December 2003

Are Two Heads Always Better Than One in Collaborative Programming?

Madeline Domino
University of South Florida

Rosann Collins
University of South Florida

Keith Garrison
University of South Florida

ARE TWO HEADS ALWAYS BETTER THAN ONE IN COLLABORATIVE PROGRAMMING?

Madeline Ann Domino
University of South Florida
mdomino@coba.usf.edu

Rosann Webb Collins
University of South Florida
rcollins@coba.usf.edu

Keith Garrison
University of South Florida
kgarrison@coba.usf.edu

Abstract

The software industry continues to struggle with producing quality software in the most efficient manner. Anecdotal evidence suggests that the use of newer, innovative development methods, which embrace high levels of collaboration, may be a viable solution to this problem. While collaboration has always been used, these techniques emphasize high levels of interpersonal collaboration during the entire development process. An example of collaborative programming that is gaining interest is pair programming. A relatively unexplored element of collaborative programming is how the diversity of the individuals paired in the programming dyad impacts performance outcomes. We propose an experiment with the primary emphasis on how the diversity of individual developer characteristics impacts collaborative programming results, with a focus on creating the "best" pairing possible in order to raise performance and maximize outcomes.

Keywords: Diversity, agile software development, collaboration, small teams

Overview

A recent U. S. Department of Commerce study concluded that software bugs, or errors, cost the U. S. economy an estimated $59.5 billion dollars annually. Although not all software errors are likely to be removed (Glass 2003), more than a third of these costs could be eliminated by an improved testing infrastructure that enables earlier and more effective identification of software defects (Trembly 2002). It is widely recognized that the early detection of software errors in development enhances quality (McConnell 1996).

The software industry continues to struggle with producing quality software efficiently in shorter periods of time. Many of the development methodologies in use today are derived from practices relevant to very different organizational and business environments. Accordingly, there is a need to reconsider newer development practices (Fitzgerald 1997).

Anecdotally it is suggested that the newer, collaborative development methods produce better quality software in reduced time with higher levels of developer satisfaction (Beck 2000; Cockburn 2000). The limited empirical work to date on pair programming shows mixed results. Nosek (1988) and Williams et. al (2000) found a positive relationship between the use of pair programming and performance outcomes, such as software quality and developer satisfaction. However, Nawrocki and Wojciechowski's (2001) research does not show these same positive results. Additionally, little explanation has been offered to explain these findings (Domino et al. 2003) or explored diversity in this context.

This study represents an initial effort to investigate whether the diversity of the individuals making up the programming dyad impacts collaborative programming outcomes. Current practice in pair programming does not address this issue. Pair programming advocates like to say simply, "two heads are better than one." We test that premise in this study by examining the relationship between the nature of the programming dyad (based on combinations of several key individual differences) and performance outcomes. In addition, we consider how the diversity of the dyads, whether homogeneous or heterogeneous on the individual differences, impacts task conflict. Research is in progress on a laboratory experiment that explores whether dyad diversity impacts collaborative programming processes and outcomes; that is, we ask the question: are two heads *always* better than one?

Research Model

The research model is shown in Figure 1.

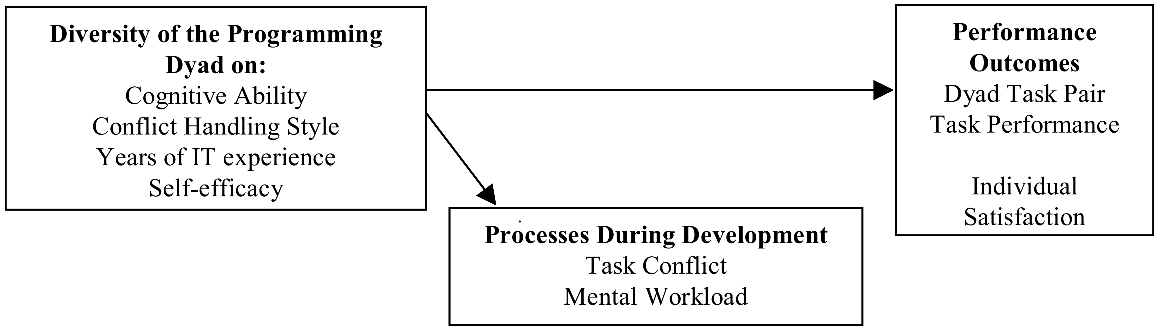

Figure 1. Research Model for the Study

This study is part of a larger research effort that investigates collaborative development methods in several settings. This study examines the relationship between outcomes in collaborative software development (in this case, pair programming), conflict during the development process, and the nature of the developer dyads.

