Software Engineering Institute

Deploying TSP on a National Scale: An Experience Report from Pilot Projects in Mexico

William R. Nichols
Rafael Salazar

March 2009

TECHNICAL REPORT
CMU/SEI-2009-TR-011
ESC-TR-2009-011

Software Engineering Process Management

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications section of our website (http://www.sei.cmu.edu/publications/).

Table of Contents

List of Figures

List of Tables

Acknowledgments

Rafael Salazar thanks the following individuals for their significant contributions. From Tec de Monterrey: Fernando Jaimes for his enthusiastic work in making the Mexican TSP Initiative a reality; Olivia Barrón, Iliana Ramírez, Juan Antonio Vega, and Antonio Mejorado for all the hours we have passed together teaching and coaching students and industry engineers; David Garza and Rodolfo Castelló for their support in establishing the TSP Initiative within Tec de Monterrey. From the SEI: Watts Humphrey for believing in us, for encouraging and guiding our work, and for all his time spent helping us to advance the Mexican TSP Initiative; Jim Over for all his support, help, and time spent paving the road for a successful implementation of TSP at a national level; Phillip Miller for his invaluable help to bring the SEI and Tec de Monterrey to work together; Julia Mullaney, William Nichols, Jim McHale, David Scherb, Gene Miluk, and Allan Willet for all their time helping us in the projects. From the Mexican Government: Sergio Carrera and Ivette García for all their support, help, and time spent in establishing the TSP as a National Initiative. From the Mexican Software Industry: Gerardo López for his passion and help to convince his peers in the industry to make Mexico No. 1 in software quality in the world through the use of TSP; Blanca Treviño, Héctor González, Ignacio Sánchez, Gonzalo Soto, Francisco and Jaime Alemán, and Omar Flores for sponsoring the TSP implementation within their organizations; Agustín de la Maza, Ricardo Garza, Hilda Alonso, Juan Valera, Eduardo Olivares, and many other PSP instructors, TSP coaches, team leaders, software engineers, and undergraduate students who are too numerous to name for their contribution, support, and enthusiasm to move forward the Mexican TSP Initiative within their organizations and teams.

William Nichols gratefully acknowledges the substantial contributions to this report. From the SEI, it began with the efforts of Jim Over and Julia Mullaney. Jim McHale and Phil Miller provided the earliest training courses in Mexico, coached teams, and provided data. Juan Antonio Vega's effort and hospitality along with the staff at EISEI at Tec's Guadalajara Campus made my first experiences in Mexico productive and memorable. In Monterrey, Fernando Jaimes has been a tireless sponsor for the initiative. Gene Miluk, Phil Miller, Bill Peterson of the SEI and Fernando Jaimes of Tec reviewed drafts of this report and provided valuable comments. I also thank Ivette Garcia from the Ministry of Economy both for her support of the initiative and review. All involved in the PSP and TSP classes and all members of the project teams who collected project data have made the experience with the Mexican TSP initiative rewarding. I regret only that I cannot properly credit everyone because they too numerous. We could not have completed this without the editorial assistance from Gerald Miller from SEI Communications. Finally, I must give credit to my co-author's vision and dedication to our project and his country.

Executive Summary

Mexico is the United States' second-largest trade partner; however, the Mexican software industry does not yet compete effectively for a share of the U.S. software market. For example, Mexico's Program for the Software Industry Development (Prosoft) has reported that in 2007 India sold $30 billion in software services to the U.S. compared to $900 million sold by Mexico [Prosoft 2008]. As the market continues to grow, no single nation will be able to satisfy the market need. This provides an opportunity to increase Mexico's participation in this growing market. Mexico has a strategy to accomplish this.

The leaders in global software development outsourcing, India and China, have a cost advantage because of relatively low wages. Rather than compete with low developer wages, an alternative is to improve productivity and product quality. The Mexican government, in part through Prosoft, has launched an aggressive program to build a national reputation as a provider of information technology (IT) products and services. The initiative will develop competitive human capital, strengthen the local industry, enhance process capabilities, improve quality assurance, and promote strategic alliances with foreign companies. A key to this program is the introduction of the Team Software Process[SM] (TSP[SM]).

As a whole, the worldwide software industry needs to improve cost and schedule management, cycle time, and product quality. Improving performance in these areas and developing the workforce capability are important Prosoft goals. Previous reports document the success of TSP in producing high-quality products on time and within budget [McAndrews 2000, Davis 2003]. TSP operationally implements high-performing development processes. These processes are managed by trained individuals and teams.

Proper training is an essential aspect of TSP implementation. Developers undergo an intense training, either the two week course, PSP I and PSP II, or the new one week course, PSP Fundamentals with the second week, PSP Advanced coming at a later time. In the course they learn to measure, estimate, plan, and develop using sound principles. The training allows the developers to practice these skills with programming exercises. The improvement in product quality at the completion of training is both substantial and remarkably consistent. Others involved in project work and management are also trained to participate on or manage these teams.

These Mexican TSP pilot projects included nine projects teams from five organizations delivering software as an outsource product. This outsourcing group is distinct from projects that produce either a commercial or internal use software product. Typically, the outsourcing projects have less control of their software development strategies, time tables, and start dates. This proved to be a significant problem in the initial planning and training phase of TSP rollout.

Nevertheless, these projects delivered their products an average only 2 percent later than planned. The schedule error for these teams ranged from 27 percent earlier than planned to 45 percent late. This compares favorably with industry benchmark data, some of which show that more than half of all software projects were either more than 100 percent late or cancelled. Among the TSP pilots

[SM] Team Software Process and TSP are service marks of Carnegie Mellon University.

launched in Mexico none were cancelled and several projects had no defects in system or acceptance test.

In the following two tables, product and project results from the pilot teams are summarized and compared to both a TSP benchmark group and an industry benchmark [Davis 2003, SEL 1993, Humphrey 2002, Jones 1995a, Jones 1996, Jones 2000, Chaos 1994]. Unlike the benchmarks, all Mexican TSP projects are pilots. Several of these Mexican projects are very small or have teams that have only been trained through PSP Fundamentals.

Table 1: System Test Quality and Performance Project Metrics Comparison

Measure (TSP)	TSP Benchmark Projects Average and Range	Typical Projects Average	Mexican TSP Project Average and Range
System test defects (defects/KLOC)	0.4 0 to 0.9	15	1.7 0.0 to 6.8
Delivered defects (defects/KLOC)	0.06 0 to 0.2	7.5	0.5 0.0 to 2.2
System test effort (percent of total effort)	4% 2% to 7%	40%	5.93% 0.25% to 26.22%
System test schedule (percent of total duration)	18% 8% to 25%	40%	6.2% 2.1% to 26.2%
Duration of system test (days/KLOC)	0.5 0.2 to 0.8	NA	5.4 0.4 to 9.5
Failure COQ	17% 4% to 38%	50%	15.2% 1.6% to 29.4%

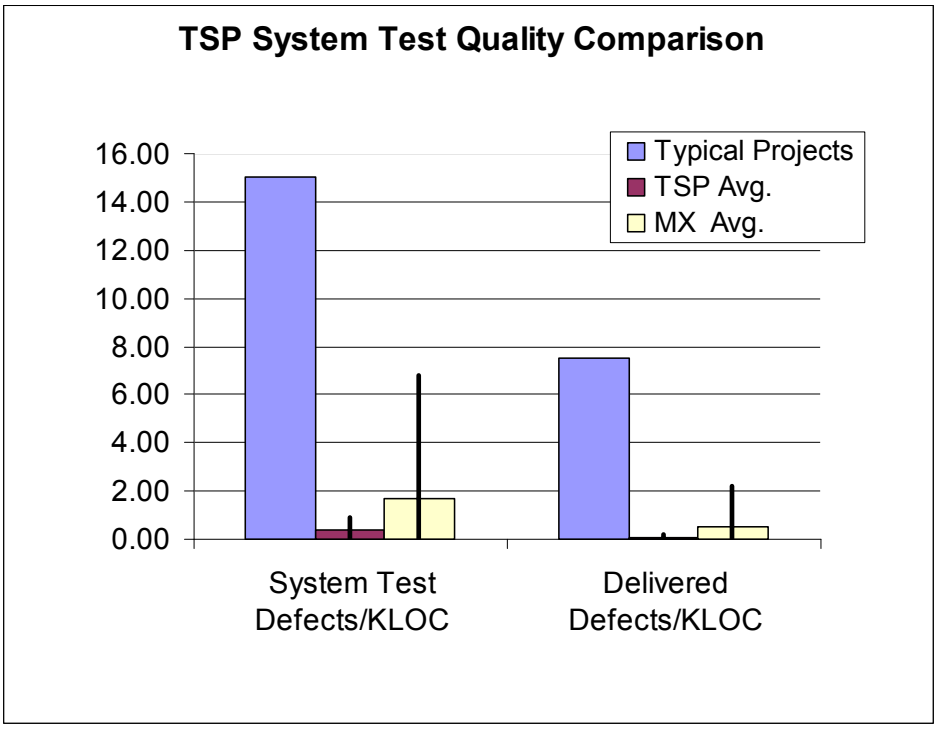

Figure 1: TSP System Test Quality Comparison

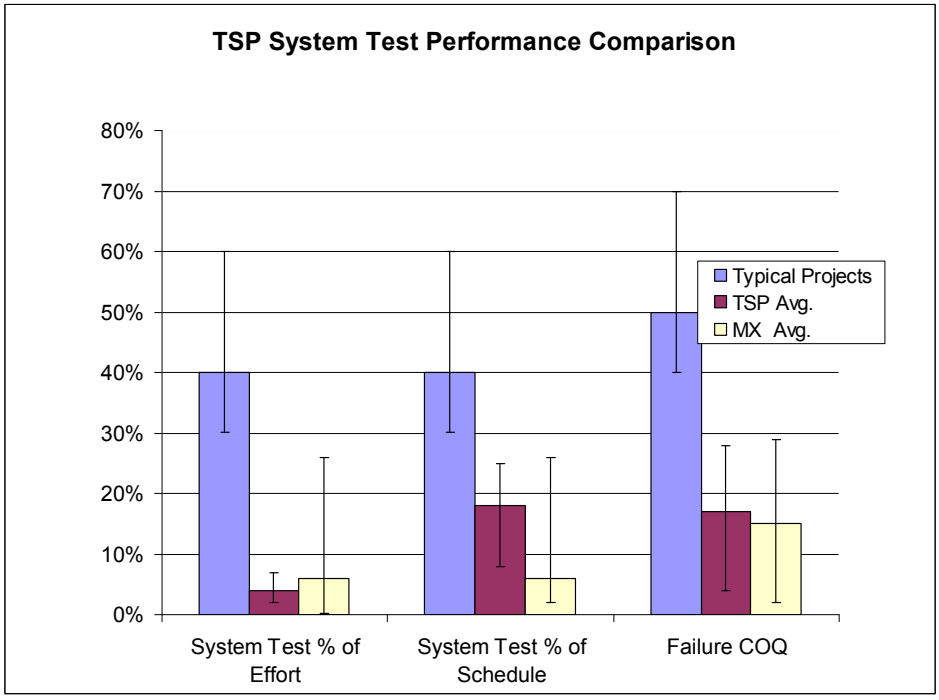

Figure 2: TSP System Test Performance Comparison

Table 2: Project Performance Metrics Comparison

Measure	**TSP Benchmark Projects Results 2003** **Average and Range**	**TSP Projects Mexico** **Average and Range**
Cost (effort) error	26% 5% to 65%	20% -9.5% to 54%
Schedule error	6% -20% to 27%	2% -27% to 45%

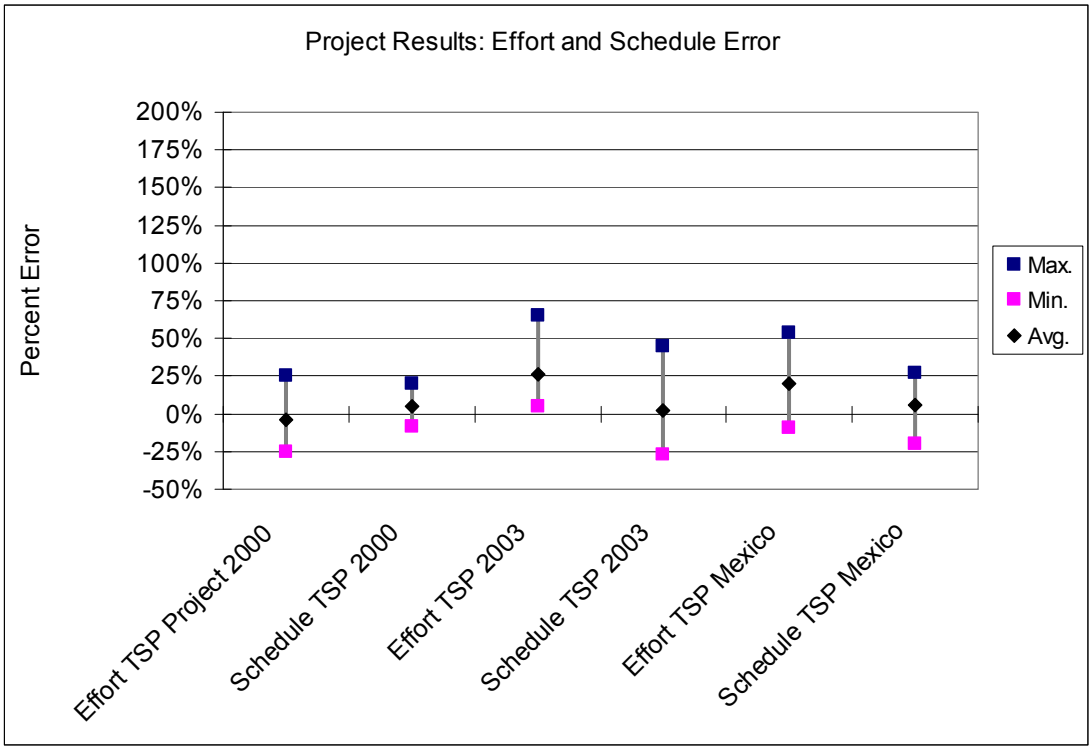

Figure 3: Project Cost (Effort) and Schedule Performance Results

Although the numbers are impressive enough, the development staff and management also speak positively about their experiences performing the work. Developers prefer the work environment of a TSP team. Management likes the depth of the data and the reliability of status reports. Worker attrition, a relative strength of Mexico, is not only maintained, but enhanced.

During the initial TSP rollout phases, a number of challenges are common. Problems include

- the up-front cost in both time and money
- resistance from the developers
- management support of self-directed teams
- appropriate use of the detailed data
- training and supporting high-quality TSP coaching

These problems are not unique to Mexico, but some are more relevant. A particularly important issue for Mexico is the number of small and medium sized enterprises (SMEs) that cannot afford the initial training. A new PSP training course, PSP Fundamentals, has been developed to reduce the time and cost required to launch teams.

TSP had been demonstrated to work for Mexican companies. Rolling out on a national level, however, is not only challenging, but unprecedented. In addition to the practical problems of the rollout, national success depends on visibility and recognition of the accomplishments. Next steps include

- training Mexican university professors so that they can train PSP developers in universities
- developing TSP as a cost effective way to implement CMMI®
- certifying and recognizing companies that effectively use TSP
- developing ways to train sufficient coaches and instructors to satisfy the nation's growing needs (scalability)

Training developers as part of their university education will significantly reduce the start up costs on the part of SME. This in turn requires trained university faculty.

TSP as a path to CMMI accomplishes two purposes. First, it will provide a cost-effective way to implement CMMI practices and evaluate maturity. The effectiveness of TSP in small settings will be especially helpful to the Mexican SMEs. Second, CMMI maturity ratings will provide widely respected recognition of Mexican commitment to process and quality. CMMI is recognized in the international market and a CMMI appraisal is required to enter this market. Because CMMI can be expensive and time consuming to implement, TSP accelerates implementation, reduces cost, and improves implementation quality. TSP does not replace CMMI, but rather implements many CMMI practices effectively and enhances CMMI effectiveness. Certifying organizations for TSP use will also support project goals in several ways. It will

- advertise both process commitment and actual results
- differentiate Mexican companies in the international market
- verify that Prosoft funds are appropriately spent

Future TSP application in Mexico will require solving the problems of scale. Scale problems affect most technologies as the use grows rapidly. TSP will need to develop methods of training sufficient numbers of developers, instructors, and coaches while maintaining high standards of quality. Another challenge will be introduction to small- and medium-sized enterprises where start-up costs must be minimized. Although TSP is less expensive than more conventional CMMI implementations and provides a positive return on investment, start-up costs are always a barrier.

® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Abstract

The purpose of this report is to communicate status, progress, lessons learned, and next steps for the Mexican Team Software Process (TSP) Initiative, a collaboration between the Software Engineering Institute (SEI) and Instituto Tecnológico y de Estudios Superiores de Monterrey (Tec de Monterrey), sponsored by the Mexican Ministry of Economy through Prosoft (the Program for the Software Industry Development). The initiative seeks to improve the standing of the Mexican software industry through the process and performance improvement benefits of the SEI's Team Software Process. We will discuss the results through Phase I and some early results from Phase II; performance results to date are compared to TSP and industry benchmarks.

1 Introduction

The purpose of this report is to provide status and results for the broad-scale national rollout of Team Software Process (TSP) in Mexico. The results include Phase I and early Phase II activities. This is provided for the project sponsors, Tecnológico de Monterey (Tec), and the Mexican Ministry of Economy, the sponsor of the Program for the Development of the Software Industry (Prosoft). This report should also be of interest to those organizations that participated in Phase I or Phase II, and industry and government organizations interested in participating in later phases of the rollout.

This report begins by describing the challenges and objectives that motivated this project. A summary of Personal Software Process (PSP)[SM] and TSP is included to provide context for the role of TSP in achieving these objectives. An overview of the long-term and near-term strategies is followed by a detailed description of the individual projects involved in Phase I. This is followed by an overall summary of performance results. This report concludes with lessons learned and how they can be applied, and the next steps of the project.

[SM] PSP is a service mark of Carnegie Mellon University.

2 Background

2.1 Project Motivation: Positioning Mexico's International Software Industry Using TSP/PSP

2.1.1 The Current Situation Of The Mexican Software Industry

Despite the growing interest and importance acquired by the electronics sector and software in the past decade, a 2001 study presented by the Ministry of Economy of Mexico showed that the sector experienced a loss of competitiveness (even disregarding an economic slowdown by the United States), which directly affects employment, competitiveness, investment and growth expectations.

Recognizing the importance of the electronics and software industries, the Mexican government launched a National Development Plan 2001-2006 to promote the industry and market information technology. The strategy was and continues to be to increase Mexico's competitiveness in all sectors through high-performing IT. A number of areas of opportunity were identified:

- Mexico lags in spending on software, 6 times less than the world average and 9 times lower than the U.S.
- similarities with successful models (Ireland, India and Singapore, among others), in which the software industry has led economic growth.
- Mexico could be attractive to investors, in large part because of the geographical proximity to the world's largest software market, (the United States), the network of commercial treaties, and familiarity with the culture of Western businesses.

To address this, the Ministry of Economy, in coordination with business organizations, educational institutions and business sector, has designed the Program for the Software Industry Development (Prosoft) [Prosoft 2008]. The program goals are as follows:

- increase annual software exports to the U.S. by $3.5 billion
- achieve the average global spending on information technologies
- position Mexico as the Latin American leader in software development and as the leading developer of Spanish language digital content

The strategies developed by the Mexican Ministry of Economy, in consensus with industry and other government agencies, include the following:

1. Promote exports and attract investment (ensuring that companies incursions into high value-added niches).
2. Educate and train competent personnel in high-quality software development. This involves training for engineers and technicians through highly specialized courses that are consistent with the needs of industry.

3. Strengthen the local industry through (1) the use of direct financing programs suited to their needs for working capital, (2) training, (3) the availability of venture capital, and (4) through government purchases to develop industry quality and to incubate new software businesses.
4. Achieve international standards in processes capability so that companies rely on the best international practices in the production of their systems. This requires standardization and certification agencies, as well as investment in training and research and development (R&D).
5. Promote the construction of basic infrastructure and telecommunications by developing high-tech parks associated with research centers.
6. Promote the harmonization of a modern legal framework that follows the international best practices while reinforcing trust and legal certainty among consumers and enterprises.

Ultimately, these strategies rely on training and R&D to achieve greater competitiveness of the software industry and, indirectly, the overall economy.

2.1.2 Problems to Solve

In comparison with companies in the global software industry, Mexican companies did not compete effectively in this market. Among the main causes were

- a lack of mature development processes that could be applied repeatedly in the software development life cycle
- extreme reluctance of small and medium-sized enterprises (SMEs) to take on large projects, thus limiting the number of small companies with export capacity
- a lack of the necessary human resources with sufficient expertise; improving the quality of software production requires time and investment in education and training
- infrastructure costs; to improve competitiveness, companies needed to adopt software engineering practices that reduce these costs
- lack of a business strategy to address the starting barriers for adoption of TSP in the industry of software development services
- international companies tended to contract large projects to large companies rather than SMEs
- SMEs were reluctant to take on large projects because of capacity and growth implications

To address these problems, Mexican companies have been introducing or acquiring the process capability and maturity needed to equal or exceed quality levels of the international competition. Unfortunately many of the maturity models are difficult to interpret and are descriptive rather than prescriptive. Descriptive models usually comprise a list prescribing what must be done rather than how to do it. This makes it difficult to translate this description into operational practices that can then be institutionalized.

The challenge, then, was to find and use software development quality models applicable to SMEs and to disseminate these models to the entire Mexican SME sector. In this way, they could become more competitive and develop the export capacity to serve the global software market .

2.2 Innovative and Technological Content

With respect to delivering within budgeted cost, on time, and with high overall product quality, the global software industry now has little credibility. This is fundamentally a problem of managing the development process. Consistent, predictable development of quality products does not happen by accident, and certainly cannot happen consistently when processes are poorly defined.

The Capability Maturity Model® Integration (CMMI®) was conceived as a model to identify process capability and guide process improvement in organizations. This process model has helped many organizations to develop quality products in a predictable and consistent manner. It is, however, difficult and complex to implement for SMEs. SMEs often lack the organizational infrastructure needed to implement and track CMMI-based improvement. Moreover, CMMI describes what should be done rather than providing an operational implementation.

