The Team Software Process^SM (TSP^SM): An Overview and Preliminary Results of Using Disciplined Practices

Donald R. McAndrews

November 2000
(Revised October 2001)

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

The Team Software Process^SM (TSP^SM): An Overview and Preliminary Results of Using Disciplined Practices

CMU/SEI-2000-TR-015
ESC-TR-2000-015

Donald R. McAndrews

November 2000
(Revised October 2001)

Team Software Process Initiative

This report was prepared for the

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.

FOR THE COMMANDER

Joanne E. Spriggs
Contracting Office Representative

Table of Contents

List of Figures

List of Tables

Abstract

The Software Engineering Institute has developed the Team Software Process[SM] (TSP[SM]) to help engineering teams more effectively build software-intensive products. The TSP addresses many of the current problems of building software-intensive products and shows teams and managers explicitly how to address these problems.

This report describes the TSP technology as an implementation strategy for teams that are attempting to apply disciplined software process methods. It provides some of the background and rationale for the TSP approach, as well as an overview of the technology. Then, the report presents initial results of the use of the TSP technology in four different organizational settings. In each of these organizations, the data show that defect densities found in system-level test activities and the actual duration of these system-level tests were reduced significantly with the use of the TSP. In addition, the accuracy of software estimates improved, and the variation in estimation accuracy was significantly reduced. Based on the analysis of these results, some assertions are made to help organizations set goals for improvement.

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

Acknowledgements

This report describes a technological process that has evolved over time through the work of many people. I thank those people involved because it is their work that is being presented.

First, the Team Software Process[SM] (TSP[SM]) technology, and this report, are products of the TSP Initiative at the SEI. Special thanks go to Jim Over and Dan Burton, original members of the TSP Initiative, as they have contributed to the maturity of the TSP technology significantly in their work with users of the technology. Their dedication and hard work have provided the tools, processes, and training materials we use today. Special thanks also go to the other original members of the TSP Initiative as they helped support this evolution as well: Janice Ryan, Marsha Pomeroy-Huff, and Gwen Moulton. Janice has played a key role in communicating with the network of authorized instructors of the Personal Software Process[SM] (PSP[SM]), publicizing the message, and gathering lessons from the field. Marsha has provided the design expertise for the training materials and documentation. Gwen has helped with administrative, editing, and general team support. Finally, special thanks go to the newest members of the TSP Initiative: Jim McHale, Julia Mullaney, and Noopur Davis, who have helped carry the TSP torch and reviewed this report.

In addition, I thank Dave Zubrow and Jim McCurley of the SEI Software Engineering Measurement and Analysis Initiative for their support in analyzing the data and reviewing the report. I also thank Bill Peterson for his support and review comments, and Bill Thomas and Bryce Walat for their editing support.

Another group who deserve credit are the industry teams who have used the TSP and helped us evolve the technology. Although there are many others, there are four people and organizations described in this report that deserve special thanks: Pat Ferguson (AIS), Bob Musson (previously Teradyne), John Vu (Boeing), and Dave Webb (Hill Air Force Base).

And finally, special thanks go to Watts Humphrey. Without Watts, this work would not have been done. We continue to use his original ideas and processes to help improve the way we develop software. His passion and dedication help to drive the TSP Initiative. He continues to help us in maturing the TSP technology and defining new forms of the TSP approach for multi-disciplined teams, very large teams, departmental teams, and more.

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

1 Introduction

Organizations that develop software recognize that controlling their software processes significantly affects their ability to be successful in business. However, organizations still struggle when trying to apply disciplined methods in the software process. Historically, this struggle has resulted from a lack of operational procedures for use by teams and individuals in developing software in a disciplined fashion. Organizations seem to know *what* they want their teams to be doing, but they struggle with *how* to do it. The Team Software Process[SM] (TSP[SM]) was designed to provide both a strategy and a set of operational procedures for using disciplined software process methods at the individual and team levels.

The purpose of this report is to explain the TSP technology and some of the initial results of projects that have used this technology. This report is written for software managers, Software Engineering Process Group (SEPG), and educators who are interested in addressing cost, schedule, and quality problems by instilling disciplined quality methods in the software process and quantitatively measuring the results. This report assumes no familiarity with the TSP technology.

