THE QUALITY REVOLUTION IN SYSTEMS DEVELOPMENT: HOW CAN THE ACADEMIC COMMUNITY INTRODUCE THESE PROCESSES INTO THE IS CURRICULUM?

Girish Seshagiri, CEO, Advanced Information Services, girishs@advinfo.net
Larry W. Cornwell, Professor, Bradley University, larryc@bradley.edu

ABSTRACT

Watts Humphrey of the Software Engineering Institute at Carnegie Mellon University has been very successful in helping industry improve the process of software development, but very little has been developed using his methods in the academic community. This paper presents the case for teaching Personal Software Process and Team Software Process in the Information Systems curriculum and is underscored by one author's experience of successfully using these processes in his company.

Keywords: Software development, system development, SEI, Watts Humphrey, Capability Maturity Model, Personal Software Process, Team Software Process, TSPi

INTRODUCTION

The Problem

In the keynote address of the Illinois Workshop On Software/Systems Development Education conference, Watts Humphrey pointed out, "The current performance of software groups is poor." His comment about software development project teams performance was based on the graph in Figure 1 below. "While there is a gradual improvement, it is very gradual and 72% of projects were not considered successful"(10). The data reported in the graphs was obtained from the Standish Group Reports of 1994 and 2000.

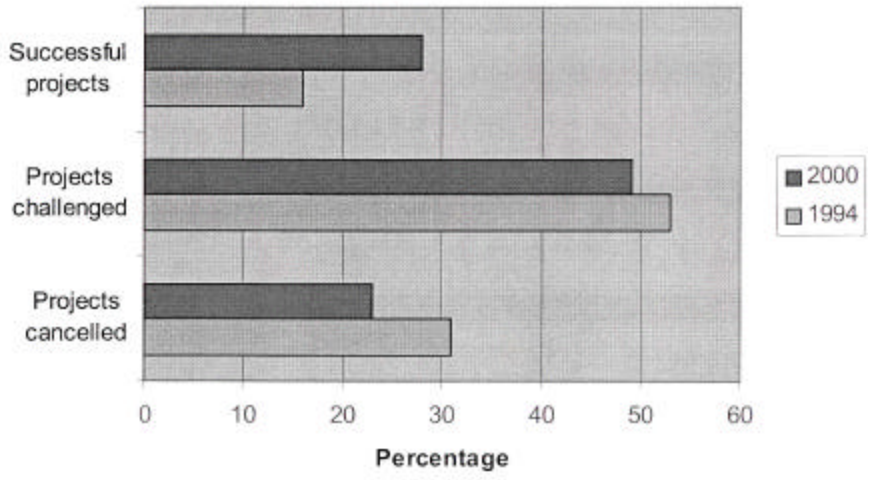

Figure 1 Standish Group Reports

The concern voiced by Humphrey is not new. Edward Yourdon published two books (14) (15) concerned with the failure of the American programmer in producing quality software and the movement to offshore software development. Unfortunately he did not identify any "silver bullets" that would solve the problem. He only suggested several that might be of help. In Watts

Humphrey's case, he has developed an approach to solving the problem and has backed up the recommendations with success in the industry.

The Solution

Watts Humphrey has been instrumental in the founding of the Software Process Program of the Software Engineering Institute at Carnegie Mellon University. He initially introduced the Capability Maturity Model, CMM, which provides an organization the framework for organizing small evolutionary steps into five maturity levels that lay successive foundations for continuous process improvement. Although the CMM provides a powerful improvement framework, its focus is on "what" organizations should do and not "how" they should do it. Watts then developed the Personal Software Process, PSP, to show engineers how to apply the CMM process principles. Finally it was observed that engineers with PSP training and proper support from management still have a problem combining their personal processes into an overall team process. Watts then developed the Team Software Process, TSP. TSP extends and refines the CMM and PSP methods to guide engineers in their work on development and maintenance teams. Companies that have adopted these models have experienced many success stories (9).

Need of Academic Involvement

In Watts Humphrey's keynote address on March 8, 2002, he described what he felt was the academic challenge: 1) To behave professionally, engineers must know how to behave as responsible citizens; 2) They must feel personally responsible for doing predictable and high quality work; 3) This responsible attitude cannot just come from a course, it must be ingrained; 4) Building convictions takes time, supervised experience, and informed guidance and 5) To provide this, software engineering programs must offer extensive laboratory experience. He feels that the values of the software community must change and this will not happen until academic values change as well (10).