Mindful of this challenge with CMMI, the original creator of the CMM—Watts Humphrey of the Software Engineering Institute (SEI)—has devoted the past few years to developing and implementing a congruent model scaled to small working groups that may even be applied at a personal level. The Team Software Process (TSP) implements a mature development process based on the individual and team practices of CMMI. TSP is applicable to the work team directly responsible for carrying out the software development project. The TSP assumes that all members of the development team know and apply the Personal Software Process (PSP). Organizations implementing TSP/PSP in developing their products effectively implement or support many of the best practices of the CMMI.

The TSP and PSP include new technologies designed to improve the performance of software development organizations using processes developed by the SEI. Organizations using TSP and PSP routinely report they [Davis 2003]

1. reduce the time to reach the market
2. increase productivity
3. reduce costs and improving compliance with the times estimated
4. improve product quality
5. accelerate the improvement of processes
6. reduce staff turnover

Results from organizations that have implemented the TSP/PSP have shown that they experience the following [Davis 2003]:

1. substantially improved product quality: an improvement of 2 to 20 times in the quality of the product delivered
2. software products delivered with an average of 60 defects per million lines of code (in many instances the products were delivered with zero defects), a substantial improvement over the 1,005 defects per million that delivered an average of organizations that are at Level 5 of CMM (see Figure 4) [Davis 2003].

® Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

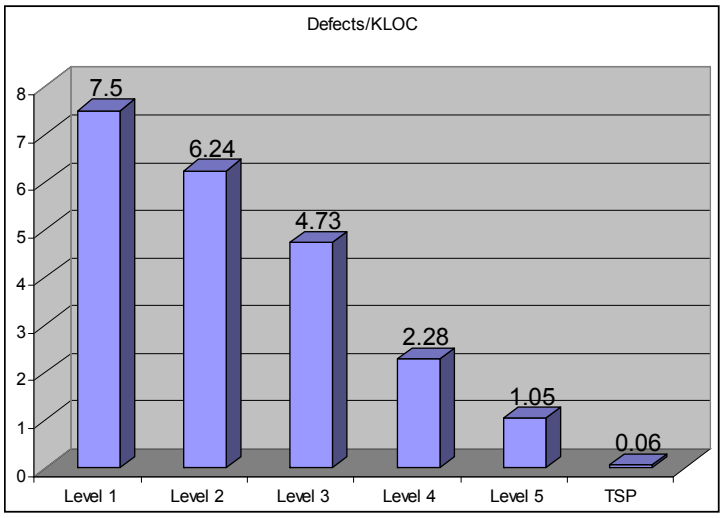

Figure 4: Average Defect Densities at CMM Maturity Levels

3. a significant reduction of the test cycle in particular (a reduction in the testing phase by a factor of between 2 and 10) and therefore in the complete software development cycle
4. improved productivity: the average project has improved team productivity by 78 percent
5. improved accuracy of estimates: the estimated cost and time have an accuracy of a +/- of 10 percent (see figure below) [Davis, 2003].

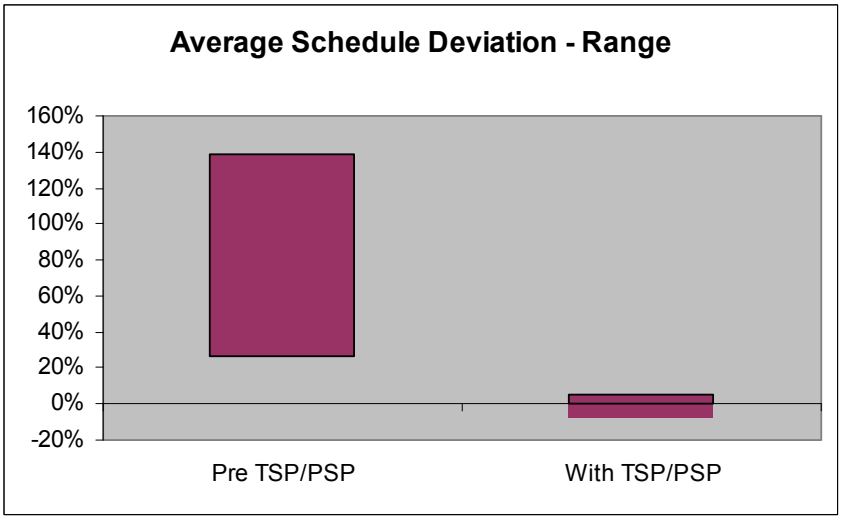

Figure 5: Average Schedule Deviations Before and After TSP

In addition to these benefits, an organization which has adopted the TSP/PSP at the organizational level is close to satisfying the requirements for CMMI maturity appraisals. This is because TSP and CMMI are based on the same principles. Wall shows the percentage of practices that are covered with TSP/PSP (see Figure 6) [Wall 2004].

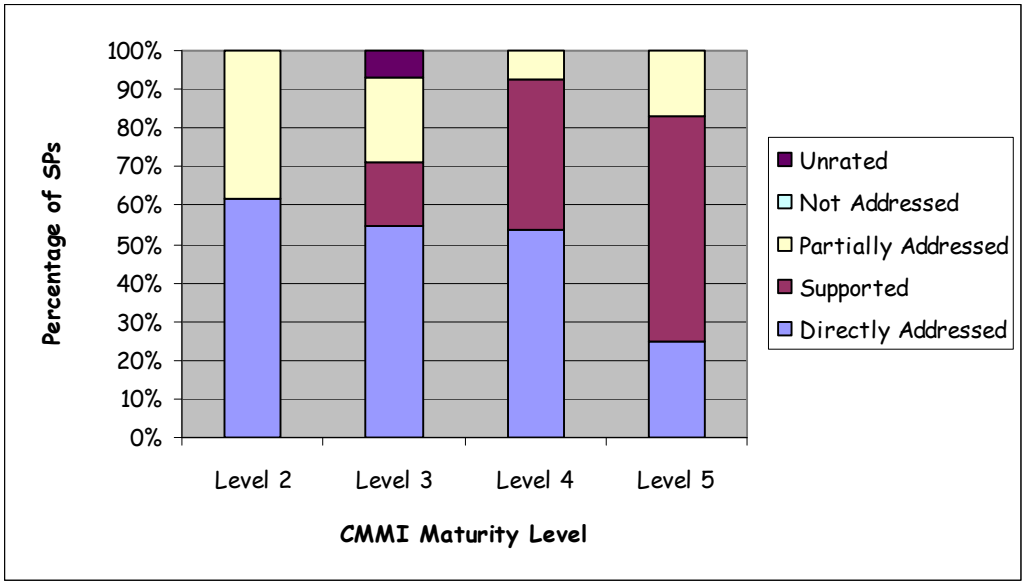

Figure 6: TSP Coverage of CMMI Specific Practices by Maturity Level

There is also evidence that organizations at Level 5 of CMMI benefit from the implementation of TSP/PSP. Individuals can now achieve certification in PSP (since March 2006) and demonstrate their contractors who have the skills required to develop high-quality software.

2.3 The Opportunity for Mexico

Just as Japan took the opportunity presented to capitalize on the initial work of the proponents of quality (e.g., W. Edwards Deming and Joseph M. Juran) to obtain a great competitive advantage, Mexico has a unique opportunity to build a technology base suited to its aspiration to take a technological leap ahead of competitors. There are multiple factors that support this opportunity. They are as follows:

1. The vast majority of companies that develop software in Mexico are SMEs.
2. The model used by Mexico's competitors (CMMI) is widely recognized and, while appropriate for large organizations, is complex and challenging to implement by SMEs. TSP makes CMMI practical for Mexican SMEs.
3. The Ministry of Economy, in consensus with industry and government agencies related to the sector, is committed to the development of the software industry in Mexico.
4. The TSP/PSP model developed at the SEI is an effective fit for the scale of SMEs.
5. The TSP/PSP, when implemented properly, has been demonstrated to be effective in greatly improving the performance of organizations. Organizations using the CMMI have reported significant performance improvement when TSP/PSP was added.
6. The TSP/PSP is in an early phase of adoption, and it is not known to the vast majority of organizations abroad. Mexican companies could take advantage of the opportunity to leapfrog the competition.
7. The SEI is seeking partners to help support the use of TSP/PSP, particularly in small and medium enterprises.

8. The SEI is transferring to the Instituto Tecnológico y de Estudios Superiores de Monterrey (Tec de Monterrey) the ability to successfully introduce TSP to Mexican companies, some of which will also seek CMMI maturity appraisals and ratings.

2.4 Objectives of This Project

The project began in 2006. The overall objectives of the project were

- promote exports from the Mexican software development industry through substantial improvement in the quality of products and by improving adherence to the agreed schedule dates of development projects
- strengthen the Mexican software development industry, focusing on the export sector by significantly improving productivity and accelerating the time the achievement of international standards in capacity processes
- educate and train qualified personnel in software development to improve both quality and productivity

The following specific objectives are used to support achieving these broad objectives:

- transfer the technology developed by the SEI for education and consulting models of TSP and PSP to Tec de Monterrey
- develop the capacity at Tec de Monterrey and through other business partners to develop the software industry in Mexico, especially in SMEs
- through the use of TSP/PSP, generate the ability to develop software with quality standards higher than those achieved with the CMMI
- educate students in academic programs related to software development in the TSP/PSP skills so that they can successfully apply them after their graduation
- develop joint research with the SEI for improved alternatives to support high-quality software development in SMEs, both by the transfer of TSP/PSP and by improving the methodology

The strategy for achieving these objectives and supporting the acceleration model proposed by Prosoft (see Figure 7), is to work with both the exporting companies that already have appropriate certifications for export (tier 1 companies), as well as with companies that provide services for project development (tier 2 businesses).

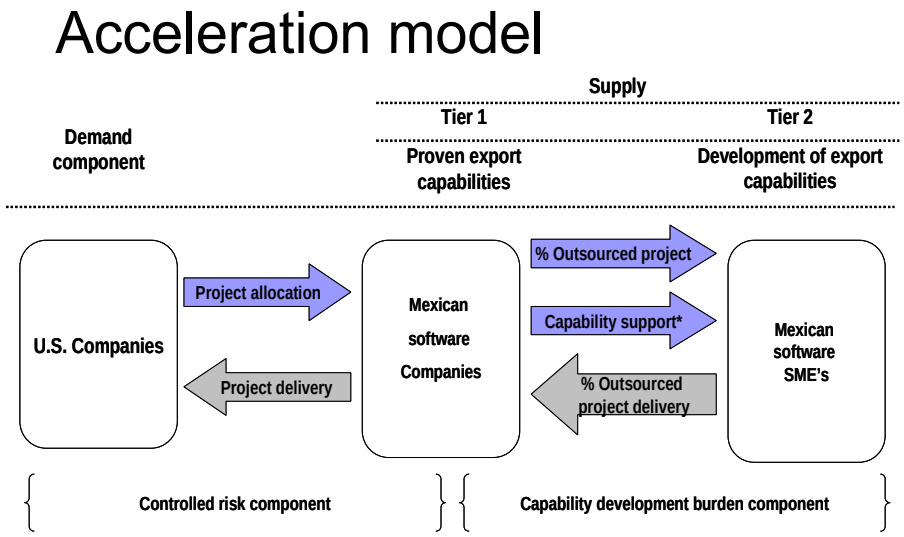

Figure 7: Acceleration Model

The direct beneficiaries of this project are exporting companies (tier 1) and their local development suppliers (tier 2). They will receive, at a more accessible cost, high-level training and implementation of a quality system using the TSP/PSP model. They not only will be more effective and, therefore, competitive on the world market, but also improve their professional image and facilitate their access to the demanding and globalized market.

Also benefiting directly are individuals trained in PSP by their companies, in seminars open to the public, and academic programs. After the technology transfer, Tec de Monterrey can guide and help other business partners and educational institutions disseminate these software engineering skills to the Mexican software development community. These business partners and educational institutions will therefore be indirect beneficiaries.

2.5 Achieving the Objectives

The SEI already had a proven methodology for introducing the practice of TSP/PSP to organizations. To meet the goal of promoting the practice of TSP/PSP in the Mexican software industry, Tec proposed a strategic approach. The SEI would empower Tec to act in place of the SEI in the initiative. The SEI agreed to train, authorize, and certify professors of Tec de Monterrey to introduce the methodology to Mexico via the following three phases.

Phase 1: Initiating the Technology Transfer, 2006 to 2007

The SEI, with the participation of Tec de Monterrey, introduced TSP to two Mexican companies. The organizations selected were exporting companies (tier 1) and within each of them were to be selected two software development projects which will implement the TSP/PSP.

Phase 2: Ensuring the technology transfer, 2007 to 2008

Tec de Monterrey (now with trained experts) carried out the introductions to tier 1 and tier 2 enterprises while the SEI oversaw and certified the Tec-led introductions. The training sessions of this phase incorporated the start of training new teachers/consultants from Tec de Monterrey.

Phase 3: Disseminate the technology transfer, 2009 and beyond

Tec de Monterrey carries out consultancy at organizations where they want to introduce the practice of TSP/PSP. During this phase the training of business partners begins.

Prior to Phase 1, two professors of Tec de Monterrey traveled in 2006 to the SEI in Pittsburgh for the training required to implement the TSP/PSP. The four courses (each with a duration of five days) is a prerequisite for a person to be authorized by the SEI to implement the TSP/PSP in an organization. The required courses are:

- PSP for Engineers - Part I: Planning
- PSP for Engineers - Part II: Quality
- PSP Instructor Training
- TSP Launch Coach Training

After completion of these courses, the SEI requires that the person be observed launching a TSP team.

The goal to educate students in academic programs suggests the following activities in each of the phases:

Phase 1: The SEI offered an "Academic Workshop TSP/PSP" the Tecnológico de Monterrey in July 2006. This workshop trained 10 teachers at Tec on how to teach these concepts in their classes.

Phase 2: Incorporate the concepts of TSP/PSP into classes of Tec de Monterrey and offer more academic TSP/PSP workshops open to other educational institutions (started in August 2006).

Phase 3: Begin the process of supporting other educational institutions wishing to incorporate the concepts of TSP/PSP in their classes (started in 2008).

The project also includes joint research projects between the SEI and Tec to measure the effectiveness of TSP/PSP innovation and improve the mechanisms for implementing these practices in SMEs, both in the transfer of TSP/PSP and in improving the TSP methodology.

2.6 Examples of Activities That Are Carried Out in Each Organization

Activities that are carried out in each organization vary depending on their own merits, but typically are as follows:

1. TSP Executive Seminar: This introduces TSP planning concepts and transition strategy to senior executives of the organization. This activity is required for a successful project. The requirements for successful implementation are explained. In return the executives explain their criteria for a successful TSP/PSP introduction. Duration: 1.5 days.

2. Training of all personnel involved in the selected pilot projects.
    a. The course "TSP/PSP for Managers" is required for all managers of the areas involved in the selected pilot projects.
    b. The course "PSP For Engineers" is required for all software engineers involved in the pilot project selected.
3. Launch. Each project has several launch sessions and a project completion (post-mortem) led by a coach TSP certified by the SEI.
    a. Coaching. To ensure the success of the project, throughout the duration of the project each team receives support from a TSP coach certified by the SEI.
4. Transition Planning. Provide the flexibility to develop the plan required to make an effective transition of TSP to the rest of the organization. Duration: variable.
5. Quarterly progress reports to executives of the organization.
6. Training of internal instructors and coaches.

2.7 Expected Project Results

At the end of the first two phases of this project the following results were expected:

- four exporting companies (tier 1), with 10 people in each trained in TSP/PSP and a plan for rollout in the rest of the organization
- four SME suppliers (tier 2), with 10 people in each trained in TSP/PSP and a plan for rollout in the rest of the organization
- two professors of Tec de Monterrey authorized by the SEI to train in PSP and TSP and be coaches, each with the ability to simultaneously respond to four projects.

In addition, the following were seen as possible, but the costs have not yet been quantified:

- at least two other tier 1 and two other tier 2 organizations (each with 10 people trained) in the process of implementation of TSP/PSP
- at least four additional teachers of the Tec de Monterrey undergoing training to become instructors and coaches in PSP/TSP
- at least 15 teachers trained to include the concepts of TSP/PSP in their classes

The remainder of this report will describe an example project, status of this plan, actual results obtained from participating organizations, the lessons learned that can inform the next steps, and the transfer of TSP to Tec de Monterrey as a SEI Strategic Partner.

3 Team Software Process Overview

The Team Software Process guides self-directed teams in addressing critical business needs of better cost and schedule management, effective quality management, and cycle-time reduction. It defines a whole product framework of customizable processes and an introduction strategy that includes building management sponsorship, training for managers and engineers, automated tool support, coaching, and mentoring.

The TSP can be used for all aspects of software development: requirements elicitation and definition, design, development, test, and maintenance. The TSP can support multi-disciplinary teams that range in size from two engineers to more than a hundred engineers. It can be used to develop various kinds of products ranging from real-time embedded control systems to commercial desktop client-server applications.

The TSP builds on and enables the Personal Software Process. The PSP guides individual software developers in following defined processes, gathering and using data, planning and tracking projects, managing quality, and continuously improving their performance. The PSP guides individual work. The TSP guides teamwork and creates an environment in which individuals can excel. Data from early pilots show that the TSP has been successful in addressing critical business needs [Ferguson 1999, McAndrews 2000].

3.1 Technology Description

3.1.1 History

In the 1980s, Watts Humphrey guided the development of the Capability Maturity Model for Software (CMM-SW). An early criticism leveled against the CMM was that it did not apply to small organizations. To address that criticism, Humphrey took on the challenge to apply the CMM to the smallest organization possible: an organization of a single individual. From 1989 to 1993, Humphrey wrote more than 60 programs and more than 25,000 lines of code (LOC) as he defined, documented, and evolved the PSP. He subsequently worked on corporate and academic methods to train others to use the PSP technology.

In 1997, a study was conducted to analyze the effect of PSP training on 298 software engineers [Hayes 1997]. This study found that effort estimates improved by a factor of 1.75, size estimates improved by a factor of 2.5, defects found at unit test improved 2.5 times and the percentage of defects found before compile increased by 50 percent. Students were able to achieve these improvements without adversely affecting their productivity. In terms of product quality and schedule variance, individuals are able to perform at a level one would expect from a CMM level 5 organization [Ferguson 1999].

The 1997 study was recently replicated with a much larger data set of about 1,000 software engineers. The results are essentially the same as the original study, with some minor differences. The results are presented in Appendix A of this report.

As engineers started applying their PSP skills on the job, it was soon discovered that they needed a supportive environment that recognized and rewarded sound engineering methods. In many or-

ganizations, the projects in crisis, and the individuals extinguishing the fires, receive all the attention. Projects and individuals who meet commitments and do not have quality problems often go unnoticed. Humphrey found that if managers do not provide a supportive environment, and ask for and constructively use PSP data, engineers soon stop using the PSP. Humphrey then developed the Team Software Process to build and sustain effective teams.

3.1.2 How PSP and TSP Work

Typical software projects are often late, over budget, of poor quality, and difficult to track. Engineers often have unrealistic schedules dictated to them and are kept in the dark as to the business objectives and customer needs. They are required to use imposed processes, tools, and standards and often take technical shortcuts to meet unrealistic schedules. Very few teams can consistently be successful in this environment. As software systems get larger and more complex, these problems will only get worse.

The best projects are an artful balance of conflicting forces. They must consider business needs, technical capability, and customer desires. Slighting any facet can jeopardize the success of the project. To balance these conflicting forces, teams must understand the complete context for their projects. This requires self-directed teams that understand business and product goals. They

- produce their own plans to address those goals
- make their own commitments
- direct their own projects
- consistently use the methods and processes they select
- manage quality

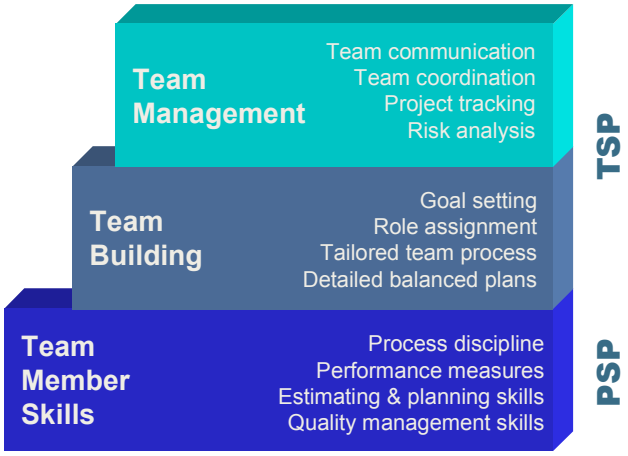

Figure 8: How the PSP And The TSP Work

Self-directed teams start with skilled and capable team members. Each instruction of a software module is handcrafted by an individual software engineer. The engineer's skills, discipline, and commitment govern the quality of that module and the schedule on which that module is produced.

A software product is a team effort. Software products are composed of software modules. These modules are designed, built, integrated, tested, and maintained by a team of software engineers. The team's skill, discipline, and commitment govern the success of the project.

The objective of the PSP is to put software professionals in charge of their work and to make them feel personally responsible for the quality of the products they produce. The objective of the TSP is to provide a team environment that supports PSP work and to build and maintain a self-directed team.

3.1.3 The PSP

The PSP is based on the following planning and quality principles:

- Every engineer is different; to be most effective, engineers must plan their work and they must base their plans on personal data.
- To consistently improve their performance, engineers must personally use well-defined and measured processes.
- To produce quality products, engineers must feel personally responsible for the quality of their products. Superior products are not produced by mistake; engineers must strive to do quality work.
- It costs less to find and fix defects earlier in a process than later.
- It is more efficient to prevent defects than to find and fix them.
- The right way is always the fastest and cheapest way to do a job.

Today, most software engineers do not plan and track their work nor do they measure and manage product quality. This is not surprising since engineers are neither trained in these disciplines nor are they required to use them. The dilemma is that until they try them, most software engineers do not believe that disciplined methods will work for them. They won't try these methods without evidence and they can't get the evidence without trying the methods. The PSP addresses this dilemma by putting an engineer in a course environment to learn these methods. The engineers use these methods in the course and can see from their personal and class data that these methods work for them.