Chapter 1 provides some background on the TSP approach and an overview of the TSP technology. Chapter 2 summarizes some of the issues that software managers face relating to the software process, capability, and maturity, and how the TSP approach offers a strategy for addressing these issues. Chapter 3 presents some initial results of using the TSP in practice. It includes references for those readers who want to find out more about the individual company data. Chapter 4 provides analysis and discussion of the data reported from all four organizations. Based on this data, assertions are provided that may help organizations set improvement goals.

1.1 The TSP Defined

The TSP is a fully defined and measured process that teams can use to plan their work, execute their plans, and continuously improve their software development processes. The TSP process is defined in a series of process scripts that describe all aspects of project planning and product development. The process includes team role definitions, defined measures, and the postmortem process. As will be described, the TSP is an instance of the Capability Maturity Model[®] (CMM[®]) Maturity Level 5, defined for teams.

[SM] Team Software Process and TSP are service marks of Carnegie Mellon University.
[®] Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

Within the TSP scripts, there are operational definitions of the measures to be used as part of the process. These measures include basic size (thousands of lines of code [KLOC]), time (minutes and hours), and quality (defects), as well as derived measures for productivity (KLOC/hour), process yield (percentage of defects removed before a particular process phase), and defect densities (defects/KLOC) of finished products. The process establishes how these measures are defined, estimated, collected, reported, and analyzed. The process also makes use of the team's historical data, as well as industry planning and quality guidelines. Tools are available to help facilitate the TSP.

1.2 The TSP Purpose

A typical software engineering team spends a great deal of time and creative energy struggling with some of the following questions:

- What are our goals?

- What are the team roles and who will fill them?

- What are the responsibilities associated with these roles?

- How will the team make decisions and settle issues?

- What standards and procedures does the team need and how do we establish them?

- What are our quality objectives?

- How will we track quality and performance? What should we do if they fall short of our goals?

- What processes should we use to develop the product?

- What should be our development strategy?

- How should we produce the design?

- How should we integrate and test the product?

- How do we produce our development plan?

- How can we minimize the development schedule?

- How can we determine project status?

- How do we assess, track, and manage project risks?

- What do we do if our plan does not meet management's objectives?

- How do we report status to management and the customer?

The TSP provides explicit guidance on how to answer these questions and accomplish the team's objectives. The TSP shows engineering teams how to produce quality products for planned costs and on aggressive schedules. It achieves this by showing teams how to manage their work and by making them owners of their plans and processes. The TSP also helps to accelerate software process improvement.

The TSP has been used with software-only teams and with mixed teams composed of hardware, software, systems, and test professionals. The TSP can be used on teams that typically range in size from 3 to 15 individuals. For larger teams, a multi-team approach is being pilot tested and will be available in the future as an alternative TSP for large, multi-disciplinary teams. The TSP has been used for both new development and enhancement, and on applications ranging from commercial software to embedded real-time systems. It is also applicable in maintenance and support environments.

1.3 History

Watts Humphrey developed the Personal Software Process[SM] (PSP[SM]) and the TSP as a follow-on to his work with the Capability Maturity Model (CMM). The PSP is a defined process for individuals [Humphrey 95]. It is the foundation from which the TSP was developed for teams. His initial PSP development began in 1989. At the time, several small organizations were asking how they could apply the CMM framework as a small organization. Humphrey took on the challenge of trying to define the process and practices to be used at the individual level, as if that individual were trying to operate at CMM Level 5. Over a period of 4 years, Humphrey wrote more than 60 programs and more than 25,000 lines of code (LOC) as he defined, documented, and evolved the PSP. He collected data on size, time, and defects, and converged on an initial PSP approach. He subsequently worked on methods of training others in using the PSP technology. Courses were developed for graduate and undergraduate students, as well as for practicing engineers within organizations [Humphrey 95]. Results of the use of the PSP technology are documented in *The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers* [Hayes 97].