The individual differences examined in this study are cognitive ability, cognitive handling style, IT experience, and self-efficacy. Each of these individual differences has been shown to impact job performance at the individual level. The most studied individual differences with strong links to job performance are cognitive ability and experience (Jex 2002). Experience has been found to be a better predictor of performance in low rather than highly complex jobs (McDaniel et al. 1988). Self-efficacy has also been shown to be a reliable predictor of task performance (Wood et al. 1989). Additionally, past research finds that that more cooperative conflict handling styles (i.e. the integrating style) are likely to produce positive outcomes. In this study these individual differences are used to understand whether a programming dyad is more homogeneous or heterogeneous.

When focusing on the diversity of the programming dyad, the collaborative process difference of interest is the amount of task conflict during development, since heterogeneous dyads are more likely to experience conflict. Past research has shown that low to moderate levels of task conflict lead to higher performance outcomes. Programming outcomes in this study include task performance (correctness of test cases and related pseudocode) and individual developer satisfaction.

Research Questions

The specific research questions addressed in this study are, in the context of collaborative programming:

- RQ1: Does the diversity of the programming dyad on individual characteristics significantly impact the programming outcomes of task performance and developer satisfaction?
- RQ2: Do more heterogeneous programming dyads (in terms of individual characteristics) have more task conflict during collaborative programming?

Literature Review

Collaborative Software Development (Pair Programming)

Agile methods are a set of development methods, derived from good practices and organized in an innovative process structure. The most immediate differences in these new agile methods are that they are *adaptive* and *people oriented* (Fowler 2000). As such, these methodologies are designed to enable rapid response to change while producing quality code in less time.

Collaborative programming (pair programming) is one such agile method that is rapidly gaining interest. This method involves two developers, working together in intense collaboration, producing one artifact. One developer takes the role of the driver, writing the code and using the keyboard, while the other functions as the navigator, monitoring results, looking for specific details and strategic defects. Periodically each partner switches roles, resulting in a highly interactive development process (Beck 2000).

Another distinctive and differentiating feature of collaborative programming is that the developers test first and then write the associated code. This somewhat unconventional sequence of formulating test cases before writing code is believed to be the reason for the reduced defect rate in code developed with pair programming (McBreen 2003).

Dyads and Small Groups

Research in small group behavior and outcomes dates back to the 1940s, with reviews of the literature beginning in the 1950s (McGrath et al. 1982). McGrath catalogued this early research and identifies three classes of variables: surround, group and member. The three classes of variables correspond to three separate levels of analysis: organizational, group (including dyad) and individual. Measurement of effectiveness can be performed along any one of the levels. In this study, we examine the combination of characteristics of the individuals in the dyad related to dyad effectiveness. The individual characteristics included in the study are cognitive ability, conflict handling style, experience and self-efficacy.

Cognitive Ability

Cognitive ability is defined as individual's capacity to process and comprehend information (Murphy 1989; Walden et al. 1989). As such, cognition is of particular relevance to the study of the intellectual activities associated with software development (Kemerer 1997).

General cognitive ability is a significant predictor of job performance in a variety of settings. Past research has shown that individuals with higher cognitive ability are typically better at problem solving (Schmidt et al. 1981). The strong and consistent importance of cognitive ability in job performance, with increased impact when tasks are novel, argues for including cognitive ability in studies of newer, less familiar development methods.

When working in groups, such as in collaborative programming, the highest performance is found when the average cognitive ability for the group is higher, although in some cases higher-cognitive-ability members of the group compensate for a low-cognitive-ability member (Barrick et al. 1998; Taggar et al. 1999). The findings from psychology that in groups, cognitive ability may have both additive (group average) and compensatory (higher ability group members help lower ability group members) impacts on performance make it particularly relevant to collaborative development environments. Roy and Dugal (1998) found that groups composed of individuals with similar cognitive processes achieved higher levels of performance than heterogeneous groups.