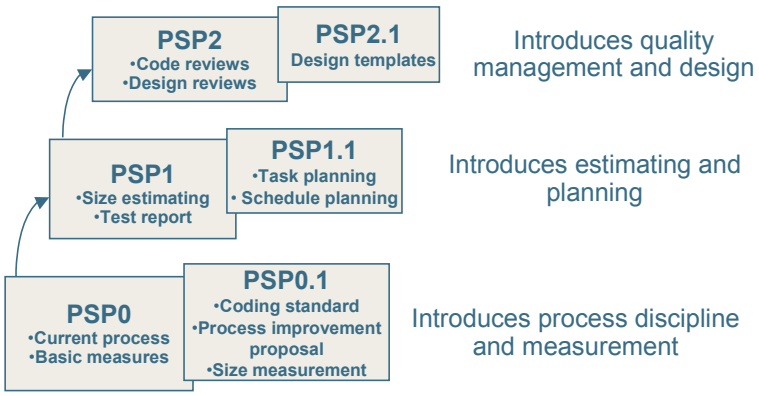

Figure 9: The PSP Course

The PSP course is composed of ten programming assignments and five reports. The process is introduced in six upwardly compatible steps. The students write one or two programs at each step and gather and analyze data on their work. They then use their data and analyses to improve their work.

PSP0 and PSP0.1
Students write three programming assignments using PSP0 and PSP0.1. The objective is for the student to learn how to follow a defined process and gather basic size, time, and defect data.

PSP1 and PSP1.1
Once students have some historical data, the focus moves to estimating and planning. Students write three programming assignments with PSP1 and PSP1.1. Students learn statistical methods for producing size and resource estimates and earned value for schedule planning and tracking.

PSP2 and PSP2.1
Once students have control of their plans and commitments, the focus then changes to quality management. Students write four programming assignments using PSP2 and PSP2.1. Students learn early defect detection methods and improved design practices.

Mid-Term and Final Reports
After the first six assignments have been completed and after all ten programming assignments have been completed, students write mid-term and final reports. These reports document their analysis of their performance. Students are required to analyze their data to understand their current performance, define challenging yet realistic goals for the second half of the class, and to identify what specific changes they will make to achieve those goals.

By the end of the course, students can plan and control their personal work, define processes that best suit them, consistently produce quality products on time and for planned costs.

3.1.4 PSP Measurement Framework

Students collect three basic measures: size, effort, and defects. Students use many other measures that are derived from these three basic measures. Both plan and actual data are gathered and recorded. Actual data are used to track and predict schedule and quality status. All data are archived to provide a personal historical repository for improving estimation accuracy and personal improvement. Derived measures include

- estimation accuracy (size/time)
- prediction intervals (size/time)
- time in phase distribution
- defect injection distribution
- defect removal distribution
- productivity
- percent of reuse
- cost performance index
- planned value
- earned value
- predicted earned value
- defect density
- defect density by phase
- defect removal rate by phase
- defect removal leverage
- review rates
- process yield
- phase yield
- failure cost of quality
- appraisal cost of quality
- appraisal/failure COQ ratio

3.1.5 The TSP

The TSP is based on the following principles:

- The engineers know the most about the job and can make the best plans.
- When engineers plan their own work, they are committed to the plan.
- Precise project tracking requires detailed plans and accurate data.
- Only the people doing the work can collect precise and accurate data.
- To minimize cycle time, the engineers must balance their workload.
- To maximize productivity, focus first on quality.

The TSP has two primary components, a team building component and a team working or management component.

Successful team building programs typically expose a group to a challenging situation that requires cooperative behavior of the entire group [Morgan 1993]. As the group's members learn to surmount this challenge, they generally form a close knit and cohesive group. The TSP follows these principles to mold development groups into self-directed teams. However, instead of using an artificial situation like rock climbing or white water rafting, it uses the team launch. The chal-

lenge in this case is to produce a detailed plan for a complex development job and then to negotiate the required schedule and resources with management [Humphrey 2002].

3.1.6 The TSP Launch

The first step in developing a team is to plan the work. This is done through the TSP launch. The launch is lead by a qualified team coach. In a TSP launch, the team reaches a common understanding of the work and the approach they will take, then produces a detailed plan to guide its work and obtain management support for its plan. A TSP launch is composed of nine meetings over a four-day period.

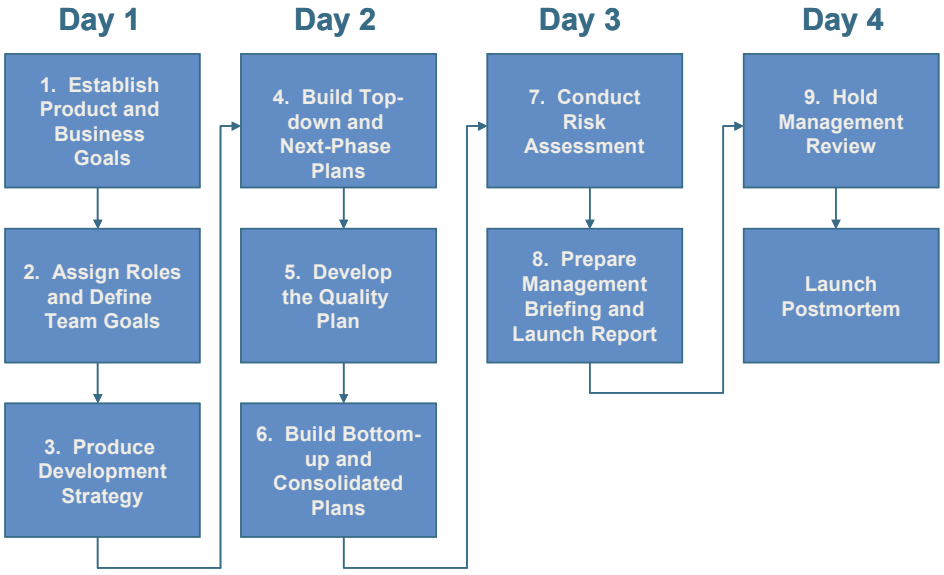

Figure 10: The TSP Launch

The first step in the launch is for the team to understand what they are being asked to do. This is accomplished in meeting 1 by having marketing (or an appropriate customer representative) and management meet with the team. Marketing describes the product needs. Management describes the importance of the project and the resources/constraints that the team has. This is also a chance for management to motivate the team.

In meeting 2, the team sets its goals and organizes itself. The team reviews the business and product goals presented in meeting 1, and derives a set of measurable team goals. Next, the team divides up the team-management tasks among the team members.

In the TSP, routine team management tasks are assigned to eight team member roles:

- customer interface manager
- design manager
- implementation manager
- test manager
- planning manager
- process manager

- support manager
- quality manager

Each team member selects a role. For teams with more than eight members, roles are shared. With smaller teams, team members may select multiple roles.

In launch meetings 3 and 4, the team creates the overall project strategy and plan. In meeting 3, the team produces a conceptual design, devises the development strategy, defines the detailed process it will use, and determines the support tools and facilities it will need. It lists the products to be produced and estimates the size of each product.

In meeting 4, the team develops the team plan. It does this by estimating the size of the products to be produced; identifying the general tasks needed to do the work and their effort estimates; defining the next phase tasks to a detailed work step level; and developing a schedule of the team members' availability week by week through the completion of the project.

In meeting 5, the team defines a plan to meet its quality goal. The team does this by estimating the number of defects injected and removed in each phase and then calculating the defect density of the final product. The team ensures that the tasks needed to achieve its quality goal are included in the team plan. The quality plan provides a measurable basis for tracking the quality of the work as it is done.

In meeting 6, the work on the team plan for the next phase of work is allocated to team members and each person creates an individual plan. In building their plans, the engineers refine the team estimates using their own data, break large tasks into smaller tasks to facilitate tracking, and refine their hours available per week. The team then meets again to review the individual task plans and to ensure that the work is balanced. The individual plans are then consolidated into a team plan. The team uses this plan to guide and track its work during the project work phase.

The team conducts a risk assessment in meeting 7. Risks are identified and their likelihood and impacts are assessed. The team defines mitigation and contingency plans for high priority risks. Risks are documented in the team plan and assigned to team members for tracking.

Meeting 8 is used to develop a management presentation of the team's plan. If the team's plan does not meet management goals, the team includes alternate plans that come closer to meeting management's goals. For instance, the team might be able to meet a schedule by adding resources to the team or by reducing the functionality delivered.

In meeting 9, the team presents the plan to management for approval to start the work. The team explains the plan, describes how it was produced, and demonstrates that all the members agree with and are committed to the plan. If the team has not met management's objectives, it presents the alternate plans. The principal reason for showing alternate plans is to provide management with options to consider in case the team's plan does not meet business needs.

At the end of the TSP launch, the team and management should agree on how the team is to proceed with the project.

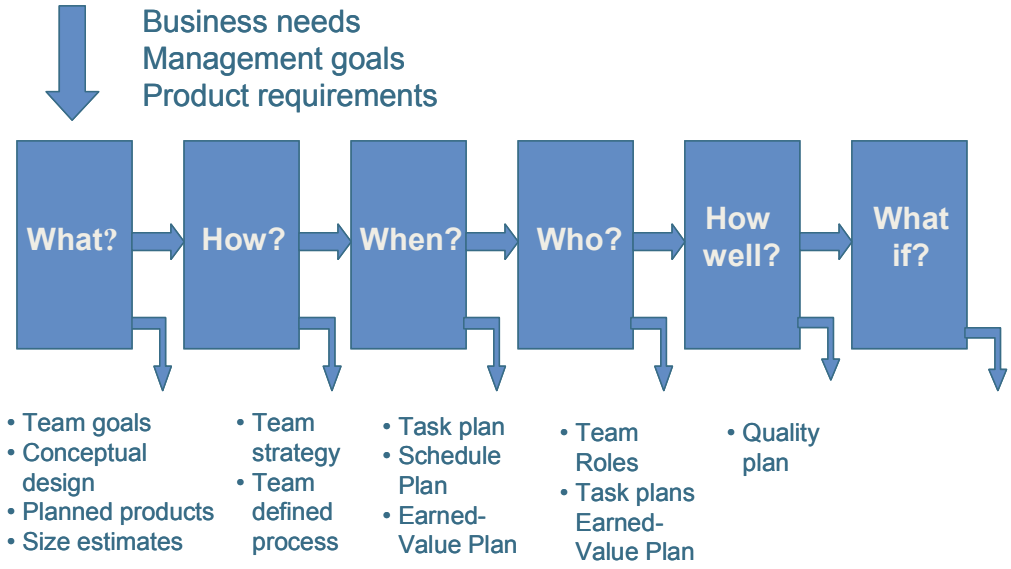

Figure 11: The TSP Launch Products

By the end of the launch, the team has formed a cohesive unit and created a plan that balances the needs of the business and customer with a technical solution. The team has agreement on what technical solution it proposes to build and an understanding of how that product will satisfy business and customer needs. The team agrees on the strategy for developing the product (e.g., the disintegration and integration strategy) and on the process that will be used to develop each part of that strategy. The team has a detailed plan that it can use to guide and track the work. Each team member knows what tasks and areas they and others are responsible for. Everyone on the team understands and agrees with the quality goal and the team can monitor progress against that goal. Finally, the team has explored all the things that might go wrong and has done its best to mitigate those risks. In short, the TSP launch provides a team with all the conditions necessary to become a self-directed team.

The TSP includes guidance for ensuring that the energy and atmosphere from a TSP launch are sustained as the team completes it work. A TSP launch coach works with the team and the team leader to help the team collect and analyze data, follow the process defined the team, track issues and risks, maintain the plan, track progress against goals (especially the team's quality goal), and report status to management.

3.1.7 TSP Measurement Framework

The TSP uses the same basic measures of the PSP: size, time, and defects, and adds task completion dates. For all measures, planned and actual data are collected at the individual level. The TSP measurement framework consolidates individual data into a team perspective. The data collected are analyzed by the team weekly to understand project status against schedule and quality goals. The TSP measurement framework also makes available other views of the data such as by product or part, by phase, by task, by week, by day, and the like. Personal and team data are archived to provide a repository of historical data for future use.

The team conducts weekly meetings to report progress against its plans and discuss team issues. The team also makes accurate status reports, based on data, to management regularly. Because management can rely on that data, its job changes from continuously checking on status to ensuring that there are no obstacles holding the team back. This also allows management to make sound business decisions, because those decisions are based on accurate engineering data. For example, when management is confident in the team's estimate, management can decide how to allocate resources to obtain a schedule that best meets business needs. When a team commitment is in jeopardy, the team solves the problem or raises the issue with management as early as possible. In all cases and at all levels, decisions are made based on data.

3.1.8 The TSP Introduction Strategy

The SEI has been transitioning TSP into organizations since 1997 and has amassed significant experience with issues surrounding the introduction of this technology. Based on these experiences, the SEI has defined an introduction strategy and developed supporting materials to facilitate the implementation of that strategy.

The introduction strategy starts with trial use. The TSP is first piloted on several small projects to evaluate both the effect of TSP on the organization and the transition approach. The pilots also build the understanding, sponsorship and support needed for broad acceptance of the TSP within the organization.

All team members and all of their management must be trained prior to start of the pilot effort. The senior management attends a one-and-a-half-day executive seminar and planning session. The middle and line management attend three days of training. The engineers complete the two-week PSP for Engineers course. The pilot teams are then launched using the TSP and data are gathered and evaluated. Pilot projects can rapidly demonstrate the benefits of using the TSP and results from the pilot projects can be used to tailor and improve both the TSP and the introduction strategy.

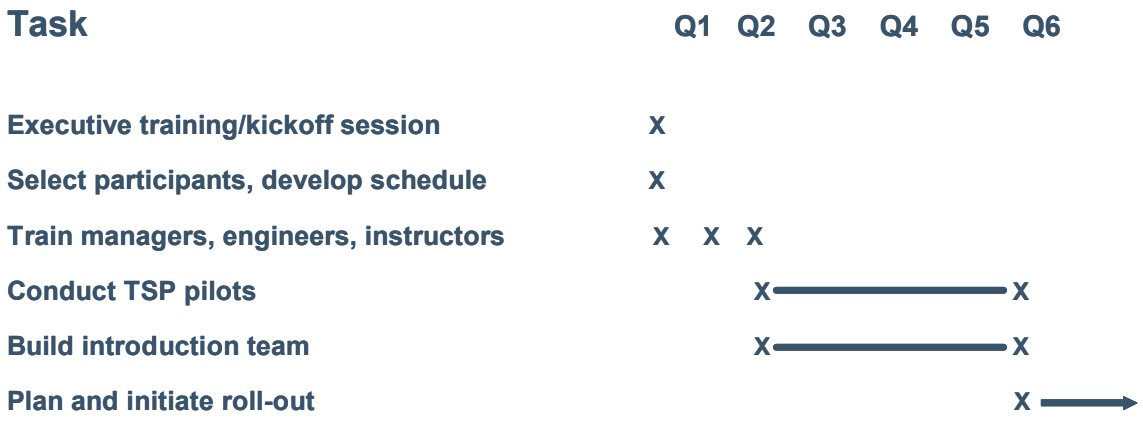

Figure 12: TSP Introduction Timeline

4 An Example First-Time TSP Project

The organization that we will talk about in this example has been involved with CMM-based software process improvement. Preparation for the initial pilots included

- an executive planning session
- a TSP executive seminar
- a "Managing PSP-Trained Engineers" class
- an "Introduction to Personal Process" class
- "PSP for Engineers" classes

Candidate projects were selected for TSP pilots. Team members from the potential pilots, project managers, and software engineering process group participated in the training classes.

4.1 PSP For Engineers Training

The engineers—along with members of the software engineering process group and project managers—attended the "PSP for Engineers" course. The class data presented in this report represents the 56 percent of students who completed all of the eight programs. All relevant estimation and quality data presented is based on new and changed lines of code that students write for each program.

Table 3 shows some key measures tracked in the PSP. The first column describes the measure, the second column shows its value at the start of PSP training (class average for first two programs), and the third column shows its value at the end of PSP training (class average for last two programs).

During the training students are required to complete eight programming assignments and two reports as they progress from PSP0, their current process, to PSP2.1, a high maturity personal process. Table 1 shows the evolution in process maturity as students progress from PSP0 to PSP2.1.

Table 3: PSP Process Levels and the Associated Programming Assignments

PSP Evolution	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 also define a defect type standard and log all defects found in the review, compile, and unit-test phases.
PSP0.1	2	Students define a coding standard, a line of code (LOC) counting standard, use Process Improvement Proposals (PIPs), and start measuring the size of their programs in LOC.

PSP1	3	Students add defined size estimation methods and effort estimation methods to their personal process. Test reports are also introduced.
PSP1.1	4	Task and schedule planning are introduced. Earned value tracking is also introduced.
PSP2	5	Quality techniques are introduced. Structured personal code and design reviews, based on individual defect data, are conducted.
PSP2.1	6, 7, 8	Design templates and design verification methods are introduced.

4.2 PSP Training

The following shows the number of students who completed each of the program assignments. The class consisted of 20 students. The class is demanding. Without commitment and support, it is common for the students not assigned to TSP teams to not complete all programs. Status is shown as follows.

Table 4: Number of Students Who Have Completed Each Assignment

Program Number	Number of students who completed
1	18
2	16
3	16
4	16
5	16
6	13
7	12
8	9

The Team Software Process relies upon the PSP skills of project team members. Typically, we require that all software developers complete the "PSP for Engineers" course prior to conducting a launch. The students may not have allocated sufficient time during and immediately after the course to complete training prior to launch.

4.2.1 Background

The estimation and quality data presented in this report are based upon new and changed LOC that students write for each program. Although the PSP encourages and tracks reuse, reused code is not included in the process data presented in this report.

The PSP is introduced in two distinct phases. Students learn how to estimate and track program size and effort during the first week of the course. Quality concepts are introduced in the second week. Introducing size and effort estimation principles first and following them with quality improvement concepts is important. Until students can estimate and track program size and effort, quality measures cannot be planned and tracked. A frequent question is why personal code and design reviews are not introduced until the second week of training. To plan effective reviews, students need to know the optimal review rates (LOC reviewed per hour), optimal time spent in review phases (percentage of total development time spent on reviews), quality of code after reviews (defects found per thousand lines of code), and the like. All these planning and tracking metrics require either a size measure or an effort measure. Thus, size and effort estimation techniques are introduced first in the PSP.

4.2.2 Improvement Summary

Table 5 shows some key measures tracked in the PSP. The first column describes the measure, the second column shows its value at the start of PSP training (class average for first two programs), and the third column shows its value at the end of PSP training (class average for last two programs). Data are shown only for the thirteen students who completed at least six programs.

As these data show, by the end of the course, engineers have the skills to produce high-quality software on time. The Team Software Process relies upon the PSP skills of project team members. Comparing the beginning and end of the course, similar effort was spent in all defect removal activities. There was minimal overall change in total code production rates. However, at course end, the data show that the students were finding and removing defects prior to compile and test. Lower defect density entering test have been found to correlate with lower defects density post test. With the same effort, students produced higher quality code using more predictable methods.

The engineers on the team have started using the skills learned in the PSP courses. It is important for team leads and the leadership team to ensure that those engineers continue to practice and use their PSP skills and methods. Onsite coaching support by PSP instructors and TSP launch coaches will be critical as teams continue their work.

Table 5: Values of PSP Measures at Beginning and End of the Course

Measure	At start of training	At end of training
Percent time spent in compile	8%	2%
Percent time spent in unit test	21%	9%
Compile defect density (number of defects found during compile per KLOC)	40 defects/KLOC	8 defects/KLOC
Unit test defect density (number of defects found during unit test per KLOC)	30 defects/KLOC	10 defects/KLOC
Yield (percentage of defects found before first compile)	25%	65%
Productivity	42 LOC/Hour	38 LOC/Hour

The PSP and the TSP use several measures to track project schedule and quality. These measures are derived from data gathered by individual engineers as they do their day-to-day work. Although a great deal of data is available to manage the project and to manage individual performance, engineers collect only three basic measures: time, size, and defects. Most other measures are derived from these three base measures.

The remainder of this report describes PSP class data on some of these measures. Some key process measures, such as the percentage of time spent in unit test and the density of defects in unit test, have not leveled off. These trends, together with the decrease in unit test defect density and increased productivity shown in Table 5, indicate that the course has provided a springboard from which even higher levels of performance can be anticipated.

4.2.3 Actual Time Range

Figure 13 shows the average time required for development of each of the eight programs together with the greatest and least time required by any of the students.

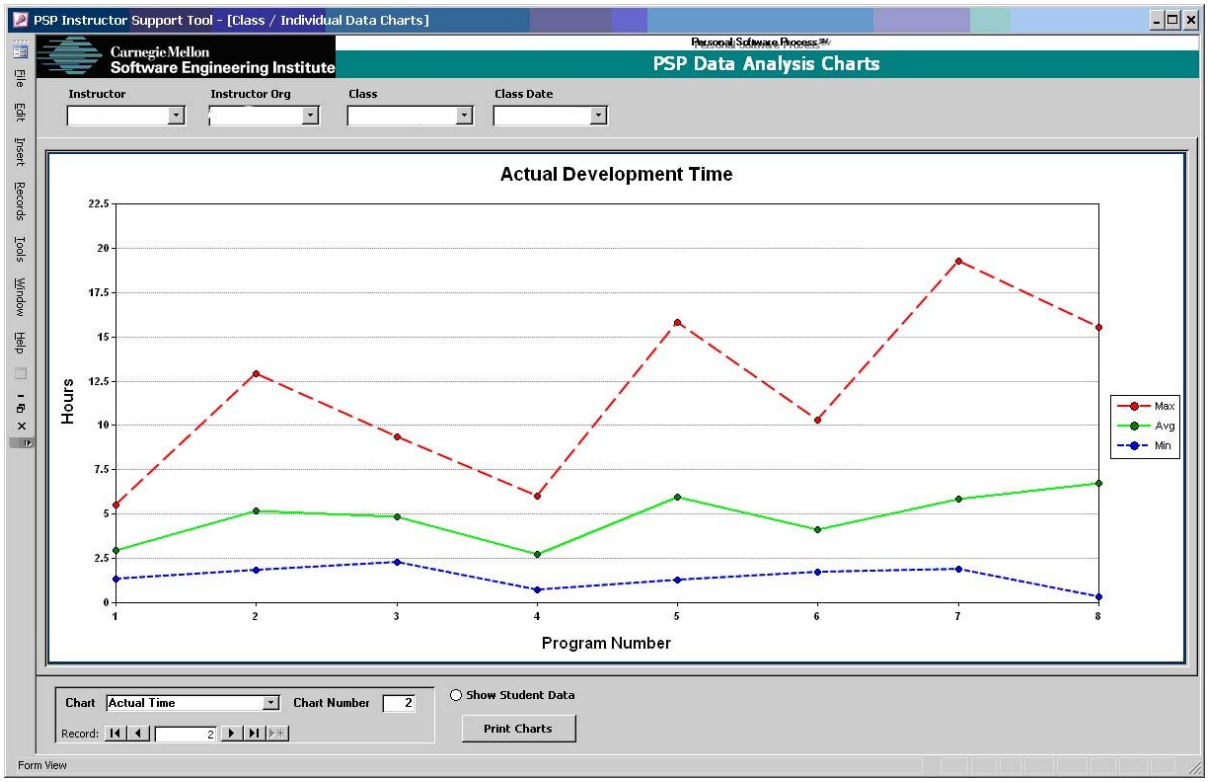

Figure 13: Actual Time Range

The average time spent on most programs was about five hours per program. The range varied from about an hour to almost 19 hours spent on a single program. This is longer than typical. A common range is two to six hours per program. Only a few students were able to complete programs on the same day as the programs were assigned.

