AN INVESTIGATION OF THE EFFICACY OF MIND MAPS IN SOFTWARE DEVELOPMENT AMONG INDIVIDUALS AND PAIRS

by

PHILIP L. BOND

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 2015

Acknowledgements

I express my gratitude to my dissertation committee chair, Dr. Sridhar Nerur for his consistent support, encouragement and insights throughout my Ph.D. program. I am grateful for Dr. Nerur's ever-patient advice and guidance during key periods in my Ph.D. student life. His conversations with me provided high insight into key questions about Information Systems, Business Administration, academic research and living a meaningful life. His professional growth during my time as his student illustrates what a great professional example he is, and I appreciate his investment in me.

I will also thank other members of my committee for their support and assistance. Dr. Craig Slinkman spent many hours discussing baseball and analytics while introducing me to thought processes fit for the academic life. Dr. Mary Whiteside taught me much about how to think about data, parametric and nonparametric, and the implications of assumptions based on analytics. Dr. Feirong Yuan reminded me of the cultural side of behavioral research and brought her insights to its impact on research.

I also thank Ganapathiraman Raman, Yoon Sang Lee, Ramakrishna Dantu, O'la al-Laymoun and Kriti Chauhan for their friendship and support all through the years of our Ph.D. studies.

Many friends helped me during my Ph.D. program including Kriti Chauhan and Goutham C. Manghnani who spent patient hours helping me refine our rubric and then grading the results of my experiment.

Lastly, I hold special thanks in my heart for my family and friends for their support and love as I strived toward this goal. Patience beyond compare and understanding during this challenge were given by my wife, Catherine, during this interesting time in our lives together.

November 19, 2015

Abstract

AN INVESTIGATION OF THE EFFICACY OF MIND MAPS

IN SOFTWARE DEVELOPMENT AMONG

INDIVIDUALS AND PAIRS

Philip L. Bond, PhD

The University of Texas at Arlington, 2015

Supervising Professor: Sridhar P. Nerur

Software development is a cognitively demanding endeavor in which the creation and exchange of knowledge is paramount. Collaborative development, particularly pair programming, has been more advocated and adopted recently by practitioners. However, empirical studies have not categorically established the efficacy of pairs vis-à-vis individuals in software design contexts. In fact, recent findings suggest that pairs seldom outperform best individuals although they tend to do better than average individuals. Group losses arising from lack of coordination and/or communication could be a plausible explanation for this. The extant literature suggests that Mind Maps have the potential to graphically capture concepts and their relationships and hence can promote a shared understanding of the problem being solved. This shared understanding, in turn, can facilitate communication and coordination. This study, anchored in the theoretical

foundations of distributed cognition, investigates the impact that Mind Maps have on the performance of pairs vis-à-vis individuals engaged in a design task.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Background and Research Questions

1.1 Background

Software design requires integration of diverse knowledge sources and is cognitively demanding. The challenges of good design include properly cataloguing, critiquing and choosing among design alternatives. Designers must decide solutions alternatives in order to reduce costs, reduce errors and deliver a product that meets specified requirements.

Industry already uses "best practices" such as pair programming, iterative development, and test-driven development. These best practices are tools to help wrestle with a cognitively challenging and "ill-structured" activity as defined by Simon (1973). Software must be built, and research into improving best practices could be enormously beneficial to practitioners.

Collaborating pairs and their social interactions during software design can be seen as socially distributed cognition. These pairs' use of artifacts and external representations (e.g., working software code, databases of previous decisions, story cards, charts and notes) can be seen as structurally distributed cognition. Good research brings theory and practice together to enhance industry practice and academic research into both practice and extending theory. Further good research would be beneficial both to practitioners and academics, helping to discover and define "best practices" and the reasons for them.

Extant literature in paired software development has been inconsistent. Research rigor has been increased by comparing non-collaborating individuals as

nominal pairs rather than comparing paired team to individuals (Balijepally, Mahapatra, Nerur, & Price, 2009; Mangalaraj, Nerur, Mahapatra, & Price, 2014). Research finds paired teams perform better than the "second best" individual of a nominal, non-collaborating pair and as good as the "first best" individual but not better than the "first best" individual in a nominal pair (Balijepally et al., 2009; Hill, 1982; Mangalaraj et al., 2014). Pairs suffer net losses when combined due to process losses (Hanks, 2008), social loafing (Kravitz & Martin, 1986) and higher costs of communication and coordination (Shaw & Ashton, 1976; Steiner, 1972).

Research into the theoretical underpinnings of agile team methodologies can benefit practitioners and academics alike (Nerur, Cannon, Ballijepally & Bond, 2010). How can we help collaborating pairs, common in the software development industry, to work better? Is it possible to improve the gains of paired development and arrive at a larger net gain?

External representations and artifacts are often used in both design and engineering problem solving. Sketches have a long history in design practice and engineers use lots of documentation of standards and lists of user requirements. Diagrams are often used in team settings to conjure explicit socially-agreed information and serve as declarative memory of user requirements, engineering standards, and documentation of these agreements.

Mind maps are spider-type diagrams for mapping concepts and their hierarchical relationships. The fundamental concept is placed in the middle of the drawing and related aspects or facets of the concept are then expanded from the center. Hierarchy among the concepts is enforced – a mind map might list "cinema"

then “popcorn” and finally “butter,” where a concept map would equate the three without hierarchy. Using diagrams as cognitive external representations has the potential to facilitate creation and exploitation of knowledge-intensive shared social and structural cognition, constructed of and by the participants situated in cognitive development processes (Zhang & Norman, 1994). Mind maps reduce cognitive loading and ease designers’ efforts to comply with rules and standards, unambiguously capturing concepts and their relationships, to ease communication and coordination between team members, and provide an external artifact to guide problem solving, as well as serve both individual and joint cognitive efforts to wrestle with more abstract aspects of the task (Eppler, 2006). This combination of effects serves usefully in cognitively challenging, diverse knowledge tasks, and is therefore fitting for analyzing paired software development.

Paired software development requires balancing diverse knowledge sources. Distributed Cognition uses both social and structural cognition (Hollan, Hutchins, & Kirsh, 2000) which for paired software developers includes the conversations and communications of the pair as social cognition, and structural cognition distributed among the pair and their diagrams, sketches and documentation. Distributed Cognition states that the combination of social and structural cognition extends and enhances the total cognition to exceed the sum of the individual cognitions involved (Flor & Hutchins, 1991). Artifacts and external representations increase cognitively abstract thought and decrease mental loading of the participants (Zhang & Norman, 1994). The impact of using mind maps in both individual and paired software development, a distributed cognition setting, is the focus of this research.

1.2 Research Questions

The importance of this research is it will benefit both academic research and practitioners alike. Academic research will benefit by bringing mind maps into the information systems field as an artifact or external representation system which could be used not only in software design but in any challenging group or team work. One example might be team research design, agreeing on research objective priorities and possible paths to them.

Practitioners could use mind maps to enhance solution alternatives quality (using mind maps to prioritize engineering requirements), facilitate communications among team members (creating the artifact forces communication and coordination, and shows socially-approved paths to goals), or bring detail to alternative solutions (mind maps act as work breakdown schedules in project management).

Using mind maps among paired developers in a design solution setting may lead to increases in pair development approaching that of the best individual developer, a long-sought quest that could bring invaluable benefit to software development and design.

Therefore, this study addresses the following research questions:

1) Do software designers working in pairs have a greater effect on the quality of software design than do individuals?

2) Do mind maps assist software developers in a design task to increase the quality of design, among individuals and in pairs?

3) Do mind maps assist collaborating pairs in a software design task equal the work of the best individuals?

1.3 Overview of Dissertation

The remainder of the dissertation is structured as follows. Chapter 2 is a literature review of the three main areas involved in the research: Agile Software Design and Development, Distributed Cognition and External Representations Theory. Chapter 3 develops the research model and discusses the hypotheses in the research. Chapter 4 outlines the experimental design methodology and discusses the data analysis techniques to be used. Chapter 5 presents the results of the experiment and the statistical analysis involved. Finally, Chapter 6 discusses the findings of the research and presents future research directions.

Chapter 2

Literature Review

This research combines eclectic streams of literature support, namely Agile Software Development, Distributed Cognition and External Representations Theory, in order to examine mind map efficacy on software development in paired developers. This is illustrated here in Figure 2.1:

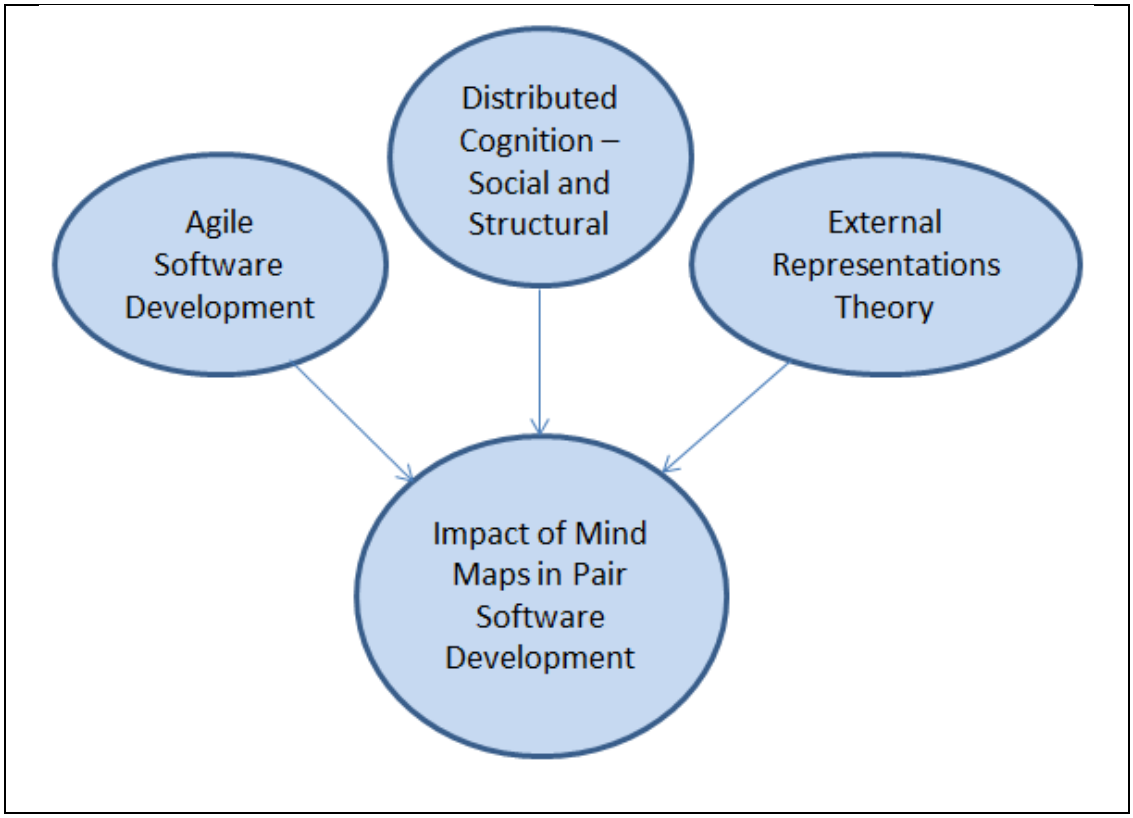

Figure 2-1: Three Theoretical Streams Informing This Research

2.1 Agile Software Development Teams

Agile software development is an approach to the business, social and engineering aspects of software development. The Agile Manifesto (Beck et al., 2001) lists four value choices – “We value

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan."

Instead of the traditional, linear, waterfall approach of system requirements analysis documentation, then system requirements design and its documentation, followed by software coding, module testing, system testing and finally delivery and implementation, agile software development prefers a different approach.

Agile software development is based on iterative and incremental development, believing “people trump process” with emphasis is on relationships among stakeholders such as software developers, management, customers and so forth. Agile processes are designed to be lightweight and dexterous. In this way, all stakeholders have the opportunity to communicate and coordinate the next sprint, setting goals for usability and feasibility.

Agile methodologies continue to grow and become more accepted and used by industry (Standish Group, 2013; VersionOne.com, 2013). Agile software methodologies use adaptive planning and design, with iterative and evolutionary development. These concepts form a problem framing schema that anticipates

changes ahead and adapts shifts in teams' functional makeup and tasking in order to flow into the next iteration of the overall project (Beck et al., 2001).

The use of collaborating paired teams in agile software engineering facilitates the requirements of being agile, and an agile framework facilitates delivery of high quality working software while increasing job satisfaction.

2.1.1 Pairs in Agile Software Development

Agile software development methodologies such as eXtreme Programming (XP) and Scrum use pair software development as a basis for assigning and completing work, incrementally and iteratively. Sprints produce stable working software in short fixed time periods, often four weeks between releases and are responsive to user changes and frequent code review. When pair software development is used, software code review is continuous – one developer codes while the second developer, in the same cubicle, watches and interacts with the first coder as they use only one computer. This practice yields higher quality (fewer mistakes, more elegant engineering design, less rework of code) software, with one developer being the "driver" and one being the "navigator," and the two changing roles many times during a day of work. This close relationship facilitates information interchange among the pair and is also beneficial by allowing newly hired developers to work closely with more experienced developers.

Pair software development brings increased cost of two persons doing the coding, but the process yields higher quality software, as well as fulfilling the need to transfer information, knowledge and context between software developers,

especially across functional areas such as webmaster, database guru, Graphical User Interface (GUI) experts, and so on (Nawrocki & Wojciechowski, 2001). Working in pairs allows stronger cognitive recall of possible implications of proposed goal changes, shortening the time between those decisions and consequences of those decisions, and transferring knowledge between team members (Hoda, Noble, & Marshall, 2010). So, while pair software development induces costs of sharing more information, it produces fewer mistakes and increases the team's competencies.

The increased costs of two developers are offset by increased speed of output and higher quality of output (Cockburn & Williams, 2000). Benefits of pair software development include better quality output, lower software defect rates, learning and knowledge transfer, shared decision making and confidence, increased morale, and increased output (Aiken, 2004; Begel & Nagappan, 2008; Cockburn & Williams, 2000; Padberg & Muller, 2003). The cross-training and deeper relationships among pairs increase trust and commitment among members. Once this trust begins, creativity and spontaneity are increased (Moe, Dingsoyr, & Dyba, 2009). Pair software development builds better software designers and higher quality software, both economic goods to the organization.

Pair software development practices often bring benefits to the individual members and to the organization as well. Williams and Kessler (2001) found increased confidence and reduced errors among pair programming students. Williams, Kessler et al found higher confidence and increased job satisfaction. (Williams et al., 2000) Increased quality (effective unit testing and adhering to coding guidelines) resulting in reduced errors, morale increases affect job retention,

trust and teamwork are increased through pair software development, knowledge transfer is increased in pair software development and further enhanced among those who pair with more than one person over time (pair rotation), learning by watching and learning by doing as partners watch others' approaches to tasks, language capabilities, and using development tools (Aiken, 2004). These findings are consistent with practitioners' reported musings about the social, employee and organizational benefits of agile practices and pair software development.

Ramasubbu et al (2012) examined the impact of structural complexity on designer and programmer strategy choices among individual and paired developers, treating each pair as a distributed cognition system. Findings support the hypothesis that pair software development choose to rely on each other's cognition to reduce the mental loading and overall effort to comprehend and understand software systems.

The nominal pair created by randomly pairing individual performers allows comparing a collaborating pair with both the best and second best members of a nominal pair (Laughlin, Bonner, & Miner, 2002). This comparison device provides better insight into paired collaborating team performance vis-à-vis individual performance (Balijepally et al., 2009; George Mangalaraj et al., 2014). With all the positives available in pair software development, why do pairs still perform lower than the best individual? How can we help pairs work better?

It is believed that communication and coordination gains are overwhelmed by losses which reduce the net gain in pair development (Balijepally, 2006; Brodbeck & Greitemeyer, 2000; Hill, 1982; Propp, 2003). Therefore, anything that increases the

communication and coordination in pair programming would be beneficial not only for academic researchers, but could prove profitable to practitioners as well.

2.1.2 Design in Agile Software Development

Groups, teams and dyads have access to a greater quantity of ideas and resources in software development than individuals (Flor & Hutchins, 1991; Hinsz, Tindale, & Vollrath, 1997) but rarely outperform the best individual (Hill, 1982). Group performance better than that of the best member is rare, due to process losses in collaborative cognitive tasks (Hill, 1982; Shaw & Ashton, 1976; Steiner, 1972). Groups improve with experience amid learning while even nominal pairs' performance improves with increased cognitive resources (Brodbeck & Greitemeyer, 2000), and specifically in the task of software development, time and repeated tasks (both of which assume learning by individual members) show improved performance in increased quality of product and less total time to completion (Lui & Chan, 2006).

Software design in pairs combines both the traditional design aspects of framing, defining and solving problems in an organizational setting, as well as restrictions brought by working in software engineering and the standards of that particular discipline. Traditional design perspectives include creativity and innovation in framing and defining the ill-structured problem at hand. Mednick (1962) posited an associative basis for creativity and illustrated that the greater number of associations than an individual has to the requisite elements of a problem, the higher probability of reaching a creative solution. Flor & Hutchins (1991) point out that software development pairs have a higher quantity of associated choices than

do individuals, from which they can build a solution. Zhang & Patel (2002) posit that good design choices must account for analysis of user requirements, representational affordances, proper system function, and the tasks and subtasks required to complete the design. Psychologists Tversky & Kahneman (1974) showed that humans often rely on heuristics to initialize work in uncharted, unfamiliar territory. Finally, Schön stated that design is "a reflective conversation with the materials of a design situation," where reflection is a continuous learning activity (Schön, 1984, p.77).

Based on these perspectives, pair software development faces design problems, ill-structured and difficult to know where to begin, in approaching software development problems. This is compounded by software engineering standards which impose their own requirements, and management's need for feedback on budget and time expectations. Finally, customer interaction and communication can provide valuable input on priorities and premiums to point out what matters most – at least for the current iteration.

Because design is an ill-structured problem, problem framing using heuristics and being reflective in iterative action during pair software development can be beneficial. Research on wicked problems in IS design indicates that these problems are subjective, are interrelated and have no stopping rules (Gasson, 2005). External representations in collaborative problem-solving facilitate flexible learning environments and sketches help this occur (Cox & Brna, 1994; Zhang & Norman, 1994). Artifacts and external representations increase the quality of design and extend individual cognition.

Use of sketching in design is a long-standing practice. Sketching helps engineering design through increased conceptualization – higher quantities of drawings and sketches during early-to-middle design phases correlate with better design outcome (Yang, 2008). Design teams use sketches to offload mental workload and increase ideation in engineering design (Bilda & Gero, 2007; Kokotovich, 2008; Shah et al, 2001). Sketching during design supports better design by cognitive externalization (Cross, 2001), and the use of external representations is enhanced in face-to-face communications by gesture and pointing (Björklund, 2013; Corrie, 2010; Čubranic, Storey, & Ryall, 2006). Therefore, use of external representations and close communication enhances design in a dyad setting.

Design can be thought of as a set of situated, iterative decisions to address each task and subtask in incrementally developing software (Canfora et al, 2007; Müller, 2006; Wirth, 1971).

Collaborative cognitive design research using reflective practice analysis and latent semantic analysis to model team cognition over time has helped to identify which aspects of team mental models are relevant to design performance. The emergence of a shared team mental model, the accuracy of that model in relation to a referent model, and the enactment of that model as goal-directed behavior were all enhanced by using collective cognitive structures and external representations (Dong, Kleinsmann, & Deken, 2013).

2.1.3 Communications in Agile Software Development

Communication in Agile software development is needed to build, enhance and improve team mental models and coordination among team members and other stakeholders (Abdullah & Honiden, 2011; Cockburn, 2002; Hulkko & Abrahamsson, 2005; Maheshwari, Kumar, & Kumar, 2012; Melnik & Maurer, 2004; Pikkarainen et al, 2008; Sfetsos, Angelis, & Stamelos, 2006; Syed-Abdullah, Holcombe, & Gheorge, 2006; Vidgen & Wang, 2009). Agile practices are supported by face-to-face communication (Misra, 2012) and evidence that software development pairs could reduce errors of confirmation bias or incorrect mental models (Goldman & Miller, 2010). Ambiguity surfaces in close, rich communication when partners continually attempt to confirm specifications and expectations, allowing conflict resolution within the pair (Abdullah & Honiden, 2011), and enhancing agile teams' ability to self-manage (Monteiro et al., 2011). Knowledge diversity among teams' members makes it more difficult for team members to develop a shared mental model due to lack of overlapping common knowledge within the team, but increases learning during the work (G. Lee & Xia, 2010; K. M. Lui, Chan, & Nosek, 2008; Williams, 2000). A bug tracking system, designed to share information about problem issues in software, engendered communication and coordination between stakeholder groups and facilitated resolution (Bertram & Voida, 2010). Agile team members' communication skills are considered important to other team members and to other stakeholders (Hoda, Kruchten, Noble, & Marshall, 2010).

Communications in small work groups and dyads is both essential and natural. Research in distributed cognition, research in use of external representations,

and research in problem solving and design have all pointed to communication among team members as both essential and natural. To illustrate how involved this can be, one researcher's dissertation found an interesting example of how different people solve problems. de la Rocha (1986) examined use of everyday math skills in Weight Watchers groups. She tells of one man directed to use ¾ of his daily allotment of 2/3 cup of cottage cheese struggling with the math (although he also mentioned that he'd had calculus in college). He finally scooped out 2/3 cup of cottage cheese in a measuring cup, and put that volume of cottage cheese on the countertop. He formed it into a circle, cut a north-south and an east-west line through the circle and used 3 of the 4 sectors of the circle. He had found a way to measure out ¾ of 2/3 cup of cottage cheese.

Simple math using 3 over 4 times 2 over 3 would have yielded 6 over 12, or ½. A calculator would have made the task even simpler. When dyads and small teams are tasked with solving a problem, sometimes there are surprises contained in effective communications. This same problem has been used in the classroom among dyads of young math students tasked with finding the square area of ¾ of 2/3 of a piece of paper. Much folding and tearing ensued before students agreed on final solutions (Shirouzu, Miyake, & Masukawa, 2002). Researchers must feel free therefore to assume this is less messy than cottage cheese.

Approaches to design problem solving can be more variegated than straightforward research and examination might indicate – people are non-linear problem solvers (Cockburn, 1999) and communications among team members is necessary to converge on the best conceptualization of the problem at hand.

Conway's law states that software solutions structures reflect the communication structures of the organization which built the software, that "organizations which design systems (in the broad sense used here) are constrained to produce designs which are copies of the communication structures of these organizations." Later research supports the law in practice among problem solving and design teams (Herbsleb & Mockus, 2003; MacCormack, Rusnak, & Baldwin, 2008). Non-linear and disjunctive shared conceptualization activities use communication and coordination to facilitate agile teams' gathering, evolving and clarifying software requirements (Abdullah & Honiden, 2011), and computer-based representations and artifacts of team members' features, bugs and inquiries support customers, managers, programmers and quality assurance personnel communications and coordination (Bertram & Voida, 2010).

Communications impact knowledge sharing and learning in development teams. Face-to-face communications is rich in meaning (Daft & Lengel, 1986; Dennis, Fuller, & Valacich, 2008) and use of gestures when using external representations is common (pointing at the artifact being the most used gesture in face-to-face communications) (Corrie, 2010; Čubranic et al., 2006; Storey, Čubranić, & German, 2005). Shared visual information representation on collaborative tasks positively impacts dyads' time to completion and overall quality in a shared visual work space (Fussell, Kraut, & Siegel, 2000; Quinones et al, 2009). A simple shared interface in a group error-detection task can establish more formalized group memory and group awareness, which foster the shared cognition necessary for implicit coordination to occur (Lowry, Roberts, & Romano, 2013).

Expertise exchange, knowledge sharing and learning in small work groups is a function of strength of communication network ties (Yuan, Fulk, Monge, & Contractor, 2009) further supporting the concept that communications and coordination are relationship based, supporting therefore the concept that software design is relationship based, and with positive impact on task satisfaction.

Research by Sonnenwald (2001) finds that design teams and development teams increasingly include participants from different domains of expertise and responsibility. These agents must explore and integrate their specialized knowledge in order to contribute to the overall project, create innovative and competitive artifacts and reduce design and development costs. In this setting, communication, including integration of specialized knowledge and negotiation of differences among domain specialists, has emerged as a fundamental component of the design process (Sonnenwald & Iivonen, 1999; Sonnenwald, 1996). Pinto and Pinto examined communications in cross-functional organizational teams and found results demonstrated that high-cooperation teams differed from low-cooperation teams both in terms of their increased use of informal methods for communication as well as their reasons for communicating. Further, cross-functional cooperation was found to be a strong predictor of certain project outcomes (Pinto & Pinto, 1990).

Communication among team members impacts many factors. Facilitating communications among network designers led to better product and easier mental loadings in other research (Reeves & Shipman, 1992). Communication's impact on team performance in a social networking analysis indicates that higher centrality to the team improved project performance, while higher centrality to the project was

negatively related to project performance (Ehrlich & Cataldo, 2012). Communication safety, the perception that blunt communications will not suffer repercussions, was the single strongest factor in determining customer ratings of project performance (Hirst & Mann, 2004). Creativity and innovation among negotiating dyads is enhanced by only one member of the dyad being more creative and introducing innovative ideas or partial solutions into the negotiations (Kurtzberg, 2005).

Daft & Lengel's Media Richness Theory stipulates that the level of richness of the media involved impacts the efficacy of achieved communications and places two people face-to-face sharing a whiteboard being the most effective. In that setting is the ability to use verbal communication and gesture, along with fluid external representation. Paper memos or notes are considered the least rich media (Daft & Lengel, 1986). Further research in media richness using drawings and sketches as the sole method and also the end-product found that Media Richness theory assisted the findings (Melnik & Maurer, 2004). Communications facilitates understanding of complex interdependencies between people, artifacts and technological systems that can be often overlooked (Rogers, 2006).

Choices of communication based on richness is balanced and countered by choosing media based on synchronicity. Email might be the least rich choice available, but the timing and synchronization of the communication might be more important. Here Dennis et al's Media Synchronicity Theory has impact. Communication might be best served by synchronous or asynchronous transmission, due to communication needs of both convergence and conveyance (Dennis et al., 2008).

Research on team cognition and shared mental models recognized that direct face-to-face collaboration performs better than technology-based collaboration of separated team members, explained by increased team learning and knowledge sharing, better team sensemaking and reflexivity and improved shared team mental model (Andres, 2013).

Designing software requires communication and coordination among paired team member developers and between many other stakeholders; agile software development boosts required communications between stakeholders.

2.1.4 Cognitive Workload in Agile Software Development

Agile software development is a cognitive task involving design considerations, customer expectations, standards and practices in software development and organizational principles.

Social cognitive loading often has two offsetting effects and results can be contradictory. While working closely in a paired software development setting, one developer may reduce cognitive load by relying on their partner. The partner then may, or may not, perceive an increase in cognitive load. If the partners agree to then transfer new knowledge to an external structural representation, both may be cognitively lightened, and both may learn from the experience, while building a team mental model (Sweller, 1988).

Often the pair develops structures external to their partnership, some form of documentation, model, sketch, or diagram to help clarify the current conversation, surface ambiguities and keep notes for future references. Technical documentation, notes and artifacts bear some portion of the cognitive workload in team and group

work (Hansen & Lyytinen, 2009). Sketching offloads cognitive stresses in design activities (Paulus & Yang, 2000; Yang, 2008), reduces time to decision (Hick, 1952) and allows partners to then pick up and use other cognitive factors (Miller, 1956).

Generally, the single most helpful thing to reduce cognitive loading is to share in a social setting and to then use external structural representations to capture declared agreements achieved in the social setting.