Humphrey recognized that the PSP technology was successful in the classroom and that student data were very consistent. It was clear that engineers could use the PSP to gain control of their personal process by improving their estimation skills and reducing the defect densities in the products they deliver, without adversely affecting their productivity [Hayes 97]. However, it was not clear how engineers could consistently apply these skills in practice within organizations. It turned out that the PSP skills were not effectively practiced within organizations unless those organizations made a commitment to the PSP approach and to the individuals who used it. 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. In these cases, engineers fell back to the chaotic practices that they had used before they were PSP-trained. Consequently, Humphrey developed the Team Software Process as an operational answer to the problem of implementing the PSP on organizational teams.

Scaling up the PSP to the team level involved a variety of problems. First was how to apply PSP planning methods at the team level. This function involved specific procedures for estimating size and effort, and more extensive use of the task and schedule planning methods used in the PSP. Measures had to be defined or extended to cover team situations. In addition,

---

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

other phases of the software project had to be addressed because the PSP covers only the software development phases of design through unit test. A more practical process had to be designed to cover requirements, integration and test, documentation, and other activities that a typical project would encounter. To add to the complexity, the TSP had to be designed to address some of the other team issues that are not related directly to the software process. This included team roles, inter-relationships within the organization, and definition of a team process to be used within the organization's existing project process.

The TSP represents an operational process for teams that may be used as a strategy for implementing the CMM framework on teams. The TSP as defined addresses nearly all of the key process areas (KPAs) in the CMM framework. Table 1 illustrates the key process areas addressed by the TSP. It should be noted that the phrase "KPAs Addressed" means that the intent of the KPA is satisfied in the context of a team developing software. It does not imply that all of the key practices and other attributes of each KPA are fully achieved.

Table 1:   Addressing the CMM KPAs

	Level	Focus	KPAs Addressed	KPAs Not Addressed
5	Optimizing	Continuous process improvement	• Defect prevention • Technology change management • Process change management	
4	Managed	Product and process quality	• Quantitative process management • Software quality management	
3	Defined	Engineering process	• Organizational process focus • Organizational process definition • Integrated software management • Software product engineering • Intergroup coordination • Peer reviews	Training program
2	Repeatable	Project management	• Requirements management • Software project planning • Software project tracking • Software quality assurance • Software configuration management	Software subcontract management
1	Initial			

The SEI is currently working to document formally the strengths and weaknesses of the TSP relative to the CMM framework for the explicit purpose of using the TSP as part of a CMM-based improvement effort. This information will be available initially as a series of SEI technical notes on each level for the Capability Maturity Model, and finally as a consolidated technical report.

2 The TSP Technology

This chapter provides a brief summary of the challenges facing software businesses. The systemic root causes of these challenges are described to set the context for the TSP technology. An overview of the TSP technology is then presented.

2.1 Software Business Challenges

Among organizations that develop software, there are varying perspectives on the challenges that they face. Although they see their software capability as a key to success, not all organizations have the same view of exactly how software capability affects a project's success. For example, in some industries, time to market may be important. In other industries, safety is a critical success measure. Still other industries view cost containment as the driving force in developing software products. From each of these perspectives (cycle time, safety, and cost) comes a seemingly different viewpoint on what can be done with the software process to help ensure project success.

Even within a single organization, various groups perceive the software challenges differently. The engineering manager may be concerned with filling open jobs and personnel turnover; the quality manager may be concerned with defects and quality measures; and the project manager may be concerned with getting the product out the door on time and within budget. Although each of these individuals has valid concerns, they often have conflicting opinions when it comes to making decisions about addressing the challenges they face. Some additional challenges that a business might face include the following:

- increased cost and schedule pressures

- inadequate cycle times and time to market

- poor or inadequate productivity

- lack of intergroup coordination

- excessive maintenance and warranty costs

- low staff morale and high turnover

- poor quality

- lack of requirements management

- lack of risk management

- poor status tracking

- difficulty achieving CMM levels and applying six sigma methods

Each of these challenges is difficult to deal with individually. Combined, it seems that the challenges are insurmountable. However, if one examines each of these challenges with respect to the organization's software process capability, there are common root problems that manifest themselves in all of these challenges. If an organization takes care of these common root problems, the challenges above are more manageable. The TSP was designed to address these root causes to ultimately address the software business challenges.

2.2 Addressing the Root Causes

Throughout the development of the TSP, several root causes of poor software development emerged. Software development often fails for the following reasons:

- Software engineers are not trained in sound development, planning, and quality practices.