4.2.4 Actual Size Range

Figure 14 shows the average size of each of the eight programs together with the greatest and least size of the assignment programs.

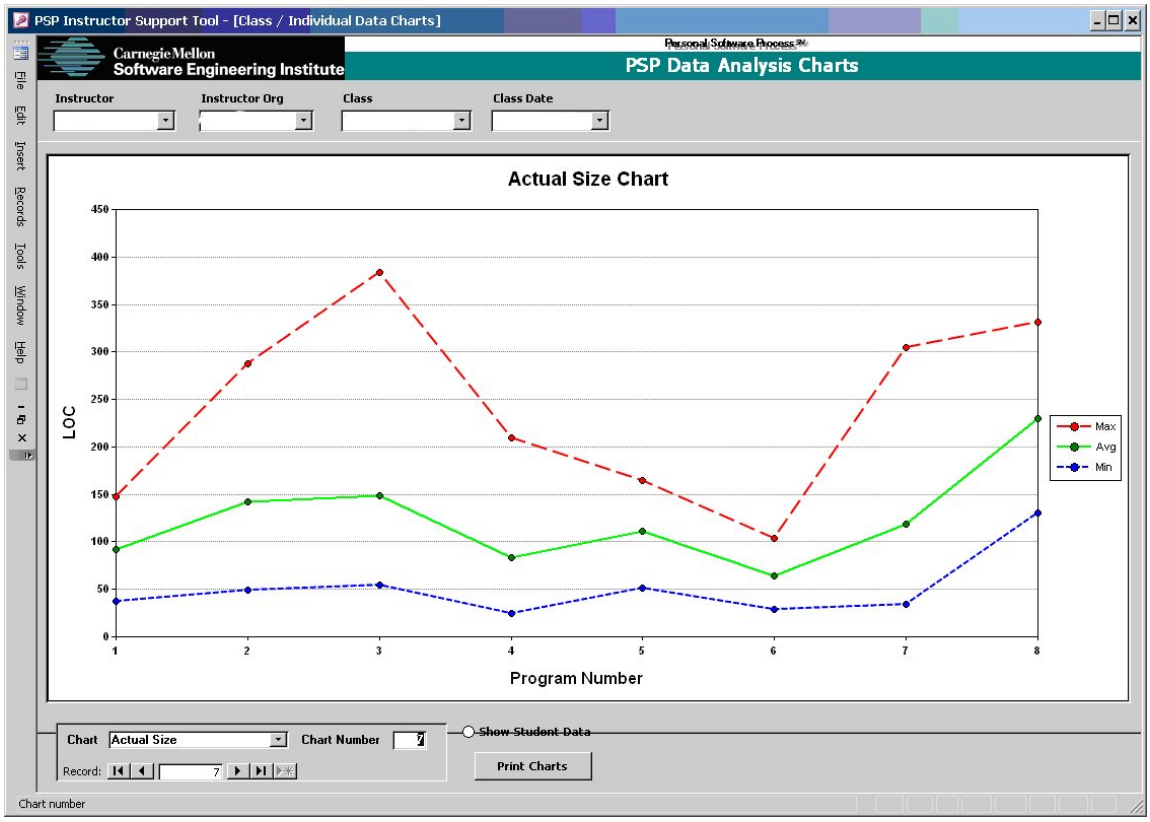

Figure 14: Actual Size Range

The average program size ranged from about 60 new and changed LOC to more than 230 new and changed LOC. One factor influencing program size was the variety of programming languages used in the class (C++, Visual Basic, Perl, and Java).

4.2.5 Composite Time Estimating Accuracy

The time estimation error is calculated as follows:

%Estimate Error = 100 * (Actual – Estimate)/Estimate

Underestimates have a positive error, and overestimates have a negative error. Figure 15 shows the percent error for estimation of development time for each program.

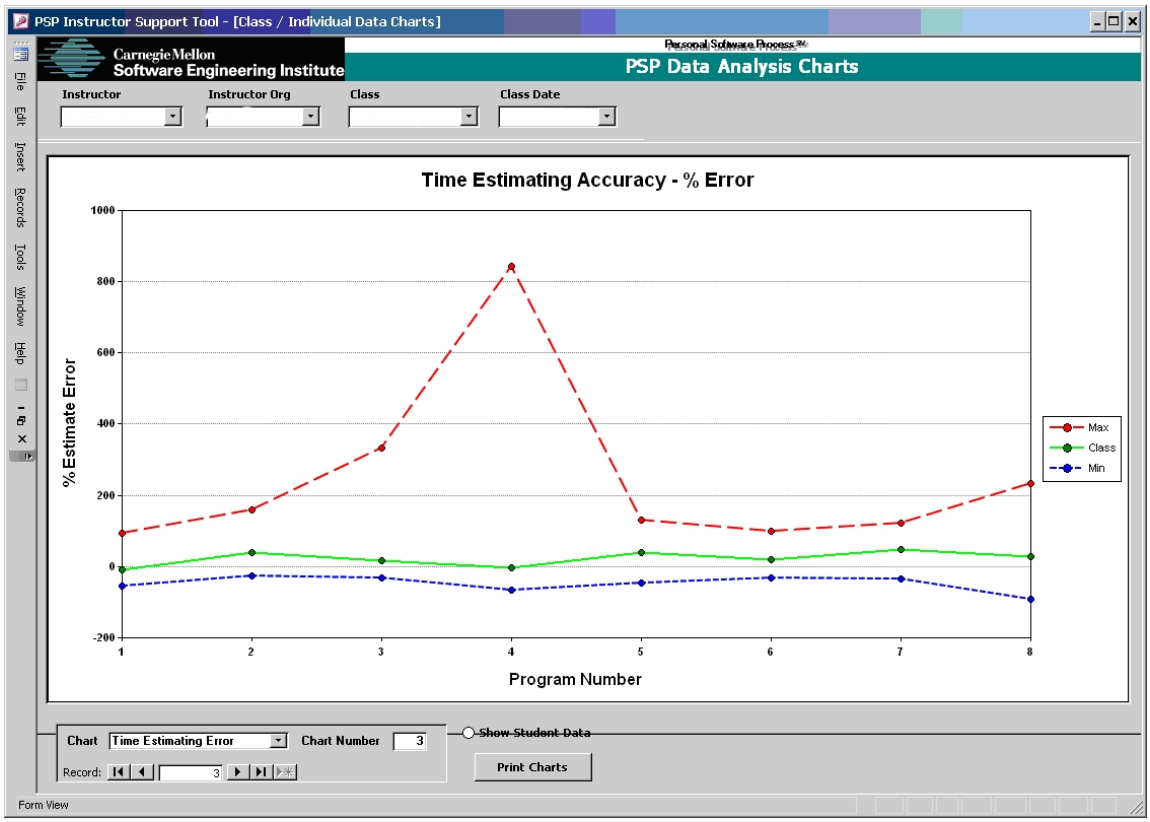

Figure 15: Time Estimating Accuracy— Percent Error

The class data are the composite estimates for all students. That is, the composite estimate is the sum of the students' estimated hours. Similarly, the composite actual time is the sum of the students' actual hours. The class error calculation is the error between the composite estimated hours and the composite actual hours. The class line illustrates the benefits of independent estimates. All students estimate each program independently. Some overestimate, while others underestimate, thus balancing the composite estimate.

The very large maximum estimation error shown on assignment 4 was for one student, who had a difficult time finding and fixing a defect in unit test. None of the other students had an estimation error for assignment 4 that exceeded the maximum estimation error shown for assignment 3.

As a group, the class underestimated the time required to write their assignments. The greatest error for overestimation is much smaller than the greatest error for underestimation; that is, the error is asymmetric. Nevertheless, as a group the class has estimates that are close to zero. This represents the balance achieved by allowing each member of the class to estimate his or her own time. The individual variation was large, but the cumulative bias was small.

4.2.6 Composite Size Estimating Accuracy

The size estimation error is calculated as follows:

%Estimate Error = 100 * (Actual – Estimate)/Estimate

Underestimates have a positive error, and overestimates have a negative error. Figure 16 shows the percent error for estimation of size for each program assignment.

The class composite is the sum of the students' estimated program size. Similarly, the class composite size is the sum of the students' actual program size. The class composite appears to be stable, with an improving trend, again illustrating the benefits of combining individual estimates. The range of the size estimation error narrows as the students learn to use a defined size estimation procedure. (Note: Students did not estimate program size for program 1, hence the zero percent error for that data point.)

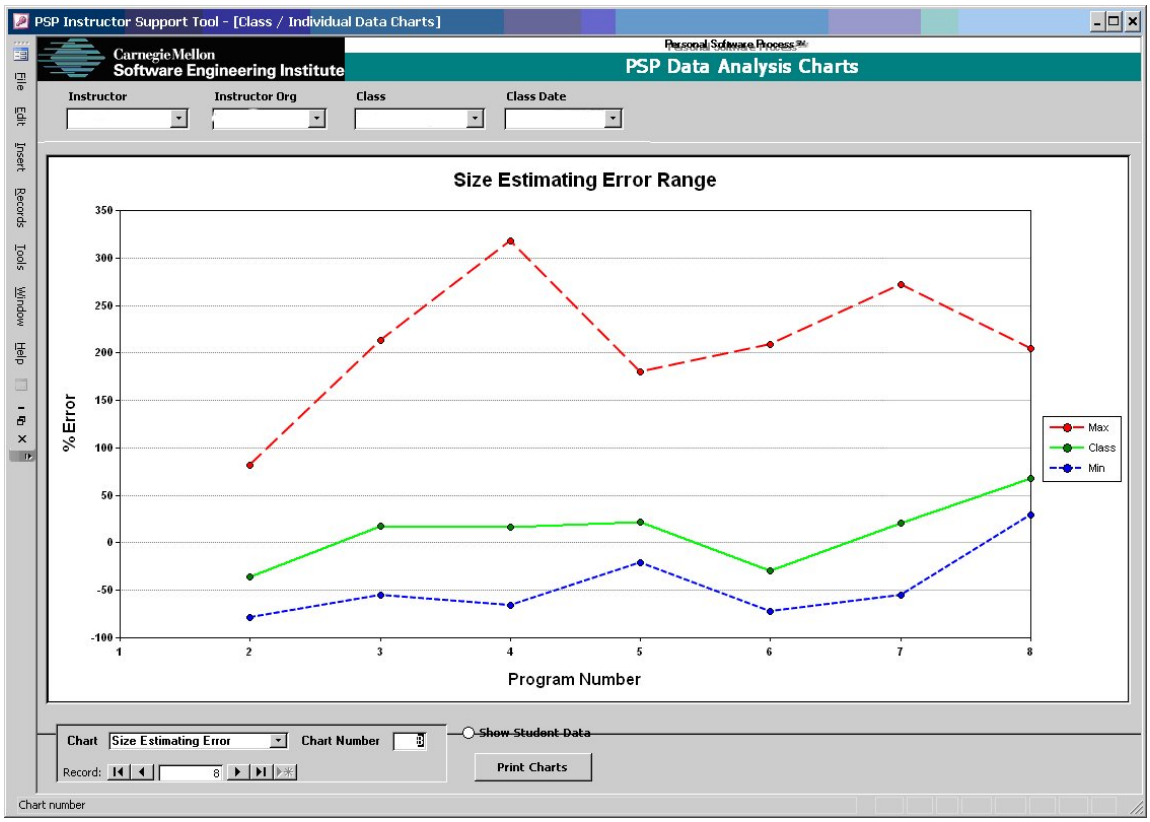

Figure 16: Size Estimating Accuracy – Percent Error

4.2.7 Compile Time Range

In the PSP, the compile phase starts when a student first begins compiling a program, and ends when the student gets a clean compile. The time spent in the compile phase is called the compile time. Incremental compiles are encouraged, with each increment adding to the total compile time. Figure 17 shows the percentage of development time spent in the compile phase.

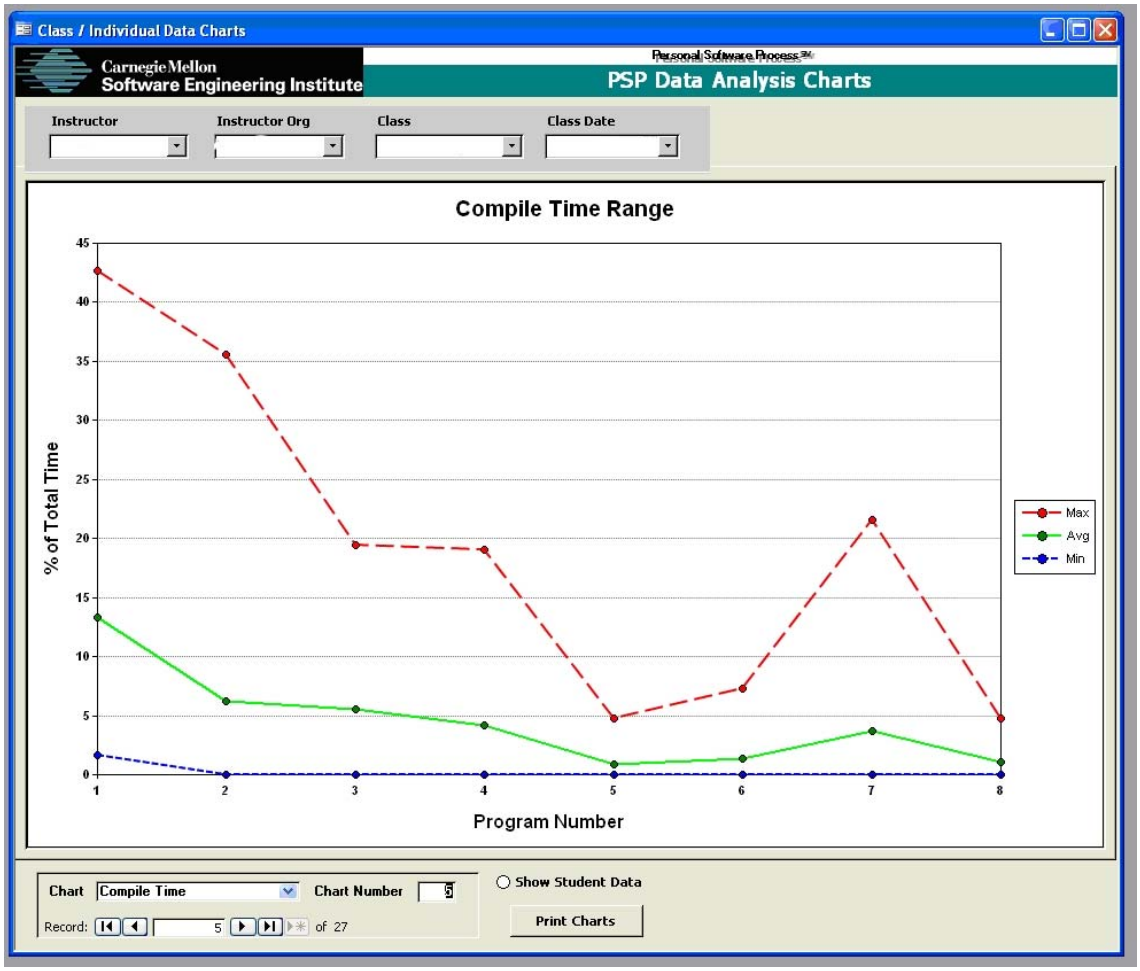

Figure 17: Percentage of Development Time in Compile.

Students started the course spending about 12 percent of total development time in compiling their programs. Towards the end of the course, this time had been reduced to about 2 percent. Students were compiling clean code. Thus, time spent in the compile phase was minimal.

The range of compile time also narrowed as the students progressed through the class. In the case of two assignments, the maximum compile time for any student who recorded compile time was 5 percent. Some students did not record compile time because their development environment performed a compilation every time source code is saved during the coding phase. Thus, the minimum compile time is shown as zero for programs 2 through 8.

4.2.8 Unit-Test Time Range

In the PSP, the test phase begins after the software has been compiled, and finishes when all tests execute error free. Time in the test phase includes the time to execute the tests, and the time to find and fix any defects found during test. As with compilation, incremental testing is encouraged. Figure 18 shows the percentage time spent in unit test for each program assignment.

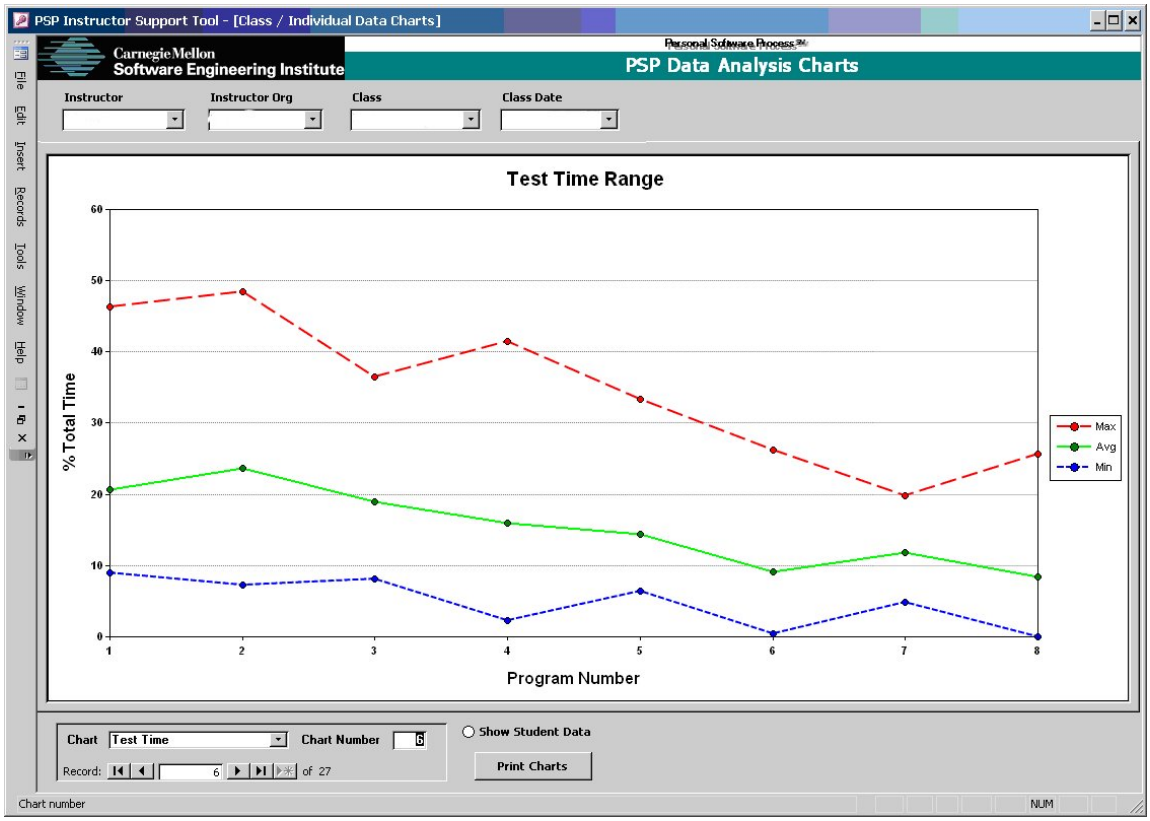

Figure 18: Percentage of Development Time in Unit Test

At the beginning of the course, the class average for total time spent in unit-test was about 20 percent. By the end of the course, the average percentage of total time spent in unit-test had been reduced to less than 10 percent and the worst case was equal to the class average at the beginning. This indicates students were putting much cleaner programs into test, and were thus spending less time in finding and fixing defects during test. This also suggests fewer defects escaped the final test phase.

For assignments 6 through 8 the trend for the average and the trend for the maximum are decidedly downward and haven't yet leveled out. For assignments 6 through 8, eight student submissions had zero defects in unit test, thus helping to bring about the decrease in percentage of time in unit test shown in Figure 18.

4.2.9 Defect Removal Yield

Process yield (generally referred to as yield) is defined as the percentage of defects injected before compile that were removed before compile. Figure 19 shows yield for each program assignment.