Conflict Handling Style and Conflict Processes

Small groups generate ideas through creative planning sessions, choosing alternatives based on logical intellective processes or preferences. Small groups also negotiate. When successful, this involves resolving viewpoints and conflicts of interest (Trimmer et al. 2000). In the small group (dyad) setting of programming, inevitably conflict arises.

Putnam and Poole (1987, p. 552) define conflict in the small group context "as the interaction of interdependent people who perceive opposition of goal, aim and values and who see the other party as potentially interfering with the realization of these goals". Knutson & Kowitz (1977) identified two general types of conflict in decision making groups: substantive or conflict rooted in the task, and affective, or conflict found in the group's interpersonal relationships. Prior research has found that conflict can have both negative and positive effects. Disagreement about a task is the most beneficial type conflict, with low to moderate levels of task conflict associated with higher performance.

One factor that has an important impact on conflict is the style that an individual uses to handle conflicts. Blake & Mouton (1964) present a conceptual schema for classifying modes of handling conflict into five distinctive modes or styles. Later, Rahim (1983a) and Rahim and Bonoma (1979) differentiate each mode based upon two basic dimensions: *concern for self* and *concern for others*. According to this model, individuals in groups have a range of five behaviors to utilize during group conflict: integrating, obliging, dominating, avoiding or compromising.

At the high level of concern for self, as well as the other party, is the integrating style (Rahim 1983b). The literature indicates that the more cooperative conflict management styles (in which a meaningful amount of concern is shown for the other party), and in particular the integrating style, are likely to produce positive individual and organizational outcomes (Burke 1970; Korbanik et al. 1993; Rahim 1983a). These findings were confirmed in a study of dyads, suggesting that an integrative conflict style is perceived as the most effective style (Gross et al. 2000).

Less cooperative styles (in which little concern is shown for the other party) frequently result in the escalation of conflict and negative outcomes (Burke 1970; Korbanik et al. 1993; Rahim 1983a). This occurs because the integrating style attends to both the outcome, as well as the effect of the conflict process on the relationship between the parties in conflict.

Experience

Empirical evidence suggests that experience has a strong positive linkage to performance (Jex 2002). McDaniel et al. (1988) and Schmidt and Hunter (1988) found that the relationship between experience and job performance is mediated by job knowledge. Since producing quality code, as is promised by collaborative programming, requires an understanding of both programming languages and general IT business knowledge, this study utilizes years of IT experience.

Self-Efficacy

Self-efficacy refers to a person's self-belief in his or her ability to perform a specific task and has been shown to be a reliable predictor of task performance (Appelbaum et al. 1996; Wood et al. 1989). According to Bandura (1991), self-efficacy beliefs are the most central and pervasive influence on the amount of effort that individuals apply to a particular task, how long they persevere at a task in the face of failure or difficulty and the amount of stress they experience. Marakas et al. (1998) posit that computer self-efficacy is a multileveled and multifaceted construct.

Performance Outcomes

Campbell (1990; 1994) proposed a job performance model, which incorporates the interaction of a number of variables including experience and interaction with others. Although often linked directly to productivity (Belinger et al. 1977; Downs et al. 1988) performance is measured by the *quality* of the outputs. In the case of collaborative programming the quality of performance has typically been measured by fewer errors, as well as reductions in total time on task.

Satisfaction is "the difference between the amount of rewards workers receive and the amount they believe they should receive" Robbins (1998, p. 25). This definition takes into account a variety of key elements including a mentally challenging work, a supportive work environment and theories related to personality – job fit. Developers' satisfaction with a novel work technique, such as pair programming, is particularly relevant, since early adopters who are satisfied are more likely to continue to use the technique, as well as serve as opinion leaders (Green et al. forthcoming).

Research Method

This research was performed at a university located in the southeastern United States. All scripts, instruments and experimental tasks were pre-tested during summer 2002. Based on these results, changes were incorporated as appropriate. A quasi-experiment using MIS majors was conducted during the fall 2002.