- The focus in software development is on schedule and not on quality.

- There is no formal team-building process.

- A top-down management style produces unrealistic plans and causes engineers to lose motivation.

The following paragraphs explain how the TSP addresses each of these causes.

2.2.1 Training in Sound Development, Planning, and Quality Methods

Most software engineers learn how to develop software as students in college, where the main focus is on learning a language such as C or C++ and writing programs. Students do whatever it takes to write programs, compile them, and turn them in. They seldom worry about disciplined methods and quality. Nor are they overly concerned with planning their assignments, with the exception that they probably want to get done as quickly as possible. The bottom line is that the students do not learn to practice sound development, planning, and quality methods. In addition, the practices they used in college are carried over into their jobs. In many companies, these practices are still considered acceptable. However, with proper training and a supportive environment, individual engineers can develop the skills necessary to develop high-quality software predictably.

In the PSP and TSP courses, individuals learn how to produce plans based on their own historical data. They learn the concepts of estimating size, effort, and schedule, and of producing plans based on planned and earned value. This enables the individual to practice the TSP planning method, which builds on individual detailed plans and includes other team factors such as project goals, risks, and contingencies. The first step in the TSP planning method is a presentation of management and marketing goals to the team. Then, the team members discuss these goals, adopt them as their own, and refine them so that they have a clear direction. They build a top-down plan as a team, using historical data and procedures that define the products that they will produce and the processes that they will use. Later, they detail their individual pieces of the team plan, which results in a bottom-up team plan. Using the appro-

priate level of detail and historical data makes the plans accurate. Making the plan themselves from the bottom up helps build commitment among the team members. With the TSP planning method, individuals and teams consistently commit to aggressive schedules, and then make every effort to meet their commitments.

In the PSP training, individuals also learn how to develop software using disciplined methods. Guidance is given on how to develop coding standards, sound design practices, and measured review procedures. They are taught how to track and analyze data on their processes, and these data are used to provide accurate status checks against their plans and also to help manage quality and process improvement.

2.2.2 Managing and Measuring Quality

Most organizations are schedule-driven because it is the schedules that usually appear to be out of control. However, once teams gain control of their schedules through sound planning methods, they can focus on quality as the top priority. This represents both a cultural change (quality-focused vs. schedule-driven) and a procedural change. Both cultural and procedural changes are realized through the training that goes along with the TSP technology.

Most organizations find that cultural change is the most difficult part of applying a technology such as the TSP. As is common with installing new technologies, both engineers and managers resist change. They must see how the new technology can benefit them before they will accept changes. In the PSP training, managers and engineers learn the value of quality management and how to plan and track quality. The engineers learn that they are personally responsible for quality. The managers learn that they cannot sacrifice quality management because they will lose control of their projects. This enables the corporate focus to shift to quality. As long as top management also makes the quality commitment, their staffs (including managers, team leaders, and individual engineers) will work to achieve high quality levels.

Procedurally, the TSP process provides a quality strategy that establishes measurements and criteria for managing the quality of a team's products and processes. With this strategy, teams are capable of producing software products with fewer defects in system-level testing and in delivered products. This helps teams to gain control of the predictability of their processes, while also reducing the overall effort and expense of fixing defects at the system level. Data in this report support the notion that using the TSP quality strategy can have significant effects on the quality of the products being produced.

2.2.3 Building Teams

Working in teams is one of the more difficult aspects of most software process improvement methods. The TSP approach builds teams that work effectively and efficiently together so that the whole is greater than the sum of its parts. This is done through a series of activities that

occur in the initial team planning session, referred to as the TSP launch. During the launch, the team focuses on goals and roles, as well as producing the team plan.

A large determinant of team success is having clearly defined goals. In the TSP, an effort is made early in the planning phase to obtain marketing and management goals, and to embrace these goals as a team. The team does this by refining the management and marketing goals into team goals. They establish measures for each goal and identify a team member who will track each goal.

During the launch, each team member also takes on a team-management role, such as process manager, support manager, or design manager. In performing these roles, each member plays a part in ensuring that the team and its members follow the defined process and behave appropriately. These team-management roles also help to give individuals a sense of ownership on the team and frees the team leader to focus on coaching and leading the team instead of managing the team's activities.

During the launch process, many of the activities undertaken to produce a team plan also serve as team-building activities. The team applies techniques for generating ideas, evaluating alternatives, and making decisions. Team members learn how to rely on each other and how to communicate effectively. The weekly TSP team meetings help to reinforce these skills and to maintain active communication among all team members.

2.2.4 Data-Driven Team Management

When teams are given a schedule by management that they must adhere to, the team has few options. They can try to live with the schedule, or they can negotiate changes to the schedule so that it is realistic. Without the TSP approach, teams often do not have a way to negotiate schedules. The team may lack the process and the skills to plan, negotiate, and track schedules and commitments. When the skills and the process are lacking, more than likely the teams do not have data to base their planning estimates from, and they do not have data to quantify progress against their commitments. Without data, the status they report during project execution may be subjective. Engineers never really know exactly where they are on a schedule, and cannot tell management with confidence when they will finish. This frustrates the management team and so management begins to put pressure on the engineers. They may try to control the project by micromanaging every detail of the work. This can be very demotivating to the engineer.

With the TSP approach, the team has detailed plans based on data, and management is committed to the team's quality focus. The team is empowered to manage itself. Team members execute their plans and collect data on a daily basis. They conduct weekly meetings to report progress against their plans and to discuss team issues. They also make accurate status reports, based on data, to management on a regular basis. Because management can rely on that data, its job changes from continuously checking on status to ensuring that the team confi-

dently follows the process of performing according to its plan. Management must also address any problems that the team brings to its attention.

At the foundation of the TSP management method is quantitative data that represent the team's plan, its process, and its product. The team estimated and planned for size, effort, schedule, and quality. It uses data on these same parameters to track progress on a regular basis, to determine if the team is meeting its plan. When deviations occur, the team must solve the problem or raise the issue with management. In all cases and at all levels, decisions are made based on data.

Before using the TSP, the individuals on the team must first develop the necessary skills for participating on a TSP team. These skills are learned and practiced by taking courses that teach the PSP. All potential team members must take the *PSP for Engineers* course or the *Introduction to the Personal Software Process* for non-software engineers. Once trained in PSP methods, the team works together using the TSP. Additional information on available courses can be obtained online at <http://www.sei.cmu.edu/tsp>.

2.3 The TSP Structure

Figure 1 illustrates the TSP process as a series of phases and activities. Each major phase of the project begins with a planning session, referred to as a launch or a relaunch. The launch is a planning period where the team first puts a TSP plan in place. During the launch, the team puts an overall plan together that spans the entire project, and produces a detailed plan for the next phase of development. The detailed plan is at a level where each team member is guided by the plan and tracks the status of activities in the plan. The subsequent phases of development are contained in the overall plan but are at a higher level of detail. After the launch, the team executes the detailed, next-phase plan, and periodically conducts a relaunch to update plans for the next development phase.

During the launch/relaunch process, the team defines the project life cycle and establishes a strategy for developing the project. The TSP permits use of a strategy that makes the most business and technical sense for each particular team. For example, teams may develop software incrementally, in phases, or in cycles. The project life cycle and strategy is defined so that the team understands what phases and activities are included in its process, and what products are produced. Each activity is well defined with a series of steps, precise measures, and entry and exit criteria.

During the launch/relaunch, the teams develop plans to execute these activities for their projects. They estimate the size of the products to be produced, the time to be spent on each activity, and the defects that will occur and are to be removed in each phase. Schedule dates are also determined for the completion of each activity.
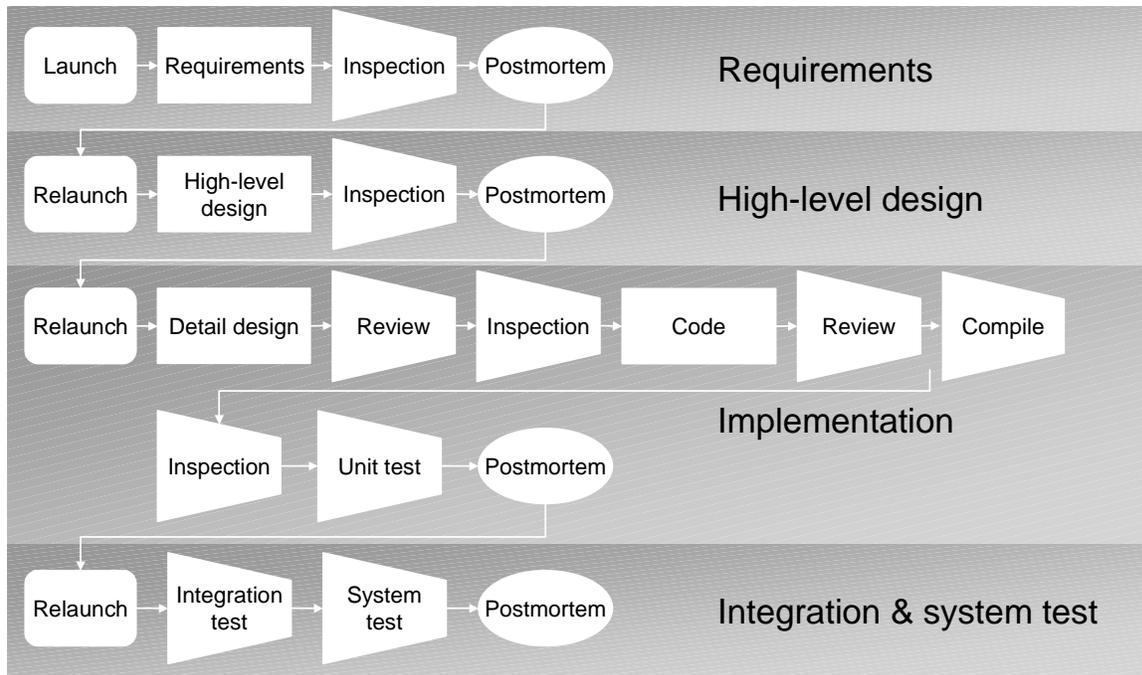
Figure 1:   The TSP Structure

It should be noted that Figure 1 represents a standard set of phases and activities that are pre-scribed for the TSP. However, it is not required that a team use a waterfall approach for de-veloping products. The TSP works equally well with a cyclical process structure, with phases for cycle 1, cycle 2, cycle 3, and so on. Additionally, the TSP approach may be applied with a prototype/version 1/version 2/version 3 structure. Many other options are possible. During the launch, the team will define its strategy, ensuring that the TSP discipline is still main-tained.

The TSP launch process is an integral part of the TSP technology. The launch is the point at which PSP-trained team members come together and apply sound development, planning, and quality methods to plan their project. They focus on how to build the products correctly, using a process that focuses on quality. They build the plan from the bottom up, and are thus more likely to commit to the schedule that is produced from this plan. In addition, the launch process provides the basic building blocks for forming successful teams. These building blocks include well-defined goals, established team roles and responsibilities, and a sense of ownership that each team member shares as a result of producing its own plan. In the next chapter, some initial results are presented from projects that have used the TSP technology in practice.

3 Results in Practice

This chapter presents some initial results from the application of TSP methods to software projects in industry. These results are limited at this time because the TSP has been available for only a short time. Note that some organizations have applied disciplined methods on projects based on PSP technology alone. For example, AIS developed its own TSP-like process, based on the use of PSP on projects. This was because the TSP approach was not available when Advanced Information Services (AIS) began its work with disciplined methods.

Since the TSP technology is new, it has not yet gained widespread use. It also takes time to obtain data on projects because software projects in industry typically take months or years to complete. Furthermore, because of competition in the software industry, and the resulting sensitivity about sharing data, it can be difficult to persuade organizations to release their data to the public and to participate in studies such as this. Consequently, there are limited data at this time on TSP application. However, there have been a few published results, and they are compelling.

Disciplined methods based on the TSP technology have evolved significantly over the past few years. The projects that have used TSP have been diverse. Project size has ranged from 3 to 15 team members. Products have included embedded systems, real-time controllers, and commercial applications. Program size has ranged from a few hundred lines of code (LOC) to nearly half a million LOC. This chapter summarizes some of the data reported to illustrate the kinds of benefits that can be obtained when using the TSP.

The data in this report are drawn from documented experiences from projects in four organizations that have published their PSP/TSP results. These companies are Teradyne, Hill Air Force Base (AFB), Boeing, and Advanced Information Services. References are provided so that readers can obtain additional details on these case studies. The following sections describe the data reported by the individual organizations. Chapter 4 summarizes these data and provides some discussion of the aggregate results.

3.1 Teradyne, Inc.

This section summarizes data reported by Bob Musson from Teradyne, Inc. at the 1999 Software Engineering Symposium in Pittsburgh [Musson 99].

3.1.1 Company Background

Teradyne is a supplier of automated test equipment. Its Telecom Division has 350 employees, of whom 100 are engineers. At the time of this report, this organization was assessed as CMM Level 1 [Musson 99]. Its products consist of both embedded systems and Unix applications. The division used the TSP approach to launch and develop the RMU750 project, which consisted of an embedded portion and a Sun Solaris-based application. Data on the success of that project were reported at the 1999 Software Engineering Symposium and are summarized in this section. Since then, the division has been working with the SEI to introduce disciplined methods and now has several teams using the PSP and TSP in product development.

Teradyne's experience illustrates how the TSP launch process helps teams to negotiate commitments. The RMU750 project consisted of an integrated team with nine software engineers and five hardware engineers. The new hardware/software product was critically needed in nine months or less. At the conclusion of the TSP launch, the team presented its schedule, calling for delivery in 18 months—twice as long as management said was required. The team explained the rationale for its plan and justified the required time and effort. Management ultimately accepted the plan and the team produced a product of exceptional quality, 6 weeks ahead of its 18-month schedule.

3.1.2 Data Reported

The data reported by Teradyne is included in Tables 2 and 3 below. Table 2 shows the estimation accuracy for size, effort, and schedule. The delivered product actually was about 18% smaller than planned. Effort and schedule were within 8% of the estimates, and were actually less than planned [Musson 99].

Table 2:   Teradyne RMU750 Project Estimation Accuracy

	Plan	**Actual**
Size Estimate	110,000 LOC	89,995 LOC
Effort Estimate	16,000 hours	14,711 hours
Schedule	77 weeks	71 weeks

Table 3 illustrates the defect density of the product at the system-level testing phases, including field trial. As indicated, the team set aggressive goals in terms of planned defect densities, including zero defects in field trials. Although the team did not meet the goal of zero defects, the product had only two defects in field trials. That is considered very high quality in this product's domain. In fact, Teradyne reports that its quality levels in some areas show 100 times improvement over the company's typical projects [Musson 99].

Table 3:   Teradyne RMU750 Project Quality—Defects Removed

	Plan	Actual
Test Phase	**Defects/KLOC**	**Defects/KLOC**
Integration	1.0	0.2
System Test	0.1	0.4
Field Trial	0.0	0.02

3.1.3 Discussion

The major benefits of this project were that quality levels improved over prior projects, and actual effort and schedule were within 8% of the plan (i.e., they were 8% early). Teradyne reports that using the TSP led to a high-quality and well-controlled project [Musson 99].

One significant anecdotal point of this story has to do with the negotiated schedule. It is important to note that the team put together a plan that was initially unacceptable to management. This is often the case with non-TSP projects as well. However, with a TSP plan behind them, each individual on the team was able to objectively support both the plan and each other. Management ultimately had to accept the team's plan. In the end, both management and marketing appreciated the discipline that the TSP provided. The bottom line on schedules is that managers will always want a plan that they are confident the team can execute successfully. As a consequence of the TSP launch, the team was able to commit to management and management was confident that the team could deliver.

3.2 Hill Air Force Base

The TaskView project was conducted using the TSP approach at Hill Air Force Base by the Technology and Industrial Support (TIS) Directorate. This section summarizes data from that project reported by David Webb in the articles "Using the TSP on the TaskView Project" and "Managing Risk with the Team Software Process" [Webb 99, Webb 00].

3.2.1 Company Background

The TIS software group develops avionics and support software for the U.S. Air Force. The group sent several engineers through the SEI's PSP instructor authorization program. It started using PSP and TSP in 1997. The group was assessed at CMM Level 5 in 1998 [Webb 99].