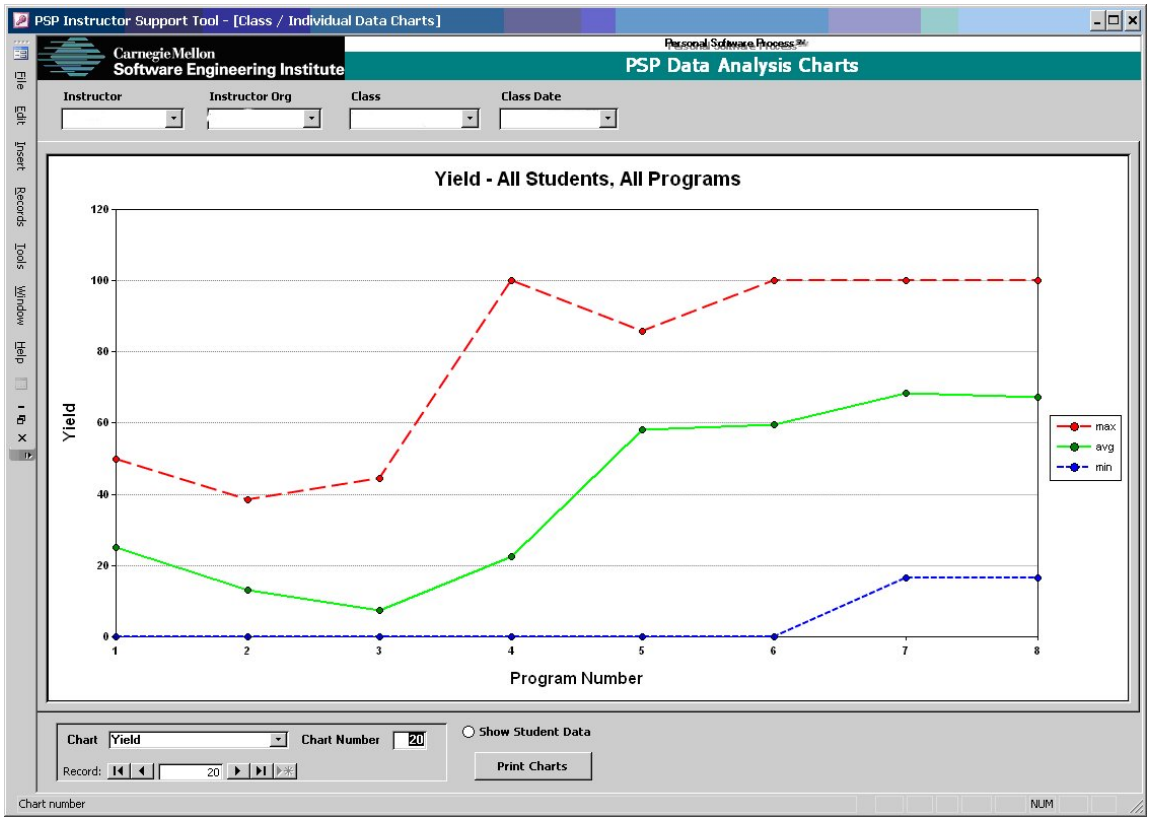

Figure 19: Yield

For assignments 6, 7, and 8 the class average yield was better than 60 percent. For assignment 8 the average yield was almost 70 percent. Thus, by program 8, students were finding and fixing almost 70 percent of all defects in their code before the first compile. It should be noted that for assignments 7 and 8, the minimum yield was greater than the average yield for programs 1 through 4. Personal reviews were introduced in assignment 5. The data indicates that the class personal reviews were effective.

4.2.10 Compile Defect Density

Figure 20 shows the number of defects per KLOC found during the compile phase.

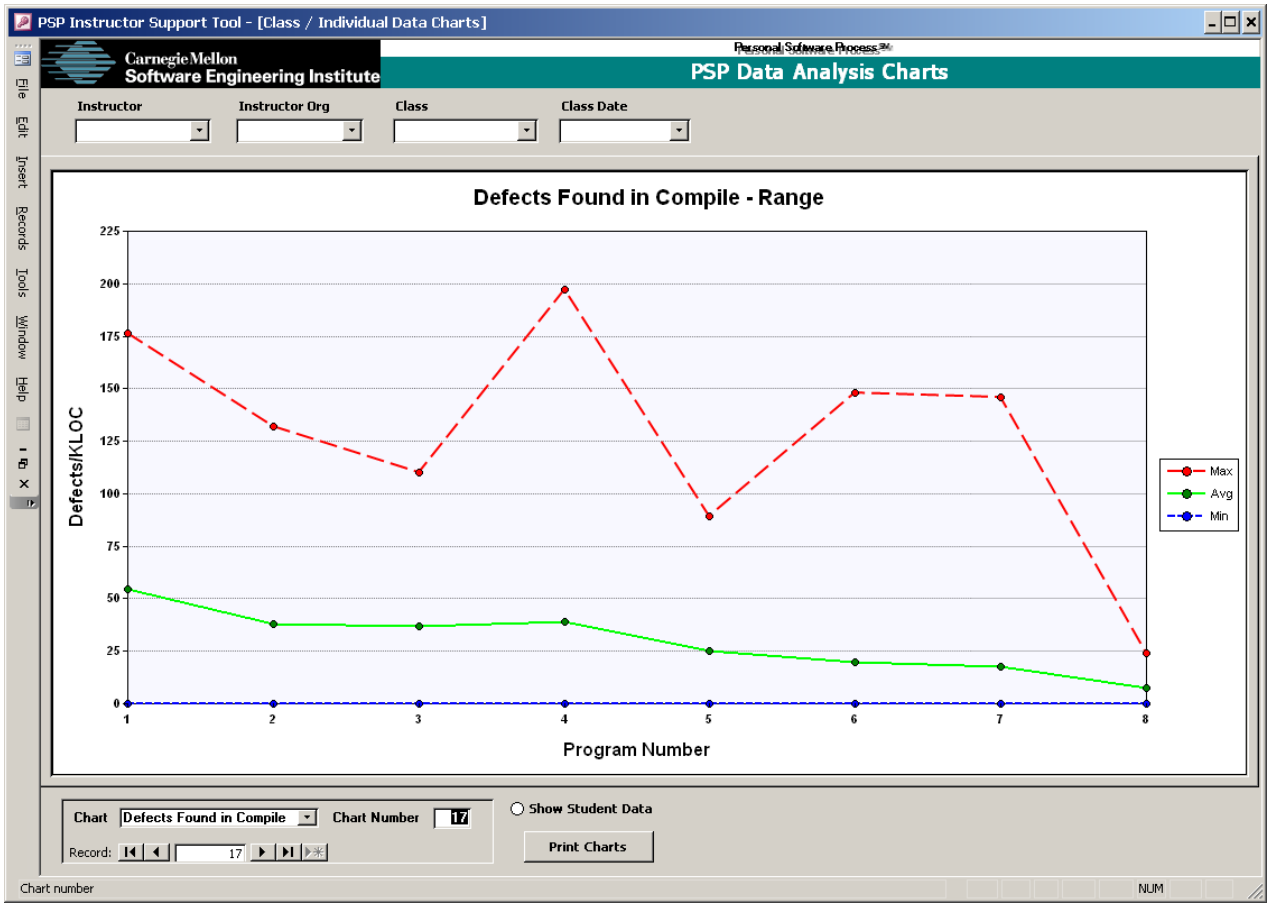

Figure 20: Defect Density in the Compile Phase

Average compile defect density (number of compile defects per thousand lines of new and changed code) for the class started at about 60 defects/KLOC. By the end of the course, this declined to almost fewer than 10.

PSP developers review their code before compilation. Code review is one of the most effective defect removal techniques; therefore, source code that is being reviewed by a PSP developer has fewer defects than would be found in the code of most other developers. Since the number of defects found in compile is highly correlated with the number of defects found in unit test, a low defect density in compile is an early indicator of a low defect density in unit test.

4.2.11 Unit Test Defect Density

Figure 21 shows the number of defects per KLOC (defects/KLOC) found during the unit test phase.

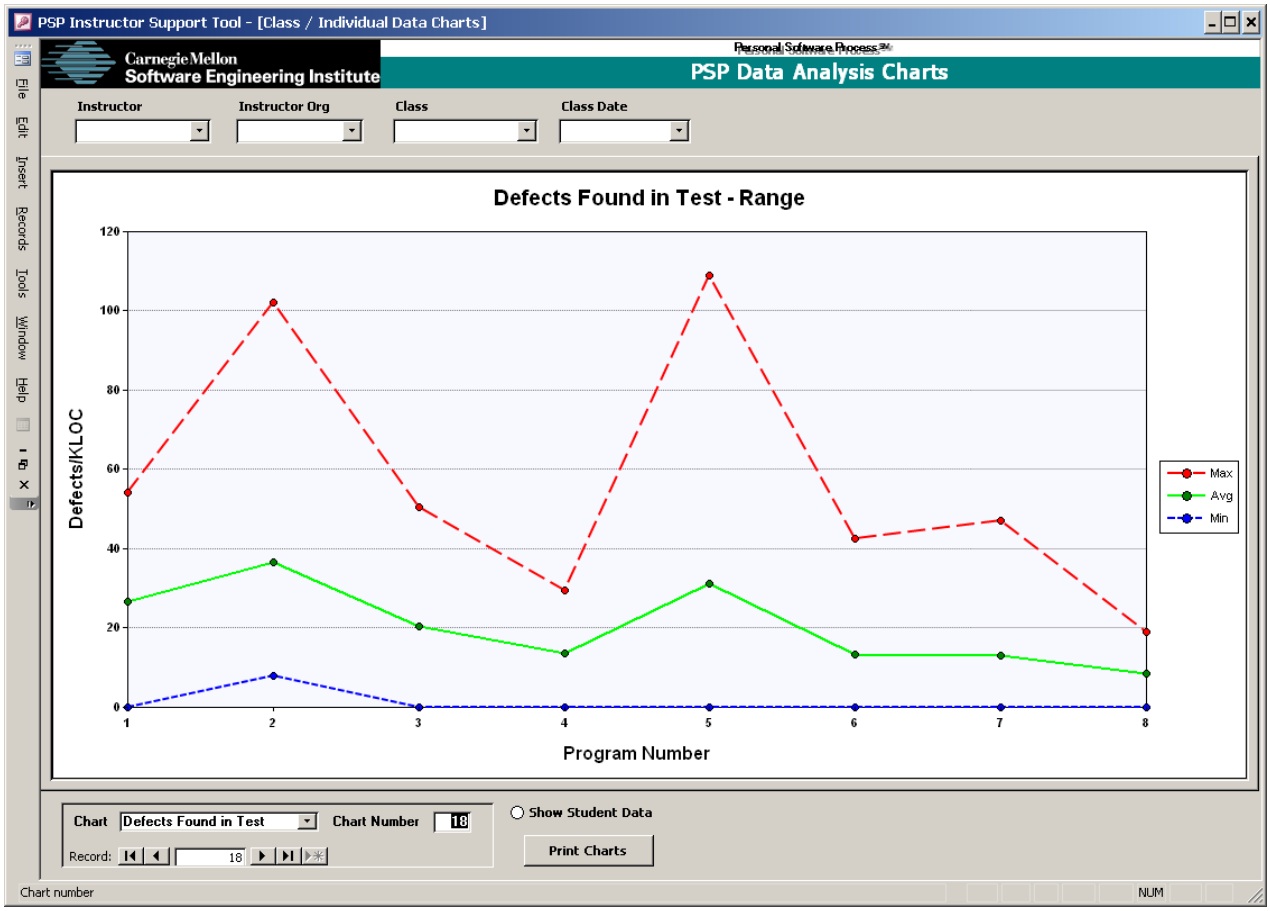

Figure 21: Defect Density in the Unit Test Phase

The class started with an average unit test defect density (number of defects per thousand lines of new and changed code found in the unit test phase) of about 30. By the end of the course, the unit-test defect density was about 10. Unit test defect density was reduced by a factor of 4.

Also, note the variance around the class average. As students learned and used PSP quality methods, not only did the class average defect density improve, but the range also narrowed. This means the gap between the student with the highest defect density and the student with the lowest defect density narrowed considerably.

4.2.12 Appraisal to Failure Ratio

The cost of quality has three components: failure costs, appraisal costs, and prevention costs. In the PSP, we calculate appraisal costs as the time spent in design and code reviews. The failure time is calculated as the time spent in compile and test. We do not explicitly track prevention costs. The appraisal to failure ratio (A/FR) is calculated by dividing appraisal costs by failure costs. Figure 22 shows the A/FR for each of the program assignments.

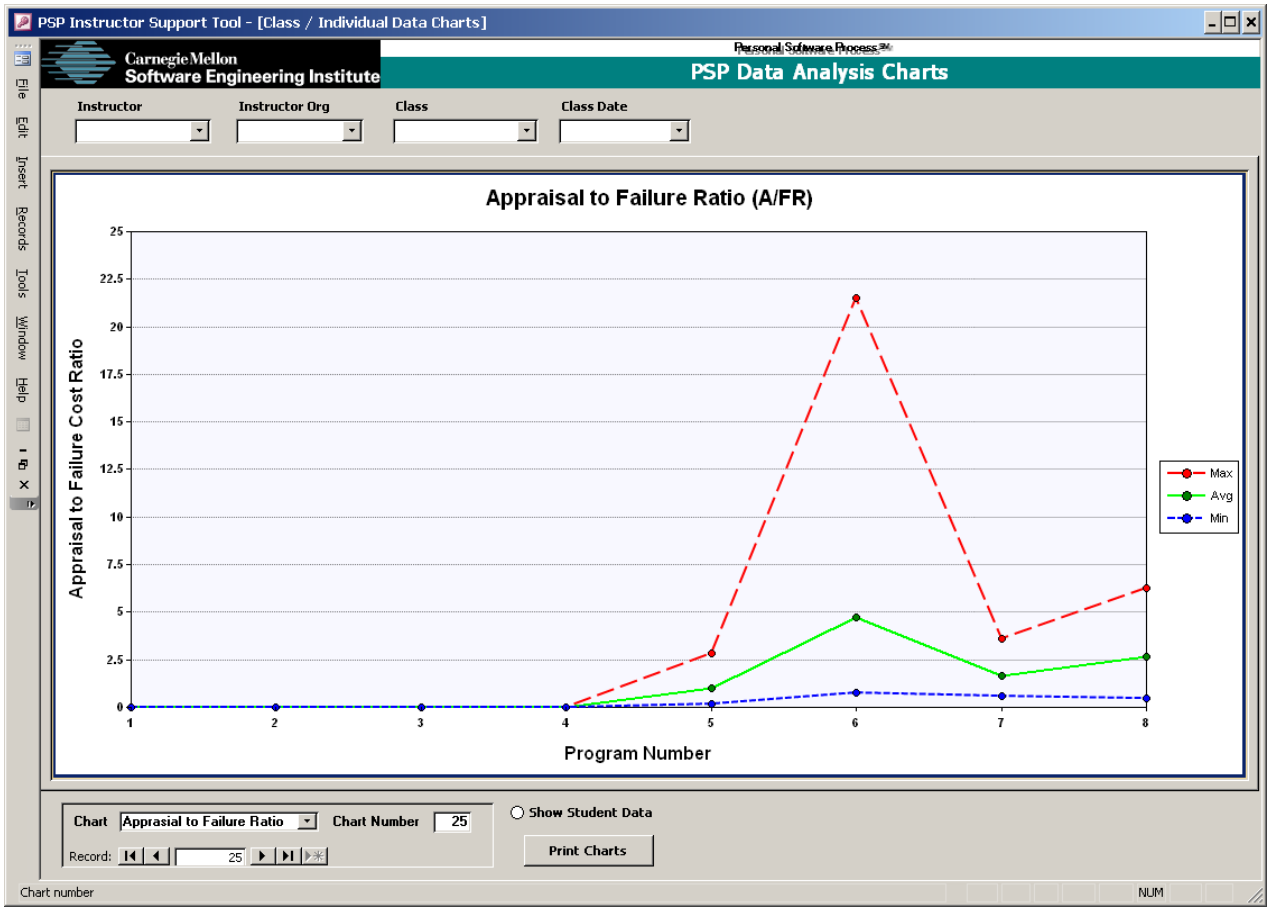

Figure 22: Appraisal to Failure Ratio

Appraisal costs (design and code reviews) were not introduced as PSP phases until assignment 6. After that, the class average shows that more than twice as much time was spent in appraisal activities (design and code reviews) than in failure activities (compiling and testing).

4.2.13 Unit Test Defect Density vs. A/FR

Figure 23 shows the relationship between A/FR and unit test defect density. Programs with A/FRs greater than 2 have fewer defects found in unit test. Our typical class data indicate that programs with A/FR greater than 2 have very few defects found in unit test.

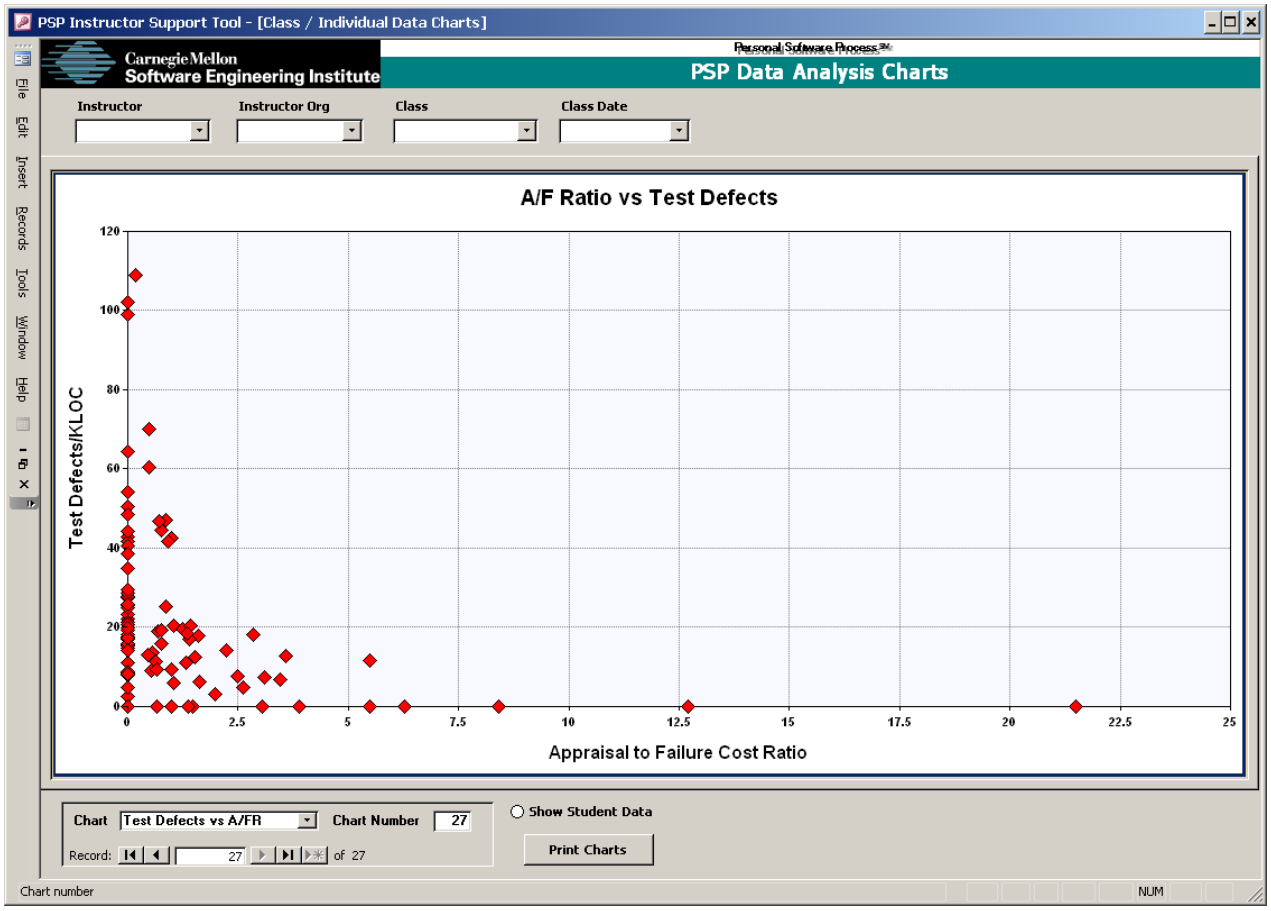

Figure 23: Relationship Between Appraisal to Failure Ratio and Unit Test Defect Density

4.2.14 Productivity

Figure 24 shows productivity (LOC/hour) for each program assignment. The overall productivity changed very little throughout the course. By the end of the course, students were producing code of a much higher quality (improving by a factor of about 4 in unit test defect density). This figure also shows that the PSP process overhead (taking data, design, code, and design review) does not affect productivity adversely.

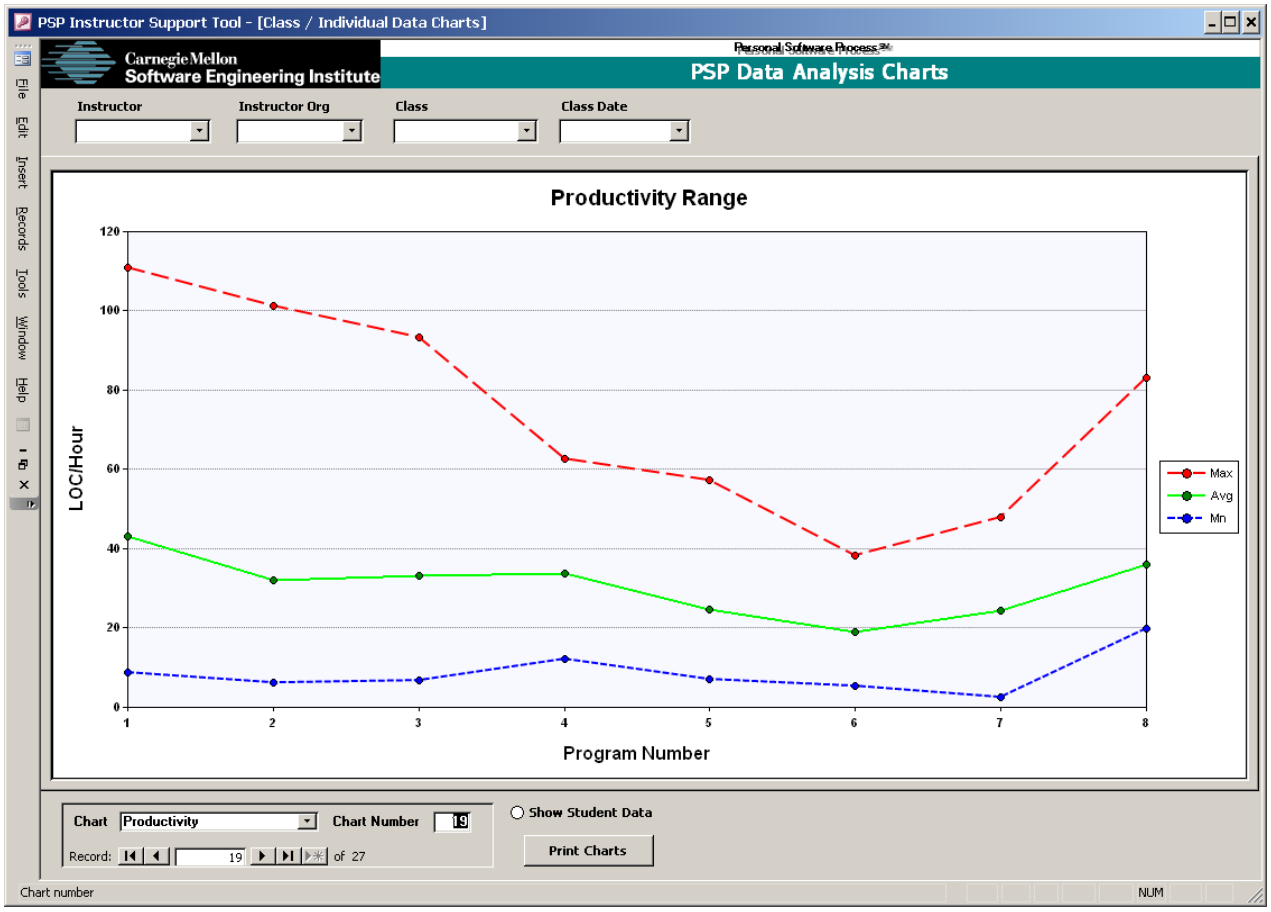

Figure 24: Productivity

Since the quality of the software leaving the unit-test phase is so much higher, less time will be spent in system and integration test, and later during field use. The cycle time for the product life cycle will be reduced. Thus, productivity through the entire product life cycle will be much improved. Moreover, the development rate will be more stable.

