A Distributed Multi-Company Software Project

Dr. William R. Nichols, Anita D. Carleton, Watts S. Humphrey, James W. Over
Software Engineering Institute

This article describes how two software development groups—in different locations and working for competing companies—were able to jointly develop a large and complex military systems product. Even with incompatible support systems and decentralized management, these groups were able to work cooperatively and deliver a quality product on schedule[1].

The DoD and almost all non-defense government agencies increasingly depend on large software-intensive systems. Unfortunately, with few exceptions, such large-system development projects are seriously troubled. While the programs may ultimately deliver workable systems, they are generally late, cost much more than planned, and some even fail completely. While guidelines such as CMMI provide useful guidance on what methods should be used, many studies have shown that seriously troubled large-scale development programs have one problem in common: They do not effectively follow the methods they currently know [1].

The two companies doing the work described in this article are the only suppliers the DoD has for a category of highly classified military equipment. These companies' development laboratories were in different cities and had very different engineering practices and support systems. This complicated the DoD's logistics and training systems, raised maintenance costs, and limited acquisition flexibility. The DoD had urged the companies to develop a common engineering design and support system, but the companies had resisted. Finally, the DoD issued a joint contract and directed the companies to develop a common system.

To meet this directive on schedule and within prescribed costs, the senior executives of both companies knew they needed a joint development team that had a level of coordination and communication that they had not previously been able to achieve. Their joint team would have approximately 30 people from two competing organizations with several different disciplines and with different processes and practices. To build such a capability, they turned to the SEI for guidance. The project described in this article is the result.

Project Goals and Strategy

The SEI recommended that the two organizations use the TSP to guide them in jointly establishing the project's goals and objectives, agreeing on the project management strategy, and launching and managing the development work. To start TSP introduction, the SEI had both management teams come to Pittsburgh for a one-day executive seminar and a half-day planning session [2]. The seminar described how the TSP team-building process melds people from different specialties and organizations into high-performing development teams with common goals, practices, processes, and plans. TSP introduction also requires that team leaders take a leadership course and that team members take Personal Software Process℠ (PSP℠) training [3].

Table 1: *The Process Steps in PSP Training*

PSP Steps	Program Number	New Process Concepts Introduced
PSP0	1	Students use their current process with two added measurements. They record time spent per process phase (planning, design, code, review, compile, unit test, and post-mortem). They define a defect type standard, and log all defects found in the review, compile, and unit-test phases.
PSP0.1	2, 3	Students define a coding standard and a line of code (LOC) counting standard, use process improvement proposals (PIPs), and start measuring the size of their programs in LOC.
PSP1	4	Students add defined size estimation methods and effort estimation methods to their personal process. Test reports are also introduced.
PSP1.1	5, 6	Task and schedule planning are introduced. Earned Value (EV) tracking is also introduced.
PSP2	7	Quality techniques are introduced. Structured personal code and design reviews, based on individual defect data, are conducted.
PSP2.1	8, 9, 10	Design templates and design verification methods are introduced.

Team Training

All of the software engineers were trained in common classes where the team members met their counterparts from the other company and completed a series of 10 programming assignments. Writing programs with the six PSP steps taught them to plan and track their work and to measure and manage product quality (see Table 1). The requirements team members were also trained in a single Introduction to Personal Process course that taught them how to plan and track their work.

The introduction strategy started by building self-directed teams that would plan, manage, and track their own work. To be on a self-directed team, however, the members had to learn how to manage themselves. PSP training showed them how to gather data, use that data to make accurate plans, precisely report project status, and manage product quality. After completing training, the team members could see from their personal data that these methods improved their performance (see Table 2).

The first column of Table 2 describes the measure, the second shows the class average value for the first three PSP programs, and the third column shows the class average value of the measure for the last three programs at the end of PSP training.

The TSP Project Launch

Once management and the team members were trained and the teams formed, the next step was to build self-directed teams. This was done in a five-day project launch where the members produced their own plans and negotiated their commitments with management and the customer [4].

℠ Personal Software Process and PSP are service marks of Carnegie Mellon University.

With the TSP, each development team and its team leader is called a unit team; the five-day multi-team launch for this project had three unit teams. Everyone was brought to one location and the leadership team was formed from the three unit team leaders and the two company project managers. It monitored launch progress and helped the unit teams coordinate their work and resolve issues. A TSP coach guided each unit team through the launch and a multi-team coach guided the leadership team and coordinated the overall launch process (see Figure 1).

The First Launch Meeting

In launch meeting 1, the three unit teams, the TSP coaches, and the leadership team met with senior management and customer representatives. The multi-team coach first described the launch process and agenda and two company executives described the product requirements and desired eight-month delivery date. The customer representatives next explained the DoD's reasons for needing a single engineering and manufacturing support system. At the end of meeting 1, all of the team members understood the job, had heard management's goals, and knew why the project was important to the customer. Following meeting 1, the unit teams, team leaders, and coaches met in separate groups with no observers or visitors to follow the launch process for meetings 2 through 8.

Launch Meetings 2 Through 8

In launch meeting 2, the unit teams each established team goals and the members each selected from among the eight standard TSP roles: customer interface manager, design manager, implementation manager, test manager, planning manager, process manager, quality manager, and support manager. By accepting a role or alternate role, each team member took responsibility for an aspect of the team's work. The test managers, for example, did not necessarily do the testing, but they were responsible for ensuring that testing issues were properly considered and addressed throughout the project. The teams decided that no additional roles were required and every team member except the team leader took at least one role or an alternate role.

In launch meetings 3 and 4, the teams each defined their unit team strategies, processes, and plans. They listed the products to be produced, estimated their sizes, and judged the time required for each process step. They then estimated the hours they would each spend on the project every week and generated the project schedule.

Measure	Start of Training	End of Training
Percent time spent in compile	10.8%	1.2%
Percent time spent in unit test	26.3%	10.4%
Compile defect density (number of defects found during compile per thousand LOC [KLOC])	54.0 defects/KLOC	11.7 defect/KLOC
Unit test defect density (number of defects found during unit test per KLOC)	32.0 defects/KLOC	9.5 defects KLOC
Yield (percentage of defects found before first compile)	2.6%	66.5%
Productivity (in LOC per hour)	33.9 LOC/hour	34.0 LOC/hr

Table 2: *Performance Improvement During PSP Training*

Once the unit teams had produced their task plans, they produced a quality plan in launch meeting 5. This included quality-tracking measures, review rates, defect injection and removal rates, yields, and defect levels. The result was an estimate for the defects to be injected and removed in each project phase, the defects in the product at system-test entry, the number of defects that would remain for customer acceptance testing, and the expected number of defects in the product at final delivery.

In meeting 6, the teams made detailed next-phase plans and balanced the workload between the teams and team members. This resulted in the most efficient workload allocation and the minimal project schedule. During meeting 7, they identified and ranked the major project risks and assigned team members to track and prepare mitigation plans for each key risk.

In meeting 8, the teams prepared for the management review in meeting 9. They had been unable to meet the requested eight-month schedule and their base plan delivered a minimum-function first release in 13 months with two subsequent releases at six-month intervals. They also prepared an alternate plan with added resources but, since no suitably skilled and PSP-trained professionals were available, it was considered impractical.

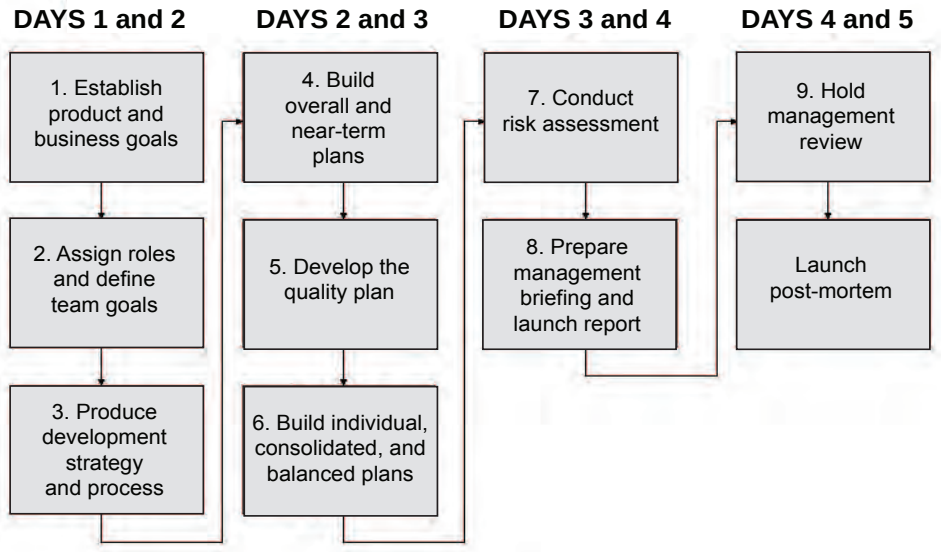
In the post-mortem step, the teams reviewed the launch process, submitted PIPs, and gathered and stored the launch data and materials.

By the morning of the fifth day, the unit team's plans were finished and consolidated into an overall project plan. Each team member had a personal plan, and the overall project had a plan that all team members had participated in producing and would defend as best they could with the available resources. The members were all committed to making the project a success and were excited about the challenges ahead.

Launch Meeting 9 With Management and the Customer

The DoD program manager was invited to

Figure 1: *The Standard TSP Unit-Team Launch Process and Meetings*



the final management meeting with all of the team members, senior management, team leaders, coaches, and involved support-group managers. The unit team leaders first described the planning process and the unit team plans. There was considerable discussion about the program schedule, the release contents, the delivery dates, and how the team had estimated the work.

Everyone was nervous about the program manager's reaction. While he had not said much during the discussion, they knew that he would be hard to convince. Both companies had a long record of missing software commitments and the developers felt that, unless he bought into this plan, they would have to continue with the same practices that had failed in the past. At the end of the plan presentation, the program manager said: "I am disappointed with the dates, but this is the best program plan I have ever seen." He congratulated them on producing a detailed and convincing plan with dates he felt he could count on. While he would have liked an earlier date, he felt it was much more important to have a solid date that he could use to make commitments to the DoD. They had won—now all they had to do was deliver! With a common plan they all believed in, they were excited and ready to tackle the challenges ahead.

Challenges During the Launch

The principal challenge during the launch was to ensure that the unit teams' plans fit together into an overall plan that minimized the program schedule. That required each unit team to identify its dependencies on the other teams and adjust its plan to minimize the overall project schedule. The leadership team met with the team leaders and the four TSP coaches every evening to guide this coordination work. The following three examples show how the TSP launch process helped the teams identify and resolve many of the problems that they would not normally have recognized until later in the program (when those problems would have caused serious delays).

1. The Requirements Schedule Problem. Even though the requirements team members were from different organizations and had a range of skills, they agreed on the team goals and roles in about three hours. Then they drafted a tentative requirements schedule and invited the two development team leaders to a review meeting. Each of the development team leaders had also prepared initial plans that did not mesh with the preliminary requirements plan. They needed some planned requirements items immediately so they could start high-level design work while other items could be deferred until detailed design or implementation. After about an hour, the development team leaders and the requirements team had produced an agreed-upon requirements priority. The requirements team then produced an initial requirements schedule that put the high-priority development items first and gave this plan to the development teams to use in producing their plans.

2. The Role-Manager Teams. Since many activities required a project-wide perspective, cross-team role-manager teams were formed of the role managers from the three unit teams. For example, the planning role-manager team tracked and coordinated the dependencies among the teams. Similarly, the test and quality role-manager teams ensured that testing and quality-management tasks were included in every development phase. The leadership team members mentored the role-manager teams and kept them working on key cross-team issues. For example, the design role-manager team's principal assignment was to ensure that all high-level design work remained consistent and to define common design methods and standards. The manager also ensured that each element of the product work breakdown structure was assigned to the unit team that was best able to handle it.

3. Integrating the Unit Team Plans. After the unit teams produced their plans on launch day 2, the leadership team found that one development team was planning on three releases and the other team on four. After considerable discussion, the leadership team agreed on a strategy with three releases. They also asked the design and customer-interface role-manager teams to produce and maintain a single official list of the functions planned for each release. The development teams then produced their detailed plans and schedules and these schedules were adjusted to synchronize the product release dates. The teams found they could only deliver the minimum required product function in 13 months rather than the desired eight months. The two follow-on releases would then complete the basic product function in 25 months.

Using the Process on the Project

Before the TSP, the managers had made the team plans. Because these were high-level plans, they did not provide the detail needed for precise project tracking, and management had never been able to get accurate project status information. The managers had been reluctant to delegate planning, tracking, and plan management responsibility to the teams; however, after the executive seminar, they agreed that the new training should enable the members to make the detailed plans needed to precisely plan, track, and manage their own work. The unit teams' detailed planning and tracking data enabled them to manage their resources, precisely report status, and get management help with problems before they delayed the project.

Every week, the teams reported planned and actual EV, the Cost Performance Index (CPI), and task hours to the leadership team. With these reports, the leadership team could precisely track and report project status to senior management; in turn, senior management could see if any milestones were exposed and identify and help resolve any problems. For example, achieving a milestone with less EV than planned generally meant that some quality steps had been skipped, while slipping milestones often indicated coordination problems or approval delays.

Part-Time Staffing

Part-time staff assignments were a particularly difficult challenge. While the unit teams had a number of part-time members, several were not even putting in the time they had committed to their teams. Because these part-time members often possessed special knowledge or skill, their work could not be shifted to others. The detailed TSP plans and effort tracking allowed the managers to see this problem in the first few weeks of the job. As the teams gathered data, it was soon evident that, when members planned to work less than half of their time, they often failed to work even that amount. Some critical and highly specialized skills were obtained on a part-time basis, but the team leaders found that many of the other part-time members were costing the project more money than their work was worth. As one team leader said, "I could trade three half-timers for one full-timer and come out ahead."

Using a Defined Process

Many people were initially skeptical about the value added by a defined process. But they soon saw that their previous informal process of circulating documents, marking them up, and having changes incorporated by the document owner did not scale up to a project with numerous stakeholders. Changes could conflict or be contentious, and baseline documents were difficult to identify. A member of the leadership team then created a simple

cross-team process for circulating baseline documents, collecting comments, and distributing the collected comments and revisions for a second review. This worked so well that an additional review was rarely needed. This experience demonstrated that a defined process would help to make the work more effective and efficient and would not add unnecessary bureaucracy.

Another key process element was standards development. Different stakeholders had different document needs. The requirements engineers used technical specifications to resolve analysis issues. Once a technical specification was approved by the customer, it was used by the programmers to design and implement the code. By establishing standards for content, format, and conventions, communication among stakeholders was improved, all needs were addressed, and the review scope could be narrowed. The programmers appreciated a consistent presentation and precise calculations while the engineers received fewer nuisance comments. Eventually, standards were developed for coding, configuration management, design representations, testing, and reports.

Process Improvement Proposals

When the teams realized that they could change the process, they started collecting process problems and analyzing causes before each re-launch. They evaluated their PIPs during re-launch preparation and made process changes before starting the next development cycle. The availability of the PIP mechanism enabled the team members to identify places where they found the process inconvenient or inefficient and then get those issues resolved. This improved process effectiveness and gave the team members a sense of process ownership.

The coaches also reviewed the teams' process fidelity at the seven-week point. They found that all teams were regularly collecting some of their data but that these data were not as complete or accurate as required. This finding suggested that further process changes, as well as some additional team member training and coaching, was needed. The planning managers also started taking a more active role. These actions improved data quality and enabled the team members to better manage their personal plans and to promptly modify their task lists to reflect work changes.

Estimating Improvement

After a year and several cycles of feedback and learning, team estimating accuracy had improved significantly. All teams were now gathering data and meeting and using team data to balance workloads. A team member who had been unable to plan and track at the previous review was now training new project members on planning and making commitments. Plans were more realistic and performance-to-plan was much improved. In the project's initial three-month checkpoint, the task hour plan-to-actual ratio improved from 1.09 to 0.96, and the CPI improved from 0.88 to 0.95 (where a value of 1 means on-time and on-cost performance, respectively). This means that instead of being 35 percent late, they were 3 percent early, and the CPI improved by 7 percent.

Use of Data

Later in the project, the teams started reviewing their data before each re-launch and analyzing test defects. Although the size and defect data were incomplete, the team could see the costs of finding and fixing defects. The most significant finding was that 10 percent of the defects were in the testing materials and another 10 percent were due to incorrect or misunderstood requirements. Coding defects seldom required more than an hour to find and fix while the non-coding defects averaged more than four hours each, and the test-case defects were often even more time-consuming. These and other findings led to improved requirements and design inspections as well as team and leadership workshops on topics shown by the data to need improvement.

Team Problems

Surprisingly, having a distributed team was not a problem for the requirements team. The members built trust by creating common documents, having weekly videoconference status meetings, arranging for regular face-to-face contact, and defining common team goals. The members felt that getting together every other month helped to keep the team jelled.

Even though the development teams were in one location, they initially had not become jelled or smoothly working units. The reason was that management had created an organizational structure with a collocated development team at each company location. This meant that each development team had responsibility for what had originally been thought to be a separate sub-product. After high-level design, however, these sub-products were found to have unforeseen dependencies that required the teams to share a large amount of common code. The teams had not planned for this level of interaction and found it difficult to coordinate their work. The team members had frequent disagreements and team cohesion was soon destroyed.

To solve these problems, the project members proposed an alternate team structure that retained the requirements team but restructured the two software development teams to have members from both companies and locations. Based on the experience of the requirements team, these teams were each given one tightly coupled component to develop and their members were geographically distributed. These restructured teams were established at the next re-launch where they each defined their processes and developed their plans. This structure produced two cohesive, productive, but distributed teams.

Another issue was that as the products matured through subsequent follow-on releases, they required more maintenance work as well as much more regression testing. The requirements team's role shifted from product specification to verification and validation, and the developers spent more time with the requirements engineers on test requirements and expected test results. This led to another project reorganization where the project was reformed into three fully integrated product teams with requirements engineers teamed with the software developers. Because the technical coupling of the work had changed, the new organization was needed so the requirements engineers and software developers could define common goals, share work priorities, have a comprehensive strategy, and agree to interim milestones.

Project Results

The planning, measurement, and quality-management skills the team members gained in PSP training enabled them to handle the large-systems challenges of this project (as shown in Table 2). Without such skills, the individual and team plans would have been incomplete and inaccurate and product quality would have been marginal at best. With accurate plans and consistently high product quality, the three project unit teams consistently met their commitments to each other and to management. As the teams learned to work collaboratively, they learned to anticipate and resolve problems, increasing their productivity and accelerating the work. They completed the originally committed product releases essentially on schedule, though some first-release functions were deferred. The customer was pleased and extended the project to include maintenance and follow-on development. The

organizations have continued to use the TSP process.

Conclusion

Prior to introducing the TSP, the two competing companies described in this article had been unable to produce the joint product for which they had contracted with the government customer. In fact, they had not even been able to agree on a plan for doing the development work. Following the introduction of the TSP, the companies formed a joint team and quickly produced a project plan that the government representative accepted. They then delivered a quality product on schedule.

By using the TSP's self-directed management style and forming integrated development teams, these competing organizations were able to build a collaborative working relationship and predictably produce quality results, even when working at distributed locations and having separate management reporting chains. This experience demonstrates the power of a properly defined, planned, measured, and quality-controlled process in helping organizations produce quality products on predictable schedules. It also shows that a properly formed, suitably trained team with common goals, processes, and plans will overcome the distrust and friction normal with teams from different organizations or separate locations. Without a defined, measured, planned, tracked, and quality-controlled process like the one used by these teams, large-scale distributed development teams have rarely been successful.◆

References

1. Hansen, Marc, and Robert F. Nesbit. Report of the Defense Science Board Task Force on Defense Software. Washington, D.C.: Office of the Under Secretary of Defense for Acquisition and Technology, 2000.
2. Humphrey, Watts S. Winning With Software: An Executive Strategy. Boston: Addison-Wesley, 2002.
3. Humphrey, Watts S. PSP: A Self-Improvement Process for Software Engineers. Boston: Addison-Wesley, 2005.
4. Humphrey, Watts S. TSP: Coaching Development Teams. Boston: Addison-Wesley, 2006.

Note

1. Since the work was highly classified, this article cannot name the application or development organizations.

About the Authors

William R. Nichols, Ph.D., joined the SEI in 2006 as a senior member of the technical staff and serves as a PSP instructor and coach with the TSP program. Prior to joining the SEI, Nichols led a software development team at the Bettis Laboratory near Pittsburgh where he developed and maintained nuclear engineering and scientific software for 14 years. His publication topics include the interaction patterns on software development teams, design and performance of a physics data acquisition system, analyses and results from a particle physics experiment, and algorithm development for use in neutron diffusion programs. He has a doctorate in physics from Carnegie Mellon University.

SEI
4500 Fifth AVE
Pittsburgh, PA 15213-2612
Phone: (412) 268-2727
Fax: (412) 268-5758
E-mail: wm@sei.cmu.edu

Anita D. Carleton is a senior member of the technical staff at the SEI. She has worked for more than 20 years on software process improvement, process measurement, and the TSP. Carleton helped to launch the software measurement initiative at the SEI in 1988. She is the co-author of "Measuring the Software Process: Statistical Process Control for Software Process Improvement." She was awarded with a commendation from Dr. Barry Boehm for her leadership in producing measurement definition frameworks for the DoD Software Action Plan. Carleton has a bachelor's degree in applied mathematics from Carnegie Mellon University and is a senior member of the IEEE Computer Society.

Phone: (412) 268-7718
Fax: (412) 268-5758
E-mail: adc@sei.cmu.edu

Watts S. Humphrey joined the SEI after his retirement from IBM. He established the SEI's Process Program and led development of the CMM for Software, the PSP, and the TSP. At IBM, he managed their commercial software development and was vice president of technical development. He is a fellow for the SEI, the Association of Computing Machinery, and the IEEE. He is also a past member of the Malcolm Baldrige National Quality Award Board of Examiners. In 2003, the President awarded Humphrey the prestigious National Medal of Technology for his contributions to the software engineering community. He holds master's degrees in physics and business administration.

Phone: (412) 268-6379
Fax: (412) 268-5758
E-mail: watts@sei.cmu.edu

James W. Over, who has been with the SEI since 1987, is manager of the TSP and is a senior member of the technical staff for the Software Engineering Process Management Program. Both in the United States and abroad, Over has led the SEI's work transitioning the TSP into several hundred organizations. He received an award from the Boeing Corporation for innovation and leadership in software process improvement. He has 35 years of technical and management experience in the software engineering industry. Over is the co-author of several SEI publications on software process definition and improvement.

Phone: (412) 268-7624
Fax: (412) 268-5758
E-mail: jwo@sei.cmu.edu