DEFINITIONS AND DESCRIPTIONS

Software Engineering Institute

The Software Engineering Institute, SEI, is a federally funded research and development center sponsored by the U. S. Department of Defense through the Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics. SEI's core purpose is to promote assistance to others to make measured improvements in their software engineering capabilities. Its vision is: "The right software, delivered defect free, on time and on cost, every time. SEI's mission is to provide the technical leadership to advance the practice of software engineering so the DoD can acquire and sustain its software-intensive systems with predictable and improved cost, schedule, and quality". Watts Humphrey was the founder of the Software Process Program of SEI and is a Fellow and research scientist of the Institute. Watts was associated with the IBM Corporation from 1959 to 1986 where he was director of programming quality and process (1).

Capability Maturity Model

The Capability Maturity Model, CMM is a model that describes the practices of an organization at a particular software process maturity level. CMM is used to guide process improvement and assess software capability. There are five CMM maturity levels: 1) Initial, 2) Repeatable (using project management), 3) Defined (using engineering processes), 4) Managed (establishing product and process quality), and 5) Optimizing (using continuous process improvement). These five levels reflect the fact that the CMM is a model for improving the capability of software organizations (2).

Personal Software Process

The Personal Software Process, PSP, provides guidance for individual engineers to improve their performance by bringing discipline to the way they develop software. The PSP guides engineers in the management of the quality of their products. PSP can be applied to many phases of the software development process, such as small-program development, requirements definition, document writing, systems testing and the maintenance and enhancement of large software systems. The PSP has been documented to successfully improve the estimating and planning ability of engineers while reducing the defects in their software developed (12).

Team Software Process

The Team Software Process, TSP, was developed to help integrated engineering teams more effectively develop software-intensive products. The four principal phases of TSP are: 1) Requirements, 2) Design, 3) Implementation, and 4) Test. TSP assists engineering groups in applying integrated team concepts to the development of software-intensive systems. It outlines a 4-day launch process that establishes goals, defines team roles, assesses risks, and produces a comprehensive team plan. It also provides a defined and measured process framework for managing, tracking, and reporting on the team's work. TSP shows teams of engineers how to produce quality products for planned costs and on aggressive schedules (13). Iraj Hirmanpour illustrates how these models work together in Figure 2 (5).

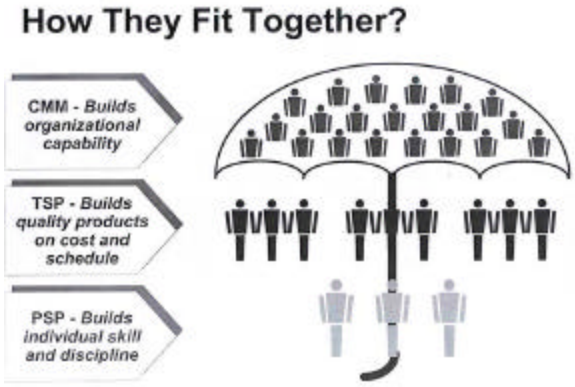

Figure 2 SEI Models

EXPERIENCE OF A PRACTITIONER

Advanced Information Services, AIS, is a software development and consulting company headquartered in Peoria, Illinois. During our early years, entire software projects became unmanageable because we were overwhelmed with quality problems and we did not know how to manage quality. We lost money more times than we made money. It was apparent that we needed to change. But it was not a simple change or just a matter of telling people what to do.

We needed an organizational transformation and it was clear that I had to personally lead this transformation.

Executive Sponsorship

One of the first things I knew I had to do was to connect with my people at an emotional level. I needed tools to communicate. I bought everyone a copy of Watts Humphrey's *Managing the Software Process* book (8). It has turned out to be the best business investment I ever made. It is to this day a requirement for employment in the AIS High Velocity Development group that they read this book. CMM itself became a good tool to communicate regarding what to change and the end state for our transformation.

Watts made an outrageous commitment to transform the world of software development. He developed CMM, TSP, and PSP to transform organizations, teams, and individuals, respectively. At AIS, we followed Watts's journey. We also realized that for any theory or concept to be useful, it must ultimately be translated into the language and context of managers and managerial action.

