Results of Applying the Personal Software Process

Too often, software developers follow inefficient methods and procedures. The Personal Software Process, developed by Watts Humphrey at the Software Engineering Institute, provides software engineers with a methodology for consistently and efficiently developing high-quality products. The value of PSP has been shown in three case studies.

Pat Ferguson
Advanced Information Services

Watts S. Humphrey
Software Engineering Institute

Soheil Khajenoori
Embry-Riddle Aeronautical University

Susan Macke
Motorola

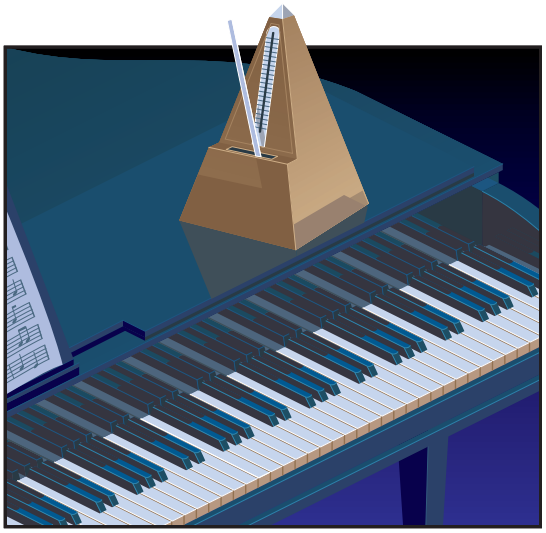
Annette Matvya
Union Switch & Signal

In most professions, competent work requires the disciplined use of established practices. It is not a matter of creativity versus discipline, but one of bringing discipline to the work so that creativity can happen. The use of plans and procedures brings order and efficiency to any job and allows workers to concentrate on producing a superior product. A disciplined effort, too, removes waste, error, and inefficiency, freeing financial resources for better uses.

Sadly, software professionals today often do not plan or track their work, and software quality is rarely measured. This is not surprising as software engineers are generally not taught planning, tracking, or quality measurement. When software organizations do plan, it is only at the project level, and few software organizations measure the quality of their work.

The Personal Software Process is a defined and measured framework that helps software engineers plan and track their work and produce high-quality products. PSP shows engineers how to manage the quality of their products and how to make commitments they can meet. It also provides them with the data to justify their plans. PSP can be applied to many parts of the software development process, including small-program development, requirements definition, document writing, systems tests, and maintenance and enhancement of large software systems.

PSP has been shown to substantially improve the estimating and planning ability of engineers while significantly reducing the defects in their products. PSP is introduced with a course, and during the course productivity improvements average around 20 percent and product quality, as measured by defects, generally improves by five times or more. While PSP is new, a growing number of engineers use it and find it helpful.



PERSONAL SOFTWARE PROCESS

In his work at the Software Engineering Institute at Carnegie Mellon University, Watts Humphrey began developing PSP in 1989. The program came about when groups began asking SEI how to apply its Capability Maturity Model to small projects. CMM, also a product of SEI, helps software organizations improve their development and maintenance capabilities by assessing their work according to five maturity levels and 18 key process areas. The CMM is widely used in both the public and private sectors to guide the evaluation and improvement of software organizations.[1-3]

CMM and PSP are thus mutually supportive, with CMM addressing management practices and PSP defining a disciplined way for engineers to do their work. With PSP, engineers practice 12 of CMM's 18 key process areas. PSP training can also help accelerate an organization's CMM process improvement program.

PSP is a process framework and set of methods that help engineers be more disciplined in their work. It shows them how to estimate and plan their projects, measure and track their work, and improve the quality of the products they produce.

0018-9162/97/$10.00 © 1997 IEEE

PSP consists of a series of scripts that define tasks, of forms for recording data, and of standards that govern such things as coding practices, size counting, and the assignment of defect types. When engineers follow PSP, they first plan their work and document the plan. As they do their work, they record their times and track and report every defect they find. At the end of the project, the engineers do a postmortem analysis and complete a project plan summary report.

For larger projects, PSP's task and schedule templates guide engineers through the steps of developing a schedule. This helps the engineers think through in advance how they will do the work, and thus they go down fewer blind alleys, make few mistakes, and follow an overall strategy for attacking the work.