All participants completed an initial questionnaire that measures demographics and individual developer differences. Cognitive ability was measured using the Wonderlic Personnel Test (Wonderlic 1999) and the Rahim Organizational Conflict Inventory (Rahim 1993) was used to measure conflict-handling style. All subjects were trained in pair programming and then completed three programming tasks. Participants completed the programming tasks over multiple sessions, utilizing psuedocode so as to mitigate the impact of known programming languages. Upon completion of the tasks, all subjects answered questionnaires that measure processes and individual satisfaction and were debriefed.

The researchers developed Likert scales to measure the amount of perceived task conflict during development. Performance outcomes are measured by analysis of the pseudocode and test case outputs of the programming dyads. Additionally, individual developer satisfaction was measured utilizing a 7-point Likert scale. Data analysis is forthcoming.

References

Appelbaum, S.H., and Hare, A. "Self-efficacy as a mediator of goal setting and performance: Some human resource applications," *Journal of Managerial Psychology* (11:3) 1996, pp 33-46.

Bandura, A. "Social Cognitive Theory of Self-regulation," *Organizational Behavior and Human Decision Processes* (50) 1991, pp 248-287.

Barrick, M.R., Stewart, G.L., Neubert, M.J., and Mount, M.K. "Relating Member Ability and Personality to Work-team Processed and Team Effectiveness," *Journal of Applied Psychology* (83:3) 1998, pp 377-391.

Beck, K. *Extreme Programming Explained* Addison-Wesley, Boston, 2000.

Belinger, F., and Collins, R.W. "Identifying Candidates for Successful Telecommuting Outcomes," *The Information Society* (13) 1977.

Blake, R., and Mouton, F. *The Managerial Grid* Gulf, Houston, 1964.

Burke, R. "Methods of Resolving Superior-subordinate Conflict: The constructive Use of Subordinate Differences and Disagreements," *Organizational Behavior and Human Performance* (5) 1970, pp 393-411.

Campbell, J.P. (ed.) *Modeling the performance prediction problem in industrial and organizational Psychology*. Consulting Psychologists Press, Palo Alto, CA, 1990.

Campbell, J.P. (ed.) *Alternative Models of job Performance and their Implications for Selection and Classification*. Erlbaurn, Hillsdale, NJ, 1994.

Cockburn, A. "Just-In-Time Methodology Construction: Humans and Technology," 2000.

Domino, M.A., Collins, R.W., Hevner, A.R., and Cohen, C.F. "Conflict in Collaborative Software Development," Proceedings of the 2003 ACMSIGCPR Conference, Philadelphia, Pennsylvania, 2003.

Downs, C.W., Clampitt, P.G., and Pfeiffer, A.L. (eds.) *Communication and Organizational Outcomes*. Able Publishing, Norwood NJ, 1988.

Fitzgerald, B. (ed.) *System Development Methodologies -Time to Advance the Clock*. Plenum Press, New York, 1997.

Fowler, M. "Extreme Programming: What is a Lightweight Mythology?," 2000.

Glass, R. L. "Error-Free Software remains Extremely Elusive," *IEEE Software*), January-February 2003.

Green, G. C., Collins, R. W., and Hevner, A. R., "Perceived Control and the Diffusion of Software Process Innovations." *Journal of High Technology Management Research,* forthcoming.

Gross, M.A., and Guerrero, L.K. "Managing conflict appropriately and effectively: An application of the competence model to Rahim's organizational conflict styles," *International Journal of Conflict Management* (11:3) 2000, pp 200 - 226.

Holland, J.L. *Making Vocational Choices: A Theory of Vocational Personalities and Work Environments*, (2nd ed.) Prentice Hall, Englewood, NJ, 1985.

Jex, S.M. *Organizational Psychology: A Scientist-Practitioner Approach* John Wiley and Sons Inc, 2002.

Kemerer, C.F. *Software Project Management Readings and Cases* The McGraw-Hill Companies, 1997, p. 35.

Knutson, T.J., and Kowitz, A.C. "Effects of Information Type and Level of Orientation on Consensus Achievement in Substantive and Affective Small-Group Conflict," *Central States Speech Journal* (28:Spring) 1977, pp 54-63.

Korbanik, K., Baril, G., and Watson, C. "Managers conflict management style and leadership effectiveness: The moderating effects of gender," *Sex Roles* (29) 1993, pp 405-420.

Marakas, G. M., Yi, Y. M., and Johnson, R. D. "The Multilevel and Multifaceted Character of Computer Self-Efficacy: Toward Clarification of the Construct and an Integrative Framework for Research", *Information Systems Research*, (9:2) 1998.

McBreen, P. *Questioning Extreme Programming* Addison-Wesley, 2003.

McConnell, S. *Rapid Development: Taming Wild Software Schedules* Microsoft Press, Redmond, Washington, 1996.

McDaniel, M.A., Schmidt, F.L., and Hunter, J.E. "Job Experience Correlates of Job Performance," *Journal of Applied Psychology* (73) 1988, pp 327 - 353.

McGrath, J.E., and Kravitz, D. "Group Research," *Annual Review of Psychology* (33) 1982, pp 192-230.

Murphy, K.R. "Is the Relationship between cognitive ability and job performance stable over time," *Human Performance*) 1989, pp 183 - 200.

Nawrocki, J., and Wojciechowski, A. "Experimental Evaluation of Pair Programming," Proceedings of the 12th European Software Control and Metrics Conference, ESCOM 2001, Shaker Publishing, London, 2001, pp. 269-276.

Nosek, J. "The Case for Collaborative Programming," *Communications of the ACM* (41:3) 1988, pp 105-108.

Putnam, L.L., and Poole, M.S. (eds.) *Conflict and Negotiation*. Sage, Newbury Park, CA, 1987.

Rahim, M. "A Measure of styles of handling interpersonal conflict," *Academy of Management Journal* (26) 1983a, pp 368-367.

Rahim, M. *Rahim Organizational Conflict Inventory-II* Consulting Psychologist Press, Palo Alto, CA, 1983.

Rahim, M., and Bonoma, T. "Managing Organizational Conflict: a Model for Diagnosis and Intervention," *Psychological Reports* (44) 1979, pp 1323 -1344.

Robbins, S.P. *Organizational Behavior, Concepts, Controversies, Applications* Simon & Schuster, Upper Sadler River, NJ, 1998.

Roy, M.H., and Dugal, S. "Developing trust: the importance of cognitive flexibility and cooperative context," *Management Decision* (36:9) 1998, pp 561-567.

Schmidt, F., Hunter, J.E., and Pearlman, K. "Task Differences and the Validity of Aptitude Tests in Selection: A Red Herring," *Journal of Applied Psychology* (66) 1981, pp 166-185.

Schmidt, F.L., and Hunter, J.E. "The validity and utility of selection methods in personnel psychology: Practical and theoretical implications of 85 years of research findings," *Psychological Bulletin* (124) 1988, pp 262 -274.

Taggar, S., Hackett, R., and Saha, S. "Leadership Emergence in Autonomous Work Teams: Antecedents and Outcomes," *Personnel Psychology* (52) 1999, pp 899-926.

Trembly, A.C. "Software bugs cost billions, study says," *National Underwriter*), July 2002.

Trimmer, K., Blanton, J.E., and Collins, R.W. "Information Systems Development: Can there be "Good" Conflict?," *Proceedings of the 2000 ACM SIGCPR Conference*, Chicago, IL, 2000, pp. 174-179.

Walden, D.A., and Spangler, W.D. "Putting together the pieces: A closer look at the determinants of job performance," *Human Performance* (2) 1989, pp 29-59.

Williams, L., Kessler, R., Cunningham, W., and Jeffries, R. "Strengthening the Case for Pair Programming," *IEEE Software* (14), July -August 2000, pp 19-25.

Wonderlic, Inc. (1999*) Wonderlic Personnel Test & Scholastic Level Exam User's Manual*. Libertyville, IL: Author.

Wood, R., and Bandura, A. "Social Cognitive Theory of Organizational Management," *Academy of Management Review* (14:3) 1989, pp 361-384.