4.2.15 Training Conclusion

PSP training at this organization demonstrated substantial improvement in estimation accuracy and program quality. The unit test defect density was reduced by a factor of 8. This improvement is impressive. We believe the improvements can be greater still. First, to date, only 56 percent of the class has finished all eight programs. Based on our teaching experience, we expect the quality data to improve as more students complete the course. Second, as indicated by the relationship between A/FR and unit test defect density, we believe that students can improve the effectiveness of their reviews. We believe the class should be able to achieve average defect densities of less than ten defects/KLOC for compile and less than five defects/KLOC for unit test, thus ensuring that all but a few defects are removed prior to integration and system test. We hope to see these improvement trends continue as engineers apply disciplined methods on their projects.

The results from PSP training were encouraging, and consistent with results documented earlier in this report. The results are also consistent with those of Mexican undergraduate students. Compare, for example, the Test Time Range from a group of undergraduates [Salazar 2008]. The reductions in total test time and narrowing of the range are typical among all groups taking the PSP I and PSP II courses.

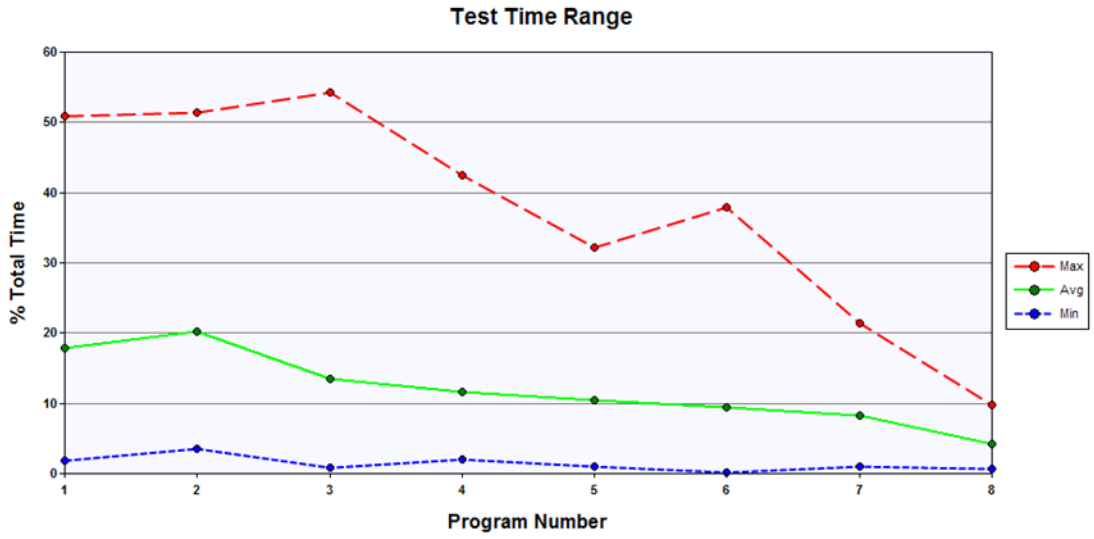

Figure 25: Undergraduates' Test Time Range

4.3 The Project Team Early Experience

The first TSP project was launched three months after beginning PSP training. Only one pilot was launched; the second project was cancelled for business reasons. The team consisted of five team members, plus the team leader and a tester. A candidate coach attended as an observer.

The team leader was open-minded and supported the use of TSP on this project. The leader also expressed concerns on two subjects. First, because the work had already been estimated prior to staffing and approving the project, a revised bottom-up estimate—even if more accurate—did not seem useful. Second, the customer was interested in the product and schedule rather than the piloting of TSP. The leader was concerned that TSP would appear only as project overhead.

4.3.1 The Launch

Senior management came prepared for the launch. Management discussed the importance of this product to the business, both from a functionality point of view, and a sales potential point of view. Management told the team that their they were to be an example for the organization and that the expectations included improving not only the planning, estimates, and quality of the project deliverables, but also to develop and reinforce teamwork and communication locally and between sites.

As the launch meetings proceeded, several difficulties were encountered. The challenges included the following:

- The coach spoke very little Spanish and more than one of the team members spoke little English.
- The tester attended the meetings remotely and there were some problems with NetMeeting and phone connections.
- Urgent work from the field required the attention of two team members, disrupting meetings 3 and 6.

In meeting 2, the team members chose roles that best suited their abilities, with more senior staff taking the roles more suited to technical work— design manager, for example. Conceptual design took a while to complete. There were two problems. First, the team already had been presented the work items and their effort estimate. It was difficult to guide the team to re-estimate the work using their personal experience. Second, the team was unsure how to structure the work for tracking and reporting. The needs of the team had to be balanced with the project management requirements imposed by the organization.

The development strategy and process appeared to be well defined by the organization and past experience. It was then necessary to fit the process into the TSP metrics framework. As the launch progressed, it became clear that the tester would be active only at the beginning and end of the project. The team leader and planning manager decided to build consolidated plans with and without the tester to facilitate progress reports.

The final team plan showed final software delivery to test six weeks beyond management's schedule goal. The problem had appeared somewhat worse earlier in the launch because of part-time staffing. The work could not be divided in a way such that the part-time staff could finish tasks in the desired time, nor could the work completions be synchronized. The team leader alerted management to this problem during the launch and obtained additional resource commitments. The TSP bottom-up plan had successfully highlighted a hidden cost of dividing a developer's time across multiple projects.

4.3.2 Plan Summary

The plan the team developed during the launch is summarized in the following table.

Table 6: Plan Summary

Delivery to testing group	Week 17
Ready to release	Week 19
Effort estimate	1,018 hours
New and changed LOC	2.94 KLOC
System test defect density	.34 defects/KLOC
Average task hours per team member per week	12.5 task hours/week

4.4 Executing The Plan

Effort problems commonly occur with new TSP teams. The team members frequently overestimate task hours available. A related problem is that part-time workers (assigned to more than one project) have task overhead on multiple projects, therefore they cannot provide the total effort hours as those assigned full time.

This occurred on this project. The following chart shows the planned and actual total team effort hours per week through the eighth week of the project. The team recognized the early shortfall and implemented work changes to address the problems.

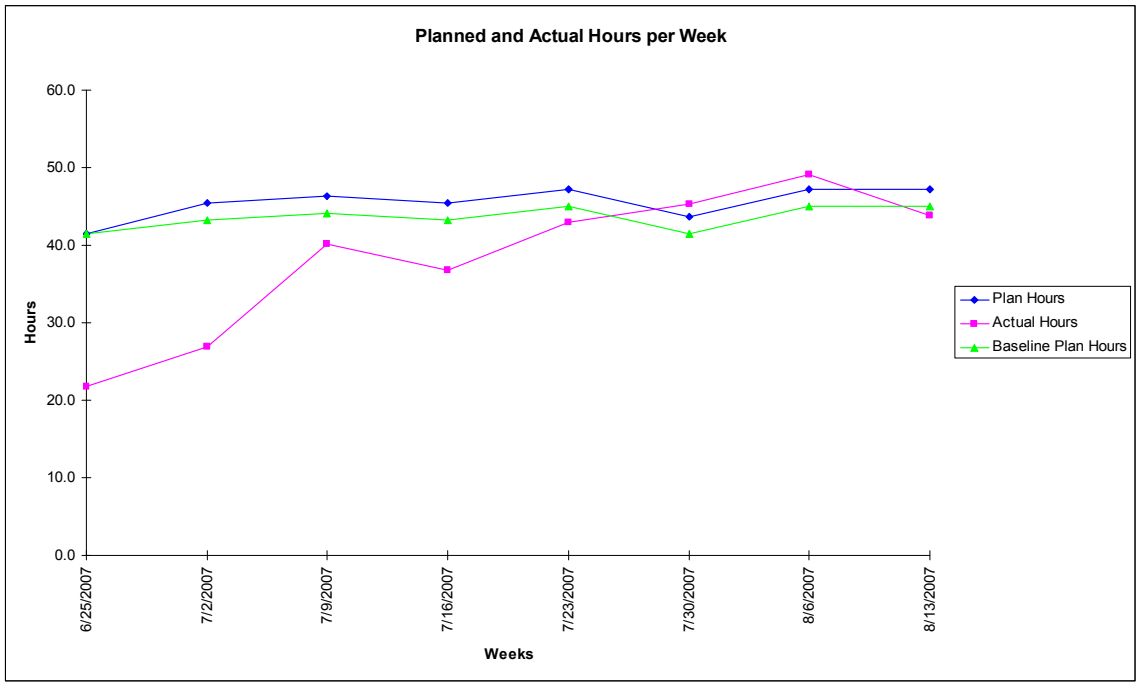

Figure 26: Team Plan vs. Actual Hours Through Week 8

Because scheduled progress depends upon the estimates of product size, production rate, and effort, fewer than planned effort hours usually cause a schedule slip. The following chart shows how scheduled progress fell behind initially but began to recover when the team took corrective actions. This is a typical of new teams; the teams learn to use their data to manage their work.

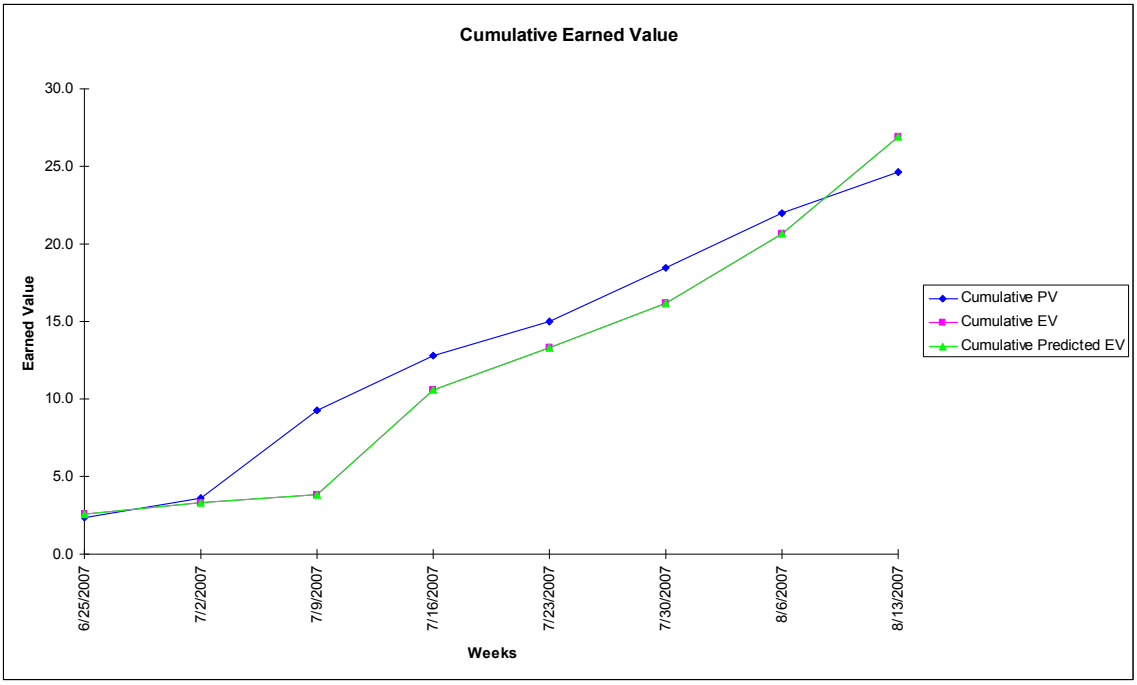

Figure 27: Cumulative Earned Through Week 8

By week 8, the team was actually ahead of the schedule plan. The following table reports status at week 8. Although effort was still behind, the deficit was compensated by a modest overestimate of the work.

Table 7: Status at Week 8

Week 8 Data	Plan	Actual	Plan/Actual
Project hours for this week	47.3	43.8	1.08
Project hours this cycle to date	364.1	306.7	1.19
Earned value for this week	2.6	6.3	0.42
Earned value this cycle to date	24.6	26.9	0.91
To-date hours for tasks completed	365.7	293.1	1.25

The team presented status weekly to management. Later, the project was temporarily suspended to satisfy an urgent request from the same customer. The team launched to plan the new project, completed the work using TSP, then resumed the original project. Only one defect escaped to user acceptance test.

5 Results, Summarized Project Data

5.1 Data Source

The data summarized in this section come from project consolidated workbooks and postmortem reports developed as part of the TSP project activities. The data presented here represent five organizations and nine projects from these organizations. Of these, only one project team had completed more than one project. The results, therefore, are representative of the early stages of TSP deployment where teams, team members, and organizations are still inexperienced. Please note that this is a full population survey of the pilot projects rather than a sample.

There has been no post-selection of the projects reported. The nine projects in the study were done in five companies. These project teams all deliver software as an outsource product. This outsourcing group is distinct from projects that produce either a commercial or internal use software product. Typically, the outsourcing projects have less control of their software development strategies, time tables, and start dates. This proved to be a significant problem in the initial planning and training phase of TSP rollout.

5.2 Context

5.2.1 Status of TSP Introduction

As of August 2008, near the end of the initial phase, there were three companies and four pilot projects underway or completed.

Training included

- 50 managers
- 63 industry software engineers
- 10 software engineering faculty
- 68 software engineering undergraduate students

The certifications and authorizations achieved were

- 159 certified PSP developers (13 undergraduate students)
- 12 authorized PSP Instructors
- 11 TSP coach candidates
- 1 authorized TSP coach

Since August of 2008

- 10 software engineering faculty members have been authorized to teach PSP and TSP.
- Two instructors have been authorized to train additional instructors.
- Three coaches have been authorized.

- As of summer 2008, Mexican coaches are guiding projects at least five companies.
- As of September 2008, Mexico leads the world in the number of certified PSP developers, 160.

5.2.2 Project Characteristics

The data presented here are from a diverse group of organizations, some of which had been appraised at CMMI Level 5. Others had never taken part in a CMMI appraisal.

Product size ranged from 200 LOC to 450,000 LOC; team size ranged from four team members to 21 (over the course of the multi-cycle project). Project duration range was from one month to over a year. Application types include real-time software, embedded software, IT software, client-server applications, and financial software. A variety of programming languages and development environments were used, mostly third- and fourth-generation languages and development environments such as C++, Java, Visual Basic, Access, .Net, and RPG. Because the data set is small, there has been no attempt to segment the data based on project or development characteristics. Instead, all the measures and the range and average of the values are reported for schedule, estimation, and quality.

5.3 Project Results

The project results are compared to benchmark data reported in by Davis and Mullaney in CMU/SEI-2003-TR-014, an earlier TSP results study, and typical projects in the software industry [Davis 2003, McAndrews 2000, Jones 2000, Chaos 1994]. Direct comparisons should be considered with caution because few non-TSP projects gather data as precise or use the same standardized operational definitions as TSP. Also, most benchmark TSP data sources are not population surveys or truly random samples, thus there is the potential for bias. For example, the data reported by Davis and Mullaney are for the self-selected sample of organizations that had chosen to publish their results.

Following the approach adopted by Mullaney and Davis, the benchmark schedule data comes from the Standish Group Chaos Reports [Chaos 1994]. For time-in-phase data we used several sources, including estimation models, data from the NASA Software Engineering Laboratory, and pre-TSP data from some of the organizations we have worked with at the SEI [SEL 1993, Humphrey 2002, Jones 1995a, Jones 1996, Jones 2000]. For quality data we used mostly Capers Jones as the source backed by pre-TSP data from some organizations the SEI has worked with, as well as data from Watts Humphrey [Jones 1995a, Jones 1996, Jones 2000, Humphrey 2002]. Jones uses function points as the size measure for normalizing defects (defects/function point). Since the TSP uses LOC as the default size measure, Davis and Mullaney had to convert function points to LOC using the "backfiring" method described for this conversion [Jones 1995b]. Jones suggests using a default of 80 LOC per function point for third-generation languages, and a default of 20 LOC per function point for fourth-generation languages. The Davis and Mullaney benchmark data used a conservative default value of 100 LOC per function point, as Jones does when discussing non-specific procedural languages.

5.3.1 Schedule Deviation

TSP starts with the best plan the team can produce by using sound estimating and planning methods. The plan is then updated as necessary whenever the team learns more about the work or when either the work scope or resources change. The teams adjust plans based on the individual and team status against the current plan. Schedule is often the most important goal, Plan changes focus on satisfying the most important project goals—often, the schedule. For example, because of the management's constant awareness of plan status, TSP teams can take actions that reduce schedule deviation from the baseline. The schedule data presented in Table 9 show that Mexican TSP teams missed their schedule by an average of 2 percent (equal weight to all projects regardless of size or cost), ranging from 27 percent early to 45 percent late. Compared to the 2003 benchmark (CMU/SEI-2003-TR-014), the results in effort and schedule estimation are similar, though with a wider range. Effort estimation from McAndrews 2000 TSP benchmark had smaller bias and narrower range than either the Mexican or Davis and Mullaney 2003 Benchmark results; schedule deviations were similar in all three groups.

Figure 31compares the schedule performance reported by the Standish group [Chaos 1999] with the Mexican teams, with the data sets segmented by schedule deviation (percent late). The Standish survey results are reported by percent of the sample in each segment. The Mexican data set, (only eight projects) also includes the number of projects in each segment. Mexican TSP results were unlike the industry benchmark because none of the Mexican projects were cancelled during execution. (For business reasons unrelated to project progress, one project was temporarily interrupted while team members were assigned to a more urgent project.) Only two of the Mexican projects (12.5 percent) exceeded 20 percent late with an extreme value of 45 percent late. By contrast, the benchmark sample included 68 percent exceeding 20 percent, with 6 percent more than 200 percent late and 29 percent cancelled.

5.3.2 Quality

One reason TSP teams are able to meet their schedule commitment is that they sharply reduce the time spent in the highly unpredictable test phases. They accomplish this by planning for quality and delivering high-quality products to test. Effort is shifted from test to appraisals (applied in personal review and team inspections) which have relatively predictable cost and defect removal rates. This not only shortens the time spent in test, but significantly reduces effort and schedule variability. The data in Table 8 show that the Mexican TSP teams are delivering software that is roughly a factor of 10 better in quality (measured by defect density) than typical benchmark projects (0.5 defects/KLOC versus 7.5 defects/KLOC). The Mexican results, however, include a factor of 10 more defects than the TSP benchmark sample. Products developed by the Mexican TSP teams have an average of 1.7 defects/KLOC in system test, with three project teams reporting no defects found in system test. The Mexican TSP teams spent an average of 6 percent (projects equally weighted) of their total effort in system test or later test activities; the maximum effort that any team spent in test was 26 percent. Although this has a higher variation than the TSP benchmark report [Davis 2003], the worst performing team in the range is still substantially better than the industry average benchmark average of 40 percent of development time spent in testing.

The average percentage of total schedule (project duration in calendar time) spent in system and acceptance test activities was 6 percent. Typical non-TSP projects routinely spend 40 percent of

both development effort and project schedule in post-development test and rework activities [Davis 2003].

5.3.3 Quality Is Free

A frequent concern expressed about disciplined methods is the perceived adverse impact on productivity. It was not possible to compare productivity before and after TSP introduction because organizations seldom gather data comparable the available from TSP teams.

The data in Table 8 suggest that the Mexican TSP projects improved their schedule predictability and productivity while at the same time reduce their failure COQ (percentage of total effort spent in failure activities) and their total COQ (percentage of total effort spent in failure and appraisal activities). The main reason for this increase in predictability is the reduced time spent in test because of higher quality products being delivered into test.

5.3.4 Comparing Result Summaries

Figure 28 compares reports of results from TSP projects, CMU/SEI-2003-TR-014 and a previous technical report summarizing TSP data from four organizations and fifteen projects [McAndrews 2000]. The data from CMU/SEI-2003-TR-014 represent a more diverse set of organizations than the earlier report (thirteen versus four organizations).

One conclusion that can be drawn from these data is that TSP teams can manage effort deviation while they meet their schedule commitments. The system test defect density, acceptance test defect density, and duration of system test show projects reporting even better quality results than those in the initial TSP report. The better quality may also account for projects meeting schedule commitments despite effort deviations.