PSP's quality improvements result from three key aspects: First, by tracking all defects, engineers are sensitized to the mistakes they personally make and therefore become more careful in their work. Second, when they analyze their defect data, they gain a clearer understanding of the cost of removing defects and thus apply the most effective ways of finding and fixing them. And third, PSP introduces a number of quality practices that have proven effective in preventing defects and in efficiently finding and fixing them.

Training is key

Successful application of PSP requires training. PSP can be studied in a one-semester graduate-level university course, which includes a PSP textbook and 10 programming and five analysis exercises.[4] PSP is now being taught at more than 20 universities in the US and at institutions in Europe, South America, and Australia. Some universities, too, now offer an introductory freshman course that is designed to start engineers off on the right track.[5] Finally, a condensed version of the university course, also with 10 programming and five analysis exercises, is available for industrial software groups.

Both university and industry PSP courses work through seven process levels while the engineers develop 10 module-sized programs. Each process level introduces several elements of PSP, accompanied by applicable scripts, forms, and templates. New levels build on material taught in preceding levels, allowing engineers to practice what they learn and to see the benefits of PSP procedures before moving on to the next level.

Seven process levels

Figure 1 shows the seven process levels. Each new level introduces new elements and more complicated material until the engineers reach the highest level, PSP 3. The Team Software Process (TSP), now in development at SEI, extends the PSP approach to the software team environment.

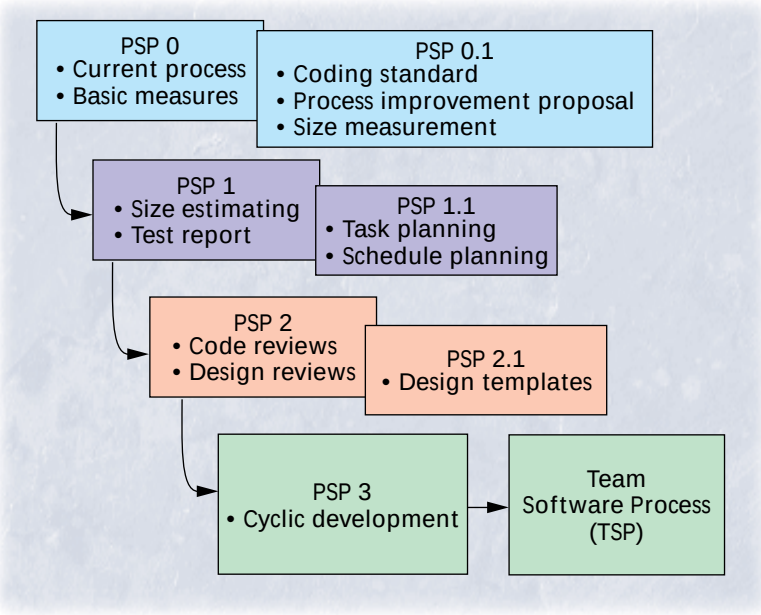


Figure 1. PSP process evolution.

- *PSP 0.* In the first level, engineers essentially follow their current practices, learning some basic PSP techniques. This level covers how to record development time and how to log each compile-and-test defect. These measurements are used in process analysis and planning and as a benchmark for assessing improvement.
- *PSP 0.1.* This level adds size measurement and the *process improvement proposal*, a form that engineers use to record the process problems they encounter as well as their ideas for addressing them. These problems can make processes very inefficient, but because many amount to minor details, engineers tend to forget them without the use of a form.
- *PSP 1.* In this level, engineers are introduced to the PROBE method, which uses historical data to estimate size and determine the accuracy of the estimate. PROBE is a regression-based size-estimating method developed specifically for PSP.
- *PSP 1.1.* This level adds resource and schedule estimating and *earned-value tracking*. Engineers often have trouble tracking their work because they do tasks in an order different from their plan. Earned-value tracking allows them to weight the relative importance of each task and to judge their progress as they finish some tasks early and others late.
- *PSP 2.* This level introduces design and code reviews, as well as quality measurement and evaluation. Using defect data from their earlier exercises, engineers also develop personal design and code review checklists.

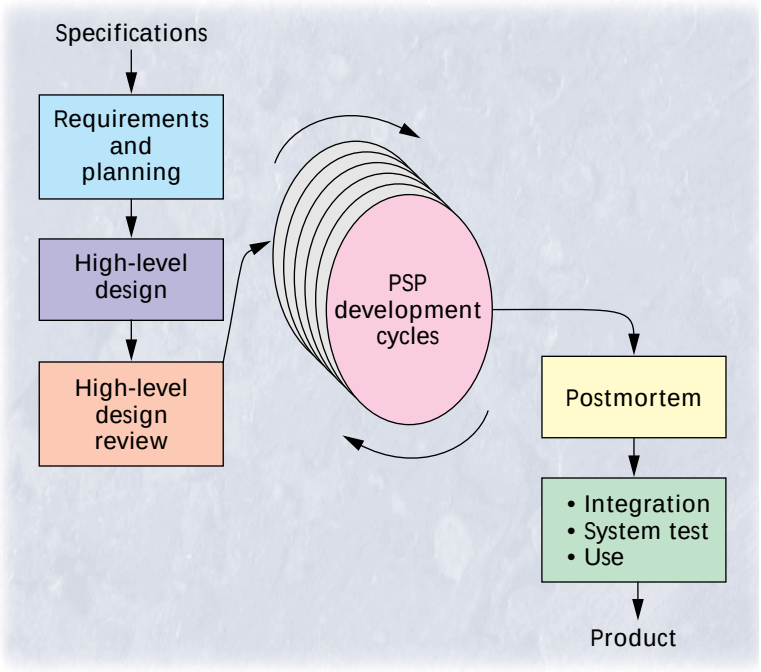


Figure 2. PSP 3 process.

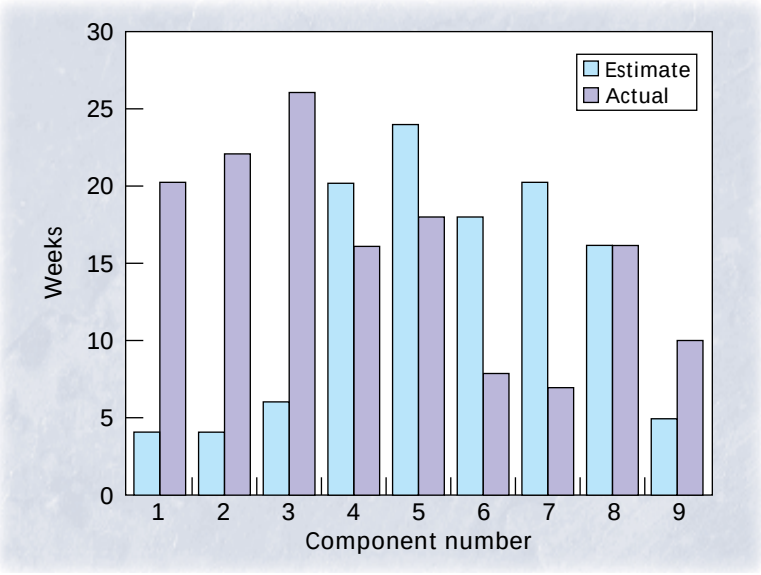


Figure 3. AIS Project A schedule estimates.

- *PSP 2.1*. In this level, engineers learn design specification techniques and ways to prevent defects.
- *PSP 3*. In this highest process level, software engineers become fully conversant in PSP. This level covers design verification techniques and methods for adapting PSP to engineers' working environments.

By applying the techniques and procedures learned during the course, engineers see how to apply PSP to large-scale software development. Figure 2 shows the spiral-like strategy of PSP level 3 for developing program modules of up to several thousand lines of code. This cyclic process builds on several well-known software engineering principles: First, by using abstraction and modular design concepts, engineers are better able to produce clean designs and to capitalize on reusable parts. Second, this cyclic development strategy follows the common practice of building large programs through a family of progressively enhanced versions. Finally, the process incorporates the divide-and-conquer strategy of Barry Boehm's spiral model for minimizing risks by attacking complex problems a step at a time.[6]

SEI's data on 104 engineers shows that, on average, PSP training reduces size-estimating errors by 25.8 percent and time-estimating errors by 40 percent. Lines of code written per hour increases on average by 20.8 percent, and the portion of engineers' development time spent compiling is reduced by 81.7 percent. Testing time is reduced by 43.3 percent, total defects by 59.8 percent, and test defects by 73.2 percent.[7,8]

CASE STUDY OVERVIEW

Because PSP was only experimentally introduced in 1994 and has been undergoing further development and introduction to the industry over the past two years, relatively little data on its use and effectiveness are available.[4,9] This problem is compounded by the time required to introduce PSP into a workplace and by the length of many software development efforts. Furthermore, pre-PSP data are often unavailable.

Nevertheless, three industrial software groups have used PSP and have collected data to show its effectiveness. They are Advanced Information Services, Inc., Motorola Paging Products Group, and Union Switch & Signal Inc. Each has trained several groups of engineers and measured the results of several projects that used PSP methods. In all cases, the projects were part of the companies' normal operations and not designed for this study.

The three companies offered a variety of situations useful for demonstrating the versatility of PSP. The projects at Motorola and US&S involved software maintenance and enhancement, while those at AIS involved new product development and enhancement. Among the companies, application areas included commercial data processing, internal manufacturing support, communications product support, and real-time process control. Work was done in C or C++.

Company sizes ranged from less than a hundred employees to thousands in a large international corporation. Most projects involved one to three engi-

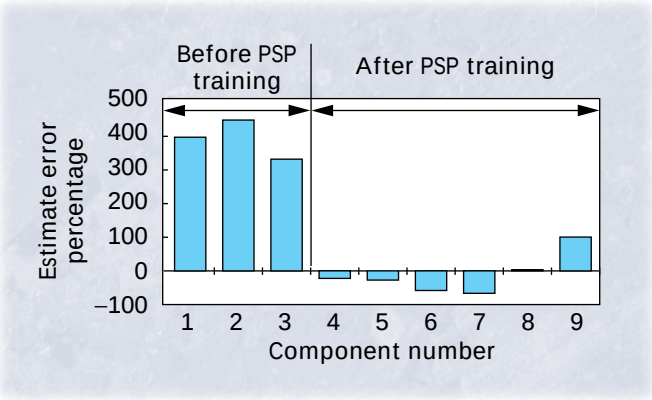


Figure 4. AIS schedule estimating error.

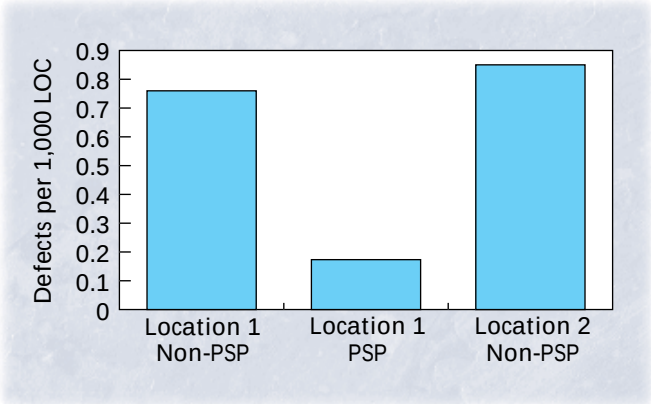


Figure 5. AIS acceptance test quality.

neers, but one of the AIS projects used two groups—one in the US and the other in India—of three to five engineers. While most of the projects were for commercial clients, one program was developed for an agency of the US government.

Advanced Information Services

AIS is located in Peoria, Illinois, with a subsidiary in Madras, India. The company offers software development, consulting services, Internet services, and process training. AIS engineers develop custom business applications for clients or do contract work at client sites.

AIS first introduced PSP with a pilot course in the spring of 1994. The course was given outside regular work hours with instructors who had not been trained in PSP, so it was perhaps not surprising that only half the engineers completed the course. As a result, AIS sent one engineer to SEI for PSP instructor training, put the course on company time, and tried various formats to shorten course time. Since then, AIS has presented five PSP courses, and essentially all the engineers have completed the work. The improvements during the course have been similar to SEI's general findings.

Project A. One of AIS's first applications of PSP was in 1995 when engineers in Peoria (working with another team in Madras) developed software for a Fortune 50 client. The Peoria components of Project A ranged from about 500 to 2,200 lines of code. By April 1995 the engineers had completed components 1, 2, and 3, but the project was not meeting its internal target dates or its external commitments. Project A had to be replanned and new delivery dates negotiated with the client.

At this point, management decided to train the Peoria engineers in PSP, and they subsequently used PSP methods to plan and develop components 4 through 9. Figure 3 shows the schedule performance for all nine components, and Figure 4 shows the schedule estimating errors. Before PSP training, schedule estimating error averaged 394 percent; afterward the average was –10.4 percent.

Quality also improved after PSP training. Figure 5 shows that in acceptance tests, the software of the Peoria engineers had 0.76 defects per 1,000 lines of code before PSP training and 0.17 defects per 1,000 lines of code after training, a 78 percent improvement. The Madras engineers, who had not received PSP training, had 0.85 acceptance-test defects per 1,000 lines of code. PSP also improved productivity. Figure 6 shows that the PSP-trained engineers wrote 7.99 lines of code per hour before training and 8.58 lines of code per hour after training, an improvement of 7.4 percent. The untrained Madras engineers wrote only 6.4 lines of code per hour. Since this product has not yet been installed, there is no usage data.

Projects B, C, and D. Project B, an application enhancement effort by three PSP-trained engineers in 1996, is similar to Projects C and D, which were com-

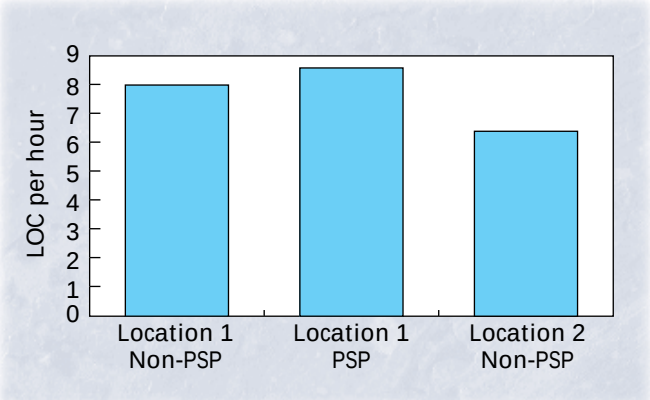


Figure 6. AIS productivity.

Table 1. Summary of AIS project data.

Project	PSP staff	Non-PSP staff	Product size	Delivery: Planned/ actual (months)	Acceptance test defects	Usage defects
B	3	0	24 requirements	7/5	1	0
C	0	3	19 requirements	2/5	11	1
D	0	3	30 requirements	10/19	6	14
E	1	0	2,255 LOC	6/6	0	0
F	1	0	1,400 LOC	2/2	0	0
G	2	1	6,196 LOC	2/2	0	3

pleted in 1995 and 1996, respectively, by AIS engineers not trained in PSP. All projects were of similar size, used the same platform, language, and database tools, and required three engineers. Of the three projects, Project C's engineers had the most experience. Table 1 shows results for these projects.

In acceptance tests, Project C had 11 defects, Project D had six, but Project B had only one. After several months of use, customers found one defect in Project C, 14 in Project D, but none in Project B. Because data on lines of code are not available, Figure 7 shows the acceptance and use defect data for these products normalized by the number of requirements. While the requirement count is only a crude size measure, it is the only data available. Project C was scheduled for two months, but took five to reach the acceptance test stage; Project D was scheduled for 10 months, but took 19; Project B, however, was scheduled for seven months, but took five.

Projects E, F, and G. AIS undertook three other projects using PSP-trained engineers. Projects E and F, with 2,255 and 1,400 lines of code, respectively, both used one PSP-trained engineer. Project G, with 6,196 lines of code, required three engineers, two of whom were trained in PSP. Both Projects E and F were completed on time, with no defects found during customer acceptance and after several months of use. Project G was also completed on time, but the customer reported three defects. Table 1 and Figure 7 show data for all three projects.

Project E is noteworthy because it was a cost-plus government contract (the government paid for actual development costs plus a percentage for profit) and was completed on schedule and substantially under budget. During the project, the engineer provided the client and AIS management with his weekly PSP task and schedule planning templates, allowing them to easily track project status. The development manager reported that these and other PSP practices sharply reduced the need for management supervision.

In addition to quality and productivity, PSP can help shorten project schedules. Table 2 shows system test time for the seven AIS projects discussed plus three newer projects. Before PSP training, system testing took as much as several months. For example, Project C's three testing cycles took several weeks. However, after PSP training, system test time was only a few days. The only exception was Project A2, whose system test took 1.5 months because it had to be tested with Project A1.