2.1.5 Summary of Agile Software Development

Agile software development and design has become popular in industry, largely due to increased project performance on time and budget. Collaborating pairs produce better software quality, more abstract and generalized and more applicable, while reducing rework and errors. Shared mental models are built, industry and organizational standards are passed on, learning is enhanced and trust is built, leading to increased learning and higher motivation to share. Use of structural artifacts and external representations in collaborating pairs is common. Table 2-1 follows and summarizes the concepts discussed here in the Agile Software Development literature review:

Table 2-1: Agile Software Development Characteristics

Concept	Description	Relevance	Citations
Incremental & Iterative Approach	Cycling through Analysis, Design, Coding, Testing and Delivery	Foundation of Agile Methodologies	(Aoyama, 1998; Beck et al., 2001; Cockburn & Highsmith, 2001)
Moves Information	Reducing layers and time to move relevant knowledge	Reduces costs, increases learning	(Beck et al., 2001; Cockburn & Highsmith, 2001)
Increases Deliverables	Bringing working software to customers in sprints	Facilitates requirements changing	(Beck et al., 2001; Cao & Ramesh, 2008; Maheshwari et al., 2012)
Increases Software Quality	Fewer mistakes, less rework	Reduces costs	(Arisholm & Gallis, 2007; Balijepally et al., 2009; Cao, Ramesh, & Abdel-Hamid, 2010)
Design Time and Quality	Increased knowledge and cognition of pair	Reduces costs and increases quality	(Abrahamsson & Warsta, 2003; Hoda, Noble, et al., 2010)
Communications	Communications in Agile is vital to all stakeholders	Decreases time between decision and consequences	(Beck et al., 2001; Gorla & Lam, 2004; Levesque, Wilson, & Wholey, 2001; Nerur & Balijepally, 2007)
Communications	Communication skills are valued by others	Increases shared understanding	(Espinosa et al, 2007; Hoda, Noble, et al., 2010; Syed-Abdullah et al., 2006)
Cognitive Load	Reduced mental loading among developers	Reduces mental loading	(Lavallée & Robillard, 2012; Ryan & O'Connor, 2012)
Learning in teams	Learning new knowledge and new contexts	Increases quality and creativity	(Beck et al., 2001; Browaeys & Fisser, 2012; Fong Boh, Slaughter, & Espinosa, 2007)

2.2 External Representation Theory

Zhang and Norman researched cognitive tasks which required the processing of information distributed across individual internal minds and external representations (Zhang & Norman, 1994). Using the Tower of Hanoi game requiring moving disks among pegs, they examined the reported internal perspectives and information processing of the participants and the external representations of the rules required to succeed. In this setting, artifacts were able to transfer cognitive loading by serving as memory aids and reminders of required rules, and able to provide information directly perceived by participants and information used without being interpreted and formulated explicitly. Further, external representations can anchor and structure cognitive behavior, limiting rational choices to those structured to the rules of the problem. External representations can alter the nature of the cognitive task at hand by limiting choices to those structured to the rules of the problem, easing effort of compliance with the rules and allowing concentration on the more abstract aspects of the task. By decomposing a hierarchical task into its component levels, the participant can then examine the representational properties of each component. Finally, external representations and artifacts are simply an indispensable part of distributed cognitive tasks and represent previous cognitive contributions.

Agile software teams promote feedback and collaboration among members. Two common practices include external representation artifacts called the story card and The Wall (Sharp, Robinson, Segal, & Furniss, 2006). The story card tells part of

the user's story, such as "As a <role>, I want <behavior> so that <benefit>." This helps capture and document user requirements. The card itself may be written with or by the customer and is posted to the Wall. The Wall is a highly visible public listing of the story cards and the requirements to implement them. Stand-up meetings are held by the Wall, with the whole team contributing to updating the Wall and removing those cards which have been completed. Cards on the Wall change daily, and with each sprint or iteration of the project. Communication through stories is the focus, rather than the mechanism of the communication for the team. Both the story cards and the Wall have notational and social value as structural and social annotation artifacts, reducing cognitive workloads without interpretation (Sharp, Robinson & Petre, 2009). This supports Zhang & Norman's findings concerning memory aids and reduced cognitive workload.

Communication analysis of novice team programmers indicate that artifacts are often used by gesture and inflection, allowing communication without explicitly specifying communication content and bolstering collaborative learning (Čubranic et al., 2006). A social action model of information systems design highlights the importance of artifacts and external representations because the way design information is represented changes the way knowledge about the design is communicated and conceptualized (Gasson, 1999). This complies with Zhang & Norman's finding that artifacts can anchor and structure cognitive behavior, and implies their finding that external representations and artifacts limit cognitive choices and facilitate more creative and abstract approaches. Kaptelinin (1996)

points out that as tools are commonly used to manufacture better tools, artifacts are used to build better artifacts.

When the software engineer arrives tomorrow to begin work, the existing written code serves as a memory aid and reduces cognitive loading. The code provides information that can be directly perceived and used without requiring interpretation and explicit formulation. The written code anchors and structures cognitive behavior of the programmer and changes the nature of the cognitive task by becoming an indispensable component of the cognitive task system. In so doing, Zhang & Norman's findings are reproduced daily.

Further work concludes that cognitive systems are distributed across individual cognitions and memories and external representations. In these cognitive systems, artifacts help minimize cognitive complexity and facilitate users' concentration of higher-order reasoning. Difficult mental operations are mitigated by use of external, mental workload reducing artifacts, and these further allow progressive, iterative evaluation of team choices (Sharp et al., 2009). Less-experienced users will rely more heavily on external representations and artifacts, while many users will never master a complex system (Horsky et al, 2003). External representations and artifacts can serve both to establish and to destabilize boundary protocols themselves, and these artifacts can be used to push and move boundaries rather than merely spanning them (C. Lee, 2005).

In discussing agile knowledge requirements sharing, Melnik & Maurer (2004) find that as system complexity increases in a professional software project, knowledge sharing through artifacts becomes less productive and more direct verbal

communication becomes more productive, due to the increasing complexity of the artifact required to carry information about a more complex system. A longitudinal case study, focused on emergence of a software ecosystem, found that artifacts not only play a role in developers' activities, but also help other actors, like suppliers and customers, both learn about the software system and provide feedback, and help to inform and guide policy-making for organizational stakeholders (Hanssen, 2011).

Artifacts help structure cognitive tasks, often setting initial conditions in reengineering tasks and carrying information about requirements engineering and user needs. As such, external representations facilitate structural distribution temporally and help actors remember what was important yesterday (Hansen & Lyytinen, 2009). This both reduces cognitive loading and smooths transition as teams finish one iteration and begin the next. Developers may remember previous innovative or opportunistic design possibilities across time as members reconsider requirements elicitation directions of previous efforts while collectively interpreting and updating existing external representations (Guindon, 1990; Halverson, 1994).

2.2.1 External Representation Theory in Agile Software Development

Software developers use informal tools such as whiteboards, story cards and stand-up meetings, and written code itself, as well as more formal sources like bug databases and design documentation in order to build and maintain awareness of team progress and project milestones (Biehl & Czerwinski, 2007; Sharp et al., 2009). All the tools mentioned here are external representations and artifacts. Williams' dissertation (Williams, 2000) discussed using artifacts from early in the design

process in order to achieve the goal of analysis and capture the requirements and some details about the requirements in artifacts. More recent research into the annotation possibilities of artifacts by Nobarany (2012) suggests that artifacts which support scalable, flexible and open collaborative workflows facilitate collaboration.

A representational determinism is suggested - the form of a representation determines what information can be perceived, what processes can be activated, and what structures can be discovered from the specific representation (Zhang, 1997). Woods posits that the presentation of artifacts include influences on the cognitive work required to solve the problem, referred to as the representation effect (Zhang & Norman, 1994). Here Woods refers to the design interpretation for better performance – to develop technological artifacts that shape cognition and collaboration. Artifact designs then become hypotheses, shaping cognition and collaboration and should be tested as such (Woods, 1998).

Artifacts and external representations are commonly used in software development, from working code to external coding standards or organizational preferences (Beck et al., 2001; Cockburn & Highsmith, 2001; Čubranic et al., 2006; Dekel & Herbsleb, 2007; Froehlich & Dourish, 2004; Scaife & Rogers, 1996; Sedig & Parsons, 2013; Storey et al., 2005).

2.2.2 Mind Maps

Mind maps are a type of spider diagram – they branch off of a word or phrase in the middle of the diagram, each branch representing another associated word or phrase and each branch capable of having sub-branches, and sub-sub-branches off

the sub-branches. Mind maps serve as external representations of cognitive associations, e.g., if the central phrase is “movies,” one branch might be “popcorn,” often associated with going to the cinema.

This type of diagram can be drawn on a single large piece of paper, can be used by one person or by a group, can be added to or branches marked over and deleted, and can serve as a common visual representation of depths of branches’ and sub-branches’ connections and the hierarchical relationship between them. Of course, there are now many software programs that produce mind maps and many of those are free and open-source. There are many ways to produce a mind map.

> “In mind mapping the problem is stated in the centre of a piece of paper with a circle around it or an image. Ideas are then brainstormed in spokes originating from the central idea. Whereas traditional thinking opts for columns and rows, this working out from a core idea suits the brain’s thinking patterns better. It encourages thoughts to flow more smoothly.” (Zampetakis, Tsironis, & Moustakis, 2007)

The focus on key words “can foster more expansive connections” and keeping to a single image or graphic “allows one to see the entire picture at once and perhaps stimulate additional associations” (Budd, 2004).

Below is an example of a mind map by Abdul Rahman, taken from the internet, found at 3.bp.blogspot.com, entitled, “What is a mind map?”

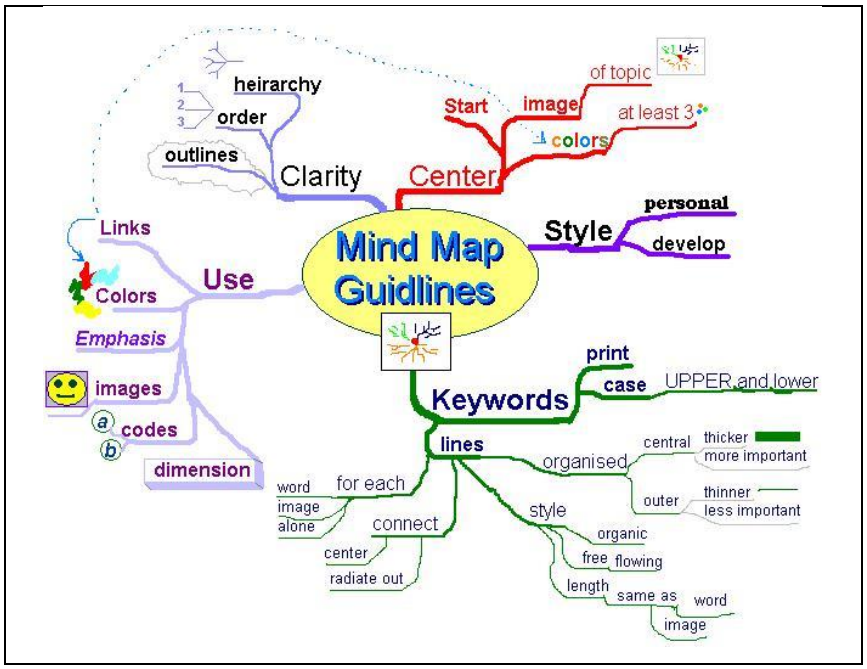

Figure 2-2: Mind Map example

Mind mapping can be used to start a group process of decision framing, as mind mapping captures an emergent, team representation of a problem. Mind maps are simple to construct (Eppler, 2001) and serve as external representations in, for, and collectively by, a team or group in a distributed cognitive task (Zhang & Norman, 1994). Mind maps are simple to understand and need no external references to follow the connections and the hierarchies illustrated (Beel, Gipp, & Stiller, 2009).

Mind maps represent information in nonlinear ways that more "closely corresponds to how the mind actually organizes information. One of the main distinctions between information and knowledge… is that knowledge is arranged in networks with meaningful connections between clusters of information. While information can be transmitted as is, knowledge needs to be constructed as a web of meaningful connections" (Rosenbaum, 2003). Mind mapping as a team or group constructs this knowledge web using information from all members, and in a connective web built by the whole team.

The act of mind mapping serves as a social lubricant in a newly formed team, with the mind map itself serving as a common and salient artifact each member can use as a common point of reference in brainstorming and getting to know other team members in the process. The act of mind mapping allows each member to contribute from their strengths without any pressure to lead the conversation. Mind maps allow direct elicitation and collectively approved representation of issues in the problem at hand (Petersen & Wohlin, 2010). Mind mapping in MBA programs to assist studying case studies enhanced creativity and productivity in participating students, facilitated learning and fostered efficiency in group work (Mento, Martinelli, & Jones, 1999).

Mind maps facilitate expert search, document summarization, keyword-based search engines, and document recommendation systems by determining word relatedness in data mining analysis and design (Beel et al., 2009). Use of mind maps with engineering students increased learning and promoted cohesion in student teams, validating a strategy aimed at reducing time necessary to incorporate creativity in engineering curricula (Zampetakis et al., 2007).

The cognitive issues of problem solving as a design team require a team mental model to be built or to emerge. Mind mapping helps build a team cognitive foundation and team cognition helps explain team performance (DeChurch & Mesmer-Magnus, 2010).

Team mental models are fashioned as groups work together to perform the task. Team mental models have been compared to "lossy compression," where some information richness is lost as the unifying mental model is built, such as a moving average which can show trend but not individual points of data. This mental model is

therefore never perfect or complete, but can be improved (the average can be raised) through concerted, increased efforts in sharing (Carley, 1997). Mind maps, especially in the beginning stages of design team work, help coordinate and build team mental models in order to reach public, team support for problem framing and solution building (Gallupe, DeSanctis, & Dickson, 1988).

Mind mapping as an individual or in a group reinforces clarity of thought upon reflection, serving as a memory aid and reducer of cognitive loading (Beel et al., 2009; Zhang & Norman, 1994). Zhang and Norman (1994) point out three aspects of external representations in distributed cognitive tasks that have effect here.

First, external representations serve as memory aids. "…the goal problem states didn't need to be memorized, because they were placed in front of the subjects either by diagrams or by physical objects. This is the most acknowledged property of external representations." By serving as memory aids, mind maps refresh working memory and reduce cognitive loading.

Second, external representations in distributed cognitive tasks become part of the representational system of any distributed cognitive tasks. The representations are "a direct reflection of the nature of distributed cognitive tasks, which require the processing of information distributed across internal and external representations."

Third, external representations as artifacts in a system change the nature of the task itself. Zhang & Norman (1994) discuss two perspectives of the overall system, one of the system itself and another of the individual in the system. From here, they posit that the system overall, relying on both internal and external representations, sees external representation as facilitating the task itself by both

serving as a memory aid and by facilitating the distribution of cognition outside the individual.

> "From the system's view (internal + external representations), external representations can make a task easier; from the person's view (internal representations only), external representations change the nature of the task. Though the two versions had the same abstract structure, their cognitive processes were different. Nevertheless, when considered as systems, I1-E23 was much easier than I123." (Zhang & Norman, 1994)

The individual, interacting with the external representation, then sees the nature of the task differently as external representations change the cognitive structure of the overall task.

The effect of these three different aspects is that the external representations, and our current focus is a mind map diagram, emerge from cognitive sharing, both in individuals and in groups. This external representation facilitates the task, but changes the mental nature of the task also. New cognitive connections are made, and this illustrates the effect the authors refer to as changing perceptions in the members, which then affect internal cognitive representations and processes.

What we perceive changes what we then think.

As a simple example, we point to traffic lights. While driving, we might notice a green light ahead change to yellow. This perception process of external representation in our system triggers an internal cognitive process – we remember that the red light will come soon. Other cognitive processes are initiated, and we might remember that this light, at this intersection, in this traffic pattern, changes slowly. This presents a decision needing urgent attention – should we speed up to make it through the intersection before the light goes red, or do we slow down and

expect to stop? The external representation given to us in this example is outside our control, but we must interact with it nonetheless.

Mind maps serve as external representations in a distributed cognitive environment. The individual can use mind mapping to serve as an external representation of internal thought processes, and groups can build mind maps to serve as external representations of collectively approved conceptual connections emergent from the group for the task at hand.

2.2.2.1 Working with Mind Maps

Mind maps are simple to construct, cognitively undemanding to understand, and easy to remember. Mind maps reduce cognitive loading and serve as memory aids, keeping relationships between concepts clear and refreshed. Mind maps bring context to information, forming knowledge. Mind maps enhance creativity by allowing nonlinear thought processes. Mind maps help team sharing and building team mental models, illustrating group-ratified representations of issues at hand. Mind maps built individually by team members can then be blended, and therefore serve to elicit explicit information sharing when teams start and members have little knowledge about other members, including others' experience in teams.

Other mapping techniques for building or sharing cognition do exist. Cognitive maps were originally developed to display mental models of associations between two or more concepts. Later usage included hierarchical levels. Mind maps help build cognitive maps, are more finely grained when doing so, and force explicit

hierarchy in building them. Thusly, mind maps include more cognitive information than cognitive maps (Eppler, 2006).

Concept maps show relationships among or between concepts, associations among equals. Concept maps are based on deductive reasoning for meaningful learning; e.g., Porter's Five Forces model and Unified Modeling Language (UML). Concept mapping was used to predict problem-solving performance by computing information uncertainty (here as EntropyAvg) and findings correlate these predictions to individuals' problem-solving performance with high predictive power (Hao et al, 2010). However, concept maps contain no intrinsic hierarchy. Mind maps bring both intrinsic hierarchy and many more levels of cognitive association (Eppler, 2006).

Mind maps allow nonlinear cognitive connections and thus spur creativity in outlining solution possibilities. Guindon (1990) posited that opportunistic thoughts in design emerging through top-down decomposition were suitable and beneficial for ill-structured design problems. Halverson found that analyzing and using existing components or aspects of systems scheduled to be replaced can reveal new uses for existing technology and reduce cognitive costs of transitioning to new systems (Halverson, 1994). Mind maps are useful in facilitating team members' efforts in beginning to form a cohesive unit and in working out team conflicts by allowing explicit member input in each member's own area of strength while allowing each member to gain understanding of other members' strengths, and combinative working toward a team-approved, publicized solution structure.

Mind mapping has been used across many disciplines and has been found to be useful for beginning exploratory research analysis of both endogenous and exogenous factors in systems under study. In fact, one researcher who has been using mind maps in psychology for years has established a link between mind maps, concepts maps and structural equation modeling (SEM), pointing out that the combination of concept map and mind maps can form a direct analog of complex mathematical structural equations in modeling (Huba, 2013). Here the mind map represents the estimation of the measurement model parameters and the concept map represents the structural model. This analog has been used in at least one professional journal research paper which used a three-stage approach to modeling transportation policy impacts on environment, society, economy, and energy (Ülengin et al, 2010). In this paper, cognitive mapping with hierarchies is used as a first stage to assist in conceptualization for the SEM model. (Cognitive mapping with strict hierarchies is mind mapping.) Another paper analyzed cognitive maps to help teams structure issues and problems in preparation for statistical modeling and simulation modeling of "messy" problems (Eden, 2004).

Using mind maps in project management practice leads to "whole brain" solutions by combining the analytical and logical business and engineering school left-brain with the art and design school right-brain approach to imagination and creativity (Mantel, Jr., Meredith, Shafer, & Sutton, 2011), increasing both quantity and quality of ideas generated across the entire project team. Using the example of a part-time MBA program whose school wanted to improve their program, the authors illustrate how to build out a mind map. Starting with the project's objective of

"Generate Ideas for Breakthrough Performance in Part-Time Program," the project team filled out 4 sub-branches and then proceeded from there to elucidate further concerns among the four sub-branches. Mantel, Jr. et al point out that when the mind map is being built out, key words and images are used to engage more of the mind than outlines and sentences when diagramming tasks. Upon completion, the mind map represents a socially-approved diagram of key concepts and linkages among them.

This is akin to a project management tool called the Work Breakdown Structure (WBS). The WBS extracts all pertinent tasks by task level, numbers each task, shows responsibility for each task, shows project milestones and helps more accurately estimate costs, risk and time for budgeting. Mind maps can be used to build WBS in a project, and facilitate more enthusiasm for the work by encouraging higher average participation from all members. Mind maps in project management help to "solve the right problem, as well as solve the problem right" (Brown & Hyer, 2002). Creating a Work Breakdown Schedule (WBS) by mind mapping it first produced faster and more detailed results compared to producing the WBS by outlining it. Also, enthusiasm for the project was raised and remained higher than average, with less opportunity for unproductive conflict and increased chances for creative inputs (Brown & Hyer, 2002).

2.2.3 Summary of External Representation Theory

Artifacts and external representations are constructed to help as working memory aids and reduce cognitive loading and to engender conversation and

communications and increase communications efficiency among group and ream members. Artifacts and external representations allow offloading mental structure, facilitating more abstract thinking. External representations change the relationship between the user and the task, anchoring and structuring cognitive choices to rational constraints. Artifacts and external representations help to decompose a hierarchical task into component sub-tasks, allowing examination of representational properties of each component. External representations are an indispensable part of structural cognition in a distributed cognition system, and help serve as temporal reminders of previous cognitive choices.

Mind mapping allows both individuals and teams to externalize cognitive thoughts and reference these maps later to decrease mental loading and extend conceptualization during design, especially in framing an ill-structured problem. Mind maps require the linking of concepts in a hierarchical way. Mind maps are easy to understand and to construct, requiring no intensive training to make or use them. As artifacts, mind maps fulfill the requirements of valuable external representations. Further, mind maps bring the creativity-enhancing aspects of sketching and diagramming to the analytical, task-oriented aspects of both structural equation modeling and project management, engaging the whole brain in the effort.

Table 2-2 is presented below, using “AER” to represent Artifacts and External Representations, and summarizing the literature review of External Representation Theory:

Table 2-2: External Representation Theory Characteristics

Concept	Description	Relevance	Citations
AER as memory aid, reducing cognitive load	AER reduce cognitive load and help serve as memory aid	Reducing cognitive loading helps designers in cognitive tasks	(Mangalaraj et al, 2014; Sharp et al, 2009; Zhang & Norman, 1994)
AER as extended cognition	AER allow efficient communication	Software developers communicate better	(Čubranic et al., 2006; Gasson, 1999; Zhang & Norman, 1994)
AER as temporal structure and anchor cognitive behaviors	AER change over time as designers iterate changes	Facilitates changing requirements and reminds of previous choices	(Kaptelinin & Nardi, 2006; Storey et al., 2005; Zhang & Norman, 1994)
Mind Mapping diagrams as AER	Capturing concepts and inter-relationships among tasks and aligning cost and time budgets	Easy to learn, easy to use, useful and detailed	(Beel et al., 2009; Budd, 2004; Eppler, 2006; Eppler, 2001; Petersen & Wohlin, 2010; Rosenbaum, 2003; Zampetakis et al., 2007)
Mind Mapping hierarchy in AER	Forces hierarchical choices without forcing task priorities	Requires developers to start, to begin, to choose	(Beel et al., 2009; Buckingham, Ahmed, & Adams, 2007; Dix, 1994; M. J. Eppler, 2006; Zampetakis et al., 2007)

2.3 Distributed Cognition

Distributed cognition is a branch of cognitive science that proposes cognition, information and knowledge are not confined to an individual; rather, it is distributed across objects, individuals, artifacts, and tools in the environment.

Distributed Cognition originated with Edwin Hutchins, an anthropologist and cognitive psychologist. A lifelong competitive sailor, he studied how navigation is coordinated on US navy ships around San Diego. He observed the necessary knowledge and cognition to operate a naval vessel does not exist solely within one's head; knowledge and cognition is distributed across objects, individuals, artifacts, and tools in the environment.

Distributed cognition is a cognitive theory which posits that data, information and knowledge not only lie within an individual but are also contained within the social and structural environment of the actor. Distributed cognition allows a person to "distribute" their knowledge throughout external artifacts as external representations of internal perspectives and also throughout social structures and ties. Distributed Cognition has illustrated how cognitive tasks can be spread among a team or group, allowing input to a team from each member, and each member changing and shifting their own individual perspective of the team's goals and progress over time due to inputs from other team members (Yvonne Rogers, 1997).

Distributed cognition uses external representations, artifacts and tools as well as internal representations. An easy example of this is a person using a handheld calculator – the cognitive process of "driving" or controlling the calculator is based inside the person's brain. The external plastic "brain" artifact will not know what the

person using wants to accomplish until the person interacts with it. Only the internal representations of the task, fitted inside the person, can direct the external representations, such as pushing the proper buttons on the calculator. Further, many people can testify that a simple calculator may require multiple stages of usage to solve a particular problem, with the person using external pencil and paper to scribe partial answers or accuracy checks along the way to the final solution. This is a simple distributed cognition system.

When a group coalesces to solve a problem, each team member requires inputs from other team members in order to begin constructing a common or team cognition, a team mental model (Bossche et al 2010). Dong et al (2013) speak of "the emergence of sharedness of the team mental model" over time. Andres posits "team cognition as a set of mental processes and intra-team communication exchanges that facilitate team learning, reflection, and shared understanding" (Andres, 2013). Teams must make sense of the problem and its context, and distributed cognition contributes to team sense-making.

In building a shared mental model under distributed cognition, Ashmos & Nathan (2002) state a "particular mental model or structure that reflects users' understanding of a system and how it works can help to direct our thinking toward new and different uses for teams that can ultimately help make teams and their organizations more flexible and effective." DeChurch & Mesmer-Magnus (2010) posit that "team mental models ought to improve team process." Banks & Millward (2000) focus on shared mental models as direct artifacts of distributed cognitive processes – a "framework for studying shared mental models is which the model is

considered to be distributed amongst the team." Kozlowski & Ilgen (2006) see distributed cognition where "team mental models refer to knowledge structures or information held in common." Distributed Cognition can be seen as humans using tools and the tools mediating the hierarchically organized motives, goals and conditions of the humans. Further, these tools continually undergo various levels and types of development to increase their utility in the system (Kaptelinin, 1996). Shared or team mental models are built while underway and making progress, and revised as needed when facing shifting goals and plans. Distributed Cognition theory is based on this evolutionary concept of problem solving.

2.3.1 Distributed Cognition in Agile Software Development

Agile software development, especially in pairs, already uses the social and the structural aspects of distributed cognition, the dual approach to total team cognition. Gasson posits three theoretical lenses to understand how individuals and groups frame IS problems and solutions. These include socially-situated cognition, socially-shared cognition, and distributed cognition. Here the author illustrates the analytical levels of organizational IS design and adaption and asks whether cognition is a property of individuals, groups or technological systems (Gasson, 2004). Rogers & Ellis (1994) examined individual, social and organizational (environmental) settings as influences on seemingly individual efforts and work, and conclude that Distributed Cognition is the better framework of analysis rather than discrete cognition, sociological and organizational lenses.

The benefits of using Distributed Cognition in software design and development team research include a systemic view of software teams and their activities, abstraction of artifacts as representational media, capture of software development techniques as transformations and representations of information coordination and communication, and consideration of individual and organizational learning (Aranda & Easterbrook, 2006). The artifacts and external representations used in distributed cognition cannot contain distributed cognition without individual cognition and dissemination - each human shares information and knowledge of how to interpret that information by using the tools available (G Salomon, 1993). Framing software engineering requirements as a Distributed Cognition environment indicates fruitful analysis of agile methodologies study (Hansen, Kharabe, & Lyytinen, 2013). Distributed Cognition is a valid mechanism for use in studying the intricacies of pair software development.

2.3.1.1 Communications in Distributed Cognition in Agile Software Development

Mature eXtreme Programming (XP) teams are highly collaborative and rely heavily on interactions between team members and their support environment (Sharp & Robinson, 2006). Agents in a distributed cognition setting benefit from communicative awareness of a dynamic and collaborative process that binds agents together in a moment-by-moment basis (Stanton et al., 2006). Members of a distributed cognition system benefit more from face-to-face communication than from collaborative technology, facilitating team cognitive functions such as team learning, team reflexivity, and shared mental model development (Andres, 2013).

Blandford & Furniss (2006) studied a large ambulance control center using distributed cognition and found that external representations elicited feedback from members concerning changes in practice and method. Davern et al (2012) examined cognition in IS which extends beyond the individual to teams, communities and systems as units of analysis. Findings include how important communications become in an iterative improvement setting. Communications include social and structural distribution by personal communications and by use of external representations. Distributed cognition communication in software development includes both social and structural cognition.

