An Alternative: The Collaborative Software Process (CSP)

Laurie Williams
North Carolina State University
Department of Computer Science
williams@csc.ncsu.edu

Anecdotal and statistical evidence [1-3] indicates that pair programming two programmers working side-by-side at one computer, collaborating on the same design, algorithm, code or test is very effective. One of the programmers, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the observer, continuously observes the work of the driver to identify tactical (syntactic, spelling, etc.) defects, and also thinks strategically about the direction of the work. On demand, the two programmers can brainstorm any challenging problem. Because the two programmers periodically switch roles, they work together as equals to develop software. This pair-programming model was recently popularized by the eXtreme Programming [4] discipline, though it has been used sparsely in industry for quite some time [5, 6].

In 1999, Williams developed the Collaborative Software Process^{SM} (CSP^{SM}) as her dissertation research [7]. CSP, a disciplined process for pair programmers, is based on Watts Humphrey's Personal Software Process (PSP) [8]. CSP assumes basic PSP knowledge, such as that gained by taking a course using the *Introduction to the Personal Software Process* [9] book. However, the activities of the CSP are specifically geared towards leveraging the power of two software engineers working together on one computer. Like PSP, CSP can also be used as a foundation for moving into the Team Software Process (TSP) [10], whereby the members of the team work in pairs in for development engineer work.

A formal experiment with the CSP was performed with advanced undergraduates at the University of Utah in 1999. The results showed that pairs were able to complete their programs in about half the time of individuals and that the pairs produced programs of statistically significantly higher quality [3, 4, 7, 11]. Further experimentation is underway, including a formal experiment of TSP with individuals vs. TSP with pairs, which will take place in a two sections of a software engineering class at North Carolina State University in Spring Semester 2001.

Electronic educational materials to support the instruction of CSP are available from Laurie Williams. As previously stated, the CSP and these materials assume student background in PSP.

References

[1] J. T. Nosek, "The Case for Collaborative Programming," in *Communications of the ACM*, vol. March 1998, 1998, pp. 105-108.
[2] Wiki, "Programming In Pairs," in *Portland Pattern Repository*, vol. June 29, 1999, 1999, pp. http://c2.com/cgi/wiki?ProgrammingInPairs.

[3] L. Williams, R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the Case for Pair-Programming," in *IEEE Software*, vol. 17, 2000.
[4] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, Massachusetts: Addison-Wesley, 2000.
[5] L. L. Constantine, *Constantine on Peopleware*. Englewood Cliffs, NJ: Yourdon Press, 1995.
[6] J. O. Coplien, "A Development Process Generative Pattern Language," in *Pattern Languages of Program Design*, James O. Coplien and Douglas C. Schmidt, Ed. Reading, MA: Addison-Wesley, 1995, pp. 183-237.
[7] L. A. Williams, "The Collaborative Software Process PhD Dissertation," in *Department of Computer Science*. Salt Lake City, UT: University of Utah, 2000.
[8] W. S. Humphrey, *A Discipline for Software Engineering*. Reading, Massachusetts: Addison Wesley Longman, Inc, 1995.
[9] W. S. Humphrey, *Introduction to the Personal Software Process*. Reading, Massachusetts: Addison-Wesley, 1997.
[10] W. S. Humphrey, *Introduction to the Team Software Process*. Reading, Massachusetts: Addison Wesley, 2000.
[11] L. Williams and H. Erdogmus, "An Economic Analysis of Collaborative Programming," submitted to Metrics 2001, London, England, 2001.