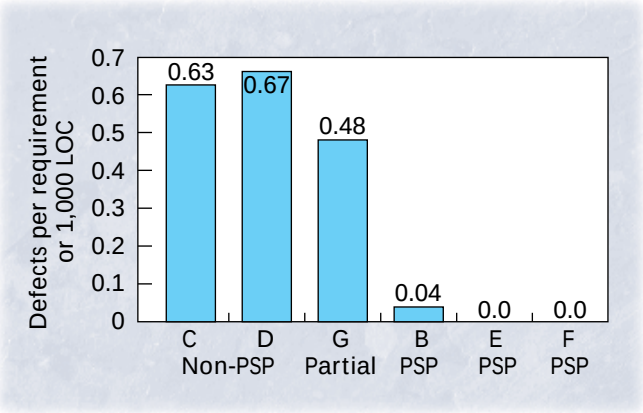


Figure 7. AIS project defects.

AIS plans for PSP

All engineers and managers at the AIS subsidiary in India have now completed PSP training. In the US, 58 percent of engineers and managers have completed training, and all should be fully trained by the end of this year. AIS now trains all new engineers in PSP before they are assigned a project. They expect this to reduce the disruption of training in the middle of a project and result in more general use of PSP.

MOTOROLA PAGING PRODUCTS GROUP

The North American Paging Subscriber Division of the Motorola Paging Products Group, located in Boynton Beach, Florida, develops and manufactures one-way numeric and alphanumeric pagers. Using simulcast broadcast techniques, these products provide digital message service. The embedded code in a pager provides message handling, user-friendly

prompts, and automatic message reception.

Three Motorola managers and two professors from Embry-Riddle Aeronautical University introduced PSP at the company. So far, Motorola has presented three PSP classes, training 40 engineers and 22 managers. The Motorola Paging Products Group has initiated a program to make all Motorola divisions aware of and literate in PSP.

Motorola's management has strongly supported PSP. The company commits eight hours per week for training and another four hours for process improvement. To build PSP team spirit, they give trained engineers customized sports shirts with a PSP logo, hold discussion lunches, and provide a graduation party where a certificate is presented in the presence of colleagues and family members. Engineers who complete the course receive financial rewards and join a user group that meets once a month.

These efforts have resulted in a high level of participation in PSP training and continued use of PSP methods. A Motorola post-training survey of 12 engineers showed that most would continue using PSP. Another survey six months after the pilot course showed that more than 80 percent of the engineers used PSP methods in their work and that 77 percent of the engineers used PSP methods in nonsoftware tasks.

Motorola project results

Engineers at Motorola first used PSP in a project that changed five lines of code in a large program supporting a manufacturing area. Although coding time was slightly more than planned, integration and test time were reduced by 60 percent. The project took 44 percent less time than planned and was completed on time.

One defect was found in code review (which took two minutes to fix), but no defects were found during system testing, and none have been found in over five months of operation—which is unusual. This was critical for Motorola because the manufacturing station was required to work on the software, and any testing and debugging interrupts production, costing several thousand dollars per hour in lost manufacturing capacity.

A second project involved adding menus to existing product-test software. This project achieved early removal of more than 80 percent of defects. Only 20 percent of development time was spent in testing, and 56 percent of development time came before coding.

A third project using PSP at Motorola was a crash effort of one engineer to add 3,894 lines of code to the 9,150 lines of code in a pager-support system. By using PSP planning and tracking methods, this engineer precisely tracked his status every day. He fell one day behind schedule on three occasions, but was able to quickly recover. He was able to find 80 percent of defects before the first test, and he completed system testing one week early. No defects have since been found in more than four months of product use.

So far, PSP-trained engineers at Motorola have completed 18 projects. As shown in Table 3, several of these products have been used for many months, and only one defect has been found in one of them. Unfortunately, detailed usage data were not gathered on the products labeled "NA," but there is no record of any defect reports for any of them. In the case of the single defect report, Motorola was unable to deter-

Table 2. AIS system test time improvement.

Project	Size	System test time
Non-PSP Projects		
A1	15,800 LOC	1.5 months
C	19 requirements	3 test cycles
D	30 requirements	2 months
H	30 requirements	2 months
PSP Projects		
A2	11,700 LOC	1.5 months
B	24 requirements	5 days
E	2,300 LOC	2 days
F	1,400 LOC	4 days
G	6,200 LOC	4 days
I	13,300 LOC	2 days

Table 3. Motorola operational defect data for PSP projects.