Table 8: Quality and System Test Project Metrics

Measure TSP	**CMU/SEI-2003-TR-014 Projects** **Average** **Range**	**Typical Projects** **Average**	**Mexican Phase I Project Average** **Range**
System test defects (defects/KLOC)	0.4 0 to 0.9	15	1.7 0.0 to 6.8
Delivered defects (defects/KLOC)	0.06 0 to 0.2	7.5	0.5 0.0 to 2.2
System test effort (% of total effort)	4% 2% to 7%	40%	5.93% 0.25% to 26.22%
System test schedule (% of total duration)	18% 8% to 25%	40%	6.2% 2.1% to 26.2%
Duration of system test (days/KLOC)	0.5 0.2 to 0.8	NA[7]	5.4 0.4 to 9.5
Failure COQ	17% 4% to 38%	50%	15.2% 1.6% to 29.4%

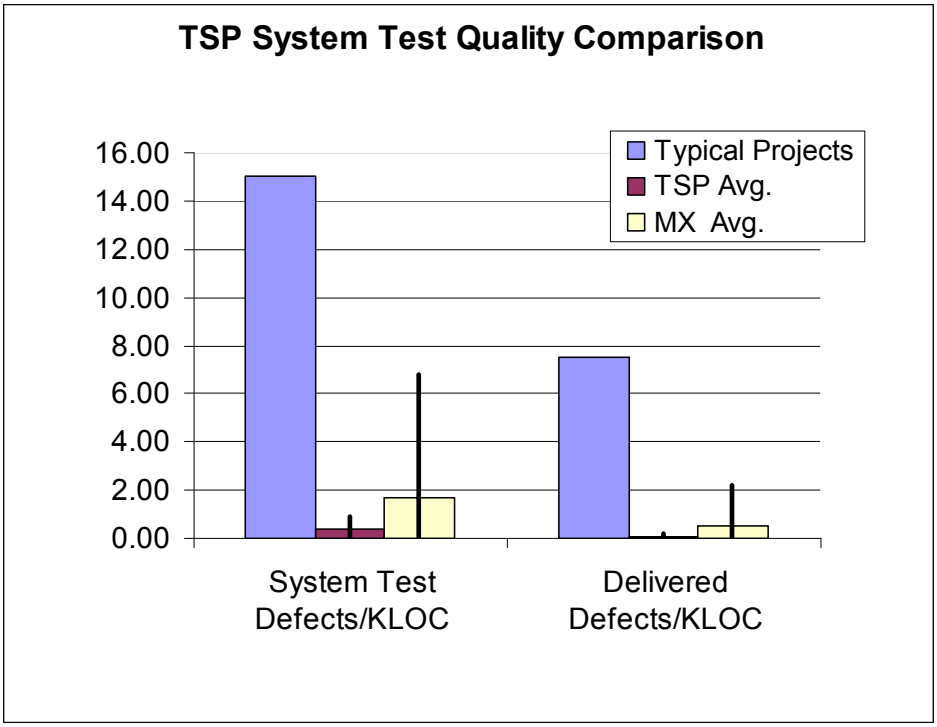

Figure 28: System Test Quality Comparison

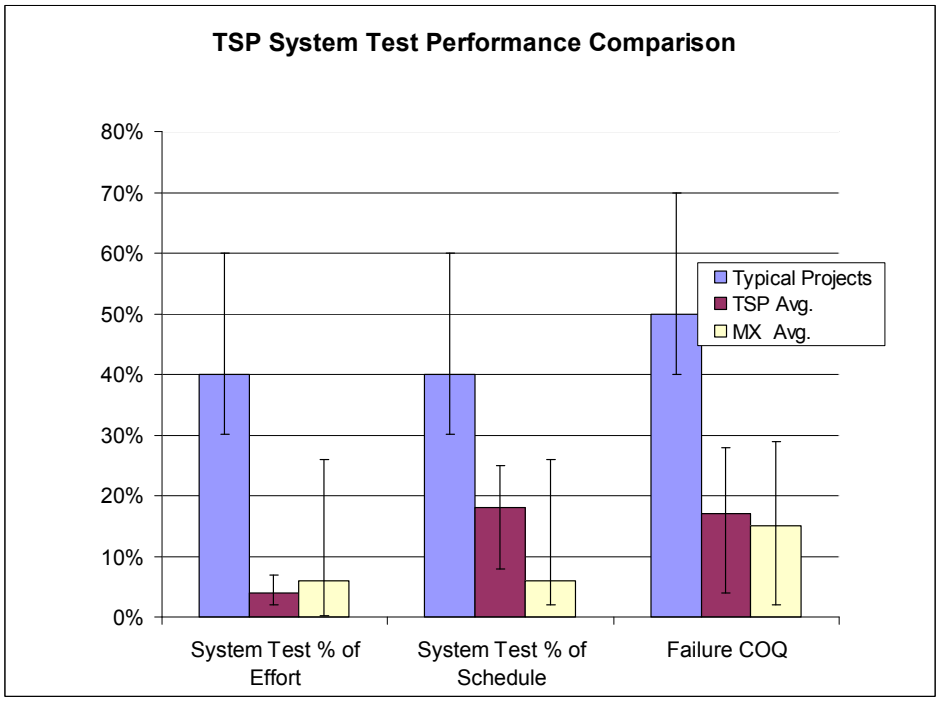

Figure 29: System Test Performance Comparison

Table 9: Cost (Effort) and Schedule Deviations

Measure	TSP Projects Results 2000[8] Average Range	TSP Projects Results 2003[9] Average Range	TSP Projects Mexico Average Range
Cost (effort) error	-4% -25% to +25%	26% 5% to 65%	20% -9.5% to 54%
Schedule error	5% -8% to +20%	6% -20% to 27%	2% -27% to 45%

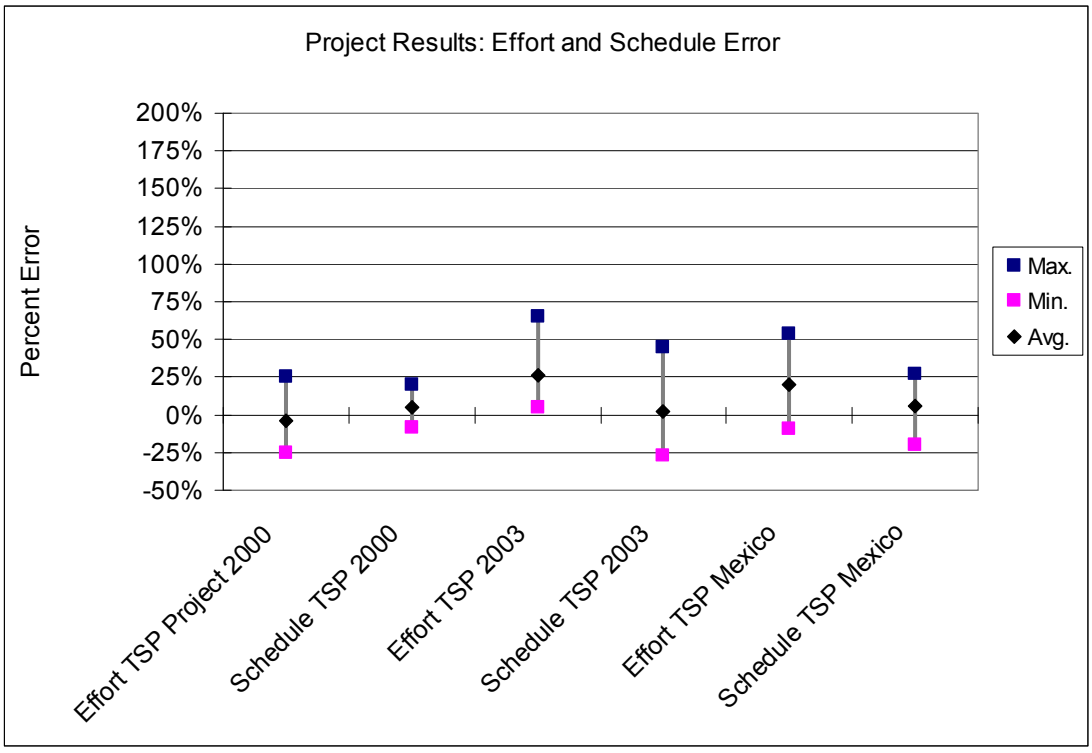

Figure 30: Project Results: Cost (Effort) and Schedule Error

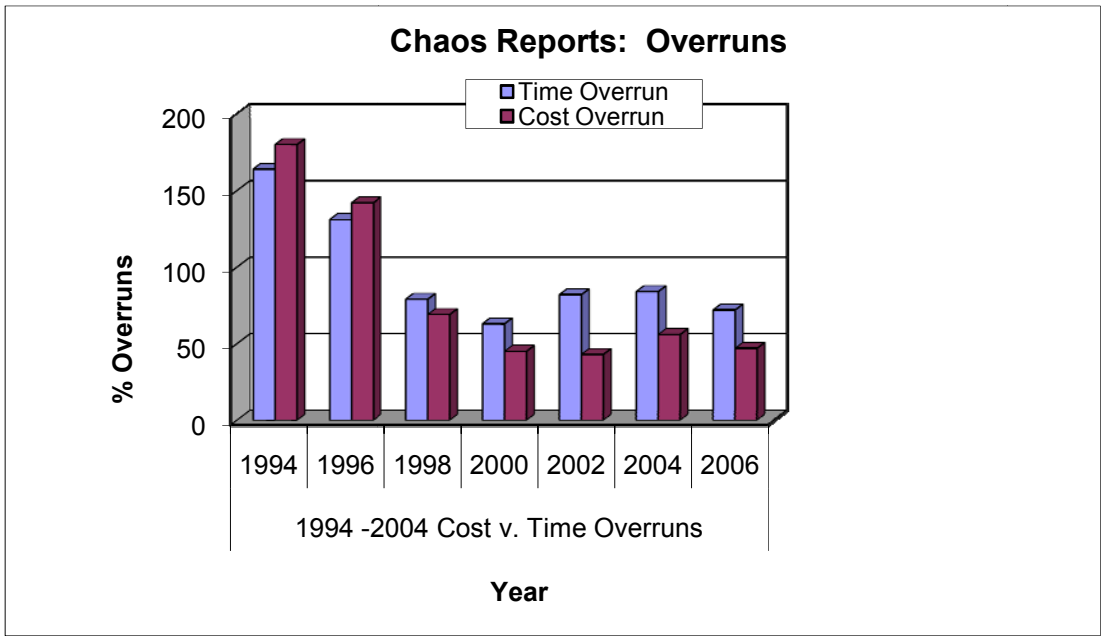

Figure 31: Chaos Reports: Overruns

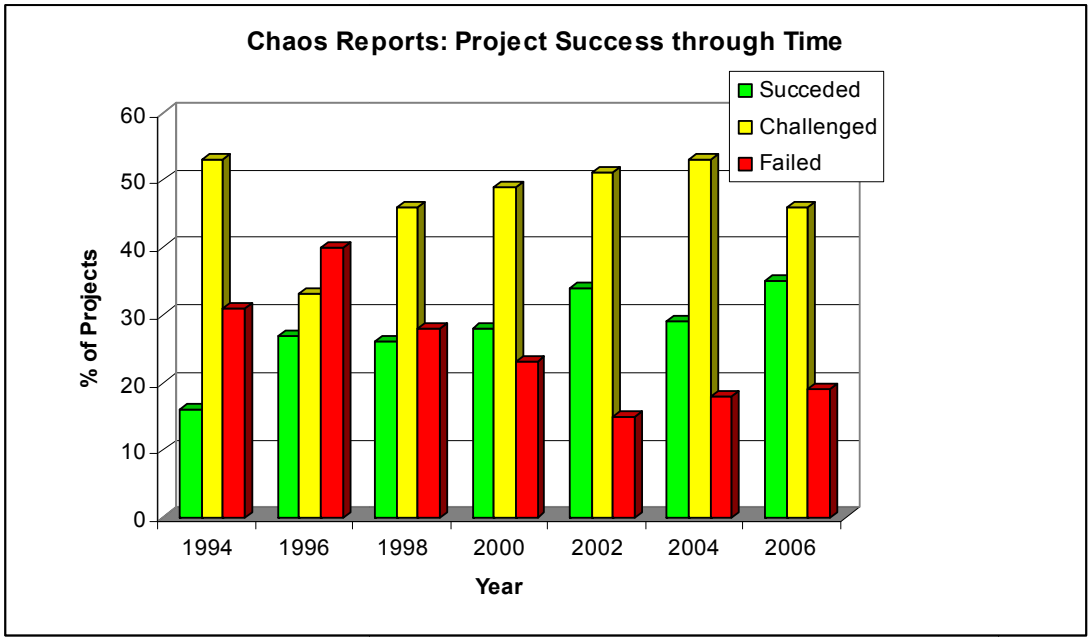

Figure 32: Chaos Reports Over a 12-Year Span

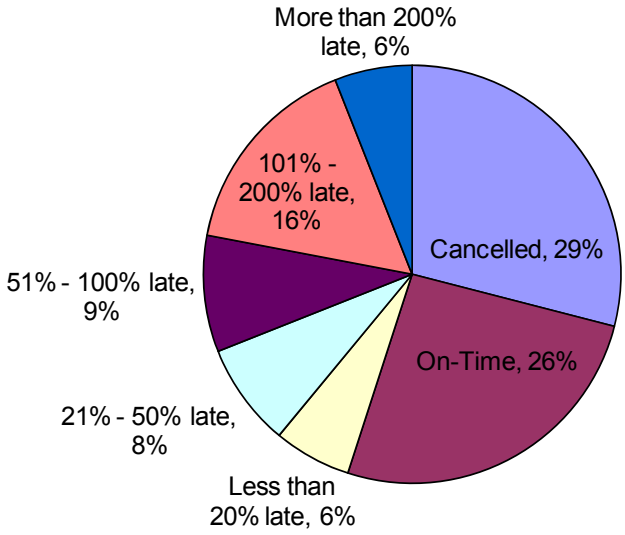

Figure 33: Standish Group Project Success Reports, 1999

5.4 Results Conclusions

The results summarized in this section compare very favorably to typical software projects. The Standish Group reported in 1999 that 74 percent of all projects were not fully successful (see Figure 33) [Chaos 1999]. In 2007, this had changed little; the Standish Group reported that 65 percent of all projects were not fully successful [Chaos 2007]. The Standish Group also reported in 1997 that unsuccessful projects accounted for over half (53 percent) of total spending on software projects [Chaos 1997]. In 1994, the same group reported that for the unsuccessful projects the average cost overrun was 189 percent and the average time overrun was 222 percent. For 2006, the Standish Group reported average cost overruns of 47 percent and schedule overruns of 72 percent. Typical projects spend 40 to 60 percent of total project time on test, and typical defect densities of delivered products range from 1 to 10 defects/KLOC [Humphrey 2002]. The Standish Group Chaos reports illustrates this overall performance record between 1994 and 2006 (see Figure 32).

By contrast, the worst-performing Mexican projects overran cost (measured by effort estimates) by 54 percent and schedule by 45 percent. This is remarkable because most of the Mexican TSP teams used inexperienced developers and TSP for the first time. Likewise, most of the managers and coaches also used TSP for the first time.

When the coaches were inexperienced, they received guidance and support from the SEI. Since TSP teams have typically improved cost, schedule, and quality performance with experience, future projects by these same teams are likely to produce even better work. In terms of international

competitiveness, it also means that once Mexico has built a foundation of experienced TSP teams, it will be ahead of other countries on its improvement learning curve and will be exceedingly hard to catch.

Although the defect density of the Mexican projects was higher by a factor of 10, roughly, than the TSP projects reported in CMU/SEI-2003-TR-014 [Davis 2003] in the average density and range, these projects had lower defect densities—again by about a factor of 10—than typical projects in the industry benchmark group. The highest reported defect density compared to that of a typical CMMI level 4 organization; the average performance compared to that of CMMI typical level 5 organization [Jones 2000].

Although effort estimation was comparable to the 2003 benchmark group and somewhat worse than the earlier TSP sample, schedule performance was similar to the TSP groups. This suggests that TSP teams are able to recover from planning errors through active management of their projects.

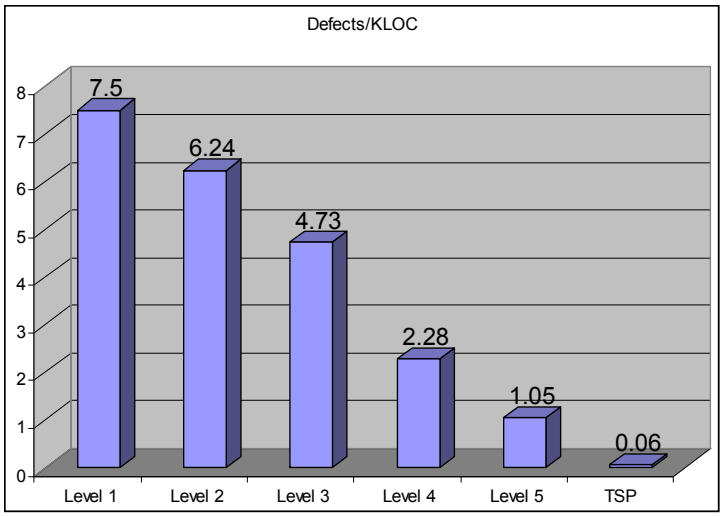

Figure 34: Average Defect Densities at CMMI Maturity Levels

A remarkable aspect of this data survey is that a diverse group of organizations using TSP was able to use common operational definitions for measures reported. For example, when projects report defect density, it is understood that they are talking about number of defects found per thousand lines of new and changed code only. Effort hours are reported; only on-task hours are measured. This includes a common language for project management—terms such as yield, cost of quality, earned value, task hours, and defect density. All have the same meaning across projects and organizations. Common operational definitions of measures and common project management language are both results of using TSP. This enables objective comparison of organizations and project performance by acquirers of software services.

6 Anecdotes

Although the data speaks for itself regarding the project outcomes, data does not speak for the people involved. Operational processes and quantitative data are essential, but they must be implemented by the development staff. The development staff must follow the process, gather, and use the data. To gather and use the data, development staff must be motivated. TSP assumes that the developers prefer to do good work and good work is enabled by giving self-directed teams the ownership and responsibility for their process.

Software development is an intellectually intensive activity, with the software created by individuals who work within teams within the organization. High-performing projects require that individuals, teams, and organizations perform in a highly cooperative, high-communication, and high-trust environment. While TSP may seem to be only data driven (for example, the decision and planning activities) it relies upon individual motivation, human interaction, and team commitment to plan and manage the work. Included in this section are comments from the participants in the pilot projects, both positive and negative. The stories and comments from the people involved provide a context for this pilot effort, that is, how the people were affected. Positive stories illustrate the qualitative benefits TSP teams have been able to achieve, while negative stories can provide lessons learned and guide future improvement efforts.

6.1 Data Source

The stories and comments in this section of the report come directly from the people involved and their data. One source is the evaluation forms and reports completed after each launch or relaunch. The TSP coach and all team members complete evaluation forms and send them to the SEI. A second source is the project post mortem conducted at project cycle completion or at project completion. This post-mortem data is also submitted to the SEI. A third source is pilot checkpoints. Checkpoints are conducted by a TSP coach in the middle of a development cycle. The purpose is to evaluate the effectiveness of TSP implementation. Team members and their management are interviewed and their comments are collected during the checkpoint.

While we would prefer anecdotes to be presented verbatim from the individual, many of the comments were originally supplied in Spanish. In the translation to English, some paraphrasing was unavoidable. Many comments were provided in English by non-native speakers. Grammar and syntax errors were not corrected. So while some of the statements are hard to interpret, we have provided the actual text so the reader can make whatever interpretation seems most appropriate. Also, to preserve anonymity, some editing was done to remove any indication of the comment source.

6.2 Participant Comments

This section includes comments from project stakeholders.

6.2.1 Project Stakeholder Comments: Working on a TSP Team

With TSP monitoring of the project is much more orderly. Everything takes control and mistakes are more easily, before delivering the product.

With TSP work is much more organized and gives a better vision about what is being done and what needs to be done.

I personally prefer to work in a TSP team, because I have tried many methods, philosophies and recommendations, and all of them, this framework I have personally validate that it really works, and that gives excellent results because it is based on sound science.

Recording and following up with TSP helps me make better plans.

The methodology has helped me to revise commitments with the customer.

I prefer to work in a TSP team rather a traditional one.

Overall the team prefers to work in a team that TSP traditional one.

6.2.2 Project Stakeholder Comments: Managing Quality

Improvements were achieved, good quality products, by the simple fact that inspections found many mistakes early and none of these mistakes had reached the customer.

Overall there is higher quality products that before applying the methodology.

I'd never used review and inspections before, but now I can see how useful they are. The product quality is higher.

6.2.3 Project Stakeholder Comments: Realistic Plans and Schedule Performance

With TSP there is greater integration of the team, better product quality, better administration, certainty and predictability.

Do not underestimate activities and better manage the "small changes" because they represent a lot of time together that we have to absorb between the project schedule and/or personnel.

I think we can raise the number of Task Hours if planned with a greater degree of detail, detect those activities outside the plan, without control of time, without agenda, and so on.

Should also budget a time for research, testing concept, prototyping and develop “best practices.”

Should begin using actual statistical information.

I think we could raise the number of productive hours in the project if we could have fewer distractions from other projects, take all necessary materials on hand (software, project report).

There is lacking a process of change control, because if there were changes and are not recorded through a process that would allow evaluation.

They presented the results of the first pilot to management and they were impressed because the level of data and information the TSP/PSP team was able to provide.

As TSP is very rigorous with the management of time, which promotes the management discipline to meet productively with the work schedule (arriving on time and finish the job planned with the quality expected within working hours) and is an example in this sense.

6.2.4 Project Stakeholder Comments: Work Environment

It was good use of Source Safe but we need to formalize the plan as to the configuration management baselines.

The development environment in general is good. But the way the code is handled on the server client made me lose a lot of time for the slow and communication failures in the system (DTR). Another problem was the lack of SQL server in order to install the Workplace NW.

6.2.5 Project Stakeholder Comments: Barriers to Success

We need more full-time team members and should respect the team's initial estimate, and even when it appears high.

Put more attention to implementing the process.

Need to arrive early for launch meetings to achieve the timetable

Work their plan together with the client so that it can be longer with the team and can give support.

We need to plan together with the team to make to be more productive and within working hours.

Team lead needs to be more involved with the team to find out the problems of everyday life and.

The team leader must know how to harness the skills, talents and time for the team.

Remove the Excel spreadsheet that (TSP) asks us to carry to record our time because we are distracted, is cumbersome having to be sending 2 times a day, it is highly impractical and is also collides with the tool and Excel TSP.

The team lead needs to plan together with their team to make us productive and within working hours.

We have had some budget troubles so we have had to delay some expenditures like the training for a PSP Instructor and a TSP Coach. This has caused us some delays in the deployment.

We need to arrive early and meet the launch meeting timetable.