Flor & Hutchins found support for communication in a distributed cognition setting by pointing out increased reuse of system knowledge, sharing of goals and plans, efficient communication (illustrated by lack of rigorous articulation and full specification of communications) in verbal communications among team members which was still effective (Flor & Hutchins, 1991). Even something as simple as gestures by team members has been found to be effective communication, especially gestures referring to external representations (Corrie, 2010; Gutwin & Greenberg, 2001). Further support for increased communications in a distributed cognition system also includes increased levels of talk when facing ambiguous aspects of joint plan segments, shared memory for previous alternatives and old plans, and division of labor among team members, where the navigator actually uses the driver as a "smart keyboard" with feedback, and the driver uses the navigator for architectural and strategic feedback to assure the larger fit of the current solution (Flor & Hutchins, 1991). Increasing task complexity engendered increased communications

among dyadic students (Iiskala et al, 2011; Shirouzu, Miyake, & Masukawa, 2002). Guindon & Curtis (1988) found support for opportunistic design possibilities in software engineering communications using design schemas, design heuristics, and design methods among team members. Further support for opportunistic opportunities in design possibilities was found in research on air traffic control systems rework in a distributed cognition setting. (Halverson, 1994) Distributed Cognition communication systems are robust, adaptable and flexible; "agile," if you will.

Hutchins (2000) points out that distributed cognition systems require communications between social agents and team members in the system as well as communications among structural members (external representations and artifacts) of the system. Further, these communications facilitate processes being distributed across time in such a way that the products of earlier events can transform the nature of later events. This provides further support for Zhang & Norman's (1994) position that artifacts and external representations transform the cognitive nature of the task (Hutchins, 2000).

2.3.1.2 Cognitive Workload in Distributed Cognition in Agile Software Development

Distributed Cognition as method uses protocol analysis, neural network simulations, and laboratory experiments as well as case studies and empirically based surveys and systemic analysis of practitioners. Challenges include integrating concepts from organizational sciences and social sciences with cognitive analysis of

representational states. Artifacts supporting coordinating activity, significant collaboration among team members and influences of the physical environment on members choosing activities to pursue in a distributed cognition setting were among the findings in a medical team technology environment (Rajkomar & Blandford, 2011).

One early paper regarding cognitive workload in collaborative software development found that standard cognitive science measures applied to software design and development activities (Robillard et al, 1998). Treating software development activity as information processing, requiring both convergent shared mental models and divergent creative searching, finds that mental models actually diverge over time as role differentiation increased (Lavallée & Robillard, 2012).

Perry & McCredie found that examining the informational content of activities and treating them as information processing events, distributed cognition supports user-centered design choices, applying a cognitive engineering approach to development and implementation of technology to mediate work (Perry & Macredie, 1999). Cognitive workload in Distributed Cognition systems in Agile software development require high levels of communication to reduce cognitive workload, yet include media rich in communication and are therefore fruitful for fostering fecund focus.

2.3.2 Summary of Distributed Cognition in Agile Software Development

Distributed cognition theory posits that individual cognition can be extended in a team setting by both social cognition, such as small teams or collaborating pairs,

and by structural cognition, such as artifacts and external representations. This distributed cognition exceeds that of any individual, even in a simple setting of using pencil and paper to extend one person's cognition in framing and solving a math problem. Distributed cognition theory fits well with agile software development due to extended use of artifacts, external representations and recognition of the expected and required communications between actors in a distributed cognition environment. Cognitive workload in a distributed cognition environment both increases and decreases whenever learning takes place and when actors offload onto cognitive artifacts and external representations. Table 2-3 follows and summarizes the concepts discussed here in the Distributed Cognition literature review:

Table 2-3 Distributed Cognition Characteristics

Concept	Description	Relevance	Citations
Cognition is extended Socially among team members	Teams extend individual cognition socially; teamwork increases solution alternatives	DCog fits Agile Methodologies small teams and pairs	(Beck et al., 2001; Cockburn & Highsmith, 2001; Dybå & Dingsøyr, 2008; Flor & Hutchins, 1991)
Cognition is extended Structurally among external artifacts	Artifacts and external representations are structural cognition	Agile Methodologies use external representations for team memory	(Aranda & Easterbrook, 2006; Flor & Hutchins, 1991; G Salomon, 1993)
Distributed Cognition focuses on systems	DCog uses team members and artifacts & tools as system for analysis	Agile Methodologies uses team and tools as system	(Andres, 2013; Banks & Millward, 2000; Bossche et al., 2010; Dong et al., 2013)
Communications in Distributed Cognition	Distributed Cognition relies on communications	Agile pairs rely on communication	(Beck et al., 2001; Espinosa, Kraut, & Lerch, 2001; Hoda, Noble, et al., 2010)
Cognitive workload in Distributed Cognition	Cognition distributed socially and structurally	Pairs in Agile distribute cognition socially and structurally	(Fong Boh et al., 2007; Lee & Xia, 2010; Maheshwari et al., 2012; Roberts, Hann, & Slaughter, 2006; Vidgen & Wang, 2009)

2.4 Literature Review Summary

A narrow ontology has been constructed here, concerning individual and paired programmers in framing ill-structured problems and designing solutions for those problems. This ontology is correspondent to existing theory and practices, coherent to the environment which it describes, and constructive for

extending the body of knowledge by supporting epistemological inquiry into socially cognitive pair design problem framing and solution propositions using structurally cognitive artifacts and external representations.

Agile methodologies are popular in industry and are increasingly used. Pair software development is a standard "best practice" in industry which uses pairs as collaborative partners, providing social cognition. However, paired programming still has not risen to the quality and productivity level of the first-best individuals. How can we help paired software developers perform better?

Artifacts and external representations abound in this environment, e.g. story cards, flip charts, bug databases, working software code, electronic tracking and CASE tools, and provide externalized structural cognition. Communications in pair software development support collaboration among team members and facilitate shared team mental models.

Distributed cognition theory fits well with existing industry practices by supporting existing use of social and structural cognition and provides opportunity to test interactions between social and structural cognition.

Mind mapping facilitates communication and coordination in small groups, especially those teams working on starting to frame ill-structured problems. Mind mapping enhances concept capture individually and in teams, provides peer-approved hierarchy among concepts, and serves as artifacts and external representations to guide problem framing and solving.

This research serves well as a basis for inquiry into distributed cognition among software designers, a much understudied area of research. By using mind maps, shared structural cognitive artifacts are created for current use and expanding future research possibilities. Perhaps mind mapping can contribute to communication and coordination among paired software developers and elevate performance in paired software designers.

This research will contribute to practice by illuminating theoretical underpinnings of industry practice and highlighting possible paths to improved software design. The benefit of net gains from pair development that approach the work of the best individual developer would be invaluable for practitioners

Chapter 3

Research Model and Hypotheses

This chapter outlines the research model designed to answer the three research questions and delineates the hypotheses included in the research model.

3.1 Research Model

The Research Model is pictured as follows, with indicated hypotheses:

Figure 3 – The Research Model

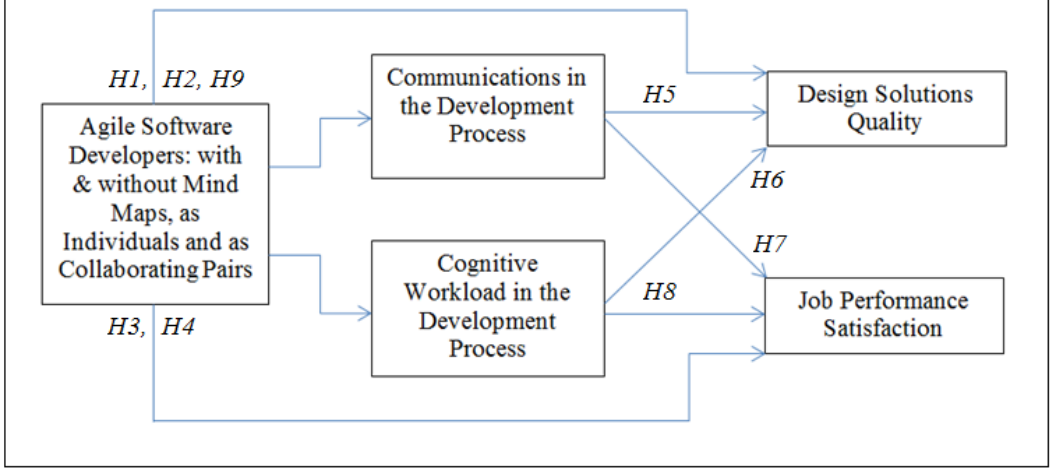

Figure 3.1: The Research Model

The model above shows Agile software developers working with and without Mind Maps, as individuals (measured as nominal pairs) and in collaborating pairs.

Partial mediation is expected by communications in the development process. Communication may be social or structural. Collaborating pairs will experience increased social communication, but individuals creating artifacts, such as mind maps, can increase self-communication. Structural communication is inherent in the creation of external representations, as ideas and thoughts must be represented in the external structure. Cognitive workload in the development process will be affected by both social and structural conditions, leading to partial mediation of cognitive workload effects.

Differences in design solution quality and in job satisfaction are hypothesized among software designers working with and without mind maps, as individuals and in collaborating pairs.

The Research Questions for this study are repeated here:

1) Do software designers working in pairs have a greater effect on the quality of software design than do individuals?

2) Do mind maps assist software developers in a design task to increase the quality of design, among individuals and in pairs?

3) Do mind maps assist collaborating pairs in a software design task equal the work of the best individuals?

We believe the research questions above have been supported by the literature review and serve to bolster the following hypotheses.

3.2 Hypotheses

3.2.1 Impact of Pair Designing on Software Quality

Collectively approved solution actions, based on communication and interaction between internal cognitive processes of individual members of a collaborating pair, evolve and emerge from interaction of complementary skills and experience in a collaborating pair of software designers.

Extant literature is rife with examples of increased software quality. Earlier studies include (Williams, Kessler & Cunningham, 2000), (Cockburn & Williams, 2000), (Nawrocki & Wojciechowski, 2001), (Lui & Chan, 2003), (Aiken, 2004). Williams et al (2000) found that among both experienced professional programmers and advanced undergraduate students, software quality, measured as better algorithms with fewer defects, increased in pair programming situations. Cockburn & Williams (2000) supported the idea that pair programming produces better design quality by forcing members to communicate and defend their solutions before finalizing the design. Nawrocki & Wojciechowski (2001) highlighted that pair programming by students produced better quality code and took less time than two serial individuals. Lui & Chan (2003) reported that pair programming produces better quality than individuals when focused on design. Aiken (2004) highlights that pair programming brings paired review and

debugging, producing fewer defects in the software code, in an early review of extant pair programming literature involving both professional programmers and students.

More recent studies of pair development programming and design also indicate increases in quality of work among paired team members (Cao et al., 2010), (Giri & Soni, 2013), (Mangalaraj et al., 2014). Cao et al (2010) found in a study of 9 organizations that pair programming lowered the cost of design quality by 36% averaged across all projects in the study. Giri & Soni (2013) highlighted the increased quality of software written by professional programmers using paired design and development. Mangalaraj et al (2014) examined software design in a distributed cognition setting and found that paired development increased the quality of design solution.

The combined, communicating cognitive strengths of more than one mind increase the chances of finding better solutions (Guindon, 1990). Flor & Hutchins (1991) found support for paired developers communicating efficiently in socially distributed cognition, without full specification of thoughts or sentences, searching through a larger solution space, sharing goals and plans, and shared memory of old plans.

However, most studies indicate that the software quality of a collaborating pair will exceed that of the second best member of a nominal pair and not exceed that of the first best member of nominal pair. The quest for the elusive prize of

paired programming quality exceeding that of the first best member of a nominal, non-collaborating pair continues without support from literature. Therefore:

> H1: Design solution quality of a collaborating pair will be higher than that of a second best member of a nominal pair in a software design task.

3.2.2 Impact of Mind Maps on Software Design Quality

Artifacts and external representations enable recall of pertinent knowledge, define and lend clarity to ill-structured design problems, facilitate shared representations, evoke schemas and conceptual operators germane to the design under consideration and lead to greater team collaboration and reflexivity (Čubranic et al., 2006; Gasson, 1999; Hansen & Lyytinen, 2009; Hanssen, 2011; Helen Sharp et al., 2009; Zhang & Norman, 1994). Čubranic et al (2006) found that designers who externalized expectations and defended decisions completed tasks faster and with less rework. Gasson (1999) highlighted design experts communicating design domain knowledge while domain experts communicated IS domain knowledge in a large project for Fujitsu Telecommunications, and both activities increased software design quality. Hansen & Lyytinen (2009) supported distributed structural artifacts in professional IS design leading to higher quality and less refactoring. Hanssen (2011) points out organizations which use external representations react faster and with more quality in their software ecosystem; here the external artifacts serve well as reminders of previous decisions and possible alternatives. Sharp et al (2009) supported using story cards and The Wall

among professional software developers to facilitate shared understanding of software requirements and produced higher quality software. Zhang & Norman (1994) point out that cognitive tasks benefit from external representations by reducing cognitive loading and enhancing understanding of the solution requirements.

Flor & Hutchins (1991) also found that structurally distributed cognition in paired development served as memory aids for previous decisions, enhancing the overall quality of the final solution reached by the pair.

Software designers using mind maps can more easily plan ahead to resolve design conflicts, integrate diverse standards from industry, from organizational expectations and history, and induce a higher quality of solution. Therefore:

> H2: Design solution quality will be higher when mind maps are used during software design.

3.2.3 Impact of Mind Maps on Task Satisfaction

Task satisfaction is one of the determinants of good software design (Aladwani, 2002). Using external representations like mind maps lowers task cognitive complexity and boost participants' confidence, leading to higher job or task satisfaction (Balijepally, 2006; Beel et al., 2009; Cockburn & Highsmith, 2001; Cross, 2001; Shah et al., 2001; van der Lugt, 2002; Zampetakis et al., 2007). Balijepally (2006) found that all workers in collaborating software developer pairs were more satisfied than individuals in nominal pairs. Beel et al

(2009) posited that use of mind maps for cognitive knowledge work heightens confidence in proposed solution, boosting satisfaction. Cockburn & Highsmith (2001) highlighted that the cost of moving information is lower in agile software development pairs, increasing knowledge in the domain and boosting confidence, thus increasing task satisfaction.

Cross (2001) examined sketching in design science, using illustrations and diagramming to increase cognitive externalization and sharing information, knowledge and plans. Shah et al (2001) discussed collaborative sketching, increasing confidence in proposed design solutions, leading to higher task satisfaction. van der Lugt (2002) found that creativity and innovation through group sketching improved members' satisfaction in the task and increased confidence in the solution. Zampetakis et al (2007) highlighted that conjoint analysis of leading students to use mind maps derived support for student's increased task satisfaction when using mind maps to provide engineering solutions.

Small & Venkatesh (2000) used a cognitive-motivational model of decision satisfaction to illustrate that increased confidence improves task satisfaction. Therefore:

> H3: Task satisfaction will be higher when mind maps are used in a software design task.

3.2.4 Impact of Pair Designing on Task Satisfaction

Pairs working on tasks report higher satisfaction than individuals working alone (Balijepally et al., 2009; Cockburn & Williams, 2000; Mangalaraj et al., 2014; Nosek, 1998; Pedrycz, Russo, & Succi, 2011). Balijepally et al (2009) highlighted increased satisfaction from paired developers, along with increased confidence equal to that of the first-best performers of nominal pairs. Cockburn & Williams (2000) found that both professional and student programmers who worked in pairs were more satisfied and more confident in their solutions. Mangalaraj et al (2014) pointed out that small groups and collaborating pairs have a higher job satisfaction than individual and nominal pairs. Nosek (1998) examined professional programmers and found that both confidence in solution and process satisfaction were significantly higher in those who worked in collaboration with others, with much smaller standard deviations in the measures. Pedrycz, Russo & Succi (2011) found that among professional programmers, pair programming has actually a strong positive effect on satisfaction.

de Vries et al found that knowledge-donating and knowledge-collecting behaviors were indicative of eagerness-to-share and willingness-to-share attitudes. Communications in collaborating pairs, small groups and teams increased job satisfaction, which then increased communications overall. (de Vries, van den Hooff, & de Ridder, 2006)

Collaborating pair software developer designers communicate and build shared understanding, increasing task engagement, reducing complexity through sharing and contributing to higher overall task satisfaction. Therefore:

> H4: Average level of task satisfaction will be higher among collaborating pairs compared to average task satisfaction level of nominal pairs.

3.2.5 Communications in Software Development Design Tasks

Effective communications in design tasks enhances good design (Adamczyk et al, 2007; Fischer & Ostwald, 2003; Giri & Soni, 2013; Kleinsmann & Valkenburg, 2008; Kokotovich & Purcell, 2000; Mangalaraj, 2006; Shah et al., 2001; Sonnenwald, 1996). Adamczyk et al (2007) examined collaboration tools such as sketching and collaborative writing in design to analyze the agility of these tools in rapidly changing design specifications scenarios, finding that communication by the design teams were the key factor in achieving design success. Fischer & Ostwald (2003) posit that communication, involving the communication media, are essential to design and design practice. Giri & Soni (2013) highlighted that better programmers (ranked by the Programmer Aptitude Test, PAT) used better communications, more efficient and more effective, than junior programmers. Kleinsmann & Valkenberg (2008) researched designers in collaborative design, regarding communications, to find that face-to-face communication led to better quality and more effective design solutions through

increased shared understanding. Kokotovitch & Purcell (2000) supported findings that visual communication students produced more creative ideas using industrial design sketches and group communications than law students when both groups were compared to industrial design students, again illustrating the power of visual communication in design studies. Mangalaraj (2006) illustrated that in software development, group and pair communication raised the performance levels of the groups. Shah et al (2001) found that collaborative sketches in design work carry more communication than originally intended, opening discussion for further shared understanding and leading to more detailed and higher quality solutions. Sonnenwald (1996) highlighted that communications in cross-discipline design teams support knowledge exploration, collaboration and task completion by negotiating differences across organizational, task , discipline and personal boundaries.

Flor & Hutchins (1991) point out that software development pairs communicate through both social and structural cognitions in a distributed cognition system, achieving effective and efficient communication, including gesture and inarticulate, underspecified sentences in conversation. Daft & Lengel (1986) posit that face-to-face communication supported by sketch, diagram and gesture is the richest possible media with which to communicate. Corrie (2010) found that researchers often use gesture instead of words to communicate when using diagrams and collaborative illustrations of research paths and posited

solutions. Communications among software developers in a design task in collaborating pairs will explore a larger solution alternatives space, increase shared understanding and induce higher quality design solutions. Therefore:

> H5: Increased communications in collaborating software development pairs in a design task will bring higher quality design solutions.

3.2.6 Cognitive Load in Software Development Design Tasks

Design is an ill-structured problem and requires integrating diverse fields of knowledge. Designers' subjective cognitive load impacts design quality. Distributed cognition enhances cognitive offloading both socially and structurally. Social cognitive offloading will occur in dyads and small groups, increasing working memory. Structural cognitive offloading, extending cognition beyond the individual through using artifacts, such as mind maps, will occur in software developers using external representations, facilitating design solution quality increases (Greenberg & Dickelman, 2000; Hansen & Lyytinen, 2009; Kaptelinin, 1996; Yvonne Rogers & Ellis, 1994; Scaife & Rogers, 1996; Sedig & Parsons, 2013; Helen Sharp et al., 2009; Yang, 2008; Zhang & Norman, 1994; Zhang, 1991).

Greenberg & Dickelman (2000) found that external artifacts offload cognitive stress in performance support systems and especially in in software analysis and design work. Rogers & Ellis (1994) posit that cognitive offloading through external representations serves to inform systems design, leading to a

more complete cognitive representation state. Sedig & Parsons (2013) examined interactive visualization-based computational tools supporting complex cognitive activities such as analytical reasoning, sense making, decision making and problem solving. Reaching solutions or decisions involves interaction between multiple actors and artifacts, leading to improved solution alternatives. Yang (2008) supported sketching systems as design representations and that increased use of external representations boosted design communication and interaction in design teams, leading to improved solution possibilities. Zhang & Norman (1994) found that distributed cognitive tasks' nature changed when artifacts were used to offload cognitive stresses, facilitating increased quality of proposed solutions. Zhang (1991) found that problem-solving representations involve both internal and external aspects in order to complete problem solvers' cognition, allowing cognitive offloading, leading to a larger space of solution alternatives and increased probabilities of proposing solutions with higher quality. Sharp et al (2009) examined both notational and social aspects of artifacts and external representations and found that software designers benefitted by offloading extant representation states to story cards and the Wall, freeing working memory and facilitating a higher, more structural perspective and promoting less rework. Scaife & Rogers (1996) researched the effects of distributed cognition and found that external representations allow computational offloading, graphical representations of constraints and boundaries, and continuous re-representation,

all aspects of framing problems for solving, leading to solutions which more closely match the engineering requirements of the original problem statement, even as the statement evolves.

Kaptelinin (1996) found that problem solvers and designers often make tools in order to make better tools, more complex and more finely tuned to problems at hand, allowing iterative, decreasing cognitive workload by software designers. Morris et al found that use of visual diagrams in software design offloaded mental stresses (Morris, Speier, & Hoffer, 1996). Further work by Speier & Morris (2003) indicated the design of a query interface, visual versus text-based, facilitated reduced cognitive loading, supporting the hypothesis that diagramming and mind maps can reduce cognitive loading. Hansen & Lyytinen (2009) found that external artifacts "bear a significant portion of the cognitive workload of the system" and allow higher thought processes in cognitive tasks. Therefore:

> H6: Decreased cognitive workload by software developers in a design task will bring higher quality design solutions.

3.2.7 Communications Brings Higher Job Satisfaction

Communication in task completion impacts the environmental aspects of the task. Increased communication during work includes technical and social feedback and impacts perceptions of task completion quality and productivity (de Vries et al., 2006; Hoegl, Praveen Parboteeah, & Gemuenden, 2003; Hulkko &

Abrahamsson, 2005; Kraut, Fish, Root, & Chalfonte, 1990; Sfetsos et al., 2006; L. A. Williams, 2000). Communication in the form of knowledge-sharing increases productivity, increasing job satisfaction (de Vries et al., 2006). Communications among team members increases project success, increasing job satisfaction. (Hoegl & Gemuenden, 2001) Communications in paired programming tasks increases job satisfaction. (Williams, 2000) Communications in paired programming tasks increases job satisfaction directly and through increased confidence. (Hulkko & Abrahamsson, 2005) Informal communications in organizations increase job satisfaction. (Kraut et al., 1990) Communications in agile software environments increases job satisfaction. (Sfetsos et al., 2006) Therefore:

> H7: Increased Communications in collaborating software development pairs in a design task will result in higher job satisfaction.

3.2.8 Decreased Cognitive Workload Brings Higher Job Satisfaction

Successful techniques to reduce cognitive workload on cognitive workers lead to higher job satisfaction (Burgess-Allen & Owen-Smith, 2010; DeChurch & Mesmer-Magnus, 2010; Hatfield & Huseman, 1982; Morris et al., 1996; Richter, Hirst, van Knippenberg, & Baer, 2012). Small group and team cognition is linked to increased job satisfaction. (DeChurch & Mesmer-Magnus, 2010) Decreased cognitive workload through perceptual congruence and improved mental model increases job satisfaction. (Hatfield & Huseman, 1982) “Knowing who knows

what" in teams decreases individuals' cognitive loads and increases creativity in task solution, leading to higher job satisfaction. (Richter et al., 2012) Mind maps can rapidly and heuristically build consensus of qualitative aspects of goal definition, decrease cognitive loading and therefore increase job satisfaction. (Burgess-Allen & Owen-Smith, 2010) Software analysis and design cognitive workload is reduced when working in teams, increasing both job satisfaction and confidence. (Morris et al., 1996) Therefore:

> H8: Decreased cognitive loading in software developers in a design task will bring higher job satisfaction.

3.2.9 Collaborating Pairs Will Produce Better Quality Than Best Individuals

Design solution quality of a collaborating pair will be higher than that of a second best member of a nominal pair in a software design task. Collectively approved solution actions, based on communication and interaction between internal cognitive processes of individual members of a collaborating pair, evolve and emerge from interaction of complementary skills and experience in a collaborating pair of software designers (Corrie, 2010; Daft & Lengel, 1986; Flor & Hutchins, 1991; Greenberg & Dickelman, 2000; Raymonde Guindon, 1990; Hansen & Lyytinen, 2009; Hanssen, 2011; K. Lui & Chan, 2003; George Mangalaraj et al., 2014; L Williams et al., 2000; Zhang & Norman, 1994). Williams et al (2000) found that among both experienced professional programmers and advanced undergraduate students, software quality, measured as

better algorithms with fewer defects, increased in pair programming situations. Lui & Chan (2003) reported that pair programming produces better quality than individuals when focused on design. Mangalaraj et al (2014) examined software design in a distributed cognition setting and found that paired development increased the quality of design solution. The combined, communicating cognitive strengths of more than one mind increase the chances of finding better solutions. (Guindon, 1990)

Design solution quality will be higher when mind maps are used during software design. Software design quality will be higher when mind maps are used during software design. Hansen & Lyytinen (2009) supported distributed structural artifacts in professional IS design leading to higher quality and less refactoring. Hanssen (2011) points out organizations which use external representations react faster and with more quality in their software ecosystem; here the external artifacts serve well as reminders of previous decisions and possible alternatives.

Increased communication in collaborating software development pairs in a design task will bring higher quality design solutions. Flor & Hutchins (1991) point out that software development pairs communicate through both social and structural cognitions in a distributed cognition system, achieving effective and efficient communication, including gesture and inarticulate, underspecified sentences in conversation. Daft & Lengel (1986) posit that face-to-face

communication supported by sketch, diagram and gesture is the richest possible media with which to communicate. Corrie (2010)found that researchers often use gesture instead of words to communicate when using diagrams and collaborative illustrations of research paths and posited solutions.

Decreased cognitive workload by software developers in a design task will bring higher quality design solutions. Designers' subjective cognitive load impacts design quality. Greenberg & Dickelman (2000) found that external artifacts offload cognitive stress in performance support systems and especially in in software analysis and design work. Zhang & Norman (1994) found that distributed cognitive tasks' nature changed when artifacts were used to offload cognitive stresses, facilitating increased quality of proposed solutions.

Due to any collaborating pair's solution quality being higher than the second best individual, the use of mind maps increasing design solution quality, increased communications among collaborating pairs bringing higher quality design solutions, and decreased cognitive workload contributing to higher quality design solutions, the collaborating paid using mind maps should produce the highest quality of design solutions. Therefore:

> H9: Design solution quality of a collaborating pair using mind maps will be higher than that of the best individual developers.

3.2.10 Hypotheses Restatements

The hypotheses stated above fit the Research Model. The Research Model indicates four possible conditions: individuals without Mind Maps, individuals with Mind Maps, collaborating pairs without Mind Maps, and collaborating pairs with Mind Maps. The hypotheses use these conditions, one way and in two way conditions, to extract predictions based on the Literature Review. Hypotheses are restated in Table 3-1 below:

Table 3-1: Hypotheses Restatements

Hypothesis	Restatement
H1	Design solution quality of a collaborating pair will be higher than that of a second best member of a nominal pair in a software design task.
H2	Design solution quality will be higher when mind maps are used during software design.
H3	Task satisfaction will be higher when mind maps are used in a software design task.
H4	Average level of task satisfaction will be higher among collaborating pairs compared to average task satisfaction level of nominal pairs.
H5	Increased communications in collaborating software development pairs in a design task will bring higher quality design solutions.
H6	Decreased cognitive loading in software developers in a design task will bring higher quality design solutions.
H7	Increased communications in collaborating software development pairs in a design task will result in higher job satisfaction.
H8	Decreased cognitive loading in software developers in a design task will bring higher job satisfaction.
H9	Design solution quality of a collaborating pair using mind maps will be higher than that of the best individual developers.

Chapter 4

Methodology and Experiment

4.1 Research Design and Variables

This study used a laboratory experiment with students as test subjects to test the various hypotheses constructed in the previous chapter. Laboratory experiments provide sufficient control to study the effects of the variables under consideration.

The study used a completely randomized design with two treatments having two levels. One treatment was individuals, which can be measured as nominal pairs, versus collaborating pairs. The other treatment was whether or not the participants used Mind Maps.

This design not only captures the four different dimensions in a simple Punnett square and provides useful information about the potential interactions possible, but also is elegant and robust. Two levels of structural distribution can be operationalized by the presence or absence of mind maps. Two levels of social distribution can be operationalized by design work as individuals in nominal pairs or design work as collaborating pairs. Individuals can also be randomly combined to form nominal pairs supporting comparison of collaborating pairs to nominal pairs, thus allowing increased analytical rigor. (Balijepally et al., 2009; Laughlin et al., 2002; Lewis, Sadosky, & Connolly, 1975; Mangalaraj et al., 2014; Shirouzu et al., 2002; Yetton & Bottger, 1982)

Participants in all four treatment sessions completed two design problems. The first was a warm-up task and the second was the main design problem. The dependent variables in this study were Design Solution Quality and Job Performance Satisfaction.

Communication and Cognitive Workload in the Development Process were used as mediating variables.

Demographics included gender, age, major, level of completed education, working hours per week, citizenry and English as primary language. Possible covariates included programming experience, object oriented programming experience, and object oriented design experience.

4.2 Experiment

This part of Chapter 4 covers the details of the subjects of the experiment, the experimental setting, sample size, research design, the task in the experiment, the pilot test, manipulation checks, and measurement of the variables.

4.2.1 Subjects

The participants in this study were advanced undergraduate and graduate students majoring in Information Systems and Computer Science Engineering. All subjects were recruited from courses which had Analysis and Design as prerequisites. Flyers encouraging students to participate were distributed in

appropriate classes. A copy of the flyer is included in Appendix A. Subjects were invited to register for the experiment online. Incentives to participate included Walmart gift cards and textbooks given away in random drawings during each experimental session. Once registered for a particular session date and time on the website, emails were sent to registrants reminding them of their commitment and exhorting their participation.

Because the experiment used human subjects in the study, approval was sought and obtained from the university's Institutional Review Board.

4.2.2 Experimental Setting

Experiment sessions were conducted in a reserved university computer laboratory. The lab contained 72 computers and participation was limited to 36 subjects per session after the first few sessions were conducted with a limit of 24 subjects. The spacing between participants enabled them to work privately in those sessions focused on individual efforts. The environment was quiet throughout the sessions and was closed to outsiders. The computers had Internet access and this access was used to complete the experiment by using one or two, depending on the focus of the session, online tools – one for mind mapping and the other for UML diagrams.

All subjects signed an Informed Consent form required by the university's Institutional Review Board as they arrived for the session. A copy of the Informed

Consent form is included in Appendix B. All subjects were trained on UML class diagrams using a PowerPoint presentation and each subject was given a copy of the PowerPoint presentation at the beginning of the session to follow along and take notes.

4.2.3 Planned Sample Size

Planned sample size for the experiment was 160 subjects. This would yield 40 participants in each condition: individuals without mind maps, individuals with mind maps, pairs without mind maps and pairs with mind maps. Because of the intention to analyze the individuals as nominal pairs in some statistical analyses, this would yield 20 pairs, nominal or collaborating, in each of the four conditions. Sessions were conducted in the Summer 2015 semester and also in the Fall 2015 semester. A total of 18 sessions were conducted. The type of each session was chosen upon learning the number of subjects registered for that date and time combination in order to attempt to balance the total participation among the four possible conditions. Each session was dedicated to a particular condition among the four possibilities and all subjects in any one session participated in the same condition.

4.2.4 Research Design

Experiment subjects signed up online, choosing from options scheduled for a particular date and time to attend. The setting for each session was determined close to the start time of each session, because this allowed a balancing between session conditions.

Session conditions included individuals working alone or individuals forming collaborating pairs, and also included using mind maps or not using mind maps prior to completing the main UML diagram task. Sessions in condition "D" of pairs using mind maps included the individual subjects (before forming pairs) working on the main problem task in mind maps individually before producing a joint mind map of the main problem task. As a pair in condition "D," the subjects then produced a UML diagram for the main problem task. In this way, all four conditions were presented in various lab sessions, shown here in Table 4-1 below:

Table 4-1: Conditions in Experimental Sessions

Conditions	No Mind Maps	Mind Maps
Individuals	Session "A"	Session "B"
Pairs	Session "C"	Session "D"

4.2.5 Experimental Task

All subjects in all session conditions performed two tasks during their participation in the experiment. Both tasks used UML class diagram modeling and both used the online tool for the tasks. UML diagramming is a software development industry standard and is commonly used in software development research (Burton-Jones & Meso, 2008). The first task was a warm-up task which allotted 12 minutes to reproduce and complete a UML class diagram showing the classes, attributes and methods associated with a standard bank ATM machine. The second task was based on a problem statement taken from McLaughlin, Pollice and West (2006). The problem statement used for the main task is included in Appendix D. Subjects using mind maps were introduced to and trained in using mind maps using the materials in the PowerPoint presentations used in the experiment sessions.

In those sessions not using mind maps, both for individuals and for pairs, an online tool was used to create the UML diagrams for the warm-up task and for the main task. In those sessions using mind maps, a different online tool was used to create mind maps prior to working on the UML main task diagram.

Each session, regardless of condition, lasted 2 hours and the warm-up task and main task were allotted 12 minutes and 55 minutes, respectively. The PowerPoint presentations, and the trainings contained in them, varied by the condition of each session.

4.2.6 Pilot Test

An experiment walk-through and pilot test were conducted to improve the experiment procedures. The walk-through tested the manipulation checks and measurements by having subjects complete the questionnaires. This process enabled us to get feedback from the participants in the walk-through about the answers, thus enabling us to validate of the questionnaires.

A pilot test was conducted using PhD students from the Information Systems Department. A total of 6 students participated. Based on the feedback from the pilot test, and the observations of the researcher, changes were made to the materials and the timings used in the main experiment. Proposed changes made were tested by second, smaller pilot test using 2 PhD students and were agreed to be improvements from the original.

4.2.7 Manipulation Checks

Two primary manipulations were used in this experiment. One was the use of mind maps in both the individual condition (Session B) and the pair condition (Session D) versus not using mind maps (Sessions A and C). The other manipulation was the individual condition (Sessions A and B) versus the collaborating pair condition (Sessions C and D).

Subjects in the experiment in Sessions B and D were asked about the impact of mind maps in their thought processes and cognitive workload while

working out solutions to the main task, while subjects in Sessions A and C were asked about the impact of their own notes and sketches in their thought processes and cognitive workload while working out solutions to the main task.

Subjects in the experiment who participated in Sessions C and D were asked about the impact of communications with their partner while working out a solution approved by both members of the dyad, while subjects in Sessions A and B were asked about their communications with themselves through notes and sketches or through mind maps. Subjects in Sessions C and D were also asked about their partners' share of contribution to the final UML class diagram solution the pair submitted for grading.

The control facilitated by using these manipulation checks in the experiment increases confidence in the findings.

4.2.8 Measurement of Variables

All variables were measured using scales in extant literature. Table 4-2 summarizes these scales. The dependent variables, Design Solutions Quality and Job Performance Satisfaction, were both continuous variables. The mediating variables, Communications in the Development Process and Cognitive Workload in the Development Process, were also continuous variables. The independent variables were the session conditions of individuals versus pairs and using mind maps versus not using mind maps, represented in the four types of experimental

sessions. The settings for the experiment consisted of using the same room, with the same presenter in the room, the same software package to produce the UML diagram and the same software package to produce the mind maps used in sessions “B” and “D” conditions. Therefore, the controls in the experiment were consistent across the session conditions.

4.2.8.1 Design Solutions Quality

The UML class diagram solutions designed by the experiment subjects were graded according to a rubric developed from the possible solutions given in the same book from which the problem statement was obtained. The five evolving solutions in the book were combined into the final best solution in the book. This solution was adapted to a rubric which gave points to the solution submitted for grading based on the five levels of completeness and correctness outlined in the book. Fifty-point breaks between the levels were used to clearly illustrate differences in design quality

Graders could also award additional points for using proper UML connections, such as the open arrowheads indicating extended subclasses, and for properly indicating multiplicities and their verbs, e.g., “extends” and so on.

The UML class diagram solutions were graded by one PhD student and one Masters student who each had deep experience in Analysis and Design, as well as industry experience and certifications. The two graders were blind to the

hypotheses used in the research experiment. Each of these students helped refine the rubric by practicing on three randomly chosen submitted solutions while conferring with the researcher during training on the rubric.

The final rubric contained five levels of performance ranging from 50 points to 250 points, excluding additional points of 5 and 10 for niceties contained in the diagrams. The graders' scores were the measure of this construct.

4.2.8.2 Job Performance Satisfaction

Satisfaction with job performance is linked to better performance by software designers (Hulkko & Abrahamsson, 2005) and is customary of best practices in the software industry. Job performance satisfaction was measured using the NASA Task Load Index (NASA TLX) as well as questions regarding job performance satisfaction from Mangalaraj (2006). The questionnaires from the individual-focused Sessions A and B included questions about individuals' perceptions of their UML solution quality as a measure of performance. The Sessions C and D, focused on pairs developing UML solutions, asked questions of each individual's perceptions of the pair's performance. A total of twelve items measured this construct.

4.2.8.3 Communications in the Development Process

Communications among individuals and dyads and small teams solving problems have many measures available. This experiment used a scale developed by Visshers-Pleijers et al (2005) for use in Problem-Based Learning settings. Considering that this experiment trained its participants in UML class diagrams and then tasked the subjects with solving a common problem, the communications involved in the original research article track very closely with this research's activities. Eleven items were used to measure this construct.

Additional questions were adapted to the questionnaire asking individuals about communications with structural distributed cognition artifacts such as notes and sketches, including individuals' mind maps, in designing the final UML class diagram solutions for grading. These questions were used for individuals in place of the communications questions from Visscher-Pleijers et al (2005) designed for dyads and small teams, which focus on socially distributed cognition settings. This was done to enhance future research into possible differences between structural and social nuances of distributed cognition.

4.2.8.4 Cognitive Workload in the Development Process

Mental stress and loading while developing solutions to the main task impact both the quality of the design solution and the job performance perceptions (Mangalaraj et al., 2014; Speier & Morris, 2003). Systems analysis and design

tasks are inherently ill-structured, thus increasing the probability of cognitive workload. The adapted NASA TLX measures mental workload directly with questions that compare the mental workload to frustration level, time pressure, the level of effort required, and perceptions of job performance, while deleting concerns about the physical demands of the task in this research. Six items were used to measure this construct.

A summary table of constructs in shown in Table 4-2 here below.

Table 4-2: Scales of Constructs Measured

Construct	Number of Items	Source
Design Solution Quality	1	McLaughlin et al (2005)
Job Performance Satisfaction	12	NASA TLX and Mangalaraj (2006)
Communications	11	Visshers-Pleijers et al (2006)
Cognitive Workload	6	NASA TLX, adapted by Morris et al (1999)

4.2.9 Statistical Analyses

Multiple levels of statistical procedures were planned for use in testing various hypotheses presented in this research study. Prior to testing the actual hypotheses, preliminary analysis of the data garnered included factor analysis and reliability analysis, as well as suitable assumption checks in addition to the manipulation checks carried out.

Hypotheses H1, H4 and H9 needed to use nominal pairs. The dependent variable in these particular hypotheses was Design Solutions Quality, measured at the group level for the collaborating pairs. Nominal pairs were to be created by randomly assigning individuals into nominal pairs. In total, 40 nominal pairs were planned. The research design, by using nominal pairs, allows the researcher to identify the 1st best and 2nd best in each nominal pair. Because of this, two-way Analysis of Variance (ANOVA) with one factor being each subject's participation type with three levels (collaborating pairs, 1st best individual in the nominal pair and 2nd best individual in the nominal pair), and the other factor using mind maps or not was used to analyze the Design Solution Quality of the participants. Each subject's experience in object oriented programming was used as a covariate for this analysis.

Hypotheses H2 and H3 use the dependent measures of Design Solutions Quality and Job Performance Satisfaction of subjects using mind maps. Therefore, a one-way ANOVA was used to discern support for these measures, with the one factor being the use of mind maps or not.

Hypotheses H5, H6, H7 and H8 reference the potential of Communications and Cognitive Workload in the Development Process as mediators. In order to test for mediation, this experiment followed general procedures for mediation are outlined by Baron and Kenny (1986). Three separate

regression analyses are needed to establish mediation. The three regression equations are:

a) regression of the mediator on the independent variable,

b) regression of the dependent variable on the independent variable, and

c) regression of the dependent variable on both the independent variable and the mediator.

This experiment used two categorical independent variables, individuals versus pairs and using mind maps or not using mind maps. Regression analysis based on using mind maps or not was performed to test the potential mediators. This was followed by using regression analysis based on individuals or pairs to test the potential mediators. An example of the three steps performed for each of the factors on each of the mediators follows:

(1) Straightforward regression analysis with use of mind maps as an independent variable and Communications as the dependent variable.

(2) Simple regression analysis with use of mind maps as the independent variable and Design Solution Quality as the dependent variable.

(3) Multiple regression analysis with use of mind maps and Communications as the independent variables and Design Solution Quality as the dependent variable.

These three steps were repeated using Cognitive Workload as the potential mediator:

(1) Simple regression analysis with use of mind maps as an independent variable and Cognitive Workload as the dependent variable.

(2) Straightforward regression analysis with use of mind maps as the independent variable and Design Solution Quality as the dependent variable.

(3) Multiple regression analysis with use of mind maps and Cognitive Workload as the independent variables and Design Solution Quality as the dependent variable.

These three steps were repeated using individual subjects or pairs, and Communications and Cognitive Workload as mediators, using the three steps to measure mediation of Design Solution Quality and Job Performance Satisfaction.

A total of eight three-step regressions were run to test for mediation effects. The results of included in Appendix F.

The following chapter presents the results of these hypotheses tests that were carried out.

Chapter 5

Analyses and Hypotheses Testing

This chapter presents the results of the analyses of the data, hypotheses testing and manipulation checks in the experiment conducted.

5.1 Analyses

Preliminary analyses of the data were accomplished prior to actual hypotheses testing. Analyses of the sample characteristics, as well as tests of validity and reliability of the dependent measures were done. Tests for various assumptions of the statistical techniques used are included.

5.1.1. Sample Characteristics

In total, 144 subjects participated in the experimental sessions conducted over the 18 sessions. Data from 6 subjects were dropped. One subject left early and did not finish the experiment; unfortunately this broke a pair in the pair-focused session and cost 2 data points. One pair did not submit their work for grading, although they did finish the experiment and the questionnaires. One individual did not complete the questionnaire, turning in no answers past the first page demographics. One individual turned in a questionnaire after choosing the maximum marks on each question, rendering the questionnaire unusable. Total usable sample size became 138. Figure 5-1 summarizes sample sizes in various

sessions of the experiment. Table 5-1 follows, and presents the demographics characteristics of the experiment participants.

Conditions	No Mind Maps	Mind Maps
Individuals	Session A n = 34 17 nominal pairs	Session B n = 34 17 nominal pairs
Pairs	Session C n = 32 16 collaborating pairs	Session D n = 38 19 collaborating pairs

Total sample size was 138

Figure 5-1: Session Sample Sizes

Table 5-1: Experiment Sample Characteristics

	Description	Subjects	Subject %%
Gender	Female	31	22.5%
	Male	107	77.5%
Age	<23	29	21.2%
	23-24	36	26.2%
	25-26	39	28.5%
	27-60	33	24.1%
	Did not provide	1	0.7%
Education	High school	95	68.8%
	Community college	1	0.7%
	Undergraduate degree	39	28.3%
	Graduate degree	1	0.7%
	Doctoral degree	1	0.7%
	Did not provide	1	0.7%
Work Hrs / Wk	0	69	50.0%
	01-19	12	8.7%
	20	29	21.0%
	21-40	24	17.4%
	>40	4	2.9%
Citizenship	India	55	39.9%
	USA	54	39.1%
	China	2	1.4%
	Other	27	19.6%
English as Primary Language	No	90	65.2%
	Yes	47	34.1%
	Did not provide	1	0.7%
Programming Experience Yrs	1	56	40.6%
	2	36	26.1%
	3	31	22.5%
	4	7	5.1%
	5	8	5.8%
Object-Oriented Experience	1	65	47.1%
	2	45	32.6%
	3	22	15.9%
	4	6	4.3%
Object-Oriented Design Experience	1	34	25.2%
	2	49	35.5%
	3	52	37.7%
	Did not provide	3	2.2%

Regression tests using analysis of variance were done to discover whether demographic characteristics of subjects differed across the four session types. Results of the tests are as follows: Age ($F = 0.906$, $p = 0.369$), Gender ($F = -1.254$, $p = 0.215$), Programming experience ($F = -0.193$, $p = 0.848$), Object oriented experience ($F = 1.026$, $p = 0.310$), and Object oriented design experience ($F = 0.448$, $p = 0.729$). Results of the analyses of Education level, GPA, Major (Information Systems or Computer Science Engineering), Hours worked per week, English as Primary language, programming experience, object oriented programming experience and Object oriented design experience showed no significant differences across the four session types.

Pairings in the experiment were randomized and pairs were formed during the training part of the sessions in which pairs were used. Therefore, it is possible that pairs in the experiment might already know each other prior to the experiment. This study did not exclude such participation but subjects were asked if they had previously worked with the partner they used during the experiment. Six persons reported they had previously worked with their partner, representing 8.5% of the collaborating pairs in Sessions C and D of this study.

Of the reported demographic variables reported and analyzed in this study, the researcher used object oriented programming experience as a covariate for subsequent analyses. Object oriented programming typically involves classes, interfaces, inheritance, polymorphism and other concepts that are very important

in evolving UML class diagrams. Therefore it is not surprising that the variable was found to be significant as a covariate ($F = 6.327$, $p = 0.021$).

5.1.2 Characteristics of Dependent Variables and Mediators

The variables measured in this research experiment include Job Performance Satisfaction, Design Solutions Quality, and Communications and Cognitive Workload in the Development Process. This section describes the analyses performed on these data.

Job Performance Satisfaction was measured using the questions from Mangalaraj (2006) as well as the direct measure of perceived Performance from the adapted NASA TLX (Speier & Morris, 2003). These questions use a Likert scale from 1 to 10. Those subjects in Session D were asked about Job Performance Satisfaction in both individual and pair settings due to their work in mind maps, where they constructed mind maps individually and then in a pair before constructing the UML class diagram for problem solution as the final product of their work. This left 6 questions for all sessions in common. The results of factor analysis under Varimax rotation loading for Job Performance Satisfaction for all experimental sessions are shown in Table 5-2:

Table 5-2: Satisfaction Loading Matrix for All Sessions

	F1	F2
Satis1a	0.925	0.101
Satis1b	0.903	0.207
Satis1c	0.843	0.045
Satis1d	0.908	0.139
Satis2a	-0.248	0.914
Satis2b	-0.238	0.913
Eigen Values	3.324	1.743
Cumulative Variance	55.394	84.451

Design Solution Quality was measured by grading each UML class diagram solution submitted against a rubric. The rubric was taken from the same book in which the problem was given; the authors of the book derived the best solution to the problem they were using to train the readers. As mentioned in Chapter 4, the grading evaluation of design solution quality was done by two graduate students, who independently graded the 103 UML class diagrams according to the rubric. Inter-rater reliability was 0.942, so the average of the grades was used for scoring the measure of Design Solution Quality.

Cognitive Workload in the Development Process was measured by an adapted NASA TLX scale. The NASA Task Load Index (TLX) is originally

conceptualized with six dimensions: mental demand, time constraints, performance required, effort required, level of frustration and physical demands. 10 of the original 15 items pertained to the experiment, and physical demands did not, thus reducing the total measures to just the 10 that were relevant to software development. The TLX scores come from asking the subjects to select a dimension among the pairs of dimensions (Mental vs Time, Effort versus Performance, and so on) which contributed to their overall cognitive workload. Score for Cognitive Workload in this study was derived based on the counts of the number of times each dimension influenced the cognitive workload and then multiplied by the salience of each dimension, indexed to a scale of 1 to 100. The adapted NASA TLX is taken from Morris (Morris et al., 1996) and Speier and Morris (2003).

Communications in the Development Process was measured using an instrument developed and tested in Problem-Based Learning. The instrument uses 11 questions, along three factors, to gauge the communications process in a dyad and small team setting (Visschers-Pleijers, Dolmans, Wolfhagen, & van der Vleuten, 2005). This instrument asks questions about exploring solutions in the group, questions about the cumulative reasoning in the group and about conflicts in the group. The questions used a Likert scale on 1 to 5. The three factors, under Varimax condition, loaded as shown here in the component matrix Table 5-3:

Table 5-3: Communications Loading Matrix

	F1	F2	F3
Comm01	0.797	-0.348	-0.090
Comm02	0.858	-0.310	0.010
Comm03	0.785	-0.263	-0.104
Comm04	0.712	0.004	0.114
Comm05	-0.040	0.748	0.012
Comm06	0.170	0.671	0.179
Comm07	0.016	0.712	0.542
Comm08	-0.250	0.748	-0.109
Comm09	0.465	-0.096	0.776
Comm10	0.424	0.122	0.801
Comm11	-0.630	0.236	0.600
Eigen Values	4.422	2.238	1.136
Cumulative Variance	40.202	60.551	70.881

Because the third factor in Communications has one question that did not load at or above 0.7, the average of the loadings for the third factor, which deals with conflict in communications, was deducted and found to be 0.725. Because of the low overall loading, the third Communications factor was used, but with caution, for the remaining analyses. The textual answers supplied by subjects indicated a very low rate of conflict in communications overall.

The reliability of the perceptual measures was addressed to increase faith in the study's statistical findings. Reliability was calculated in terms of Cronbach's alpha and the results are: Job Performance Satisfaction = 0.87, Communications in the Development Process = 0.78, and Cognitive Workload in the Development Process = 0.93. The reliability of Job Performance Satisfaction and Cognitive Workload in the Development Process are quite satisfactory, while the reliability of the Communications measure is of some concern. This experiment attempted to introduce a measure of communications from the Problem-Based Learning literature into a problem-solving design context. Because of this attempt, the reliability of the Communications measure used here is acceptable with extant literature regarding reliability of measures (Nunnally, 1978), where ratings above 0.7 are considered acceptable, but is of some concern in the findings.

Scores for Design Solution Quality were based on individuals not using mind maps (Session A), individuals using mind maps (Session B) and collaborating pairs (Sessions C and D). Bifurcation of individuals in Sessions A and B into 1st best scores and 2nd best scores in random nominal pairings yield nominal pairs for direct comparison with collaborating pairs regarding Design Solutions Quality. Individual scores for Job Performance Satisfaction, Communications in the Development Process and Cognitive Workload in the Development Process were gathered in all experiment cycles. Correlation between

Design Solution Quality and Job Performance Satisfaction as dependent variables was -0.119. The means and standard deviations for Design Solution Quality (scale 0 to 250), Job Performance Satisfaction (indexed scale 0 to 100), Communications in the Development Process (indexed scale 0 to 100) and Cognitive Workload in the Development Process (indexed scale 0 to 100) are contained in Table 5-4 shown below:

Table 5-4: Means and Standard Deviations

	Mean	StDev
Design Solution Quality	178.80	59.08
Job Performance Satisfaction	73.41	33.06
Communications	72.46	18.14
Cognitive Workload	66.57	19.15

Power analyses were conducted for the experiment's research results. Effect sizes for analysis of variance methods are classified by Cohen into small effect size = 0.10, medium effect size = 0.25 and large effect size = 0.40 (Cohen, 1992). In this study, eta-squared effect sizes were expected to be small to medium, based on the pilot test and previous studies in pair programming (Balijepally et al., 2009; Mangalaraj et al., 2014). The reports from analyses are based on two-tail tests in the statistical software packages. One-tail tests have more power than two-tail tests for any given sample size and effect size, therefore

the power observations from the statistical packages included in Table 5-5 understate the power levels and are considered conservative. Table 5-5 is shown here below:

Table 5-5: Power Analysis

Dependent Variable	Parameter	Effect Size (Non-centrality Parameter)	Power at alpha = 0.05
Design Solution Quality	IW 2nd best Indiv	1.986	0.352
	IW 1st best Indiv	2.853	0.832
	IM 2nd best Indiv	1.979	0.348
	IM 1st best Indiv	2.327	0.714
	PW Collab Prs	1.863	0.454
	PM Collab Prs	1.915	0.339
Job Performance Satisfaction	IW 2nd best Indiv	1.683	0.313
	IW 1st best Indiv	2.551	0.703
	IM 2nd best Indiv	2.108	0.671
	IM 1st best Indiv	2.518	0.694
	PW Collab Prs	2.218	0.621
	PM Collab Prs	3.191	0.826
Communications in the Development Process	IW 2nd best Indiv	1.120	0.212
	IW 1st best Indiv	0.971	0.195
	IM 2nd best Indiv	1.003	0.201
	IM 1st best Indiv	0.814	0.176
	PW Collab Prs	1.604	0.308
	PM Collab Prs	1.839	0.427
Cognitive Workload in the Development Process	IW 2nd best Indiv	1.752	0.340
	IW 1st best Indiv	1.083	0.222
	IM 2nd best Indiv	2.034	0.649
	IM 1st best Indiv	2.417	0.667
	PW Collab Prs	2.101	0.663
	PM Collab Prs	1.693	0.318

LEGEND:
IW 2nd best Indiv = Individual, w/o Mind Maps, 2nd best
IW 1st best Indiv = Individual, w/o Mind Maps, 1st best
IM 2nd best Indiv = Individual, w Mind Maps, 2nd best
IM 1st best Indiv = Individual, w Mind Maps, 1st best
PW Collab Prs = Collaborating Pairs without Mind Maps
PM Collab Prs = Collaborating Pairs with Mind Maps

5.1.3 Assumptions Tests

This study used various analyses of variance (ANOVA/ANCOVA) to analyze the various hypotheses. The underlying assumptions of ANOVA models include: (1) homogeneity or constancy of error variance, (2) independence of error terms, and (3) normally distributed error terms and, like all parametric tests, (4) normal distribution of results. ANCOVA models rely on two additional assumptions: (1) equality of slopes of the different treatment regression lines and (2) linearity of regression lines. Histograms indicated no significant outliers in the data, indicating no violation of normal distribution requirements.

Normality of distribution and homogeneity of error variance was test using the Omnibus Test for Normality distribution and the Modified Levene test homogeneity of error variance for ANOVA procedures. The results of the test indicated that the assumptions for normal distribution and for constancy of error variance cannot be rejected. The results are shown in Table 5-6 below:

Table 5-6: Normality and Homoscedasticity of Error

	Omnibus Test for Normality		Modified Levene Test for Constancy of Error Variance	
Variable	Statistic	Significance	Statistic	Significance
Design Solution Quality	1.763	0.384	0.532	0.712
Job Performance Satisfaction	2.479	0.285	0.277	0.615
Communications	1.783	0.446	0.094	0.681
Cognitive Workload	2.578	0.271	0.134	0.825

The ANOVA assumption of independence of error terms is addressed in the research design itself. Hair et al (1998) suggest that randomization in a research study is the most important safeguard against correlation in the error terms. A simple plot of error terms visually indicated no discernable patterns in the error terms. This plot, along with the full randomization of the experimental conditions in the various Sessions A, B, C and D that were held, are good support for arguing independence in the error terms.

Requirements for assumptions of ANCOVA include all the above requirements for ANOVA, and additionally, require equal slopes between

experiment treatments regressions and linearity of the slopes of treatment regressions.

Regarding the assumption of equality of slopes between treatments in the experiment, an F-test was used to test for equality of slopes (Kutner et al, 2005) using object oriented programming experience as a covariate. Multiple regression analysis with indicator variables for the treatment conditions was used to create a full regression model and a reduced regressions model. The full model used all the treatment conditions and their interaction with the covariate. The reduced model had the treatment variables and the covariate without the interaction. At alpha = 0.05, the equality of slopes between treatments could not be rejected.

Finally, regarding the assumption of linearity of slopes among treatments, a general linear model, including the covariate, was utilized to test the linearity assumption. Kutner et al (2005) specify the linearity of a regression function can be judged by examining a plot of residuals against the fitted values. The plot of residuals against the fitted values indicated no apparent departure from linear slopes. Therefore, all the assumptions of ANOVA/ANCOVA techniques of analysis were met.

The dependent variables and the mediator variables are shown reliable, have good power, and have met all the assumptions required to establish confidence in the findings of this study.

5.1.4 Tests for Interaction

Referring to Kutner et al (2005), analysis of factor effects in a two factor analysis of variance study should occur after checking for interactions. The presence of significant interactions between the two factors would then require using a cell means model. Tests for significant interaction were carried out using ANCOVA procedures for Design Solutions Quality scores, and MANCOVA for other dependent variables between subjects in nominal pair or collaborating pair conditions. As before, object oriented programming was used as a covariate in these analyses. Each dependent measure was tested separately in a 3x2 ANCOVA design, the first factor being the three possible 1st best individual, 2nd best individual and collaborating pair and the second factor being the use of mind maps or not. There were no significant interaction effects at alpha = 0.05 for all the four dependent variables. Because no significant interactions were found, factor level means were used for the remaining analyses. Table 5-7 shows the results of the interaction tests below:

Table 5-7: Test Results for Interactions

Dependent Variables	Individuals vs Pairs F-ratio (p-value)	Mind Maps vs No Mind Maps F-ratio (p-value)	Interaction F-ratio // p-value
Design Solution Quality	8.124 (0.001)	10.398 (0.001)	0.364 (0.631)
Job Performance Satisfaction	6.845 (0.008)	5.769 (0.027)	0.129 (0.903)
Communications	4.223 (0.036)	7.3315 (0.003)	0.664 (0.391)
Cognitive Workload	12. 353 (0.001)	5.467 (0.025)	0.449 (0.518)

MANCOVA techniques are used for multiple dependent variables with a concomitant continuous covariate. The MANCOVA techniques may be seen as MANOVA of the regression residuals, measuring variance in the dependent variable while controlling for the covariate (Hair et al, 2005). The Omnibus Normality test across the factors on the dependent measures supports the assumption of normality of error terms for this MANCOVA and could not be rejected at alpha = 0.05.

The Modified Levene test was conducted to test the equality of error term variances of the dependent variables across the two factors individually. The assumption of equal variance in the error terms was supported and could not be rejected at alpha = 0.05 for this evolution.

The assumption of equality of covariance matrices was tested using Box's M test. Box's M test is highly sensitive whenever used, and also not robust when using unequal sample sizes. The differences in sample sizes in this study were kept to maximize use of data collected. If factor means differences are found in "small" cells with "large" covariances, and further, if non-significance is found by the test, then the null hypothesis of assumption of homogeneity of covariance matrices can safely be sustained. However, significant results leading to rejection of the null hypothesis are suspect if the main criterion is simple Box's M test under unequal sample sizes with small cells and large covariances. Because this study did separate the individuals into 1st best and 2nd best based on their Design Solution Quality scores, the sample sizes can be considered small. Therefore, results of the Box's M test should be viewed with caution. The Box's M statistical findings were significant with a p-value of 0.108. This supports failure to reject the assumption of equality of covariance matrices, even under the cautions outlined above, at a significance level of alpha = 0.05.

Results of the diagnostics for the underlying assumptions of the MANCOVA techniques are illustrated in Table 5-8 shown below:

Table 5-8: Diagnostic Information for MANCOVA Assumptions

	Omnibus Normality test		Modified Levene test		Box's M test	
	Statistic	Sig	Statistic	Sig	Statistic	Sig
Dependent Variables versus Individual or Pair condition						
Design Solution Quality	0.331	0.848	0.024	0.876	7.517	0.127
Job Performance Satisfaction	3.878	0.245	0.131	0.718		
Communications	1.284	0.459	1.823	0. 361		
Cognitive Workload	1.590	0.412	1.992	0.347		
Dependent Variables versus no Mind Maps or Mind Maps						
Design Solution Quality	3.747	0.154	0.414	0.743	16.712	0.864
Job Performance Satisfaction	4.071	0.131	0.200	0.896		
Communications	1.837	0.339	1.293	0.415		
Cognitive Workload	3.101	0.219	2.592	0.262		

5.1.5. Tests of Significance

This study used object oriented programming experience as a covariate and this was collected at the individual level in the demographics. For ANCOVA analyses, the average object oriented programming experience level of the pair,

nominal or collaborating, was as the covariate measure where applicable. Again, the 1st best and 2nd best individuals in nominal pairs were determined by the Design Solution Quality score. Results of the 3x2 ANCOVA analysis on Design Solution Quality scores, where the three levels of 1st best individuals, 2nd best individuals and collaborating pairs were matched against the two levels of using mind maps or not using mind maps, indicated a significant main effect for the three groupings (1st best, 2nd best and collaborating pairs) with a p-value of 0.000. The analysis also showed a significant main effect for the mind map dimension with a p-value of 0.006. The interaction between the two main factors was found to be significant with a p-value of 0.008. Table 5-9 summarizes the one-way and two-way ANOVA results for Design Solution Quality.

5-9: ANOVA Results on Design Solution Quality

	Individual or Pair							
	1st best Indiv		2nd best Indiv		Collab Pair		F-ratio	p-value
	Mean	SD	Mean	SD	Mean	SD		
Design Solution Quality	215.15	26.61	123.82	38.61	183.00	57.11	16.272	0.000
	No Mind Maps or Mind Maps							
	No Mind Maps		Mind Maps				F-ratio	p-value
	Mean	SD	Mean	SD				
Design Solution Quality	170.76	60.11	186.18	57.50			8.028	0.006
Individual or Pair versus No Mind Maps or Mind Maps								
Design Solution Quality	1st best Indiv		2nd best Indiv		Collab Pair		F-ratio	p-value
	Mean	SD	Mean	SD	Mean	SD		
No Mind Maps	214.41	23.78	118.53	41.30	170.00	57.70	7.593	0.008
Mind Maps	215.88	29.91	129.12	36.20	193.95	55.76		

Against an alpha = 0.05 specification, inflated type I error might be missed in a multivariate analysis. The Bonferroni alpha adjustment was examined. Having 4 variables at 0.05 each, the adjusted Bonferroni alpha would be 0.0125 (Kutner et al., 2005). The resulting p-values from this study are all below 0.0125 and therefore the experiment-wide error-rate is acceptable.

By using group comparisons among four dependent variables, there are two ways these comparisons can be made with multiple outcome variables. One way is to conduct multiple analyses of variances (ANOVAs) and the other way is to conduct a Multivariate Analysis of Variance (MANOVA) followed by multiple ANOVAs. In this study the dependent variables of Design Solution Quality and Job Performance Satisfaction are conceptually different from each other. Further, MANOVA requires a minimum cell size of 20. Because the cell sizes of the 1st best and 2nd best individuals were all 17, this study did not meet the criteria for MANOVA.

5.1.6 Manipulation Checks

This research study had two treatments with two levels each. The two treatments were individuals versus pairs as the two levels in the 1st treatment and no mind maps versus using mind maps as the two levels in the 2nd treatment.

The 1st condition was the mode of participation. In this treatment this study had sessions with individuals and sessions in pairs. Those subjects in collaborating pairs were asked individually about the active participation of their partner in the questionnaire for that session type. Descriptive statistics results indicate a mean level of 5.985 and a standard deviation of 0.891 on a Likert scale on 1 to 7 for the active level of participation in the partnerships during the experiment. Of a total of 70 subjects in the collaborating pair condition sessions,

only 2 indicated less than a neutral (4.0) rating for their partner's participations, leaving a 1 out of 35 pairs (2.86%) problem rate. This yields a 97.14% success rate for the manipulation check. Based on these findings, there is support for the Design Solution Quality scores being achieved on a collaborative partnership in the pair condition during the experimental sessions Session C (collaborating pairs not using mind maps) and Session D (collaborating pairs using mind maps) experimental cycles.

The 2nd condition was the use of mind maps, in both individuals and in collaborating pairs. In this condition the experiment had cycles of subject who used mind maps and cycles of subjects who did not use mind maps. Those subjects, individuals and collaborating pairs, who did not use mind maps were asked questions about their use of notes and sketches and the impact of the notes and sketches during the design of the UML class diagrams. Those subjects, individuals and collaborating pairs, who used mind maps were asked questions about their use of their constructed mind maps and its' impact on the design of the UML class diagram. Those who used mind maps during design praised the mind maps as tools to help them envision and innovate a better design (mean = 6.48, SD = 0.631) more than those who used notes and sketches (mean = 5.53, SD = 1.762). This provides support for mind maps having impact on the design of UML class diagrams as a manipulation check.

Results of the manipulation checks indicate both of the manipulation checks have worked as intended.

5.2 Hypothesis Testing

This study used ANOVA/ANCOVA techniques and simple and multiple regressions to test the various hypotheses outlined earlier in the research study and the data collected in the experimental sessions. The following analyses were done: (1) Two-way ANCOVA for dependent measures of Design Solution Quality and Job Performance Satisfaction, (2) regression analyses for Communications in the Development Process and Cognitive Workload in the Development Process, (3) simple and multiple regression analyses for Mediator analysis, and finally (4) regression analyses to illustrate the overall effectiveness of the experimental results.

5.2.1 Individuals versus Pairs for Design Solution Quality

This research study compared individuals in nominal pairs to collaborating pairs of subjects. Nominal pairs were formed by randomly combining two individuals in a given treatment session. Based on these nominal pairings, the 1st best and 2nd best individuals were discerned in each pair by comparing Design Solution Quality scores. Hypotheses concerning individuals in nominal pairs and collaborating pairs were analyzed using a 3 (1st best individual, 2nd best individual

in nominal pairs and collaborating pairs) by 2 (not using mind maps and using mind maps) ANCOVA design. Object oriented programming experience was used as the covariate. In the collaborating pair, the average experience level of the pair was used as the covariate. Table 5-10 presents the results of this analysis here:

Table 5-10: ANCOVA Model Results for Design Solution Quality

Source	Sum of Squares	df	Mean Square	F-ratio	p-value
Object Oriented Programming Experience	8231.12	1	8231.12	7.819	0.002
IndivPrs	12399.41	2	6199.71	7.315	0.000
Mind Maps	8732.64	1	8732.64	8.874	0.006
IndivPrs * Mind Maps	10128.39	2	64.19	0.186	0.837
Error	3669.14	97	164.00		
Corrected Total	40719.10	103			

ANCOVA results indicated the presence of significant main effects of both the mode of participation (individuals versus pairs) and the presence or absence of mind maps, along with no interaction effects. Because of these findings further analysis of factor levels was pursued.

Hypotheses 1 and 9 use nominal pairings which result in comparisons of Design Solution Quality between 1st best and 2nd best individuals in nominal pairs compared to collaborating pairs. Bonferroni's custom comparison procedure performs well when the number of contrasts of interest is small and has been

specified in advance, and when the number of contrasts of interest is about the same as the number of factor levels, or fewer (Kutner et al, 2005). These hypotheses are repeated here.

H1: Design Solution Quality of a collaborating pair will be higher than that of a 2nd best member of a nominal pair in a software design task.

H9: Design Solution Quality of a collaborating pair using mind maps will outperform the 1st best individual developers.

Table 5-11 presents the results of the comparisons tested using Bonferroni's custom comparisons.

Table 5-11: Bonferroni's Custom Comparisons for Design Solution Quality

	Mean	SD	t-value	p-value
Hypothesis 1				
2nd best Indiv	123.82	38.61	2.847	0.005
Collab Pair	183.00	57.11		
Hypothesis 9				
1st best Indiv	215.15	26.61	1.773	0.083
Collab Pair	193.95	55.76		

H1 was supported. Performance of collaborating pairs was better than the 2nd best individual in nominal pairs. H9 was not supported, indicating failure to reject the hypothesis that collaborating pairs would not outperform the 1st best individual in a nominal pair. Further statistical analysis comparing collaborating

pairs' Design Solution Quality scores to 1st best individuals' scores failed to reveal any statistical difference between them.

5.2.2 Individuals versus Pairs on Job Performance Satisfaction

Hypotheses about collaborating pairs versus individuals in terms of task satisfaction were analyzed using a 3 (1st best individual, 2nd best individual and collaborating pair) by 2 (no mind maps, mind maps) ANCOVA design. Table 5-12 shown below presents the results of this testing.

Table 5-12: ANCOVA Model Results for Job Performance Satisfaction

Source	Sum of Squares	Df	Mean Square	F-ratio	p-value
Object Oriented Programming Experience	6332.36	1	6332.36	4.423	0.032
IndivPrs	3482.80	2	1741.40	6.204	0.010
Mind Maps	4494.64	1	4494.64	8.874	0.006
IndivPrs * Mind Maps	8012.54	2	4006.27	0.581	0.726
Error	13579.47	97	139.99		
Corrected Total	35993.02	103			

ANCOVA results indicated the presence of significant main effects of both the mode of subjects' participation as individuals or collaborating pairs and use or absence of mind maps, along with no interaction effects. Due to these findings, further factor level means analysis was pursued.

Hypothesis 4 addresses Job Performance Satisfaction and comparisons between collaborating pairs and nominal pairs' averages rather than 1st best and 2nd best individuals. The hypothesis is repeated here.

H4: Average level of Job Performance Satisfaction will be higher among collaborating pairs compared to average task satisfaction levels of nominal pairs.

Table 5-13 presents the Bonferroni's custom comparison for Job Performance Satisfaction.

Table 5-13: Bonferroni's Custom Comparisons for Job Performance Satisfaction

	Mean	SD	t-value	p-value
Nominal Pairs	65.10	24.72	-3.47	0.022
Collab Pairs	56.22	30.71		

H4 was not supported. In fact, it was inversely supported, indicating that the average of nominal pairs exceeded the average of collaborating pairs.

The correlation between Design Solution Quality and Job Performance Satisfaction was reviewed previously, where it was found to be -0.119. Although the correlation is not large, it is negative. This indicates that persons who thought they did well in the task and were satisfied with their performance actually did less well than those who perceived their performance lower, ceteris paribus. That negative correlation bears out here, where the t-value indicates such.

5.2.3 Communications in the Development Process

Hypotheses 5 and 7 address increased communications in collaborating pairs regarding both Design Solution Quality and Job Performance Satisfaction. The MANCOVA results for Communications were provided in Table 5-11. Because of these findings, further factor means statistical tests were conducted for Communications in the Development Process. The hypotheses are repeated here.

H5: Increased communications in collaborating pairs in a design task will bring higher Design Solution Quality.

H7: Increased communications in collaborating pairs in a design task will bring higher Job Performance Satisfaction.

The results are presented here in Table 5-14 below:

Table 5-14: Significance of Communications in the Development Process

	Mean	SD	t-value	p-value
Nominal Pairs	75.65	22.76	2.713	0.038
Collab Pairs	88.57	41.37		

5.2.4 Cognitive Workload in the Development Process

Hypotheses 6 and 8 address decreased cognitive workload regarding both Design Solution Quality and Job Performance Satisfaction. The MANCOVA results for Communications were provided in Table 5-11. Because of these

findings, further factor means statistical tests were conducted for Cognitive Workload in the Development Process. The hypotheses are repeated here.

H6: Decreased cognitive loading in software developers in a design task will bring higher Design Solution Quality.

H7: Decreased cognitive loading in software developers in a design task will bring higher Job Performance Satisfaction.

The results are presented here in Table 5-15 below:

Table 5-15: Significance of Cognitive Workload in the Development Process

	Mean	SD	t-value	p-value
Nominal Pairs	75.65	22.76	4.193	0.011
Collab Pairs	88.57	41.37		

5.3 Mediation Testing

Perceptions of Communications in the Development Process and Cognitive Loading in the Development Process were hypothesized to mediate the relationships between the mode of participation (individuals in nominal pairs and collaborating pairs) and the second factor of distributed cognition (no mind maps and using mind maps) versus the Design Solution Quality scores and Job Performance Satisfaction. In order to test for any mediation, three regressions must be conducted and analyzed (Baron & Kenny, 1986): (1) regressing

Communications on mode of participation, (2) regressing Design Solution Quality on mode of participation and (3) regressing Design Solution Quality both on the mode of participation and on the use of mind maps as the first example. In order to test for mediation in this study, regressions were required for the relationships and are shown here in Figure 5-2:

	Design Solution Quality (DSQ)	Job Performance Satisfaction (JPS)
Communications (Comms)	IorP → Comms IorP → DSQ IorP & Comms → DSQ	IorP → Comms IorP → JPS IorP & Comms → JPS
Cognitive Workload (Cogni)	IorP → Cogni IorP → DSQ IorP & Cogni → DSQ	IorP → Cogni IorP → JPS IorP & Cogni → JPS
Communications (Comms)	MM → Comms MM → DSQ MM & Comms → DSQ	MM → Comms MM → JPS MM & Comms → JPS
Cognitive Workload (Cogni)	MM → Cogni MM → DSQ MM & Cogni → DSQ	MM → Cogni MM → JPS MM & Cogni → JPS

Legend: MM = Use of Mind Maps or not
IorP = Individuals or Pairs

Figure 5-2: Mediation Regressions Required for this Study

As can be seen from Figure 5-2, this study required eight total mediation tests to analyze the impact of Communications in the Development Process on both Design Solution Quality and Job Performance Satisfaction as well as the

impact of Cognitive Workload in the Development Process on both Design Solution Quality and Job Performance Satisfaction. All 24 regressions were conducted and 7 of the 8 mediations are valid. Failure occurred in 1 of the 8 mediation tests.

Collaborating pairs, both using mind maps and not using mind maps did not lead to higher Job Performance Satisfaction as hypothesized. As demonstrated in the tests for Hypothesis 4 concerning higher Job Performance Satisfaction, the original hypothesis failed to find support, and indeed it was found to have inverse support. Job Performance Satisfaction was lower for collaborating pairs, violating Hypothesis 4. In the mediation test for the mediating effects of Cognitive Workload on Job Performance Satisfaction, the inverse support for Hypothesis 4 was revealed to have derived from a higher Cognitive Workload in the Development Process for collaborating pairs. Although the Design solution Quality scores were higher for collaborating pairs than for individuals, the perceptions of performance were lower. This previously caused the failure of H4 and, here in the mediation tests, also caused the failure to find support for Hypothesis 8, which is now repeated here.

H8: Decreased cognitive loading in software developers in a design task will bring higher job satisfaction.

The expected decreased Cognitive Workload was determined to be, in fact, reversed and was perceived as an increased Cognitive Workload.

Hypothesis 5, 6 and 7 found support in the mediation tests of Communication in the Development Process and Cognitive Workload in the Development Process. These hypotheses are repeated here.

H5: Increased communications in collaborating software developers in a design task will bring higher quality design solutions.

H6: Decreased cognitive loading in software developers in a design task will bring higher quality design solutions.

H7: Increased communications in collaborating software development pairs will result in higher job satisfaction.

5.4 Final Regressions

There remain two hypotheses, linked by simple comparison of those who used mind maps and those who did not. Hypothesis 2 regards higher quality scores when using mind maps and Hypothesis 3 addresses higher task satisfaction when using mind maps. These hypotheses are repeated here.

H2: Design solution quality will be higher when mind maps are used during software design.

H3: Task satisfaction will be higher when mind maps are used in a software design task.

These hypotheses found support in regression analyses comparing factor means between those not using mind maps and those using mind maps. The results are shown in Table 5-16 shown here.

Table 5-16: Results of Mind Maps regressions

Hypothesis 2				
	Mean	SD	t-value	p-value
no Mind Maps	170.76	60.51	2.378	0.000
Mind Maps	189.18	57.29		
Hypothesis 3				
	Mean	SD	t-value	p-value
no Mind Maps	56.58	26.50	2.163	0.016
Mind Maps	67.04	28.76		

Support was found for benefits of using mind maps in design tasks in individuals and in pairs.

One final regression analysis was conducted to see the overall model in a multiple linear regression. Results are presented here in Table 5-17. The results include significant findings for use of mind maps or not, whether individuals or pairs, the interaction of those two factors (negative coefficient) and the covariate of object oriented programming experience.

Table 5-17: Regression of Overall Model

Coefficient	Estimate	Std Error	t-value	p-value
Mind Maps or no Mind Maps	51.3918	10.1526	4.210	0.0001
Individuals or Pairs	43.5528	10.3461	5.062	0.0000
Interaction	-14.4858	5.30946	3.101	0.0024
OO Exper	17.7101	5.71186	-2.728	0.0072

Summary Analysis of Variance Table

Source	df	SS	MS	F	p-value
Regression	4	4445942.	1111486.	335.54	0.0000
Residual	134	443883.	3312.56		
Lack of fit	11	36691.4	3335.58	1.01	0.4437
Pure Error	123	407191.	3310.5		

5.5 Summary of Hypotheses Testing

The following Table 5-19 summarizes the hypotheses testing of this research study. The majority of hypotheses were supported – six of the original nine. Those hypotheses addressing Job Performance Satisfaction, which included H4 predicting average task satisfaction among collaborating pairs to be higher than that of nominal pairs, and H8, which predicted decreased cognitive loading in software developers would bring higher task satisfaction, were not supported. Hypothesis testing for H4 found inverse support, indicating that collaborating pairs perceived poor performance satisfaction although they actually performed better than individuals.

Table 5-18: Summary of Hypotheses Testings

Hypothesis	Restatement	Finding
H1	Design solution quality of a collaborating pair will be higher than that of a second best member of a nominal pair in a software design task.	Supported
H2	Design solution quality will be higher when mind maps are used during software design.	Supported
H3	Task satisfaction will be higher when mind maps are used in a software design task.	Supported
H4	Average level of task satisfaction will be higher among collaborating pairs compared to average task satisfaction level of nominal pairs.	Not Supported (Inverse)
H5	Increased communications in collaborating software development pairs in a design task will bring higher quality design solutions.	Supported
H6	Decreased cognitive loading in software developers in a design task will bring higher quality design solutions.	Supported
H7	Increased communications in collaborating software development pairs in a design task will result in higher job satisfaction.	Supported
H8	Decreased cognitive loading in software developers in a design task will bring higher job satisfaction.	Not Supported
H9	Design solution quality of a collaborating pair using mind maps will be higher than that of the best individual developers.	Not Supported

The following chapter discusses the findings of the research study, the limitations of the study and future research possibilities.

Chapter 6

Results and Implications

This chapter presents a summary of the research study and its findings, limitations of the study and future research possibilities.

Agile programming continues to increase among practitioners (Standish Group, 2013) yet research into the dynamics of agile programming continues to be an understudied stream of research. Contributions by academic researchers have the potential to contribute to industry and practitioners while enhancing standing of agile research among both practitioners and academics. Although empirical evidence indicating benefits such as increased satisfaction, decreased error and rework, and knowledge transfer between pair programmers enhancing the software development process continues to increase, the evidence increases slowly. Any possible benefits found to facilitate efficacy in the development process would be worth investigating.

This study examined the performance of individual and pair programmers in a software analysis and design task. The efficacy of mind maps in a software analysis and design task was also investigated. In addition to these main factors, this research also examined the effects of communications - social and structural - in a distributed cognition setting, and the impact of cognitive workload on both job performance satisfaction and design solution quality.

6.1 Summary of Research Findings

The following paragraphs summarize the results of this research study.

6.1.1 Design Solution Quality

Consistent with Mangalaraj et al.'s finding, this study found that in a design task, collaborating pairs did indeed outperform the 2nd best individual in a nominal pair. Collaborating pairs using mind maps outperformed the average individual software designers, but their quality scores were not higher than the 1st best individuals. Post-hoc tests indicate that quality scores of 1st best individuals using mind maps did not significantly exceed those of 1st best individuals not using mind maps. The mean quality score of collaborating pairs using mind maps was statistically significantly higher than the scores of all other groups.

Extant literature from small groups research indicates that the performance of groups is not likely to be better than the best individuals but is likely better than the average individual. Findings from this research study are consistent with the findings from small groups research literature as well as with prior studies, such as Balijepally et al. (2009) and Mangalaraj et al. (2014) on pair development.

The efficacy of mind maps on design solution quality scores was also analyzed in this study. The impact of using mind maps on quality scores was found to be positive, leading to higher quality scores at a statistically significant

difference. By creating a mind map prior to designing the final product, mental models are elucidated individually or jointly.

6.1.2 Job Performance Satisfaction

Research literature from agile programming studies (e.g., Balijepally et al., 2009; Mangalaraj et al., 2014) list increased task satisfaction among the benefits of pair programming over individual development. In light of previous findings, this study expected that members of collaborating pairs would experience higher job performance satisfaction compared to individuals working alone. The performance satisfaction scores do not support this expectation. The job performance satisfaction scores for collaborating pairs were lower than those for individuals. This reversal was hinted at by a negative correlation between Design Solution Quality and Job Performance Satisfaction early in the analysis and was later found in the lower Job Performance Satisfaction scores for collaborating pairs. While perceiving they could have done better, their quality scores were actually higher than the scores for individuals.

Analysis for the effects on mind maps on task satisfaction yielded support for the expectation of higher job performance satisfaction for developers using mind maps, both individuals and pairs.

6.1.3 Communications in the Development Process

Increased communications in the development process was hypothesized to increase both task satisfaction and quality scores. This expectation was supported in this study. The mediation effect of communications on task satisfaction and quality scores was statistically significant.

Increased communications were expected in individuals using mind maps as structurally distributed cognition and in collaborating pairs as socially distributed cognition. Specifically, collaborating pairs using mind maps would benefit from both structural and social distributed cognition, and further, the structurally distributed cognition (mind maps) would be socially constructed as individuals prior to constructing a joint mind map socially. This increased communications among developers in a design task was supported in this study, leading to higher task satisfaction and to higher quality scores.

6.1.4 Cognitive Workload in the Development Process

Decreased cognitive workload was expected to increase both job satisfaction and quality scores. This was not supported by the findings in this research study. The researcher currently believes that this finding is the result of individuals' reports on cognitive workload being higher in both of the paired conditions, due to exposure of the individuals to other individuals' perceptions of proposed solutions. This exposure may have weakened perceptions of self-

efficacy not studied in this experiment. Previous research in pair programming highlights the need for time for the paired subjects to jell with and learn from their partners (Mangalaraj, 2006) and thereby promote learning and increase self-confidence. Without sufficient time to increase the social ease of working with a new partner, the pairs did not decrease cognitive workload but increased it. This, in turn, contributed to individuals' reports of increased cognitive workload and decreased Job Performance Satisfaction.

Higher Design Solution Quality scores did indeed stem from decreased cognitive workload, in individuals and in pairs, using mind maps or not using mind maps. This supports the idea that using external artifacts, whether notes and sketches or mind maps, lowers cognitive workload as was predicted in the literature.

Higher Job Performance Satisfaction was not found in this study. Individuals, using mind maps or not, reported higher task satisfaction than collaborating pairs, using mind maps or not. Decreased cognitive workload as a mediator was degraded because it did not work for collaborating pairs. The pairs reported a higher perceived cognitive workload than the individuals, regardless of the use of mind maps, leading to lower task satisfaction scores. This inverse finding was statistically significant, and is not well supported by the literature review of this study.

6.2 Significance of Findings

This study has made important contributions to theoretical research into agile programming, both in the individual and pair programming contexts. Both individuals and pairs used mind maps to "pre-engineer" their thoughts on software design solutions. Mind maps contributed to increases in solution quality and task satisfaction. Although pair programming is gaining popularity in practice, research into the dynamics of pair programming remains sparse. Prior studies on pair development suggest that communications and coordination losses occurring during collaboration may be the reason why pairs seldom outperform best individuals, although they tend to do better than second-best individuals (e.g., Balijepally et al., 2009; Mangalaraj et al., 2014). One of the goals of this experiment, therefore, was to examine the role of mind maps in facilitating communication and coordination during software development. This study also analyzed potential mediators concerning cognitive loading and communications in the software design process, which has often been described as an ill-structured activity.

6.2.1 Significance of Findings for Research

Although pair programming has an academic literature stream, research into the dynamics involved in the process remains sparse. Further, studying the

design aspects of software development specifically for academic research is even rarer. Processes in software design are still in need of research. The social-technical aspects of this study highlight the need for research into individual differences not captured by using object oriented programming experience as a covariate, yet none of the other possible covariates were statistically significant. By utilizing nominal pairs, this study increased the rigor of the findings, where many previous studies used equal number of individuals and pairs and included twice as many persons working in the pair condition as the individual condition. Important outcomes of this research study are presented here. First, this research study confirmed that the collaborating pair will produce higher quality scores than the 2nd best individual. This finding is consistent with research literature focused on small groups.

Second, this study found task satisfaction to be higher in individuals but not in collaborating pairs. Because this finding is not consistent with small group literature, it requires further study.

Third, using communication as a mediator in the development process was successful, increasing both task satisfaction and quality scores. While many studies have shown the direct effects of pairing on task satisfaction and pairing, few have looked into the actual processes that mediate the relationships. This establishes a new possibility for further research to investigate the different

mechanisms (e.g., distributed cognition and external representations) that might facilitate communications.

Fourth, cognitive workload in the development process did not succeed as a mediator for one condition (pairs) but did mediate three different conditions. For this mediator to work this way was not expected and requires further study. Few studies in pair programming have found this result, while most find the proposed hypothesis which is the reverse of this study's result. This experiment did not control for self-efficacy in software development, although object oriented programming experience was a significant covariate. It was suggested earlier in these findings that cognitive workload may have been reported as increasing when exposed to other individuals' perceptions of the best solution, without sufficient time to jell, establish trust and learn from the individual's partner. If so, future research should include some measure of self-efficacy in order to establish a better control in the experiment. If this were done, at least these findings might be more explainable and further rigor would be added to the research, increasing confidence in the findings.

Fifth, mind maps were used in two of the four Session conditions and increased both quality scores and task satisfaction overall. Mind maps represent external artifacts and seemed to work well in capturing thought processes and building mental models of a design task in software development, a task that is difficult to start. Mind maps force a hierarchy without pressing priorities. Using a

distributed cognition setting and recognizing the social and structural aspects of both having a design partner and of mind maps as external representations, this study found that mind maps increased design solution quality and task performance satisfaction. Further research into use of artifacts and external representations contains exciting possibilities. Other aspects of software design could be studied using mind maps as a structural distributed cognition tool.

6.2.2. Significance of Findings for Practitioners

Agile programming continues to increase in practice and in popularity and practitioners expect these techniques to reduce information systems' project failure rates (Standish Group, 2013). Systems that promote more thoughtful artifacts for use in software analysis & design are becoming more popular among practitioners. Therefore, this study's findings are significant to practitioners in many ways.

First, the benefits of collaborative work, such as higher quality and increased task satisfaction, are demonstrated here, although mitigated by higher cognitive loading among collaborating pairs. Practitioners could use the extant distributed cognition aspects of socially distributed cognition and add mind maps as a valid socially constructed structural aspect of pair programming.

Second, hiring exceptional individual developers in industry is an increasingly difficult task. The finding that pairs of average developers using

mind maps can produce the equivalent quality allows more feasible options to the practicing development manager

Third, knowledge capture and transfer in design tasks can be improved by external artifacts, specifically mind maps. Tools that capture knowledge will become increasingly important, and mind maps are simple to construct and simple to use.

Fourth, the mediators of communications and cognitive loading can be employed in industry to facilitate better quality of design. While communications and coordination are costs in the beginning of a paired development team, time to build trust and to jell as a team is expected in practice, and difficult to reproduce in a lab experiment. Longitudinal research experiments would be required to address this shortcoming in academic studies, but practitioners already employ the social and structural aspects of distributed cognition. Directly addressing the communications and cognitive workload aspects of the experiment presented here could improve the agile programming results for practitioners.

6.3 Limitations of Study

There are certainly limitations present in this study. All academic studies suffer from a lack of environmental aspects pertaining to the specific area of investigative focus. Generalizability of the findings is of concern for those experiments designed to address existing industrial practices such as agile

programming. While this study did indeed use students as experimental subjects, this was mitigated somewhat by including graduate students, most of whom are working in the field already. Their experience brings additional validity to the findings regarding design as an often ill-structured task. Additionally, the main problem to be solved in this study might be considered trivial in an industrial setting, although it is based on foundational industry standards and used UML class diagrams, a common method of capturing design decisions in software development.

First, this was a lab study under controlled conditions of execution. Important situational variables which might be present in a workplace may influence the results. The laboratory-based environment, however, is necessary to produce the controls required for research and cannot be avoided by academic researchers.

Second, the pairs in this study were randomly chosen, not the usual practice for practitioners. Pairs in this study were mostly strangers, having met and worked together only during their time in the lab. Previous research indicates that the performance of pairs will improve over time as the pair jells with each other (Williams et al., 2000). The warm-up task in this study was performed individually in order to acquaint the subjects with the online tools used in the study. The mind maps were produced individually, even in the collaborating pair condition Session D, before the joint mind map was produced. The final UML

class diagram came after the joint mind map, but the total time working together prior to the UML class diagram for was about 13 minutes. Increasing the time to jell might increase the quality produced by the pair.

Third, mind maps in this research study were introduced, trained and used in a relatively short amount of time during the experiment. Increased cognitive loading in collaborating pairs was somewhat mitigated by using mind maps, but the exact effect remains unknown at this time.

Fourth, a single design task was used for all condition Sessions. Although this bolsters the reliability of the findings, practitioners are aware that small groups and dyads perform inconsistently based on the characteristics of the task itself. Use of external artifacts changes the way the performer perceives the task (Zhang & Norman, 1994) and this specific finding needs further work before practitioners can make use of it. In order to properly ascertain the effects of the task on solution quality and perceptions of task satisfaction more research would be required (Balijepally, 2006).

6.4 Future Research Directions

This study remains an early point in research into software design examining pair programming phenomena. The social and behavioral aspects of the software design process overall remain an area rich in opportunity. The increasing use of technology to assist in designing technology continues to call for

further research in an environment that is changing rapidly. As such, there are many opportunities for further research and some are outlined here below.

First, a longitudinal study could be beneficial to discern whether pair performance improves in sequential tasks. Longitudinal studies help untangle and discern learning effects present during small group work. Extant literature on small group work can be reviewed for longitudinal studies and for learning effects in longitudinal studies, setting the stage for longitudinal studies in software design research.

Second, this study randomly paired subjects into collaborating pairs. Further research opportunities are available in the area of creating pairs based on specific demographics, abilities and knowledge levels and creating pairs based on these characteristics rather than random pairing. This direct approach to using the individual differences found in subjects could extract interesting findings regarding creativity, innovation, technical engineering thought and approaches and could possibly unveil unexpected findings.

Third, the use of mind maps can be continued, expanding into other design tasks and into software development tasks outside of strictly design tasks. Mind maps proved to be a simple tool to introduce, train and use, making them attractive in other areas of research such as building joint mental models or problem definition and solution. Knowledge sharing and learning in the problem-

solving task should be examined as well as aspects of creativity and innovation in the design tasks.

Fourth, research into the similarities and differences between mind maps and other conceptual tools could be sponsored in order to further develop understanding of mental models among individuals and pairs. One research paper in mathematics posited that concept maps align with exploratory analysis and that mind maps are more akin to confirmatory analysis, for example.

Fifth, multiple tasks could be used instead of a single task. This would allow researchers to further refine the impact of task characteristics on productivity, quality, cognitive loading and task satisfaction. In addition, this would allow a slightly more realistic result closer to industry practices.

Sixth, previous research into agile programming has used the best individual in a nominal group as a standard for judging performance of pairs. However, the social aspects of individual differences, and the contextual and process variables in such research, would be a very interesting area of research in the elusive search for pairs and groups that could outperform the best individuals.

Sixth, further research into why the increased cognitive loading in collaborating pairs leads to lower task satisfaction while actually producing higher quality scores, whereas individuals with higher task satisfaction obtained lower quality scores, would be fascinating and might yield important insights into self-perceptions of software developers in a design task.

6.5 Conclusions

This research study empirically scrutinized important concepts in software design. This study exists in a sparsely populated stream of research examining some of the more difficult aspects of software development. By using the two factors of mode of participation (individuals versus pairs) and external artifacts (using mind maps or not), this study furthers research into improvement in software design tasks. Results of the study indicate that using mind maps improves solution quality and task satisfaction in both individuals and pairs. Results also indicate that communications in the development process increases solution quality and task satisfaction across the board. This study found that decreased cognitive loading in individuals did lead to increases in task satisfaction, but collaborating pairs had lower task satisfaction while garnering higher solution quality.

Appendix A

Copy of Participant Recruitment Flyer

HEY, INSY MAJORS!!

DO YOU EVEN SCIENCE?

Well, here's your chance!!

We need advanced undergraduate and graduate student participants in an Analysis & Design experiment over in the Business College

- Learn to create UML class diagrams
- Experiment done in COBA 339
- The experiment will be 2 hours
- Three $50 Walmart cards and three $25 Walmart cards as door prizes in random drawings during each session of the experiment
- Four total sessions: Saturday, 10am 12 Sep, 12:30pm 12 Sep
  Sunday, 10am 13 Sep, 12:30pm 13 Sep

Just go to: http://tinyurl.com/pfpnyk4 and sign up for a session!

Principal Investigator: Philip L. Bond, PhD candidate, ISOM Dept., COBA
Contact: 817-272-3094 philip.bond@mavs.uta.edu COBA Room 516
Faculty Advisor: Dr. Nerur, ISOM Dept., COBA
This research is approved by the U.T. Arlington Institutional Review Board, 2015-0129

Appendix B

Copy of Institutional Review Board-Approved

Informed Consent Form

INFORMED CONSENT

PRINCIPAL INVESTIGATOR NAME:
Philip L. Bond, PhD student
Information Systems and Operations Management Department
Room 516, COBA philip.bond@mavs.uta.edu 817-272-3084
Faculty Advisor: Dr. Sridhar Nerur
Information Systems and Operations Management Dept.
Room 518, COBA snerur@uta.edu 817-272-3530

TITLE OF PROJECT:
An Investigation of the Efficacy of Mind Maps In Software Development Among Individuals and Pairs, IRB 2015-0129

INTRODUCTION
You are being asked to participate in a research study. Your participation is voluntary. Please ask questions if there is anything you do not understand.

PURPOSE:

The purpose of the research is to identify factors of quality and completeness of software analysis and design using Universal Markup Language (UML) class diagrams commonly used in the software industry. You will learn how to construct UML class diagrams through a presentation lecture and then produce UML class diagrams as solutions to a problem statement. UML diagrams are often used in software projects to document the requirements analysis and to guide design of the software system. UML diagrams are used in advanced IS coursework and textbooks and are common in the software industry.

UML class diagrams are pictures of the relationships between classes in a software system. Each class will have attributes (the class "Student" would have FirstName, LastName, StudentID, Major and so on) and also have methods (Student would be able to Enroll, Drop, Graduate and so on). Attributes are often called things the classes "are" and methods are things the classes "do." Other classes which interact with Student should be able to use the attributes and methods of the Student class. For example, Registrar should also have methods which Enroll, Drop or Graduate a Student. By examining the attributes and methods of multiple classes in a software system, good software analysts and designers will produce quality UML diagrams which have all the necessary attributes and methods, but no extra needless information.

Factors which may produce better UML diagrams are the focus of this experiment, specifically using mind maps before constructing the UML diagrams, both individually and in pairs. Therefore, different sessions of the experiment will either feature individuals or pairs and either using mind maps or not, as factors influencing the quality and completeness of the final UML diagram solution. You will be asked to produce a UML class diagram as a solution to a problem statement.

DURATION:
Approximately 2 hours

PROCEDURES:

1) You will arrive at the computer lab, sign in, receive a random number for confidentiality, and sign the Informed Consent form. Your random number will be used to separate your personal identity from your submitted work as well as be used in random drawings for door prizes.

2) You will receive training on Universal Markup Language (UML) class diagrams, and discuss examples, their usage and their specifics. UML class diagrams are used in software projects to document the attributes and methods of various classes in a software system. Building good UML class diagrams helps software analysts and designers ensure that the specifics of each class are addressed by other classes in the system. For example each class will have attributes (the class "Student" would have FirstName, LastName, StudentID, Major and so on) and also have methods (Student would be able to Enroll, Drop, Graduate and so on). Attributes are often called things the classes "are" and methods are things the classes "do."

3) You will then register at www.gliffy.com in order to use that website to create UML class diagrams. This will require you to provide your name and email address and accept their terms of conditions.

4) A warm-up task with a problem statement will be discussed and approximately 20 minutes will be given to solve the problem by producing a UML class diagram.

5) If you participate in sessions with only individuals not using mind maps, you will then have the remaining time to produce a UML class diagram for solving a different, main problem statement. Sessions with pairs not using mind maps will then have the remaining time to produce a UML class diagram for solving a different, main problem statement.

If you participate in sessions using mind maps, both individuals and in pairs, you will receive training in mind maps, discussing examples their usage and their specifics after the warm-up task. You will then register at www.mindmup.com in order to use that website to create mind maps. This will require you to provide your name and email address and accept their terms of conditions.

All participants using mind maps will complete a mind map as a solution to the main problem statement. Sessions with individuals using mind maps will then have the remaining time to complete a UML class diagram for solving the main problem statement. Sessions with pairs will then pair up randomly to produce a joint UML class diagram for solving the main problem statement.

6) At the end of the experiment time, you will be asked to complete the Questionnaire. The Questionnaire will be of four types: individuals without mind maps, pairs without mind maps, individuals with mind maps and pairs with mind maps. Each session of the experiment will only have one Questionnaire type specific to that session. The front page of the Questionnaire which contains demographic information and previous software experience will be separated

from the remaining Questionnaire in order to screen the Questionnaire answers from nationality, race, gender, culture and other factors.

POSSIBLE BENEFITS:
The main benefit to the participant from this research is to know you are contributing to further future knowledge. Other possible benefits include learning UML class diagramming if currently unknown and learning more about mind mapping.

COMPENSATION:
Possible compensation will be through random drawings for ten $50 Walmart gift cards.

POSSIBLE RISKS/DISCOMFORTS:
Possible fatigue in a classroom setting, boredom.

ALTERNATIVE PROCEDURES/TREATMENTS:
No alternatives in the initial study.

WITHDRAWAL FROM THE STUDY:
Experimental subjects may withdraw from the study at any time.

NUMBER OF PARTICIPANTS: We expect 200 participants to enroll in this study.

CONFIDENTIALITY:
We are not collecting signatures or personal information attached to your answers. The information collected from you during this study will not be linked to your identity. Your signed consent form will be retained separately from the data, and your responses will be coded by using a random number attached to your submitted answers.

If in the unlikely event it becomes necessary for the Institutional Review Board to review your research records, then The University of Texas at Arlington will protect the confidentiality of those records to the extent permitted by law. Your research records will not be released without your consent unless required by law or a court order. The data resulting from your participation may be made available to other researchers in the future for research purposes not detailed within this consent form. In these cases, the data will contain no identifying information that could associate you with it, or with your participation in any study.

CONTACT FOR QUESTIONS:
Questions about this research may be directed to Philip L. Bond, Principal Investigator, at philip.bond@mavs.uta.edu, or Room 516, College of Business Administration, or 817-272-3084. The Faculty Advisor for this protocol is Dr. S. Nerur, and may be contacted at 817-272-3530. Any questions you may have about your rights as a research participant may be directed to the Office of Research Administration; Regulatory Services at 817-272-2105 or to regulatoryservices@uta.edu.

CONSENT:

By signing below, you confirm that you are 18 years of age or older and have read or had this document read to you. You have been informed about this study's purpose, procedures, possible benefits and risks, and you have received a copy of this form. You have been given the opportunity to ask questions before you sign, and you have been told that you can ask other questions at any time.

You voluntarily agree to participate in this study. By signing this form, you are not waiving any of your legal rights. Refusal to participate will involve no penalty or loss of benefits to which you are otherwise entitled. You may discontinue participation at any time without penalty or loss of benefits, to which you are otherwise entitled.

Signature:

By signing below, you confirm that you have read this document or had this document read to you.

SIGNATURE OF VOLUNTEER **DATE**

APPROVED

APR 15 2015

Appendix C

Copy of Questionnaires for Sessions

Questionnaire “A”

Demographics Questions

1) What is your gender? ☐ Female ☐ Male

2) What is your age? _____

3) What is your major course of study? ☐ INSY ☐ CSE

3) Highest education level? ☐ High School ☐ Technical School

☐ Undergraduate degree ☐ Graduate degree

☐ Doctorate degree ☐ Other:

_____

4) How many hours per week do you work? _____

5) Please write the country of your citizenship:

_____

6) Is English your first language? ☐ Yes ☐ No

7) Your overall GPA _____

8) GPA in your major _____

Software Background Questions

1) How many years of programming experience do you have?

☐ 0 to 1 ☐ 1 to 2 ☐ 2 to 4 ☐ 4 to 6 ☐ 6 or more

2) How many years of programming experience in object-oriented languages?

☐ 0 to 1 ☐ 1 to 2 ☐ 2 to 4 ☐ 4 to 6 ☐ 6 or more

3) What level of experience do you have in object-oriented design?

☐ No experience ☐ Novice ☐ Intermediate ☐ Expert

4) What object-oriented programming languages are you familiar with?

☐ C++ ☐ C# ☐ Java ☐ Python ☐ Small Talk

☐ Eiffel ☐ VB ☐ .Net ☐Objective-C ☐ Other

Task Load Index

We would like to know about the mental task load you experienced in during this experiment. Feelings of workload come from several different things. Some people feel that mental or time demands are the most important factors while others may feel that their performance or frustration level is the most important part.

Here are the factors that contribute to mental task load.

Factor	Definition
Mental Demand	How much mental 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?

Using the 5 definitions above, for these next 10 questions, choose and mark 1 of the 2 boxes to show which factor was more important to you.

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 the following questions, mark an "X" on each scale at the point that matches your experience in this experiment. Thank you for helping us.

The level of Mental Demand for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

The Time Demand for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

The level of Effort for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

The Frustration Level for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

My level of Performance on this task was:

Good |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| Poor

Software Design Quality Alone

1) I explored multiple solutions for the given problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2) I found flaws with the design and made changes

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3) I could easily identify the right solutions to the problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

Task satisfaction

1) How do you feel about your time spent working on the solution 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 Content

Absolutely Terrible 1 2 3 4 5 6 7 Absolutely Delighted

2) How do you feel about the work 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

Questionnaire “B”

Demographics Questions

1) What is your gender? ☐ Female ☐ Male

2) What is your age? _____

3) What is your major course of study? ☐ INSY ☐ CSE

3) Highest education level? ☐ High School ☐ Technical School

☐ Undergraduate degree ☐ Graduate degree

☐ Doctorate degree ☐ Other:

_____

4) How many hours per week do you work? _____

5) Please write the country of your citizenship:

_____

6) Is English your first language? ☐ Yes ☐ No

7) Your overall GPA _____

8) GPA in your major _____

Software Background Questions

1) How many years of programming experience do you have?

☐ 0 to 1 ☐ 1 to 2 ☐ 2 to 4 ☐ 4 to 6 ☐ 6 or more

2) How many years of programming experience in object-oriented languages?

☐ 0 to 1 ☐ 1 to 2 ☐ 2 to 4 ☐ 4 to 6 ☐ 6 or more

3) What level of experience do you have in object-oriented design?

☐ No experience ☐ Novice ☐ Intermediate ☐ Expert

4) What object-oriented programming languages are you familiar with?

☐ C++ ☐ C# ☐ Java ☐ Python ☐ Small Talk

☐ Eiffel ☐ VB ☐ .Net ☐Objective-C ☐ Other

Task Load Index

We would like to know about the mental task load you felt during this experiment. Feelings of workload come from several different things. Some people feel that mental or time demands are the most important factors while others may feel that their performance or frustration level is the most important part.

Here are the factors that contribute to mental task load.

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?

Using the 5 definitions above, for these next 10 questions, choose and mark 1 of the 2 boxes to show which factor was more important to you.

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 the following questions, mark an "X" on each scale at the point that matches your experience in this experiment. Thank you for helping us.

The level of Mental Demand for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

The Time Demand for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

The level of Effort for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

The Frustration Level for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

My level of Performance on this task was:

Good |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
Poor

Communications Alone

1) I helped myself with thoughts and ideas by using external notes and drawings

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

2) My notes and drawings helped me keep ideas and thoughts fresh in my mind

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

3) Creating and using a mind map, even alone, helped me form solutions to the design task

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

4) Using mind maps first helped me design a better solution

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

5) Using mind maps before design increased my confidence in my design solution

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

Using Mind Maps before Designing

1) Creating the mind map before design made the design task easier

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

2) Mind maps give flexibility/extensibility to the design created

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

3) Having a partner helped in creating and using better mind maps

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

4) Please briefly state any problems in using mind maps for the design task performed

_____

_____

5) In what ways did using mind maps help you in solving the design problem today?

_____

_____

Software Design Quality Alone

1) I explored multiple solutions for the given problem

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

2) I found flaws with the design and made changes

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

3) I could easily identify the right solutions to the problem

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

Task satisfaction

1) How do you feel about your time spent working 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 Content
Absolutely Terrible	1	2	3	4	5	6	7	Absolutely Delighted

2) How do you feel about the work 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

Questionnaire “C”

Demographics Questions

1) What is your gender? ☐ Female ☐ Male

2) What is your age? _____

3) What is your major course of study? ☐ INSY ☐ CSE

3) Highest education level? ☐ High School ☐ Technical School

☐ Undergraduate degree ☐ Graduate degree

☐ Doctorate degree ☐ Other:

_____

4) How many hours per week do you work? _____

5) Please write the country of your citizenship:

_____

6) Is English your first language? ☐ Yes ☐ No

7) Your overall GPA _____

8) GPA in your major _____

Software Background Questions

1) How many years of programming experience do you have?

☐ 0 to 1 ☐ 1 to 2 ☐ 2 to 4 ☐ 4 to 6 ☐ 6 or more

2) How many years of programming experience in object-oriented languages?

☐ 0 to 1 ☐ 1 to 2 ☐ 2 to 4 ☐ 4 to 6 ☐ 6 or more

3) What level of experience do you have in object-oriented design?

☐ No experience ☐ Novice ☐ Intermediate ☐ Expert

4) What object-oriented programming languages are you familiar with?

☐ C++ ☐ C# ☐ Java ☐ Python ☐ Small Talk

☐ Eiffel ☐ VB ☐ .Net ☐Objective-C ☐ Other

Task Load Index

We would like to know about the mental task load you experienced in during this experiment. Feelings of workload come from several different things. Some people feel that mental or time demands are the most important factors while others may feel that their performance or frustration level is the most important part.

Here are the factors that contribute to mental task load.

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?

Using the 5 definitions above, for these next 10 questions, choose and mark 1 of the 2 boxes to show which factor was more important to you.

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 the following questions, mark an "X" on each scale at the point that matches your experience in this experiment. Thank you for helping us.

The level of Mental Demand for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

The Time Demand for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

The level of Effort for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

The Frustration Level for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____| High

My level of Performance on this task was:

Good |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
Poor

Communications with Your Partner

1) Both partners posed adequate questions to each other in order to understand the learning content (e.g. questions on meaning of concepts, differences, reasons and concrete examples)

Strongly disagree 1 2 3 4 5 Strongly agree

2) What both members said was checked by asking each other critical questions

Strongly disagree 1 2 3 4 5 Strongly agree

3) A partner who was formulating an explanation about the problem asked his/her partner whether his/her explanation was right

Strongly disagree 1 2 3 4 5 Strongly agree

4) One explanation did not suffice for partners; alternative explanations were also mentioned

Strongly disagree 1 2 3 4 5 Strongly agree

5) Partners discussed and elaborated on each other's arguments

Strongly disagree 1 2 3 4 5 Strongly agree

6) When someone argued something, then that statement was motivated

Strongly disagree 1 2 3 4 5 Strongly agree

7) Explanations of group members were completed with explanations of other group members

Strongly disagree 1 2 3 4 5 Strongly agree

8) I and my partner drew conclusions from the information that was discussed in the group

Strongly disagree 1 2 3 4 5 Strongly agree

9) My partner and I experienced contradictions about the information we discussed

Strongly disagree 1 2 3 4 5 Strongly agree

10) One or more group members was/were contradicted by the others

Strongly disagree 1 2 3 4 5 Strongly agree

11) When one partner contradicted the other partner, both partners discussed it

Strongly disagree 1 2 3 4 5 Strongly agree

Working in Pairs

1) Overall, we recognized and used each other's unique skills

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

2) Overall, we could relate to each other's unique expertise

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

3) I could see the value of my partner's expertise

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

4) What were the greatest benefits of working with a partner to solve the design task today?

_____

_____

Design Quality with Your Partner

1) We explored multiple solutions for the problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2) We helped each other in finding flaws with the design solution

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) Working with a partner increased my confidence in our design solution over working alone

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

6) Working with a partner increased the quality of the design solution over working alone

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

7) Before today's task performance, have you ever worked with your partner?

No Yes

Task satisfaction

1) How do you feel about your time spent working with your partner 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 Content
Absolutely Terrible	1	2	3	4	5	6	7	Absolutely Delighted

2) How do you feel about the work you performed with a partner today?

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

Questionnaire “D”

Demographics Questions

1) What is your gender? ☐ Female ☐ Male

2) What is your age? _____

3) What is your major course of study? ☐ INSY ☐ CSE

3) Highest education level? ☐ High School ☐ Technical School

☐ Undergraduate degree ☐ Graduate degree

☐ Doctorate degree ☐ Other:

_____

4) How many hours per week do you work? _____

5) Please write the country of your citizenship:

_____

6) Is English your first language? ☐ Yes ☐ No

7) Your overall GPA _____

8) GPA in your major _____

Software Background Questions

1) How many years of programming experience do you have?

☐ 0 to 1 ☐ 1 to 2 ☐ 2 to 4 ☐ 4 to 6 ☐ 6 or more

2) How many years of programming experience in object-oriented languages?

☐ 0 to 1 ☐ 1 to 2 ☐ 2 to 4 ☐ 4 to 6 ☐ 6 or more

3) What level of experience do you have in object-oriented design?

☐ No experience ☐ Novice ☐ Intermediate ☐ Expert

4) What object-oriented programming languages are you familiar with?

☐ C++ ☐ C# ☐ Java ☐ Python ☐ Small Talk

☐ Eiffel ☐ VB ☐ .Net ☐Objective-C ☐ Other

Task Load Index

We would like to know about the mental task load you experienced in during this experiment. Feelings of workload come from several different things. Some people feel that mental or time demands are the most important factors while others may feel that their performance or frustration level is the most important part.

Here are the factors that contribute to mental task load.

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?

Using the 5 definitions above, for these next 10 questions, choose and mark 1 of the 2 boxes to show which factor was more important to you.

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 the following questions, mark an "X" on each scale at the point that matches your experience in this experiment. Thank you for helping us.

The level of Mental Demand for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
High

The Time Demand for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
High

The level of Effort for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
High

The Frustration Level for this task was:

Low |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
High

My level of Performance on this task was:

Good |_____|_____|_____|_____|_____|_____|_____|_____|_____|_____|
Poor

Communications Alone

1) I helped myself with thoughts and ideas by using external notes and drawings

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

2) My notes and drawings helped me keep ideas and thoughts fresh in my mind

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

3) Creating and using a mind map, even alone, helped me form solutions to the design task

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

4) Using mind maps first helped me design a better solution

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

5) Using mind maps before design increased my confidence in my design solution

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

Communications with Your Partner

1) Both partners posed adequate questions to each other in order to understand the learning content (e.g. questions on meaning of concepts, differences, reasons and concrete examples)

Strongly disagree 1 2 3 4 5 Strongly agree

2) What both members said was checked by asking each other critical questions

Strongly disagree 1 2 3 4 5 Strongly agree

3) A partner who was formulating an explanation about the problem asked his/her partner whether his/her explanation was right

Strongly disagree 1 2 3 4 5 Strongly agree

4) One explanation did not suffice for partners; alternative explanations were also mentioned

Strongly disagree 1 2 3 4 5 Strongly agree

5) Partners discussed and elaborated on each other's arguments

Strongly disagree 1 2 3 4 5 Strongly agree

6) When someone argued something, then that statement was motivated

Strongly disagree 1 2 3 4 5 Strongly agree

7) Explanations of group members were completed with explanations of other group members

Strongly disagree 1 2 3 4 5 Strongly agree

8) I and my partner drew conclusions from the information that was discussed in the group

Strongly disagree 1 2 3 4 5 Strongly agree

9) My partner and I experienced contradictions about the information we discussed

Strongly disagree 1 2 3 4 5 Strongly agree

10) One or more group members was/were contradicted by the others

Strongly disagree 1 2 3 4 5 Strongly agree

11) When someone contradicted a group member, that person stated a counter-argument

Strongly disagree 1 2 3 4 5 Strongly agree

Using Mind Maps before Designing

1) Creating the mind map before design made the design task easier

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2) Mind maps give flexibility/extensibility to the design created

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3) Having a partner helped in creating and using better mind maps

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4) Please briefly state any problems in using mind maps for the design task performed

_____

_____

5) In what ways did using mind maps help you in solving the design problem today?

_____

_____

Working in Pairs

1) Overall, we found and used each other's unique skills

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2) Overall, we could relate to each other's unique expertise

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

3) I could see the value of my partner's expertise

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

4) What were the greatest benefits of working with a partner to solve the design task today?

_____

_____

Software Design Quality Alone

1) I explored multiple solutions for the given problem

Strongly disagree 1 2 3 4 5 6 7 Strongly agree

2) I found flaws with the design and made changes

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

3) I could easily identify the right solutions to the problem

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

Software Design Quality with Your Partner

1) We explored multiple solutions for the given problem

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

2) We helped each other in finding flaws with the design solution

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) Working with a partner increased my confidence in our design solution over working alone

Strongly disagree 1 2 3 4 5 6 7
Strongly agree

6) Working with a partner increased the quality of the design solution over working alone

Strongly disagree	1	2	3	4	5	6	7	Strongly agree

7) Before today's task performance, have you ever worked with your partner?

No Yes

Task satisfaction

1) How do you feel about your time spent working alone 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 Content
Absolutely Terrible	1	2	3	4	5	6	7	Absolutely Delighted

2) How do you feel about the work you performed alone today?

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

3) How do you feel about your time spent working with your partner 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 Content
Absolutely Terrible	1	2	3	4	5	6	7	Absolutely Delighted

4) How do you feel about the work you performed with a partner today?

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

Appendix D

Copy of Problem Statement for Experimental Sessions

IRB 2015-0129: Philip L. Bond, Principal Investigator philip.bond@mavs.uta.edu Room 516, COBA
Faculty Advisor: Dr. Sridhar Nerur snerur@uta.edu Room 518, COBA

Rick's Guitars and Strings – Search Program Write-up and Problem Statement

Rick's Guitars and Strings needs a new search engine so that Rick's customers can search his store's inventory to find instruments to buy. Rick sells guitars, mandolins, banjos, fiddles and basses.

Your task is to create a UML class diagram illustrating the classes, their attributes and methods, and their associations with other classes, to build a search engine for Rick's Guitars and Strings inventory. The specifications for a customer's search are based on the attributes of the various instruments carried in the store's inventory. Below is a list of the specifications the search engine should return:

-serialNumber: A unique number assigned to each instrument
-Price1: The price paid by Rick for the instrument
-Price2: The price charged by Rick for the instrument
+Builder: The brand name of the instrument, such as Fender, Gibson, and so on
+Model: The name of the model of instrument, e.g., Stratocaster and so on
+Type: The type of instrument, e.g., Acoustic or Electric
+topWood: The wood on the face of the instrument, such as Spruce, Cedar, Mahogany and so on
+backWood: The wood used on the back of the instrument, e.g., Maple, Rosewood and so on
+Finish: The color or type of paint or lacquer on the instrument

The classes you build in your UML class diagram should allow a customer to specify an instrument's Builder, Model, Type, topWood, backWood or Finish. Those specifications should be used to search the inventory for instruments that match the customer's desires. The serial number will not be specified by the searching customer; it is a result of the search. Each serial number is a unique identifier for that one instrument. Rick also tracks two prices for each instrument – one is the price Rick paid for the instrument and the other is the price he wants the customer to pay. These two prices are set when the instrument is entered into the inventory system. These prices are a result of the search and not specified by the customer.

Not all instruments have the same specifications list. Guitars vary by Builder, Model, Type, Top Wood, and Back Wood. Mandolins vary by Builder, Model, Type, Top Wood and Back Wood. Banjos also have a Builder and Model, but do not have Top Wood; they are topped with stretched plastic. Fiddles have a Builder, Model, Top Wood and Back Wood but also vary in Finish. Basses, like Fiddles, also often vary in Finish, though they have a Builder, Model, Top Wood and Back Wood.

Your task is to create a UML class diagram illustrating the classes, their attributes and methods, and their associations with other classes, to build a search engine for Rick's Guitars and Strings inventory. The customer will enter various choices in order to search Rick's inventory and the search engine should return maximum results based on the customer's search entries.

Hint: You should construct a class of instrument specification that will work with one type of instrument and then extend that to include other types of instruments.

Appendix E

Copy of Grading Rubric

2015-0129 Grading Rubric

Thank you for your time and effort in grading the results of my experiment!

~~~~~~~~~~~~~~

The guiding principle of grading these results is to award points for good design, whenever and wherever possible

~~~~~~~~~~~~~~

Award points for each diagram but remember the points below are based on separate accomplishments. We will award 50 pts for each of the following:

- extending multiple classes of instruments, like “Guitar,” “Mandolin,” etc.
- extending from a superclass “Instrument” or “InstrumentSpecs” or so
- including a “Inventory” class related to “Instrument” class
- including a search, either as a “Search” class or as a method
- including a “Customer” class

  If all of the above are present, the diagram would gain 250 points total. Not including two of the above five would yield 150 points.

Additional points possible include:
10 pts for assigning the verbs of association, like “extends” or even “search”
5 pts for multiplicities
5 pts for other nice things like color, interesting line styles, creative fonts, etc

Appendix F

Specific Results of Mediation Test Regressions

As mentioned in Chapter 5, testing for mediation requires three regressions per test grouping (Baron and Kenny, 1986). Having two factors for independent variables, two possible mediators under test and also two dependent variables yields eight required test groupings for mediation. Eight test groupings, requiring three regressions each, yields twenty-four total regressions. Figure 5-2 is repeated here to illustrate the eight test groupings, each with three regressions:

	Design Solution Quality (DSQ)	Job Performance Satisfaction (JPS)
Communications (Comms)	IorP → Comms IorP → DSQ IorP & Comms → DSQ	IorP → Comms IorP → JPS IorP & Comms → JPS
Cognitive Workload (Cogni)	IorP → Cogni IorP → DSQ IorP & Cogni → DSQ	IorP → Cogni IorP → JPS IorP & Cogni → JPS
Communications (Comms)	MM → Comms MM → DSQ MM & Comms → DSQ	MM → Comms MM → JPS MM & Comms → JPS
Cognitive Workload (Cogni)	MM → Cogni MM → DSQ MM & Cogni → DSQ	MM → Cogni MM → JPS MM & Cogni → JPS

Legend: MM = Use of Mind Maps or not
IorP = Individuals or Pairs

Figure 5-2: Mediation Regressions Required for this Study

Starting with the four test groupings focused on Design Solution Quality (DSQ), the Individuals or Pairs factor set the first test grouping of Individuals or

Pairs (IorP)→ Communications (Comms) → Design Solution Quality (DSQ) shown here below in Table F-1:

Table F-1: Regressions for Communications as a mediator between the Individuals or Pairs factor and the Design Solution Quality dependent variable

Step	Independent Variable	Dependent Variable	t-value	p-value
1	IorP	Comms	2.138	0.034
2	IorP	DSQ	2.440	0.016
3	IorP & Comms	DSQ	2.768	0.006

The results indicate that Communications is a valid mediator between Individuals or Pairs and Design Solution Quality, as the p-values are each and all significant.

Continuing with the four test groupings from Design Solution Quality, the Individuals or Pairs factor set the second test grouping of Individuals or Pairs (IorP) → Cognitive Workload (Cogni) → Design Solution Quality shown here below in Table F-2:

Table F-2: Regressions for Cognitive Workload as a mediator between the Individuals or Pairs factor and the Design Solution Quality dependent variable

Step	Independent Variable	Dependent Variable	t-value	p-value
1	IorP	Cogni	1.992	0.048
2	IorP	DSQ	2.440	0.016
3	IorP & Cogni	DSQ	2.016	0.046

The results indicate that Cognitive Workload is a valid mediator between Individuals or Pairs and Design Solution Quality, as the p-values are each and all significant.

Changing to the third and fourth test groupings, the Mind Maps or no Mind Maps factor set the third grouping of Mind Maps or no Mind Maps (MM) → Communications → Design Solution Quality shown here in Table F-3:

Table F-3: Regressions for Communications as a mediator between the Mind Maps or no Mind Maps factor and the Design Solution Quality dependent variable

Step	Independent Variable	Dependent Variable	t-value	p-value
1	MM	Comms	2.638	0.009
2	MM	DSQ	3.013	0.003
3	MM & Comms	DSQ	2.881	0.005

The results indicate that Communications is a valid mediator between Mind Maps or no Mind Maps and Design Solution Quality, as the p-values are each and all significant.

The final test grouping for Design Solution quality is set as the fourth test grouping of Mind Maps or no Mind Maps → Cognitive Workload → Design Solution Quality show here in Table F-4:

Table F-4: Regressions for Cognitive Workload as a mediator between the Mind Maps or no Mind Maps factor and the Design Solution Quality dependent variable

Step	Independent Variable	Dependent Variable	t-value	p-value
1	MM	Cogni	2.363	0.020
2	MM	DSQ	2.264	0.025
3	MM & Cogni	DSQ	2.003	0.047

The results indicate that Cognitive Workload is a valid mediator between Mind Maps or no Mind Maps and Design Solution Quality, as the p-values are each and all significant.

The remaining four test groupings are focused on Job Performance Satisfaction (JPS). The factor of Individuals or Pairs is the basis of the three following regressions of Individuals or Pairs (IorP) → Communications (Comms) → Job Performance Satisfaction (JPS) shown here below in Table F-5:

Table F-5: Regressions for Communications as a mediator between the Individuals or Pairs factor and the Job Performance Satisfaction dependent variable

Step	Independent Variable	Dependent Variable	t-value	p-value
1	IorP	Comms	2.119	0.036
2	IorP	JPS	3.367	0.001
3	IorP & Comms	JPS	2.653	0.009

The results indicate that Communications is a valid mediator between Individuals or Pairs and Job Performance Satisfaction as the p-values are each and all significant.

Continuing with the second set of test groupings based on Job Performance Satisfaction, the Cognitive Workload mediator is included as Individuals or Pairs → Cognitive Workload (Cogni) → Job Performance Satisfaction shown below in Table F-6:

Table F-6: Regressions for Cognitive Workload as a mediator between the Individuals or Pairs factor and the Job Performance Satisfaction dependent variable

Step	Independent Variable	Dependent Variable	t-value	p-value
1	IorP	Cogni	-2.109	0.036
2	IorP	JPS	-2.833	0.004
3	IorP & Cogni	JPS	-2.254	0.026

The result of the test grouping Individuals or Pairs → Cognitive Workload → Job Performance Satisfaction is the inverse of expectation based on literature review and contradicts Hypothesis 4, which states "Average level of task satisfaction will be higher among collaborating pairs compared to average task satisfaction level of nominal pairs." The extant literature review concerning job task satisfaction for small groups and dyads generally found increases in communications and coordination which supported perceived decreased cognitive workload (Williams

et al., 2000; Williams and Kessler, 2001). However, the literature review also mentioned the gains being overwhelmed by losses leading to lower job task satisfaction (Balijepally, 2006; Brodbeck & Greitemeyer, 2000; Hill, 1982; Propp, 2003). Trust within the collaborating pair increases over time (Aiken, 2004) and this cycle might not have been long enough for the pair to jell (Mangalaraj, 2006).

This finding also affects Hypothesis 8, which states "Decreased cognitive loading in software developers in a design task will bring higher job satisfaction." Although decreased cognitive loading was supported in nominal pairs using Mind Maps (not tested here) it was overwhelmed by the increased cognitive loading reported by collaborating pairs, with or without Mind Maps.

Simple correlation discovered early in the analyses between Design Solution Quality and Job Performance Satisfaction was negative, although small, and that forewarning seems to bear out here in this regression. Review of the findings indicates that individuals had higher Job Performance Satisfaction than collaborating pairs, thus lending inverse support for Hypothesis 4 and removing any support for Hypothesis 8.

The remaining two mediation test groupings focused on Job Performance Satisfaction (JPS) and used the Mind Maps or no Mind Maps factor. The test grouping here is the seventh of eight totals and the third of the Job Performance Satisfaction mediation tests. Shown here below in Table F-7 is the result of the

Mind Maps or no Mind Maps (MM) → Communications (Comms) → Job Performance Satisfaction regressions:

Table F-7: Regressions for Communications as a mediator between the Mind Maps or no Mind Maps and the Job Performance Satisfaction dependent variable

Step	Independent Variable	Dependent Variable	t-value	p-value
1	MM	Comms	2.775	0.006
2	MM	JPS	2.351	0.020
3	MM & Comms	JPS	2.668	0.009

The results indicate that Communications is a valid mediator between Individuals or Pairs and Job Performance Satisfaction as the p-values are each and all significant.

The final of the regression test grouping used Mind Maps or no Mind Maps → Cognitive Workload (Cogni) → Job Performance Satisfaction and the results of the test are shown here below in Table F-8:

Table F-8: Regressions for Cognitive Workload as a mediator between the Mind Maps or no Mind Maps and the Job Performance Satisfaction dependent variable

Step	Independent Variable	Dependent Variable	t-value	p-value
1	MM	Cogni	3.023	0.003
2	MM	JPS	2.731	0.007
3	MM & Cogni	JPS	2.911	0.004

The results indicate that Cognitive Workload is a valid mediator between Mind Maps or no Mind Maps and Job Performance Satisfaction as the p-values are each and all significant.

Of the eight total test groupings required to test the possible mediators of Communications in the Development Process and Cognitive Workload in the Development Process, only Cognitive Workload failed and that was only on the Individuals or Pairs factor with Job Performance Satisfaction as the dependent variable. Therefore, even this wounded mediator worked for Mind Maps or no Mind Maps. Although Hypothesis 4 was not supported and was even inversely supported and Hypothesis 8 also found no support, this research study has successfully introduced a new mediator, Communications in the Development Process, into the meager pair software design literature.

References

Abdullah, N., & Honiden, S. (2011). Communication patterns of agile requirements engineering. *… on Agile Requirements …*, 1. Retrieved from http://dl.acm.org/citation.cfm?id=2068784

Abrahamsson, P., & Warsta, J. (2003). New directions on agile methods: a comparative analysis. *Software …*, 244–254. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1201204

Adamczyk, P. D., Hamilton, K., Twidale, M. B., & Bailey, B. P. (2007). Tools in support of creative collaboration. *Proceedings of the 6th ACM SIGCHI Conference on Creativity & Cognition - C&C '07*, 273. doi:10.1145/1254960.1255010

Aiken, J. (2004). Technical and human perspectives on pair programming. *ACM SIGSOFT Software Engineering Notes*, *29*(5), 1–14. Retrieved from http://dl.acm.org/citation.cfm?id=1022512

Aladwani, A. (2002). An integrated performance model of information systems projects. *Journal of Management Information Systems*, *19*(1), 185–210. Retrieved from http://mesharpe.metapress.com/index/793b5gle357aveqm.pdf

Andres, H. P. (2013). Team cognition using collaborative technology: a behavioral analysis. *Journal of Managerial Psychology*, *28*(1), 38–54. doi:10.1108/02683941311298850

Aoyama, M. (1998). Agile Software Process and its experience. *Proceedings of the 20th International Conference on Software Engineering*, 3–12. doi:10.1109/ICSE.1998.671097

Aranda, J., & Easterbrook, S. (2006). Distributed cognition in software engineering research: Can it be made to work? *… the Social Side of Large Scale Software …*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.503&rep=rep1&type=pdf#page=35

Arisholm, E., & Gallis, H. (2007). Evaluating pair programming with respect to system complexity and programmer expertise. *… , IEEE Transactions on*,

33(2), 65–86. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4052584

Balijepally, V. (2006). *Task complexity and effectiveness of pair programming: an experimental study*. University of Texas at Arlington. Retrieved from https://dspace.uta.edu/handle/10106/242

Balijepally, V., Mahapatra, R., Nerur, S., & Price, K. H. (2009). Are Two Heads Better than One for Software Development? The Productivity Paradox of Pair Programming. *MIS Quarterly*, *33*(1), 91–118. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=02767783&AN=36525659&h=i2CSehzpTLxg4GAdYPl4ak2r1M7sOBtdQvcy7SoOTkWPhxwF5HhKGPKM74WTS3NHgKWPE6rmSjSOlabXJ0gaYg==&crl=c

Banks, A., & Millward, L. (2000). Running shared mental models as a distributed cognitive process. *British Journal of Psychology*, *91*, 513–531. Retrieved from http://onlinelibrary.wiley.com/doi/10.1348/000712600161961/abstract

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

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Agile Manifesto. *Software Development*. [San Francisco, CA: Miller Freeman, Inc., 1993-. Retrieved from http://agilemanifesto.org/

Beel, J., Gipp, B., & Stiller, J. (2009). Information retrieval on mind maps-what could it be good for? *IEEE CollaborateCom'09*, (November), 1–4. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5364172

Begel, A., & Nagappan, N. (2008). Pair Programming: What's in it for Me? In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '08* (p. 120). doi:10.1145/1414004.1414026

Bertram, D., & Voida, A. (2010). Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams. *Proceedings of the 2010 …*. Retrieved from http://dl.acm.org/citation.cfm?id=1718972

Biehl, J., & Czerwinski, M. (2007). FASTDash: a visual dashboard for fostering awareness in software teams. *Proceedings of the …*, 1313–1322. Retrieved from http://dl.acm.org/citation.cfm?id=1240823

Bilda, Z., & Gero, J. (2007). The impact of working memory limitations on the design process during conceptualization. *Design Studies*, 1–28. Retrieved from http://www.sciencedirect.com/science/article/pii/S0142694X07000245

Björklund, T. a. (2013). Initial mental representations of design problems: Differences between experts and novices. *Design Studies*, *34*(2), 135–160. doi:10.1016/j.destud.2012.08.005

Bossche, P., Gijselaers, W., Segers, M., Woltjer, G., & Kirschner, P. (2010). Team learning: building shared mental models. *Instructional Science*, *39*(3), 283–301. doi:10.1007/s11251-010-9128-3

Brodbeck, F., & Greitemeyer, T. (2000). A Dynamic Model of Group Performance: Considering the Group Members' Capacity to Learn. *Group Processes & Intergroup Relations*, *3*(2), 159–182. doi:10.1177/1368430200003002004

Browaeys, M.-J., & Fisser, S. (2012). Lean and agile: an epistemological reflection. *The Learning Organization*, *19*(3), 207–218. doi:10.1108/09696471211219903

Brown, K., & Hyer, N. L. (2002). Whole-brain thinking for project management. *Business Horizons*, 47–57. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Whole-brain+thinking+for+project+management#0

Buckingham, C. D., Ahmed, a, & Adams, a E. (2007). Using XML and XSLT for flexible elicitation of mental-health risk knowledge. *Medical Informatics and the Internet in Medicine*, *32*(1), 65–81. doi:10.1080/14639230601097895

Budd, J. W. (2004). Mind Maps as Classroom Exercises. *Journal of Economic Education*, (Winter), 35–46.

Burgess-Allen, J., & Owen-Smith, V. (2010). Using mind mapping techniques for rapid qualitative data analysis in public participation processes. *Health Expectations : An International Journal of Public Participation in Health*

Care and Health Policy, *13*(4), 406–15. doi:10.1111/j.1369-7625.2010.00594.x

Burton-Jones, A., & Meso, P. (2008). The effects of decomposition quality and multiple forms of information on novices' understanding of a domain from a conceptual model. *Journal of the Association for Information …*, *9*(12), 748–802. Retrieved from http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1487&context=jais

Canfora, G., Cimitile, A., Garcia, F., Piattini, M., & Visaggio, C. A. (2007). Evaluating performances of pair designing in industry. *Journal of Systems and Software*, *80*(8), 1317–1327. doi:10.1016/j.jss.2006.11.004

Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *Software, IEEE*, *25*(1), 60–67. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4420071

Cao, L., Ramesh, B., & Abdel-Hamid, T. (2010). Modeling dynamics in agile software development. *ACM Transactions on Management Information Systems (TMIS)*, *1*(1), 5. Retrieved from http://dl.acm.org/citation.cfm?id=1877730

Carley, K. M. (1997). Extracting team mental models through textual analysis. *Journal of Organizational Behavior*, *18*(S1), 533–558. doi:10.1002/(SICI)1099-1379(199711)18:1+<533::AID-JOB906>3.3.CO;2-V

Cockburn, A. (1999). Characterizing people as non-linear, first-order components in software development. *International Conference on Software Engineering …*. Retrieved from http://alistair.cockburn.us/Characterizing+people+as+non-linear,+first-order+components+in+software+development/v/slim

Cockburn, A. (2002). Agile Software Development Joins the" Would-Be" Crowd. *Cutter IT Journal*, (January), 6–12. Retrieved from http://cf.agilealliance.org/articles/system/article/file/782/file.pdf

Cockburn, A., & Highsmith, J. (2001). Agile software development: The business of innovation. *Computer*, *34*(11), 131–133. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=963450

Cockburn, A., & Williams, L. (2000). The costs and benefits of pair programming. *Extreme Programming Examined*, 1–11. Retrieved from http://www.dsc.ufcg.edu.br/~jacques/cursos/map/recursos/XPSardinia.pdf

Cohen, J. (1992). A power primer. *Quantitative Methods in Psychology*, *112*(1), 155–159. doi:10.1038/141613a0

Corrie, B. (2010). *Human communication channels in distributed, artifact-centric, scientific collaboration*. Retrieved from http://dl.acm.org/citation.cfm?id=2231778

Cox, R., & Brna, P. (1994). *Supporting the use of external representations in problem solving: The need for flexible learning environments*. *Journal of Artificial Intelligence in Education*. Lancaster, England. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.57.5837&rep=rep1&type=pdf

Cross, N. (2001). Design cognition: Results from protocol and other empirical studies of design activity. *Design Knowing and Learning: Cognition in Design …*, 1–20. Retrieved from http://www.cc.gatech.edu/classes/AY2013/cs7601_spring/papers/Cross-DesignCognition.pdf

Čubranic, D., Storey, M., & Ryall, J. (2006). A comparison of communication technologies to support novice team programming. *Proceedings of the 28th International …*, 695–698. Retrieved from http://dl.acm.org/citation.cfm?id=1134394

Daft, R., & Lengel, R. (1986). Organizational information requirements, media richness and structural design. *Management Science*, *32*(5). Retrieved from http://pubsonline.informs.org/doi/abs/10.1287/mnsc.32.5.554

Davern, M., Shaft, T., & Te'eni, D. (2012). Cognition Matters: Enduring Questions in Cognitive IS Research. *Journal of the Association for …*, *13*(April 2011), 273–314. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&profile=ehost&scope=site&authtype=crawler&jrnl=15369323&AN=77807708&h=+2EQXlqvoIQX+eMRoDXeOY/6KUnaqVwzewKCLiQ6MNMm2OSAI4DUoUk6+djsKOSo9+2n/pWgnWpNGHHjD+9w7A==&crl=c

De la Rocha, O. L. (1986). *Problems of sense and problems of scale: An ethnographic study of arithmetic in everyday life*. University of California, Irvine. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Problems+of+Sense+and+Problems+of+Scale:+An+Ethnographic+Study+of+Arithmetic+in+Everyday+Life#0

De Vries, R. E., van den Hooff, B., & de Ridder, J. A. (2006). Explaining Knowledge Sharing: The Role of Team Communication Styles, Job Satisfaction, and Performance Beliefs. *Communication Research*, *33*(2), 115–135. doi:10.1177/0093650205285366

DeChurch, L. a, & Mesmer-Magnus, J. R. (2010). The cognitive underpinnings of effective teamwork: a meta-analysis. *The Journal of Applied Psychology*, *95*(1), 32–53. doi:10.1037/a0017328

Dekel, U., & Herbsleb, J. (2007). Notation and representation in collaborative object-oriented design: an observational study. In *ACM OOPSLA* (pp. 261–280). Montreal, Quebec, Canada: ACM Press. Retrieved from http://dl.acm.org/citation.cfm?id=1297047

Dennis, A., Fuller, R., & Valacich, J. (2008). Media, tasks, and communication processes: A theory of media synchronicity. *MIS Quarterly*, *32*(3), 575–600. Retrieved from http://dl.acm.org/citation.cfm?id=2017395

Dix, A. J. (1994). Computer-supported cooperative work - a framework. In D. Rosenburg & C. Hutchison (Eds.), *Design Issues in CSCW* (pp. 23–37). London: Springer-Verlag London.

Dong, A., Kleinsmann, M. S., & Deken, F. (2013). Investigating design cognition in the construction and enactment of team mental models. *Design Studies*, *34*(1), 1–33. doi:10.1016/j.destud.2012.05.003

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, *50*(9-10), 833–859. doi:10.1016/j.infsof.2008.01.006

Dybå, T., Sjøberg, D., & Cruzes, D. (2012). What works for whom, where, when, and why? On the role of context in empirical software engineering. *... on Empirical Software Engineering ...*, (7465), 19–28. Retrieved from http://dl.acm.org/citation.cfm?id=2372256

Eden, C. (2004). Analyzing cognitive maps to help structure issues or problems. *European Journal of Operational Research*, *159*(3), 673–686. doi:10.1016/S0377-2217(03)00431-4

Ehrlich, K., & Cataldo, M. (2012). All-for-One and One-for-All? A Multi-Level Analysis of Communication Networks and Individual Performance in Geographically Distributed Software Development. In *ACM CSCW 2012* (pp. 945–954). Seattle. Retrieved from http://dl.acm.org/citation.cfm?id=2145345

Eppler, M. (2001). Making knowledge visible through intranet knowledge maps: concepts, elements, cases. *System Sciences, 2001. Proceedings of the 34th …*, *00*(c), 1–10. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=926495

Eppler, M. J. (2006). A comparison between concept maps, mind maps, conceptual diagrams, and visual metaphors as complementary tools for knowledge construction and sharing. *Information Visualization*, *5*(3), 202–210. doi:10.1057/palgrave.ivs.9500131

Espinosa, J., Kraut, R., & Lerch, F. (2001). Shared Mental Models and Coordination in Large-Scale, Distributed Software Development. *ICIS*, 513–518. Retrieved from https://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/jdh/collaboratory/research_papers/ICIS_2001.pdf

Espinosa, J., Slaughter, S., Kraut, R., & Herbsleb, J. (2007). Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems*, *24*(1), 135–169. doi:10.2753/MIS0742-1222240104

Fischer, G., & Ostwald, J. (2003). Knowledge communication in design communities. In R. Bromme, H. Friedrich, & H. Spada (Eds.), *Barriers and Biases in computer-mediated knowledge communication* (pp. 1–27). Netherlands: Kluwer Academic Publishers. Retrieved from http://link.springer.com/chapter/10.1007/0-387-24319-4_10

Flor, N., & Hutchins, E. (1991). Analyzing Distributed Cognition in Software Teams: A Case Study of Team Programming During Perfective Software Maintenance. *… Studies of Programmers: Fourth Workshop, Norwood, …*, 36–64. Retrieved from

http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Analyzing+Distributed+Cognition+in+Software+Teams:+A+Case+Study+of+Team+Progrmaming+During+Perfective+Software+Maintenance#0

Fong Boh, W., Slaughter, S. A., & Espinosa, J. a. (2007). Learning from Experience in Software Development: A Multilevel Analysis. *Management Science*, *53*(8), 1315–1331. doi:10.1287/mnsc.1060.0687

Froehlich, J., & Dourish, P. (2004). Unifying artifacts and activities in a visual tool for distributed software development teams. *... of the 26th International Conference on Software ...*. Retrieved from http://dl.acm.org/citation.cfm?id=999443

Fussell, S., Kraut, R., & Siegel, J. (2000). Coordination of communication: Effects of shared visual context on collaborative work. *... Computer Supported Cooperative Work*. Retrieved from http://dl.acm.org/citation.cfm?id=358947

Gallupe, R., DeSanctis, G., & Dickson, G. (1988). Computer-based support for group problem-finding: An experimental investigation. *MIS Quarterly*, (June), 277–297. Retrieved from http://www.jstor.org/stable/10.2307/248853

Gasson, S. (1999). A social action model of situated information systems design. *ACM SIGMIS Database*, *30*(2), 82–97. doi:10.1145/383371.383377

Gasson, S. (2004). A framework for behavioral studies of social cognition in information systems. *Proceedings of the ISOneWorld Conference, Las Vegas, April 2004*, (April). Retrieved from http://idea.library.drexel.edu/handle/1860/1988

Gasson, S. (2005). Resolving wicked problems: collaborative information systems design in boundary-spanning groups. *American Conference on Information Systems - Omaha*. Retrieved from http://idea.library.drexel.edu/handle/1860/2004

Giri, M., & Soni, S. (2013). Effectiveness of Software Development Process Using Programmer Ranker Algorithm in Pair Programming. *International Journal of Engineering Sciences and Research Technology*, *2*(6), 1524–1535. Retrieved from http://www.ijesrt.com/issues pdf file/Archives 2013/june-2013/18.pdf

Goldman, M., & Miller, R. C. (2010). Test-driven roles for pair programming. *2010 ACM/IEEE 32nd International Conference on Software Engineering*, *2*, 515–516. doi:10.1145/1810295.1810458

Gorla, N., & Lam, Y. W. Y. (2004). Who should work with whom?: building effective software project teams. *Communications of the ACM*, *47*(6), 79–82. Retrieved from http://dl.acm.org/citation.cfm?id=990684

Greenberg, J., & Dickelman, G. (2000). Distributed cognition: a foundation for performance support. *Performance Improvement*, (July), 18–24. Retrieved from http://onlinelibrary.wiley.com/doi/10.1002/pfi.4140390608/abstract

Group, S. (2013). *CHAOS Manifesto 2013*.