It was up to me to set this context in terms of our business goals for our transformation. I knew that if I did not do this, no one else will. I set our goal as transformation to be an excellent software company. What does that mean for our managers and engineers?

I feel that an excellent software company is one who delivers products on time with zero defects. The organization does this by not relying on testing, but by detecting and removing up to 95%of defects prior to compile and test. The organization believes and practices defect prevention. As Crosby said, defect prevention is easy to sell, but hard to do (3).

In an excellent company, employees know and understand the company's mission. Also, they know what excellent practices are and actually use them. I might add that the employees of AIS came up with the reason why we are in business. It was to continuously advance the boundaries of quality. I could not have come up with a lofty goal like that myself.

The excellent company also uses mature processes and has mechanisms in place for the employees to continuously improve the process. The results we are looking for are driving the cost of quality down to 5% and doubling our productivity every 3 to 5 years. Silver bullet solutions will not get us there.

The ultimate business goal is that we want not just satisfied but delighted customers. I submit to you that we cannot delight our customers if we deliver defective products.

AIS Transformation

With CMM based improvement, our average project schedule overrun was reduced from 112 % to 37%. When engineers were trained in the PSP, the average schedule overrun decreased dramatically to around 12%. The current industry average is around 100%, as published by the Standish group. The average budget overrun was reduced from 87% to 17% with CMM based

improvement. When engineers were trained in PSP, and we made plans based on engineers' personal plans, the budget overrun decreased dramatically to –4%. We now average less than one defect per 10,000 lines of code. One half of the products delivered in the past three years have been defect free. The average time in test has been reduced dramatically.

The men and women at AIS who produced such outstanding results were recognized by being awarded the 1999 IEEE Computer Society Software Process Achievement Award.

You have seen what others had to say about PSP and TSP and the results experienced by their organizations. AIS reported the following in the IEEE Software magazine in late 2000 (4).

- "You actually get your money back after 1200 lines of code"
- "The data says that if you follow this, you get absolutely the cheapest time to market"
- "The first company to really get this right will have a competitive advantage nobody will be able to keep up with"
- "It is so revolutionary that I remember the exact date I was introduced to it"
- "It is hard to believe unless you do it"

These results were made possible because of a culture of discipline. When you have disciplined people, you don't need hierarchy. The industry needs the help of the academic community to instill discipline in the software professional early in student's academic careers. It is too late afterwards. PSP and TSP in the IS curriculum will provide the foundation for the culture of discipline. When we combine the culture of discipline with entrepreneurship, we can achieve great performance. For true transformation to happen, it must begin with the educational system.

ACADEMIC INVOLVEMENT

In a presentation at the Illinois Workshop On Software/Systems Development Education conference, Iraj Hirmanpour listed several problems with software education today: 1) Little understanding of use of design in development; 2) Heavy reliance on testing to assure quality; 3) No basis for evaluating and improving the way students do their work; and 4) Difficulty in scaling up what is learned in small projects to larger projects. He suggested the students need to learn: 1) Apply engineering principals to software development; 2) Tedious detailed work; 3) Act of construction not creation; and 4) CMM, PSP, TSP provide the roadmap to apply engineering principals to software construction (5).

The suggested curriculum implementation was an Introduction to Software Engineering or similar course be modified to teach full PSP, using the textbook *A Discipline for Software Engineering*. (6) Then assuming students have learned PSP in previous courses, TSP can then be incorporated into project based software development course using *Introduction to Team Software Process* as the textbook. (7)

Introduction to Software Engineering

The first course would include PSP. PSP is based on the following principles: 1) Plan the work before committing to a job, 2) Use a defined process to plan and to do the job, 3) Measure and

track time, size, and defects, 4) Emphasize quality from the beginning of the job, 5) Plan, measure, and track program quality, and 6) Analyze every job and use the results to do better the next time. PSP uses the process elements: 1) scripts, 2) measures, 3) forms, and 4) standards. The process scripts and standards guide the PSP process: 1) plan, 2) design, 3) design review, 4) code, 5) code review, 6) compile, 7) test, and 8) postmortem. The PSP process produces time and a defect log that is used to produce a project plan summary.

Project Based Software Development Course

TSP is based on the following principles: 1) The engineers know the most about the job and can make the best plan, 2) When engineers plan their own work, they are committed to the plan, 3) Precise project tracking requires detailed plans and accurate data, 4) Data is most accurate when gathered by engineers while they work, and 5) To maximize productivity, focus first on quality.

