 18.12% of the variance. The three scales combined accounted for 66.36% of the variance.

A Chronbach's Alpha evaluation was performed resulting in a value of .9. This value shows a high internal consistency for the questionnaire.

To complete the analysis, an ANOVA was performed on the scale one which will be compared to Solution Satisfaction questions. The results, shown in Table 6.14, indicate no significant results. Therefore, the hypotheses were not supported.

Table 6.14 Solution Satisfaction Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 38.94 SD: 8.89	Mean: 39.0 SD: 5.58	38.97
NO TOOL	Mean: 38.83 SD: 7.44	Mean: 35.08 SD: 11.98	36.96
ALL	38.89	37.04	Grand Mean: 37.96

Tools:
F = .804 p = .374
Model:
F = .677 p = .414
Tools X Model:
F = .718 p = .4

An ANOVA was performed on the scale two which will be compared to the Process Satisfaction questions. The results, shown in Table 6.15, indicate no significance. Therefore, the hypotheses were not supported.

Table 6.15 Process Satisfaction Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 28.72 SD: 5.91	Mean: 26.15 SD: 7.76	27.42
NO TOOL	Mean: 28.89 SD: 4.96	Mean: 26.42 SD: 7.49	27.66
ALL	28.81	26.29	Grand Mean: 27.55

Tools:
F = .016 p = .898
Model:
F = 2.27 p = .14
Tools X Model:
F = .001 p = .98

An ANOVA was also performed on the scale three which will be compared to Task Validation questions. The results, shown in Table 6.16, indicate no significance.

Table 6.16 Task Validation Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 21.5 SD: 5.55	Mean: 18.77 SD: 5.51	20.14
NO TOOL	Mean: 20.44 SD: 4.02	Mean: 19.0 SD: 6.22	19.72
ALL	20.97	18.89	Grand Mean: 19.93

Tools:
F = .09 p = .765
Model:
F = 2.3 p = .135
Tools X Model:
F = .219 p = .642

6.3.13 Dependent Variable Correlations

The model shown in the previous chapter (Figure 5.1) depicted correlations predicted between some of the dependent variables. A path analysis of the data collected was performed to test the model for consistency. The hypothesized model depicted the predicted relationships and how each variable combines to form a total model. The result of the analysis was a Chi-Square value of .79 equating to a .37 significance indicating that there was not a significant difference between the hypothesized model and the actual model. Figure 6.1 shows the actual correlation values. Any value over 2.0 was considered significant.

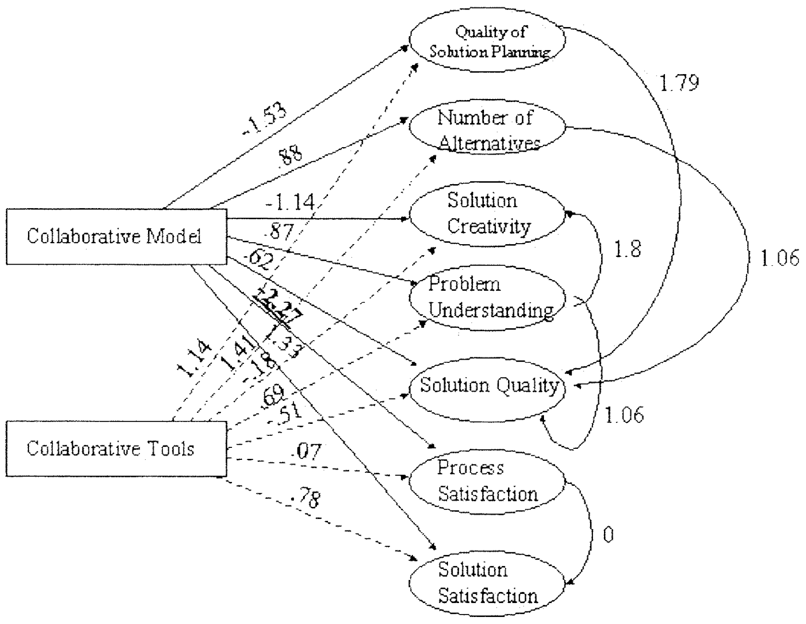


Figure 6.1 Actual Path Model for Pilot.

As shown in the actual path model there was a correlation between the model and process satisfaction.

6.3.14 Pilot Study Summary

There were 14 ANOVA evaluations performed on the pilot data. There was only one significant variable result that could be a reflection of the small number of groups per condition and an uneven distribution of groups among the conditions due to subject dropouts. The pilot study did however prove the task to be sound and sufficient for the experimental study.

6.4 Experiment Hypotheses Analysis

Twelve 2X2 ANOVAs were performed on the data collected as well as a Factor Analysis on post-task questionnaire data. The results were compared against the hypotheses formulated and presented in the following sections.

6.4.1 Problem Understanding Results

The problem understanding variable was measured by the judges' evaluation of the Problem Formulation document that was written by each group. The results of an ANOVA evaluation of the data showed a .017 significant difference of the problem understanding between subjects exposed to the collaborative model and the subjects not exposed to the model. Where subjects that used the collaborative model had a higher problem understanding then the subjects that did not have access to the collaborative model. Table 6.17 shows the results from an ANOVA evaluation of the problem understanding dependent variable.

Table 6.17 Problem Understanding Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 6.25 SD: 2.27	Mean: 4.73 SD: 2.67	5.49
NO TOOL	Mean: 7.25 SD: 1.83	Mean: 4.89 SD: 3.36	6.07
ALL	6.75	4.81	Grand Mean: 5.81

Tools:
F = .553 p = .462
Model:
F = 6.196 p = .017
Tools X Model:
F = .288 p = .594

These results supported the main effect, hypothesis H4b. Hypothesis H4a and H4c were not supported.

6.4.2 Quality of Solution Planning Results

The quality of solution planning variable was measured by the judges' evaluation of the Solution Plan document that was written by each group. The results of an ANOVA evaluation of the data showed a .007 significant difference in the quality of solution planning between subjects exposed to the collaborative model and the subjects not exposed to the model. Table 6.18 shows the subjects that used the collaborative model scored higher for the quality of solution planning than the subjects that did not use the model.

Table 6.18 Quality of Solution Planning Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 7.0 SD: 1.81	Mean: 4.3 SD: 2.87	5.65
NO TOOL	Mean: 6.64 SD: 1.87	Mean: 5.23 SD: 2.85	5.94
ALL	6.82	4.77	Grand Mean: 5.79

Tools:
F = .154 p = .697
Model:
F = 8.172 p = .007
Tools X Model:
F = .807 p = .375

These results supported the main effect, hypothesis H6b. Hypothesis H6a and H6c were not supported.

6.4.3 Number of Alternatives Results

The number of alternatives variable was measured by the judges' evaluation of the Solution Plan document that was written by each group. The results of an ANOVA evaluation test showed an interaction significance of .045 for access to the collaborative model and Groove. Table 6.19 shows that the groups with access to the collaborative model and no access to Groove presented the most solution alternatives for the task. The interaction that occurred with the groups having access to Groove and no access to the collaborative model and the groups with access to the model and no access to Groove.

Table 6.19 Number of Alternatives Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 1.7 SD: .39	Mean: 1.6 SD: 1.35	1.65
NO TOOL	Mean: 1.82 SD: .98	Mean: 1.18 SD: .40	1.5
ALL	1.76	1.39	Grand Mean: 1.58

Tools:
F = .203 p = .655
Model:
F = .154 p = .697
Tools X Model:
F = 4.27 p = .045

The interaction effect was further evaluated by doing a Post Hoc Bonferroni procedure to determine where the interactions lie. Six independent T-tests were performed. Four of the six possible tests showed significance. The first significant result is the analysis of condition 1 (tool + model) with condition 2 (tool + no model). Table 6.20 shows the group statistics. Table 6.21 shows the results from the T-test, which is an F value of 6.47 translating to a significance of .02.

Table 6.20 Group Statistics (condition 1 and condition 2)

	COND	N	Mean	Std. Deviation	Std. Error Mean
NUMAL	1.00	12	1.1667	.3892	.1124
	2.00	10	1.6000	1.3499	.4269

Table 6.21 Independent Samples T-Test (condition 1 + condition 2)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
NUMAL	Equal variances assumed	6.407	.020	-1.065	20	.300	-.4333	.4070	-1.2822	.4156
	Equal variances not assumed			-.982	10.250	.349	-.4333	.4414	-1.4136	.5470

The second significant result was between condition 1 (tool + model) and condition 3 (no tool + model). Table 6.22 shows the group statistics. Table 6.23 shows the results from the T-test, which is an F value of 6.1 translating to a significance of .022.

Table 6.22 Group Statistics (condition 1 + condition 3)

	COND	N	Mean	Std. Deviation	Std. Error Mean
NUMAL	1.00	12	1.1667	.3892	.1124
	3.00	11	1.8182	.9816	.2960

Table 6.23 Independent Samples T-Test (condition 1 + condition 3)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
NUMAL	Equal variances assumed	6.099	.022	-2.127	21	.045	-.6515	.3062	-1.2884	-1.47E-02
	Equal variances not assumed			-2.058	12.848	.060	-.6515	.3166	-1.3363	3.326E-02

The third significant result was between condition 2 (tool + no model) and condition 4 (no tool + no model). Table 6.24 shows the group statistics. Table 6.25 shows the results from the T-test, which is an F value of 5.57 translating to a significance of .029.

Table 6.24 Group Statistics (condition 2 + condition 4)

Group Statistics

	COND	N	Mean	Std. Deviation	Std. Error Mean
NUMAL	2.00	10	1.6000	1.3499	.4269
	4.00	11	1.1818	.4045	.1220

Table 6.25 Independent Samples T-Test (condition 2 + condition 4)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
NUMAL	Equal variances assumed	5.567	.029	.982	19	.338	.4182	.4257	-.4728	1.3092
	Equal variances not assumed			.942	10.467	.367	.4182	.4440	-.5651	1.4014

The fourth and final significant result was between condition 3 (no tool + model) and condition 4 (no tool + no model). Table 6.26 shows the group statistics. Table 6.27 shows the results from the T-test, which is an F value of 5.2 translating to a significance of .034.

Table 6.26 Group Statistics (condition 3 + condition 4)

Group Statistics

	COND	N	Mean	Std. Deviation	Std. Error Mean
NUMAL	3.00	11	1.8182	.9816	.2960
	4.00	11	1.1818	.4045	.1220

Table 6.27 Independent Samples T-Test (condition 3 + condition 4)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
NUMAL	Equal variances assumed	5.200	.034	1.988	20	.061	.6364	.3201	-3.14E-02	1.3041
	Equal variances not assumed			1.988	13.301	.068	.6364	.3201	-5.36E-02	1.3264

These results indicate that either the tool or the model alone was significantly better. However, the combination or the absence of was not significantly better. Hypothesis H 5c was supported.

6.4.4 Solution Creativity Results

The Solution Creativity variable was measured by the judges' evaluation of the Solution Plan document that was written by each group. The results showed no significance with any of the independent variables as shown in Table 6.28. Therefore, hypotheses H1a, H1b, and H1c were not supported.

Table 6.28 Solution Creativity Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 7.63 SD: .933	Mean: 6.7 SD: 2.18	7.163
NO TOOL	Mean: 7.14 SD: 1.47	Mean: 7.41 SD: 1.0	7.273
ALL	7.38	7.06	Grand Mean: 7.22

Tools:
F = .064 p = .802
Model:
F = .557 p = .460
Tools X Model:
F = 1.88 p = .178

6.4.5 Solution Quality Results

The Solution Quality variable was measured by the judges' evaluation of the Solution Plan document that was written by each group. The results, shown in Table 6.29

showed no significance with any of the independent variables. Therefore, hypotheses H2a, H2b, and H2c were not supported.

Table 6.29 Solution Quality Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 7.71 SD: .941	Mean: 7.35 SD: 1.6	7.53
NO TOOL	Mean: 6.91 SD: 1.67	Mean: 6.91 SD: 1.77	6.91
ALL	7.31	7.13	Grand Mean: 7.22

Tools:
F = 1.83 p = .184
Model:
F = .153 p = .698
Tools X Model:
F = .153 p = .698

6.4.6 Task Oriented E-mail Message Results

The Task Oriented e-mail variable was evaluated by the judges' categorization of the e-mails passed between group members during the experiment. The e-mails were judged to be either socially oriented or task oriented. The results of the categorization were then evaluated by performing 2X2 ANOVAs on both variables. The results of the task-oriented e-mails are shown in Table 6.30. The results of the socially oriented are discussed in section 6.4.9.

Table 6.30 Task Oriented E-mail Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 28.17 SD: 16.85	Mean: 20.85 SD: 15.99	24.51
NO TOOL	Mean: 75.41 SD: 48.67	Mean: 22.41 SD: 7.87	48.91
ALL	51.79	21.63	Grand Mean: 36.71

Tools:
F = 8.773 p = .005
Model:
F = 13.402 p = .001
Tools X Model:
F = 7.688 p = .008

These results showed that subjects without access to Groove passed more task-oriented e-mails and subjects with access to the collaborative model passed more task-oriented e-mails. The significant interaction effect was further evaluated by doing a Post Hoc Bonferroni procedure to determine where the interactions lie. Six independent T-tests were performed. Four of the six possible tests showed significance. The first significant result is the analysis of condition 1 (tool + model) with condition 3 (no tool + model). Table 6.31 shows the group statistics. Table 6.32 shows the results from the T-test, which is an F value of 8.114 translating to a significance of .010.

Table 6.31 Group Statistics (condition 1 + condition 3)

	COND	N	Mean	Std. Deviation	Std. Error Mean
QUAL	1	12	28.1667	16.8514	4.8646
	3	11	75.4091	48.6733	14.6756

Table 6.32 Independent Samples T-Test (condition 1 + condition 3)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
QUAL	Equal variances assumed	8.114	.010	-3.167	21	.005	-47.2424	14.9160	-78.2620	-16.2229
	Equal variances not assumed			-3.056	12.185	.010	-47.2424	15.4608	-80.8721	-13.6127

The second significant result was between condition 1 (tool + model) and condition 4 (no tool + no model). Table 6.33 shows the group statistics. Table 6.34 shows the results from the T-test, which is an F value of 4.95 translating to a significance of .037.

Table 6.33 Group Statistics (condition 1 + condition 4)

	COND	N	Mean	Std. Deviation	Std. Error Mean
QUAL	1	12	28.1667	16.8514	4.8646
	4	11	22.4091	7.8703	2.3730

Table 6.34 Independent Samples T-Test (condition 1 + condition 4)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
QUAL	Equal variances assumed	4.952	.037	1.033	21	.313	5.7576	5.5729	-5.8319	17.3471
	Equal variances not assumed			1.064	15.869	.303	5.7576	5.4125	-5.7241	17.2392

The third significant result was between condition 2 (tool + no model) and condition 3 (no tool + model). Table 6.35 shows the group statistics. Table 6.36 shows the results from the T-test, which is an F value of 7.6 translating to a significance of .013.

Table 6.35 Group Statistics (condition 2 + condition 3)

	COND	N	Mean	Std. Deviation	Std. Error Mean
QUAL	2	10	20.8500	15.9931	5.0575
	3	11	75.4091	48.6733	14.6756

Table 6.36 Independent Samples T-Test (condition 2 + condition 3)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
QUAL	Equal variances assumed	7.602	.013	-3.376	19	.003	-54.5591	16.1608	-88.3841	-20.7340
	Equal variances not assumed			-3.515	12.323	.004	-54.5591	15.5226	-88.2817	-20.8364

The fourth and final significant result was between condition 3 (no tool + model) and condition 4 (no tool + no model). Table 6.37 shows the group statistics. Table 6.38 shows the results from the T-test, which is an F value of 13.59 translating to a significance of .001.

Table 6.37 Group Statistics (condition 3 + condition 4)

	COND	N	Mean	Std. Deviation	Std. Error Mean
QUAL	3	11	75.4091	48.6733	14.6756
	4	11	22.4091	7.8703	2.3730

Table 6.38 Independent Samples T-Test (condition 3 + condition 4)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
QUAL	Equal variances assumed	13.587	.001	3.565	20	.002	53.0000	14.8662	21.9897	84.0103
	Equal variances not assumed			3.565	10.523	.005	53.0000	14.8662	20.0977	85.9023

This data shows the model alone caused the interaction effect on all other conditions. In addition, the combination of the model and the tool caused an interaction when compared to the condition with the tool and model absent.

These results showed that all three hypotheses, H 7a, H 7b, and H 7c were supported.

6.4.7 Socially Oriented E-mail Message Results

The Socially Oriented e-mail variable was evaluated by the judges' categorization of the e-mails passed between group members during the experiment. The e-mails were judged to be either socially oriented or task oriented. The results of the categorization were then evaluated by performing 2X2 ANOVAs on both variables. The results of the socially oriented e-mails are shown in Table 6.39.

Table 6.39 Socially Oriented E-mails Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 11.42 SD: 5.73	Mean: 10.9 SD: 9.64	11.16
NO TOOL	Mean: 18.23 SD: 7.97	Mean: 5.91 SD: 4.01	12.07
ALL	14.83	8.41	Grand Mean: 11.62

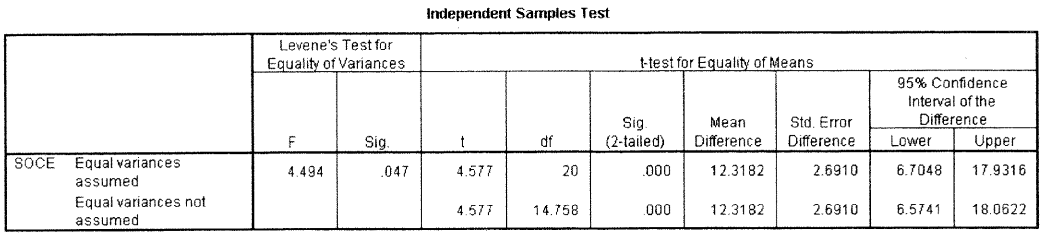
Tools:
F = .182 p = .672
Model:
F = 9.053 p = .005
Tools X Model:
F = 7.654 p = .009

These results showed that subjects with access to the collaborative model passed more social e-mails. The significant interaction effect was further evaluated by doing a Post Hoc Bonferroni procedure to determine where the interactions lie. Six independent T-tests were performed. One of the six possible tests showed significance. The significant result was from condition 3 (no tool + model) with condition 4 (no tool + no model). Table 6.40 shows the group statistics. Table 6.41 shows the results from the T-test, which is an F value of 4.49 translating to a significance of .047.

Table 6.40 Group Statistics (condition 3 + condition 4)

	COND	N	Mean	Std. Deviation	Std. Error Mean
SOCE	3	11	18.2273	7.9730	2.4039
	4	11	5.9091	4.0113	1.2095

Table 6.41 Independent Samples T-Test (condition 3 + condition 4)



Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
SOCE	Equal variances assumed	4.494	.047	4.577	20	.000	12.3182	2.6910	6.7048	17.9316
	Equal variances not assumed			4.577	14.758	.000	12.3182	2.6910	6.5741	18.0622

These results showed that the condition having access to only the model and only e-mail caused the interaction when compared to the condition with neither access to Groove nor the collaborative model. These results supported two hypotheses: H 8b and H 8c.

6.4.8 Average E-mail Messages Passed Per Day Results

E-mail Messages passed per day was evaluated by averaging the total number of e-mails that were passed by each group. This variable was evaluated by performing a 2X2 ANOVA. The results are shown in Table 6.42.

Table 6.42 Average E-mail Messages Passed Per Day Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 3.6 SD: 2.18	Mean: 2.04 SD: 2.04	2.82
NO TOOL	Mean: 7.1 SD: 4.3	Mean: 2.33 SD: .72	4.715
ALL	5.35	2.19	Grand Mean: 3.77

Tools:
F = 4.185 p = .047
Model:
F = 13.123 p = .001
Tools X Model:
F = 5.421 p = .025

These results showed that without the tool the groups passed more e-mails per day. In addition, the groups with access to the CM passed more e-mails per day. There was also a significant interaction effect. The significant interaction effect was further evaluated by doing a Post Hoc Bonferroni procedure to determine where the interactions lie. Six independent T-tests were performed. Five of the six tests showed significance. The first significant result is the analysis of condition 1 (tool + model) with condition 3 (no tool + model). Table 6.43 shows the group statistics. Table 6.44 shows the results from the T-test, which is an F value of 4.93 translating to a significance of .037.

Table 6.43 Group Statistics (condition 1 + condition 3)

	Condition	N	Mean	Std. Deviation	Std. Error Mean
MESSPER	1.00	12	3.5950	2.1771	.6285
	3.00	11	7.0964	4.3119	1.3001

Table 6.44 Independent Samples T-Test (condition 1 + condition 3)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
MESSPERD	Equal variances assumed	4.933	.037	-2.491	21	.021	-3.5014	1.4055	-6.4242	-.5786
	Equal variances not assumed			-2.425	14.500	.029	-3.5014	1.4440	-6.5885	-.4142

The second significant result was between condition 1 (tool + model) and condition 4 (no tool + no model). Table 6.45 shows the group statistics. Table 6.46 shows the results from the T-test, which is an F value of 7.18 translating to a significance of .014.

Table 6.45 Group Statistics (condition 1 + condition 4)

	Condition	N	Mean	Std. Deviation	Std. Error Mean
MESSPER	1.00	12	3.5950	2.1771	.6285
	4.00	11	2.3327	.7216	.2176

Table 6.46 Independent Samples T-Test (condition 1 + condition 4)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
MESSPERD	Equal variances assumed	7.180	.014	1.830	21	.081	1.2623	.6898	-.1722	2.6968
	Equal variances not assumed			1.898	13.580	.079	1.2623	.6651	-.1683	2.6929

The third significant result was between condition 2 (tool + no model) and condition 3 (no tool + model). Table 6.47 shows the group statistics. Table 6.48 shows the results from the T-test, which is an F value of 5.29 translating to a significance of .033.

Table 6.47 Group Statistics (condition 2 + condition 3)

	Condition	N	Mean	Std. Deviation	Std. Error Mean
MESSPER	2.00	10	2.5590	2.0411	.6454
	3.00	11	7.0964	4.3119	1.3001

Table 6.48 Independent Samples T-Test (condition 2 + condition 3)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
MESSPERD	Equal variances assumed	5.292	.033	-3.028	19	.007	-4.5374	1.4983	-7.6734	-1.4014
	Equal variances not assumed			-3.126	14.554	.007	-4.5374	1.4515	-7.6394	-1.4353

The fourth significant result was between condition 2 (tool + no model) and condition 4 (no tool + no model). Table 6.49 shows the group statistics. Table 6.50 shows the results from the T-test, which is an F value of 4.36 translating to a significance of .05.

Table 6.49 Group Statistics (condition 2 + condition 4)

	Condition	N	Mean	Std. Deviation	Std. Error Mean
MESSPER	2.00	10	2.5590	2.0411	.6454
	4.00	11	2.3327	.7216	.2176

Table 6.50 Independent Samples T-Test (condition 2 + condition 4)

Independent Samples Test

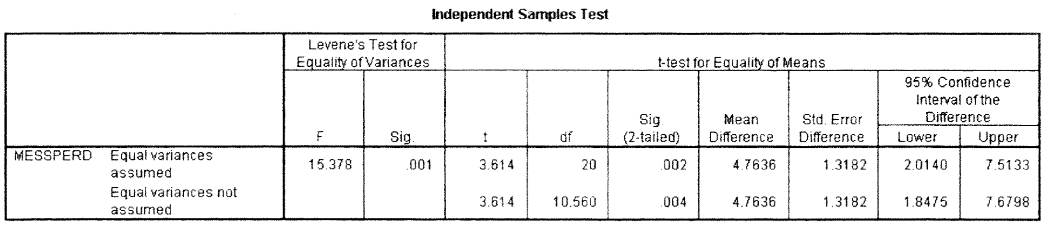
		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
MESSPERD	Equal variances assumed	4.363	.050	.345	19	.734	.2263	.6550	-1.1447	1.5972
	Equal variances not assumed			.332	11.033	.746	.2263	.6811	-1.2723	1.7249

The fifth and final significant result was between condition 3 (no tool + model) and condition 4 (no tool + no model). Table 6.51 shows the group statistics. Table 6.52 shows the results from the T-test, which is an F value of 15.38 translating to a significance of .001.

Table 6.51 Group Statistics (condition 3 + condition 4)

	Condition	N	Mean	Std. Deviation	Std. Error Mean
MESSPER	3.00	11	7.0964	4.3119	1.3001
	4.00	11	2.3327	.7216	.2176

Table 6.52 Independent Samples T-Test (condition 3 + condition 4)



Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
MESSPERD	Equal variances assumed	15.378	.001	3.614	20	.002	4.7636	1.3182	2.0140	7.5133
	Equal variances not assumed			3.614	10.560	.004	4.7636	1.3182	1.8475	7.6798

This data shows that the model alone caused the interaction effect except if both conditions had access to the tool. If neither condition had access to the CM then the team with tool access caused the interaction.

These results showed that all three hypotheses, H 10a, H 10b, and H 10c were supported.

6.4.9 Average Number of Lines Per E-mail Passed Per Group Results

Average number of lines per e-mail was evaluated by performing a 2X2 ANOVA. The results are shown in Table 6.53.

Table 6.53 Average Number of Lines Per E-mail Passed Per Group Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 7.12 SD: 3.9	Mean: 6.31 SD: 2.2	6.72
NO TOOL	Mean: 11.31 SD: 4.75	Mean: 10.96 SD: 7.25	11.14
ALL	9.22	8.64	Grand Mean: 8.93

Tools:
F = 8.86 p = .005
Model:
F = .154 p = .697
Tools X Model:
F = .025 p = .875

These results showed that subjects without tool access averaged a greater number of lines per e-mail. These results supported hypothesis H 11a.

6.4.10 Total E-mail Messages Passed Per Group Results

Total number of e-mails was evaluated by performing a 2X2 ANOVA. The results are shown in Table 6.54.

Table 6.54 Total E-mail Messages Passed Per Group Results

	MODEL	NO MODEL	ALL
TOOL	Mean: 53.92 SD: 32.65	Mean: 38.4 SD: 30.62	46.16
NO TOOL	Mean: 106.45 SD: 64.68	Mean: 35.0 SD: 10.83	70.73
ALL	80.19	36.7	Grand Mean: 58.44

Tools:
F = 4.19 p = .047
Model:
F = 13.12 p = .001
Tools X Model:
F = 5.43 p = .025

These results showed that teams without tool access passed more e-mails during the entire experiment that teams with tool access. The results also showed that teams with access to the CM passed a higher number of e-mails during this experiment than teams without CM access. There was also a significant interaction effect. The significant interaction effect was further evaluated by doing a Post Hoc Bonferroni procedure to determine where the interactions lie. Six independent T-tests were performed. Four of the six tests showed significance. The first significant result is the analysis of condition 1 (tool + model) with condition 3 (no tool + model).

Table 6.55 shows the group statistics. Table 6.56 shows the results from the T-test, which is an F value of 4.93 translating to a significance of .038.

Table 6.55 Group Statistics (condition 1 + condition 3)

	Condition	N	Mean	Std. Deviation	Std. Error Mean
NUMMES	1.00	12	53.9167	32.6537	9.4263
	3.00	11	106.4545	64.6844	19.5031

Table 6.56 Independent Samples T-Test (condition 1 + condition 3)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
NUMMESS	Equal variances assumed	4.931	.038	-2.492	21	.021	-52.5379	21.0827	-96.3818	-8.6940
	Equal variances not assumed			-2.425	14.498	.029	-52.5379	21.6616	-98.8480	-6.2277

The second significant result was between condition 1 (tool + model) and condition 4 (no tool + no model). Table 6.57 shows the group statistics. Table 6.58 shows the results from the T-test, which is an F value of 7.2 translating to a significance of .014.

Table 6.57 Group Statistics (condition 1 + condition 4)

	Condition	N	Mean	Std. Deviation	Std. Error Mean
NUMMES	1.00	12	53.9167	32.6537	9.4263
	4.00	11	35.0000	10.8259	3.2641

Table 6.58 Independent Samples T-Test (condition 1 + condition 4)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
NUMMESS	Equal variances assumed	7.198	.014	1.828	21	.082	18.9167	10.3461	-2.5993	40.4326
	Equal variances not assumed			1.896	13.581	.079	18.9167	9.9755	-2.5406	40.3740

The third significant result was between condition 2 (tool + no model) and condition 3 (no tool + model). Table 6.59 shows the group statistics. Table 6.60 shows the results from the T-test, which is an F value of 5.29 translating to a significance of .033.

Table 6.59 Group Statistics (condition 2 + condition 3)

	Condition	N	Mean	Std. Deviation	Std. Error Mean
NUMMES	2.00	10	38.4000	30.6166	9.6818
	3.00	11	106.4545	64.6844	19.5031

Table 6.60 Independent Samples T-Test (condition 2 + condition 3)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
NUMMESS	Equal variances assumed	5.291	.033	-3.028	19	.007	-68.0545	22.4761	-115.0976	-21.0114
	Equal variances not assumed			-3.125	14.554	.007	-68.0545	21.7740	-114.5890	-21.5201

The fourth and final significant result was between condition 3 (no tool + model) and condition 4 (no tool + no model). Table 6.61 shows the group statistics. Table 6.62 shows the results from the T-test, which is an F value of 15.38 translating to a significance of .001.

Table 6.61 Group Statistics (condition 3 + condition 4)

	Condition	N	Mean	Std. Deviation	Std. Error Mean
NUMMES	3.00	11	106.4545	64.6844	19.5031
	4.00	11	35.0000	10.8259	3.2641

Table 6.62 Independent Samples T-Test (condition 3 + condition 4)

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
									95% Confidence Interval of the Difference	
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
NUMMESS	Equal variances assumed	15.375	.001	3.613	20	.002	71.4545	19.7743	30.2060	112.7031
	Equal variances not assumed			3.613	10.560	.004	71.4545	19.7743	27.7091	115.2000

This data shows that the model caused the interaction. These results showed that all three hypotheses, H 12a, H 12b, and H 12c were supported.

6.4.11 E-mail Process Conflict Results

E-mail Process Conflict was evaluated by the judges' categorization of the e-mails passed during the experiment. There was no significance as shown in Table 6.63. Hypothesis H 9c was supported in that no interaction effect was predicted.

Table 6.63 Process Conflict Results

	MODEL	NO MODEL	ALL
TOOL	Mean: .08 SD: .19	Mean: .05 SD: .16	.07
NO TOOL	Mean: .59 SD: 1.1	Mean: .14 SD: .45	.37
ALL	.34	.1	Grand Mean: .22

Tools:
F = 1.99 p = .39
Model:
F = 1.34 p = .45
Tools X Model:
F = 1.33 p = .26

6.4.12 Questionnaire Evaluation

The post task questionnaire, shown in Appendix B, was evaluated via a Factor Analysis and Chronbach's Alpha to determine the factors; then an ANOVA was done on the resulting scales.

The results from the Factor Analysis, using a factor loading of .55 or greater and an Eigen value greater than one, were three scales. Scale one included questions 1, 2, 3, 4, 5, 6, 7, 9, and 10. It had an Eigen value of 6.73 explaining 48.1% of the variance un-rotated. Scale one, when rotated, has an Eigen value of 4.2 explaining 30% of the variance. Scale two included questions 2, 8, 10, 12, and 14. It had an Eigen value of 1.34 explaining 9.5% of the variance un-rotated. Scale two, when rotated, has an Eigen value of 3.2 explaining 22.6% of the variance. Scale three included questions 11 and 13. It had an Eigen value of 1.03 explaining 7.4% of the variance. Scale three, when rotated, has an Eigen value of 1.7 explaining 12.4% of the variance. The three scales combined accounted for 65.0% of the variance.

Further analyzing the results, it made logical sense to only use scale one for the 'Satisfaction' variable since it included every satisfaction question except question eight. It also made logical sense to use scale three for 'task validation' since it included two of the four 'task validation' questions and no satisfaction questions. A Chronbach's Alpha evaluation was also performed resulting in a value of .9. This value shows a high internal consistency for the questionnaire.

To complete the analysis, an ANOVA was performed on the Satisfaction questions that resulted from the Factor Analysis. The results, shown in Table 6.64, showed no significance for this variable. Therefore, H 3a, H 3b, and H 3c were not supported.

Table 6.64 Satisfaction Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 49.43 SD: 10.52	Mean: 51.65 SD: 8.57	50.54
NO TOOL	Mean: 47.88 SD: 11.69	Mean: 51.39 SD: 8.85	49.64
ALL	48.66	51.52	Grand Mean: 50.09

Tools:
F = .32 p = .57
Model:
F = 3.23 p = .07
Tools X Model:
F = .674 p = .68

An ANOVA was also performed on the Task Validation questions that resulted from the Factor Analysis. The results, shown in Table 6.65, showed no significance for this variable.

Table 6.65 Task Validation Results

	MODEL	**NO MODEL**	**ALL**
TOOL	Mean: 9.11 SD: 3.46	Mean: 9.62 SD: 3.34	9.37
NO TOOL	Mean: 9.6 SD: 2.77	Mean: 10.26 SD: 2.84	9.93
ALL	9.36	9.94	Grand Mean: 9.65

Tools:
F = 1.29 p = .26
Model:
F = 1.39 p = .241
Tools X Model:
F = .02 p = .88

6.5 Summary of Hypotheses Analysis

Table 6.66 shows a summary of the hypotheses results of the experiment. Please note that hypotheses H 8a, H 8b, H 8c, H 9a, H 9b, H 9c, H 3a, H 3b, H 3c, H 7a, H 7b, and H 7c were each refined as the data analysis was performed. The original hypotheses were the following:

H 8a. **Quality of e-mail participation** will be lower in teams with tool access than that of the teams working without tool access.

H 8b. **Quality of e-mail participation** will be higher in teams having access to the CM than the teams working under the condition without the CM.

H 8c. When evaluating **e-mail participation quality**, no interaction effect will occur between the tools and the CM.

H 9a. **E-mail Message Pattern** will be lower in teams with tool access that that of the teams working without tool access.

H 9b. **E-mail Message Pattern** will be higher in teams with CM access.

H 9c. When evaluating **e-mail message pattern** no interaction effect will occur between the tools and the CM.

H 3a. **Solution Satisfaction** will be higher in the teams with tools than for teams working without tools.

H 3b. **Solution Satisfaction** will be higher in the teams having access to the CM than the teams working under the condition without the CM.

H 3c. When evaluating **solution satisfaction**, a positive synergistic effect will occur between the tools and the CM.

H 7a. **Process Satisfaction** will be higher in the teams with tool access than that of the teams working without tool access.

H 7b. **Process Satisfaction** will be higher in the teams having access to the CM than the teams working under the condition without the CM access.

H 7c. When evaluating **process satisfaction**, a positive synergistic effect will occur between the tools and the CM.

The refined hypotheses are shown in Table 6.66 and Table 6.67. The H 8 hypotheses series were refined into H 7 and H 8 series of hypotheses. The original H 9 series of hypotheses were refined into H10, H11, and H12 series hypotheses. The H 3 and H 7 series of hypotheses were combined into one hypothesis due to the results of the Factor analysis discussed in section 6.4.12.

Table 6.66 Summary of Hypotheses Results

HYPOTHESIS	RESULT
H 1a. Teams working with the tools will produce **more creative solutions** than the teams working without tools.	**Unsupported**
H 1b. The teams having access to the CM will produce **more creative solutions** than the teams working under the condition without the CM.	**Unsupported**
H 1c. When evaluating **solution creativity**, a positive synergistic effect will occur between the tools and the CM.	**Unsupported**
H 2a. The teams working with the tools will produce **higher quality solutions** than teams working without tools.	**Unsupported**
H 2b. Teams with access to the CM will produce **higher quality solutions** than the teams working under the condition without the CM.	**Unsupported**
H 2c. When evaluating **solution quality,** a positive synergistic effect will occur between the tools and the CM.	**Unsupported**
H 3a. **Satisfaction** will be higher in the teams with tools than for teams working without tools.	**Unsupported**
H 3b. **Satisfaction** will be higher in the teams having access to the CM than the teams working under the condition without the CM.	**Unsupported**
H 3c. When evaluating **satisfaction**, a positive synergistic effect will occur between the tools and the CM.	**Unsupported**
H 4a. The teams having access to the collaborative tools will show **superior understanding of the problem** as demonstrated by their ability to clearly and correctly state problems and extract problem facts better than teams without tool access.	**Unsupported**
H 4b. The teams having access to the CM will show **superior understanding of the problem** as demonstrated by their ability to clearly and correctly state problems and extract problem facts better teams without CM access.	**Supported**
H 4c. When evaluating **problem understanding,** a positive synergistic effect will occur between the tools and the CM.	**Unsupported**
H 5a. Teams working with the tools will **generate more alternatives** than those teams working without tools.	**Unsupported**
H 5b. The teams having access to the CM will **generate more alternatives** than the teams working under the condition without the CM.	**Unsupported**
H 5c. When evaluating the **number of alternative** generated, a synergistic effect will occur between the tools and the CM. *It was found that either the tool or the model alone was significantly better. However, the combination or the absence of was not significantly better.*	**Supported**
H 6a. The teams having access to the collaborative tools will show **higher quality solution planning** as demonstrated by their ability to provide detailed and clear plans, complete goal refinements and representation of facts better then the teams working under the condition without tool access.	**Unsupported**
H 6b. The teams having access to the CM will show **higher quality solution planning** as demonstrated by their ability to provide detailed and clear plans, complete goal refinements and representation of facts better then the teams working under the condition without the CM.	**Supported**
H 6c. When evaluating **solution planning quality**, a positive synergistic effect will occur between the tools and the CM.	**Unsupported**

Table 6.67 Summary of Hypotheses Results (Continued)

HYPOTHESIS	RESULT
H 7a. Less **Task Oriented** e-mails will be sent from teams with tool access than teams working without tool access.	**Supported**
H 7b. More **Task Oriented** e-mails will be sent from teams with CM access than teams working without CM access.	**Supported**
H 7c. When evaluating **Task Oriented** e-mails, a synergistic effect will occur between the tools and the CM. *This data shows the model alone caused the interaction effect on all other conditions. In addition, the combination of the model and the tool caused an interaction when compared to the condition with the tool and model absent.*	**Supported**
H 8a. Less **Socially Oriented** e-mails will be sent from teams with tool access than teams working without tool access.	**Unsupported**
H 8b. More **Socially Oriented** e-mails will be sent from teams with CM access than teams working without CM access.	**Supported**
H 8c. When evaluating **Socially Oriented** e-mails, a synergistic effect will occur between the tools and the CM. *The results showed that the condition having access to only the model and only e-mail caused the interaction when compared to the condition with neither access to Groove nor the collaborative model.*	**Supported**
H 9a. **Process Conflict** will be lower in teams with tool access that that of teams with tool access.	**Unsupported**
H 9b. **Process Conflict** will be higher in teams having access to the CM that that of teams working under the condition without the CM.	**Unsupported**
H 9c. When evaluating **process conflict** no interaction effect will occur between the tools and the CM.	**Supported**
H 10a. **E-mail Messages Passed Per day** will be lower in teams with tool access that that of the teams working without tool access.	**Supported**
H 10b. **E-mail Messages Passed Per day** will be higher in teams with CM access.	**Supported**
H 10c. When evaluating **E-mail Messages Passed Per day** synergistic effect will occur between the tools and the CM. *The data showed that the model alone caused the interaction effect except if both conditions had access to the tool. If neither condition had access to the CM then the team with tool access caused the interaction.*	**Supported**
H 11a. **Average Number of Lines Per E-mail** will be lower in teams with tool access that that of the teams working without tool access.	**Supported**
H 11b. **Average Number of Lines Per E-mail** will be higher in teams with CM access.	**Unsupported**
H 11c. When evaluating **Average Number of Lines Per E-mail** synergistic effect will occur between the tools and the CM.	**Unsupported**
H 12a. **Total E-mail Messages** will be lower in teams with tool access that that of the teams working without tool access.	**Supported**
H 12b. **Total E-mail Messages** will be higher in teams with CM access.	**Supported**
H 12c. When evaluating **Total E-mail Messages** a synergistic effect will occur between the tools and the CM. *This data shows that the model caused the interaction.*	**Supported**

In summary, out of the 12 hypotheses, four were supported for the tool, six were supported for the model, and six were supported for the interaction.

The hypotheses that most represented the collaborative model were H1, H2, H4, H5, and H6. Two of these proved significant for the model: quality of solution planning and problem understanding. One proved a significant interaction that showed that either the tool or model alone was significantly better when creating solution alternatives. The hypotheses supported by the tool and hypotheses that supported additional interactions were related to the e-mail statistics.

6.6 Dependent Variable Correlations

The model shown in the previous chapter (Figure 5.1) depicted correlations predicted between the dependent variables. This model was slightly modified after the post-task questionnaire factor analysis results indicated to combine the solution and process satisfaction variables into one 'satisfaction' variable. The new model is shown in Figure 6.2 below.

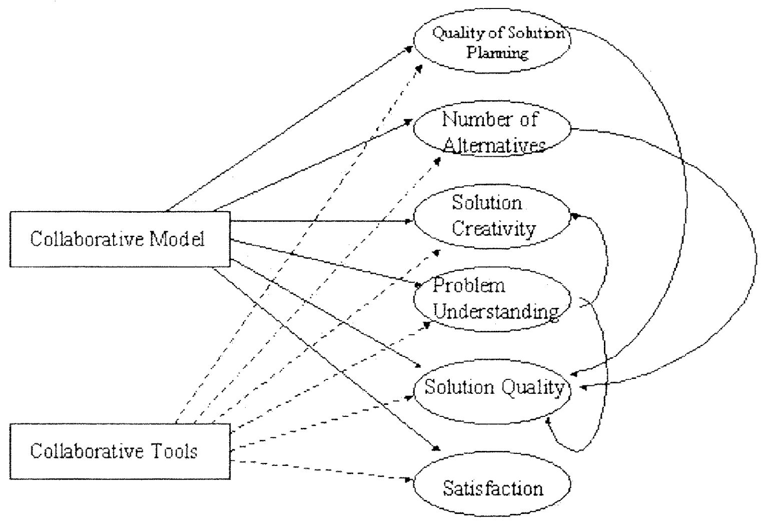


Figure 6.2 New Theoretical Model.

A path analysis of the data collected was performed to test the model for consistency. The hypothesized model, Figure 6.2, depicts the predicted relationships and how each variable combines to form a total model. The result of the analysis was a Chi-Square value of 1.46 equating to a .23 significance indicating that there was not a significant difference between the hypothesized model and the actual model. Figure 6.3 shows the actual correlation values. Any value over 2.0 was considered significant.

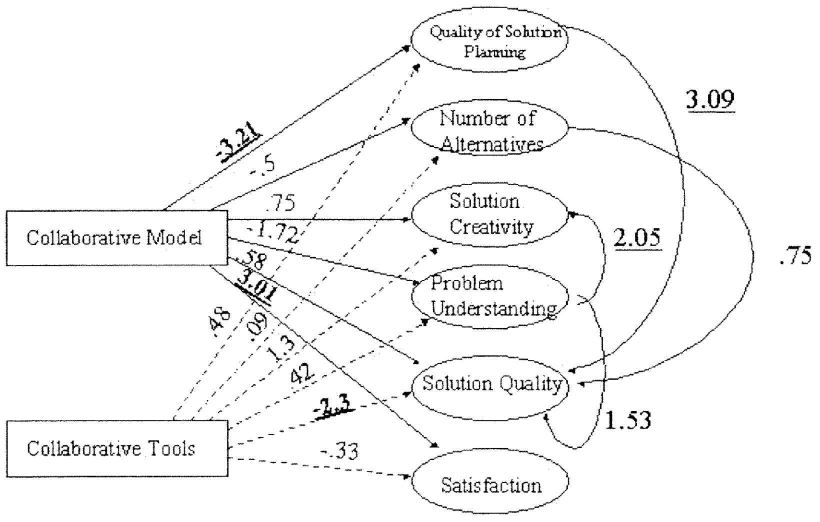


Figure 6.3 Actual Path Model.

As shown in the actual path model there was a correlation between the model and quality of solution planning, the model and satisfaction, and the tool and solution quality. The actual model also showed a correlation between problem understanding and solution creativity and solution quality and quality of solution planning.

CHAPTER 7

DISCUSSION, CONCLUSIONS AND FUTURE WORK

This chapter concludes this dissertation with a summary of the evaluation results presented in the previous chapter. This summary includes a discussion of the experimental results and their implications. Questions will be answered as to why certain variables had positive results and why others did not. In addition, the implications of the experimental results will also be compared to the theory presented in chapter 3 of this dissertation.

This chapter will also include a discussion of the various research contributions this dissertation provides. Finally, a conclusion with proposed enhancements to the experimentation of the Collaborative Problem Solving and Program Development model, proposed further experimentation and future work plans will be provided.

7.1 Evaluation Results

Taken as a whole this experiment proved some benefits of implementing a structured framework during the collaborative problem solving and program development process. This section will summarize the supported hypotheses of the experiment as well as attempt to explain the unsupported hypotheses.

7.1.1 Problem Formulation Hypothesis

The resulting data from the problem formulation document showed that subjects having access to the collaborative model had a greater understanding of the problem they were attempting to solve than the groups that did not have access to the collaborative model. Problem understanding is specifically associated with the first stage of the collaborative model that included the preliminary problem description, preliminary mental model, and structured problem representation phases. The subjects were able to show a clear understanding of the problem description, they were able to determine goals, givens and unknowns, and they were able to extract facts from the problem description and organize them in order to better understand the problem. Specifically, the subjects were instructed to interpret and verbalize the problem. If they were in a condition that had access to Groove they were able to use a brainstorming or discussion tool to verbalize their problem understanding with their team members. If the were not assigned a condition which had access to Groove they accomplished their verbalization through e-mail. Following the verbalization task, team members were to agree upon a problem description in which the entire team would follow.

The team problem understanding success was further enhanced by the team answering a few questions regarding the problem such as: *what is the goal, do the goals require clarification, are there any other explicit or implicit problem requirements, what are the givens, what are the unknowns, are there any conditions and constraints*? These answers were organized and used to begin the design and planning of a solution. Lack of support for the tool main effect may have to do with either the learning curve factor of using a new tool or possibly Groove not facilitating the problem understanding tasks in the CM.

7.1.2 Quality of Solution Planning Hypothesis

The resulting data from the solution plan document showed that subjects having access to the collaborative model performed better on this step than subjects without access to the collaborative model.

The success of the model for this variable may have to do with the specific planning tasks such as the decomposition of the task into specific sub goals and further showing a plan on how to accomplish this task, which was suggested by the model. The lack of support for the tool main effect may have to do with the fact that the tools associated with Groove were too complicated to learn in the limited amount of time for the experiment.

7.1.3 Number of alternatives Hypothesis

The number of alternatives variable, measured by the solution plan document, showed a significant interaction effect. Further analysis of the interactions showed that the number of alternatives was significantly higher in teams that had access to the collaborative model alone or had access to Groove alone. Teams with access to the combination of Groove and the collaborative model or involved in the condition where both Groove and the collaborative model were absent were found to have created significantly fewer alternatives. This could be explained by the learning curve factor with both Groove and the collaborative model. Subjects having to learn both may have had a slight disadvantage compared to those who only had to learn either a new tool or the new model.

Adding time and a simple task to the training session could possibly remedy this problem. Modifying the training session will be discussed further in the next section of this chapter. The success of the conditions with only access to the tool could be explained by the increased ability to communicate such as having access to a brainstorming tool as well as a chat tool. The teams with access to only the collaborative model may have had success because of the explicit tasks involved in using the collaborative model, which encourage a well thought out solution where many alternatives are discussed to determine that the correct solution plan was chosen.

7.1.4 Task-oriented E-mail Hypothesis

Evaluation of the e-mails passed between group members showed that teams with access to Groove showed significantly less task-oriented e-mails. This is most likely due to the fact that teams with Groove access have access to other communication tools in addition to e-mail such as brainstorming, discussion, and chat tools. Therefore, most of the team's task-oriented conversation was accomplished with the Groove tool and not e-mail.

Teams with access to the CM had a significantly higher number of task e-mails than teams without access to the CM. The teams with CM access had more direction as far as team interaction. Therefore, they were encouraged by the model to discuss the task due to specific problem solving exercises involved in the CM.

The task oriented e-mails variable also produced an interaction effect. Further analysis of the interaction showed that it was explained by access to the CM.

7.1.5 Socially-oriented E-mail Hypothesis

Evaluation of the e-mails passed between the team members also showed that the teams with access to the CM had written significantly more social e-mails than the other experimental conditions. As mentioned previously, the team members were prompted to interact more than teams without access to the CM. This increased the probability of social interaction between the team members. It can be hypothesized that even though there was no significance with the tool main effect, teams with access to Groove and the CM accomplished their socializing within the Groove application due to the additional communication possibilities with Groove. There was also an interaction effect with this variable. Further analysis of the interaction showed that the interaction was explained by access to the CM, where the subjects were prompted to interact more than teams without access to the CM.

7.1.6 Number of E-mails Per Day Hypothesis

The number of e-mails passed per day between the team members was significantly lower in teams with access to Groove and significantly higher in teams with access to the CM. This result can be explained by similar reasons as discussed in the task and socially oriented e-mail results. For example, teams with access to Groove did not need to use their e-mail tool since they had a choice of many communication tools with Groove. The teams with CM access were encouraged to communicate, therefore had more reasons to use their e-mail tool.

7.1.7 Average Number of Lines Per E-mail Hypothesis

The average number of lines per e-mail was lower in the teams with tool access. All of these results occurred for the same reasons as stated above where teams with Groove access had other means of communication in addition to e-mail and teams with CM access were prompted to communicate more. The interaction effect for the number of e-mails passed per day was explained again by use of the CM.

7.1.8 Total Number of E-mails Hypothesis

Evaluating the total number of e-mails passed per group was lower in teams with Groove access and higher in teams with model access. The interaction effect was again explained by access to the CM. The support for the tool with the e-mail statistics hypotheses showed that teams with Groove access did use e-mail less possibly indicating that the teams used Groove to communicate. This shows that the teams utilized the tool but possibly had more of a focus on the new tool than facilitating the use of the collaborative model.

7.1.9 Process Conflict Hypothesis

The final supported hypothesis was that there was no interaction between teams with access to both Groove and the collaborative model when evaluating process conflict. The process conflict variable had no main support. It was predicted that teams with access to the CM would have more conflict due to the amount of interaction and the amount of decisions to be made among the group members. This was not the case and possibly due to the fact the tasks involved in using the CM produced a positive side effect such as cognitive synchronization and this eliminated many of the conflicts possible in a group environment. The tool main effect predicted that the teams with groove access would show less conflict via e-mail. There was no significance probably due to the fact that any conflict was resolved via Groove or cognitive synchronization occurred.

7.1.10 Solution Creativity Hypothesis

Solution creativity is the first of the three totally unsupported dependent variables in that there were no main effects for either the tool or the model or an interaction effect. The judge's evaluation of creativity did not show a significant difference between the conditions. This could be a result of the task given to the subjects in this experiment (supermarket system modeling, a well structured task with components pre-specified) may not have produced sufficient variance in creativity. In other words, it was too obvious of a task that produced similar solutions by the subjects in the various conditions. To increase the variance for creativity in the experimental conditions, the experimental task difficulty, novelty, or ambiguity could be increased. The increased task difficulty would result in a larger range of solutions for the task.

7.1.11 Solution Satisfaction Hypothesis

Solution satisfaction was also an unsupported dependent variable. The lack of support could be attributed to insufficient training time prior to the experiment. In the training period of the experiment the subjects were expected to download the Groove software, install it, and create a workspace for their team. This could have been overwhelming to some of the team members and in effect slowing down the connection process with their team. This would create dissatisfaction for the overwhelmed team members as well as the team members that were waiting for their team to connect. There was also no training task therefore; teams with access to the CM were working with a model they have never had the opportunity to use prior to the experiment creating possibly another overwhelming situation. Adding a training task would possible remedy that situation.

An experiment on the CyberCollaboratory (Dufner et. al., 2002) also produced similar results for efficiency, coordination, fairness and satisfaction. Dufner's experiment suggested subjects in a condition with access to the CyberCollaboratory system felt the process was less efficient, coordinated, fair and satisfying. It was suggested by Dufner et. al. (2002) that this result was possibly due to the insufficient training time combined with a short amount of time using the tools for a fairly simple group training task.

7.1.12 Solution Quality Hypothesis

Solution quality is equally affected by the short training period as well as the absence of a training task. The teams with Groove access may have been focusing more on the new tools available to them and not on a quality solution to the task. The teams with CM access may not have been able to fully utilize the model since it was the first time the model was seen by the subjects.

The unsupported variables of this experiment, creativity and quality, are also consistent with Ocker's (1995) research of software requirements creation. Ocker's results stated that using a "problem solving approach did not significantly impact creativity or quality".

7.1.13 Results Overview

To summarize the results, the hypotheses variables that most represented the collaborative model were problem understanding, quality of solution planning, creativity, quality, and number of alternatives. Two of these proved significant for the model: quality of solution planning and problem understanding. One proved a significant interaction, number of alternatives, which showed that either the tool or model alone was significantly better when creating solution alternatives. The hypotheses supported by the tool and hypotheses that supported additional interactions were related to the e-mail statistic variables.

It is probable that the lack of support for Groove with the hypotheses related to the model has to do with the learning curve of using a new tool. Adding a simple training task to this experiment to be performed during the training period could have increased the hypothesis support for Groove. This task would have acted as a practice problem to familiarize the team with Groove and all of its features. More time would have been required for this addition; however, adding a practice problem may have increased the quality, creativity, and satisfaction variables.

Another possible cause of the lack of Groove support could be that the model and tool did not bear a close enough resemblance. Subjects in the condition where they had access to both the tool and the model had to learn both, subsequently increasing the apparent learning curve. The reason Groove was chosen was to facilitate certain aspects of the model. Further study of tool assistance with the CM would be possible future work that will be discussed in the next section.

7.2 Summary of Research Contributions

This dissertation has proposed a framework for a Collaborative Problem Solving and Program Development Model that detailed the cognitive processes and the social activities that occur during problem solving and program development. This model has shown improvement on the output and success of a group attempting to solve a problem with software.

In the past, most groupware systems have focused on the communication aspect of collaboration but not the coordination and cognitive issues that need to be addressed during problem solving and software development. The CM does address such issues by detailing the cognitive activities and collaborative structure in each phase of the model. Previous studies in this area have only examined software requirement development with use of different modes of collaboration (Ocker, Hiltz, Turoff, Fjermestad, 1995; Ocker & Fjermestad 1998; Ocker et. al. 1998; Ocker, 2001) and use of a decision making model with modes of collaboration (Ocker, Hiltz, Turoff, Fjermestad, 1995).

The experiment performed to test the CM considered a much larger aspect of the problem solving and software development process. The focus was on the first two stages of problem solving and software development: Problem Formulation and Solution Planning. Several objectives have been accomplished by this research:

1. The cognitive processes and collaborative structure required for the six stages of collaborative problem solving and program development were defined and detailed. These cognitive processes and collaborative structure take into consideration the psychological and sociological issues present during problem solving and program development collaboration. Collaborative problem solving is characterized by the cognitive processes it identifies for problem solving and by the collaborative structure it utilizes. A collaborative structure was defined both by the modality of the collaboration and the dynamics of the group. The modality of collaboration refers to the variety of possible interaction modes, ranging from chat to asynchronous messaging. The group dynamics of a collaboration encompasses the processes that define the collaboration: negotiation, scheduling, coordination, integration, acceptance, etc.; the side effects of these collaborative processes: cognitive bias, conflict resolution, group cohesion, distributed learning, etc, the administration of these collaborative processes: task initiation, delegation of functions, subcomponent integration, on going evaluation, etc, and the management of side effects.

2. An extensive literature review beginning with a discussion of individual problem solving prior to discussing the background literature on collaborative

problem solving was presented. Additional background literature on groupware systems, general groupware tools, and groupware tools specific for problem solving and software development was also presented. This review determined the lack of collaborative problem solving models and tools to enhance the problem solving and program development needs of teams.

3. A review and case study of groupware tools was performed and critiqued. This review resulted in first determining that a tool available to assist in the entire collaborative problem solving and program development process did not exist. Secondly, the review resulted in finding a tool to facilitate the collaborative modality of the model during the experimentation with the Collaborative Problem Solving and Program Development model.

4. Results of an experiment showing the enhancement of solution planning and problem understanding for subjects using the collaborative model were detailed. In addition, a few of the measured variables of this experiment also showed similar results to previous studies experimenting with groupware tools.

7.3 Future Work

Future work should consist of thorough experimentation on the remaining four stages of the collaborative problem solving and program development model. This should further show the benefits of the CM during the solution design, solution translation, solution testing, and solution delivery stages of the model. This type of experiment would be rather extensive in that an entire project from problem understanding to code implementation would be necessary. At least six to eight weeks of time should be allocated to test these stages of the model. The allocated experiment time would be dependent upon the complexity of the problem. It should be noted again that having a more complex problem could create the right amount of variance for the measured creativity variable.

In addition to testing the remaining stages of the CM, modifying of the training portion of the experiment by adding a simple training task and additional time to the training session may show positive results when measuring the quality, creativity and satisfaction variables. The extra training time would also lessen the effects of the learning curve that occurred with using Groove and the CM. The extra training time may also increase satisfaction among the subjects given that satisfaction was decreased due to learning curve issues and possibly feeling overwhelmed which may have occurred with learning a new tool and model.

Future work should also consist of integrating the collaborative model with existing groupware tools such as Groove. This would eliminate a portion of the apparent learning curve during the experiment and ultimately during use of the tool by software developers.

The combination of Groove and the CM would only be prudent if positive results from the tool main effects of the hypotheses resulted from testing the remaining stages of the CM. If the tool main effects do not show positive results, a new collaborative tool should be designed that has a closer resemblance to the collaborative modality of the CM. This tool could contain the necessary technology to facilitate the collaborative dynamics imbedded in each phase of the CM.

A newly designed tool would have a higher probability in facilitating the experimental results of the CM such as problem understanding and solution planning and possibly the non-positive experimental results such as creativity, quality, and satisfaction. This tool could take into consideration the group dynamics of software development and facilitate the processes that define the collaboration such as: negotiation, scheduling, coordination, integration, acceptance, etc. The tool could also be designed to eliminate the negative side effects of collaboration such as cognitive bias and conflict resolution and the positive side effects such as group cohesion and distributed learning.

7.4 Conclusion

Contemporary system developers work in environments where projects require a team effort. This fact implies that collaboration or group problem solving is an expected skill for current software and systems engineers. Factors driving this implication include the scale of contemporary engineering projects that necessitate collaborative development, the logistical difficulties of divergent work schedules, the geographical dispersion of expertise, and the availability of platform-independent communications provided by the web.

Collaborative development has a variety of advantages beyond alleviating logistical difficulties, ranging from demonstrable improvements in design efficiency, effectiveness of problem specification, substantial benefits from group learning, the reliability afforded through group understanding of the problem and the current state of the project, to other advantages indicated in our analysis. By integrating the problem solving underpinnings of collaborative development, the technological, psycho-social, and cognitive factors that arise in these systems, the requirements needed for collaboration during software development have been identified.

APPENDIX A

CONSENT FORMS

This appendix shows the four different consent forms used for each condition of the experiment.

NEW JERSEY INSTITUTE OF TECHNOLOGY
323 MARTIN LUTHER KING BLVD.
NEWARK, NJ 07102

CONSENT TO PARTICIPATE IN A RESEARCH STUDY – Form 1

TITLE OF STUDY:

RESEARCH STUDY:

I, ______________________________________, have been asked to participate in a research study under the direction of Joanna DeFranco-Tommarello.
Other professional persons who work with her as study staff may assist to act for her.

PURPOSE:

To test the problem solving and software development effectiveness of using the specified collaborative software model with the appropriate groupware tools.

DURATION:

My participation in this study will last for 3 weeks.

PROCEDURES:

I have been told that, during the course of this study, the following will occur:

My team will be given a software problem to solve collaboratively using the collaborative model and Groove that I will download as per the instructions given to me at the outset of the experiment. I was told that Groove would provide tools such as brainstorming, voting, documentation storage, asynchronous messaging, member contacts, scheduling etc.

No talking with team members is allowed. The team will be required to perform the tasks under the Problem Formulation and Solution Planning stages of a collaborative model. Following the completion of the tasks required, I am required to fill out a questionnaire and participate in a debriefing session in my sections web board conference.

I was given a choice of either participating in this experiment or working on a similar project. My grade will be based on my ability to follow directions and the quality of my performance on the specific tasks.

My team has decided that the process facilitator is _______________________________ and the content facilitator is _______________________________. The facilitators have agreed to follow the instructions as outlined in the task document provided at the outset of this experiment. The facilitators will receive up to an extra 10 points added to their score depending on how well they carry out the role responsibilities.

PARTICIPANTS:

I will be one of about 192 participants to participate in this trial.

EXCLUSIONS:

I will inform the researcher if any of the following apply to me: N/A

RISK/DISCOMFORTS:

I have been told that the study described above may involve the following risks and/or discomforts:

There are no known risks or discomforts.

There also may be risks and discomforts that are not yet known – N/A.

CONFIDENTIALITY:

Every effort will be made to maintain the confidentiality of my study records. Officials of NJIT will be allowed to inspect sections of my research records related to this study. If the findings from the study are published, I will not be identified by name. My identity will remain confidential unless disclosure is required by law.

PAYMENT FOR PARTICIPATION:

I have been told that I will receive $0 compensation for my participation in this study.

CONSENT AND RELEASE:

I fully recognize that there are risks that I might be exposed to by volunteering in this study which are inherent in participating in any study; I understand that I am not covered by NJIT's insurance policy for any injury or loss I might sustain in the course of participating in the study.

I agree to assume and take on myself all risks and responsibilities in any way associated with this activity. I release NJIT, its trustees, agents, employees and students from any and all liability, claims and actions that may arise as a result of my participation in the study. I understand that this means that I am giving up my right to sue NJIT, its trustees, agents and employees for injuries, damages or losses I may incur.

RIGHT TO REFUSE OR WITHDRAW:

I understand that my participation is voluntary and I may refuse to participate, or may discontinue my participation at any time with no adverse consequence. I also understand that the investigator has the right to withdraw me from the study at any time.

INDIVIDUAL TO CONTACT:

If I have any questions about my treatment or research procedures that I discuss them with the principle investigator. If I have any addition questions about my rights as a research subject, I may contact:

Robin-Ann Klotsky, Executive Director of Research and Development at (973) 596-5227.

SIGNATURE OF PARTICIPANT

I have read this entire form, or it has been read to me, and I understand it completely. All of my questions regarding this form or this study have been answered to my complete satisfaction. I agree to participate in this research study.

Subject: Name: ____________________
Signature:__________________________

Date: ____________________________

SIGNATURE OF READER/TRANSLATOR IF THE PARTICIPANT DOES NOT READ ENGLISH WELL

The person who has signed above, ____, does not read English well, I read English well and am fluent in (name of the language) ________________________________, a language the subject understands well. I have translated for the subject the entire content of this form. To the best of my knowledge, the participant understands the content of this form and has had an opportunity to ask questions regarding the consent form and the study, and these questions have been answered to the complete satisfaction of the participant (his/her parent/legal guardian).

Reader/Translator Name: ______________________
Signature: _________________________________
Date: _____________________________________

SIGNATURE OF INVESTIGATOR OR RESPONSIBLE INDIVIDUAL

To the best of my knowledge, the participant, ____, has understood the entire content of the above consent form, and comprehends the study. The participants and those of his/her parent/legal guardian have been accurately answered to his/her/their complete satisfaction.

Investigator's Name: ______________________
Signature: _______________________________
Date: ___________________________________

NEW JERSEY INSTITUTE OF TECHNOLOGY
323 MARTIN LUTHER KING BLVD.
NEWARK, NJ 07102

CONSENT TO PARTICIPATE IN A RESEARCH STUDY – Form 2

TITLE OF STUDY:

RESEARCH STUDY:

I, ____, have been asked to participate in a research study under the direction of Joanna DeFranco-Tommarello.
Other professional persons who work with her as study staff may assist to act for her.

PURPOSE:

To test the problem solving and software development effectiveness of using the appropriate groupware tools.

DURATION:

My participation in this study will last for 3 weeks.

PROCEDURES:

I have been told that, during the course of this study, the following will occur:

My team will be given a software problem to solve collaboratively using Groove that I will download as per the instructions given to me at the outset of the experiment. I was told that the groupware systems would provide tools such as brainstorming, voting, documentation storage, asynchronous messaging, member contacts, scheduling etc.

No talking with team members is allowed. The team will be required to perform the tasks under the Problem Formulation and Solution Planning stages of a collaborative model. Following the completion of the tasks required, I am required to fill out a questionnaire and participate in a debriefing session in my sections web board conference.

I was given a choice of either participating in this experiment or working on a similar project. My grade will be based on my ability to follow directions and the quality of my performance on the specific tasks.

My team has decided that the process facilitator is ______________________ and the content facilitator is ______________________. The facilitators have agreed to follow the instructions as outlined in the task document provided at the outset of this experiment. The facilitators will receive up to an extra 10 points added to their score depending on how well they carry out the role responsibilities.

PARTICIPANTS:

I will be one of about 192 participants to participate in this trial.

EXCLUSIONS:

I will inform the researcher if any of the following apply to me: N/A

RISK/DISCOMFORTS:

I have been told that the study described above may involve the following risks and/or discomforts:

There are no known risks or discomforts.

There also may be risks and discomforts that are not yet known – N/A.

CONFIDENTIALITY:

Every effort will be made to maintain the confidentiality of my study records. Officials of NJIT will be allowed to inspect sections of my research records related to this study. If the findings from the study are published, I will not be identified by name. My identity will remain confidential unless disclosure is required by law.

PAYMENT FOR PARTICIPATION:

I have been told that I will receive $0 compensation for my participation in this study.

CONSENT AND RELEASE:

I fully recognize that there are risks that I might be exposed to by volunteering in this study which are inherent in participating in any study; I understand that

I am not covered by NJIT's insurance policy for any injury or loss I might sustain in the course of participating in the study.

I agree to assume and take on myself all risks and responsibilities in any way associated with this activity. I release NJIT, its trustees, agents, employees and students from any and all liability, claims and actions that may arise as a result of my participation in the study. I understand that this means that I am giving up my right to sue NJIT, its trustees, agents and employees for injuries, damages or losses I may incur.

RIGHT TO REFUSE OR WITHDRAW:

I understand that my participation is voluntary and I may refuse to participate, or may discontinue my participation at any time with no adverse consequence. I also understand that the investigator has the right to withdraw me from the study at any time.

INDIVIDUAL TO CONTACT:

If I have any questions about my treatment or research procedures that I discuss them with the principle investigator. If I have any addition questions about my rights as a research subject, I may contact:

Robin-Ann Klotsky, Executive Director of Research and Development at (973) 596-5227.

SIGNATURE OF PARTICIPANT

I have read this entire form, or it has been read to me, and I understand it completely. All of my questions regarding this form or this study have been answered to my complete satisfaction. I agree to participate in this research study.

Subject: Name: ____________________
Signature:___________________________

Date: ____________________________

SIGNATURE OF READER/TRANSLATOR IF THE PARTICIPANT DOES NOT READ ENGLISH WELL

The person who has signed above, ____, does not read English well, I read English well and am fluent in (name of the language) ________________________________, a language the subject understands well. I have translated for the subject the entire content of this form. To the best of my knowledge, the participant understands the content of this form and has had an opportunity to ask questions regarding the consent form and the study, and these questions have been answered to the complete satisfaction of the participant (his/her parent/legal guardian).

Reader/Translator Name: ______________________
Signature: ________________________________
Date: ____________________________________

SIGNATURE OF INVESTIGATOR OR RESPONSIBLE INDIVIDUAL

To the best of my knowledge, the participant, ____, has understood the entire content of the above consent form, and comprehends the study. The participants and those of his/her parent/legal guardian have been accurately answered to his/her/their complete satisfaction.

Investigator's Name: ______________________
Signature: ________________________________
Date: ____________________________________

NEW JERSEY INSTITUTE OF TECHNOLOGY
323 MARTIN LUTHER KING BLVD.
NEWARK, NJ 07102

CONSENT TO PARTICIPATE IN A RESEARCH STUDY – Form 3

TITLE OF STUDY:

RESEARCH STUDY:

I, ____, have been asked to participate in a research study under the direction of Joanna DeFranco-Tommarello.
Other professional persons who work with her as study staff may assist to act for her.

PURPOSE:

To test problem solving and software development effectiveness when using the specified collaborative software development model.

DURATION:

My participation in this study will last for 3 weeks.

PROCEDURES:

I have been told that, during the course of this study, the following will occur:

My team will be given a software problem to solve collaboratively using only the collaborative model and e-mail. No talking with team members is allowed. The team will be required to perform the tasks under the Problem Formulation and Solution Planning stages of a collaborative model. Following the completion of the tasks required, I am required to fill out a questionnaire and participate in a debriefing session in my sections web board conference.

I was given a choice of either participating in this experiment or working on a similar project. My grade will be based on my ability to follow directions and the quality of my performance on the specific tasks.

My team has decided that the process facilitator is ____________________________ and the content facilitator is ____________________________. The facilitators have agreed to follow the instructions as outlined in the task document provided at the outset of this experiment. The facilitators will receive up to an extra 10 points added to their score depending on how well they carry out the role responsibilities.

PARTICIPANTS:

I will be one of about 192 participants to participate in this trial.

EXCLUSIONS:

I will inform the researcher if any of the following apply to me: N/A

RISK/DISCOMFORTS:

I have been told that the study described above may involve the following risks and/or discomforts:

There are no known risks or discomforts.

There also may be risks and discomforts that are not yet known – N/A.

CONFIDENTIALITY:

Every effort will be made to maintain the confidentiality of my study records. Officials of NJIT will be allowed to inspect sections of my research records related to this study. If the findings from the study are published, I will not be identified by name. My identity will remain confidential unless disclosure is required by law.

PAYMENT FOR PARTICIPATION:

I have been told that I will receive $0 compensation for my participation in this study.

CONSENT AND RELEASE:

I fully recognize that there are risks that I might be exposed to by volunteering in this study which are inherent in participating in any study; I understand that I am not covered by NJIT's insurance policy for any injury or loss I might sustain in the course of participating in the study.

I agree to assume and take on myself all risks and responsibilities in any way associated with this activity. I release NJIT, its trustees, agents, employees and students from any and all liability, claims and actions that may arise as a result of my participation in the study. I understand that this means that I am giving up my right to sue NJIT, its trustees, agents and employees for injuries, damages or losses I may incur.

RIGHT TO REFUSE OR WITHDRAW:

I understand that my participation is voluntary and I may refuse to participate, or may discontinue my participation at any time with no adverse consequence. I also understand that the investigator has the right to withdraw me from the study at any time.

INDIVIDUAL TO CONTACT:

If I have any questions about my treatment or research procedures that I discuss them with the principle investigator. If I have any addition questions about my rights as a research subject, I may contact:

Robin-Ann Klotsky, Executive Director of Research and Development at (973) 596-5227.

SIGNATURE OF PARTICIPANT

I have read this entire form, or it has been read to me, and I understand it completely. All of my questions regarding this form or this study have been answered to my complete satisfaction. I agree to participate in this research study.

Subject: Name: ____________________
Signature:__________________________

Date: ____________________________

SIGNATURE OF READER/TRANSLATOR IF THE PARTICIPANT DOES NOT READ ENGLISH WELL

The person who has signed above, ____, does not read English well, I read English well and am fluent in (name of the language) ____________________________________, a language the subject understands well. I have translated for the subject the entire content of this form. To the best of my knowledge, the participant understands the content of this form and has had an opportunity to ask questions regarding the consent form and the study, and these questions have been answered to the complete satisfaction of the participant (his/her parent/legal guardian).

Reader/Translator Name: ______________________
Signature: __________________________________
Date: ______________________________________

SIGNATURE OF INVESTIGATOR OR RESPONSIBLE INDIVIDUAL

To the best of my knowledge, the participant, ____, has understood the entire content of the above consent form, and comprehends the study. The participants and those of his/her parent/legal guardian have been accurately answered to his/her/their complete satisfaction.

Investigator's Name: ______________________
Signature: _______________________________
Date: ___________________________________

NEW JERSEY INSTITUTE OF TECHNOLOGY
323 MARTIN LUTHER KING BLVD.
NEWARK, NJ 07102

CONSENT TO PARTICIPATE IN A RESEARCH STUDY – Form 4

TITLE OF STUDY:

RESEARCH STUDY:

I, ____, have been asked to participate in a research study under the direction of Joanna DeFranco-Tommarello.
Other professional persons who work with her as study staff may assist to act for her.

PURPOSE:

To test the problem solving and software development effectiveness of groups.

DURATION:

My participation in this study will last for 3 weeks.

PROCEDURES:

I have been told that, during the course of this study, the following will occur:

My team will be given a software problem to solve collaboratively using only e-mail. No talking with team members is allowed. The team will be required to perform the tasks under the Problem Formulation and Solution Planning stages of a collaborative model. Following the completion of the tasks required, I am required to fill out a questionnaire and participate in a debriefing session in my sections web board conference.

I was given a choice of either participating in this experiment or working on a similar project. My grade will be based on my ability to follow directions and the quality of my performance on the specific tasks.

My team has decided that the process facilitator is ____________________________ and the content facilitator is ____________________________. The facilitators have agreed to follow the instructions as outlined in the task document provided at the outset of this experiment.

The facilitators will receive up to an extra 10 points added to their score depending on how well they carry out the role responsibilities.

PARTICIPANTS:

I will be one of about 192 participants to participate in this trial.

EXCLUSIONS:

I will inform the researcher if any of the following apply to me: N/A

RISK/DISCOMFORTS:

I have been told that the study described above may involve the following risks and/or discomforts:

There are no known risks or discomforts.

There also may be risks and discomforts that are not yet known – N/A.

CONFIDENTIALITY:

Every effort will be made to maintain the confidentiality of my study records. Officials of NJIT will be allowed to inspect sections of my research records related to this study. If the findings from the study are published, I will not be identified by name. My identity will remain confidential unless disclosure is required by law.

PAYMENT FOR PARTICIPATION:

I have been told that I will receive $0 compensation for my participation in this study.

CONSENT AND RELEASE:

I fully recognize that there are risks that I might be exposed to by volunteering in this study which are inherent in participating in any study; I understand that I am not covered by NJIT's insurance policy for any injury or loss I might sustain in the course of participating in the study.

I agree to assume and take on myself all risks and responsibilities in any way associated with this activity. I release NJIT, its trustees, agents, employees and students from any and all liability, claims and actions that may arise as a result of my participation in the study. I understand that this means that I am giving up my right to sue NJIT, its trustees, agents and employees for injuries, damages or losses I may incur.

RIGHT TO REFUSE OR WITHDRAW:

I understand that my participation is voluntary and I may refuse to participate, or may discontinue my participation at any time with no adverse consequence. I also understand that the investigator has the right to withdraw me from the study at any time.

INDIVIDUAL TO CONTACT:

If I have any questions about my treatment or research procedures that I discuss them with the principle investigator. If I have any addition questions about my rights as a research subject, I may contact:

Robin-Ann Klotsky, Executive Director of Research and Development at (973) 596-5227.

SIGNATURE OF PARTICIPANT

I have read this entire form, or it has been read to me, and I understand it completely. All of my questions regarding this form or this study have been answered to my complete satisfaction. I agree to participate in this research study.

Subject: Name: ____________________
Signature:___________________________

Date: ____________________________

SIGNATURE OF READER/TRANSLATOR IF THE PARTICIPANT DOES NOT READ ENGLISH WELL

The person who has signed above,
____, does not read English well, I read English well and am fluent in (name of the language) ___________________________________, a language the subject

understands well. I have translated for the subject the entire content of this form. To the best of my knowledge, the participant understands the content of this form and has had an opportunity to ask questions regarding the consent form and the study, and these questions have been answered to the complete satisfaction of the participant (his/her parent/legal guardian).

Reader/Translator Name: ______________________
Signature: ________________________________
Date: ____________________________________

SIGNATURE OF INVESTIGATOR OR RESPONSIBLE INDIVIDUAL

To the best of my knowledge, the participant, ____, has understood the entire content of the above consent form, and comprehends the study. The participants and those of his/her parent/legal guardian have been accurately answered to his/her/their complete satisfaction.

Investigator's Name: ______________________
Signature: ______________________________
Date: __________________________________

APPENDIX B

SURVEY INSTRUMENTS

This appendix contains the post-task questionnaire that the subjects filled out to measure solution satisfaction, process satisfaction and task validation.

POST-EXPERIMENT QUESTIONNAIRE

Group ID: ________
Last 4 digits of your SS#: ________
Class (601 or 602): ________
Section #: ________

1. I am very satisfied with the quality of my group's solution.

Strongly Agree			Undecided			Strongly Disagree
7	6	5	4	3	2	1

2. My group's problem solving process was efficient.

Strongly Agree			Undecided			Strongly Disagree
7	6	5	4	3	2	1

3. I am not confident in the group's final solution.

Strongly Agree			Undecided			Strongly Disagree
1	2	3	4	5	6	7

4. My group's problem solving process was coordinated.

Strongly Agree			Undecided			Strongly Disagree
7	6	5	4	3	2	1

5. I am very committed to my group's final solution.

Strongly Agree			Undecided			Strongly Disagree
7	6	5	4	3	2	1

6. My group's problem solving process was unfair.

Strongly Agree			Undecided			Strongly Disagree
1	2	3	4	5	6	7

7. The final solution and formal reports do not reflect my inputs.

Strongly Agree			Undecided			Strongly Disagree
1	2	3	4	5	6	7

8. My group's problem solving process was confusing.

Strongly Agree			Undecided			Strongly Disagree
1	2	3	4	5	6	7

9. I feel I had an equal part in the correctness of the group's final solution.

Strongly Agree			Undecided			Strongly Disagree
7	6	5	4	3	2	1

10. My group's problem solving process was satisfying.

Strongly Agree			Undecided			Strongly Disagree
7	6	5	4	3	2	1

11. I felt the task was too difficult.

Strongly Agree			Undecided			Strongly Disagree
1	2	3	4	5	6	7

12. I understood the task.

Strongly Agree			Undecided			Strongly Disagree
7	6	5	4	3	2	1

13. I felt there wasn't enough time to complete the task.

Strongly Agree			Undecided			Strongly Disagree
1	2	3	4	5	6	7

14. I felt that everyone on my team understood the task.

Strongly Agree			Undecided			Strongly Disagree
7	6	5	4	3	2	1

End of Questionnaire

Questionnaire to validate Post-Task Questionnaire

Questionnaire items for measuring satisfaction with the process:

How would you describe your group's problem-solving process?

q18.	efficient	1	2	3	4	5	inefficient
q19.	coordinated	1	2	3	4	5	uncoordinated
q20.	fair	1	2	3	4	5	unfair
q21.	understandable	1	2	3	4	5	confusing
q22.	satisfying	1	2	3	4	5	unsatisfying

Questionnaire items for measuring satisfaction with the outcome

q23. How satisfied or dissatisfied were you with the quality of your group's solutions

very dissatisfied 1 2 3 4 5 very satisfied

q43. To what extent does the final design and formal report reflect your inputs?

q45. To what extent are you confident that the group's solutions are correct?

q47. To what extent do you feel committed to the group's solutions?

q53. To what extent do you feel personally responsible for the correctness of the group's solutions (decision or recommendation)?

Scale for items 43,45,47,53:

Not at all	little extent	some extent	great extent	very great extent

1-------------------2------------------3-----------------4------------------5

APPENDIX C

TASK LISTS

This appendix shows the four different task lists used for each condition of the experiment.

Task List – Condition 1

Rules:

1. You are only to contact your team members via an e-mail list (see below) or the Groove application. It is very important that you do not talk to your team members on the phone or face-to-face.
2. Keep the daily schedule of this experiment. There are only tasks assigned Monday thru Friday. The tasks are not time consuming but an entire day is allotted to allow for every team member to accomplish the task.
3. Approximately half your grade will be based on how well each individual **follows the directions** and completes all the steps at the designated time. The other half of your grade will be based on the quality and timeliness of the required documents to be turned in:
    a. Consent form.
    b. Post-task questionnaire.
    c. Participating in the debriefing session
    d. Contributing/reviewing to the teams output documents:
        i. Problem Formulation Document: This document will contain the group's output from the days you worked on the ***Preliminary Problem Description, Preliminary Mental Model, and Structured Problem Representation.***
        ii. Solution Plan Document: This document will contain the group's output from the days you worked on, ***Strategy Discovery, Goal Decomposition, and Data Modeling.***

<u>Training</u>

Day 1 – February 5, 2002

1. Fill out Consent form. **Sign and Mail to:**
   NJIT College of Computing Sciences
   Attn: Joanna DeFranco-Tommarello (Group Experiment)
   University Heights
   Newark, NJ 07102-1982
2. Create a mailing list (distribution list) of your group listed in the course web board conference and include my e-mail <u>joannadt@njit.edu</u>. The **subject** of the e-mail **MUST** always include your ID.
    a. Every mail system creates mailing lists differently – you need to check the on-line help of your e-mail application.
    b. The purpose of the mailing list is to make sure everyone on your team knows what is going on. Treat me like a silent team member. I will only intervene if the collaborative activities are getting off track.
3. Send a test message to your group.
4. Choose a Content Facilitator and a Process Facilitator. Each person taking these jobs will receive up to an extra 10 points added to their final project

score depending on how well they carry out the role responsibilities. Detailed explanations of theses roles are described at the end of this document.
   a. Content Facilitator: This person is responsible for turning in the output documents created by the team.
   b. Process Facilitator: This person is responsible for initiating the tasks for each day of the experiment.

Day 2 – February 6, 2002

1. Download groove from http://www.groove.net/
2. Install it on your system.
3. Skim the Groove document to familiarize yourself with the application.
4. Do not create a project workspace unless you are the "Process Facilitator"
5. The Process Facilitator will be inviting you to join the project workspace today so make sure you have completed the tasks from Day 1. Refer to your groove document to perform the following tasks once you receive your invitation via e-mail from the process facilitator:
   a. Accept the invitation to join the project workspace.
   b. Use the Contacts tool and add your information.

Day 3 – February 7, 2002
1. Use the discussion tool to introduce yourself.
2. Familiarize yourself with the other tools in Groove by reading the Groove document and using the application.
3. Read the entire schedule. Three brainstorming activities are suggested to be scheduled. If possible agree on a time via e-mail or the discussion tool with your team members where you can all synchronously brainstorm to accomplish that day's task.

TASK:

The task is to plan a solution for a super market simulation program using the groupware application "Groove" and by following the tasks assigned to you each day. Neither implementation nor coding is required, only the solution's plan. Each task needed to devise a plan will be outlined for you in the schedule.

The final plan of the supermarket should include the different aspects of a supermarket using object-oriented concepts. The user of the simulation program should be able to input the customer frequency, the number of stockers re-stocking the shelves, and the number of cashiers working, where customer frequency is how often a customer will enter the store. Determine any additional objects needed to simulate a supermarket and what functions all of the objects need to perform during simulation. The output of the design will be any statistical information from the different objects in the supermarket you feel necessary.

Experiment Week 1

Day 1 – February 8, 2002

1. Create a *Preliminary Problem Description.* This is accomplished by:
    a. Each team member will evaluate the task and enter their interpretation of the task in the discussion tool.
    b. The team will decide which description or combination of descriptions the team will follow. This can all be accomplished in the discussion or brainstorming tool.
2. The Content Facilitator will **document the results in the Problem Formulation document**. Other team members should review this document for correctness.

Day 2 – February 11, 2002

1. Create a *Preliminary Mental Model.* This is accomplished by refining the problem description by determining the problem/task goals, givens, unknowns, conditions, and any requirements for understanding. Answer the following questions in the discussion tool. Come up with one answer for the entire group.

Q: What is the goal?
Q: Do any of these goals require clarification?
Q: Are there any other explicit or implicit problem requirements?
Q: What are the givens?
Q: Are there any flow control related inputs or givens?
Q: What are the unknowns?
Q: Are there any conditions and constraints.

2. The Content Facilitator will **document the results in the Problem Formulation document**. Other team members should review this document for correctness.

Day 3 – February 12, 2002

1. Finish the Preliminary Mental Model.
2. The Content Facilitator will **document the results in the Problem Formulation document**. Other team members should review this document for correctness.

Day 4 – February 13, 2002

1. Create a *Structured Problem Representation.* This is accomplished by extracting facts from the problem description. Basically you need to organize the information you produced from the Mental Model yesterday. Your output should look as follows:
    Goal: xxx
    Givens: xxx
    Unknowns: xxx
    Conditions and Constraints: xxx
2. The Content Facilitator will **document the results in the Problem Formulation document**. Other team members should review this document for correctness.

Day 5 – February 14, 2002

1. Work on *Strategy Discovery*. This means generate alternatives and select a solution strategy. **Content Facilitator will document *all alternatives* generated and the option chosen by the group in the Problem Formulation Document**. Answer the following questions for each alternative in the discussion tool or a brainstorming activity.

Q: Identify a strategy for solving the problem.
Q: Are there any difficulties to this strategy?
Q: Identify an alternative strategy for solving the problem.
Q: Are there any difficulties to this strategy?

<u>Experiment Week 2</u>

Day 6 – February 15, 2002

Strategy Discovery (Continued)

1. Finish the Strategy Discovery Task by answering the following questions after reviewing all the alternatives listed by your teammates.

Q: Identify the best alternative and explain your choice.
Q: Are there any special formulas and techniques needed to implement this strategy?

2. Have a vote using the discussion tool as to which alternative is best.
3. The Content Facilitator will document the results in the Solution Plan document. Other team members should review this document for correctness.

Day 7 – February 18, 2002

1. Today you will be performing *Goal Decomposition*. This is accomplished by breaking down the problem into major components. Use either the discussion or brainstorming tool. The output of this activity should be as follows.

Sub-goal 1
Sub-goal 2
Sub-goal 3
Sub-goal 4
Etc.

Day 8 – February 19, 2002

Goal Decomposition (Continued)

1. Finish the Goal Decomposition Task by distributing the goals among each team member.
2. Enter the distribution in the Task List tool.
3. The Content Facilitator will document the results in the Solution Plan document. Other team members should review this document for correctness.

Day 9 – February 20, 2002

1. Today you are working on *Data Modeling*. Organize and associate facts with your components. This is accomplished by integrating the givens and unknowns from the elicitation task with the refined goals identified at goal decomposition, yielding a preliminary data model. The output of this task should look as follows:

Input (givens)

Name	*Description/Units*	*Origin*	*Used in*
Given 1	*description*	*Where does this information originate.*	*Subgoal X*

Output (unknowns)

Name	*Description*	*Destination*	*Used in*
Output 1	*Type of output*	*Screen*	*Subgoal X*

Day 10 – February 21, 2002
Data Modeling (Continued)
1. Finish the Data Modeling Task.
2. The Content Facilitator will document the results in the Solution Plan document. Other team members should review this document for correctness.

February 22, 2002
1. Fill out questionnaire
2. E-mail your questionnaire to **joannadt@njit.edu** Subject of the e-mail MUST be "Questionnaire – your name - Section XXX"
3. Participate in the debriefing session located in your sections web board conference. In the debriefing session will be asked about the task, the interactions between your team mates, any modification you would make to the processes or task, and what you learned.

Process Facilitator Description

Day 2 of Training – February 6, 2002
1. After installing the Groove application on your pc:
    a. Create a project workspace in Groove.
    b. Invite you team members to join the project workspace (including me).
2. Use the schedule tool to display the team's schedule that is in this document. Use the monthly schedule, refer to the Groove Document, and input the daily tasks for each team member as well as any team activities such as brainstorming sessions.

Day 3 thru Day 10 – You are responsible for initiating any brainstorming activities, task list tool activities, and discussion tool activities noted in any day's tasks.

Content Facilitator Description

Day 2 of Training - February 6, 2002
After you have accepted your invitation to join the Groove conference, create the output documents and store them in the "documents tool" in groove.

Day 3 thru Day 10 – You are responsible for updating daily the output documents required for submission. The document format should have each day listed with the output following. The documents should always be available in the document tool.

Day 10 – E-mail me, joannadt@njit.edu both output documents (Problem Formulation and Solution Planning). The subject of your e-mail MUST be "Team ID – Section XXX – Output Documents".

Task List – Condition 2

Rules:

1. You are only to contact your team members via an e-mail list (see below) or the Groove application. It is very important that you do not talk to your team members on the phone or face-to-face.
2. Approximately half your grade will be based on how well each individual follows the directions and completes all the steps at the designated time. The other half of your grade will be based on the quality and timeliness of the required documents:
    a. Consent form.
    b. Post-task questionnaire.
    c. Participating in the debriefing session
    d. Contributing/reviewing to the teams output documents:
        i. Problem Formulation Document - This document will contain the following information:
            1. The problem description in your own words
            2. Any information you know regarding the problem
        ii. Solution Plan Document - This document will contain the following information:
            1. A strategy to accomplish a solution, i.e. document any alternatives you come up with and the final alternative chosen by the team.
            2. An exact plan to accomplish the solution.
            3. Any facts associated with the plan

<u>Training</u>

Day 1 – February 5, 2002

1. Fill out Consent form. Sign and Mail to:
   NJIT
   College of Computing Sciences
   Attn: Joanna DeFranco-Tommarello (Group Experiment)
   University Heights
   Newark, NJ 07102-1982

2. Create a mailing list (distribution list) of your group listed in the course web board conference and include my e-mail joannadt@njit.edu. The **subject** of the e-mail **MUST** always include your group ID.
    a. Every mail system creates mailing lists differently – you need to check the on-line help of your e-mail application.

   b. The purpose of the mailing list is to make sure everyone on your team knows what is going on. Treat me like a silent team member. I will only intervene if the collaborative activities are getting off track.
3. Send a test message to your group.
4. Choose a Content Facilitator and a Process Facilitator. Each person taking these jobs will receive up to an extra 10 points added to their final project score depending on how well they carry out the role responsibilities. Detailed explanations of theses roles are described at the end of this document.
   a. Content Facilitator: This person is responsible for turning in the output documents created by the team.
   b. Process Facilitator: This person is responsible for initiating the necessary activities to complete the task of the experiment.

Day 2 – February 6, 2002

1. Download groove from **http://www.groove.net/**
2. Install it on your system.
3. Skim the Groove document to familiarize yourself with the application.
4. Do not create a project workspace unless you are the "Process Facilitator"
5. The Process Facilitator will be inviting you to join the project workspace today so make sure you have completed the tasks from Day 1. Refer to your groove document to perform the following tasks once you receive your invitation via e-mail from the process facilitator:
   a. Accept the invitation to join the project workspace.
   b. Use the Contacts tool and add your information.

Day 3 – February 7, 2002

1. Use the discussion tool to introduce yourself.
2. Familiarize yourself with the other tools in Groove by reading the Groove document and using the application.
3. Read this entire document carefully. Discuss with your team via groove or e-mail how you want to allocate your time to complete this experimental task.

TASK:

The task is to plan a solution for a super market simulation program using the groupware application "Groove" and by following the tasks assigned to you each day. Neither implementation nor coding is required, only the solution's plan. Each task needed to devise a plan will be outlined for you in the schedule.

The final plan of the supermarket should include the different aspects of a supermarket using object-oriented concepts. The user of the simulation program should be able to input the customer frequency, the number of stockers re-stocking the shelves, and the number of cashiers working, where customer frequency is how often a customer will enter the store. Determine any additional objects needed to simulate a supermarket and what functions all of the objects need to perform during simulation. The output of the design will be any statistical information from the different objects in the supermarket you feel necessary.

You have two weeks to complete the output documents described. Begin on February 8th and end on February 21st. On February 22nd you have specific items to turn in that are outlined below.

February 22, 2002

1. Fill out questionnaire
2. E-mail questionnaire to **joannadt@njit.edu** Subject of the e-mail MUST be "Questionnaire – your name - Section X"
3. Participate in the debriefing session located in your sections web board conference. In the debriefing session will be asked about the task, the interactions between your team mates, any modification you would make to the processes or task, and what you learned.

Process Facilitator Description

Day 2 of Training – February 6, 2002

1.After installing the Groove application on your pc:
   a. Create a project workspace in Groove.
   b. Invite you team members to join the project workspace (including me).

2.Use the schedule tool to display the team's schedule that is in this document. Use the monthly schedule, refer to the Groove Document, and input the daily tasks for each team member as well as any team activities such as brainstorming sessions.

Day 3 thru Day 10 – You are responsible for initiating any brainstorming activities, task list tool activities, and discussion tool activities noted in any day's tasks.

Content Facilitator Description

Day 2 of Training – February 6, 2002

After you have accepted your invitation to join the Groove conference, create the output documents and store them in the "documents tool" in groove.

Day 3 thru Day 10 – You are responsible for updating daily the output documents required for submission. The document format should organized with each day listed with any output following that date. The documents should always be available for review in the document tool.

Day 10 – E-mail me, joannadt@njit.edu both output documents (Problem Formulation and Solution Planning). The subject of your e-mail MUST be "Team ID – Section XXX – Output Documents".

Task List – Condition 3

Rules:

1. You are only to contact your team members via an e-mail list (see below). It is very important that you do not talk to your team members on the phone or face-to-face. I am to be CC'd on all e-mails during this entire experiment.
2. Keep the daily schedule of this experiment. There are only tasks assigned Monday thru Friday. The tasks are not time consuming but an entire day is allotted to allow for every team member to accomplish the task.
3. Approximately half your grade will be based on how well each individual follows the directions and completes all the steps at the designated time. The other half of your grade will be based on the quality and timeliness of the required documents:
    a. Consent form.
    b. Post-task questionnaire.
    c. Participating in the debriefing session
    d. Contributing/reviewing to the teams output documents:
        i. Problem Formulation Document: This document will contain the group's output from the days you worked on the ***Preliminary Problem Description, Preliminary Mental Model, and Structured Problem Representation.***
        ii. Solution Plan Document: This document will contain the group's output from the days you worked on, ***Strategy Discovery, Goal Decomposition, and Data Modeling.***

Training

Day 1 – February 5, 2002

1. Fill out and sign Consent form. Mail to:
   NJIT
   College of Computing Sciences
   Attn: Joanna DeFranco-Tommarello (Group Experiment)
   University Heights
   Newark, NJ 07102-1982
2. Create a mailing list (distribution list) of your group listed in the course web board conference and **include me: e-mail** joannadt@njit.edu. The subject of the e-mail MUST always include your GROUP ID.
    a. Every mail system creates mailing lists differently – you need to check the on-line help of your e-mail application.
    b. The purpose of the mailing list is to make sure everyone on your team knows what is going on. Treat me like a silent team member. I will only intervene if the collaborative activities are getting off track.
3. Send a test message to your group.
4. Choose a Content Facilitator and a Process Facilitator. Each person taking these jobs will receive up to an extra 10 points added to their final project

score depending on how well they carry out the role responsibilities. Detailed explanations of theses roles are described at the end of this document.

a. Content Facilitator: This person is responsible for turning in the output documents created by the team.
b. Process Facilitator: This person is responsible for initiating the tasks for each day of the experiment.

Day 2 – February 6, 2002

Send an e-mail to your group to introduce yourself.

Day 3 – February 7, 2002

Read the entire schedule in this document. Three brainstorming activities are suggested to be scheduled. If possible agree on a time via e-mail with your team members where you can all semi-synchronously brainstorm using e-mail to accomplish that day's task.

TASK:

The task is to plan a solution for a super market simulation program using e-mail and by following the tasks assigned to you each day. Neither implementation nor coding is required, only the solution's plan. Each task needed to devise a plan will be outlined for you in the schedule.

The final plan of the supermarket should include the different aspects of a supermarket using object-oriented concepts. The user of the simulation program should be able to input the customer frequency, the number of stockers re-stocking the shelves, and the number of cashiers working, where customer frequency is how often a customer will enter the store. Determine any additional objects needed to simulate a supermarket and what functions all of the objects need to perform during simulation. The output of the design will be any statistical information from the different objects in the supermarket you feel necessary.

Experiment Week 1

Day 1 – February 8, 2002

1. Create a *Preliminary Problem Description.* This is accomplished by:
    c. Each team member will evaluate the task and e-mail their interpretation of the task.
    d. The team will decide which description or combination of descriptions the team will follow.
2. The Content Facilitator will **document the results in the Problem Formulation document**. Other team members should review this document for correctness.

Day 2 – February 11, 2002

2. Create a *Preliminary Mental Model.* This is accomplished by refining the problem description by determining the problem/task goals, givens, unknowns, conditions, and any requirements for understanding. Answer the following questions and come up with one answer for the group.

Q: What is the goal?
Q: Do any of these goals require clarification?
Q: Are there any other explicit or implicit problem requirements?
Q: What are the givens?
Q: Are there any flow control related inputs or givens?
Q: What are the unknowns?
Q: Are there any conditions and constraints.

2. The Content Facilitator will **document the results in the Problem Formulation document**. Other team members should review this document for correctness.

Day 3 – February 12, 2002

1. Finish the *Preliminary Mental Model.*
2. The Content Facilitator will **document the results in the Problem Formulation document**. Other team members should review this document for correctness.

Day 4 – February 13, 2002

1. Create a *Structured Problem Representation.* This is accomplished by extracting facts from the problem description. Basically you need to organize the information you produced from the Mental Model yesterday. Your output should look as follows:
   Goal: xxx
   Givens: xxx
   Unknowns: xxx
   Conditions and Constraints: xxx
2. The Content Facilitator will **document the results in the Problem Formulation document**. Other team members should review this document for correctness.

Day 5 – February 14, 2002

Work on *Strategy Discovery.* This means generate alternatives and select a solution strategy. **Content Facilitator will document *all alternatives* generated and the option chosen by the group in the Problem Formulation Document.** Answer the following questions for each alternative and e-mail them to your group.

Q: Identify a strategy for solving the problem.
Q: Are there any difficulties to this strategy?
Q: Identify an alternative strategy for solving the problem.
Q: Are there any difficulties to this strategy?

Experiment Week 2

Day 6 – February 15, 2002
Strategy Discovery (Continued)

1. Finish the Strategy Discovery Task by answering the following questions after reviewing all the alternatives listed by your teammates.

Q: Identify the best alternative and explain your choice.
Q: Are there any special formulas and techniques needed to implement this strategy?

2. Have a vote as to which alternative is best.
3. The Content Facilitator will document the results in the Solution Plan document. Other team members should review this document for correctness.

Day 7 – February 18 2002

1. Today you will be performing *Goal Decomposition.* This is accomplished by breaking down the problem into major components. The output of this activity should be as follows.

Sub-goal 1
Sub-goal 2
Sub-goal 3
Sub-goal 4
Etc.

Day 8 – February 19, 2002
Goal Decomposition (Continued)

1. Finish the Goal Decomposition Task by distributing the goals among each team member.
2. The Content Facilitator will document the results in the Solution Plan document. Other team members should review this document for correctness.

Day 9 – February 20, 2002

1. Today you are working on *Data Modeling.* Organize and associate facts with your components. This is accomplished by integrating the givens and unknowns from the elicitation task with the refined goals identified at goal decomposition, yielding a preliminary data model. The output of this task should look as follows:

Input (givens)

Name	***Description/Units***	***Origin***	***Used in***
Given 1	*description*	*Where does this information originate.*	*Subgoal X*

Output (unknowns)

Name	*Description*	*Destination*	*Used in*
Output 1	*Type of output*	*Screen*	*Subgoal X*

Day 10 – February 21, 2002
Data Modeling (Continued)
1. Finish the Data Modeling Task.
2. The Content Facilitator will document the results in the Solution Plan document. Other team members should review this document for correctness.

February 22, 2002
1. Fill out questionnaire that will be posted on the web board today.
2. E-mail your questionnaire to **joannadt@njit.edu** Subject of the e-mail MUST be "Questionnaire – your name - Section XXX"
3. Participate in the debriefing session located in your sections web board conference. In the debriefing session will be asked about the task, the interactions between your team mates, any modification you would make to the processes or task, and what you learned.

Process Facilitator Description

Day 2 of Training – February 6, 2002
Each day e-mail your group about that day's task/activity.

Day 3 thru Day 10 – You are responsible for initiating any brainstorming/voting activities, and discussion activities noted in any day's tasks.

Content Facilitator Description

Day 2 of Training – February 6, 2002
Create the output documents and store them on your PC.

Day 3 thru Day 10 – You are responsible for updating daily the output documents required for submission. The document format should have each day listed with the output following. The documents should always be e-mailed to your group each day.

Day 10 – E-mail me, joannadt@njit.edu both output documents (Problem Formulation and Solution Planning). The subject of your e-mail MUST be "Team ID – Section XXX – Output Documents".

Task List – Condition 4

Rules:

1. You are only to contact your team members via an e-mail list (see below). It is very important that you do not talk to your team members on the phone or face-to-face. I am to be CC'd on all e-mails during this entire experiment.
2. Approximately half your grade will be based on how well each individual follows the directions and completes all the steps at the designated time. The other half of your grade will be based on the quality and timeliness of the required documents:
    a. Consent form.
    b. Post-task questionnaire.
    c. Participating in the debriefing session
    d. Contributing/reviewing to the teams output documents:
        i. Problem Formulation Document - This document will contain the following information:
            1. The problem description in your own words
            2. Any information you know regarding the problem
        ii. Solution Plan Document - This document will contain the following information:
            1. A strategy to accomplish a solution, i.e. document any alternatives you come up with and the final alternative chosen by the team.
            2. An exact plan to accomplish the solution.
            3. Any facts associated with the plan

Training

Day 1 – February 5, 2002

1. Fill out Consent form. Sign and Mail to:
   NJIT
   College of Computing Sciences
   Attn: Joanna DeFranco-Tommarello (Group Experiment)
   University Heights
   Newark, NJ 07102-1982
2. Create a mailing list (distribution list) of your group listed in the course web board conference and include my e-mail joannadt@njit.edu. The subject of the e-mail MUST always include your group letter.
    a. Every mail system creates mailing lists differently – you need to check the on-line help of your e-mail application.
    b. The purpose of the mailing list is to make sure everyone on your team knows what is going on. Treat me like a silent team member. I will only intervene if the collaborative activities are getting off track.
3. Send a test message to your group.

4. Choose a Content Facilitator and a Process Facilitator. Each person taking these jobs will receive up to an extra 10 points added to their final project score depending on how well they carry out the role responsibilities. Detailed explanations of theses roles are described at the end of this document.
    a. Content Facilitator: This person is responsible for turning in the output documents created by the team.
    b. Process Facilitator: This person is responsible for initiating the necessary activities to complete the task of the experiment.

Day 2 – February 6, 2002

Send an e-mail to your group to introduce yourself.

Day 3 – February 7, 2002

Read this entire document carefully. Discuss with your team via e-mail how you want to allocate your time to complete this experimental task.

TASK:

The task is to plan a solution for a super market simulation program by following the tasks assigned to you each day. Neither implementation nor coding is required, only the solution's plan.

The final plan of the supermarket should include the different aspects of a supermarket using object-oriented concepts. The user of the simulation program should be able to input the customer frequency, the number of stockers re-stocking the shelves, and the number of cashiers working, where customer frequency is how often a customer will enter the store. Determine any additional objects needed to simulate a supermarket and what functions all of the objects need to perform during simulation. The output of the design will be any statistical information from the different objects in the supermarket you feel necessary.

You have two weeks to complete the output documents described. Begin on February 8th and end on February 21st. On February 22nd you have specific items to turn in that are outlined below.

February 22, 2002

1. Fill out questionnaire
2. E-mail your questionnaire to joannadt@njit.edu Subject of the e-mail MUST be "Questionnaire – your name - Section X"
3. Participate in the debriefing session located in your sections web board conference. In the debriefing session will be asked about the task, the interactions between your team mates, any modification you would make to the processes or task, and what you learned.

Process Facilitator Description

Day 2 of Training - February 6, 2002
Each day e-mail your group about that day's task/activity.

Day 3 thru Day 10 – You are responsible for initiating any activities noted in any day's tasks.

Content Facilitator Description

Day 2 of Training - February 6, 2002
Create the output documents and store them on your PC.

Day 3 thru Day 10 – You are responsible for updating daily the output documents required for submission. The document format should organized with each day listed with any output following that date.

Day 10 – E-mail me, joannadt@njit.edu both output documents (Problem Formulation and Solution Planning). The subject of your e-mail MUST be "Team ID – Section XXX – Output Documents".

Alternate Task List

Rules:

1. Your group has 3 weeks to do this project. It is due February 15, 2002.
2. You are only to contact your team members via an e-mail list (see below). It is very important that you do not talk to your team members on the phone or face-to-face.
3. Follow the activities listed to solve the task below.
4. Approximately half your grade will be based on how well you follow the directions and complete all the steps outlined. The other half of your grade will be based on the quality and timeliness of the required documents:
    a. Problem Formulation Document: This document will contain your output from the days you worked on the ***Preliminary Problem Description, Preliminary Mental Model, and Structured Problem Representation.***
    b. Solution Plan Document: This document will contain your output from the days you worked on ***Strategy Discovery, Goal Decomposition, and Data Modeling.***
    c. Submit these documents to me joannadt@njit.edu with the subject heading **"Alternate Task – Section XXX – Group ID".**

TASK:

The task is to plan a solution for a super market simulation program by following the tasks outlined for you. Neither implementation nor coding is required, only the solution's plan. Each task needed to devise a plan will be outlined for you in the schedule.

The final plan of the supermarket should include the different aspects of a supermarket using object-oriented concepts. The user of the simulation program should be able to input the customer frequency, the number of stockers re-stocking the shelves, and the number of cashiers working, where customer frequency is how often a customer will enter the store. Determine any additional objects needed to simulate a supermarket and what functions all of the objects need to perform during simulation. The output of the design will be any statistical information from the different objects in the supermarket you feel necessary.

1. Create a *Preliminary Problem Description.* This is accomplished by:
    e. Evaluating the task by writing down your interpretation of the task.
    f. Document the results in the Problem Formulation document.
2. Create a *Preliminary Mental Model.* This is accomplished by refining the problem description by determining the problem/task goals, givens, unknowns, conditions, and any requirements for understanding. Answer the following questions:

Q: What is the goal?

Q: Do any of these goals require clarification?

Q: Are there any other explicit or implicit problem requirements?
Q: What are the givens?
Q: Are there any flow control related inputs or givens?
Q: What are the unknowns?

3. Document the results in the Problem Formulation document.
4. Create a *Structured Problem Representation.* This is accomplished by extracting facts from the problem description. Basically you need to organize the information you produced from the Mental Model. Your output should look as follows:
   Goal: xxx
   Givens: xxx
   Unknowns: xxx
   Conditions and Constraints: xxx

5. Document the results in the Problem Formulation document.
6. Work on *Strategy Discovery.* This means generate alternatives and select a solution strategy. Answer the following questions.

Q: Identify a strategy for solving the problem.
Q: Are there any difficulties to this strategy?
Q: Identify an alternative strategy for solving the problem.
Q: Are there any difficulties to this strategy?
Q: Identify the best alternative and explain your choice.
Q: Are there any special formulas and techniques needed to implement this strategy?

7. Document the results in the Solution Plan document.
8. *Goal Decomposition.* This is accomplished by breaking down the problem into major components. The output of this activity should be as follows.

Sub-goal 1
Sub-goal 2
Sub-goal 3
Sub-goal 4
Etc.

9. Document the results in the Solution Plan document.
10. *Data Modeling.* Organize and associate facts with your components. This is accomplished by integrating the givens and unknowns from the elicitation task with the refined goals identified at goal decomposition, yielding a preliminary data model. The output of this task should look as follows:

Input (givens)

Name	***Description/Units***	***Origin***	***Used in***
Given 1	*description*	*Where does this information originate.*	*Subgoal X*

Output (unknowns)

Name	***Description***	***Destination***	***Used in***
Output 1	*Type of output*	*Screen*	*Subgoal X*

11. Document the results in the Solution Plan document.

REFERENCES

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

Anderson, J.R., The Architecture of Cognition, Cambridge, Massachusetts: Harvard University Press, 1983.

Anderson, W.L., "Group Relations Psychology and Computer Supported Work Some New Directions for Research and Development", ACM, 1991.

Barber, G., Chapter 9 of Human Factors and Interactive Computer Systems, Edited by Yannis Vassiliou, Ablex Publishing Corporation, Norwood, NJ, 1984.

Benbasat, I., Lim, L. "The Effects of Group, Task, Context, and Technology", Small Group Research, Volume 24, Issue 4, November 1993, pp. 430- 463.

Benbasat, I., Taylor, R.N., "Behavioral aspects of information processing for the design of management information systems", IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-12 (4), July/August 1982.

Bentley, R., Horstmann, T., Sikkel, K., Trevor, J., "Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System", The World Wide Web Journal: Proceedings of the 4th International WWW Conference, Issue 1, December 1995, pp. 63-74.

Bloom, B.S., (Ed.), Taxonomy of Educational Objectives, Handbook I: Cognitive Domain, New York, New York: McKay, 1956.

Booch, G., Object Oriented Design with Applications, Redwood City, California: Benjamin/Cummings, 1991.

Brereton, O., Lees, S., Bedson, R., Boldyreff, C., Drummond, S., Layzell, P., Macaulay, L., Young, R., "Student Collaboration across Universities: A Case Study in Software Engineering", Thirteenth Conference on Software Education and Training, pp. 76-86, March, 2000.

Butler, D., and P. Winne, "Feedback and self-regulated learning: A theoretical synthesis", Review of Educational Research, 65 (3), pp. 245-281, 1995.

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

Charles, R., F. Lester, and P. O'Daffer, "How to Evaluate Progress in Problem Solving", Reston, Virginia: National Council of Teachers of Mathematics, 1987.

Constantine, L., "Teamwork Paradigms and the Structured Open Team", Software Development '90: proceedings of Miller Freeman Publications, Oakland, California, pp. 87 – 93.

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

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.

Deek, F. P., "The Software Process: A Parallel Approach through Problem Solving and Program Development", Journal of Computer Science Education, Volume 9, Number 1, pp. 43-70, April 1999.

Deek, F.P., Hiltz, S.R., Kimmel, H., Rotter, N., "Cognitive Assessment of Students' Problem Solving and Program Development Skills", Journal of Engineering Education, Volume 88, Number 3, pp. 317-326, July 1999.

Deek, F., McHugh, J., "SOLVEIT: An Experimental Environment for Problem Solving and Program Development", Journal of Applied Systems Studies, Volume 2, Number 2, 2001.

Deek, F.P., McHugh, J., Hiltz, S.R., "Methodology and Technology for Learning Programming", to appear in Journal of Systems and Information Technology, 2000.

Deek, F., McHugh, J., "Empirical Evaluation of a Problem Solving and Program Development Environment", to appear in the Journal of Interactive Learning Research, 2002.

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.

Dennis, A.R., "Information Exchange and Use in Group Decision Making: You Can Lead a Group to Information, But You Can't Make it Think", Management Information Systems Quarterly, 20, 4, pp. 433-457, 1996.

Denning, R., Smith, P., "Teaching Problem-Solving Through a Cooperative Learning Environment", CHI '95 Mosaic of Creativity, pp. 9-10, May 1995.

Dennis, A.R., Tyran, C.K., Vogel, D.R., Nunamaker, J.F., "Group Support Systems for Strategic Planning", JMIS, 14, 1, pp. 155-184, 1997.

Dennis, A.R., Easton, A.C., Easton, G.K., George, J.F. and Nunamaker, J.F., "Ad Hoc versus Established Groups in an Electronic Meeting System Environment", Proceedings of the 23rd Hawaii International Conference on System Sciences, III, pp. 23-29, 1990.

Dennis, A.R., Nunamaker, J.F. and Vogel, D.R., "A Comparison of Laboratory and Field Research in the Study of Electronic Meeting Systems", Journal of Management Information Systems, 7, 3, pp. 107-135, 1991.

Dennis, A.R., Valacich, J.S., Connolly, T. and Wynne, B.E., "Process Structuring in Electronic Brainstorming", Information Systems Research, 7, 2, pp. 268-277, 1996.

Dennis, A.R., Valacich, J.S, "Computer Brainstorms: More Heads are Better than One", Journal of Applied Psychology, 78, 4, pp. 531-537, 1993.

Dennis, A.R., Valacich, J.S. and Nunamaker, J.F., "An Experimental Investigation of the Effects of Group Size in an Electronic Meeting Environment", IEEE Transactions on Systems, Man and Cybernetics, 25, 5, pp.1049-1057, 1990.

Dix, A., Beale, R., Remote Cooperation: CSCW Issues for Mobile and Teleworkers, Springer-Verlag London Limited, Great Britain, 1996.

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].

Dufner, D., Kwon, O., Park, Y., Qing, P., "Asynchronous Team Support: Perceptions of the Group Problem Solving Process When Using a CyberCollaboratory," HICSS-35, Los Alamitos, CA: IEEE Computer SocietyPress, January 6-10, Hawaii, 2002.

Ellis, C., Gibbs, S., Rein, G., "Groupware Some Issues and Experiences", Communications of the ACM, Volume 34, No. 3, pp. 39-58, January 1991.

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, pp. 211-221, September 1996.

Fischer, G., Lindstaedt, S., Ostwald, J., Stolze, M., Sumner, T., Zimmermann, B., "From Domain Modeling to Collaborative Domain Construction", pp. 75-85, DIS '95.

Fraser, S., Marshall, L., Bailetti, T., "Team Approaches to OO Design", OOPSLA, Vancouver, British Columbia, Canada, pp. 85-92, October 1992.

Fussell, S.R., Kraut, R.E., Lerch, F.J., Scherlis, W.L., McNally, M.M., Cadiz, J.J., "Coordination, Overload and Team Performance: Effects of Team Communication Strategies", CSCW, Seattle Washington, pp. 275-284, 1998.

Galegher, J., Kraut, R.E., "Computer-Mediated Communication and Collaborative Writing: Media Influence and Adaptation to Communication Constraints", CSCW, New York, pp. 155-162, 1992.

Gallupe, R. B., DeSanctis, G., Dickson, G., "Computer-Based Support for Group Problem-Finding: An Experimental Investigation", MIS Quarterly, pp. 277 – 296, June 1988.

Gallupe, R.B., Dennis, A. R., Cooper, W. H., Valacich, J.S., Nunamaker J., Bastianutti, L., "Electronic Brainstorming and Group Size", Academy of Management Journal, Volume 35, Number 2, pp. 350-369, 1992.

Gagne, R.M., The Conditions of Learning, Fourth edition, New York: Holt, Rinehart and Winston, 1985.

Gillmor, S., "A New Groove" Winter 2001 XML Magazine, Available [Online]: http://www.xmlmag.com/upload/free/features/xml/2000/05win00/ro0005/ro0005.asp, [31 March 2001].

Goldberg, M., "WebCT and first year: student reaction to and use of a web-based resource in first year Computer Science", pp. 127-129, ITiCSE '97.

Grinberg, L., Sor, D., de Bianchedi, E. T., Introduction to the Work of Bion, Jason Aronson, New York, 1977.

GroupSystems.com, Available [Online]: http://www.groupsystems.com [22 May 2001].

Grudin, J., "Groupware and Social Dynamics", Communications of the ACM, Volume 37, Number 1, pp. 93–105, 1994.

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, pp. 43-45, 1996.

Harasim, L., "A Framework for Online Learning: The Virtual-U", IEEE, pp. 44-49, September 1999.

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, pp. 83-101, September 1990.

Hare, A. P., Handbook of Small Group Research, Second Edition, The Free Press, New York, 1976.

Herbsleb, J., Kuwana, E., "An Empirical Study of Information Needs in Collaborative Software Design", Transactions of Information Processing Society of Japan, Volume 39, No. 10, October 1998.

Hepplestone, S., "coMentor News", December 2000, Available [Online]: http://comentor.hud.ac.uk, [09 December 2000].

Hiltz, S.R., Turoff, M., "Structuring Computer-Mediated Communication Systems to avoid Information Overload", CACM, Vol. 28, Number 7, pp. 682-689, July 1985.

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

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

Huitt, W., "Problem solving and decision making: Consideration of individual differences using the Myers-Briggs Type Indicator", Journal of Psychological Type, Volume 24, pp. 33-44, 1992.

Hussain, D., Hussain, K. M., Information Resource Management, Irwin Inc., Homewood, Illinois, 1984.

Jacobson, I., Booch, G., Rumbaugh J., The Unified Software Development Process, Reading, Massachusetts: Addison-Wesley, 1999.

Janis, I., Groupthink: Psychological Studies of Policy Decision, 1982.

Jarvela, S., Bonk, C.J., Lehtinen, E., Lehti, S., "A Theoretical Analysis of Social Interactions in Computer-Based Learning Environments: Evidence for Reciprocal Understandings", Journal of Educational Computing Research, Volume 21, Number 3, pp. 363-388, 1999.

Jarzabek, S., Huang, R., "The Case For User-Centered CASE Tools", Communications of the ACM, Vol. 41, Number 8, pp. 93-99, August 1998.

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

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

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

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, pp. 776-791, 1998.

Kirkpatrick, D., "Here Comes the Payoff from PCs", Fortune Magazine, March 1992.

Klein, M., Chapter 11 of <u>Design Issues in CSCW</u> – Edited by Dan Diaper And Colston Sanger, Springer-Verlag London Limited, Great Britain, 1994.

Lauer, T.W., Peacock E., Graesser, A.C., <u>Questions and Information Systems</u>, Hillsdale, New Jersey: Lawrence Erlbaum, 1992.

Lyles, M.A., Mitroff, I.I., "Organizational problem formulation: An empirical study", Administrative Science Quarterly, 25, pp. 102-119, 1980.

MacGregor, S.K., "Collaborative Learning: Reframing the Classroom", <u>Collaborative Learning: A Sourcebook for Higher Education</u>, University Park, PA, National Center on Post-secondary Teaching and Learning Assessment, 1992.

Mahaney, R., Lederer, A., "Runaway Information Systems Projects and Escalating Commitment", Proceedings of the 1999 ACM SIGCPR conference on Computer personnel research, pp. 291-296, 1999.

McCracken, M., Waters, R., "WHY? When an Otherwise Successful Intervention Fails", ITiCSE, June 1999.

McGuire, E., Randall, K., "Process Improvement Competencies for IS Professionals: A Survey of Perceived Needs", Boston, MA, pp. 1-8, CPR 98.

Miller, G., "The magical number seven, plus or minus two", Psychological Review, 63 (2), pp. 81-97, 1956.

Mintzberg, H., The Structuring of Organizations, Prentice-Hall Inc., Englewood Cliffs, N.J., 1979.

Mitroff, I.I., and M. Turoff, "Technological forecasting and assessment: science and/or mythology?", Technological Forecasting and Social Change, 5, pp. 113-134, 1973.

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, pp. 400-401, 1995.

Newell, A., and Simon, H.A., Human Problem Solving, Prentice-Hall, Englewood Cliffs, NJ, 1972.

Nidamarthi S., Allen R.H., Sriram R.D., "Observations From Supplementing the Traditional Design Process via Internet-based Collaboration Tools", International Journal of Computer Integrated Manufacturing, Volume 14, Number 1, pp. 95-107, 2001.

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.

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

Nunamaker, J.F., Dennis, A.R., Valacich, J.S., Vogel, D.R. and George, J.F., "Electronic Meeting Systems to Support Group Work", Communications of the ACM, 34, 7, pp. 40-61, 1991b.

Nunamaker, J. F., "The Case for Virtual Teaming Systems", IT Pro, pp. 52-57, October 1999.

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

Nunamaker, J., Briggs, R., Mittleman D., "Lessons from a Decade of Group Support Systems Research", HICSS, Volume 3, pp. 418-427, 1996.

Ocker, R., Hiltz. S. R., Turoff, M., Fjermestad, J., "Computer Support for Distributed Asynchronous Software Design Teams: Experimental Results and Creativity and Quality", HICCS, 1995.

Ocker, R., "The elationship Between Interaction, Group Development, and Outcome: A Study of Virtual Communication", HICCS, 2001.

Ocker, R., Fjermestad, J., "Web-Based Computer-Mediated Communication: An Experimental Investigation Comparing Three Communication Modes for Determining Software Requirements", HICCS 1998.

Ocker R., Fjermestad, J., Hiltz S. R., Turoff, M., Johnson, K, "Effects of Four Modes of Group Communication on the Outcomes of Software Requirements Determination", Journal of Management Information Systems, Volume 15, Issue 1, pp. 99-118, 1998.

Ocker, R., Hiltz, S. R., Turoff, M., Fjermestad, J., "The Effects of Distributed Group Support and Process Structuring on Software Requirements Development Teams", Journal of Management Information Systems, Volume 12, Issue 3, pp. 127 – 153, 1995.

Olson, G., Olson, J., "Distance Matters", Human Computer Interaction, Volume 15, pp. 139-178, 2000.

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

Polack-Wahl, J., "Group Projects: Woman and Men Can Work Together in the Computer Science Realm", International Symposium on Technology and Society-Woman and Technology, pp. 245-248, July 1999.

Polya, G., How to Solve It, Princeton, New Jersey: Princeton University Press, 1945.

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

Quintas, P., Chapter 1 of Social Dimensions of Systems Engineering, Edited by Paul Quintas, Redwood Books, Trowbridge, Wiltshire, Great Britain, 1993.

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

Romano, N., Nunamaker, J., Briggs, R., Vogel, D., "Architecture, Design and Development of an HTML/JavaScript Web-Based Group Support System", JASIS, Volume 49, Issue 7, pp. 649-667, 1998.

Rossetti, M., Nembhard, H., "Using Cooperative Learning to Activate Your Simulation Classroom", Proceedings of the 1998 Winter Simulation Conference, pp. 67-74.

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, pp. 196, 1999.

Rubinstein, M., Patterns of Problem Solving, Englewood Cliffs: New Jersey, Prentice Hall, 1975.

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

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

Simon, H.A., The Sciences of the Artificial, Cambridge, Massachusetts: MIT Press, 1969.

Simon, H. A., The New Science of Management, Harper and Row, New York, 1960.

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

Sonnentag, S., Brodbeck, F., Heinbokel, T., Stolte, W., "Stressor-burnout relationship in software development teams", Journal of Occupational and Organizational Psychology, Volume 67, pp. 327-344, December 1994.

Sommerville, I., Software Engineering, Fifth Edition, Addison-Wesley Publishers, 1996.

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

Stefik, M., Foster, G., Bobrow, D., Kahn, K., Lanning, S., Suchman, L., "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings", Communications of the ACM, Volume 30, Number 1, pp. 32 –47, January 1987.

Steiner, I.D. Group Process and Productivity, Academic Press: New York, 1972.

Stepien, W.J., Gallagher S. A., Workman D., "Problem-based learning for traditional and interdisciplinary classrooms", Journal for the Education of the Gifted, 16, (4), pp. 338-357, 1993.

Sternberg, R.J., Beyond IQ: A Triarchic Theory of Human Intelligence, Cambridge, Massachusetts: Cambridge University Press, 1985.

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].

Thomas, J., Chapter 2 of Human Factors and Interactive Computer Systems, Edited by Yannis Vassiliou, Ablex Publishing Corporation, Norwood, NJ, 1984.

Tinzmann, M.B., Jones, B.F., Fennimore, T.F., Bakker, J., Fine, C., Pierce, J., "The Collaborative Classroom", NCREL, Oak Brook, 1990.

Tinzmann, M. B., Jones, B. F., Fennimore, T. F., Bakker, J., Fine, C., Pierce, J., "What is the Collaborative Classroom?", NCREL, Oak Brook, 1990, Available [Online]: http://trackstar.hprtec.org/main/display.php3?option=frames&track_id=2897 [26 November 2000].

TogetherSoft Corporation,Available [Online]: http://www.togethersoft.com [20 April 2001].

Tremblay, J.P., and R.B Bunt, Introduction to Computer Science: An Algorithmic Approach, second edition, New York: McGraw-Hill, 1989.

Turoff, M., Chapter 12 of Human Factors and Interactive Computer Systems, Edited by Yannis Vassiliou, Ablex Publishing Corporation, Norwood, NJ, 1984.

Turoff, M., "Designing a Virtual Classroom", 1995 International Conference on Computer Assisted Instruction, ICCAI '95.

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

Ware, J., "Problem Solving and Conflict Resolution in Groups", Chapter 17 in Managing People and Organizations, Edited by John J. Gabarro, Harvard Business School Publications, Boston, Massachusetts, 1992.

Whitworth, B., Gallupe, G. and McQueen, R. "A Cognitive Three-Process Model of Computer-Mediated Group Interaction", Group Decision and Interaction, Volume 9, pp. 431-456, 2000.

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

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

Venkatesh, V. "Determinants of Perceived Ease of Use: Integrating Control, Intrinsic Motivation, and Emotion into the Technology Acceptance Model," Information Systems Research, 11,4 (Dec 2000)

Volkema, R.J., "Problem formulation in planning and design, Management Science, vol. 29, pp. 639-652, 1983.

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

Zwass, V., <u>Foundations of Information Systems</u>, Irwin McGraw-Hill, Boston, Massachusetts, 1998.