Guindon, R. (1990). Designing the design process. Exploiting opportunistic thoughts. *HumanComputer Interaction*, *5*, 305–344.

Guindon, R., & Curtis, B. (1988). Control of cognitive processes during software design: what tools are needed? In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88* (pp. 263–268). New York, New York, USA: ACM Press. doi:10.1145/57167.57211

Gutwin, C., & Greenberg, S. (2001). *The importance of awareness for team cognition in distributed collaboration* (No. 2001-696-19). *Team cognition: Understanding the factors that …*. Calgary, Alberta, Canada. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.6293&rep=rep1&type=pdf

Halverson, C. (1994). Distributed Cognition as a theoretical framework for HCI: Don't throw the Baby out with the bathwater-the importance of the cursor in Air Traffic Control. *Cognitive Science Department, UCSD Report*, 0–22. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.6736&rep=rep1&type=pdf

Hanks, B. (2008). Empirical evaluation of distributed pair programming. *International Journal of Human Computer Studies*, *66*, 530–544. doi:10.1016/j.ijhcs.2007.10.003

Hansen, S., Kharabe, A., & Lyytinen, K. (2013). The Structures of Computation: A Distributed Cognitive Analysis of Requirements Evolution in Diverse

Software Development Environments. In *8th Pre-ICIS International …* (pp. 111–124). Retrieved from http://start.aisnet.org/resource/group/b11928ad-01b7-4939-bfe9-0750223c2f4e/irwitpm2013workshopproceedin.pdf#page=111

Hansen, S., & Lyytinen, K. (2009). Distributed Cognition in the Management of Design Requirements. In M. Jarke, K. Lyytinen, & J. Mylopoulos (Eds.), *Perspectives Workshop: Science of Design: High-Impact Requirements for Software-Intensive Systems* (pp. 1–10). Dagstuhl, Germany. Retrieved from http://drops.dagstuhl.de/opus/volltexte/2009/1930/

Hanssen, G. K. (2011). A longitudinal case study of an emerging software ecosystem: Implications for practice and theory. *Journal of Systems and Software*, *85*(7), 1455–1466. doi:10.1016/j.jss.2011.04.020

Hao, J.-X., Kwok, R. C.-W., Lau, R. Y.-K., & Yu, A. Y. (2010). Predicting problem-solving performance with concept maps: An information-theoretic approach. *Decision Support Systems*, *48*(4), 613–621. doi:10.1016/j.dss.2009.12.001

Hatfield, J. D., & Huseman, R. C. (1982). Perceptual Congruence About Communication as Related to Satisfaction: Moderating Effects of Individual Characteristics. *Academy of Management Journal*, *25*(2), 349–358. doi:10.2307/255996

Herbsleb, J. D., & Mockus, A. (2003). Formulation and preliminary test of an empirical theory of coordination in software engineering. *ACM SIGSOFT Software Engineering Notes*, *28*(5), 138. doi:10.1145/949952.940091

Hick, W. (1952). On the rate of gain of information. *Quarterly Journal of Experimental Psychology*, *4*(1), 11–26. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/17470215208416600

Hill, G. (1982). Group versus individual performance: Are N+ 1 heads better than one? *Psychological Bulletin*, *91*(3), 517–539. Retrieved from http://psycnet.apa.org/journals/bul/91/3/517/

Hinsz, V. B., Tindale, R. S., & Vollrath, D. a. (1997). The emerging conceptualization of groups as information processors. *Psychological Bulletin*, *121*(1), 43–64. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/9000891

Hirst, G., & Mann, L. (2004). A model of R&D leadership and team communication: the relationship with project performance. *R and D Management*, *34*(2), 147–160. doi:10.1111/j.1467-9310.2004.00330.x

Hoda, R., Kruchten, P., Noble, J., & Marshall, S. (2010). Agility in context. *ACM Sigplan Notices*, *45*, 74–88. Retrieved from http://dl.acm.org/citation.cfm?id=1869467

Hoda, R., Noble, J., & Marshall, S. (2010). Balancing acts: walking the Agile tightrope. *Proceedings of the 2010 ICSE Workshop …*, 5–12. Retrieved from http://dl.acm.org/citation.cfm?id=1833312

Hoegl, M., & Gemuenden, H. G. (2001). Teamwork Quality and the Success of Innovative Projects: A Theoretical Concept and Empirical Evidence. *Organization Science*, *12*(4), 435–449. doi:10.1287/orsc.12.4.435.10635

Hoegl, M., Praveen Parboteeah, K., & Gemuenden, H. G. (2003). 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), 281–302. doi:10.1016/j.jengtecman.2003.08.001

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction*, *7*(2), 174–196. doi:10.1145/353485.353487

Horsky, J., Kaufman, D. R., Oppenheim, M. I., & Patel, V. L. (2003). A framework for analyzing the cognitive complexity of computer-assisted clinical ordering. *Journal of Biomedical Informatics*, *36*(1-2), 4–22. doi:10.1016/S1532-0464(03)00062-5

Huba, G. J. (2013). Mind Maps, Concept Maps and SEM. *hubaisms.com*. Retrieved from http://hubaisms.com/2013/06/01/structural-equation-statistical-models-mind-maps-concept-maps/

Hulkko, H., & Abrahamsson, P. (2005). A multiple case study on the impact of pair programming on product quality. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.* (pp. 495–504). IEEe. doi:10.1109/ICSE.2005.1553595

Hutchins, E. (2000). Distributed cognition. *Internacional Enciclopedia of the Social and …*, *34*(6), 726–735. Retrieved from http://www.artmap-research.com/wp-content/uploads/2009/11/Hutchins_DistributedCognition.pdf

Iiskala, T., Vauras, M., Lehtinen, E., & Salonen, P. (2011). Socially shared metacognition of dyads of pupils in collaborative mathematical problem-solving processes. *Learning and Instruction*, *21*(3), 379–393. doi:10.1016/j.learninstruc.2010.05.002

Kaptelinin, V. (1996). Distribution of cognition between minds and artifacts: Augmentation of mediation? *AI & Society*, *10*(1), 15–25. doi:10.1007/BF02716751

Kaptelinin, V., & Nardi, B. (2006). Acting With Technology–Activity Theory and Interaction Design. *Amazon Media Sarl (Kindle Edition) p345*. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Acting+with+Technology+Activity+Theory+and+Interaction+Design#1

Kleinsmann, M., & Valkenburg, R. (2008). Barriers and enablers for creating shared understanding in co-design projects. *Design Studies*, *29*(4), 369–386. doi:10.1016/j.destud.2008.03.003

Kokotovich, V. (2008). Problem analysis and thinking tools: an empirical study of non-hierarchical mind mapping. *Design Studies*, *29*(1), 49–69. doi:10.1016/j.destud.2007.09.001

Kokotovich, V., & Purcell, T. (2000). Mental synthesis and creativity in design: an experimental examination. *Design Studies*, *21*, 437–449. Retrieved from http://www.sciencedirect.com/science/article/pii/S0142694X0000017X

Kozlowski, S. W., & Ilgen, D. R. (2006). Enhancing the effectiveness of work groups and teams. *Psychological Science in the Public Interest*, *7*(3), 77–124. Retrieved from http://psi.sagepub.com/content/7/3/77.short

Kraut, R., Fish, R., Root, R., & Chalfonte, B. (1990). Informal communication in organizations: Form, function, and technology. *… Reactions to Technology: …*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.9721&rep=rep1&type=pdf

Kravitz, D. a., & Martin, B. (1986). Ringelmann rediscovered: The original article. *Journal of Personality and Social Psychology*, *50*(5), 936–941. doi:10.1037//0022-3514.50.5.936

Kurtzberg, T. R. (2005). Feeling Creative, Being Creative: An Empirical Study of Diversity and Creativity in Teams. *Creativity Research Journal*, *17*(1), 51–65. doi:10.1207/s15326934crj1701_5

Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2005). *Applied Linear Statistical Models* (5th Editio.). New York, NY: MdGraw-Hill Irwin.

Laughlin, P. R., Bonner, B. L., & Miner, A. G. (2002). Groups perform better than the best individuals on Letters-to-Numbers problems. *Organizational Behavior and Human Decision Processes*, *88*(2), 605–620. doi:10.1016/S0749-5978(02)00003-1

Lavallée, M., & Robillard, P. (2012). The impacts of software process improvement on developers: a systematic review. *… 2012 International Conference on Software …*, 113–122. Retrieved from http://dl.acm.org/citation.cfm?id=2337237

Lee, C. (2005). Between chaos and routine: Boundary negotiating artifacts in collaboration. *ECSCW 2005*, (September), 387–406. Retrieved from http://link.springer.com/chapter/10.1007/1-4020-4023-7_20

Lee, G., & Xia, W. (2010). Toward agile: an integrated analysis of quantitative and qualitative field data on software development agility. *Mis Quarterly*, *34*(1), 87. Retrieved from http://www.cse.chalmers.se/~feldt/courses/agile/lee_2010_integrated_analysis_of_sw_dev_agility.pdf

Levesque, L. L., Wilson, J. M., & Wholey, D. R. (2001). Cognitive divergence and shared mental models in software development project teams. *Journal of Organizational Behavior*, *22*(2), 135–144. doi:10.1002/job.87

Lewis, A. C., Sadosky, T. L., & Connolly, T. (1975). The effectiveness of group brainstorming in engineering problem solving. *IEEE Transactions on Engineering Management*, *EM-22*(3), 119–124. doi:10.1109/TEM.1975.6447219

Lowry, P. B., Roberts, T. L., & Romano, N. C. (2013). What signal is your inspection team sending to each other? Using a shared collaborative interface to improve shared cognition and implicit coordination in error-detection teams. *International Journal of Human-Computer Studies*, *71*(4), 455–474. doi:10.1016/j.ijhcs.2012.11.004

Lui, K., & Chan, K. (2003). When does a pair outperform two individuals? *Extreme Programming and Agile Processes in …*. Retrieved from http://link.springer.com/chapter/10.1007/3-540-44870-5_28

Lui, K. M., & Chan, K. C. C. (2006). Pair programming productivity: Novice–novice vs. expert–expert. *International Journal of Human-Computer Studies*, *64*(9), 915–925. doi:10.1016/j.ijhcs.2006.04.010

Lui, K. M., Chan, K. C. C., & Nosek, J. T. . (2008). The Effect of Pairs in Program Design Tasks. *IEEE Transactions on Software Engineering*, *34*(2), 197–211. doi:10.1109/TSE.2007.70755

MacCormack, A., Rusnak, J., & Baldwin, C. (2008). *Exploring the duality between product and organizational architectures: A test of the mirroring hypothesis* (No. 08-039). Boston, Mass. Retrieved from http://673computationalcooking.googlecode.com/svn/trunk/628_SoftwareEngineering_Paper/papers/HBS_ExploringTheDualityBetweenProductAndOrganizational ArchitecturesATestOfTheMirroringHypothesis_2008.pdf

Maheshwari, M., Kumar, U., & Kumar, V. (2012). Alignment between social and technical capability in software development teams: An empirical study. *Team Performance Management*, *18*(1/2), 7–26. doi:10.1108/13527591211207680

Mangalaraj, G. (2006). *Influence of codified knowledge on software design task performance: a comparison of pairs with individuals*. University of Texas at Arlington. Retrieved from https://dspace.uta.edu/handle/10106/439

Mangalaraj, G., Nerur, S., Mahapatra, R., & Price, K. H. (2014). Distributed Cognition in Software Design: An Experimental Investigation of the Role of Design Patterns and Collaboration. *MIS Quarterly*, *38*(1), 249–274.

Mantel, Jr., S., Meredith, J., Shafer, S., & Sutton, M. (2011). *Project Management Practice* (Fourth.). Hoboken, New Jersey: John Wiley & Sons, Incorporated.

Melnik, G., & Maurer, F. (2004). Direct verbal communication as a catalyst of agile knowledge sharing. *Agile Development Conference, 2004*, 21–31. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1359792

Mento, A. J., Martinelli, P., & Jones, R. M. (1999). Mind mapping in executive education: applications and outcomes. *Journal of Management Development*, *18*(4), 390–416. doi:10.1108/02621719910265577

Meso, P., & Jain, R. (2006). Agile software development: adaptive systems principles and best practices. *Information Systems Management*. Retrieved from http://www.tandfonline.com/doi/pdf/10.1201/1078.10580530/46108.23.3.20060601/93704.3

Miller, G. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*, *63*(2), 81–97. Retrieved from http://psycnet.apa.org/journals/rev/63/2/81/

Misra, S. (2012). Agile software development practices: evolution, principles, and criticisms. *International Journal of Quality & Reliability Management*, *29*(9), 972–980. doi:10.1108/02656711211272863

Moe, N. N. B., Dingsoyr, T., & Dyba, T. (2009). Overcoming barriers to self-management in software teams. *Software, IEEE*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5287005

Monteiro, C. V. F., da Silva, F. Q. B., dos Santos, I. R. M., Farias, F., Cardozo, E. S. F., do A. Leitão, A. R. G., … Pernambuco Filho, M. J. a. (2011). A qualitative study of the determinants of self-managing team effectiveness in a scrum team. *Proceeding of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE '11*, 16. doi:10.1145/1984642.1984646

Morris, M. G., Speier, C., & Hoffer, J. a. (1996). The impact of experience on individual performance and workload differences using object-oriented and process-oriented systems analysis techniques. *Proceedings of HICSS-29: 29th Hawaii International Conference on System Sciences*, 232–241 vol.2. doi:10.1109/HICSS.1996.495404

Müller, M. M. (2006). A preliminary study on the impact of a pair design phase on pair programming and solo programming. *Information and Software Technology*, *48*(5), 335–344. doi:10.1016/j.infsof.2005.09.008

Nawrocki, J., & Jasiñski, M. (2002). Extreme programming modified: embrace requirements engineering practices. *Requirements …*, 303–310. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1048543

Nawrocki, J., & Wojciechowski, A. (2001). Experimental evaluation of pair programming. *European Software Control and Metrics (…*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.19.1689&rep=rep1&type=pdf

Nerur, S., & Balijepally, V. (2007). Theoretical reflections on agile development methodologies. *Communications of the ACM*, *50*(3), 79–83. Retrieved from http://dl.acm.org/citation.cfm?id=1226739

Nerur, S., Cannon, A., Ballijepally, V., & Bond, P. (2010). Towards an Understanding of the Conceptual Underpinnings of Agile Development Methodologies. In T. Dingsøyr, T. Dybå, & N. B. Moe (Eds.), *Agile Software Development* (pp. 15–29). Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-12575-1_2

Nobarany, S. (2012). *Annotations for Supporting Collaboration through Artifacts*. *arXiv preprint arXiv:1211.1634*. Retrieved from http://arxiv.org/abs/1211.1634

Nosek, J. T. (1998). The case for collaborative programming. *Communications of the ACM*, *41*(3), 105–108. doi:10.1145/272287.272333

Nunnally, J. C. (1978). *Psychometric Theory* (2nd Editio.). McGraw-Hill, New York.

Padberg, F., & Muller, M. M. (2003). Analyzing the cost and benefit of pair programming. *Proceedings. 5th International Workshop on Enterprise Networking and Computing in Healthcare Industry (IEEE Cat. No.03EX717)*. doi:10.1109/METRIC.2003.1232465

Paulus, P., & Yang, H. (2000). Idea generation in groups: A basis for creativity in organizations. *Organizational Behavior and Human Decision …*, *82*(1), 76–

87. Retrieved from http://www.sciencedirect.com/science/article/pii/S0749597800928887

Pedrycz, W., Russo, B., & Succi, G. (2011). A model of job satisfaction for collaborative development processes. *Journal of Systems and Software*, *84*(5), 739–752. doi:10.1016/j.jss.2010.12.018

Perry, M., & Macredie, R. (1999). Distributed cognition: investigating collaboration in large and highly complex organisational systems. *Brunel University Technical Paper, Available at: …*, 1–24. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.138.6853&rep=rep1&type=pdf

Petersen, K., & Wohlin, C. (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Software Engineering*, *15*(6), 654–693. doi:10.1007/s10664-010-9136-6

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, *13*(3), 303–337. doi:10.1007/s10664-008-9065-9

Pinto, M., & Pinto, J. (1990). Project team communication and cross-functional cooperation in new program development. *Journal of Product Innovation Management*. Retrieved from http://www.sciencedirect.com/science/article/pii/073767829090004X

Propp, K. M. (2003). In Search of the Assembly Bonus Effect. *Human Communication Research*, *29*(4), 600–606. doi:10.1111/j.1468-2958.2003.tb00858.x

Quinones, P., Fussell, S. R., Soibelman, L., & Akinci, B. (2009). Bridging the Gap : Discovering Mental Models in Globally Collaborative Contexts. In *ACM IWIC'09* (pp. 101–110). Palo Alto, CA.

Rajkomar, A., & Blandford, A. (2011). Distributed cognition for evaluating healthcare technology. *Proceedings of the 25th BCS Conference on …*, 341–350. Retrieved from http://dl.acm.org/citation.cfm?id=2305375

Reeves, B., & Shipman, F. (1992). Supporting communication between designers with artifact-centered evolving information spaces. *Proceedings of the 1992*

ACM Conference on …, (November). Retrieved from http://dl.acm.org/citation.cfm?id=143556

Richter, A. W., Hirst, G., van Knippenberg, D., & Baer, M. (2012). Creative self-efficacy and individual creativity in team contexts: cross-level interactions with team informational resources. *The Journal of Applied Psychology*, *97*(6), 1282–90. doi:10.1037/a0029359

Roberts, J. A., Hann, I.-H., & Slaughter, S. A. (2006). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science*, *52*(7), 984–999. doi:10.1287/mnsc.1060.0554

Robillard, P. N., d'Astous, P., Detienne, F., & Visser, W. (1998). Measuring cognitive activities in software engineering. *Proceedings of the 20th International Conference on Software Engineering*, 292–300. doi:10.1109/ICSE.1998.671342

Robinson, H., & Sharp, H. (2004). The characteristics of XP teams. *Extreme Programming and Agile Processes in …*, 139–147. Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-24853-8_16

Rogers, Y. (1997). A brief introduction to distributed cognition. *Retrieved July*, (August). Retrieved from http://www.id-book.com/downloads/chapter 8 dcog-brief-intro.pdf

Rogers, Y. (2006). Distributed Cognition and Communication. In *Encyclopedia of Language and Linguistics*. doi:10.1016/B0-08-044854-2/00862-2

Rogers, Y., & Ellis, J. (1994). Distributed cognition: an alternative framework for analysing and explaining collaborative working. *Journal of Information Technology*, *9*(2), 119–128. doi:10.1057/jit.1994.12

Rosenbaum, A. (2003). Chart the course of your negotiation. *Harvard Management Communication Letter, …*, 3–5. Retrieved from http://rowbokay.co.uk/Harvard3.pdf

Ryan, S., & O'Connor, R. (2012). *Social interaction, team tacit knowledge and transactive memory: empirical support for the agile approach*. Retrieved from http://ulir.ul.ie/handle/10344/2698

Salo, O., & Abrahamsson, P. (2006). An Iterative Improvement Process for Agile Software. *Software Process Improvement and Practice*, (Beck 1999). doi:10.1002/spip

Salomon, G. (1993). *Distributed cognitions: Psychological and educational considerations*. (G. Salomon, Ed.). Cambridge University Press. Retrieved from http://books.google.com/books?hl=en&lr=&id=m8Yna0cjxAgC&oi=fnd&pg=PR7&dq=Distributed+cognitions+-+Psychological+and+educational+considerations&ots=-sEz-MxYMp&sig=W1FpgLolfHHBQ9Z7j1CwffUUIx4

Scaife, M., & Rogers, Y. (1996). External cognition: how do graphical representations work? *International Journal of Human-Computer Studies*, *45*, 185–213. Retrieved from http://www.sciencedirect.com/science/article/pii/S1071581996900488

Schön, D. (1984). *The Reflective Practitioner: How Professionals Think In Action*. Basic Books.

Sedig, K., & Parsons, P. (2013). Interaction design for complex cognitive activities with visual representations: A pattern-based approach. *... Transactions on Human-Computer Interaction*, *5*(2), 84–133. Retrieved from http://aisel.aisnet.org/thci/vol5/iss2/1/

Sfetsos, P., Angelis, L., & Stamelos, I. (2006). Investigating the extreme programming system–An empirical study. *Empirical Software Engineering*, *11*(2), 269–301. doi:10.1007/s10664-006-6404-6

Shah, J. J., Vargas-Hernandez, N., Summers, J. D., & Kulkarni, S. (2001). Collaborative Sketching (C-Sketch) - An Idea Generation Technique for Engineering Design. *The Journal of Creative Behavior*, *35*(3), 168–198. doi:10.1002/j.2162-6057.2001.tb01045.x

Sharp, H., & Robinson, H. (2006). A distributed cognition account of mature XP teams. *Extreme Programming and Agile Processes in …*, 1–10. Retrieved from http://link.springer.com/chapter/10.1007/11774129_1

Sharp, H., Robinson, H., & Petre, M. (2009). The role of physical artefacts in agile software development: Two complementary perspectives. *Interacting with Computers*, *21*(1-2), 108–116. doi:10.1016/j.intcom.2008.10.006

Sharp, H., Robinson, H., Segal, J., & Furniss, D. (2006). The Role of Story Cards and the Wall in XP teams: A Distributed Cognition Perspective. *Agile 2006 (Agile'06)*, 65–75. doi:10.1109/AGILE.2006.56

Shaw, M., & Ashton, N. (1976). Do assembly bonus effects occur on disjunctive tasks? A test of Steiner's theory. *Bulletin of the Psychonomic Society*, *8*(6), 469–471. Retrieved from http://link.springer.com/article/10.3758/BF03335201

Shirouzu, H., Miyake, N., & Masukawa, H. (2002). Cognitively active externalization for situated reflection. *Cognitive Science*, *26*(January 2001), 469–501. Retrieved from http://www.sciencedirect.com/science/article/pii/S0364021302000666

Small, R., & Venkatesh, M. (2000). A cognitive-motivational model of decision satisfaction. *Instructional Science*, *28*, 1–22. Retrieved from http://link.springer.com/article/10.1023/A:1003574312599

Sonnenwald, D. (1996). Communication roles that support collaboration during the design process. *Design Studies*, *17*, 277–301. Retrieved from http://www.sciencedirect.com/science/article/pii/0142694X96000026

Sonnenwald, D. (2001). Using innovation diffusion theory to guide collaboration technology evaluation: Work in progress. *... Technologies: ...*, 114–119. doi:10.1109/ENABL.2001.953399

Sonnenwald, D. H., & Iivonen, M. (1999). Research Framework for Information Studies. *Library & Information Science Research*, *21*(4), 429–457.

Speier, C., & Morris, M. (2003). The influence of query interface design on decision-making performance. *Mis Quarterly*, *27*(3), 397–423. Retrieved from http://dl.acm.org/citation.cfm?id=2017201

Stanton, N. a, Stewart, R., Harris, D., Houghton, R. J., Baber, C., McMaster, R., … Green, D. (2006). Distributed situation awareness in dynamic systems: theoretical development and application of an ergonomics methodology. *Ergonomics*, *49*(12-13), 1288–311. doi:10.1080/00140130600612762

Steiner, I. D. (1972). *Group Processes and Productivity*. New York, New York, USA: Academic Press.

Storey, M., Čubranić, D., & German, D. (2005). On the use of visualization to support awareness of human activities in software development: a survey and a framework. *... on Software Visualization*, *1*(212), 193–203. Retrieved from http://dl.acm.org/citation.cfm?id=1056045

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*(2), 257–285. doi:10.1016/0364-0213(88)90023-7

Syed-Abdullah, S., Holcombe, M., & Gheorge, M. (2006). The Impact of an Agile Methodology on the Well Being of Development Teams. *Empirical Software Engineering*, *11*(1), 143–167. doi:10.1007/s10664-006-5968-5

Ülengin, F., Kabak, Ö., Önsel, Ş., Ülengin, B., & Aktaş, E. (2010). A problem-structuring model for analyzing transportation–environment relationships. *European Journal of Operational Research*, *200*(3), 844–859. doi:10.1016/j.ejor.2009.01.023

Van der Lugt, R. (2002). Brainsketching and How it Differs from Brainstorming. *Creativity and Innovation Management*, *11*(1), 43–54. doi:10.1111/1467-8691.00235

VersionOne.com. (2013). 7th Annual State of Agile Development Survey.

Vidgen, R., & Wang, X. (2009). Coevolving Systems and the Organization of Agile Software Development. *Information Systems Research*, *20*(3), 355–376. doi:10.1287/isre.1090.0237

Visschers-Pleijers, A. J. S. F., Dolmans, D. H. J. M., Wolfhagen, I. H. a P., & van der Vleuten, C. P. M. (2005). Development and validation of a questionnaire to identify learning-oriented group interactions in PBL. *Medical Teacher*, *27*(4), 375–381. doi:10.1080/01421590500046395

Whitworth, E., & Biddle, R. (2007). The Social Nature of Agile Teams. *Agile 2007 (Agile 2007)*, 26–36. doi:10.1109/AGILE.2007.60

Williams, L. A. (2000). *The Collaborative Software Process*. University of Utah.

Williams, L., & Kessler, R. (2001). Experimenting with Industry's "Pair-Programming" Model in the Computer Science Classroom. *Journal on Software Engineering Education*. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Experime

nting+with+Industry's+"Pair-Programming"+Model+in+the+Computer+Science+Classroom#1

Williams, L., Kessler, R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software*. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=854064

Wirth, N. (1971). Program development by stepwise refinement. *Communications of the ACM*, *14*(4), 221–227. doi:10.1145/362575.362577

Woods, D. (1998). Commentary Designs are hypotheses about how artifacts shape cognition and collaboration. *Ergonomics*, *41*, 168–173. Retrieved from http://www.tandfonline.com/doi/pdf/10.1080/001401398187215

Yang, M. C. (2008). Observations on concept generation and sketching in engineering design. *Research in Engineering Design*, *20*(1), 1–11. doi:10.1007/s00163-008-0055-0

Yetton, P. W., & Bottger, P. C. (1982). Individual versus group problem solving: An empirical test of a best-member strategy. *Organizational Behavior and Human Performance*, *29*(3), 307–321. doi:10.1016/0030-5073(82)90248-3

Yuan, Y. C., Fulk, J., Monge, P. R., & Contractor, N. (2009). Expertise Directory Development, Shared Task Interdependence, and Strength of Communication Network Ties as Multilevel Predictors of Expertise Exchange in Transactive Memory Work Groups. *Communication Research*, *37*(1), 20–47. doi:10.1177/0093650209351469

Zampetakis, L. a., Tsironis, L., & Moustakis, V. (2007). Creativity development in engineering education: the case of mind mapping. *Journal of Management Development*, *26*(4), 370–380. doi:10.1108/02621710710740110

Zhang, J. (1991). The interaction of internal and external representations in a problem solving task. *Proceedings of the Thirteenth Annual Conference of …*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.6277&rep=rep1&type=pdf

Zhang, J. (1997). The nature of external representations in problem solving. *Cognitive Science*, *21*(2), 179–217. Retrieved from http://onlinelibrary.wiley.com/doi/10.1207/s15516709cog2102_3/abstract

Zhang, J., & Norman, D. A. (1994). Representations in Distributed Cognitive Tasks. *Cognitive Science*, *18*, 87–122. doi:10.1207/s15516709cog1801_3

Biographical Information

Philip L. Bond holds a BA degree and an MA degree in Economics from the University of Rhode Island and an MS degree in Information Systems from the University of Texas at Arlington. He has experience in Information Systems from Desktop Support to Consultant which includes a focus in networking. He has published research in conference proceedings and has reviewed research articles for Information Systems journals and conference proceedings.

With his teaching and research, he hopes to have impact on both the academic and practitioner community.