To help the academic community gear up for the PSP/TSP concepts, SEI is offering faculty development workshops.

SOFTWARE ENGINEERING INSTITUTE FACULTY DEVELOPMENT WORKSHOPS

SEI has been co-sponsoring summer faculty workshops on the PSP and Introductory TSP, TSPi, for the past six years. This past July a new, one-week workshop, "Teaching A Software Team Project Course: Improve Team Performance by Teaching Engineering to Teams" was offered at Southern Polytechnic State University in Atlanta, Georgia. The workshop is co-sponsored by SEI, Southern Polytechnic State University, the Yamacraw project and a faculty development grant from the National Science Foundation.

The workshops are based on SEI's many years of experience in teaching previous workshops on PSP and TSP. TSP, supported by PSP, was designed for industrial software teams and has been very successful. TSPi was developed for academic use, with student teams in a software project course.

In the workshop, faculty members are shown how teams can use the TSPi to produce quality software, on time and within budget. Faculty will be provided guidelines, process scripts, tools, methods and techniques for developing a software product by a team. In addition, faculty will be provided advice, techniques, and activities for building effective teams. Faculty will experience hands-on activities with other faculty working on a case-study project. They will learn about these techniques and also how to teach them (10).

CONCLUSIONS

This paper suggests that the academic community include in the curriculum of software developers the processes developed and proposed by Watts Humphrey of the Software Engineering Institute. Engineers/programmers who are participating in software development need to experience and practice the processes of PSP and TSP early in their academic career. It is suggested that PSP be introduced in the Introduction to Software Engineering or similar

course. Then TSP can be taught in a project based software development course. Students with this preparation will be very successful in the software development process and also be very attractive to those companies working toward level five of the Capability Maturity Model.

REFERENCES

1. About the SEI. Retrieved June 1, 2002 from the Software Engineering Institute Web site: http://www.sei.cmu.edu/about/about.html
2. Capability Maturity Model for Software. Retrieved June 1, 2002 from the Software Engineering Institute Web site: http://www.sei.cmu.edu/about/about.html
3. Crosby, Philip B. *Quality is Free – The Art of Making Quality Certain*. New York: McGraw Hill.
4. Goth, Greg. (November/December 2000). The Team Software Process: A Quiet Revolution. *IEEE Software*, Vol. 17, No. 6.
5. Hirmanpour, Ira. (March 2002). Process Centric Software Education: Curricula & Implementation Issues. Retrieved June 1, 2002, from Illinois State University Extended University Web site: http://www.exu.ilstu.edu/IWSSDE2002/
6. Humphrey, Watts. (1995). *A Discipline for Software Engineering*. Reading, MA: Addison-Wesley.
7. Humphrey, Watts. (2000). *Introduction to Team Software Process*. Reading, MA: Addison-Wesley.
8. Humphrey, Watts. (1990). *Managing the Software Process*. Reading, MA: Addison-Wesley.
9. Humphrey, Watts (June, 1999). Pathways to Process Maturity: The Personal Software Process and Team Software Process. Retrieved June 1, 2002, from Software Engineering Institute of Carnegie Mellon University Web site: http://interactive.sei.cmu.edu/Features/1999/June/Background/Background.jun99.htm
10. Humphrey, Watts. (March 2002). Teaching Disciplined Software Engineering. Retrieved June 1, 2002, from Illinois State University Extended University Web site: http://www.exu.ilstu.edu/IWSSDE2002/
11. "PSP and TSP Faculty Workshops." Retrieved June 1, 2002 from the Software Engineering Institute Web site: http://www.sei.cmu.edu/tsp/workshop.html/
12. "The Personal Software Process (PSP)". Retrieved June 1, 2002 from the Software Engineering Institute Web site: http://www.sei.cmu.edu/tsp/psp.html/
13. "The Team Software Process (TSP)". Retrieved June 1, 2002 from the Software Engineering Institute Web site: http://www.sei.cum.edu/tsp/tsp.html/
14. Yourdon, Edward. (1993). Decline & Fall of the American Programmer. Englewood Cliffs, NJ: Prentice Hall, Inc.
15. Yourdon, Edwards. (1996) Rise & Resurrection of the American Programmer. Englewood Cliffs, NJ: Prentice Hall, Inc.