Project number	Size (LOC)	Months used	Total defects	Test defects	Use defects
1	463	18	13	5	0
2	4,565	NA	69	10	0
3	1,571	NA	47	8	0
4	3,381	NA	69	22	0
5	5	9	0	0	0
6	22	5	2	0	0
7	1	18	1	0	0
8	2,081	10	34	0	1
9	114	8	15	2	0
10	364	NA	29	2	0
11	7	5	0	0	0
12	620	3	12	2	0
13	720	NA	9	2	0
14	3,894	NA	20	2	0
15	2,075	NA	79	27	0
16	1,270	NA	20	1	0
17	467	NA	17	3	0
18	3,494	8	139	50	0
Total	**25,114**	**NA**	**575**	**136**	**1**

Table 4. US&S usage data.

Product	Lines of code	Months of use	Defects in test	Defects in use
M45	193	9.0	4	0
M10	453	7.5	2	0
M77	6,133	4.0	25	0
M54	477	3.5	5	0
M53	1,160	1.0	21	0
Total	**8,416**	**NA**	**57**	**0**

mine whether this was a latent problem with the prior code or a newly injected defect with this project.

UNION SWITCH & SIGNAL

Union Switch & Signal manufactures a wide range of hardware products for the railroad and transit industries. It also develops and installs software-intensive process-control systems for real-time control of railroad and transit operations. The Automation and Information Systems business unit of the US&S Engineering Division is located in Pittsburgh and has approximately 100 software managers and engineers developing software and hardware for railroad and transit control centers.

US&S has given three PSP classes to nine managers and 25 engineers. The managers' class was held during regular work hours, but managers had to complete homework assignments on their own time. This turned out to be too heavy a workload, and only about half the managers completed all the work. Subsequently, the engineers' classes have been given during work hours, and a full day is provided to complete each of the 10 programming exercises. This has proved adequate as long as the managers track the engineers' progress and encourage them to finish the work. To date, 72 percent of the engineers have completed the course assignments.

At US&S, PSP-trained engineers have completed five projects. All were maintenance and enhancement releases for a large railroad information and control system, and each project required only one engineer. Using PSP techniques, the engineers completed their projects on schedule, and no defects have been found in any project during installation or customer use.

Data on these five projects are shown in Table 4. As of the time of this writing, no defects have been found in any of these products.

The effective use of PSP depends on proper training. The three companies described here followed course plans similar to, though shorter than, the standard 15-week academic course, with one lecture given each week. Since its third class, AIS has offered a two-week course, with a lecture given in the morning and with the remainder of the day devoted to completing the exercises. The engineers use a third week for additional work. Either strategy can be effective as long as management uses qualified instructors, provides sufficient time for the engineers to complete the exercises, and monitors the training.

In both courses, engineers require about 125 hours to do the work. In the most successful classes, management has provided a full working day for each lecture and assignment, and the engineers have been willing to spend some personal time studying. Unfortunately, it has been found that some engineers fail to do the assignments, even when work time is provided. Because the exercises are critical for learning PSP, it is important that managers treat PSP training as part of the engineers' jobs and monitor their progress.

After learning PSP, engineers require some discipline to continue following PSP methods and procedures. Management needs to constantly stress the importance of quality goals, of thorough planning, and of effective tracking. A US&S manager, for example, strongly supports PSP and holds weekly meetings with his project team to review status, plans, and data. Another manager, however, has not emphasized the importance of PSP, and his engineers have essentially reverted to their old practices.

With proper training and with continuing management interest, PSP is an effective methodology for efficiently developing quality software. Although extensive data are not yet available, the case studies here show that PSP can improve planning and scheduling, reduce development time, and produce better software. Of the PSP projects reported to date, all have been delivered on or ahead of schedule and only one has had any customer-reported defects. PSP also has been found to accelerate an organization's CMM process improvement efforts.

As the effectiveness of PSP becomes evident, interest in the program is growing. SEI continues to introduce PSP to the industry, offering presentations, training, and on-site consultations. The institute has also developed several training programs complementary to PSP. One is designed for managers who supervise PSP-trained engineers. Another, the Team Software Process, now in early development, extends PSP beyond the individual engineer to the software development team.

These efforts—for management, engineers, and software teams—will help companies consistently and efficiently produce high-quality products. At a time of growing competition, when engineers are under increasing pressure to quickly produce error-free products, software companies can little afford to ignore better ways of doing their work. ❖

Acknowledgments

We thank the many people who have participated in this work, as well as our associates who reviewed and commented on this article. From AIS, we particularly thank Girish Seshagiri, Vikas Khanna, Gloria Leman, Srikanth Nallapareddy, Bob Pauwels, and Prasad Perini. From Motorola, we especially thank John Wirth, John Pange, Efrain Nieto, Kamran Nili, Jeff New, and Jed Coxon. At ERAU, Iraj Hirmanpour was a great help. From US&S, we recognize the help of Julia Mullaney, Nadine Bounds, Bob Elder, Linda Falcione, Ron Morton, and John Staub. From SEI, we thank Andy Huber, Dan Roy, and Bill Peterson.

This work was supported by the US Department of Defense.

References

1. D.R. Goldenson and J.D. Herbsleb, *After the Appraisal: A Systematic Survey of Process Improvement, Its Benefits, and Factors that Influence Success*, Tech. Report CMU/SEI-95-TR-009, Software Eng. Inst., Pittsburgh, 1995.
2. W. Hayes and D. Zubrow, *Moving On Up: Data and Experience Doing CMM-Based Process Improvement*, Tech. Report CMU/SEI-95-TR-008, Software Eng. Inst., Pittsburgh, 1995.
3. M.C. Paulk et al., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, Reading, Mass., 1995.
4. W.S. Humphrey, *A Discipline for Software Engineering*, Addison-Wesley, Reading, Mass., 1995.
5. W.S. Humphrey, *Introduction to the Personal Software Process*, Addison-Wesley, Reading, Mass., 1997.
6. B.W. Boehm, "A Spiral Model of Software Development and Enhancement, " *Computer*, May 1988, pp. 61–72.
7. W.S. Humphrey, "A Personal Commitment to Quality," *Am. Programmer*, Apr. 1995, pp. 2–12.
8. W.S. Humphrey, "Using a Defined and Measured Personal Software Process," *IEEE Software*, May 1996, pp. 77–88.
9. W.S. Humphrey, "The Power of Personal Data," *Software Process Improvement and Practice*, Vol. 1, Issue 2, Dec. 1995, pp. 69–81.

Pat Ferguson has worked as a software engineer, as a project manager, and most recently as the development manager at Advanced Information Services, Inc. She currently serves on the steering committees of the Chicago Software Process Improvement Network and the Heartland SPIN. Ferguson received a BS in mathematics and an MS in computer science, both from Bradley University. She is a member of the IEEE Computer Society.

Watts S. Humphrey founded the Software Process Program at the Software Engineering Institute at Carnegie Mellon University. He is a fellow of the institute and is a research scientist on its staff. He has written many technical papers and six books, most recently A Discipline for Software Engineering *(Addison-Wesley, 1995),* Managing Technical People *(Addison-Wesley, 1996), and* Introduction to the Personal Software Process *(Addison-Wesley, 1997). He holds five US patents. Humphrey received a BS in physics from the University of Chicago, an MS in physics from the Illinois Institute of Technology, and an MBA from the University of Chicago. He is a member of ACM and a fellow of IEEE.*

Soheil Khajenoori is a professor and director of the Master of Software Engineering Program at Embry-Riddle Aeronautical University. He is currently working with Motorola Paging Products Group and McDonnell Douglas Space Division on the PSP. Khajenoori currently teaches courses on PSP, software requirements engineering, and software architecture and design. Khajenoori's research interests are in the areas of software development methodologies, software metrics, and software process engineering and improvement. Khajenoori received a PhD in computer engineering from the University of Central Florida. He is a member of IEEE.

Susan Macke is a manager at Motorola, leading the decoder and software engineering teams for the North American Paging Subscriber Division. Macke received a BS in computer science and operations research and processes from Lebanon Valley College.

Annette Matvya has 20 years of experience in software engineering at Union Switch & Signal Inc. and is currently a member of that company's Software Engineering Process Group. She has completed the personal software process course and is currently involved in implementing PSP at US&S. Matvya graduated from the University of Pittsburgh with a BS in computer science.

Further information on SEI's PSP offerings and activities can be found at http://www.sei.cmu.edu/technology/psp.

Contact Ferguson at patf@pjstar.com, Humphrey at watts@sei.cmu.edu, Khajenoori at soheil@ db.erau.edu, Macke at Susan_Macke-FMS010@ email.mot.com, and Matvya at almatvya@ switch.com.