6.2.6 Project Stakeholder Comments: General

We must continue what has been achieved, to improve aspects that we adversely affected during the project and that we continue the process taking at least the same level with which we carry this cycle.

6.3 Anecdote Conclusions

The comments presented in this section show how the TSP introduction strategy builds skills and prepares a team for using the TSP, how the launch creates a cohesive team that is committed to the team goals, how the team develops a plan with realistic and achievable schedules, and how teams focus on quality throughout the project life cycle. Some problems faced by TSP teams are also described. The comments illustrate how people internalize their experiences with the TSP development process.

Common concerns are leadership, TSP tool support, and fidelity to the process. Common fidelity issues include having everyone present during the launch, allocating adequate time for the meetings, and trusting the team to produce a good yet aggressive estimate. Other issues include lack of sufficient skills to accomplish the task. In either case, the team members are taking ownership of the process and identifying opportunities for improvement.

The positive comments demonstrate that the developers recognize how a structured process helps them to perform their work in an orderly, efficient, and high-quality manner. Comments reflect both the effectiveness of technical activities such as personal reviews, and the benefits of the personal interaction and communication activities. An overwhelming majority of the personnel preferred to work this way.

7 Lessons Learned

The TSP introduction in Mexico was similar to previous experience in several very important ways:

- management support is crucial
- TSP developers really like to do good work
- TSP works

On the other hand, experienced coaches offered observations about how Mexico may differ:

- Compared to their U.S. and European counterparts, Mexican TSP developers initially seem less inclined to discuss or argue; rather, they are more inclined to try, without question, during PSP training and early TSP launches.
- TSP developers seem generally to need more consistent feedback, ongoing coaching, and encouragement.
- Mexican managers are often younger and more directive than their U.S. counterparts.
- Mexican development staffs tend to have fewer senior developers who can lead by example.

Critical success factors for this effort include having leadership support, management support, coaching support, and customized training.

Willing and effective participants, particularly project and middle management support, were essential to successful adoption. It is helpful not only to set aggressive goals, but also to establish a reward and recognition system for meeting commitments and following the processes.

Many team members can function in written English but have more difficulty with spoken English. We need to supply Spanish-speaking TSP coaches.

We learned that effective and timely coaching, especially at the beginning, is crucial to project and TSP adoption success. Coaches need to do the following:

- Assist individuals and teams in a timely manner. The coach must available to answer questions provide clarifications, and assist interpreting data.
- Spend significant one-on-one time with project participants. Remote access tools, such as instant messaging, can help coaches stay connected to the team.
- Provide TSP tool instruction and support.
- Facilitate both project outcomes and process adoption. There will always be contention among competing project goals, while the requirements and resources change frequently.
- Assess a team's progress on the "quality journey" and push the team to keep improving. The quality results of these projects, although good by industry standards, leave room for improvement.

- Provide pragmatic judgment about when to be flexible and when to hold the line on TSP principles. TSP should be followed to the letter only when it makes sense to do so. Recognizing when guidelines should be flexible or tailored requires experience.
- Assist in carefully selecting pilot projects and facilitating their success.
- Participate in weekly team meetings, especially during the initial weeks of the project.

Mexican organizations are more likely than U.S. or European firms to be outsource providers. The business of outsourcing presents challenges. First, it can be difficult to identify a pilot project because once the project is awarded, work must start immediately. Identifying and training the project team can, therefore, be impractical. Second, the reality of maintenance work is that it is difficult to keep teams together while supporting legacy projects. Third, since companies are growing, engineers, leads, and coaches get promoted to more senior positions. It is thus difficult to keep development teams together.

A good training experience is essential for success. All team participants—engineers, non-engineers, team leaders, as well as the leadership team—must have appropriate training. Team members who were not fully trained had a difficult time appreciating the value of recording time, using the data, and following the defined process.

Overall completion rates for the training have been only around 50 percent. Because of financial and business pressure, companies resist providing the recommended time for the training activities. Course participants with insufficient time do not complete coursework. One way to address this difficulty would be to develop training that reduces the time and expense prior to the first team launch to get to the TSP launch. This was the intent of the "PSP Fundamentals" class. The teams trained using "PSP Fundamentals" had successful launches and projects. "PSP Advanced" should provide the skills they will require for process improvement prior to a relaunch.

In this training model, "PSP Fundamentals" provides the skills necessary to function on a team, and track and record data. When delivered shortly after the first development cycle, the "PSP Advanced" course provides skills needed for data analysis and process improvement. This reduces the time required prior to launch, distributes the training over a longer period, and provides analysis training when it is needed. Early results from "PSP Fundamentals" have been promising. Results from following this with "PSP Advanced" are not yet available. Alternate introduction models and piloting should be developed and piloted, and the successful approaches should then be deployed.

Two distinct types of training are needed for effective teams. PSP training is provided in class to learn specific technical concepts and skills. PSP is not, by itself, sufficient. TSP skills are learned on the job through coaching and job experience. The TSP skills demonstrate practical application of the skills and the social team work skills needed to manage and collaborate on a project. PSP has a certification examination; however, there is no TSP developer certification. Experience indicates that team members typically need about 15 weeks or about 225 task hours or experience to be fully effective. Managers may not recognize this need for experience, coaching, and mentoring, in addition to the class training. For example, task hour management is a problem for almost all new teams and team members. The example provided in the section describing a first-time team was typical. With adequate coaching and experience, teams quickly learn how to manage their work.

With many projects being of short duration with small teams and starting on short notice, it was difficult to find two pilots to run in parallel. It was also found that developing coaches and instructors was usually not a company priority. Despite success, the effort would not be self sustaining. One approach used to address this was to launch two projects using the same team in sequence. During the first project an internal coach can be trained so that he or she can become a coach for second pilot. During this time, the experienced coach mentors the new coach.

8 Next Steps

8.1 Diffusion of Innovation and Crossing the Chasm

Everett Rogers described the diffusion of technological innovation as following an "S" curve [Rogers 2003]. He characterized people willing to use a new technology based on the maturity of the technology and how widely it is used. The population in each category follows the familiar "bell curve" or normal distribution. New technology is adopted sequentially by innovators (2.5 percent), early adopters (13.5 percent), early majority (34 percent), late majority (34 percent) and laggards (16 percent). The scale of the technology adoption, therefore, grows as the cumulative of the normal distribution. The shape of the cumulative distribution resembles an "S" and, unlike the more familiar normal or bell curve, grows very slowly initially then very rapidly, eventually dipping as the innovation becomes institutionalized.

Geoffrey Moore used this model to discuss the problems of marketing new technologies [Moore 1999]. We believe PSP and TSP are in the Early Adopter portion of this model curve. That is, the initial phases of the Mexican TSP initiative focused on the innovators and Early Adopters.

8.1.1 Innovation Adoption Models Applied to TSP in Mexico

In Mexico, institutionalizing TSP adoption will require expanding use among Early Adopters, then "crossing the chasm." The chasm represents the barriers preventing the more pragmatic Early Majority from adopting the technology. That is, the Early Majority expect the product to be mature, usable, and demonstrated to be suitable for solving their practical problems. The barriers, and their implications on how to proceed with the TSP deployment, will be discussed.

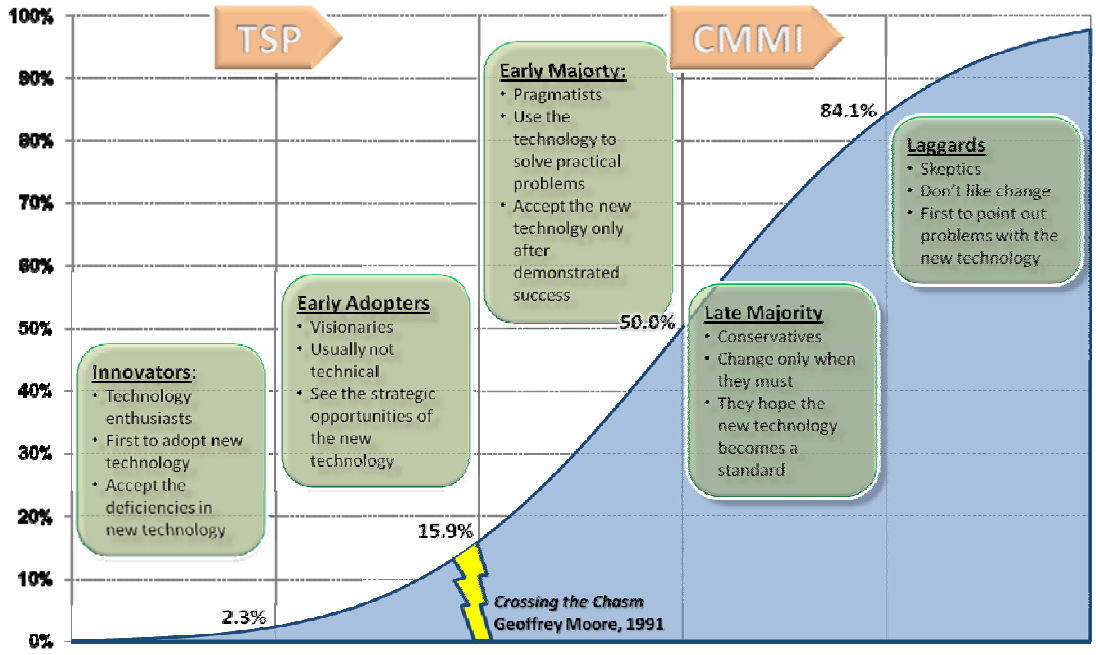

Figure 35: Model for Diffusion of Innovation

The S-curve can be used to project the resources required to satisfy the Prosoft-stated goals. The initial stage of adoption is characterized by constant but slow growth. As the technology becomes more widely adopted, the growth accelerates. The chasm occurs in the early stage of rapid growth. If the chasm is crossed, rapid growth is followed by saturation. For example, assume the following parameters (these numbers are approximate and used for illustration):

- Start: January 2008
- Time periods of four months
- Begin accelerating growth in the 7th period (May 2011)
- End of growth in period 21 (January 2016)

Overall goals for 2013:

- USD $3.5 billion of exports using TSP for the project management
- Median Project = USD $90,000
- Total projects in 2013 = 24,017

Table 10: TSP Support Needed to Satisfy Prosoft Goals

	2008	2009	2010	2011	2012	2013
Companies Using TSP	18	50	152	379	593	648
Projects Using TSP	57	302	1,543	6,642	16,872	24,017
TSP Trained Software Engineers	241	1,334	6,631	22,163	37,690	42,987
Instructors Required	10	10	17	67	164	164
Coaches Required	7	26	130	480	886	1,010

Table 10 represents the TSP support needed to satisfy Prosoft goals. TSP requires resources and assumed conditions such as trained engineers and coaches. As deployment proceeds to later stages, supporting the scale of the deployment and the expected high rate of growth becomes a significant consideration. This cost may limit the number of companies able to afford the up-front cost. The limiting factor is likely to be the number of companies able to fund the introduction effort.

8.2 Next Steps to Prevent Skills Shortages

The number of TSP teams that can be supported can be limited by the supply of

- PSP trained engineers
- PSP instructors
- TSP coaches

It is critical to note that not only the quantity of trained and credentialed people is at issue, but also that quality must be maintained. TSP relies on faithful implementation and discipline. TSP fidelity must not be sacrificed to achieve deployment goals because high performance correlates with the faithful practice of TSP. Proper training of engineers, coaches and instructors is necessary for assuring ongoing quality of the deployment. It is worth noting that while this projection is for TSP, any initiative dependent on human capital will face similar training and support problems during the scale-up period.

Next steps to mitigate skills shortages are

1. training PSP developers while they are students at the universities
2. credentialing Mexican university professors to deliver PSP and TSP classes
3. establishing Tec as a TSP strategic partner

8.2.1 Preparation of PSP Developers In the Universities

Training developers as part of their university education will significantly reduce the start-up costs on the part of SMEs. This in turn requires trained, credentialed university faculty. To date, faculty in Monterrey (Tec de Monterrey, Universidad Regiomontana, and Universidad Monterrey

in Monterrey) Guadalajara, Chihuahua, and Zacatecas have been authorized as PSP instructors. An important step is to expand faculty training and credentialing without sacrificing the quality of instruction received by the students.

8.2.2 The TSP Coach Bottleneck

Presently the availability of properly trained and credentialed coaches is a bottleneck. As can be seen by reviewing Table 10 this problem can become dramatically worse as the rate of TSP adoption accelerates. Although Tec can train both instructors and coaches, presently only the SEI can observe and authorize coaches. Tec is becoming a strategic partner with the SEI. As such qualified Tec coaches will be authorized as mentor aspiring coaches. The mentor coach, under appropriate supervision by the SEI, will train and authorize the additional Mexican coaches who are needed for the rapid growth phases of the initiative. Theoretically the same issues could apply to the preparation of PSP instructors but in practice that is not expected to be a problem.

8.3 Next Steps to Improve Market Acceptance

TSP works for the Mexican companies where it has been deployed, as is seen in Section 5. Deploying on a national level, however, is both challenging and unprecedented. In addition to the practical and scaling problems of the rollout, national success depends on visibility and recognition of the accomplishments.

Next steps for branding TSP in the marketplace include:

1. leveraging the international recognition of CMMI
2. certifying and recognizing companies that effectively use TSP
3. promoting the use and results of TSP

8.3.1 Leveraging the International Recognition of CMMI

There is no doubt that successful CMMI Maturity Level appraisals are an important way that a business signals its readiness to be a trusted software development partner. The best way to tie TSP to the brand recognition of CMMI is the use TSP implement CMMI. Using TSP as a path to CMMI accomplishes several purposes. First, TSP provides a cost-effective way to implement CMMI practices and evaluate CMMI maturity. Second, TSP has been successfully implemented on shorter timescales than are typical for CMMI. Third, TSP is especially useful to the SMEs. Fourth, CMMI maturity ratings provide widely respected recognition of the Mexican commitment to and accomplishment of process and quality improvement. TSP can, therefore, help address some of the scaling and cost issues that come with CMMI deployment. Because TSP, when faithfully practiced, brings very high performance, using TSP to implement CMMI will also boost the reputation of CMMI itself. To date the SEI has developed extensions to the standard TSP so that CMMI implementation is facilitated in the TC-AIM (TSP CMMI–Accelerated Implementation Method) project which has been funded, in part, by Prosoft.

8.3.2 Certifying and Recognizing Companies That Effectively Use TSP

Certifying organizations for TSP provides an easy way for the marketplace to recognize the organizations that do a good job of practicing the TSP. We enumerate four distinct benefits to this next step, which is named TSP organizational evaluation and certification (TSP-OEC):

- monitor organizational TSP fidelity and performance
- advertise both process commitment and the actual performance results
- differentiate Mexican companies in the international market
- verify that Prosoft funds are appropriately spent

8.3.3 Promoting the Use and Results of TSP

All of the above contribute to the promotion of the use and results of TSP. However more is needed. The SEI and Mexico plan to work closely together to feature the achievement of Mexican companies and Mexico. This will be done through SEI publications and events.

8.4 Next Steps to Address Additional Issues

Three other, self-explanatory next steps include:

1. translating TSP/PSP materials into Spanish from English
2. adjusting training so that SMEs do not need to shut down production during training
3. collaborating with or overcoming competing process improvement initiatives

Table 11: Next Steps and Their Impact Areas

Next Step/Issue	Skills	Fidelity	Marketing	ROI	Other
University training for engineers	X	X		X	
Credential professors	X	X		X	
Strategic partner	X	X		X	
Leveraging CMMI			X	X	
Certify organizational TSP fidelity		X	X		
Promote results			X		
Materials translation					X
Adjusted training	X				X
Competing initiatives			X		X

9 Conclusions

This report begins by summarizing goals for the Mexican software industry and how TSP can help achieve world-class status. This is followed by an example of the training and work experiences of a first-time team to demonstrate how the TSP creates an environment where skilled engineers can apply disciplined methods to achieve schedule and quality goals. This team was able to achieve impressive results on their first use of the TSP. These results were typical, as seen by data summarized in the Results section and other TSP experience reports. The individual perspectives provided in Anecdotes section illustrate that individuals are motivated and prefer to work this way. People like doing excellent work and the TSP enables them to do so. In Lessons Learned we discussed problems specific to Mexico and training. In Next Steps, we discussed near-term tasks and long-term strategy for achieving Mexican goals. While PSP and TSP may appear to be primarily planning-driven, data-oriented technologies, it is the human interactions and commitment enabled by TSP that allow individuals and teams to be successful. This reflects both the tension and synergy between disciplined and creative work. Although some might feel that discipline inhibits creative work, in fact the opposite is true. The discipline is required to perform truly superior work on time and within budget. The same holds true with teamwork and data. In order for a team to jell, they need the data to manage their routine tasks.

That each of the projects presented in this report succeeded was remarkable. Each project faced problems and challenges that were recognized early by the teams, using their data. Early corrective actions mitigated the problems so that project goals were achieved. Project and product success resulted from the knowledge, awareness, and depth of commitment of the team members. Ultimately, TSP projects succeed by building superior project teams composed of trained and committed individuals.

Thus far, substantial effort has been applied to implementing TSP in Mexican software development outsourcing companies, formulating the TSP organizational certification, developing strategic TSP capabilities in Mexico through Tec, and planning the widespread adoption of TSP nationally. The pilot organizations and projects demonstrated that TSP successfully helps Mexican projects to deliver high quality products on time and within budget.

For organizations acquiring software services, the TSP teams have demonstrated the ability to deliver software that is nearly defect free, with full functionality, on committed schedules. The performance and success of these first-time TSP teams demonstrates that organizations with some TSP experience should be expected to introduce TSP with additional teams and have better than world-class results on their first TSP project.

References/Bibliography

URLs are valid as of the publication date of this document

[Chaos 1994]
"CHAOS '94 – Charting the Seas of Information Technology." The Standish Group International, Inc., 1994.

[Chaos 1997]
"CHAOS '97 – The Changing Tide." A Standish Group Research Note. The Standish Group International, Inc., 1997.

[Chaos 1999]
"CHAOS: A Recipe for Success. Project Resolution: The 5-Year View." The Standish Group International, Inc., 1999.

[Chaos 2007]
"CHAOS Report 2007 – The Ten Laws of Chaos." The Standish Group International, Inc., 2007

[Davis 2003]
Davis, N. and Mullaney, J. *The Team Software Process (TSP) in Practice: A Summary of Recent Results* (CMU/SEI-2003-TR-014, ADA418430). Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tr014.html

[Ferguson 1999]
Ferguson, P.; Leman, G; Perini, P; Renner, S.; & Seshagiri, G. *Software Process Improvement Works!* (CMU/SEI-99-TR-027, ADA371804). Software Engineering Institute, Carnegie Mellon University, 1999.
http://www.sei.cmu.edu/publications/documents/99.reports/99tr027/99tr027abstract.html

[Hayes 1997]
Hayes, W. & Over, J. W. *The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers* (CMU/SEI-97-TR-001, ADA335543). The Software Engineering Institute, Carnegie Mellon University, 1997.
http://www.sei.cmu.edu/publications/documents/97.reports/97tr001/97tr001abstract.html

[Humphrey 2002]
Humphrey, Watts S. *Winning with Software: An Executive Strategy*. Addison-Wesley, 2002

[Jones 1995a]
Jones, Capers. *Patterns of Software Systems Failure and Success*. International Thomson Computer Press, 1995.

[Jones 1995b]
Jones, Capers. "Backfiring: Converting Lines of Code to Function Points." *IEEE Computer 28*, 11 (November 1995): 87-88.

[Jones 1996]
Jones, Capers. *Applied Software Measurement*. McGraw-Hill, 1996.

[Jones 2000]
Jones, Capers. *Software Assessments, Benchmarks, and Best Practices*. Addison-Wesley, 2000.

[McAndrews 2000]
McAndrews, D. *The Team Software Process SM: An Overview and Preliminary Results of Using Disciplined Practices* (CMU/SEI-2000-TR-015, ADA387260). Software Engineering Institute, Carnegie Mellon University, 2000.
http://www.sei.cmu.edu/publications/documents/00.reports/00tr015.html

[Morgan 1993]
Morgan, Ben B., Jr.; Salas, Eduardo; & Glickman, Albert S. "An Analysis of Team Evolution and Maturation." *Journal of General Psychology 120*, 3: 277-291.

[Moore 1999]
Geoffrey A. Moore. *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customers, Revised Edition.* HarperBusiness, 1999

[Prosoft 2008]
United Mexican States. *Programa de Desarrollo del Sector de Servicios de Tecnologías de*, March 2008.
http://www.canieti.com.mx/assets/files/972/AGENDA%20PROSOFT%202.0.pdf

[Rogers 2003]
Rogers, Everett M. *Diffusion of Innovations, Fifth Edition.* Free Press, 2003.

[Salazar 2008]
Preparing Undergraduate Students for Industry's TSP Needs.
http://www.sei.cmu.edu/tsp/symposium/2008/Monday%20115PM.pdf

[SEL 1993]
Condon, S.; Regardie, M.; Stark, M.; & Waligora, S. *Cost and Schedule Estimation Study Report* (Software Engineering Laboratory Series SEL-93-002). NASA Goddard SpaceFlight Center, 1993.

[Wall 2004]
Wall, D. S., & McHale, J. *Mapping TSP to CMMI* (CMU/SEI-2004-TR-014, ADA441876). Software Engineering Institute, Carnegie Mellon University, 2004.
http://www.sei.cmu.edu/publications/documents/04.reports/04tr014.html

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	March 2009	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Deploying TSP on a National Scale: An Experience Report from Pilot Projects in Mexico	FA8721-05-C-0003

6. AUTHOR(S)

William R. Nichols, Rafael Salazar

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2009-TR-011

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2009-011

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The purpose of this report is to communicate status, progress, lessons learned, and next steps for the Mexican Team Software Process (TSP) Initiative, a collaboration between the Software Engineering Institute (SEI) and Instituto Tecnológico y de Estudios Superiores de Monterrey (Tec de Monterrey), sponsored by the Mexican Ministry of Economy through Prosoft (the Program for the Software Industry Development). The initiative seeks to improve the standing of the Mexican software industry through the process and performance improvement benefits of the SEI's Team Software Process. We will discuss the results through Phase I and some early results from Phase II; performance results to date are compared to TSP and industry benchmarks.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Team Software Process, TSP, process improvement, CMMI	89

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL