INFLUENCE OF CODIFIED KNOWLEDGE ON SOFTWARE DESIGN TASK PERFORMANCE: A COMPARISON OF PAIRS WITH INDIVIDUALS

by

GEORGE MANGALARAJ

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2006

ACKNOWLEDGEMENTS

I would like to express my gratitude to my dissertation committee chairs, Dr. Radha Mahapatra and Dr. Sridhar Nerur for their consistent support and encouragement throughout my Ph.D. program. I am grateful for Dr. Mahapatra's advice and guidance during key periods in my Ph.D. student life. Dr. Nerur helped me in many ways and I am forever indebted for them. He gave me the opportunity to work on his research projects. Dr. Nerur also sacrificed many of his weekends to conduct the seminar that was offered in conjunction with my dissertation research.

I would also like to express my gratitude to other members of my committee for their support and assistance. Dr. Kenneth Price spent countless hours in guiding me through the nuances of conducting experiments and analysis of data thereof. Dr. James Teng exposed me to various theories and the importance of them. Dr. Craig Slinkman introduced me to the concept of extreme programming. Dr. Mark Eakin guided me in analyzing the data.

I would also like to thank Anil Singh and Aakash Taneja for their friendship and they were always a source of support all through the years. Special thanks to Venugopal Balijepally for giving me an opportunity to see and learn the logistics of running an experimental study.

Many of my friends in the Ph.D. program helped me at various points during my dissertation research. Ralph Yeh and Vishal Sachdev spent hours grading the solutions

and I am grateful for that. Kishen Iyengar, Vikram Bhaduria, and Anil Gurung helped me in conducting some of the experimental sessions.

Last, but not the least, I would like thank my family members for their support. My father was instrumental in motivating me to pursue higher studies and I am thankful for that. I also thank my mom and my wife Johnsy, for their patience and understanding during the time I pursued this research.

November 29, 2006

ABSTRACT

INFLUENCE OF CODIFIED KNOWLEDGE ON SOFTWARE DESIGN TASK PERFORMANCE: A COMPARISON OF PAIRS WITH INDIVIDUALS

Publication No. _____

George Mangalaraj, Ph.D.

The University of Texas at Arlington, 2006

Supervising Professor: Dr. Radha Mahapatra

The need to improve the success rates of software development projects has prompted the software engineering community to come-up with various initiatives. These initiatives include: new software development processes such as Extreme Programming (XP), leveraging the development process by reusing existing knowledge of software artifacts. XP utilizes pairs in the performance of various software development tasks. Moreover, development of software applications is a knowledge intensive process that utilizes both tacit and explicit knowledge. This experimental study utilized software development professionals as subjects and manipulated the

mode of participation (individual or pairs) and availability of codified knowledge (design patterns). Results of the study indicate that the performance of collaborating pairs were better than the 2nd best individual in nominal pairs. Collaborating pairs also took more time than the average time taken by nominal pairs to complete the task and they were more satisfied than the individuals. This study also found that the codified knowledge in the form of design patterns helped in arriving at a better solution. One interesting finding of this study is the effect of design self-efficacy/collective-efficacy on task performance.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The need to improve software development project success has prompted the software engineering community to implement various initiatives. These initiatives include: efforts aimed at developing better software development processes such as Extreme Programming (XP), and reuse of existing knowledge and software artifacts.

Agile software development methodologies are a new breed of software development methodologies that promise higher success rates for software development projects. According to a recent survey, agile software development methodologies are gaining widespread acceptance in the industry (Ambler, 2006). These methodologies are strikingly different from the traditional software development methodologies. Extreme programming is a popular Agile Software Development Methodology which calls for the use of pairs in development of software applications.

Pairs are claimed to provide better solutions than individuals. Efficacy of pairs working in software programming tasks has been studied to a great extent in the past whereas studies that compared the performance of pairs with individuals in software design tasks are limited (Al-Kilidar, et al., 2005). Characteristics of software design tasks are different from those of programming tasks. Errors that occur during the design

phase of software development would require far greater resources to fix than the errors that occur in other phases of the project (Beck, 1999).

Development of software applications is a knowledge intensive process utilizing both tacit and explicit knowledge. The development of software application has often been characterized as knowledge work. Knowledge has also been described as the raw material in software development (Walz, et al., 1993). The development of software applications require the integration of various knowledge elements (Robillard, 1999). Two types of knowledge had been identified by researchers in developing software applications: domain knowledge and technical knowledge (Tiwana, 2004). These knowledge elements contain both explicit and tacit components. Integration of these diverse knowledge elements is required to effectively meet the user requirements of a particular system.

In order to manage the knowledge intensive process, various tools and techniques are available during the development of software applications. One such technique is the use of design patterns. Design patterns are formulated based on the knowledge of expert software designers and they provide standard solutions to recurring software design problems. Design patterns codify software design knowledge that can be readily used. Design patterns also enable novices to apply expert solutions for frequently appearing design problems. Differences between experts and novices in the development of software have been highlighted in the literature. Mayer (1988) illustrates how experts and novices differ in the use of syntactic, semantic, schematic, and strategic knowledge during computer programming task. Experts are claimed to

formulate higher-level conceptual models whereas novices in general lack this capability. Literature in the knowledge transfer area mentions the difficulty in transferring knowledge from one situation to another. Kerievsky (2005) claims that the availability of design patterns knowledge does not guarantee their use in a design task. Cognitive science approach to knowledge transfer also views the availability of knowledge to solve a particular problem does not guarantee the application of it to solve similar problems (Gray and Orasanu, 1987).

The experiment proposed and conducted in this study utilized individuals and pairs in the performance of software design tasks. This study also manipulated the availability of codified knowledge during the performance of the tasks. Results from this study are expected to unravel the effectiveness of practices to codify software development knowledge and also the efficacy of performing software design tasks in pairs. Findings of this study will also be applicable to the practitioners in areas such as knowledge management, software reuse, dyads versus individuals' performance on software design tasks, and group processes variables that are salient.

1.1 Importance of Research

Companies are investing millions of dollars in the development of new software applications. Literature highlights that many of the software development projects fail to meet the project objectives. According to a report compiled by the Standish group, only 28% of the IT projects were successful in meeting time, budget and functionality criteria (Standish Group, 2001). A more recent report, published in 2004 placed the project success rates at 29%, thus showing virtually no change in the success of

software development projects. Design of software applications is one of the important phases in the development of any software application. In a specific software development project, requirements specification and design phases consume more than half the development cost (Walz, et al., 1993). Expensive errors are claimed to be committed during the design phase of a project (Guindon, 1990). Moreover, the cost of fixing a software bug early in the design phase is lot cheaper than fixing the bug after the software application is fully implemented (Beck, 1999). This highlights the importance of the design phase in software development.

Scholarly research on project failures had found variety of reasons such as escalation of commitment to failing projects, non reporting of negative information about projects in due time, and improper management of software development project risks (Montealegre and Keil, 2000; Smith and Keil, 2003; Wallace and Keil, 2004). Some of the reasons attributed to the success of the software development projects are: user involvement, minimized scope with small milestones, standard software infrastructure, and adherence to formal software development methodology (Standish Group, 2001). The software engineering community and the industry has come up with ways and means to increase the success of software projects. These initiatives include: efforts to create better software development processes, and reuse of existing knowledge and software artifacts.

1.1.1 Software Development Process Improvements

There are divergent opinions on how the software development process should be improved. On the one end, there is a move to make software development processes

more formalized. This is reflected in the efforts of the Software Engineering Institute's Capability Maturity Model (CMM). CMM defines five levels of process maturity and organizations are rated based on their process maturity levels. Organizations at level 1 of process maturity do not have formal software development processes in place. Organizations at level 5 of process maturity have optimized software development processes (Boehm and Turner, 2004).

On the other end, there are advocates for software development methodologies that are less process oriented. The movement for Agile Software Development is an example of this phenomenon. Agile software development encompasses a wide array of methodologies that subscribe to the "Agile Manifesto" proposed by the Agile Alliance (Highsmith, 2002). Examples of agile methods are: Extreme Programming, SCRUM, Adaptive Software Development, Dynamic Systems Development Methodology, and Feature Driven Development.

Extreme programming (XP) is one of the most popular agile software development methodologies. XP utilizes individuals working together in pairs on the software development tasks. Some of the benefits of pairing are: reduction of programming errors that leads to better software quality, transfer and sharing of knowledge between the pair, and increased morale in the development group (Beck, 1999). There are anecdotal as well as some empirical evidence supporting this claim.

1.1.2 Knowledge Reuse and Software Artifacts

The software engineering community has developed tools and technologies that assist in the reuse of existing software artifacts. Reuse of existing artifacts help in

improving the productivity as well as the quality of software development projects (Prieto-Diaz, 1993). There are various levels at which reuse of software artifacts can be done. At the simplest level, concrete implementations such as code or modules are reused. Reuse of common class libraries is an example of simple reuse. At a more complex level, abstract software design components are reused after suitable adaptation. Reuse of design patterns is an example of complex reuse.

Design patterns capture the expertise of seasoned software developers and they provide standard solutions to recurring software design problems. The concept of patterns had its origin in the pioneering work of Chris Alexander in the field of architecture. There are many different types of design patterns available to meet the needs of various design problems. Design patterns are a form of codified knowledge. Application of such knowledge to solve a new problem is akin to knowledge transfer. Some of the benefits of using design patterns are: provides standard vocabulary among developers, documents the design, abstracts the design problem from the details, and creates adaptable solutions (Cline, 1996). There are a handful of empirical studies on design patterns (Prechelt, et al., 2001; Prechelt, et al., 2002).

1.2 Research Questions

This study addresses the following research questions:

1. Do software designers working in pairs have a greater effect on the outcome of the design task than individuals?
2. Does the use of codified knowledge in the form of design patterns, influences the outcome of the design task?

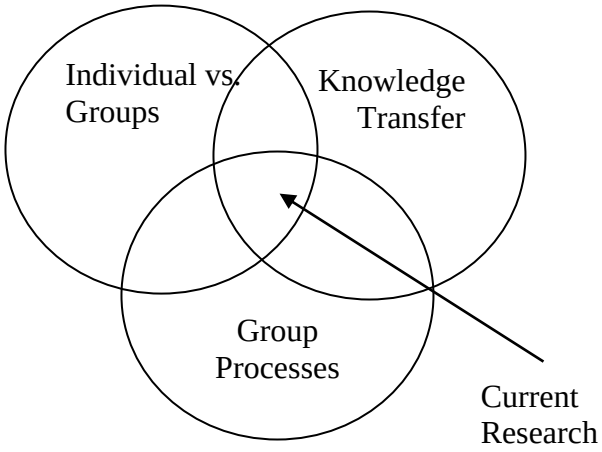
1.3 Overview of this Research

Here is an overview of this dissertation research: Chapter 2 reviews literature from four areas that were found to be pertinent to this study. These areas are: knowledge transfer, groups versus individuals, software engineering, and group processes. Chapter 3 develops the research models and discusses the theoretical rationale to support the model. Chapter 4 outlines the research design to be used and discusses the data analysis technique to be adopted. Measurements of various constructs are also presented in this section. Chapter 5 presents the results of the various statistical analyses that were performed. The final chapter discusses the findings of the study and presents future research directions.

CHAPTER 2

LITERATURE REVIEW

This chapter reviews literature pertaining to knowledge transfer and the performance of software design tasks by pairs. In consonance with the questions posed in the study, this chapter reviews the pertinent literature in knowledge transfer, individual and group differences in task performance, group processes and software engineering. The first section reviews the literature on knowledge transfer with emphasis on knowledge transfer in groups and knowledge transfer in software development. The second section reviews literature on the performances of groups versus individuals on a variety of tasks. The third section reviews literature on group processes such as collective efficacy beliefs and communication, which influence the functioning of groups. The fourth section reviews literature on software engineering with emphasis on characteristics of design tasks, reuse of software artifacts, and design patterns. Figure 2.1, graphically depicts the research areas to be reviewed for this study.



Task Environment: Software design problem

Figure 2.1: Research areas to be reviewed

2.1 Knowledge Transfer

Knowledge transfer refers to how knowledge acquired in one situation applies or fails to apply in other situations (Singley and Anderson, 1989). Failure in the transfer of knowledge indicates that availability of knowledge about a task does not always lead to superior performance. The individual must also know how to apply the knowledge to the task at hand (Singley and Anderson, 1989).

Researchers distinguish “transfer of knowledge” from “learning” on the basis of the task performed after the learning episode. In “transfer of knowledge”, the follow-up task is different from the task performed during the learning episode, whereas in “learning” the same task is repeated (Gick and Holyoak, 1987).

Knowledge transfer has been studied at various levels of analysis such as individual, group, and organizational level. Knowledge transfer at the individual level has been studied to a greater extent in the literature. These studies pertain to the transfer

of knowledge during training sessions (Argote, et al., 2000). Researches in the strategic management area were primarily oriented at the organizational level outcomes of knowledge transfer. Zellmer-Bruhn (2003) highlights that the existing research in knowledge transfer is predominantly oriented at the macro level and relatively less attention has been given to group level knowledge transfer activities. Importance of studying knowledge management at the group level has also been underscored in the literature (Nonaka and Takeuchi, 1995).

2.1.1 Knowledge

Knowledge is a multifaceted concept with multilayered meanings. The never-ending search, for the meaning, is characterized by the history of philosophy (Nonaka, 1994). This study does not go into the details about the definition of knowledge as enunciated by philosophers and researchers. Though there is no consensus in the definition of knowledge, there is agreement in the various types of knowledge. Different types of knowledge and their implications has been discussed in the literature (Blackler, 1995). Two common types of knowledge are tacit and explicit knowledge.

Tacit Knowledge: Tacit knowledge is rooted in action, experience, and involvement in a specific context (Alavi and Leidner, 2001). Tacit knowledge encompasses insights, intuition, and implied assumptions (Majchrzak, et al., 2004).

Explicit Knowledge: Explicit knowledge is articulated, codified, and communicated in symbolic and normal language (Alavi and Leidner, 2001). Explicit knowledge has also been referred to as codified knowledge, declarative knowledge, etc.

2.1.2 Knowledge Transfer: Individual Level

Transfer of knowledge in individuals across broad domains and tasks has been studied extensively in the past. Locke propounded the doctrine of formal discipline which claimed that the mind was composed of a collection of general faculties, and transfer between dissimilar domains are possible due to the utilization of common faculty (Singley and Anderson, 1989). Refuting this faculty view of transfer, which emphasized on more broader transfer of knowledge, Throndike proposed the theory of identical elements (Thorndike, 1906). According to this theory, knowledge transfer is much more specific in nature. For example, knowing addition helps learning multiplication and this has been credited to the identical elements present between them (Singley and Anderson, 1989). Theory of identical elements met with limited empirical support. Cognitive theories such as Gestalt theory consider transfer as a pattern of dynamic relationship discovered or understood in one situation may be applicable to another (Bower and Hilgard, 1981). Unlike Throndike's identical elements view, Gestalt theory considers that identical elements alone cannot bring about transfer of knowledge and there should be common patterns, configurations, or relationships.

Problems with transfer of knowledge bring to focus the need to consider similarities between the original problem and the subsequent problem. One of the problems associated with knowledge transfer is that people tend to access previous knowledge that bears surface rather than structural similarity to the problem at hand (Thompson, et al., 2000).

2.1.3 Knowledge Transfer: Group Level

Two types of studies are prevalent at the group level knowledge transfer. These are: within group knowledge transfer (Thomas-Hunt, et al., 2003) and across team knowledge transfer (Kane, et al., 2005). Research in social cognition suggests that in team level problem solving, knowledge of the individual member should be retrieved and articulated in order for it to be transferred to the new problem (Larson and Christensen, 1993).

Hansen (1999) studied transfer of codified knowledge between organizational units. The codified knowledge studied was either context dependent or context independent. Strength of ties was found to play a major role in the transfer of knowledge between various organizational units.

2.1.4 Knowledge Management Processes

Research on issues related to knowledge management has been the focus of attention of both Management and Information Systems researchers. Corporations are spending enormous amounts of money on building Knowledge Management Systems. Knowledge management processes has been classified in the literature either as knowledge creation or knowledge reuse. Though the literature gives importance to knowledge creation, knowledge reuse is clearly related to organizational effectiveness (Markus, 2001). Argote et al. (2003) classifies studies in knowledge management based on the knowledge management context and knowledge management outcomes. Table 2.1 presents the classification of knowledge management. Knowledge management outcome consists of knowledge creation, knowledge retention, and knowledge transfer.

The context in which knowledge management takes place are the properties of units involved in knowledge management, properties of the relationship between units involved in knowledge management, and the properties of the knowledge that is managed.

The current study is primarily interested in the knowledge transfer aspect of knowledge management. Knowledge transfer is evident when experience acquired in one context is transferred to another context (Argote, et al., 2000). The context for knowledge transfer is based on the properties of units, relationships, and knowledge and is highlighted in figure 2.1. Properties of units refer to the properties of individuals / groups involved in knowledge transfer. Properties of relationship between units refer to things such as the network structure. Properties of knowledge refer to the type of knowledge used such as explicit or tacit and this study utilized codified knowledge.

Table 2.1 Knowledge Management Classification

	Knowledge Management Context		
Knowledge Management Outcome	Properties of units	Properties of the relationship between units	Properties of knowledge
Creation			
Retention			
Transfer			

2.1.5 Transfer of Knowledge in Software Development

Though Information Systems researchers have extensively studied, systems that help transfer knowledge or reuse existing components, few of the studies are specifically focused on knowledge transfer during software development. Knowledge

transfer in virtual software development teams was studied by (Sarker, et al., 2005). Their study found that credibility, communication, capability, and culture of the software development team to influence knowledge sharing. Tiwana and McLean (2005) studied how expertise of software development team members was integrated to achieve team creativity. They argued that an individual's expertise is of no use unless it is transferred/shared among team members. They also highlight the difficulty in achieving this transfer during software development. Faraj and Sproull (2000) coined the term 'expertise coordination' to highlight the importance of knowledge in software development. Expertise coordination consists of socially shared cognitive processes that develop and evolve in order to meet task-based skill and knowledge dependencies. Their empirical study on software development teams found that expertise coordination plays a significant role in explaining team performance above and beyond traditional factors.

Walz, et al.(1993) specifically studied the role of knowledge acquisition, sharing, and integration during the design of software applications. They carried out an industrial case-study and found that most conflicts found during software design were dialectic or educational, and were not personal. Team members exchanged knowledge through discussions to overcome the apparent differences in carrying out the design task. One interesting finding of their case study was that 75% of time spent during the design phase involved learning about requirements, and technical and domain knowledge about the project.

Roberts, et al. (2001) studied the role of consulting practices and universities as "knowledge link" in the implementation of formal software development methodologies. The "knowledge links" transfer knowledge about software development methodologies to the adopting organizations.

2.2 Groups vs. Individuals

Although, writing computer programs is a solitary activity, development of a complete software application is mostly a team-based activity. Individuals with specialized roles collaborate to produce the application. Newer software development methodologies, such as Extreme Programming go one step further by utilizing pairs to do the development task. Limited empirical support has been put forward for the effectiveness of such pairs working together (Nosek, 1998).

Small group research is replete with many studies that compare the benefits of group vis-à-vis individuals in physical activity, problem solving, and memory recall tasks. Various theories and concepts are used to explain the apparent discrepancy in the output of groups in comparison to individuals. According to Steiner (1972), determinants of group productivity depends on task demands, resources that are brought to bear on the task, and processes involved in transforming the resources. Actual productivity is hampered by the loss due to faulty processes which could be: loss in motivation of individual members and loss due to situational constraints resulting from social interaction processes that are characterized as coordination loss.

There are other reasons that explain process gains in task performing groups. These are: assembly effects where group interaction results in newer insights for the problem at hand; social compensation effects that occur when the task is of high importance to the group and the group members compensate for the inability of another group member to perform the task; and the Kohler effect. In the Kohler effect, the less able group member steps-up his performance to match the performance of others in the group (Brodbeck and Greitemeyer, 2000). The following paragraphs review the literature on individual versus group performance for diverse tasks.

2.2.1 Group Performance

Physical Activity: Initial studies on groups vs. individual performance were focused on physical activities such as pulling a rope (Ringelmann, 1913), and cycling competition results (Triplett, 1898), both cited in (Brown 2001). Ringelmann found that group productivity is less than the individual's productivity. Triplett found improvement in the performance of individuals in the presence of another person.

Problem Solving Tasks: In problem solving tasks, in general, groups were found to be better than the average individual, but seldom better than the best individual (Hare, 1994). Potential performance of the group is higher than the actual performance in certain conditions and Steiner's concept of process losses and gains were used to explain this mismatch (Steiner, 1972). Apart from various group properties, task characteristics also influenced group performance to a great extent. Groups are better than individuals in judgment tasks as the probability of getting accurate judgment in groups is higher. Under certain conditions groups are also found to perform better than

the best individuals in solving complex tasks. Performance of groups relative to individuals is dependent on the characteristics of the task, the group processes, group size, and time availability.

Remembering: Research findings on group remembering in memory recognition tasks is that although groups on average typically do not perform as well as their "best member", they perform better than individuals on an average (Argote, 1999). When groups were compared to the best individuals, groups performed significantly better than the best individuals on recall of random items but not of organized stories. Superiority of groups on recall is due to several processes (a) groups have access to a wider pool of information than individuals, (b) groups make fewer errors than individuals, and (c) groups are better than individuals at determining what they could and could not recognize correctly (Argote, 1999).

Brainstorming: Individuals working alone produced more ideas than same number of persons working as a group (Hare, 1994). This has been attributed to "production blocking" due to group members taking turns talking and group leaders taking time to talk more.

2.2.2 Group Performance - Theories

Differences in the performance of task performing groups have been explained through various theories. Two prominent theoretical concepts in this area are social facilitation and social loafing.

Social Facilitation: This theory has been used to explain improvements in individual performance due to the presence of others. Moreover, performance

improvements were present when tasks were simple and disappeared when tasks were complex (Brown, 2001). Zajonc (1965) proposed that the presence of another member of same species increased the arousal or drive and such increased drive in the task performing person increases the likelihood of well learned responses while hampering novel or poorly learned responses. Bond and Titus (1983), in their meta-analysis of 241 studies, found that the presence of others increases the speed of simple task performance and decreases the speed of complex task performance. They also found that the presence of others impairs complex performance accuracy and slightly facilitates simple performance accuracy.

Social Loafing: Social loafing explains the decrease in group performance due to motivational loss resulting in withholding of effort by the group members. Individuals will exert less effort if the outcome responsibility is shared or when they believe their efforts are dispensable (Hare, 1994). Social loafing is widely prevalent during the performance of mundane tasks (Brown, 2001) that are additive and conjunctive. Social loafing can be eliminated by telling subjects that their individual outputs can be identified or by increasing the difficulty of the task (Hare, 1994).

Motivational gains in groups are also said to occur under certain conditions. Social compensation effect and the Kohler effect are used to explain such performance gains. Social compensation occurs when the individuals expect their partners to be less-able and the superior group member works hard to make-up for the apparent deficiencies of the partner (Brown, 2001).

Under certain conditions, the less-able member may step-up his performance to match that of the superior performer. This type of effect has been observed in physical endurance tasks when the discrepancy between the superior and the less-able member's skills are not great. This effect is called as the Kohler effect (Brown, 2001).

2.2.3 Social Cognition

Social cognition is a subset of cognitive psychology pertaining to the study of group cognition. Social cognition refers to those social processes that relate to the acquisition, storage, transmission, manipulation and use of information for the purpose of creating a group-level intellective product (Larson and Christensen, 1993). In a similar vein, researchers have studied distributed cognition that pertains to the representation of knowledge, propagation of knowledge between different individuals and artifacts, and transformation of it when operated by individuals and artifacts (Flor and Hutchins, 1991).

Social cognition differs from individual cognition because recalling a piece of information from memory is not the same as mentioning it in a group context (Larson and Christensen, 1993). In the individual level, recalling information is enough while solving a problem whereas for a task performing group, that information needs to be recalled and shared by a member of the group.

Collective cognition also refers to the cognitive process involved in groups. According to Gibson (2001), collective cognition involves four phases: accumulation – acquire knowledge and information through perceiving, filtering, and storing; interaction – group members share knowledge through retrieving, exchanging, and

structuring; examination – group members examine information through negotiating, interpreting, and evaluating; and accommodation – group members integrate, decide and act. These phases are not sequential as the groups may traverse from one phase to another iteratively (Gibson, 2001). Processes involved in collective cognition highlight the importance of sharing knowledge in task groups.

Some of the properties of the distributed cognition system are the exploration of a larger number of alternatives when there is less commonly shared information, reuse of existing system knowledge in the form of declarative memory, sharing of goals and plans that help in efficient communication and shared memory of alternative plans (Flor and Hutchins, 1991).

2.2.4 Information sharing

Information sharing in groups is an important aspect in the transfer of knowledge in a group environment. Research on information sharing revealed that group members are more likely to share ideas that members already have in common than to discuss unshared ideas that are unique to the members of the group (Wittenbaum and Staseer, 1996). Stasser and Titus (1985) argue, through their information-sampling model, that the statistical probability of an information to be mentioned during group discussion increases with the number of people possessing that information in the group.

Group size affects the sampling advantage of shared over unshared information with large groups focusing only on shared information compared to smaller groups

(Argote, 1999). Presence of experts and duration of the problem solving task were also found to increase the amount of unshared information among group members.

2.3 Group Processes

In recent times, software development utilizing teams with pairs working together on a certain task is becoming popular. The advantages of such co-operative work are many and are emphasized in the literature.

One of the prominent ways of studying teams/pairs is the use of Input-Process-Output model (McGrath 2000). Many studies on teams have utilized this model in analyzing team performance. This model traces causal paths that were hypothesized to influence group performance in solving non-eureka problems (Littlepage, et al., 1995). Limitations of the I-P-O model have been emphasized based on the difficulty in accounting for mediators in the model (Ilgen, et al., 2005).

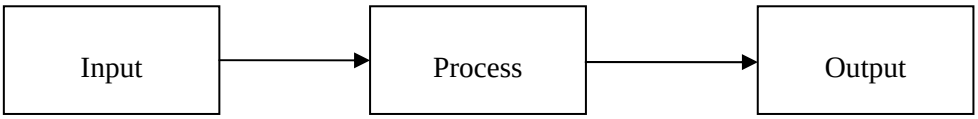


Figure 2.2 Input-process-output model

Development of software applications is a knowledge intensive process, and the industry and the research community are developing ways to manage this process. One of the ways past knowledge is utilized in software development is through the use of design patterns. This study is geared to understand these two phenomena.

In order to explain for the performance differences in the utilization of patterns while working in pairs, it would be germane to study the presence of potential process/mediator variables. Two streams of literature were tapped into towards this end. The first stream is based on the social cognitive theory that explains the relationship between knowledge and performance in individuals and in groups. The second stream is from the small group literature that pertains to the effect of communication on performance gains. Following paragraphs review these two streams of research.

2.3.1 Social Cognitive Theory

Bandura's, Social Cognitive Theory(SCT) explains psychosocial functioning in terms of triadic reciprocal causation between: personal and cognitive factors, environmental factors, and behavior (Wood and Bandura, 1989). This theory tries to explain why people sometimes would not perform optimally in spite of knowing well how to do it. SCT explains the disconnect in performance through a self-referent thought that mediates the relationship between knowledge and action (Bandura, 1986). This self-referent thought has been conceptualized as *self-efficacy* at the individual level and *collective efficacy* at the group level.

Self-efficacy: The concept of self-efficacy is based on Bandura's *Social Cognitive Theory*. According to Bandura (1986), self-efficacy is defined as "people's judgments of their capabilities to organize and execute courses of action required to attain designated types of performances". Perceived self-efficacy is also claimed to determine an individuals' effort expenditure and persistence on task-related activities.

Self-efficacy differs from self-esteem in that the latter is more of a trait measure and is concerned with the evaluation of self-worth in relation to personal standards or cultural values (Bandura, 1986).

Self-efficacy beliefs emerge based on four sources of information: past performance accomplishments, vicarious experience, verbal persuasion, and physiological and emotional arousal (Bandura, 1986). Though the experience gained through various sources is a major contributor for the development of self-efficacy, the literature highlights other sources of self-efficacy development. Gist and Mitchell (1992) outline three types of assessment processes which are involved in the formation of self-efficacy: analysis of task requirements, attributional analysis of experience, and assessment of personal and situational resources/constraints. For novel problems, efficacy beliefs are based partly on a detailed assessment of personal and situational resources and constraints (Gist and Mitchell, 1992).

In the Information Systems literature, self-efficacy has been widely used. Many of the studies utilized the self-efficacy construct pertaining to the use of computers and the effect of users' efficacy beliefs on it. Compeau and Higgins (1995) developed a construct called computer self-efficacy that was used to explain the use of computers by individuals. Research on this stream also utilized training to increase self-efficacy of the individuals (Yi and Davis, 2003). In these studies, self-efficacy was found to be a strong predictor of performance. Sharing of knowledge in the formation of knowledge repositories has also been shown to be associated with the employee's belief about his 'knowledge self-efficacy' (Kankanhalli, et al., 2005).

In the software development area, self-efficacy has been used. Hunton and Beeler (1997) found that self-efficacy beliefs of users' were related to their participation in systems development initiatives in an organization. Hertel, et al.(2003) studied programmers who had contributed to the development of the Linux operating system and found self-efficacy beliefs of individual programmers to be strongly related to their level of contribution in software development.

Two types of self-efficacy have been discussed in the literature: general self-efficacy, and task specific self-efficacy. General self-efficacy has been described as a trait variable and refers to a person's beliefs about generally performing tasks. Specific self-efficacy is domain and task specific, and it varies depending on the context studied (Stajkovic and Luthans, 1998). The majority of the studies have utilized task specific efficacy measures (Gibson, et al., 2000).

Collective Efficacy: Collective efficacy is the collective belief that the group is capable of doing a certain task. This phenomenon has been increasingly studied in the management literature in the context of task performance. Efficacy at the group level has been studied in three different ways with little variation among them: Group efficacy, Team efficacy, and Collective efficacy. Group efficacy is a broader construct and is measured in different ways whereas collective efficacy and team efficacy are task specific. Distinction between collective and team efficacy is primarily on the level of analysis. The collective efficacy construct can pertain to team, organization, country level and team efficacy beliefs which pertain to a particular team (Gully, et al., 2002).

Zaccaro, et al. (1995) define collective efficacy to represent “a sense of collective competence shared among individuals when allocating, coordinating, and integrating their resources in a successful concerted response to specific situational demands”. Hence some of the important elements in the definition of collective efficacy are: (1) shared beliefs, (2) competence in a collective’s coordination activities, (3) inclusion of other members’ resources, and (4) the situational and behavioral, or task specificity of collective efficacy.

Past literature has indicated that the antecedents for collective efficacy beliefs are to be somewhat similar to the ones in the formation of self-efficacy beliefs. Collective efficacy forms through group interaction wherein group members combine and integrate information about each other and about their task, context, process, and prior performance (Gibson, 1999; Zaccaro, et al., 1995). It involves: interactive, coordinative, and synergetic social dynamics (Fernández-Ballesteros, et al., 2002).

Group potency, a group belief is closely related to the concept of collective efficacy. Collective efficacy differs from group potency with regard to the specificity of the task. Group potency is associated with the belief about overall group effectiveness (Baker, 2001).

2.3.2 Communication

There are differing opinions about the role of communication on the group performance in problem solving tasks. According to Steiner(1972), faulty group interaction and communications contribute to process loss during task performance. Simultaneously, other researchers argue that group communication has the potential to

enhance coordination thereby reducing process losses. They claim that communication helps in the optimal combination of group efforts, discuss understanding of the task, share and combine views, and structure group processes (Tschan, 1995). This has been conceptualized as assembly bonus effects (Collins and Guetzkow, 1964). Researchers who subscribe to reductionism contend that groups will perform, at best, as well as their members and at worst, process loss will be there in groups due to group interaction (Pavitt, 2003). Propp (2003) argues that communication is central to the exploration of assembly bonus effects and the process gains in group performance of tasks. Assembly bonus effects are claimed to occur through a variety of mechanisms such as overcoming omission/commission errors, and integration of information that results in new information, raising awareness for new information, and elicitation and evaluation of the information. Hirokawa (1982a) claims that the empirical evidence of the effect of communication on group decision making and problem solving is not consistent due to diverse research methodologies adopted in such studies.

Perspectives on group communication: Hirokawa and Salazar, 1999(1999) outlines different perspectives on the role of communication on group decision making performance. Three perspectives outlined by them are: (a) mediational: communication mediates the relationship between knowledge/information resources and performance; (b) functional: interaction plays a major role in the social interaction necessary for effective performance; and (c) constitutive: communication is conceptualized as developmental that allows social construction of meaning.

Function of communication: Some of the functions put forward by group communication researchers are (Poole and Hirokawa, 1996): (a) social information processing that involves analysis and combination of information, (b) analysis of contingencies in choice making, (c) procedural maintenance during task performance, (d) establishment and monitoring of goals, (e) coordination and motivation of group members, and (f) persuasion and social influence.

Measurement of communication: Communication in a task can be analyzed with respect to the amount, content, sequence, and quality of communication (Tschan, 1995). Measurement of communication related constructs were done in a variety of ways in the literature. Some of the group interaction variables pertaining to communication, used in past research, include: communication quality (Salazar, et al., 1994), communication process (Oetzel, 2001), communication cycles (Tschan, 1995), and frequency of communication (Hirokawa, 1982b).

Communication and software development: Studies pertaining to software development teams have investigated communication processes in different ways. Many of the studies surveyed software development organizations to find how communication is related to project performance (Brodbeck, 2001; Hoegl and Gemuenden, 2001; Hoegl, et al., 2003; Sarker, et al., 2005). Sonnentag (2000) found that in software development teams, excellent developers are described by their co-workers to have good communication and cooperative capabilities. Ocker, et al.(1998) studied how various group communication technologies are helpful in gathering user requirements

for systems development and found face-to-face communication in tandem with asynchronous electronic communication to be helpful during software development.

2.3.3 Task Satisfaction

Group work is claimed to increase satisfaction of members and satisfaction has also been used as an effectiveness criterion in groups (Campion, et al., 1993). In the context of job satisfaction, Locke defines job satisfaction as, "a pleasurable or positive emotional state resulting from the appraisal of one's job or job experiences"(Locke, 1976). In a similar fashion, a member's satisfaction with the task can be defined as the positive emotional state resulting from the appraisal of one's task experiences.

Some of the factors that have an impact on task satisfaction of group members are (Shaw, et al., 2000): (a) task interdependence, (b) reward interdependence, and (c) a members' preferences for group work. Group member satisfaction is said to be more in the context of cooperative work rather than in the context of competition between members of the group (Hare, 1994).

Human beings are said to prefer working in groups due to various reasons. These include: the need for intimacy, need for power, and the need for affiliation (Forsyth, 1999). Social identity theory has also been used to explain the preference for group work (Hinsz and Nickell, 2004).

Hinsz and Nickell (2004) argues that the positive attitude associated with being a group member should be reflected in the increased satisfaction of group members while performing a task.

2.4 Software Engineering

2.4.1 Software Development

Development of software application is generally done through a sequence of steps. The six primary phases involved in a systems development life cycle are (Blum, 1994; Guindon, 1990): (a) Requirement Analysis, (b) Design and Development, (c) Testing, (d) Implementation, (e) Maintenance, and (e) Support.

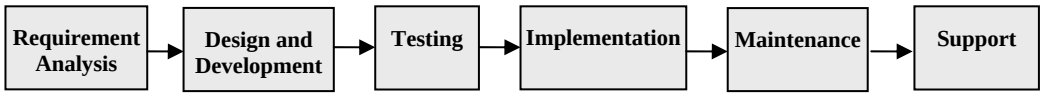


Figure 2.3 Systems Development Life Cycle (SDLC)

Figure 2.3 schematically represents the various phases involved in software development. Requirement analysis is the first phase in the development of any software application and in this phase, requirements are gathered from the end users. The next phase is the analysis and design phase where software is designed to suit the customers' requirements. The design stage is followed by the development phase where the actual software is developed. Testing of the developed software is done before it is implemented and used. In the maintenance phase, on-going customization and enhancements of the implemented software is done. Customer support is the last phase in the systems development life cycle and here the end user support is given to the users' of the developed IS application.

2.4.2 Software Design Task Characteristics

This study considers the design and development phase in software development. In this phase, detailed architectural design of the software application is

created based on the requirements of the software application. Requirements specification and design phases in software development consume more than half the development cost (Walz, et al., 1993) and moreover, errors committed during this phase are very expensive to correct (Guindon, 1990).

Problem space theory explains the complexity of design tasks. A problem space is composed of a description of the initial state, the goal state, and a set of operators to transform a state into another state. Appropriate operators are selected and applied based on the knowledge of the current state and control knowledge (Newell and Simon, 1972). There are always several alternate paths available for traversing from the initial state to the goal state. According to Simon (1973), the characteristics of ill-structured problems are: incomplete specification of the problem, no definite way to evaluate whether solution has been reached, knowledge from many sources to be integrated, and no predetermined solution path.

Software design tasks are characterized by incompletely specified requirements with no predetermined solution path. Design problems also require the integration of knowledge from multiple domains which are at varying abstraction levels (Guindon, 1990). Hence software design tasks are classified as an ill-structured problem. A data driven approach is recommended to solve ill-structured problems. Knowledge from previous experience is retrieved and inferences are made based on the new requirements (Guindon, 1990). Concepts from the problem space theory have been used to study the difference between different design techniques and their impact on the mental workload (Morris, et al., 1999).

Various tools have been developed to carryout the design tasks. Computer Aided Software Engineering (CASE) tools are widely used in this phase to create the design documents. Apart from the tools, there are techniques, such as, object-oriented analysis, that are available to formalize this process and also to produce artifacts that are consistent with general notations such as the Unified Modeling Language (UML).

2.4.3 Design Patterns

Design patterns specify standard solutions to software development problems (Guerraoui, 1996). These solutions represent both static and dynamic structures that occur repeatedly in developing applications in a particular domain (Coplien and Schmidt, 1995). Though the concept of design patterns for object-oriented programming was highlighted earlier by Coad (1992), Gamma, et al. (1995) seminal book on design patterns made these quite popular among the software development community. Use of pattern oriented design techniques have now spread to the conceptual data modeling domain (Batra, 2005).

The concept of design patterns in software engineering is derived from the works of Christopher Alexander, who promoted the use of architectural patterns in design of buildings. Gamma, et al. in their book describe design patterns as "descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context".

There are numerous design patterns available and these patterns can also be specific to a particular domain. There are formal processes through which these patterns are developed and acknowledged. Design patterns have four essential elements: pattern

name, problem description, solution description, and the consequences of using the pattern (Gamma, et al., 1995).

Patterns are claimed to capture engineering knowledge (Hagge and Lippe, 2005; May and Taylor, 2003) and communicate architectural knowledge. Experts are claimed to exercise their knowledge by finding patterns based on their experience in different contexts, and applying these to a new context (Swap, et al., 2001). Importance of design patterns in capturing and documenting practices and expertise of expert software architects are highlighted in the literature (Schmidt, 1996). Researchers had even predicted that a wealth of software design knowledge would be captured in the form of patterns and frameworks(Guerraoui, 1996).

Some of the benefits of design patterns are they: (1) provide standard vocabulary among developers, (2) document the design, (3) abstract the design problem from the details, (4) create adaptable solutions (Cline, 1996), (5) increase programmer productivity, (6) encourage the use of best practices, (7) improve communication among developers, (8) help novices to hone their design skills (Prechelt, et al., 2002), (9) give system's flexibility, extensibility, or portability (Gamma, et al., 1995), and finally (10) reduce time spent in documenting the code (Goldfedder and Rising, 1996). Some of the difficulties in using design patterns include the difficulty to learn these patterns, and the hard to learn classification schemes used to describe them (Cline, 1996).

2.4.3.1 Empirical Research on Design Patterns:

Researchers have attempted to empirically validate the claimed benefits of design patterns. Studies carried out in a maintenance environment by Prechelt et al. (2002) found design patterns to improve the maintainability of programs when pattern comment lines were used. In another study, they also found that at times the use of design patterns made a program difficult to maintain and provided unnecessary flexibility (Prechelt, et al., 2001). Purao, et al.(2003) developed a semi-automated system that used a reuse-based design approach that implemented learning through analogical reasoning. They developed and tested a design assistant system and found that system with learning capability outperformed the ordinary approach without learning.

2.4.3.2 Use of Design Patterns:

Using patterns requires (1) domain independent patterns to be interpreted in order to find their applicability, (2) adjustment of the selected patterns to suit the problem domain, and (3) integration of the selected pattern with other design elements (Purao, et al., 2003). Knowing design patterns is not enough to use the patterns; and the user should be able to intelligently apply it to the problem (Kerievsky, 2005). Goldfedder and Rising (1996), in their article on design patterns training highlight the importance of explaining design patterns in the context of users' domain. They quote from the authors' personal communication with Ralph Johnson's "… people can't learn patterns without trying them out. Also, people need to find them in their own problem domain"

2.4.4 Software Reuse

Use of design patterns could also be conceptualized as the reuse of an Information System design artifact. Software reuse refers to the use of existing software artifacts such as code, design frameworks, etc. Software reuse has been vigorously pursued in the industry due to the resultant productivity and quality gains in using existing software artifacts (Prieto-Diaz, 1993). Extensive research has been done on software reuse, and there are few review articles in this field (Krueger, 1992; Mili, et al., 1995). Tools such as Computer Aided Software Engineering and technologies such as Object-oriented programming languages have facilitated reuse in software development. IS research on reuse has also mirrored this with many studies proposing and evaluating new tools that aid in the reuse of software artifacts.

Software reuse has been characterized by three stages: retrieval, adaptation, and integration (Prieto-Diaz, 1993). Two common types of reuse have been discussed in the literature: black box reuse and white box reuse. Black box reuse refers to the reuse of software components without any modifications, whereas white box reuse refers to the reuse of components after modification and adaptation (Prieto-Diaz, 1993). Sen (1997) classified software design reuse into the demand-side and supply-side reuse.

There are many studies in software reuse and they are oriented towards programming code/module reuse, there are few specific studies that are available on design artifact reuse. Existing literature on reuse in software engineering has focused on technical and organizational factors, while ignoring cognitive characteristics of

individual developers (Parsons and Saunders, 2004). Some of the studies in reuse that studied cognitive aspects include (Parsons and Saunders, 2004; Sen, 1997).

Sen (1997) studied the role of opportunism in software design reuse and concluded that designers indeed use opportunism in software reuse. He concluded that sequence of steps for reusing a design artifact cannot be decided a priori. Parsons and Saunders (2004) analyzed anchoring and adjustment for design artifact reuse. Anchoring refers to the tendency of problem solver to make final estimates closer to the initial estimate for a solvable problem without performing adjustment. In the reuse scenario, they contend that developers tend to use the extraneous functionality of the reuse artifact even when it is not needed in the implementation.

Behavioral aspects in software reuse have been studied more extensively. Rothenberger (2003) studied organizational factors that aid reuse. Some of the factors considered in the study include: client influence, project culture, project attributes, and developer reuse experience. Morisio (2002) studied the factors that aid in the success or failure of software reuse. They found human factors, management's commitment to introduce the reuse process, and existence of commonality between applications to be critical.

Reuse of object-oriented analysis models have also been studied in the literature (Irwin, 2002). Reuse of such models did not have any effect on the quality of the object-oriented analysis models. Similarity of the source and target problems did affect the reuse of the artifacts.

Use of design patterns will require adaptation of domain neutral patterns to the target domain. Based on the knowledge of software reuse, use of design patterns can be considered to be a white box reuse focusing on the demand-side of the reuse process. Both cognitive and behavioral aspects influence the success of reuse of design components.

2.4.5 Extreme Programming/Pair-Programming

Extreme Programming (XP) is one of the Agile Software Development approaches that promise to make software development projects more successful. XP focuses on small teams of 3 to 10 programmers and requires the participation of one or more customers as members of the development team for providing ongoing expertise. XP methodology has pair programming as one of its core requirements. Pair programming involves two programmers working side by side at one computer, collaborating on the designing, coding and testing of the software application.

The performance of pairs is expected to be of much higher quality. Pair programming encourages each programmer to drive the other partner a little harder to excel. There are also some anecdotal evidence to indicate that collaboration improved both the performance and enjoyment of the whole problem solving process for the programmers (Nosek, 1998). Past research has shown that when two programmers work together, they work twice as fast and generate twice as many solutions to a problem as two working alone, while achieving higher defect prevention and removal, and thus leading to a higher quality product (Williams and Kessler, 2000). Parrish, et al.(2004) did a study based on data from 48 programmers who worked at various levels of

collaboration and found that productivity did not improve as a result of programming concurrently. Since their data was derived from projects that did not directly utilize pair-programming, they speculated that XP's formalized pair-programming environment may increase productivity.

Empirical research on pair working in software development has primarily concentrated on the programming aspect of software development. Hitherto only one study has been done which addresses the effectiveness of pair-designers vis-à-vis individuals (Al-Kilidar, et al., 2005). Of the two types of tasks they studied, they found pairs to be more effective in simple design tasks, but did not find differences between pairs and individuals in the complex design task. One major limitation they highlighted was the likely presence of learning effects as they used same subjects once in pair condition and again in solo condition.

2.5 Literature Review – Implications for the Current Study

Based on the review of the literature, the following statements succinctly capture the salient points pertinent to the current study. Figure 2.4 graphically represents the areas that were reviewed in this study.

- Software development is a knowledge intensive process involving integration as well as transfer of various knowledge elements.
- Codification of design knowledge through design patterns and the subsequent transfer of it are not always easy.

    - Transfer of knowledge is greatly dependent on the problem solvers' perception of similarity between the source and target problems.
    - Reuse of design artifacts which can be conceptualized as a transfer involves both cognitive as well as behavioral issues.
- Knowledge Transfer
    - Types of knowledge
    - Different knowledge transfer processes
- Pairs working together in software design and development tasks are becoming increasingly popular.
    - Pairs are not always better than the best individual in problem solving tasks.
    - Systems design is an intellective problem solving task.
    - Groups share common knowledge the most and sharing of unique knowledge is hampered by various group dynamics.
    - Social cognition literature highlights the benefits of group interaction and its impact on cognitive processes.
    - Group processes play a major role in the effective functioning of groups.

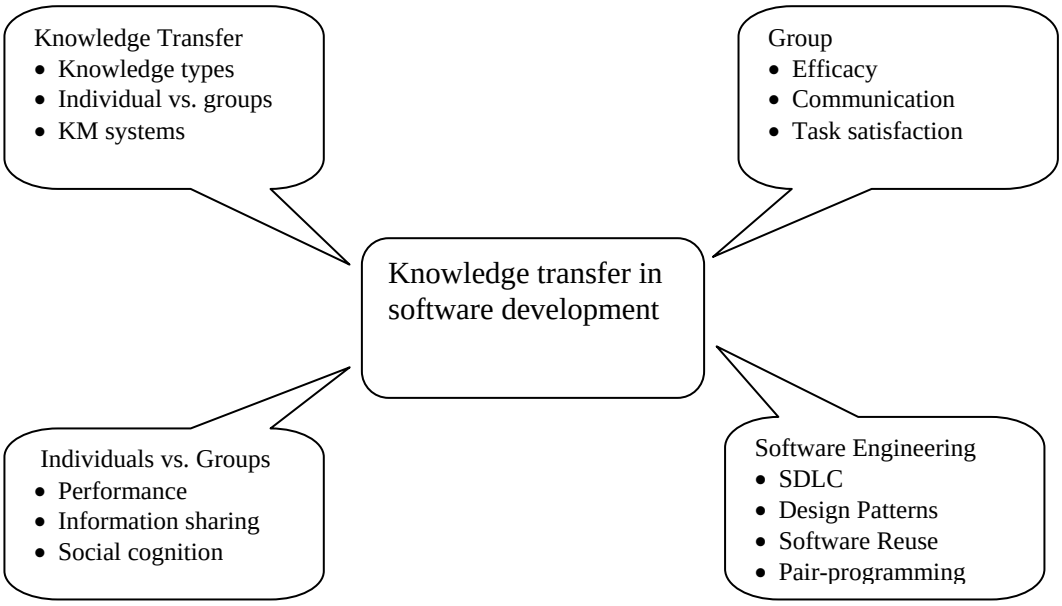


Figure 2.4 Summary of research streams reviewed

Based on review of literature, the next chapter develops the various hypotheses that are planned to be analyzed in this study.

CHAPTER 3

RESEARCH MODEL AND HYPOTHESIS DEVELOPMENT

This study investigates the effect of collaborating pairs on the performance of design tasks. It also analyses the role of design patterns, a codified form of knowledge, in task performance. To this end, hypotheses were developed based on the past literature.

This chapter presents the hypotheses that were proposed in this study in the following fashion. The first part of this chapter presents the hypotheses pertaining to the individual and collaborating pair comparison in terms of solution quality, time taken, subjective mental workload and overall task satisfaction. The subsequent part of this chapter presents the hypotheses pertaining to the individual condition. The last part of this chapter presents hypotheses that are applicable to the collaborating pair condition.

The primary research questions addressed by this study are:

1. *Are there any differences in the design task performance outcomes between an individual working alone and a collaborating pair working together?*
2. *Are there any differences in the design task performance outcomes based on the availability of codified knowledge?*

3. *Are there any differences in the time taken to solve the design problems between individuals and the collaborating pairs?*
4. *Are there any differences in the overall task satisfaction between an individuals working alone and a collaborating pairs working together?*

Apart from addressing these primary research questions, this study also examines the role of design self-efficacy and communication on task performance.

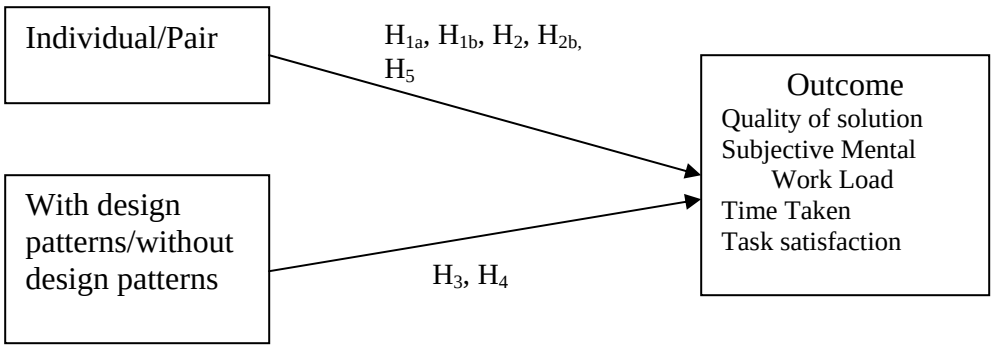


Figure 3.1 Research model

3.1 Pairs versus Individuals

Research in the comparison of group and individual performance has been enduring with Shaw's work appearing in the early 1930s. Since then, numerous studies have appeared in this area. According to Hill (1982), in the past, research on group versus individual performance comparisons was done in four different ways: (a) group versus individual direct comparison, (b) group versus the most competent member of a statistical aggregate, (c) group versus statistically pooled responses, and (d) group versus math models.

3.1.1 Pairs and Nominal Pairs – Solution Quality

In the group versus individual direct comparison, collaborating groups perform better than independent individuals on a wide range of tasks (Laughlin, et al., 2006). These tasks varied from learning/concept attainment, concept mastery/creativity, and abstract/complex problem solving (Hill, 1982). Here the comparison is done between an equal number of collaborating groups and individuals, for example, 20 multiple member groups with 20 individuals.

Shaw (1932) found groups to be superior to individuals in solving complex problems. Subsequently, Marquart (1955) performed similar experiments, but this time she utilized statistical aggregates obtained by combining individual performances and co-acting groups. Marquart's comparison of groups versus statistical aggregates showed no statistical differences in the performance of groups vs. individuals. The discrepant findings were explained by the increased probability of the presence of a competent individual in the groups who could solve the problem. Hence groups benefited from the aggregation of members which increases the probability of having a competent individual to solve the problem (Hill, 1982).

Groups can also be compared to the most competent member of a statistical aggregate. According to Laughlin et al. (2006), a more stringent way to test groups versus individual performance would be comparing 'n' groups of size 'm' with equivalent number of 'n x m' individuals. This type of comparison can use nominal groups and allow us to compare groups with 1^{st} best, 2^{nd} best, and so on.

Superior performance of groups relative to individuals is dependent on the characteristics of the task, group processes, group size, and time availability. According to Davis and Harless (1996), the sensitivity of relative group performance to task type and conditions makes it important to continue research in group performance in order to understand social processing of information.

Task characteristics: Characteristics of the task play a major role in the apparent discrepancies between group and individual performance. In complex problems, groups performed better than the average individuals due to the pooling of information and the correction of errors by each other which results in assembly bonus effects (Hill, 1982). In simple tasks, the superiority of groups was primarily attributed to the groups' increased probability of getting a competent person.

Tasks can be classified using various typologies (McGrath, 1984; Steiner, 1972). One of Steiner's classifications of tasks is conjunctive and disjunctive tasks. In disjunctive tasks, a single competent person in the group can lead the group to a correct solution whereas in a conjunctive task, the whole group should perform the task and know the correct answer to solve the problem. Working in pairs has an advantage over, individuals because there is an increased chance that one member of the pair will have the correct solution (Steiner, 1972). Brodbeck and Greitemeyer (2000), in their dynamic model of group performance, highlight that group members working collaboratively learn to correct each others' error.

Laughlin and Ellis (1986) classifies tasks into a continuum that ranges from intellective to judgmental. An intellective task is a problem solving task that has a

correct solution and high result demonstrability. A judgmental task on the other hand is an evaluative task that does not have a demonstrably correct solution.

According to Laughlin, et al. (2002), groups perform at the level of the best individual (independent or best group member) on highly demonstrable problems. Groups perform at the level of the second-best individual (independent or best group member) on less demonstrable vocabulary and analogy problems.

Superiority of groups in problem solving tasks is not consistently found in the literature. Small groups, typically do a better job of solving unitary, optimizing tasks, such as solving a mathematical problem, than the average of individuals in nominal groups of comparable size. Groups however suffer in performance relative to the best-performing member of nominal groups (Davis and Harless, 1996).

Group Processes: Group processes also determine the performances of groups in comparison to individuals working alone. For instance, in groups performing brainstorming, the performance of real groups suffers in comparison to statisticized groups in both quantity and quality of ideas generated (Diehl and Stroebe, 1987). The possible reasons attributed to this include: production blocking, evaluation apprehension, and free riding.

Groups resolve disagreement among their members in proposing a collective response through various social combination processes. Laughlin, et al. (1991) in their experiments found groups to perform at the level of the second best individual for recognizing the correct hypotheses (truth), and the level of best individuals for nonplausible hypotheses (rejection of error).

Group size: Size of the group also was found to be factor in comparing the performance of groups versus individuals on problem solving tasks. Laughlin, et al.(2006) found that groups with at least 3 members are needed to achieve superior group performance compared to the best individual performance on highly intellective problems. They also found groups' performance to improve significantly from two member to three member groups, and the improvements to decline as the group size increased from 3 to 4, and 4 to 5.

Available Time: Time that is available to perform a task is also found to influence group performance. Laughlin, et al. (1991) found that groups performed at the level of best individuals when the time allowed was more for information rich rule induction problems.

Software design problems are called as wicked problems (Poppendieck, 2002). Software design problems usually have multiple correct solutions. For example, there may be many alternate feasible solutions for a given problem. Because of this a design task cannot be called as a purely intellective task with a single solution. Groups have been found to perform better in intellective task. The size of groups used in this study is two and according to Laughlin (2006), group size of at least three is needed for the group to perform better than the best individual. Moreover, the time allowed is the same for the subjects in the individual as well as in the pair condition. Groups are found to perform better than the best individual when the available time is more. Based on the above arguments the following hypotheses were developed.

H_{1a}: Performance in terms of solution quality of the best individual in a nominal pair will be better than the collaborating pair's performance.

H_{1b}: Performance in terms of solution quality of collaborating pairs will be better than the second best individual in nominal pairs.

3.1.2 Pairs and Nominal Pairs – Time Taken

This study uses time taken to solve the problem as one of the measures of performance. In the past research on pair programming, Nosek (1998) found that individuals took more time than a collaborating pair in a programming task, but they were not statistically different from each other. Pairs were found to take less time to complete tasks than individuals. However, since two people were involved, their total time for a task in terms of man-hours was more. This has often been highlighted in the literature on pair-programming (Williams, et al., 2000; Williams and Kessler, 2000).

However, small group literature indicates that groups take longer to solve problems that are not amenable to division of labor (Hare, 1976). Software design tasks are inherently indivisible hence division of labor is not possible. Some of the reasons attributed for the longer time taken by groups are: the time involved in error-checking (Hare, 1976), the time involved in communication (Hill, 1982), the time required to become familiar with each other (Hill, 1982), low motivation, and personality conflict (Hare, 1976).

Based on the above arguments, the following hypothesis is arrived at.

H_2: Collaborating pairs will take longer time to complete the design task than the nominal pairs.

3.2 Impact of Codified Knowledge on Task Performance

Development of software application is a knowledge intensive endeavor. Knowledge of the software application domain, programming languages, tools and technologies are needed to be integrated during the development of a software application. Knowledge that is gained through past experience in software development has to be transferred and applied to the new problem situation. Research on knowledge management calls this as transfer of knowledge. Markus (2001) claims that the reuse of knowledge is clearly related to organizational effectiveness.

3.2.1 Availability of Codified Knowledge and Solution Quality

Software developers reuse designs that have worked well in the past and their repertoire of design experience grows with their personal experience (Beck, et al., 1996). Design patterns formalize the reuse of past design experience by explicitly codifying past design knowledge and making it available to both experts and novice designers. Various benefits of design patterns are given in the literature. These include: increase in programmer productivity, creation of adaptable solutions, use of best practices, and development of flexible and extensible systems.

In the knowledge management literature, knowledge transfer is said to occur through various mechanisms such as internalization and socialization (Nonaka, 1994). In socialization, individuals share their experience with others. In internalization, individuals apply explicit knowledge and transform it to tacit knowledge. In the pair condition, socialization is present and in pairs with design patterns conditions both socialization and internalization are present. Hence, pattern condition provides an

additional source of knowledge which the subjects can utilize through the process of internalization.

Distributed cognitive systems literature informs us that the cognitive properties of such a system are influenced by structures within the individual as well as the artifacts outside the individuals (Flor and Hutchins, 1991). In distributed cognition parlance, artifacts such as design patterns can be equated to 'external structured representational media'. Hutchins (1995) argues that the use of such artifacts in groups can transform a seemingly complex task into a simple perceptual task.

Problem solving involves the application of declarative knowledge which is used to move from the initial representation of the problem to appropriate goal state by applying correct operators. There are four general ways of enhancing the search for correct operators during human problem solving (Newell and Simon, 1972): (a) applying algorithms, (b) applying heuristics, (c) following creative techniques, and (d) increasing the availability of declarative knowledge. Batra (2005) argues that during problem solving, the presence of patterns enhances the search for correct operators by employing heuristics, and stimulates existing operators in new ways. Guindon (1990), argues that the cognitive processes involved in software design calls for the recognition of partial solutions at various abstraction levels that were based on previous experience. Design patterns provide partial solutions to the design problem at hand and this should benefit the system designers.

According to Laughlin and Ellis (1986), intellective task is a problem solving task that has a correct solution and demonstrability of result is high. Conditions for

result demonstrability are (Laughlin and Ellis, 1986): (1) Group consensus on the problem solving system, i.e. vocabulary, syntax, terms, axioms, and relationships; (2) sufficient information within the system; (3) group members should have sufficient knowledge of the system to recognize and accept the solution proposed by another member; and (4) enough motivation should be there for the member who knows the correct solution to demonstrate that solution to others.

The availability of patterns helps in improving the result demonstrability. First of all, patterns give a shared vocabulary to the problem solving group members (Freeman, et al., 2004) that creates group consensus on the problem solving system. Second of all, presence of patterns increases the availability of information within the problem solving system. Moreover, groups with the design pattern knowledge can infer an appropriate solution based on the available design patterns and hence increase the result demonstrability of the solution.

Research in group information sharing highlights that when the result demonstrability of the solution increases, two things happen: groups are likely to discuss more extensively, and share unique or unshared information (Stasser and Stewart, 1992).

There are very few studies that have empirically analyzed the role of design patterns on the quality of design solutions. Prechelt et al. (2002) carried out an experiment and found pattern comment lines to help in the maintainability of programs in a maintenance environment. In a separate study they also found that, at times, the use of design patterns made a program difficult to maintain and resulted in unnecessary

flexibility of systems (Prechelt, et al., 2001). There is a dearth of studies that explicitly study the role of design patterns on solution quality.

Some of the benefits of design patterns are they: (1) create adaptable solutions (Cline, 1996), (2) increase programmer productivity, (3) improve communication among developers, and (4) help novices to hone their design skills (Prechelt, et al., 2002),. Based on the claimed benefits of design patterns, this study argues that availability of codified knowledge in the form of design patterns will improve the solution quality of design problems.

H_3: Performance in terms of solution quality will be higher in the design pattern condition than in the condition without design pattern.

3.3 Mental Work Load

Concepts from the human problem solving domain are widely used in analysis, design, and programming aspects of software application development. Availability of design patterns during the course of solving design problems can also be studied with the perspective of human problem solving pertaining to changes in the cognitive processing requirements. Cognitive processing requirements for a task are based on the degree to which the problem representation is cognitively easy to access, modify, and integrate within the new problem (Morris, et al., 1999). Human beings are considered as information processors and the same concept is applied to groups as well (Hinsz, et al., 1997).

Codified knowledge helps in formulating solutions that require minimal cognitive strain. Gibson (2001) highlights how a recalled script leads to a behavior with

minimal cognitive strain, due to the prior knowledge and feedback substituting for an explicit and detailed analysis of a complex task.

In the context of human problem solving, pattern recognition reduces the effort of processing facts individually and speeds up understanding or the generalization of insight (Reeves, 1996). Research on distributed cognition, and stimulating structures inform us that the external artifacts such as the design patterns in this case will reduce the cognitive load of the problem solver (Hayne, et al., 2003; Hutchins, 1995). Guindon (1990) argues that the resolution of ill-structured problems involves the use of a data driven approach that requires little cognitive cost as opposed to goal-directed behavior. A data-driven approach relies more on past experience. Availability of design patterns should help developers utilize more of a data-driven approach to solve the problem as compared to a goal-directed approach.

H_4: Subjective mental work load will be lower in the design pattern condition than in the condition without design pattern.

3.4 Task Satisfaction

Shaw argues that higher task interdependence will lead to higher task satisfaction (Shaw, et al., 2000). Software design task done jointly by the collaborating pairs would be a highly interdependent task and this should lead to better task satisfaction.

Hinsz and Nickell (2004), in their study, found that groups are more satisfied than individuals in the performance of goal setting tasks even though the performance levels were not statistically different between the groups and individuals. In

brainstorming tasks, collaborating groups are found to be more satisfied than individuals even though the quantity and quality of ideas were inferior to individuals (Nijstad, et al., 2006).

Past research in pair-programming, has also shown that pairs working on programming tasks exhibited higher task satisfaction than individuals (Cockburn and Williams, 2001; Nosek, 1998).

Based on the argument above the following hypothesis was arrived at.

H_5: Task satisfaction will be higher in the pair condition than in the individual condition.

3.5 Proposed Model – Individual Condition

Two separate models for design task performance are proposed to explain the differences in the performances of individuals and pairs.

3.5.1Patterns and Design Self-efficacy

This study argues that availability of patterns, an additional resource which captures the experience of seasoned software developers, will help in the increase of a person's perceived self-efficacy. Patterns are claimed to provide proven solutions to recurring design problems. These patterns provide codified form of knowledge to the developers.

Self-efficacy beliefs are formed based on the knowledge/abilities of the individuals. The individuals, in the pattern condition will have both tacit knowledge and the codified knowledge. Tacit knowledge comes from their experience in the field and the codified knowledge is provided by the design patterns.

According to Gist and Mitchell (1992), individuals use both internal and external cues to form self-efficacy beliefs. Some of the highlighted external cues are the availability of resources to perform a task, task complexity, and interdependence of the task with others. Moreover, if the situation in which the task is performed is new then detailed assessment of self-efficacy will be done by the individuals (Gist and Mitchell, 1992). Because of this, individuals in the pattern condition could consider the availability of patterns while formulating their efficacy beliefs.

This study utilized design self-efficacy, which is a domain specific construct that pertains to the performance of software engineering design. Researchers have indicated the need for the domain specific self-efficacy constructs that provide more robust results than general measures of self-efficacy (Salanova, et al., 2003)

H_{i1}: Individuals in the pattern condition will exhibit significantly higher design self-efficacy than individuals in the no pattern condition.

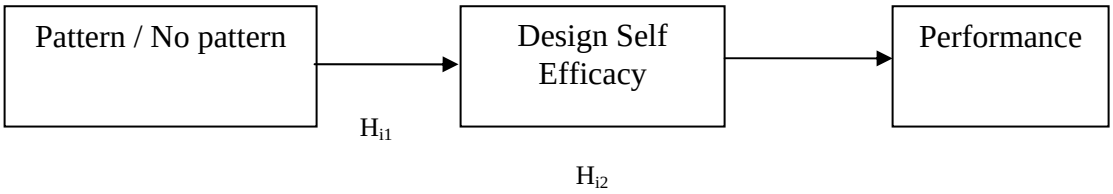


Figure 3.2 Individual condition mediator model

3.5.2 Mediating Role of Self-efficacy

Perception about efficacy has always been related to the improvements in the task performance. Individuals with high self-efficacy are willing to exert more effort and they are more persistent in overcoming obstacles while solving problems (Bandura,

1986) and consequently results in better task performance. They are also more willing to work toward a difficult goal.

Numerous studies have analyzed the relationship between self-efficacy and task performance. For example, meta-analytic studies have quantitatively analyzed this relationship under various settings (Judge and Bono, 2001; Stajkovic and Luthans, 1998). Both task specific self-efficacy and generalized self-efficacy are found to impact task performance. Stajkovic and Luthans (1998) found self-efficacy to be a strong predictor of task performance under varying task complexities. They also found this relationship to be strongest under a low task complexity condition.

Many times, self-efficacy was claimed to have mediated the relationship between task knowledge, experience, and task performance. Bandura (1986) conceptualized self-efficacy to be the mediator between knowledge and task performance. Self-efficacy explains how individuals make judgments about their capabilities and how their self-percepts of efficacy affect their motivation and behavior. Based on these arguments, the following hypothesis was formulated.

H_{i2}: Perceptions of design self-efficacy will mediate the positive relationship between pattern availability and design task performance.

3.6 Proposed Model - Pair-condition

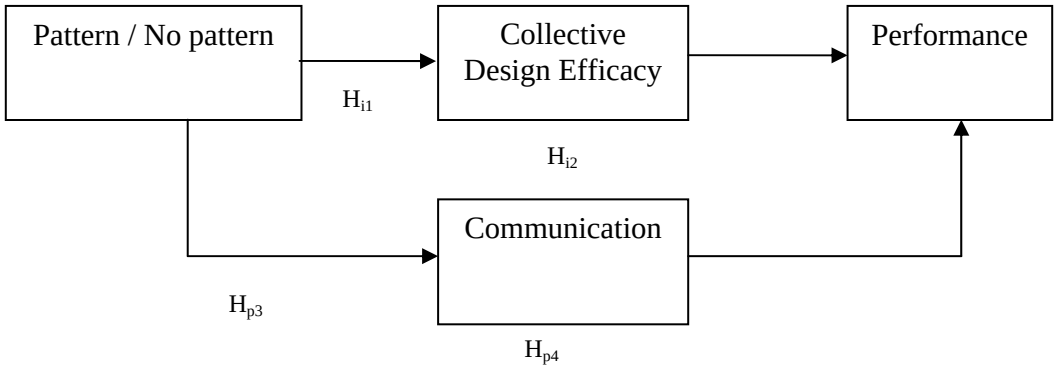


Figure 3.3 Pair condition mediator model

3.6.1 Patterns and Collective efficacy

Antecedents in the formation of collective efficacy are also claimed to be similar to the ones in the formation of self-efficacy beliefs. Taggar and Seijts (2003) outlined how the formation of collective efficacy beliefs utilize the availability of additional resources. Individuals, in the pattern condition have additional resource in the form of codified knowledge. Availability of expert knowledge in the form of design patterns could significantly increase the efficacy beliefs of individuals. Arguments that were put forward in an earlier section on design patterns and self-efficacy also hold good for the relationship between the availability of design patterns and collective efficacy.

H_{p1}: Pairs in the pattern condition will exhibit significantly higher collective efficacy than pairs in the no pattern condition.

3.6.2 Mediating Role of Design Collective Efficacy

Like self-efficacy, which has been found to impact task performance, collective efficacy has also been found to be a predictor of task performance. When the members feel more confident about their work in a certain domain, they will be more motivated to work and persist in the face of difficulties (Zaccaro, et al., 1995). Collective efficacy has been found to positively influence group performance in variety of settings (Gully, et al., 2002; Pescosolido, 2001).

Gully, et al.(2002) performed a meta-analysis on the studies in collective efficacy, team potency and team performance. Their findings suggest that both collective efficacy and team potency were strongly related to team performance. They also found that collective efficacy is task specific, whereas team potency is task general, and the results mirrored this distinction.

H_{p2}: Perceptions of design collective efficacy will mediate the positive relationship between pattern availability and design task performance.

3.6.3 Patterns and Communication

Availability of design patterns can help the group to focus on what needs to be communicated during group problem solving. The role of referents which affect communication in performing a task has been studied in the literature. Although the "referential communication task" is done in a controlled setting, the referents simulate the process of reference that occurs in natural settings (Krauss and Fussell, 1990). Referential communication restrains the participants from straying away from the topical domain that is defined by the task. Hence, design patterns can be argued to

improve communication effectiveness by orienting the subjects to solve design problems while utilizing the design patterns.

One of the serious problems in the effectiveness of interpersonal communication is differences in the meaning of words attributed by the participants (Phillips, 1966). Patterns provide a standard shared vocabulary for the designers (Cline, 1996; Freeman, et al., 2004). Shared vocabulary must help in reducing the ambiguity in communication. Rising (1999) states that the expertise captured by design patterns, with a well defined vocabulary of pattern names and standard solutions, help improve communication during software development. Schmidt, et al. (1996) notes that software projects often fail because of the inability of developers to communicate good designs. Design pattern descriptions communicate designs by articulating the structure and behavior of solutions.

Based on the above arguments, the following hypothesis is presented.

H_{p3}: Pairs in the pattern condition will exhibit significantly higher communication quality than pairs in the no pattern condition.

3.6.4 Mediating Role of Communication

Researchers have divergent views on the role of communication on task performance. Assembly bonus effects, as outlined, are present when there is effective communication (Propp, 2003). Many of the group effectiveness models explicitly linked communication, coordination, and cooperation in relation to group performance; a compilation of such models can be found in Yeatts and Hyten (1998). Various theoretical perspectives in communication research: mediational, functional, and

constitutive approach links communication to task performance through one mechanism or another.

Transfer of tacit knowledge between individuals is claimed to occur through the process of socialization (Nonaka, 1994). Proper communication between the pairs will greatly help in mutual sharing of knowledge thereby creating a socializing environment. This study argues that the communication quality is a mediator between design pattern availability and task performance.

H_{p4}: Perceptions of communication quality will mediate the positive relationship between pattern availability and design task performance.

The next chapter presents the research method adopted to test the various hypotheses outlined in this study.

CHAPTER 4

RESEARCH METHODOLOGY

This study utilized a laboratory experiment with practitioners as subjects to test the various hypotheses formulated in the previous chapter. Laboratory experiments provide sufficient control to study the real effects of the variables under consideration. Using practitioners as subjects provided realism to the study and improved the generalizability of the outcomes.

This study utilized a completely randomized design with two treatments having two levels:

(a) Individual/collaborating pair condition

(b) With design patterns/without design patterns.

Subjects belonging to all four treatments worked on two design problems. The first one was the warm-up task, and the second one was the main design task. The following variables were measured and studied.

Dependent variables: Quality of the solution, time taken, subjective mental work load, and task satisfaction.

Mediating variables: Design self-efficacy/collective efficacy, and communication quality.

Demographics, as well as, potential covariates such as gender, years of software development experience, and experience in object-oriented programming and design were collected and utilized in the data analysis.

4.1 Subjects

The target participants for this study were software development professionals who had not used design patterns in their workplace and were willing to volunteer time for the study. As an incentive, participants in the study were offered a free seminar on Object-oriented design with design patterns offered by Dr. Nerur, an experienced IS faculty member. Flyers were distributed to members of various organizations in the Dallas/Fort Worth Metroplex. A copy of the flyer is attached as Appendix A. Interested participants were asked to register at a website. A total of 177 volunteers registered and expressed their willingness to participate in the study. E-mails were sent to the registrants with the dates and times of the experimental and seminar sessions.

Since the study involved human subjects, appropriate approval was obtained from the University's Office of Research Compliance.

4.2 Experimental Setting

Experiments were conducted in the university's behavioral lab. This lab has 9 closed cubicles. Each cubicle has a table with seating for two. Experimental subjects were provided with scratch papers, and pencils for carrying out the problem solving task. Stop clocks were also placed in each cubicle during the experimental session. They served two purposes: to measure the time taken for task completion and to keep the subjects informed of their progress. All subjects were provided with a glossary of

Unified Modeling Language class diagramming notations that they could use in their solution (see appendix H). Subjects were also briefed on these notations. Subjects drew the class UML diagrams on sheet of paper. Subjects in the pair condition were asked to work collaboratively and arrive at a single solution that is agreeable to both the individuals in the pair. Subjects were also presented with the inform consent for their participation (see appendix B). All the four treatment conditions had their own task instructions and these instructions are presented in appendix D.

4.3 Planned Sample Size

The planned sample size for each of the four experimental conditions was 24. This study utilized hypothesis with nominal groups which called for equal number of subjects in the individual and the pair condition. Each of the pair condition had 24 subjects resulting in 12 collaborating pairs. In the two individual conditions, there were 24 subjects each, yielding 12 nominal pairs for the statistical analysis. The total sample size for the experiment was expected to be 96. Eight experimental sessions, spread over a three-month period, were planned and conducted. The maximum participation in an experimental session was 18 subjects.

4.4 Research Design

Subjects were randomly assigned to various treatment conditions. From the pool of potential participants, e-mails were sent to select individuals asking for their participation in a particular day's session. If the participants confirm their participation, subsequent e-mails were sent with detailed instruction about the time and the venue for the experimental sessions. Two experimental sessions were held on the same day. The

first experimental session did not provide the design patterns and the second experimental session had the pattern condition. After the subjects arrived for a particular experimental session, they were randomly assigned to the individual or to the collaborating pair condition.

In the pattern condition, four patterns were provided to the participants in the form of design documents and these patterns were useful in solving the warm-up and the main design tasks.

	With patterns	Without patterns
Individual	Condition I	Condition II
Pair	Condition III	Condition IV

Figure 4.1 Research design – treatment conditions

4.5 Experimental Task

There were two experimental tasks that the subjects performed. Both tasks required the creation of Unified Modeling Language (UML) class diagrams for the problems. These tasks consisted of a warm-up and a main experimental task. The warm-up task lasted 20 minutes and the experimental task had duration of 80 minutes to complete. The warm-up task was based on the problem statement outlined in Freeman, et al.(2004). The main design task was based on the problem statement found in Richter (2004). Appendix E has both the problem statements. Subjects in the pattern condition

were provided with documentation for the necessary design patterns. Appendix I shows the materials on design patterns given to the subjects in the design patterns condition.

Subjects were asked to complete the questionnaire on efficacy beliefs before they started working on the design task. Measurements of the remaining variables were done after the completion of the task. The whole experimental session lasted for approximately two hours.

4.6 Pilot Test

A walk through and a pilot test were done to fine-tune the experimental procedures before the start of the actual experimental sessions. The walk through tested the manipulations and experimental materials in all treatment conditions. The purpose of the walkthrough was to assess all aspects of the experiment such as the task instructions, the effectiveness of manipulations, and appropriateness of the tasks.

A pilot test of the experiment was conducted prior to the main experimental sessions. For the pilot test, eleven students from the Advanced Analysis and Design course, and 4 from local software development organizations participated. A total of 15 subjects participated in the pilot study. All the treatment conditions had at least two subjects. Based on the feedback, as well as the observation of the experimenter, the required changes were made in the scripts as well as other materials used for the experiment. Some of the items were changed based on the insights gained from the pilot study. The time required to perform the task was also found to be adequate. On an average, 49.31 minutes were used by the subjects to finish the task.

4.7 Manipulation Checks

Two primary manipulations were done in this experiment. These pertain to the availability of design patterns and the mode of task performance. A manipulation check was done to ensure that the pattern condition group utilized patterns in solving the design problems. Before the task was performed the experimenter asked the subjects in the pattern condition to utilize the patterns. After the completion of the experimental task, subjects in the pattern condition were asked to describe in writing the steps they followed in solving the problem and the design patterns they used.

Subjects were randomly assigned either to the individual condition or to the collaborating pair condition. In the collaborating pair condition, subjects were asked to come-up with a single solution that was agreeable to both the participants. Moreover, after the completion of the task, subjects were asked about their collaborators level of contribution in the task performance.

4.8 Measurement of Variables

All variables, with the exception of solution quality and time, were measured using scales developed in the extant literature. Table 4.1 summarizes these scales. Task performance was operationalized using two measures: (1) quality of solution, and (2) time taken

4.8.1 Solution Quality

Past literature highlights the difficulty in measuring the quality of conceptual solutions. Rehder, et al.(1997) outlined an unbiased way to compare software designs that were produced using divergent languages, and methodologies. Their method

basically decomposed a design into a large number of atomic design "features" that are not biased towards one language, paradigm, or methodology. Their scoring system also allowed design alternatives and optional features.

The solution of an expert designer was utilized to decompose the problem into atomic design features. Purao, et al.(2003) utilized a coding scheme that took note of Type I – omission error and Type II – commission error. Completeness of the solution is captured by omission errors and correctness of the solution is captured by commission errors. Solutions developed by subjects were evaluated based on these guidelines. Appendix G presents the scoring scheme used to grade these solutions. The quality of solution was be evaluated by two Ph.D. students who were blind to various hypotheses in the study.

4.8.2 Completion Time

In the past, empirical studies involving design patterns used time taken to complete a design task as a dependent variable (Prechelt, et al., 2001). In line with this, the time taken to solve the main design task was used as one of the dependent variables. Stop clocks were used to measure time.

4.8.3 Design Self-efficacy

The scale for design self-efficacy was developed based on the existing self-efficacy measures. Bandura (1986) states that the time elapsed between the assessments of self-efficacy and action is an important factor in affecting the degree of relationship. In light of this, design self-efficacy was measured before the subjects worked on the design task. There are diverse ways of measuring the self-efficacy of an individual. Past

literature indicates five distinct ways to measure this construct (Lee and Bobko, 1994). Measurement scale used in this study for design self-efficacy was formulated based on the self-efficacy scales that have been used in the past (Jones, 1986; Kankanhalli, et al., 2005). In total six items were formulated and a seven point likert scale was used to measure this construct.

4.8.4 Design Collective-efficacy

Collective efficacy differs from group efficacy in terms of the way it is measured. Collective efficacy is aggregation of individual members' belief about group's efficacy whereas group efficacy is consensus about a group's efficacy (Mulvey and Klein, 1998). Three different ways of measuring collective efficacy has been highlighted in the literature. The first method utilizes the aggregation of perceptions of individual self-efficacies, the second method uses the average individual perceptions of group efficacy, and the third method involves the generation of a single consensual measure of group efficacy (Katz-Navon and Erez, 2005).

Whiteoak (2004) found that the three methods of measuring group efficacy highlighted earlier did not differ in terms of their consistency, the magnitude of their relationship with goals or the degree to which they were affected by performance for the kind of task they studied.

This study adopted the method of aggregating individual self-efficacies in arriving at the pair's collective efficacy. However, there are concerns in this way of measuring the efficacy beliefs (Katz-Navon and Erez, 2005) due to the varying levels of analysis with one being at the individual level and the other being at the group level. In

measuring the individual self-efficacy beliefs and aggregating the scores, Bandura notes that it is not possible to fully disentangle individual beliefs from the group context as the individual will use group referent implicitly while forming the collective efficacy beliefs.

4.8.5 Communication

Communication has been measured in many ways in the literature. In some of the studies, entire communication, in the group, was transcribed and coded using various coding schemes such as Bales, Interaction Process Analysis, and Hirokawa's functional coding scheme. Observers were also used in the past to code while the group was performing the task. Based on the code, communication patterns were analyzed and it was related to group decision making/problem solving effectiveness. Researchers also used network analysis to study the communication patterns in the software development teams (Brodbeck, 2001). Perceptual measures for measuring various communication constructs have also been attempted to. Most of such studies have used survey methodology to measure these constructs using well validated scales.

Hoegl and Gemuenden (2001) analyzed software development teams and measured quality of communication on these dimensions: frequency, formalization, structure, and openness of information exchange. Of these two of the dimensions were used in this study to measure communication between the partners in a collaborating pair. Formalization and structure dimensions of communication quality were not measured in this study as it pertained to teams of larger size. Six items were used to measure this construct.

4.8.6 Subjective Mental Work Load (SMWL)

System analysis tasks are equated to an ill-structured problem solving exercise (Agarwal, et al., 1996). Past research in system analysis have tried to capture the cognitive processes of the analysts through process tracing techniques such as protocol analysis (Sen, 1997). Instead of relying on verbalization techniques, (Morris, et al., 1999) used NASA's Subjective Mental Workload questionnaire to asses the subject's mental work load during object-oriented and structured system analysis task. According to Hart and Staveland (1988), mental workload emerges from the interaction between requirements of a task, the circumstances under which it is performed, and the skills, behaviors, and perceptions of the operator.

Past research has measured subjective mental workload with NASA's Task Load Index (NASA-TLX)(Morris, et al., 1999; Speier and Morris, 2003). In the NASA's TLX, SMW consists of seven dimensions: mental demand, physical demand, temporal demand, performance, effort, and frustration level. This study adapted NASA's TLX instrument by eliminating physical demand in order to suit the study conditions. In total five dimensions of mental workload were measured in this study.

4.8.7 Overall Task Satisfaction

Overall task satisfaction represents the affective response of the individual to the overall task performance. The scale for overall task satisfaction was based on the existing task satisfaction measures in the literature (Balijepally, 2005).

Table 4.1 Scales for the Constructs

Construct	Number of items	Source
Design-self efficacy	6	Adapted from self-efficacy scales of Jones (1986) and Kankanhalli, et al.(2005)
Communication	6	Adapted from communication measure of (Hoegl and Gemuenden, 2001)
Subjective mental workload	15	Adapted NASA TLX instrument (Morris, et al., 1999)
Task satisfaction	4	Balijepally (2005)

4.9 Debriefing

Debriefing of the subjects in all the four treatment conditions were done. Subjects were informed of the research objective and were asked not to reveal details about the experiment to their friends and colleagues who were planning to participate. Subjects were also given an opportunity to get clarification on any aspect of the study. Appendix C presents the debriefing form used to debrief the subjects.

4.10 Statistical Analysis

Multiple statistical procedures were planned to be utilized to test the various hypotheses presented in the study. Before the actual hypotheses testing, preliminary analysis of the data was done using procedures such as factor analysis, and reliability analysis. Suitable assumption checks as well as the manipulation checks were also carried out.

Testing hypothesis H_{1a}, H_{1b}, H_2, and H_3 require the use of nominal pairs. The dependent measures utilized in these hypotheses: quality of solution, and time taken

were measured at the group level for the collaborating pairs. Nominal pairs were be created by randomly assigning individuals to nominal pairs. In total 24 nominal pairs were planned to be made. Nominal pair design allows us to identify first best in the nominal pair and 2nd best in the nominal pair. Hence two-way Analysis of Variance (ANOVA) with one factor being mode of participation with three levels (collaborating pairs, 1st best individual in the nominal pair, and 2nd best individual in the nominal pair) with second factor being the availability of design patterns was used to analyze the solution quality of subjects. Subject's experience in object-oriented programming language was used as a covariate in this analysis.

Hypothesis H_4 and H_5 utilizes dependent measures, subjective mental work load and solution quality that were measured individually in the pair condition. Hence two-way ANOVA procedure will be utilized. The first factor, the mode of participation had two levels: individual and collaborating pair. The second factor, availability of codified knowledge had two levels: with design patterns and without design patterns.

Analysis pertaining to potential mediator hypotheses were done using the general procedures outlined by Baron and Kenny (1986). According to Baron and Kenny (1986), to test the presence of mediators, three separate regression analysis are needed. The three regression equations are: a) regression of mediator on the independent variable, b) regression of the dependent variable on the independent variable, and c) regression of dependent variable on both the independent variable and on the mediator.

This study used categorical independent variable. Regression analysis with indicator variable for the availability of patterns could be performed to test the mediators.

(1) Simple regression analysis with availability of pattern as an independent variable design self-efficacy as the dependent variable.

(2) Simple regression analysis with availability of pattern as an independent variable and quality of solution as the dependent variable.

(3) Multiple regression analysis with availability of pattern and design self-efficacy as the independent variables and quality of solution as the dependent variable.

For the pair condition, two mediators were proposed in this study. These variables are collective design efficacy and communication quality. Mediator analysis very similar to one that was carried out for the individual condition was utilized for the pair condition also. Next chapter presents the hypotheses tests that were carried out.

CHAPTER 5

RESEARCH RESULTS

This chapter presents the results of preliminary analyses of the data, hypotheses testing and manipulation checks in the experimental conditions.

5.1 Preliminary Analyses

Preliminary analyses of the data were carried out before proceeding to the actual hypotheses testing. Analyses of the sample characteristics, and validity and reliability of the dependent measures were done. Tests for various assumptions of the statistical techniques used for the hypotheses testing were also done.

5.1.1 Sample Characteristics

In total 100 subjects participated in the experiments conducted over a period of eight experimental sessions. Data from 4 subjects were dropped and this balanced the design.

Reasons for dropping 4 subject's data are: (1) Subject deciding to leave early: one subject in individual – pattern condition, (2) Subject not completing the dependent measures questionnaire: one subject in individual – no pattern condition and a pair of subjects in the pattern condition. After dropping these 4 subjects, the usable sample size became 96. Figure 5.1 describes various experimental conditions with the related sample sizes. Table 5.1 presents the demographics characteristics of the subjects.

Mode of Participation	Codified Knowledge: Patterns	Codified Knowledge: No patterns
Individual	Condition I n = 24[a] 12 nominal pairs	Condition II n = 24[b] 12 nominal pairs
Pair	Condition III n = 24[b] 12 collaborating pairs	Condition IV n = 24 12 collaborating pairs

Total sample size (N) = 96
a – a subject was dropped from the sample for not completing the task.
b – a subject and a pair were dropped for not completing the measures completely.

Figure 5.1 Two-by-Two Factorial Research Design

Table 5.1 Subject Characteristics

	Description	Subjects	% of subjects
Gender	Male	85	88.5 %
	Female	11	11.5 %
Age	< 25	14	14.6 %
	26 – 30	31	32.3 %
	31 – 35	25	26.0 %
	36 – 40	10	10.4 %
	41 – 45	7	7.3 %
	46 <	6	6.3 %
	Did not provide	3	3.1 %
Education	High school	2	2.1 %
	Community college	6	6.3 %
	Undergraduate degree	30	31.3 %
	Graduate degree	55	57.3 %
	Doctoral degree	1	1.0 %
	Did not provide	2	2.1 %
Programming Experience	0-1 years	4	4.2 %
	1-2 years	4	4.2 %
	2-4 years	16	16.7 %
	4-6 years	28	29.2 %
	6 < years	44	45.8 %
OO Programming Experience	0-1 years	11	11.5 %
	1-2 years	17	17.7 %
	2-4 years	36	37.5 %
	4-6 years	14	14.6 %
	6 < years	17	17.7 %
	Did not provide	1	1.0 %
OO Design Experience	No experience	2	2.1 %
	Novice	25	26.0 %
	Intermediate	61	63.5 %
	Expert	8	8.3 %

Analysis of variance tests were done to check whether the demographic characteristics of subjects differed across the four treatment conditions. Results of the tests are as follows: education ($F = 0.206$; $p = 0.892$), programming experience ($F = 0.609$; $p = 0.611$), object-oriented programming experience ($F = 0.408$; $p = 0.748$), and object-oriented design experience ($F = 0.361$; $p = 0.781$). Results of the ANOVA analyses of self-reported educational level, programming experience, object-oriented programming experience, and object-oriented designing experience showed no significant differences across the four treatment conditions.

One of the primary manipulations done in this study was the availability of codified knowledge in the form of design patterns. If the participants were quite familiar with design patterns then they might not need the documents related to design patterns to apply it in a given design problem. Hence, if the subjects were experienced in the use of design pattern then the manipulation of design pattern availability would lose its meaning. In order to avoid this situation, prospective subjects for the study were clearly told that the study utilizes subjects who have not used Design Patterns in their work (refer to appendix A).

In order to check whether the subjects were familiar with design patterns, before the experiment began, subjects were asked to rate their familiarity with design patterns. In the past, Shaft and Vessey (2006) had taken such an approach in their experiment on software comprehension to ascertain the domain experience of subjects. Response of the subjects indicated that 98% of the subjects had little or no experience on design patterns. An ANOVA test of the subjects' familiarity with design patterns across four

treatment conditions showed no significant differences ($F = 1.412$; $p = 0.244$). Results from these tests indicated that the subjects were sufficiently unfamiliar with the design patterns and the subject characteristics did not vary across the treatment conditions.

Subjects were randomly paired to form the collaborating pairs. There was a chance that the members of a resultant pair had previous experience of working together in their workplaces. This study was not designed to exclude such pairs; nevertheless the collaborating pairs were asked whether they had worked together before. Of the 24 collaborating pairs, members of 3 pairs had worked together before the experiment. Hence, 88% of the members collaborating pairs used in the study did not have previous experience of working together.

Of the reported demographics variables, object-oriented programming experience was used as a covariate in the subsequent analysis. The design task used in the study pertained to the creation of class diagrams that in reality will be translated in to object-oriented programming code. This variable was found to be a significant covariate ($F = 7.818$; $p= 0.007$).

5.1.2 Characteristics of Dependent Variables and Mediators

The variables that were measured in this study are: 1. Quality of solution; 2. Time taken; 3. Subjective Mental Work Load; 4. Task satisfaction; 5. Design self-efficacy; and 6. Communication quality. The following paragraphs describe the preliminary analyses done with regard to these measures.

Quality of solution: Solutions provided by the various subjects were evaluated based on a predefined grading scheme. This grading scheme was devised based on the

solution of an expert. Appendix G outlines the grading scheme. Two PhD students independently graded all the 72 solutions. The resultant scores were assessed for inter-rater reliability and it was found to be 0.960. Because of the high inter-rater reliability, the average of the two scores was used in the subsequent analysis as the score for solution quality.

Subjective Mental Work Load (SMWL): This was measured using an adapted version of NASA's Task Load Instrument. SMWL is conceptualized with five dimensions: mental demand, time constraints, performance, effort, and level of frustration. This instrument had 15 items in total. Of these 15 items, 10 items pertained to the pairs of the underlying dimensions. Subjects were asked to select a dimension in the pair that contributed mostly to their experienced mental work load. Score for SMWL was arrived based on the counts of number of times each dimension influenced the mental workload and multiplied by the salience of each dimension that was rated on a scale of 0 to 100. Dimension scores were added to together to obtain an overall measure of SMWL. Procedures for calculating the SMWL was based on the works of Speier and Morris (2003).

Time: Time taken to complete the main design task was measured as one of the dependent variables. Maximum of 80 minutes was allowed for the main task. The average time taken to complete the task was 57.24 minutes with a standard deviation of 17.49. In total, 6 individuals and 6 pairs took the whole allotted time of 80 minutes.

Perceptual Measures: There were three perceptual constructs used in the study and they were design self-efficacy (DSE), overall task satisfaction (TS), and

communication (COMF and COMQ). These variables were measured using sixteen items. Of the three variables Communication quality was measured only in the pair condition. The other two variables were measured across all the treatments conditions.

Since not all the perceptual measures were measured in all of the four treatment conditions, two separate factor analyses were done in order to determine the factor structure. In the first factor analysis, items pertaining to design self-efficacy and task satisfaction were used. Table 5.2 presents the factor loadings. Exploratory factor analysis with varimax rotation of the perceptual measures yielded two factors. These factors pertained to Design Self-efficacy and Task satisfaction.

Table 5.2 Rotated Factor Matrix for the Perceptual Measures

	Factor 1 Design Efficacy	Factor 2 Task Satisfaction	Communality
DSE1	**0.797**	0.210	0.679
DSE2	**0.891**	0.195	0.833
DSE3	**0.855**	0.280	0.810
DSE4	**0.720**	0.108	0.530
DSE5	**0.865**	0.343	0.866
DSE6	**0.855**	0.185	0.765
TS1	0.259	**0.810**	0.723
TS2	0.178	**0.894**	0.831
TS3	0.275	**0.840**	0.782
TS4	0.146	**0.812**	0.680
Eigen Values	4.353	3.145	
Cumulative Variance Explained	43.53%	74.98%	

In the pair conditions, aspects of communication were also measured using a six item scale. Separate factor analysis was done for this to ascertain the factor structure.

Communication was measured using 6 items and they loaded in to two separate factors. These factors pertained to communication frequency and quality. Hence two dimension of communication was used to analyze the data separately. Following table 5.3 presents the results of this factor analysis.

Table 5.3 Rotated Factor Matrix for Communication Measures

	Component		
	Factor 1	Factor 2	Communality
COM1	**0.911**	0.195	0.868
COM2	**0.966**	0.127	0.948
COM3	**0.880**	-0.022	0.774
COM4	0.171	**0.643**	0.443
COM5	0.001	**0.891**	0.794
COM6	0.062	**0.900**	0.814
Eigen Values	2.569	2.072	
Cumulative Variance Explained	42.82%	77.35%	

Reliability of Measures: Reliability of the perceptual measures were calculated in terms of Cronbach's alpha and they are as follows: Design Self-Efficacy (0.93), Task satisfaction (0.89), Communication Frequency (0.91), and Communication Quality (0.75). According to Nunnally (1978), reliabilities exceeding 0.70 are considered to be acceptable.

The time taken to solve the design problem and the solution quality was assessed at the individual level. Correlation between the solution quality and time taken was 0.268. The mean and standard deviation solution quality score was 38.951 and 15.462. The mean and standard deviation of time taken was 56.888 and 17.590. The following table 5.4 presents the correlations of measures that were assessed at the

individual level. The diagonals values depict the mean and standard deviations (in parenthesis) for all of these measures.

Table 5.4 Correlations of Measures at the Individual Level

	OOEV	DSE	SMWL	TS	COMF	COMQ
Object-oriented Programming Experience (OOEV)	3.094 (1.223)					
Design Self-efficacy (DSE)	0.347	4.639 (1.147)				
Subjective Mental Work Load (SMWL)	-0.164	-0.158	62.207 (13.336)			
Task Satisfaction (TS)	0.143	0.430	-0.256	5.117 (1.226)		
Communication Frequency (COMF)	-0.123	0.175	0.410	-0.271	6.383 (1.043)	
Communication Quality (COMQ)	-0.043	0.173	0.168	0.136	0.195	5.882 (0.976)

Power Analysis: Power analysis was conducted for the experimental results. Effect sizes were calculated using the procedures outlined by Levine and Hullett (2002). Cohen(1992) classified effect sizes into three general levels: small effect size = 0.10, medium effect size = 0.25 and large effect size = 0.40. In this study, most of the effect sizes (eta-squared) were of small in nature. In some cases, especially the analysis of pair mediation model, and analysis of subjective mental workload involves very low effect size and they correspond to very low power level and this is of concern. Power levels for other factors are adequate to find differences in the various treatment conditions. The following table 5.5 summarizes the results of this analysis.

Table 5.5 Power Analysis

Dependent Variable	Factor	Effect Size (Eta Squared)	Partial Eta Squared	Power at alpha = 0.05
Solution Quality	Mode of participation	0.141	0.184	0.927
	Availability of Pattern	0.086	0.120	0.835
Time	Mode of participation	0.135	0.149	0.849
	Availability of Pattern	0.082	0.096	0.736
Subjective Mental Work Load	Mode of participation	0.000	0.005	0.095
	Availability of Pattern	0.001	0.001	0.055
Task Satisfaction	Mode of participation	0.043	0.046	0.545
	Availability of Pattern	0.050	0.053	0.612
Individual Mediation Model				
Solution Quality	Availability of Pattern	0.115	0.115	0.667
Design Self-efficacy	Availability of Pattern	0.002	0.002	0.061
Solution Quality	Availability of Pattern	0.106	0.119	0.673
	Design Self-efficacy	0.105	0.120	0.679
Pair Mediation Model				
Solution Quality	Availability of Pattern	0.050	0.050	0.178
Design Collective-efficacy	Availability of Pattern	0.205	0.205	0.625
Solution Quality	Availability of Pattern	0.001	0.001	0.052
	Design Collective-efficacy	0.147	0.155	0.466
Communication Frequency	Availability of Pattern	0.865	0.865	0.145
Communication Quality	Availability of Pattern	0.005	0.005	0.061
Solution Quality	Availability of Pattern	0.048	0.048	0.166
	Communication Frequency	0.000	0.000	0.050
Solution Quality	Availability of Pattern	0.047	0.047	0.164
	Communication Quality	0.011	0.011	0.077

5.1.3 Assumptions Tests

This study utilized ANOVA/ANCOVA techniques to analyze the various hypotheses. Underlying assumptions of ANOVA models are: (1) constancy of error variance, (2) independence of error terms, and (3) normality of error terms were tested for. ANCOVA models also rely on two additional assumptions: (1) Equality of slopes of the different treatment regression lines, and (2) linearity of regression lines (Neter, et

al., 1996). Apart from these assumption tests, tests were performed to detect outliers in the data.

Assumptions of constancy of error terms: This assumption was tested using the Modified Levene test for the ANOVA procedure. Results of these tests indicated that the assumption of constancy of error terms cannot be rejected. Table 5.6 presents the results of these tests.

Table 5.6 Assumptions of Normality and Constancy of Error Variance

Variable	Omnibus Test for Normality		Modified Levene test for Constant Variance	
	Statistic	Significance	Statistic	Significance
Solution Quality	5.392	0.068	0.551	0.737
Time	1.993	0.369	1.139	0.349
Task Satisfaction	13.665	0.001	0.123	0.946
Subjective Mental Work Load	4.489	0.106	2.061	0.112
Individual Mediation Model				
Design Self-efficacy	0.299	0.861	0.456	0.503
Solution Quality	6.030	0.050	0.068	0.795
Pairs Mediation Models				
Design Collective Efficacy	2.487	0.288	0.269	0.609
Solution Quality	0.051	0.975	0.284	0.600
Communication Frequency	12.300	0.002	0.608	0.444
Communication Quality	1.753	0.416	0.051	0.823

Assumptions of Normality: Data were coded with indicator variables and multiple regression procedure was used to test assumptions. Normality of error terms were checked using Omnibus Normality tests. The normality assumptions could not be rejected for most of variables. Task satisfaction and communication frequency variables alone had normality assumptions violations according to the Omnibus test. Further

analyses were done taking this aspect into consideration. Following table 5.6 presents results of these tests.

Assumptions of independence of error terms: When the data is obtained in a time sequence or there is some logical sequence in the way data is ordered then assumptions of independence of error terms may be of an issue. Neter et al. (1996) suggest that the randomization in a study could be the most important insurance policy. This experiment randomly assigned subjects to various treatments and hence violation of this assumption is unlikely.

Assumptions on equality of treatment slopes: This study utilized experience with object-oriented programming as the covariate. An F-test was conducted to test the equality of slopes across the treatment conditions as specified by Neter, et al (1996). Multiple regression analysis with indicator variables for the treatment conditions was used to create a full model and a reduced model. Full model utilized all the treatment conditions and their interaction with the covariate. Reduced model had the treatment variables and the covariate alone. At alpha = 0.05, the equality of slopes assumptions could not be rejected.

Assumptions regarding linearity of regression relation: A general linear regression model with the covariate was used to test the linearity assumption. Neter et al (1996) specify that linearity of regression function can be evaluated using the residual plot against the fitted values. Hence a residual plot was used to test this and there was no apparent departure from linear relationship.

5.1.4 Tests for Interactions and Significance

According to Neter, et al. (1996), analysis of factor effects in a two factor analysis of variance should be done after considering the effect of interactions. Presence of significant interactions between the factors would require the use of cell means model.

Tests for significant interactions were carried out utilizing ANCOVA procedure. As mentioned earlier, experience in Object-oriented programming was considered as the covariate. Each of the dependent measure was tested separately in a 3 (best individual, second best individual, collaborating pair) and 2 (Pattern and no pattern) ANCOVA design.

Table 5.7 Results of Test for Interactions

Dependent Variables	Mode of Participation F-ratio (p-value)	Availability of Design Patterns F-ratio (p-value)	Interaction F-ratio (p-value)
Quality of Solution	7.315 (0.001)	8.874 (0.004)	0.391(0.678)
Time	6.584 (0.002)	5.982 (0.017)	0.060 (0.941)
Task Satisfaction	4.378 (0.039)	5.140 (0.026)	0.784 (0.378)
Subjective Mental Work Load	37.142 (0.205)	0.366 (0.547)	1.081 (0.302)

The table 5.7 summarizes the results and it can be seen that there is no significant interaction effects at alpha < 0.05 for all the four dependent variables. Since there were no significant interactions, factor level means were used for further analysis.

Control for experiment-wide error rate: This study utilized four dependent variables. There are two ways in which group comparison can be made with multiple outcome variables: (a) conduct multiple Analysis of Variance (ANOVA)s or (b) conduct a Multivariate Analysis of Variance (MANOVA) followed by multiple

ANOVAs (Huberty and Morris, 1989). Multiple ANOVAs are done when the outcome variables are conceptually different. In this study, baring the performance measures of time and solution quality, the other two variables are conceptually different. Moreover, according to Hair, et al.(1998), MANOVA requires a recommended minimum cell size of 20. The minimum cell size in this study is 12 and does not meet the criteria for MANOVA.

Huberty and Morris (1989) outline a procedure for adjusting the overall Type I (Bonferroni type adjustment) error probability while using multiple univariate tests. For m tests, the alpha level for each test (α_1) is given by the overall alpha level α_m divided by m. Hence for this study m = 4 and for a family wide $\alpha = 0.05$, we get $\alpha_m = 0.0125$. As can be seen from the table 5.7, most of the main effects that were found significant had p-values less than 0.0125. This should dispel any concerns about the experiment-wide error rate. Significance of mode of participation on task satisfaction was p = 0.039 and this exceeded the overall experiment-wide α_m of 0.0125. Caution is advised on the interpretation of results pertaining to the task satisfaction outcome measure.

5.1.5 Manipulation Checks

This study had two treatments: mode of participation and availability of design patterns (codified knowledge). Manipulation checks were done to see whether these treatments had the intended effects.

Mode of Participation: In this treatment there are two levels: Individual and collaborating pair. Subjects in the collaborating pairs were asked individually about the active participation of the other member of the pair using a questionnaire item that

utilized a 7 point Likert scale (1 -did not agree to 7 – strongly agree). Descriptive statistics results indicate that the mean level of active participation is 6.28 with standard deviation of 0.886. No subject rated below 4.0 (i.e.) neutral rating for their partner's participation during the design task performance. Based on this, it is argued that the design solution was indeed arrived collaboratively in the pair condition.

Availability of Design Patterns: Subjects in the availability of design condition were provided with design patterns to be used during the task performance. Participants under all the four treatment conditions were asked about their use of design patterns to arrive at the solution. In the availability of pattern condition, 94% of the participants used one or more design patterns to arrive at the solution. In the non-availability of pattern condition, only 5.5% of the participants used the correct design patterns to arrive at the solution. In order to ascertain whether the use of design patterns in the no patterns was a cause for concern, two separate ANOVA analyses were done. First ANOVA test included the data from the subjects that had used design patterns in the no pattern condition and other ANOVA test excluded the same data. Results from these tests did not differ significantly and hence the subsequent analyses were done utilizing the complete data set. Table 5.8 presents the results of the tests.

Table 5.8 Pattern Manipulation on Solution Quality

ANOVA Results	Mode of Participation F-ratio (p-value)	Availability of Design Patterns F-ratio (p-value)	Interaction F-ratio (p-value)
With all available data	7.315 (0.001)	8.874 (0.004)	0.391(0.678)
Excluding data of subjects who used patterns in the no pattern condition	7.094 (0.002)	10.801 (0.002)	0.373(0.690)

Results of the manipulation checks indicate that both the manipulations have worked in the intended way. Hence the next section presents the subsequent hypotheses testing that were done.

5.2 Hypothesis Testing

This study utilized ANOVA/ANCOVA models to test the various hypotheses elucidated earlier. Following analyses were done:

(1) Two-Way ANCOVA for dependent measure of quality of solution, and time

(2) One-way ANOVA for Task Satisfaction and Subjective Mental Work Load

(3) Mediator Analyses were done using series of three simple/multiple regression analysis.

5.2.1 Pairs versus Individuals Comparison for Solution Quality

This study utilized nominal pairs to compare with collaborating pairs. Nominal pairs were formed by randomly grouping two individuals in a given treatment condition. Based on these nominal groupings, best and second best individual in nominal pairs were arrived at. Hypotheses concerning collaborating pairs and individual performance were analyzed using a 3 (Best individual, second best individual in nominal pair, and collaborating pair) x 2 (Pattern, No Pattern) ANCOVA design. Prior programming experience with object-oriented languages was used as the covariate. In the collaborating pair, the average experience of the pair was used as the covariate. Table 5.9 presents the results of this analysis.

Table 5.9 ANCOVA Model Results for Solution Quality

Source	Sum of Squares	df	Mean Square	F-ratio	p-value
Object-oriented programming Experience	1282.311	1	1282.311	7.819	.004
Participation	2399.411	2	1199.705	7.315	.001
Pattern	1455.441	1	1455.441	8.874	.004
Participation * Pattern	128.387	2	64.193	.391	.678
Error	10660.543	65	164.008		
Corrected Total	16973.580	71			

ANCOVA results indicated the presence of significant main effects of both participation and the availability of patterns and absence of interaction effect present. Because of this further analysis on factor levels means was done.

Hypothesis H_{1a} and H_{1b} were examined using pair-wise comparison between the levels of participation. Bonferroni's procedure is best when the number of contrasts of interest is small and has been specified in advance and the number of contrasts of interest is about the same as the number of factor levels or less (Neter, et al., 1996).

H_{1a}: Performance in terms of solution quality of the best individual in a nominal pair will be better than the collaborating pair's performance.

H_{1b}: Performance in terms of solution quality of collaborating pairs will be better than the second best individual in nominal pairs.

The table 5.10 presents the comparisons that were tested using Bonferroni's custom comparisons.

Table 5.10 Bonferroni's Custom Comparison for Solution Quality

	Mean	Standard Deviation	t-value	p-value
Hypothesis H_{1a}				
Best Individual – Nominal Pair	45.292	16.082	0.842	0.403
Collaborating Pair	42.104	12.993		
Hypothesis H_{1b}				
2nd best Individual – Nominal Pair	29.458	12.813	2.875	0.005
Collaborating Pair	42.104	12.993		

As can been seen hypothesis H_{1b} was supported. Performance of collaborating pairs was better than the second best individual in the nominal pair. Performance of the best individual in a nominal pair is not significantly better than the performance of the collaborating pair though the mean solution quality value is somewhat higher than the collaborating pair's value (45.3 vs. 42.1).

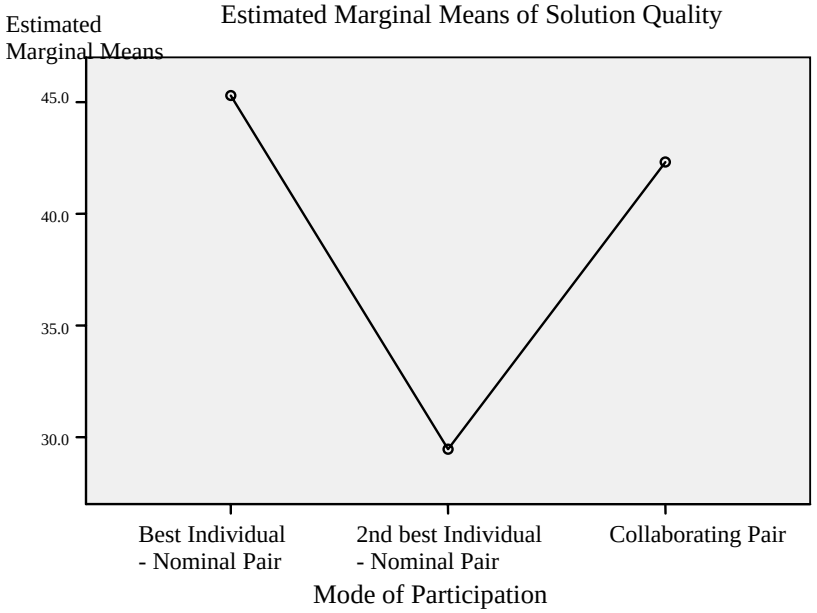


Figure 5.2 Treatment Means Plot for Solution Quality

Further analysis of solution quality in terms of average individual performance of a nominal pair over the collaborating pairs' performance revealed no statistical difference between them. Collaborating pairs had higher average score in terms of solution quality when compared to average individual scores in the nominal pairs (42.10 vs. 37.1).

5.2.2 Pairs versus Individuals Comparison for Time

Hypotheses concerning collaborating pairs and individual performance in terms of time taken to complete the design task were analyzed using a 3 (Best individual, second best individual in nominal pair and collaborating pair) x 2 (Pattern, No Pattern) ANOVA design. The analysis was quite similar to the earlier one regarding the solution quality of the design. Table 5.11 presents the results of this analysis. ANCOVA results indicated the presence of significant main effects of both participation and the availability of patterns and there were no significant interaction effect present. Because of this further analysis on factor levels means was done.

Table 5.11 ANCOVA Model Results for Time

Source	Sum of Squares	df	Mean Square	F-ratio	p-value
Object-oriented programming Experience	633.236	1	633.236	2.444	.062
Participation	2955.259	2	1477.629	5.704	.005
Pattern	1790.595	1	1790.595	6.912	.011
Participation * Pattern	99.105	2	49.552	.191	.826
Error	16839.097	65	259.063		
Corrected Total	21969.111	71			

Hypothesis 2 was examined using pair-wise comparison between the levels of participation and time taken. Bonferroni's custom comparison was done to compare the time taken by collaborating pairs with the average time taken by individuals. The table 5.12 presents the results of this analysis.

H_2: Collaborating pairs will take longer to complete the design task than the average nominal pair.

Table 5.12 Bonferroni's Custom Comparison for Time

Factor Level	Mean	Standard Deviation	t-value	p-vale
Best Individual – Nominal Pair	63.83	19.174	2.832074	0.006
2nd best Individual – Nominal Pair	48.96	16.241		
Collaborating Pair	64.29	13.741		

This hypothesis was supported and collaborating pairs took significantly more time than the average nominal group performance time at p=0.006. Moreover, further analysis revealed that the best individual in a nominal pair took almost similar time to complete the design task.

5.2.3 Availability of Codified Knowledge and Solution Quality

Effect of codified knowledge on the solution quality was analyzed using a 3 (Best individual, second best individual in nominal pair and collaborating pair) x 2 (Pattern, No Pattern) ANCOVA design. Experience with object-oriented programming was used as the covariate for this analysis.

Refer table 5.8 for the ANCOVA test results. Since the interaction term (mode of participation x availability of codified knowledge) is not significant (p-value = .678), analysis of main effects became salient. Hypothesis H_3 explored the main effects of the codified knowledge.

H_3: Solution quality will be higher in the design pattern condition than in the condition without design pattern.

The following table 5.13 presents the means and standard deviations for both, pattern and no pattern condition.

Table 5.13 Solution Quality across Codified Knowledge Treatment

Condition	Mean Solution Quality	Standard Deviation
No Pattern	34.319	15.304
Patterns	43.722	14.457

Results of the ANCOVA analysis show that these means are significantly different and the p-value = 0.004. Refer table 5.8 for the results of the ANCOVA analysis. Because of this we can conclude that the mean solution quality under pattern condition is significantly better than the mean solution quality in the no pattern condition.

Table 5.14 presents the means and standard deviations for both the performance measures. Figure 5.3, plots the means for solution quality and time taken across various treatment conditions.

Table 5.14 Solution Quality and Time across Treatments

Treatment	Count	Solution Quality		Time Taken	
		Mean	Standard Deviation	Mean	Standard Deviation
1stBestNP: NoPatterns	12	40.13	15.49	63.83	15.82
1stBestNP: Patterns	12	50.45	15.57	53.08	21.34
2ndBestNP: NoPatterns	12	23.58	10.99	52.50	19.14
2ndBestNP: Patterns	12	35.33	12.13	45.42	12.56
CPairs: NoPatterns	12	39.25	13.97	69.58	13.37
CPairs: Patterns	12	44.96	11.83	56.92	13.41

Note:
1stBestNP: 1st Best Nominal Pair
2ndBestNP: 2nd Best Nominal pair
CPairs: Collaborating Pair

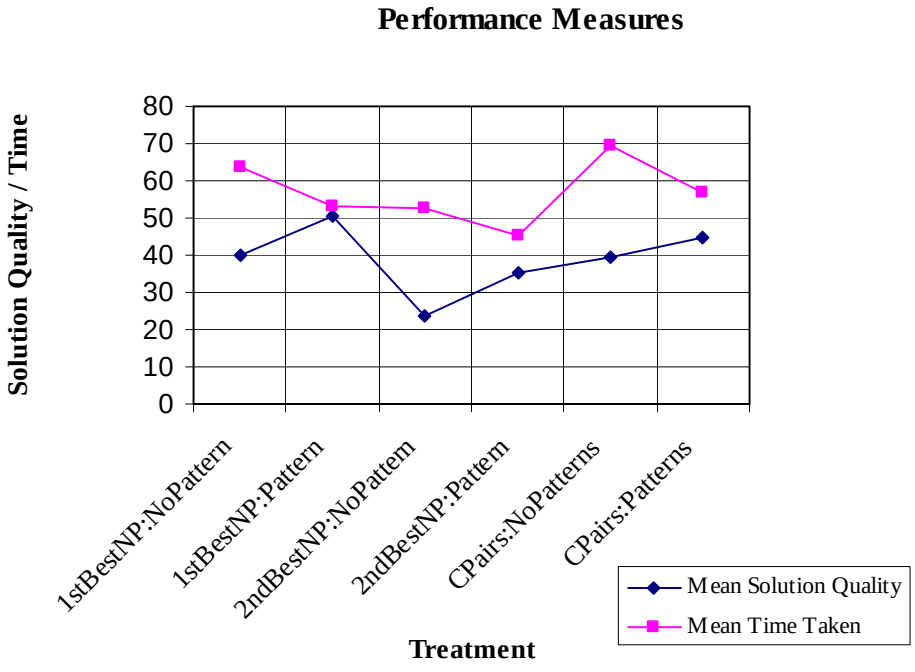


Figure 5.3 Plot for Solution Quality and Time across Treatments

5.2.4 Subjective Mental Work Load: Patterns and No Patterns

Subjective Mental Work Load was measured individually in the collaborating pair conditions. Hence two-way ANOVA model with two levels for each of the factor was used to examine the hypothesis pertaining to the mental workload. Subject's Object-oriented programming experience was used as the covariate. Fourteen of the 96 subjects did not fully complete the SMWL instrument. Due to this, reduced sample size of 82 was used to examine the following hypothesis.

H_4: Subjective mental work load will be lower in the design pattern condition than in the condition without design pattern.

Table 5.15 ANCOVA Model Results for SMWL

Source	Sum of Squares	df	Mean Square	F-ratio	p-value
Object-oriented programming experience	377.001	1	377.001	2.115	.075
Participation	7.198	1	7.198	.040	.841
Pattern	70.737	1	70.737	.397	.531
Participation * Pattern	210.513	1	210.513	1.181	.281
Error	13727.21	77	178.275		
Corrected Total	14405.476	81			

The table 5.15 presents ANCOVA results and as can be seen SMWL is not significantly influenced by the availability of patterns. Hence this hypothesis was not supported.

5.2.5 Task Performance Satisfaction: Individuals and Groups

Satisfaction with the task performed was measured individually and hence separate data for both the members of the collaborating pairs were available.

Assumptions of normality of error terms assumptions were violated in the Omnibus Normality test by the task satisfaction variable. Lack of normality is not a major concern for fixed ANOVA models and Kurtosis of error distribution is more important than skewness in terms of the effects on inference (Neter et al, 1996). Assumption regarding Kurtosis could not be rejected at alpha = 0.05. Hence for analyzing this hypothesis, two-way ANOVA model with two levels for each of the factor was used.

H_5: Satisfaction with the task performance will be higher in the collaborating pair condition than in the individual condition.

Table 5.16 ANOVA Model Results for Task Satisfaction

Source	Sum of Squares	df	Mean Square	F-ratio	p-value
Participation	6.126	1	6.126	4.387	.039
Pattern	7.178	1	7.178	5.140	.026
Participation * Pattern	1.094	1	1.094	.784	.378
Error	128.471	92	1.396		
Corrected Total	142.869	95			

Results of the tests are presented in table 5.16. As can be seen there are significant differences at alpha = 0.05 between the satisfaction levels of subjects who performed alone and the subjects who worked in collaborating pairs. Figure 5.4 presents the means of overall task satisfaction and the table 5.17 presents the factor levels means for overall task satisfaction. Caution is advised in interpreting the results of this hypothesis tests as the p-value exceeds the experiment-wide α_m value of 0.0125.

Table 5.17 Factor Level Means of Overall Task Satisfaction

Factor Level	Task Satisfaction Mean	Standard Deviation
Individual	4.865	1.282
Pair	5.370	1.125

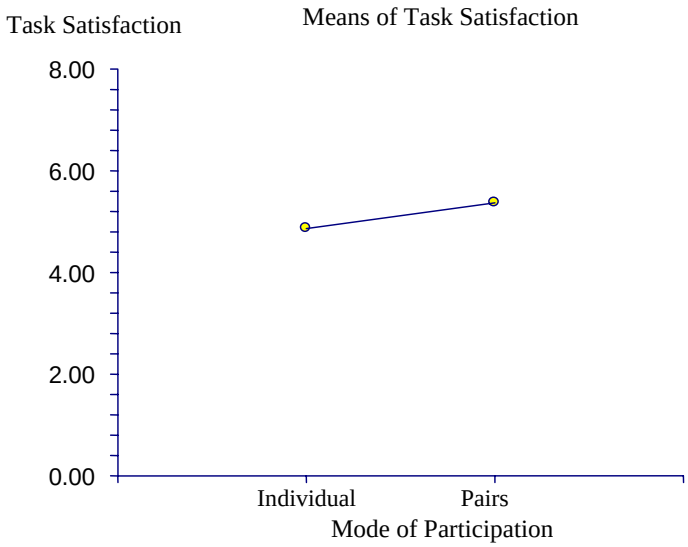


Figure 5.4 Plot for Mean Overall Task Satisfaction across Participation

5.3 Testing of the Mediation Model – Individual Condition

Perception about design self-efficacy (DSE) was hypothesized to be the mediating variable between pattern availability and task performance. Based on this argument, following two hypotheses were formulated:

H_{i1}: Individuals in the pattern condition will exhibit significantly higher design self-efficacy than individuals in no pattern condition.

H_{i2}: Perceptions of design self-efficacy will mediate the positive relationship between pattern availability and design task performance.

To examine these hypotheses, three regression analyses were performed. Availability of patterns was represented by an indicator variable.

a) Regressing design self-efficacy on the availability of pattern treatment,

b) Regressing solution quality on the availability of pattern treatment, and

c) Regressing of solution quality on both the availability of pattern and on the design self-efficacy.

Following table 5.18 summarizes the results of these tests.

Table 5.18 Results for the Mediator Analysis – Individual Condition

Step	Dependent Variable	Independent Variables	t-value	Significance
1	Design self-efficacy (DSE)	Pattern	0.312	0.378
2	Solution Quality	Pattern	2.444	0.009
3	Solution Quality	DSE Pattern	2.477 2.460	0.009 0.009

From the results of this analysis, it can be inferred that the hypothesis H_{i1} is not supported. Since design pattern does not affect the perceptions of design self-efficacy the hypothesis about it being a mediator cannot be supported too. Subsequent regression analysis revealed that design self-efficacy itself is significant predictor of solution quality (t=2.462, p=0.009). Table 5.19 also shows that the design self-efficacy is significant (t-value=2.477, p=0.009) when it is added as an independent variable in the model with patterns.

5.4 Testing of the Mediation Models – Pair Condition

Two mediators were hypothesized for the collaborating pair condition: collective design self-efficacy and communication quality. Collective efficacy was arrived at based on the average of individual self-efficacy of a collaborating pair. In the literature, it has been found that this way of measuring collective efficacy is comparable to measuring efficacy beliefs at the group level. Communication quality was measured using two dimensions with focus on frequency of communication exchange and quality of the exchange. Hypotheses that pertain to mediation analyzes are:

H_{p1}: Pairs in the pattern condition will exhibit significantly higher collective efficacy than pairs in the no pattern condition.

H_{p2}: Perceptions of design collective efficacy will mediate the positive relationship between pattern availability and design task performance.

H_{p3}: Pairs in the pattern condition will exhibit significantly higher communication quality and frequency than pairs in the no pattern condition.

H_{p4}: Perceptions of the communication quality and frequency will mediate the positive relationship between pattern availability and design task performance.

Tests of these hypotheses were done quite similar to the test for mediation that was done for the individual condition. The following table 5.19 summarizes the results of these tests.

Table 5.19 Results for the Mediator Analysis – Pair Condition

Step	Dependent Variable	Independent Variables	t-value	Significance
Design collective-efficacy (DCE) as the mediating variable/factor: availability of pattern				
1	DCE	Pattern	2.206	0.019
2	Solution Quality	Pattern	1.080	0.146
3	Solution Quality	DCE Pattern	1.817 0.254	0.042 0.401
Communication quality (COMQ)/Communication frequency (COMF) as the mediating variable/factor: availability of pattern				
1	COMQ	Pattern	0.327	0.374
2	COMF	Pattern	-0.930	0.819
3	Solution Quality	Pattern	1.080	0.146
4	Solution Quality	COMQ Pattern	-0.930 1.139	0.819 0.134
5	Solution Quality	COMF Pattern	-0.028 1.030	0.511 0.158

From the results of this analysis, it can be inferred that except for hypothesis H_{p1} none of the other three hypotheses are supported. Hence the argument of design collective efficacy, and communication quality to be the mediator could not be supported too. Availability of design patterns had a positive effect on the perceptions of design collective-efficacy. Solution quality was not statistically different between the pattern condition and no pattern condition. This meant the subsequent analyzes for design collective efficacy and communication quality as mediators for patterns and solution quality is not going find support Table 5.19 above also shows that the design self-efficacy is significant at higher alpha level of 0.05 when it is added as a mediator in the model with patterns.

Subsequent regression analysis revealed that design collective-efficacy itself is significant predictor of solution quality (t=2.174, p=0.021). This study measured design self-efficacy individually and then arrived at the aggregated design collective efficacy for the pair. Subsequent analysis also indicated that the design self-efficacy in the pair condition was significantly higher than the design self-efficacy in the individual condition (F = 5.166 and p = 0.025).

5.5 Summary of Results

The following table 5.20 summarizes the findings of this study. As can be seen few of the hypotheses were supported. The next chapter concludes this research with discussion on the findings and presents future research directions.

Table 5.20 Hypotheses Test Results

Hypothesis	Result
Effect of mode of participation on performance	
H_{1a}: Performance in terms of solution quality of the best individual in a nominal pair will be better than the collaborating pair's performance.	Not Supported
H_{1b}: Performance in terms of solution quality of collaborating pairs will be better than the second best individual in nominal pairs.	Supported
H_2: Collaborating pairs will take longer to complete the design task than the average nominal pair.	Supported
Effect of codified knowledge on performance	
H_3: Solution quality will be higher in the design pattern condition than in the condition without design pattern.	Supported
Effect of codified knowledge on subjective mental workload	
H_5: Subjective mental work load will be lower in the design pattern condition than in the condition without design pattern.	Not supported
Effect of mode of participation on overall task satisfaction	
H_6: Satisfaction with the task performance will be higher in the pair condition than in the individual condition.	Supported

Table 5.20 - continued

Hypothesis	Result
Individual Mediator Model – Design Self-efficacy	
H_{i1}: Individuals in the pattern condition will exhibit significantly higher design self-efficacy than individuals in no pattern condition.	Not Supported
H_{i2}: Perceptions of design self-efficacy will mediate the positive relationship between pattern availability and design task performance.	Not Supported
Pair Mediator Model – Design Collective-efficacy/Communication	
H_{p1}: Pairs in the pattern condition will exhibit significantly higher collective efficacy than pairs in the no pattern condition.	Supported
H_{p2}: Perceptions of design collective efficacy will mediate the positive relationship between pattern availability and design task performance.	Not Supported
H_{p3}: Pairs in the pattern condition will exhibit significantly higher communication quality than pairs in the no pattern condition.	Not supported
H_{p4}: Perceptions of the communication quality will mediate the positive relationship between pattern availability and design task performance.	Not Supported

Next chapter discusses the findings of the study and presents future research directions.

CHAPTER 6

DISCUSSION AND CONCLUSIONS

Development of software application is a complex enterprise. High failure rates of software development projects have been documented (Standish Group, 2001). Academicians and practitioners are spearheading the efforts to find better ways of developing software. Some of the initiatives in this area are: (a) streamlining the software development processes and (b) reusing existing knowledge of prior design experiences in the design of software applications.

(a) Software development process improvement: Extreme programming, an agile software development methodology, claims to increase the success rate of software development projects. Popularity of these methodologies can be seen in a recent survey that claimed wide acceptance of these development methodologies (Ambler, 2006). Extreme programming calls for the use of pairs in various activities related to software development. Some of the benefits attributed to pair programming are reduced errors in tasks performed, knowledge transfer between pairs, and increased satisfaction.

There exists anecdotal evidence that the use of pairs has a positive effect on task performance. Researchers have looked at the perceived benefits of pair-programming to a great extent. Many of the experimental studies utilized students as subjects. The

effectiveness of pairs in designing software application is yet to be rigorously researched in the academic community. This study borrowed concepts from the small group research area to analyze the effectiveness of pairs in comparison to individuals.

(b) Reuse of existing knowledge: Design patterns provide standard solutions to recurring software design problems. Various benefits are claimed due to the reuse of existing knowledge through design patterns. Some of these benefits include: improved communication through the use of standard vocabulary, improved design documentation, and creation of adaptable software solutions.

There are many different kinds of design patterns in use. Of these, the design patterns elucidated by Gamma et al.(1995) are more popular. Modern programming languages, such as Sun's Java programming language, utilize these patterns extensively. Software development tools used in modeling also provide built in support for these design patterns.

Though design patterns have been in vogue for a long time, few studies have analyzed the role of design patterns on solution quality. This study has borrowed concepts from knowledge transfer literature to analyze the effectiveness of design patterns.

This research studied two important aspects of software application development through a controlled experiment. First was the role of pairs in the design of software applications. Second was the role of codified knowledge in the design of software application. Apart from these major aspects, this study also addressed the role of efficacy beliefs and communication quality on task performance.

6.1 Summary of Research Findings

The following paragraphs summarize the results of this study.

6.1.1 Solution Quality

This study found that while working on a design task, collaborating pairs performed better than the second best performing individual in a nominal pair. As argued, the best individuals performed slightly better than the collaborating pairs but the differences were not statistically significant. Moreover, post-hoc tests revealed that the performance in terms of solution quality of collaborating pairs was slightly higher than the performance of average nominal pair though the performance difference was not statistically significant.

Findings from small groups research indicate, that the performance of groups will not be better than the best performing individual, but is likely to be better than the average individual's performance (Hare, 1994). Under certain conditions, such as when the solution demonstrability is high and the group size is larger than two, groups are claimed to outperform at the level of best individuals in a nominal pair (Laughlin, et al., 2006). Findings from this study are consistent with the research findings in the small group area.

This study also analyzed the role of design patterns on task performance. Research aimed at empirically evaluating the role of design patterns on solution quality is sparse. This study argued that design patterns are a form of codified knowledge and

that their use would positively affect solution quality. Study results indicated that the availability of design patterns significantly affect solution quality.

6.1.2 Completion Time

Findings of this study indicated that collaborating pairs utilized significantly higher amount of time to arrive at the solution compared to the average time taken by individuals. This finding is consistent with the findings in small groups research. In contrast, proponents of pair programming often cited research done on pair programming to show that the collaborating pairs took less time to complete the task than the individuals (Williams, et al., 2000). Findings of this study, however, are consistent with research in the small group area where collaborating groups were found to take more time to complete the tasks.

6.1.3 Subjective Mental Workload

Research in the past analyzed the mental workload during the performance of computer programming tasks. Subjective mental work load (SMWL) was measured using an instrument developed by NASA, and adapted to this particular study. Subjective mental work load was measured at the individual level. As a result of this, two separate values of SMWL pertaining to both the individuals in a collaborating pair condition were available.

This study hypothesized that the subjective mental workload would be lower for the pattern condition than in the condition without design patterns. Results did not support this contention. Analysis also revealed that the mode of participation, pairs versus individual, also had no impact on the SMWL.

Subsequent power analysis revealed the effect size was very small (0.001) and the power level (5.5%) was inadequate to find any significant relationships between the treatment conditions. Because of this, it is difficult to draw any conclusion based on the current study. Obviously a higher sample size would be helpful in obtaining a better perspective on the role of design patterns on subjective mental work load.

6.1.4 Overall Task Satisfaction

This study hypothesized that the members of a collaborating pair is likely to have a higher level of overall task satisfaction compared to individuals working alone. Analysis of the data supported this contention. Collaborating pairs were found to experience significantly higher overall task satisfaction than individuals. This result is consistent with prior findings in the area of small group research. It is also inline with arguments presented by the proponents of extreme programming who claim that pair programming leads to increased task satisfaction. Research in the Information Systems field on pair programming also indicated that pair programming was more enjoyable (Nosek, 1998).

Further analysis also revealed that subjects in the pattern condition also had significantly higher overall task satisfaction compared to those who were in the no design pattern condition. Though this relationship was not hypothesized in this study, nevertheless it is an interesting finding. Further research on the availability of codified knowledge and the overall task satisfaction could be of great importance in the field of knowledge management.

6.1.5 Individual Condition - Self-efficacy as a Mediator

Data collected in the individual condition was analyzed separately to test the mediating role of design self-efficacy. This study hypothesized that design self-efficacy was a mediator in the relationship between the availability of design patterns and performance. Results of the analysis revealed no relationship between the availability of design patterns and the efficacy beliefs of the subjects. Because of this the hypothesis pertaining to the mediating role of design self-efficacy was not supported.

Efficacy beliefs were not affected by the presence of design patterns and this could be due to various reasons. For routine tasks, self-efficacy beliefs are formed by individuals based on their past experience (Gist and Mitchell, 1992). Subjects of this study were experienced software development professionals. Hence these subjects could have formed the efficacy beliefs without carefully evaluating the additional resources that were made available to them in the pattern condition.

Further analysis indicated that self-efficacy, instead of being a mediator, was a predictor of task performance. This finding is interesting as hitherto the concept of self-efficacy, in the Information Research area, is primarily studied in the context of computer usage and in the context of software training (Compeau and Higgins, 1995; Yi and Davis, 2003).

6.1.6 Pair Condition - Collective-efficacy as a Mediator

This study measured individual self-efficacy and based on that it computed the design collective efficacy of pairs. In total, the sample size used for this analysis was 24. This study hypothesized that the design collective efficacy was a mediator in the

relationship between the availability of design patterns and performance. Presence of design patterns was found to significantly influence the efficacy beliefs of pairs. Availability of design patterns was not found to influence the solution quality. Because of this the hypothesis pertaining to the mediating role of design collective-efficacy was not supported.

This finding was also quite contrary to the findings in the individual condition. In the individual condition, presence of design patterns did not have an impact on the efficacy beliefs and whereas in the pair condition, there was an impact. Moreover, availability of design patterns did not have an influence on the solution quality and this result was contrary to the findings in the individual condition where the design patterns had an impact on solution quality. Reason for this discrepancy could be in the pair condition individuals could have done a more detailed evaluation of their resources in arriving at the efficacy beliefs and whereas it was not done in the individual condition. An evidence for this could be the significantly higher levels of design self-efficacy prevalent among the subjects in the pair condition in comparison to the subjects in the individual condition. In the pattern condition, subjects participated in the design pattern seminar before they performed the task. Members of the pairs in the pattern condition could have believed that their partner must be capable of utilizing the design patterns during task performance. This could have lead to higher design self-efficacy for the subjects in the pattern condition.

There could be many reasons for the lack of significant relationship between availability of design patterns and solution quality. The power level for this comparison

was quite low (18%). The mean solution quality score in the pattern condition was 44.96 with standard deviation of 11.83 and whereas in condition without design pattern the mean solution quality score was 39.25 with standard deviation of 13.97. An increased sample size could have given a better picture of the influence of design patterns on task performance in the pair condition.

Further analysis indicated that design collective-efficacy, instead of being a mediator, was a predictor of task performance. Further analysis also indicated subjects who performed in the collaborating pair condition to have significantly higher levels of self-efficacy in comparison to subjects in the individual condition. This finding is interesting because higher levels of design self-efficacy beliefs leads to better task performance, and could potentially make a case for pair-programming.

6.1.7 Pair Condition – Communication Quality as a Mediator

This study hypothesized that the quality of communication in the pair conditions to mediate the relationship between the availability of patterns and solution quality. Availability of design patterns was found to have no influence on the solution quality. Hence, it was decided not to proceed with further analysis on the mediating role of communication quality.

As mentioned earlier, the effect size of design pattern availability on task performance was very small (0.047) and the resulting power level was (18%). Two aspects of communication were analyzed in this study: communication frequency and communication quality. Design patterns did not have an impact on either of these constructs. Here again the power levels for the statistical tests were low (14% and 6%

respectively). The sample size used in this analysis was 24 and a higher sample size could have revealed a better picture.

6.2 Significance of the Findings

This study made important contributions to the stream of research aimed at evaluating the usefulness of pairs in software development activities. Apart from this primary objective, this study also evaluated the role of design patterns in the design of software applications. This study analyzed potential mediators and process variables in the performance of tasks by individuals and pairs.

6.2.1 Significance of Findings for Research

Though programming in pairs has been studied in the past, this is the first major study to evaluate the performance of pairs in a software design task. This study also utilized professional developers from various software development organizations and therefore, the study's findings are more generalizable. This study differed from many of the studies that were done in pair programming wherein the comparison was between equal numbers of individuals and equal number of pairs. Pair superiority in those circumstances was established due to the greater number of people working on the task. Probability of a pair having somebody with the knowledge to perform the task is higher in those circumstances. This study used nominal pairs, a more rigorous way of comparing pairs with individuals on task performance. In this study equal number of subjects performed the tasks in all the four conditions. Some of the significant outcomes of this research are presented here.

First, this study found that the collaborating pair's performance in terms of software quality was significantly better than the performance of the second best individual in a nominal pair. Though statistically not significant, the performance of the collaborating pairs was also found to be somewhat better than the performance of the average individual's performance. These findings are consistent with the research on small groups.

Second, this study found collaborating pairs to take more time to finish the design task compared to the average time taken by individuals. Again, this finding is quite consistent with the research on small groups.

Third, use of design patterns was found to significantly influence the quality of design solution. This is study has made a beginning in ascertaining the effectiveness of design patterns.

Fourth, collaborating pairs had more overall task satisfaction than the subjects who worked alone. This finding is quite consistent with the research done in small groups as well as in pair-programming. Likewise availability of codified knowledge in the form of design patterns made the participants more satisfied.

Fifth, design self-efficacy was found to be an important predictor of object-oriented design task performance. Though design self-efficacy was conceptualized as a mediator between the availability of design patterns and solution quality, results indicated it to be a predicator in both the individual and collaborating pair conditions.

6.2.2 Significance of Findings for Practitioners

Extreme programming and associated agile software development methodologies are gaining increasing popularity in the industry (Ambler, 2006). Likewise systems that promote the reuse of software artifacts and knowledge are increasingly becoming popular in the industry. Hence, the findings of this study are significant to practitioners in many ways.

First, collaborating pairs find their overall task satisfaction to be higher than individuals working alone. This benefit with collaborative work can be used as one of the major reason for the adoption of extreme programming.

Second, this study found that the best individuals will perform somewhat better than the collaborating pairs though statistically they were not significantly different. Hiring exceptional developers may not always be feasible in organizations. Hence pairing individuals during software development helps organizations to get near the quality levels of best individuals.

Third, this study has found collaborative pairs took more time to complete the design task than the average time taken by individuals. Hence organizations should take this into consideration while scheduling projects using extreme programming practices.

Fourth, findings of this study have reaffirmed the case for knowledge codification strategies. Tools that have built-in facilities for the exploitation of codified knowledge are likely to benefit the software engineering community. For example, visual modeling tools that can apply design patterns directly while designing the system will be of great help.

Fifth, this study also found efficacy beliefs about software design to be an important predictor of performance. Research on social cognitive theory indicates that efficacy beliefs can be enhanced through behavioral modeling (Gist and Mitchell, 1992). Organizations may initiate programs that increase the efficacy beliefs of software designers. This is likely to enhance the quality of software designs.

6.3 Limitations of the Study

There are some potential limitations present in this study and the conclusions should be viewed in light of them. First, though this study utilized software development professionals as subjects it is still a laboratory experiment. Because of this some important situational variables that may be present in an actual workplace, could be absent in the laboratory environment. However, laboratory experiment provides the necessary control to make causal inferences of the phenomena studied.

Second, the sample size used to test some of the hypotheses was small. This obviously resulted in insufficient power, as revealed in the power analysis, to unravel some hypothesized relationships. A larger sample might show support for some of these hypotheses that did not find support in this study.

The third limitation pertains to the way pairs are made up in the study. Though this study utilized software development professionals as subjects, the way the pairs were formed was different from ongoing pairs in any organization. Pairs in this study were strangers. They have very little opportunity to interact with each other before performing the experimental task. Past studies have indicated that the performance of pairs to improve with sufficient amount jelling period between the pairs (Williams, et

al., 2000). This study provided 20 minutes for the warm-up task, and for the members of the pairs to jell. There may be differences in the performance of pairs with increased time for jelling.

6.4 Future Research Directions

This is one of the early studies of software design investigating the phenomenon involving pairs. There are ample opportunities to expand on the current findings.

First, a longitudinal study can be done to see whether the performance of pairs improves on a task that is performed subsequent to the main task. Longitudinal studies can help to unravel any learning effects that may be present during group work.

Second, this study randomly paired individuals to arrive at the collaborating pairs. Individual ability/knowledge levels were not taken in to account while pairing members. It will be interesting to see how the effectiveness of pair varies with the differences between the knowledge levels of the pairs.

Third, this study analyzed the use of design patterns as a form of codified knowledge. There could be other factors that could be salient in the utilization of codified knowledge and those could be explored. For instance, this study provided 4 design patterns to the subjects during the task performance. Research could be done to see the effect of amount of codified knowledge (i.e) the number of design patterns on the utilization of the same.

Fourth, design self/collective efficacy was found be an important predictor in both the individual and pair condition. Research is needed in this area to further analyze this relationship between self-efficacy and software design performance.

Fifth, this study utilized a single design task across all the treatment conditions. Characteristics of the task impact the performance of groups. Further studies with varying task characteristics can shed more light in to the effectiveness of groups.

6.5 Conclusions

This study empirically examined some of the important concepts in software engineering. This is one of the early studies of software design investigating the phenomenon involving pairs and design pattern. This experimental study utilized software development professionals as subjects and manipulated the mode of participation (individual or pairs) and availability of codified knowledge (design patterns). Results of the study indicate that the performance of collaborating pairs were better than the 2nd best individual in nominal pairs. Collaborating pairs also took more time than the average time taken by nominal pairs to complete the task and they were more satisfied than the individuals. This study also found that the codified knowledge in the form of design patterns helped in arriving at a better solution. Findings of the study have great implications to both the research as well as the practitioner communities.

APPENDIX A

SUBJECT RECRUITMENT FLYER

Subject Recruitment Flyer

Free Seminar on Object-Oriented Design with Patterns

Dr. Radha Mahapatra and Dr. Sridhar Nerur of the University of Texas at Arlington are conducting a study to understand the role of Design Patterns in object-oriented design. We are currently recruiting volunteers to participate in this study.

Participants will receive **a free** seminar on Design Patterns covering the following topics:

- What are design patterns
- Classification of design patterns (GOF's)
- How to use design patterns in designing OO applications
- Design patterns and effective OO design

In addition they will also receive supplementary resources on design patterns and their usage. Study participants will also work individually or collectively to solve one or more software design problems and will fill out a questionnaire.

Who is eligible to participate?

If you are experienced in OO development but have not used design patterns at work and want to gain a broader understanding of design patterns then this seminar is right for you.

When and where?

These seminars are currently being held in the Business Building of UT Arlington. You can register for this seminar at: http://www3.uta.edu/faculty/mangalaraj/seminar.html

For more information please contact George Mangalaraj at mangalaraj@uta.edu

Or call 817-272-3562

APPENDIX B

INFORMED CONSENT

INFORMED CONSENT

Software Design Experiment

In this study you will be asked to work on a software design problem using object-oriented design principles. You may be working individually or with another partner. You will work initially on a warm-up task before working on the main design task. The total duration of the experimental session will be approximately two hours. Problem statements will be provided to you with any additional materials required for solving the problems. You will also be asked to complete a questionnaire about your reactions to working on the task.

Since you may be working with another person you may experience emotional discomfort, similar to what you could experience in the work place when working on tasks of this nature. Those discomforts may include fatigue, boredom or frustration when you work with other people to solve design problems.

The major benefit you will get in participating in this study is to get an understanding of modern software design practices that may be of great value. You will also be presented with a certificate of participation in the "Design Pattern" seminar. You will be debriefed immediately after the experiment. The benefits to the investigator are increased understandings of pair working on design problems and the usefulness of design techniques involving software design patterns.

Your participation in this experiment is voluntary and you can withdraw your informed consent at any time, if you find any procedures objectionable. Records of your participation and any data collected will be held in strict confidence.

This research study has been reviewed and approved by the University of Texas at Arlington Institutional Review Board. This research is under the supervision of Dr. Radha Mahapatra. Dr. Mahapatra's office is in 521 of the Business Building and his phone number is (817) 272-3590. Please feel free to contact him if you have questions. If you have any questions about your rights as a subject or about a research related injury, you may contact a member of the Office or Research Compliance at 817-272-3723.

I had a chance to ask all questions regarding this study. I hereby consent to participate in the experiment and understand the above procedures.

Signature: …….. Print Name: …….

Date: …….

APPENDIX C

DEBRIEFING

Debriefing

In this study we are interested in examining the effectiveness of working individually versus working in pairs while performing design tasks. We are also interested in studying the effect of codified knowledge on the effectiveness of pair versus individuals in performing the design task. Pairs are increasingly used in software development projects for various tasks and they are claimed to increase the quality and productivity. Similarly, design patterns are claimed to provide codified form of knowledge and increase the software design quality. Our specific interest in this study is to see whether there were differences in the quality of solution, and productivity due to the use of pairs/individuals and with design patterns or without it. We are also interested in the work load, efficacy beliefs, task communication, and satisfaction when tasks were done individual or pairs with or without design patterns.

In studying this we randomly assigned members to individual or pair condition, and with or without design pattern conditions. Quality of your design and time taken to solve the design problem will be used to judge the effect of various factors on the effectiveness of pair/individuals and with/without design patterns in performing software design tasks.

We will be conducting this study with more professionals from your organization. It is vitally important to us and the success of this experiment that you keep the information that you have learned here in confidence. Please do not tell anyone about this experiment. We are confident that we can trust you and thank you.

This research is under the supervision of Dr. Radha Mahapatra and you can contact him at his office which is in room 521 of the Business Building and his phone is (817) 272-3590. If you are interested in knowing the results of this study, then please feel free to contact him after 5 weeks

Please sign below to indicate that you understand this debriefing and that you promise to keep what you have learned in confidence. Again, thank you very much for your participation.

Signature: …….. Print Name: …….

Date: …….

APPENDIX D

INSTRUCTIONS FOR VARIOUS TREATMENTS

Individual – No Pattern Condition

Task Instructions

Today you will work on two software design problems. The first one is a practice task and the second one is the actual experimental task. Duration for the practice task is 20 minutes and the duration of the experimental task is of 80 minutes. Relevant problem descriptions are given in separate sheets.

- You are required to come up with an object-oriented solution
- Your solution should strive for flexibility of design. Try to adhere to good OO design principles.
- The deliverable is a class diagram using the UML notations. You may refer to the summary of relevant UML notations given in the handout.
- Your class diagram should show the classes and the relationships between them.
- Please present your solution in the space provided for it.
- You are welcome to use the scratch papers for preliminary solutions.
- Briefly describe the approach you adopted to identify the classes and their relationships. In other words, what was the rationale for your design?

Basic documentation of UML notations and glossary of terms used in OO design are provided in your packet. While you may talk your ideas loud, you may not converse with any other person during the task performance. You are requested to complete the questionnaire and turn it in with the solution.

Individual – Pattern Condition

Task Instructions

Today you will work on two software design problems. The first one is a practice task and the second one is the actual experimental task. Duration for the practice task is 20 minutes and the duration of the experimental task is of 80 minutes. Relevant problem descriptions are given in separate sheets.

- You are required to come up with an object-oriented solution
- Your solution should strive for flexibility of design. Try to adhere to good OO design principles.
- The deliverable is a class diagram using the UML notations. You may refer to the summary of relevant UML notations given in the handout.
- You are encouraged to refer to the handout on design patterns and use them in your solution.
- Your class diagram should show the classes and the relationships between them.
- Please present your solution in the space provided for it.
- You are welcome to use the scratch papers for preliminary solutions.
- Briefly describe the approach you adopted to identify the classes and their relationships. In other words, what was the rationale for your design?

Basic documentation of UML notations and glossary of terms used in OO design are provided. Catalog of design patterns are also provided in the handout. While you may talk your ideas loud, you may not converse with any other person during the task performance. You are requested to complete the questionnaire and turn it in with the solution.

Pair – No Pattern Condition

Task Instructions

Today you will work on two software design problems collaboratively with another partner. The first one is a practice task and the second one is the actual experimental task. Duration for the practice task is 20 minutes and the duration of the experimental task is of 80 minutes. Relevant problem descriptions are given in separate sheets.

- You are required to come up with an object-oriented solution
- Your solution should strive for flexibility of design. Try to adhere to good OO design principles.
- The deliverable is a class diagram using the UML notations. You may refer to the summary of relevant UML notations given in the handout.
- Your class diagram should show the classes and the relationships between them.
- Please present your solution in the space provided for it.
- You are welcome to use the scratch papers for preliminary solutions.
- Briefly describe the approach you adopted to identify the classes and their relationships. In other words, what was the rationale for your design?

You may actively converse with each other to discuss the tasks at hand. You are encouraged to collaboratively arrive at a consensual solution. Basic documentation of UML notations and glossary of terms used in OO design are provided in your packet. Please work together to arrive at a single solution that both of you agree to. You are requested to complete the questionnaires and turn it in with the solution.

Task Instructions

Today you will work on two software design problems collaboratively with another partner. The first one is a practice task and the second one is the actual experimental task. Duration for the practice task is 20 minutes and the duration of the experimental task is of 80 minutes. Relevant problem descriptions are given in separate sheets.

- You are required to come up with an object-oriented solution
- Your solution should strive for flexibility of design. Try to adhere to good OO design principles.
- The deliverable is a class diagram using the UML notations. You may refer to the summary of relevant UML notations given in the handout.
- You are encouraged to refer to the handout on design patterns and use them in your solution.
- Your class diagram should show the classes and the relationships between them.
- Please present your solution in the space provided for it.
- You are welcome to use the scratch papers for preliminary solutions.
- Briefly describe the approach you adopted to identify the classes and their relationships. In other words, what was the rationale for your design?

Basic documentation of UML notations and glossary of terms used in OO design are provided in your packet. Catalog of design patterns are also provided in the handout. Please work together to arrive at a single solution that both of you agree to. You are requested to complete the questionnaires and turn it in with the solution. You are requested to complete the questionnaires and turn it in with the solution.

APPENDIX E

EXPERIMENTAL TASKS

Warm-up Task

Time: 20 minutes

Duck Pond Simulation Game*

XYZ Corporation is in the process of designing a duck pond simulation game. The game can display a large variety of duck species swimming and making quacking sounds. Some of the duck types that are planned to be used in the simulation game include: Mallard Duck, Redhead Duck, Bufflehead Duck, and Pintail Duck. They are also planning to add Rubber Duck to the game. Real ducks are capable of quacking and flying whereas the rubber ducks can only squeak and cannot fly. All the ducks have a behavior to display themselves on the screen.

You are required to draw a class diagram that can be used to implement the system described above.

* Note: This problem description is adapted from the one found in Freeman, E., Freeman, E., Sierra, K., and Bates, B. *Head First design patterns*, O'Reilly, Sebastopol, CA, 2004.

Main Task

Maximum Time: 80 minutes

Weather Monitor[†]

You must develop software that allows clients to periodically check for changes in the status of sensors in a weather monitoring station. Example of various sensors could be temperature gauge, pressure gauge, humidity sensor, etc. A client must be able to create a monitor that will periodically check a particular sensor in the network. If the state of that sensor has changed since the monitor last checked the sensor, the monitor should write an event to a global event log.

A sensor is uniquely designated by its sensor id. When making its initial monitoring request, the client specifies the id of the sensor to be monitored and the monitoring period. At that point, the monitor is initialized but has not yet been started. The client makes a subsequent request to start the monitor. The client should be able to start and stop the monitor at any time.

Each sensor has an interface to check its current state, although the precise interface differs from sensor to sensor. As an example, to check a temperature sensor, you invoke its getTemparature method, whereas to check a relative humidity sensor, you call its getRh method. The various sensor classes are provided by different vendors, so you are not permitted to change the interfaces of those classes.

The components that make up the states of different sensors may also differ. For example, the state of a temperature sensor is defined by a floating point value. A relative humidity sensor state, on the other hand, is represented by a string.

Assume the existence of an Event Log and an Event class. The Event Log classes define a method, logEvent that takes an Event as an argument and places that Event in the Log. You should write a specific type of Event, a Sensor Change Event that includes the id of the sensor.

You are required to draw a class diagram that can be used to implement the system described above.

[†] Note: This problem description is adapted from the one found in Richter, C. "Design Problems and Object-Oriented Solutions," Objective Engineering, Inc.,http://www.oeng.com/problemsandsolutions.htm, 2004, Last Accessed: 6th December, 2006

APPENDIX F

QUESTIONNAIRE

Individuals – No Patterns

Demographics and Background Questions

1. Please circle your gender:

   Male Female

2. Please indicate your age on your last birthday _____

3. Highest educational level:

   a) High school degree b) Technical school or community college c) Undergraduate

   d) Graduate degree d) Doctoral degree e) Others _____

4. Indicate number of years of your programming experience in any programming language?

   a) 0 – 1 b) 1 – 2 c) 2-4 d) 4-6 e) >6

5. Indicate number of years of your programming experience in object-oriented languages?

   a) 0 – 1 b) 1 – 2 c) 2-4 d) 4-6 e) >6

6. What would you consider to be your level of experience in object-oriented design?

   a) No experience b) Novice c) Intermediate d) Expert

7. What would you consider to be your level of experience in design patterns?

   a) No experience b) Novice c) Intermediate d) Expert

8. What object-orient programming languages are you familiar with?

   a) C++ b) C# c) Java d) Small Talk e) Objective-C

   f) Eiffel g) Python h) VB.NET i) Others _____

---

1. I have confidence in my ability to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2. I have the necessary knowledge to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3. This design problem is well within the scope of my abilities

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4. I do not anticipate any problems in doing the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

5. I have the necessary expertise/resources to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

6. My past experiences and accomplishments increase my confidence in solving the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

Task Load Index

We would like to know about the workload you experienced in performing this task. Feelings of workload can come from several different factors. For example, some people feel that mental or time demands are the most important factors in perceived workload. Others may feel that their performance or amount of frustration is the most important part of their feelings of workload.
Here are the definitions of factors that contribute to workload.

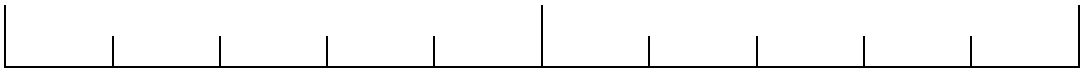
Factor	**Definition**
Mental Demand	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)?
Time Demand	How much time pressure did you feel due to the rate or place at which the tasks occurred?
Effort	How hard did you have to work (mentally) to accomplish your level of performance?
Frustration Level	How insecure, discouraged, irritated, stressed, and annoyed versus secure, gratified, content, relaxed, and complacent did you feel during the task?
Performance	How successful do you think you were in accomplishing the goals of the task?

For each pair of the 10 items listed below, select the item that represents the **more important** contributor to workload for the task you performed this session.

1	Effort □	-	Performance □
2	Time Demand □	-	Effort □
3	Performance □	-	Frustration □
4	Time Demand □	-	Frustration □
5	Time Demand □	-	Mental Demand □
6	Frustration □	-	Effort □
7	Performance □	-	Time Demand □
8	Frustration □	-	Mental Demand □
9	Performance □	-	Mental Demand □
10	Mental Demand □	-	Effort □

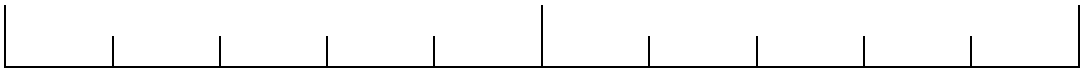
For questions 1 to 5, place an "X" on each scale at the point that matches your experience. Your ratings will play an important role in evaluation being conducted; therefore, your participation is greatly appreciated.

1. Mental Demand



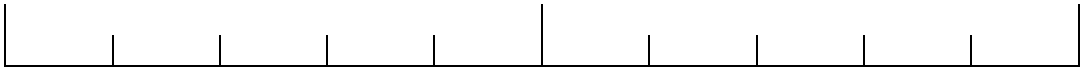
Low High

2. Time Demand



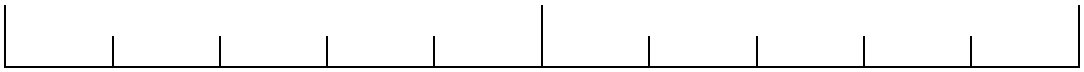
Low High

3. Performance



Good Poor

4. Effort



Low High

5. Frustration

Low High

How do you feel about your overall experience of working on the task today?

Very Dissatisfied Satisfied	1	2	3	4	5	6	7	Very
Very Displeased	1	2	3	4	5	6	7	Very Pleased
Very Frustrated Contended	1	2	3	4	5	6	7	Very
Absolutely Terrible Delighted	1	2	3	4	5	6	7	Absolutely

---

1. How do you feel about the main design task you performed today?

Very Easy	1	2	3	4	5	6	7	Very Difficult
Very Simple	1	2	3	4	5	6	7	Very Complex

2. How do you feel about the main design task you performed, as compared to the warm up task?

Very Easy	1	2	3	4	5	6	7	Very Difficult
Very Simple	1	2	3	4	5	6	7	Very Complex

Briefly describe the approach you adopted to identify the classes and their relationships. In other words, what was the rationale for your design?

Did you use any pattern? If so could you use specify the patterns you used?

Individuals – Patterns

Demographics and Background Questions

1. Please circle your gender:

   Male Female

2. Please indicate your age on your last birthday _____

3. Highest educational level:

   a) High school b) Technical school or community college c) Undergraduate degree

   d) Graduate degree d) Doctoral degree e) Others _____

4. Indicate number of years of your programming experience in any programming language?

   a) 0 – 1 b) 1 – 2 c) 2-4 d) 4-6 e) >6

5. Indicate number of years of your programming experience in object-oriented languages?

   a) 0 – 1 b) 1 – 2 c) 2-4 d) 4-6 e) >6

6. What would you consider to be your level of experience in object-oriented design?

   a) No experience b) Novice c) Intermediate d) Expert

7. What would you consider to be your level of experience in design patterns?

   a) No experience b) Novice c) Intermediate d) Expert

8. What object-orient programming languages are you familiar with?

   a) C++ b) C# c) Java d) Small Talk e) Objective-C

   f) Eiffel g) Python h) VB.NET i) Others _____

---

1. I have confidence in my ability to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2. I have the necessary knowledge to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3. This design problem is well within the scope of my abilities

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4. I do not anticipate any problems in doing the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

5. I have the necessary expertise/resources to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

6. My past experiences and accomplishments increase my confidence in solving the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

Task Load Index

We would like to know about the workload you experienced in performing this task. Feelings of workload can come from several different factors. For example, some people feel that mental or time demands are the most important factors in perceived workload. Others may feel that their performance or amount of frustration is the most important part of their feelings of workload.
Here are the definitions of factors that contribute to workload.

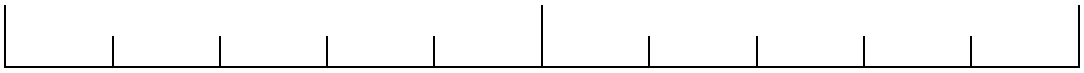
Factor	**Definition**
Mental Demand	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)?
Time Demand	How much time pressure did you feel due to the rate or place at which the tasks occurred?
Effort	How hard did you have to work (mentally) to accomplish your level of performance?
Frustration Level	How insecure, discouraged, irritated, stressed, and annoyed versus secure, gratified, content, relaxed, and complacent did you feel during the task?
Performance	How successful do you think you were in accomplishing the goals of the task?

For each pair of the 10 items listed below, select the item that represents the **more important** contributor to workload for the task you performed this session.

1	Effort □	-	Performance □
2	Time Demand □	-	Effort □
3	Performance □	-	Frustration □
4	Time Demand □	-	Frustration □
5	Time Demand □	-	Mental Demand □
6	Frustration □	-	Effort □
7	Performance □	-	Time Demand □
8	Frustration □	-	Mental Demand □
9	Performance □	-	Mental Demand □
10	Mental Demand □	-	Effort □

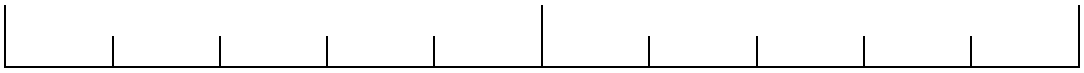
For questions 1 to 5, place an "X" on each scale at the point that matches your experience. Your ratings will play an important role in evaluation being conducted; therefore, your participation is greatly appreciated.

1. Mental Demand



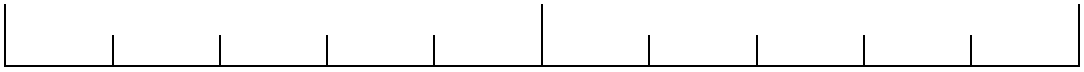
Low High

2. Time Demand



Low High

3. Performance



Good Poor

4. Effort

Low High

5. Frustration

Low High

How do you feel about your overall experience of working on the task today?

Very Dissatisfied	1	2	3	4	5	6	7	Very Satisfied
Very Displeased	1	2	3	4	5	6	7	Very Pleased
Very Frustrated	1	2	3	4	5	6	7	Very Contended
Absolutely Terrible	1	2	3	4	5	6	7	Absolutely Delighted

---

1. How do you feel about the main design task you performed today?

Very Easy	1	2	3	4	5	6	7	Very Difficult
Very Simple	1	2	3	4	5	6	7	Very Complex

2. How do you feel about the main design task you performed, as compared to the warm up task?

Very Easy	1	2	3	4	5	6	7	Very Difficult
Very Simple	1	2	3	4	5	6	7	Very Complex

---

1. Design patterns were easy to use for the tasks done

Strongly disagree	1	2	3	4	5	6	7	Strongly agree

2. Design patterns gives flexibility/extensibility to the design we did

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3. Please briefly state any problems in using design patterns for the tasks performed

4. In what ways did the design patterns help you in solving the problems today?

Briefly describe the approach you adopted to identify the classes and their relationships. In other words, what was the rationale for your design?

What patterns did you use to arrive at the solution?

Pairs – No Patterns

Demographics and Background Questions

1. Please circle your gender:

   Male Female

2. Please indicate your age on your last birthday _____

3. Highest educational level:

   a) High school b) Technical school or community college c) Undergraduate degree

   d) Graduate degree d) Doctoral degree e) Others _____

4. Indicate number of years of your programming experience in any programming language?

   a) 0 – 1 b) 1 – 2 c) 2-4 d) 4-6 e) >6

5. Indicate number of years of your programming experience in object-oriented languages?

   a) 0 – 1 b) 1 – 2 c) 2-4 d) 4-6 e) >6

6. What would you consider to be your level of experience in object-oriented design?

   a) No experience b) Novice c) Intermediate d) Expert

7. What would you consider to be your level of experience in design patterns?

   a) No experience b) Novice c) Intermediate d) Expert

8. What object-orient programming languages are you familiar with?

   a) C++ b) C# c) Java d) Small Talk e) Objective-C

   f) Eiffel g) Python h) VB.NET i) Others _____

9. Before today's task performance, have you ever worked with your partner?

   a) Yes b) No

1. I have confidence in my ability to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2. I have the necessary knowledge to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3. This design problem is well within the scope of my abilities

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4. I do not anticipate any problems in doing the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

5. I have the necessary expertise/resources to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

6. My past experiences and accomplishments increase my confidence in solving the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

Task Load Index

We would like to know about the workload you experienced in performing this task. Feelings of workload can come from several different factors. For example, some people feel that mental or time demands are the most important factors in perceived workload. Others may feel that their performance or amount of frustration is the most important part of their feelings of workload.

Here are the definitions of factors that contribute to workload.

Factor	**Definition**
Mental Demand	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)?
Time Demand	How much time pressure did you feel due to the rate or place at which the tasks occurred?
Effort	How hard did you have to work (mentally) to accomplish your level of performance?

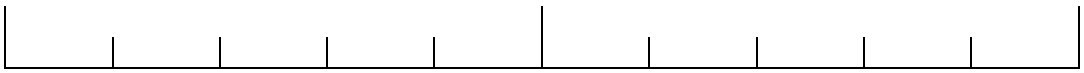
Frustration Level	How insecure, discouraged, irritated, stressed, and annoyed versus secure, gratified, content, relaxed, and complacent did you feel during the task?
Performance	How successful do you think you were in accomplishing the goals of the task?

For each pair of the 10 items listed below, select the item that represents the **more important** contributor to workload for the task you performed this session.

1	Effort □	-	Performance □
2	Time Demand □	-	Effort □
3	Performance □	-	Frustration □
4	Time Demand □	-	Frustration □
5	Time Demand □	-	Mental Demand □
6	Frustration □	-	Effort □
7	Performance □	-	Time Demand □
8	Frustration □	-	Mental Demand □
9	Performance □	-	Mental Demand □
10	Mental Demand □	-	Effort □

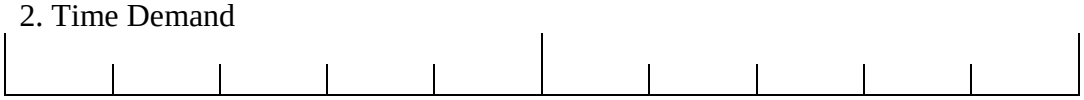
For questions 1 to 5, place an "X" on each scale at the point that matches your experience. Your ratings will play an important role in evaluation being conducted; therefore, your participation is greatly appreciated.

1. Mental Demand



Low High

2. Time Demand



Low High

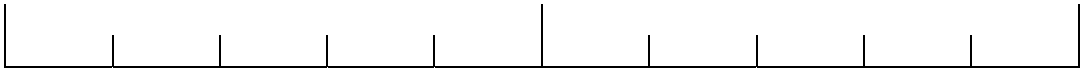
3. Performance

Good Poor

4. Effort

Low High

5. Frustration



Low High

---

1. We frequently communicated with each other while performing the task.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2. We frequently exchanged ideas about the problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3. We frequently exchanged ideas about solutions to the problem.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4. I had no difficulty understanding my partner's ideas about the task.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

5. Our discussion improved my understanding of the problem.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

6. Our discussions helped us craft a better solution to the problem.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

How do you feel about your overall experience of working on the task today?

Very Dissatisfied 1 2 3 4 5 6 7 Very Satisfied

Very Displeased	1	2	3	4	5	6	7	Very Pleased
Very Frustrated	1	2	3	4	5	6	7	Very Contended
Absolutely Terrible Delighted	1	2	3	4	5	6	7	Absolutely

---

1. How do you feel about the main design task you performed today?

Very Easy	1	2	3	4	5	6	7	Very Difficult
Very Simple	1	2	3	4	5	6	7	Very Complex

2. How do you feel about the main design task you performed, as compared to the warm up task?

Very Easy	1	2	3	4	5	6	7	Very Difficult
Very Simple	1	2	3	4	5	6	7	Very Complex

---

1. Overall, we could interrelate to each other's unique skills

Strongly disagree	1	2	3	4	5	6	7	Strongly agree

2. Overall, we could interrelate to each other's unique expertise

Strongly disagree	1	2	3	4	5	6	7	Strongly agree

3. I could recognize the potential value of my partner's expertise

Strongly disagree	1	2	3	4	5	6	7	Strongly agree

---

1. We explored multiple solutions for the given problems

Strongly disagree	1	2	3	4	5	6	7	Strongly agree

2. We helped each other in finding flaws with the design

Strongly disagree	1	2	3	4	5	6	7	Strongly agree

3. We could easily identify the right solutions to the problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4. My partner actively participated in solving the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

Briefly describe the approach you adopted to identify the classes and their relationships. In other words, what was the rationale for your design?

What patterns did you use to arrive at the solution?

Pairs – Patterns

Demographics and Background Questions

1. Please circle your gender:

   Male Female

2. Please indicate your age on your last birthday _____

3. Highest educational level:

   a) High school b) Technical school or community college c) Undergraduate degree

   d) Graduate degree d) Doctoral degree e) Others _____

4. Indicate number of years of your programming experience in any programming language?

   a) 0 – 1 b) 1 – 2 c) 2-4 d) 4-6 e) >6

5. Indicate number of years of your programming experience in object-oriented languages?

   a) 0 – 1 b) 1 – 2 c) 2-4 d) 4-6 e) >6

6. What would you consider to be your level of experience in object-oriented design?

   a) No experience b) Novice c) Intermediate d) Expert

7. What would you consider to be your level of experience in design patterns?

   a) No experience b) Novice c) Intermediate d) Expert

8. What object-orient programming languages are you familiar with?

   a) C++ b) C# c) Java d) Small Talk e) Objective-C

   f) Eiffel g) Python h) VB.NET i) Others _____

9. Before today's task performance, have you ever worked with your partner?

   a) Yes b) No

1. I have confidence in my ability to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2. I have the necessary knowledge to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3. This design problem is well within the scope of my abilities

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4. I do not anticipate any problems in doing the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

5. I have the necessary expertise/resources to solve this design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

6. My past experiences and accomplishments increase my confidence in solving the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

Task Load Index

We would like to know about the workload you experienced in performing this task. Feelings of workload can come from several different factors. For example, some people feel that mental or time demands are the most important factors in perceived workload. Others may feel that their performance or amount of frustration is the most important part of their feelings of workload.

Here are the definitions of factors that contribute to workload.

Factor	**Definition**
Mental Demand	How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)?
Time Demand	How much time pressure did you feel due to the rate or place at which the tasks occurred?
Effort	How hard did you have to work (mentally) to accomplish your level of performance?
Frustration	How insecure, discouraged, irritated, stressed, and annoyed versus

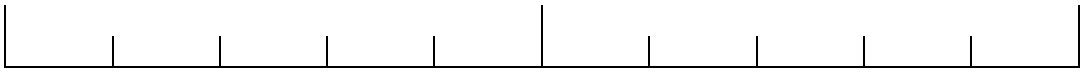
Level	secure, gratified, content, relaxed, and complacent did you feel during the task?
Performance	How successful do you think you were in accomplishing the goals of the task?

For each pair of the 10 items listed below, select the item that represents the **more important** contributor to workload for the task you performed this session.

1	Effort □	-	Performance □
2	Time Demand □	-	Effort □
3	Performance □	-	Frustration □
4	Time Demand □	-	Frustration □
5	Time Demand □	-	Mental Demand □
6	Frustration □	-	Effort □
7	Performance □	-	Time Demand □
8	Frustration □	-	Mental Demand □
9	Performance □	-	Mental Demand □
10	Mental Demand □	-	Effort □

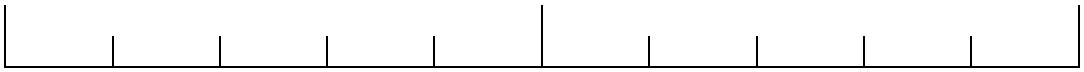
For questions 1 to 5, place an "X" on each scale at the point that matches your experience. Your ratings will play an important role in evaluation being conducted; therefore, your participation is greatly appreciated.

1. Mental Demand



Low High

2. Time Demand



Low High

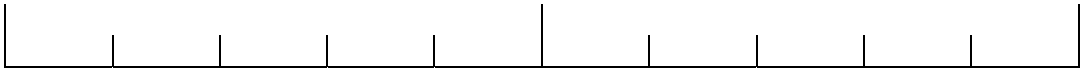
3. Performance

Good Poor

4. Effort

Low High

5. Frustration



Low High

---

1. We frequently communicated with each other while performing the task.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2. We frequently exchanged ideas about the problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3. We frequently exchanged ideas about solutions to the problem.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4. I had no difficulty understanding my partner's ideas about the task.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

5. Our discussion improved my understanding of the problem.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

6. Our discussions helped us craft a better solution to the problem.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

How do you feel about your overall experience of working on the task today?

Very Dissatisfied	1	2	3	4	5	6	7	Very Satisfied
Very Displeased	1	2	3	4	5	6	7	Very Pleased
Very Frustrated	1	2	3	4	5	6	7	Very Contended
Absolutely Terrible Delighted	1	2	3	4	5	6	7	Absolutely

---

1. How do you feel about the main design task you performed today?

Very Easy	1	2	3	4	5	6	7	Very Difficult
Very Simple	1	2	3	4	5	6	7	Very Complex

2. How do you feel about the main design task you performed, as compared to the warm up task?

Very Easy	1	2	3	4	5	6	7	Very Difficult
Very Simple	1	2	3	4	5	6	7	Very Complex

---

5. Design patterns were easy to use for the tasks done

Strongly disagree	1	2	3	4	5	6	7	Strongly agree

6. Design patterns gives flexibility/extensibility to the design we did

Strongly disagree	1	2	3	4	5	6	7	Strongly agree

7. Please briefly state any problems in using design patterns for the tasks performed

8. In what ways did the design patterns help you in solving the problems today?

---

4. Overall, we could interrelate to each other's unique skills

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

5. Overall, we could interrelate to each other's unique expertise

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

6. I could recognize the potential value of my partner's expertise

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

---

1. We explored multiple solutions for the given problems

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2. We helped each other in finding flaws with the design

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3. We could easily identify the right solutions to the problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4. My partner actively participated in solving the design problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

5. Presence of another person helped in better utilizing the design patterns.

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

Briefly describe the approach you adopted to identify the classes and their relationships. In other words, what was the rationale for your design?

What patterns did you use to arrive at the solution?

APPENDIX G

GRADING SCHEME FOR THE MAIN TASK

Grading Sheet for Weather Monitoring Station Problem

Subject ID:

No	Description	Purpose	Points	Points Scored	Remarks
A. Classes					
1	Monitor	Sensors are monitored by this class	5		
2	Sensor	Can be abstract or an interface	5		
3	Specific Sensor	Temperature, Pressure sensors etc	5		
4	Event log	Log of various events	5		
5	Event	Generic class	5		
6	Change Event	This is the sensor change event	5		
7	Adapter class	Adapter for the sensors to handle different states	10		
8	PollToken	To store state and compare	10		
B. Associations					
1	Monitor/Sensors Interface	Polltoken object is returned to monitor	5		
2	Event logging	Monitor detects changes when they occur, creates change event	10		
3	Specific sensor / PollToken	Specific sensors creating polltoken	5		
4	Comparing states	Monitor should be able to compare polltokens (previous state polltoken is the current state polltoken)	10		
6	Sensor Adapter/ Sensor	Sensor Adapters forward poll to specific sensors	10		
C. Optional Classes					
1					
2					
3					
D. Unnecessary/Wrong Classes					
1			-		
2			-		
3			-		
	Total				

Patterns Used:

Note: Class names are given for illustration only and the evaluated solution can have different names for them.

APPENDIX H

GLOSSARY GIVEN TO ALL PARTICIPANTS

Glossary[‡]

Abstract class: A class whose primary purpose is to define an interface. An abstract class defers some or all of its implementation to subclasses. An abstract class cannot be instantiated.

Abstract operation: An operation that declares a signature but doesn't implement it.

Class: A class defines an object's interface and implementation. It specifies the object's internal representation and defines the operations the object can perform.

Class diagram: A diagram that depicts classes, their internal structure and operations, and the static relationships between them.

Concrete class: A class having no abstract operations. It can be instantiated.

Delegation: An implementation mechanism in which an object forwards or delegates a request to another object. The delegate carries out the request on behalf of the original object.

Design pattern: A design pattern systematically names, motivates, and explains a general design that addresses a recurring design problem in object-oriented systems. It describes the problem, the solution, when to apply the solution, and its consequences. The solution is a general arrangement of objects and classes that solve the problem.

Encapsulation: The result of hiding a representation and implementation in an object. The representation is not visible and cannot be accessed directly from outside the object. Operations are the only way to access and modify an object's representation.

Inheritance: A relationship that defines one entity in terms of another. Class inheritance defines a new class in terms of one or more parent classes. The new class inherits its interface and its implementation from its parents. The new class is called a subclass or a derived class.

Instance variable: A piece of data that defines part of an object's representation.

Interface: The set of all signatures defined by an object's operations. The interface describes the set of requests to which an object can respond.

Object: A run-time entity that packages both data and the procedures that operates on that data.

Object composition: Assembling or composing objects to get more complex behavior.

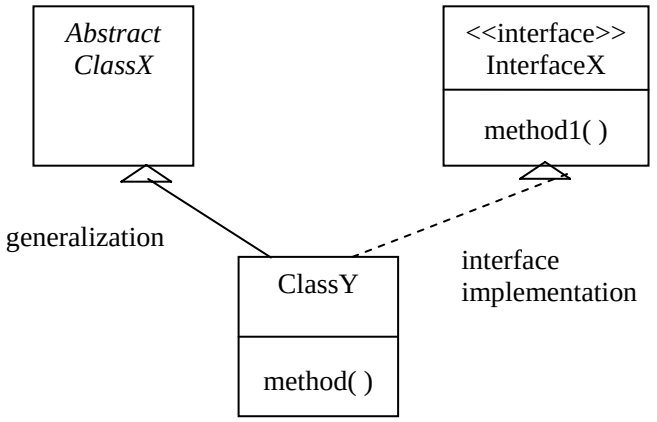
Object reference: A value that identifies another object.

Overriding: Redefining an operation in a subclass.

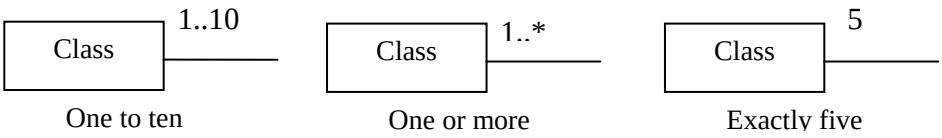
Polymorphism: The ability to substitute objects of matching interface for one another at run-time.

[‡] Note: These descriptions are based on Gamma, E., Helm, R., Johnson, R.E., and Vlissides, J. *Design patterns : elements of reusable object-oriented software*, Addison-Wesley, Reading, Mass., 1995.

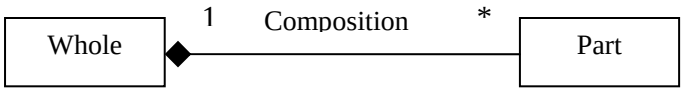
UML Notations for Class Diagram[§]



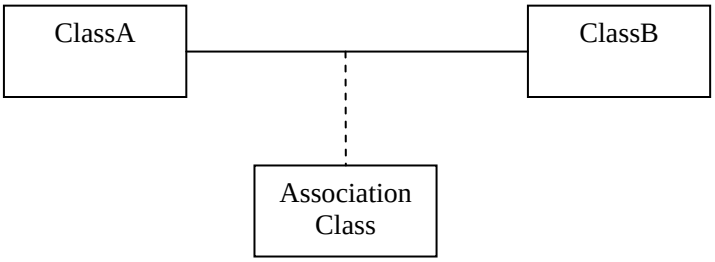
Multiplicity:



Composition:



Associations:



[§] Note: These UML notations are based on Larman, C. *Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development*, Prentice Hall PTR, Upper Saddle River, N.J., 2004.

APPENDIX I

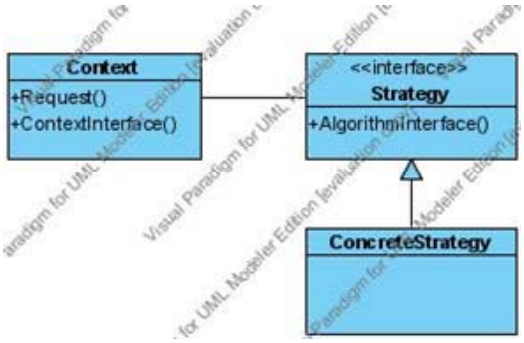
DESIGN PATTERNS MATERIALS FOR PATTERN CONDITION

The Strategy Pattern

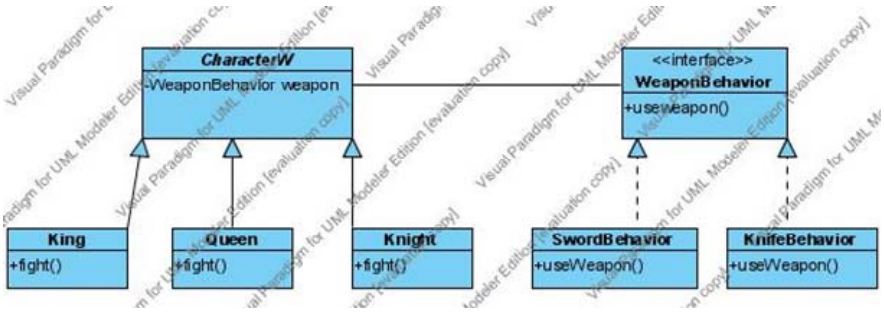
- It defines a family of algorithms, encapsulates each one, and makes them interchangeable.
- Strategy lets the algorithm vary independently from clients that use it.

[**] These patterns are based on the outlines presented in the following books:
(a) Gamma, E., Helm, R., Johnson, R.E., and Vlissides, J. *Design patterns : elements of reusable object-oriented software*, Addison-Wesley, Reading, Mass., 1995.,
(b) Freeman, E., Freeman, E., Sierra, K., and Bates, B. *Head First design patterns*, O'Reilly, Sebastopol, CA, 2004., and
(c) Martin, R.C. *Agile software development : principles, patterns, and practices*, Prentice Hall, Upper Saddle River, N.J., 2003.

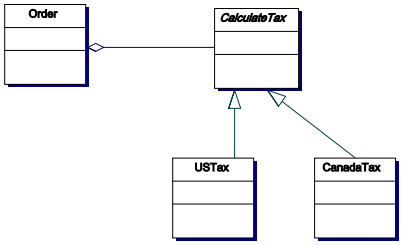
The Strategy Pattern



The Strategy Pattern - Example



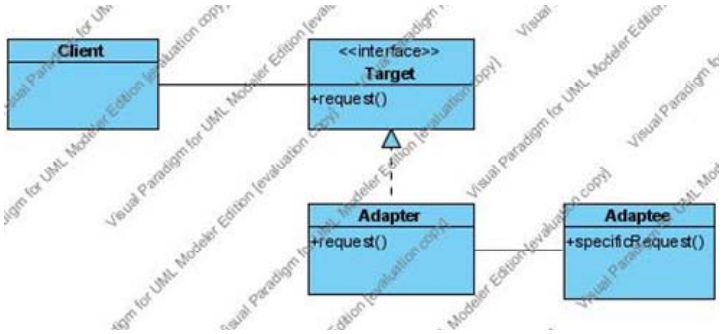
The Strategy Pattern



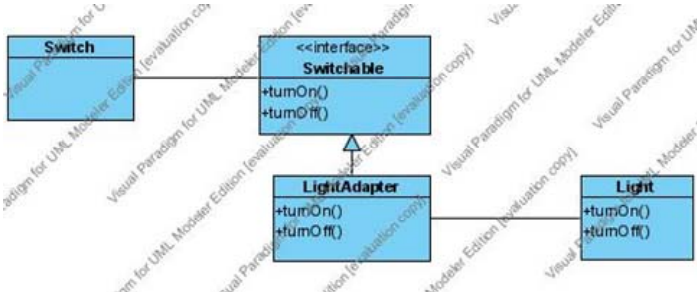
The Adapter Pattern

- It converts the interface of a class into another interface the clients expect.
- Adapter lets classes work together that couldn’t otherwise because of incompatible interfaces.

The Adapter Pattern



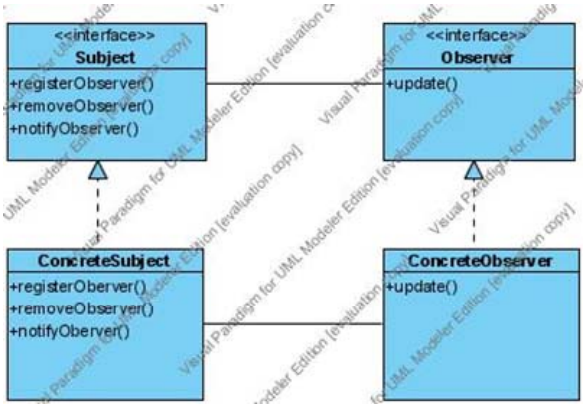
The Adapter Pattern - Example



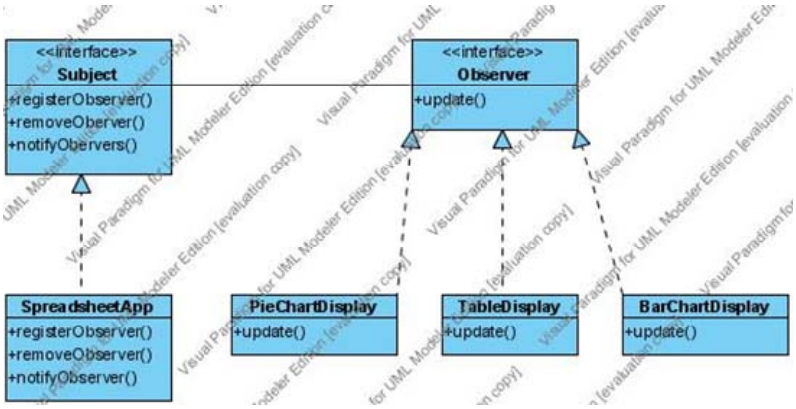
The Observer Pattern

- Defines a one-to-many dependency between objects so that when one object changes state, all of its dependents are notified and updated automatically.

The Observer Pattern



The Observer Pattern Example

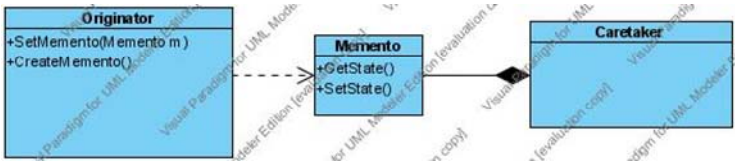


The Memento Pattern

- It is used when there is a need to return an object to one of its previous states.

- Example: request for an “undo”

The Memento Pattern



REFERENCES

Agarwal, R., Sinha, A.P., and Tanniru, M. "The role of prior experience and task characteristics in object-oriented modeling: An empirical study," *International Journal of Human-Computer Studies* (45:6), 1996, pp. 639-667.

Alavi, M., and Leidner, D.E. "Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues," *MIS Quarterly* (25:1), 2001, pp. 107-136.

Al-Kilidar, H., Parkin, P., Aurum, A., and Jeffery, R. "Evaluation of effects of pair work on quality of designs," *Proceedings of the Australian Software Engineering Conference*, 2005, pp. 78- 87.

Ambler, S. "Survey Says: Agile Works in Practice," *Dr. Dobb's Journal*, (31:9), Sep 2006.

Argote, L. *Organizational learning: creating, retaining, and transferring knowledge*, Kluwer Academic, Boston, 1999.

Argote, L., Ingram, P., Levine, J.M., and Moreland, R.L. "Knowledge transfer in organizations: Learning from the experience of others," *Organizational Behavior and Human Decision Processes* (82:1), 2000, pp. 1-8.

Argote, L., McEvily, B., and Reagans, R. "Managing knowledge in organizations: An integrative framework and review of emerging themes," *Management Science* (49:4), 2003, pp. 571-582.

Baker, D.F. "The development of collective efficacy in small task groups," *Small Group Research* (32:4), 2001, pp. 451-474.

Balijepally, V. "Task complexity and effectiveness of pair programming: an experimental study," University of Texas at Arlington, 2005.

Bandura, A. *Social foundations of thought and action: a social cognitive theory*, Prentice-Hall, Englewood Cliffs, N.J., 1986.

Baron, R.M., and Kenny, D.A. "The Moderator Mediator Variable Distinction in Social Psychological-Research - Conceptual, Strategic, and Statistical Considerations," *Journal of Personality and Social Psychology* (51:6), 1986, pp. 1173-1182.

Batra, D. "Conceptual data modeling patterns: Representation and validation," *Journal of Database Management* (16:2), 2005, pp. 84-106.

Beck, K. *Extreme programming explained : embrace change*, Addison-Wesley, Harlow, 1999.

Beck, K., Crocker, R., Meszaros, G., Coplien, J.O., Dominick, L., Paulisch, F., and Vlissides, J. "Industrial experience with design patterns," *Proceedings of the Software Engineering, 1996., Proceedings of the 18th International Conference on*, Berlin, 1996, pp. 103 - 114.

Blackler, F. "Knowledge, knowledge work and organizations: An overview and interpretation," *Organization Studies* (16:6), 1995, pp. 1021-1046.

Blum, B.I. "A taxonomy of software development methods," *Communications of the ACM* (37:11), 1994, pp. 82 - 94.

Boehm, B.W., and Turner, R. *Balancing agility and discipline : a guide for the perplexed*, Addison-Wesley, Boston, 2004.

Bond, C.F., and Titus, L.J. "Social Facilitation - a Meta-Analysis of 241 Studies," *Psychological Bulletin* (94:2), 1983, pp. 265-292.

Bower, G.H., and Hilgard, E.R. *Theories of learning*, Prentice-Hall, Englewood Cliffs, N.J., 1981.

Brodbeck, F., and Greitemeyer, T. "A Dynamic Model of Group Performance: Considering the Group Members' Capacity To Learn.," *Group Processes & Intergroup Relations* (3:2), 2000, pp. 159.

Brodbeck, F.C. "Communication and performance in software development projects," *European Journal of Work and Organizational Psychology* (10:1), 2001, pp. 73-94.

Brown, R. *Group processes : dynamics within and between groups*, Blackwell Publishers, Oxford, UK ; Malden, Mass., 2001.

Campion, M.A., Medsker, G.J., and Higgs, A.C. "Relations between Work Group Characteristics and Effectiveness - Implications for Designing Effective Work Groups," *Personnel Psychology* (46:4), 1993, pp. 823-850.

Cline, M.P. "The pros and cons of adopting and applying design patterns in the real world," *Communications of the ACM* (39:10), 1996, pp. 47-49.

Coad, P. "Object-Oriented Patterns.," *Communications of the ACM* (35:9), 1992, pp. 152-159.

Cockburn, A., and Williams, L. "The Costs and Benefits of Pair Programming," In *Extreme Programming Examined*, G. Succi and M. Marchesi (eds.), Pearson Education, Upper Saddle River, NJ, 2001, pp. 223-243.

Cohen, J. "A Power Primer," *Psychological Bulletin* (112:1), 1992, pp. 155-159.

Collins, B.E., and Guetzkow, H. *A social psychology of group processes for decision making.*, Wiley, New York, 1964.

Compeau, D.R., and Higgins, C.A. "Computer Self-Efficacy - Development of a Measure and Initial Test," *MIS Quarterly* (19:2), 1995, pp. 189-211.

Coplien, J.O., and Schmidt, D.C. *Pattern languages of program design*, Addison-Wesley, Reading, Mass., 1995.

Davis, D.D., and Harless, D.W. "Group vs. Individual Performance in a Price-Searching Experiment," *Organizational Behavior and Human Decision Processes* (66:2), 1996, pp. 215-227.

Diehl, M., and Stroebe, W. "Productivity loss in brainstorming groups: Toward the solution of riddle," *Journal of Personality and Social Psychology* (53:3), 1987, pp. 497-509.

Faraj, S., and Sproull, L. "Coordinating Expertise in Software Development Teams," *Management Science* (46:12), 2000, pp. 1554.

Fernández-Ballesteros, R., Díez-Nicolás, J., Caprara, G.V., Barbaranelli, C., and Bandura, A. "Determinants and Structural Relation of Personal Efficacy to Collective Efficacy.," *Applied Psychology: An International Review* (51:1), 2002, pp. 107.

Flor, N.V., and Hutchins, E.L. "Analyzing distributed cognition in software teams: A case study of team programming during perfective software maintenance," *Proceedings of the Empirical Studies of Programmers: Fourth Workshop*, 1991, pp. 36-64.

Forsyth, D.R. *Group dynamics*, Brooks/Cole, Belmont, Calif., 1999.

Freeman, E., Freeman, E., Sierra, K., and Bates, B. *Head First design patterns*, O'Reilly, Sebastopol, CA, 2004.

Gamma, E., Helm, R., Johnson, R.E., and Vlissides, J. *Design patterns : elements of reusable object-oriented software*, Addison-Wesley, Reading, Mass., 1995.

Gibson, C.B. "Do they do what they believe they can? Group efficacy and group effectiveness across tasks and cultures," *Academy of Management Journal* (42:2), 1999, pp. 138-152.

Gibson, C.B. "From knowledge accumulation to accommodation: cycles of collective cognition in work groups," *Journal of Organizational Behavior* (22), 2001, pp. 121-134.

Gibson, C.B., Randel, A.E., and Earley, P.C. "Understanding group efficacy - An empirical test of multiple assessment methods," *Group & Organization Management* (25:1), 2000, pp. 67-97.

Gick, M.L., and Holyoak, K.J. "The cognitive basis of knowledge transfer," In *Transfer of learning : contemporary research and applications*, S. M. Cormier and J. D. Hagman (eds.), Academic Press, San Diego, 1987, pp. 9-46.

Gist, M.E., and Mitchell, T.R. "Self-Efficacy - a Theoretical-Analysis of Its Determinants and Malleability," *Academy of Management Review* (17:2), 1992, pp. 183-211.

Goldfedder, B., and Rising, L. "A Training Experience with Patterns.," *Communications of the ACM* (39:10), 1996, pp. 60-64.

Gray, W.D., and Orasanu, J.M. "Transfer of cognitive skills," In *Transfer of learning : contemporary research and applications*, S. M. Cormier and J. D. Hagman (eds.), Academic Press, San Diego, 1987, pp. 183-215.

Guerraoui, R. "Strategic Directions in Object-Oriented Programming.," *ACM Computing Surveys* (28:4), 1996, pp. 691-700.

Guindon, R. "Knowledge Exploited by Experts During Software System-Design," *International Journal of Man-Machine Studies* (33:3), 1990, pp. 279-304.

Gully, S.M., Incalcaterra, K.A., Joshi, A., and Beaubien, J.M. "A meta-analysis of team-efficacy, potency, and performance: Interdependence and level of analysis as moderators of observed relationships," *Journal of Applied Psychology* (87:5), 2002, pp. 819-832.

Hagge, L., and Lippe, K. "Sharing Requirements Engineering Experience Using Patterns.," *IEEE Software* (22:1), 2005, pp. 24-31.

Hair, J.F., Tatham, R.L., Anderson, R.E., and Black, W. *Multivariate data analysis*, Prentice Hall, Upper Saddle River, N.J., 1998.

Hansen, M.T. "The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge across Organization Subunits.," *Administrative Science Quarterly* (44:1), 1999, pp. 82.

Hare, A.P. *Handbook of small group research*, Free Press, New York, 1976.

Hare, A.P. "Individual versus groups," In *Small group research : a handbook*, A. P. Hare (ed.) Ablex Pub., Norwood, N.J., 1994, pp. 261-270.

Hart, S.G., and Staveland, L.E. "Development of NASA-TLX: Results of empirical and theoretical research," In *Human mental workload*, P. A. Hancock and N. Meshkati (eds.), North-Holland, New York, N.Y., 1988, pp. 139-183.

Hayne, S.C., Smith, C.A.P., and Turk, D. "The effectiveness of groups recognizing patterns," *International Journal of Human-Computer Studies* (59:5), 2003, pp. 523-543.

Hertel, G., Niedner, S., and Herrmann, S. "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel," *Research Policy* (32:7), 2003, pp. 1159-1177.

Highsmith, J.A. *Agile software development ecosystems*, Addison-Wesley, Boston, 2002.

Hill, G.W. "Group Versus Individual-Performance - Are N + 1 Heads Better Than One," *Psychological Bulletin* (91:3), 1982, pp. 517-539.

Hinsz, V.B., and Nickell, G.S. "Positive reactions to working in groups in a study of group and individual goal decision making," *Group Dynamics-Theory Research and Practice* (8:4), 2004, pp. 253-264.

Hinsz, V.B., Tindale, R.S., and Vollrath, D.A. "The emerging conceptualization of groups as information processors," *Psychological Bulletin* (121:1), 1997, pp. 43-64.

Hirokawa, R.Y. "Group communication and problem-solving effectiveness I: A critical review of inconsistent findings," *Communication Quarterly* (30:2), 1982a, pp. 134-141.

Hirokawa, R.Y. "Group communication and problem-solving effectiveness II: An exploratory investigations of procedural functions," *The Western Journal of Speech Communication* (47), 1982b, pp. 59-74.

Hirokawa, R.Y., and Salazar, A.J. "Task-group communication and decision-making performance," In *The handbook of group communication theory & research*, L. R. Frey, D. S. Gouran and M. S. Poole (eds.), Sage Publications Inc., Thousand Oaks, Calif., 1999, pp. 167-191.

Hoegl, M., and Gemuenden, H.G. "Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence," *Organization Science* (12:4), 2001, pp. 435-449.

Hoegl, M., Parboteeah, K.P., and Gemuenden, H.G. "When teamwork really matters: task innovativeness as a moderator of the teamwork-performance relationship in software development projects," *Journal of Engineering and Technology Management* (20:4), 2003, pp. 281-302.

Huberty, C.J., and Morris, J.D. "Multivariate Analysis of Versus Multiple Univariate Analyses," *Psychological Bulletin* (105:2), 1989, pp. 302-308.

Hunton, J.E., and Beeler, J.O. "Effects of user participation in systems development: A longitudinal field experiment.," *MIS Quarterly* (21:4), 1997, pp. 359.

Hutchins, E. "How a Cockpit Remembers Its Speeds," *Cognitive Science* (19:3), 1995, pp. 265-288.

Ilgen, D.R., Hollenbeck, J.R., Johnson, M., and Jundt, D. "Teams in organizations: From input-process-output models to IMOI models," *Annual Review of Psychology* (56), 2005, pp. 517-543.

Irwin, G. "The role of similarity in the reuse of object-oriented analysis models," *Journal of Management Information Systems* (19:2), 2002, pp. 219-248.

Jones, G.R. "Socialization Tactics, Self-Efficacy, and Newcomers Adjustments to Organizations," *Academy of Management Journal* (29:2), 1986, pp. 262-279.

Judge, T.A., and Bono, J.E. "Relationship of Core Self-Evaluations Traits -- Self-Esteem, Generalized Self-Efficacy, Locus of Control, and Emotional Stability -- With Job Satisfaction and Job Performance: A Meta-Analysis.," *Journal of Applied Psychology* (86:1), 2001, pp. 80-92.

Kane, A.A., Argote, L., and Levine, J.M. "Knowledge transfer between groups via personnel rotation: Effects of social identity and knowledge quality," *Organizational Behavior and Human Decision Processes* (96:1), 2005, pp. 56-71.

Kankanhalli, A., Tan, B.C.Y., and Wei, K.K. "Contributing knowledge to electronic knowledge repositories: An empirical investigation," *Mis Quarterly* (29:1), 2005, pp. 113-143.

Katz-Navon, T.Y., and Erez, M. "When collective- and self-efficacy affect team performance," *Small Group Research* (36:4), 2005, pp. 437-465.

Kerievsky, J. *Refactoring to patterns*, Addison-Wesley, Boston, 2005.

Krauss, R.M., and Fussell, S.R. "Mututal knowledge and communicative effectiveness," In *Intellectual teamwork : social and technological foundations of cooperative work*, J. R. Galegher, R. E. Kraut and C. Egido (eds.), L. Erlbaum Associates, Hillsdale, N.J., 1990, pp. 111-145.

Krueger, C.W. "Software Reuse," *Computing Surveys* (24:2), 1992, pp. 131-183.

Larman, C. *Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development*, Prentice Hall PTR, Upper Saddle River, N.J., 2004.

Larson, J.R., and Christensen, C. "Groups as Problem-Solving Units - toward a New Meaning of Social Cognition," *British Journal of Social Psychology* (32), 1993, pp. 5-30.

Laughlin, P.R., Bonner, B.L., and Miner, A.G. "Groups perform better than the best individuals on Letters-to-Numbers problems," *Organizational Behavior & Human Decision Processes* (88:2), 2002, pp. 605-620.

Laughlin, P.R., and Ellis, A.L. "Demonstrability and Social Combination Processes on Mathematical Intellective Tasks," *Journal of Experimental Social Psychology* (22:3), 1986, pp. 177-189.

Laughlin, P.R., Hatch, E.C., Silver, J.S., and Boh, L. "Groups perform better than the best individuals on letters-to-numbers problems: effects of group size," *Journal of Personality and Social Psychology* (90:4), 2006, pp. 644-651.

Laughlin, P.R., VanderStoep, S.W., and Hollingshead, A.B. "Collective Individual Induction: Recognition of Truth, Rejection of Error, and Collective Information Processing," *Journal of Personality and Social Psychology* (61:1), 1991, pp. 50-67.

Lee, C., and Bobko, P. "Self-Efficacy Beliefs - Comparison of 5 Measures," *Journal of Applied Psychology* (79:3), 1994, pp. 364-369.

Levine, T.R., and Hullett, C.R. "Eta squared, partial eta squared, and misreporting of effect size in communication research," *Human Communication Research* (28:4), 2002, pp. 612-625.

Littlepage, G.E., Schmidt, G.W., Whisler, E.W., and Frost, A.G. "An Input-Process-Output Analysis of Influence and Performance in Problem-Solving Groups," *Journal of Personality and Social Psychology* (69:5), 1995, pp. 877-889.

Locke, E.A. "The nature and causes of job satisfaction," In *Handbook of industrial and organizational psychology*, M. D. Dunnette (ed.) Rand McNally College Pub. Co., Chicago, 1976, pp. 1297-1349.

Majchrzak, A., Cooper, L.P., and Neece, O.E. "Knowledge reuse for innovation," *Management Science* (50:2), 2004, pp. 174-188.

Markus, M.L. "Toward a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success," *Journal of Management Information Systems* (18:1), 2001, pp. 57-93.

Marquart, D.I. "Group problem solving," *Journal of Social Psychology* (41), 1955, pp. 102-113.

Martin, R.C. *Agile software development : principles, patterns, and practices*, Prentice Hall, Upper Saddle River, N.J., 2003.

May, D., and Taylor, P. "Knowledge management with patterns.," *Communications of the ACM* (46:7), 2003, pp. 94-99.

Mayer, R.E. "From novice to expert," In *Handbook of human-computer interaction*, M. Helander (ed.) North-Holland, Amsterdam; New York, 1988, pp. 569-580.

McGrath, J.E. *Groups : interaction and performance*, Prentice-Hall, Englewood Cliffs, N.J., 1984.

Mili, H., Mili, F., and Mili, A. "Reusing Software - Issues and Research Directions," *Ieee Transactions on Software Engineering* (21:6), 1995, pp. 528-562.

Montealegre, R., and Keil, M. "De-escalating information technology projects: lessons from the denver international airport.," *MIS Quarterly* (24:3), 2000, pp. 417-447.

Morisio, M., Ezran, M., and Tully, C. "Success and failure factors in software reuse," *Ieee Transactions on Software Engineering* (28:4), 2002, pp. 340-357.

Morris, M.G., Speier, C., and Hoffer, J.A. "An examination of procedural and object-oriented systems analysis methods: Does prior experience help or hinder performance?," *Decision Sciences* (30:1), 1999, pp. 107-136.

Mulvey, P.W., and Klein, H.J. "The impact of perceived loafing and collective efficacy on group goal processes and group performance," *Organizational Behavior and Human Decision Processes* (74:1), 1998, pp. 62-87.

Neter, J., Kutner, M.H., Wasserman, W., and Nachtsheim, C.J. *Applied linear statistical models*, Irwin, Chicago, 1996.

Newell, A., and Simon, H.A. *Human problem solving*, Prentice-Hall, Englewood Cliffs, N.J.,, 1972.

Nijstad, B.A., Stroebe, W., and Lodewijkx, H.F.M. "The illusion of group productivity: a reduction of failures explanation," *European Journal of Social Psychology* (36:1), 2006, pp. 31-48.

Nonaka, I. "A Dynamic Theory of Organizational Knowledge Creation," *Organization Science* (5:1), 1994, pp. 14-37.

Nonaka, I.o., and Takeuchi, H. *The knowledge-creating company*, Oxford University Press, New York Oxford, 1995.

Nosek, J.T. "The Case for Collaborative Programming," *Communications of the ACM* (41:3), 1998, pp. 105-108.

Nunnally, J.C. *Psychometric theory*, McGraw-Hill, New York, 1978.

Ocker, R., Fjermestad, J., Hiltz, S.R., and Johnson, K. "Effects of four modes of group communication on the outcomes of software requirements determination," *Journal of Management Information Systems* (15:1), 1998, pp. 99.

Oetzel, J.G. "Self-construals, communication processes, and group outcomes in homogeneous and heterogeneous groups," *Small Group Research* (32:1), 2001, pp. 19-54.

Parrish, A., Smith, R., Hale, D., and Hale, J. "A field study of developer pairs: Productivity impacts and implications," *IEEE Software* (21:5), 2004, pp. 76-79.

Parsons, J., and Saunders, C. "Cognitive heuristics in software engineering: Applying and extending anchoring and adjustment to artifact reuse," *IEEE Transactions on Software Engineering* (30:12), 2004, pp. 873-888.

Pavitt, C. "Why we still have to be reductionists about group memory," *Human Communication Research* (29:4), 2003, pp. 624-629.

Pescosolido, A.T. "Informal leaders and the development of group efficacy.," *Small Group Research* (32:1), 2001, pp. 74.

Phillips, G.M. *Communication and the small group*, Bobbs-Merrill, Indianapolis,, 1966.

Poole, M.S., and Hirokawa, R.Y. "Introduction: Communication and group decision making," In *Communication and group decision making*, R. Y. Hirokawa and M. S. Poole (eds.), SAGE Publications, Thousand Oaks, Calif., 1996, pp. 3-18.

Poppendieck, M. "Wicked Projects," *Software Development Magazine*, May 2002.

Prechelt, L., Unger, B., Tichy, W.F., Brossler, P., and Votta, L.G. "A controlled experiment in maintenance comparing design patterns to simpler solutions," *IEEE Transactions on Software Engineering* (27:12), 2001, pp. 1134-1144.

Prechelt, L., Unger-Lamprecht, B., Philippsen, M., and Tichy, W.F. "Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance," *IEEE Transactions on Software Engineering* (28:6), 2002, pp. 595-606.

Prieto-Diaz, R. "Status report: software reusability," *Software, IEEE* (10:3), 1993, pp. 61-66.

Propp, K.M. "In search of the assembly bonus effect: Continued exploration of communication's role in group memory," *Human Communication Research* (29:4), 2003, pp. 600-606.

Purao, S., Storey, V.C., and Han, T.D. "Improving analysis pattern reuse in conceptual design: Augmenting automated processes with supervised learning," *Information Systems Research* (14:3), 2003, pp. 269-290.

Reeves, W.W. *Cognition and complexity : the cognitive science of managing complexity*, Scarecrow Press, Lanham, Md., 1996.

Rehder, B., Pennington, N., and Lee, A.Y. "Scoring the completeness of software designs," *Journal of Systems and Software* (36:1), 1997, pp. 33-68.

Richter, C. "Design Problems and Object-Oriented Solutions," Objective Engineering, Inc.,http://www.oeng.com/problemsandsolutions.htm, 2004, Last Accessed: 6th December, 2006

Ringelmann, M. "Recherches sur les moteurs animés: travail de l'homme," *Annales de l'Insitut National Agronomique* (2nd series:12), 1913, pp. 1-40.

Rising, L. "Patterns: A way to reuse expertise," *Ieee Communications Magazine* (37:4), 1999, pp. 34-36.

Roberts, T.L., Leigh, W., Purvis, R.L., and Parzinger, M.J. "Utilizing knowledge links in the implementation of system development methodologies," *Information and Software Technology* (43:11), 2001, pp. 635-640.

Robillard, P.N. "The role of knowledge in software development.," *Communications of the ACM* (42:1), 1999, pp. 87-92.

Rothenberger, M.A. "Project-level reuse factors: Drivers for variation within software development environments," *Decision Sciences* (34:1), 2003, pp. 83-106.

Salanova, M., Llorens, S., Cifre, E., Martinez, I.M., and Schaufeli, W.B. "Perceived collective efficacy, subjective well-being and task performance among electronic work groups - An experimental study," *Small Group Research* (34:1), 2003, pp. 43-73.

Salazar, A.J., Hirokawa, R.Y., Propp, K.M., Julian, K.M., and Leatham, G.B. "In Search of True Causes - Examination of the Effect of Group Potential and Group-Interaction on Decision Performance," *Human Communication Research* (20:4), 1994, pp. 529-559.

Sarker, S., Sarker, S., Nicholson, D.B., and Joshi, K.D. "Knowledge transfer in virtual systems development teams: An exploratory study of four key enablers," *IEEE Transactions on Professional Communication* (48:2), 2005, pp. 201-218.

Schmidt, D. "Using design patterns to guide the development of reusable object-oriented software," *ACM Computing Surveys* (28:4es), 1996, pp. 162.

Schmidt, D.C., Fayad, M., and Johnson, R.E. "Software Patterns.," *Communications of the ACM* (39:10), 1996, pp. 36-39.

Sen, A. "The Role of Opportunism in the Software Design Reuse Process.," *IEEE Transactions on Software Engineering* (23:7), 1997, pp. 418-436.

Shaft, T.M., and Vessey, I. "The Role of Cognitive Fit in the Relationship Between Software Comprehension and Modification," *MIS Quarterly* (30:1), 2006, pp. 29-55.

Shaw, J.D., Duffy, M.K., and Stark, E.M. "Interdependence and preference for group work: Main and congruence effects on the satisfaction and performance of group members," *Journal of Management* (26:2), 2000, pp. 259-279.

Shaw, M.E. "A comparison of individuals and small groups in the rational solution to complex problems," *American Journal of Psychology* (44), 1932, pp. 491-504.

Simon, H.A. "The structure of ill structured problems," *Artificial Intelligence* (4), 1973, pp. 181-201.

Singley, M.K., and Anderson, J.R. *The transfer of cognitive skill*, Harvard University press, Cambridge, Mass., 1989.

Smith, H.J., and Keil, M. "The reluctance to report bad news on troubled software projects: a theoretical model.," *Information Systems Journal* (13:1), 2003, pp. 69-95.

Sonnentag, S. "Excellent performance: The role of communication and cooperation processes," *Applied Psychology-an International Review-Psychologie Appliquee-Revue Internationale* (49:3), 2000, pp. 483-497.

Speier, C., and Morris, M.G. "The influence of query interface design on decision-making performance," *MIS Quarterly* (27:3), 2003, pp. 397-423.

Stajkovic, A.D., and Luthans, F. "Self-efficacy and work-related performance: A meta-analysis.," *Psychological Bulletin* (124:2), 1998, pp. 240.

Standish Group "Extreme Chaos," The Standish Group, 2001.

Stasser, G., and Stewart, D. "Discovery of Hidden Profiles by Decision-Making Groups - Solving a Problem Versus Making a Judgment," *Journal of Personality and Social Psychology* (63:3), 1992, pp. 426-434.

Stasser, G., and Titus, W. "Pooling of Unshared Information in Group Decision-Making - Biased Information Sampling During Discussion," *Journal of Personality and Social Psychology* (48:6), 1985, pp. 1467-1478.

Steiner, I.D. *Group process and productivity*, Academic Press, New York, 1972.

Swap, W., Leonard, D., Shields, M., and Abrams, L. "Using mentoring and storytelling to transfer knowledge in the workplace," *Journal of Management Information Systems* (18:1), 2001, pp. 95-114.

Taggar, S., and Seijts, G.H. "Leader and staff role-efficacy as antecedents of collective-efficacy and team performance," *Human Performance* (16:2), 2003, pp. 131-156.

Thomas-Hunt, M.C., Ogden, T.Y., and Neale, M.A. "Who's really sharing? Effects of social and expert status on knowledge exchange within groups," *Management Science* (49:4), 2003, pp. 464-477.

Thompson, L., Gentner, D., and Loewenstein, J. "Avoiding missed opportunities in managerial life: Analogical training more powerful than individual case training," *Organizational Behavior and Human Decision Processes* (82:1), 2000, pp. 60-75.

Thorndike, E.L. *Principles of Teaching*, A. G. Seiler, New York, 1906.

Tiwana, A. "An empirical study of the effect of knowledge integration on software development performance," *Information and Software Technology* (46:13), 2004, pp. 899-906.

Tiwana, A., and McLean, E.R. "Expertise integration and creativity in Information Systems Development," *Journal of Management Information Systems* (22:1), 2005, pp. 13-43.

Triplett, N. "The dynamogenic factors in pacemaking and competition," *American Journal of Psychology* (9), 1898, pp. 507-33.

Tschan, F. "Communication Enhances Small-Group Performance If It Conforms to Task Requirements - the Concept of Ideal Communication Cycles," *Basic and Applied Social Psychology* (17:3), 1995, pp. 371-393.

Wallace, L., and Keil, M. "Software projects risks and their effect on outcomes.," *Communications of the ACM* (47:4), 2004, pp. 68-73.

Walz, D.B., Elam, J.J., and Curtis, B. "Inside a software design team: Knowledge acquisition, sharing, and integration.," *Communications of the ACM* (36:10), 1993, pp. 63-77.

Whiteoak, J.W., Chalip, L., and Hort, L.K. "Assessing group efficacy - Comparing three methods of measurement," *Small Group Research* (35:2), 2004, pp. 158-173.

Williams, L., Kessler, R.R., Cunningham, W., and Jeffries, R. "Strengthening the case for pair programming," *IEEE Software* (17:4), 2000, pp. 19-25.

Williams, L.A., and Kessler, R.R. "All I Really Need to Know About Pair Programming I Learned in Kindergarten," *Communications of the ACM* (43:5), 2000, pp. 108-114.

Wittenbaum, G.M., and Staseer, G. "Management of information in small groups," In *What's social about social cognition? : research on socially shared cognition in small*

groups, J. L. Nye and A. M. Brower (eds.), Sage Publications, Thousand Oaks, Calif., 1996, pp. 3-28.

Wood, R., and Bandura, A. "Social Cognitive Theory of Organizational Management," *Academy of Management Review* (14:3), 1989, pp. 361-384.

Yeatts, D.E., and Hyten, C. *High-performing self-managed work teams : a comparison of theory to practice*, Sage Publications, Thousand Oaks, 1998.

Yi, M.Y., and Davis, F.D. "Developing and Validating an Observational Learning Model of Computer Software Training and Skill Acquisition.," *Information Systems Research* (14:2), 2003, pp. 146.

Zaccaro, S.J., Blair, V., Peterson, C., and Zazanis, M. "Collective Efficacy," In *Self-efficacy, adaptation, and adjustment : theory, research, and application*, J. E. Maddux (ed.) Plenum Press, New York, 1995, pp. 395.

Zajonc, R.B. "Social facilitation.," *Science* (149:3681), 1965, pp. 269-274.

Zellmer-Bruhn, M.E. "Interruptive events and team knowledge acquisition," *Management Science* (49:4), 2003, pp. 514-528.

BIOGRAPHICAL INFORMATION

George Mangalaraj holds Bachelors in Mechanical Engineering degree from the Coimbatore Institute of Technology, India; Post Graduate Diploma in Management from Indira Gandhi National Open University, India; and Masters in Information Systems degree from the University of Texas at Arlington. He has nearly five years of industry experience that include experience in software development. He has published his research in refereed journals and conference proceedings. With his teaching and research, he hopes to make a positive impact on the academic community.