Because the process at TIS is mature (CMM Level 5), meeting schedule commitments is routine. Thus, Webb does not discuss effort and schedule deviation data in detail [Webb 99, Webb 00]. It was reported that the delivery date for TaskView was within days of the deadline

given in the plan, and cost deviations were negligible. More significantly, improvements in quality were gained. This was a significant outcome because the quality at TIS was already high, thanks to its existing Level 5 capability.

3.2.2 Data Reported

Webb presents a discussion of productivity improvements experienced on the TaskView project [Webb 99]. TIS tracks average productivity in LOC/hour for every completed project. Although exact numbers are proprietary, the TaskView project represented an improvement in productivity by 16% above the average TIS project. In fact, the team that developed TaskView had an increase in productivity of 123% over that particular team's previous project. TIS concluded that these results indicate that the TSP improves productivity.
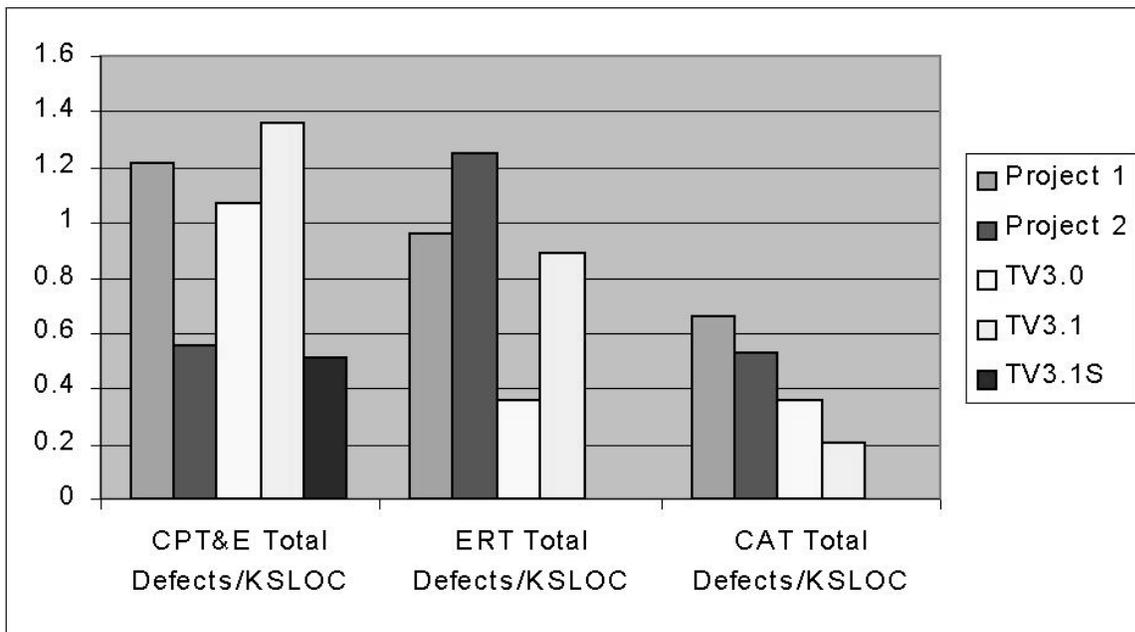
Figure 2:   Hill AFB Defect Density

In Figure 2, defect densities are illustrated for two prior projects that did not use TSP, as well as the current TaskView projects (3.0 and 3.1) that use TSP. At TIS, products go through several phases of testing before final delivery. These phases include functional test, candidate evaluation, system test, acceptance test, and operational test and evaluation. The product(s) then enter into operational usage. TaskView 3.1S is a small, special project and is not discussed in detail either in the Webb reports or in this report. The groups of data are listed by phase of testing. In these data, the three phases included are candidate evaluation (CPT&E), system test (ERT), and acceptance test (CAT). In general, the TaskView project reports that more defects are found during integration (and actually before integration, which is not shown). In the later phases of testing, the TaskView projects have lower defect densities.

As would be expected, the lower defect densities in system test correspond with shorter test durations. Figure 3 illustrates the relationship between test durations on the TaskView projects versus the prior projects 1 and 2 done by the same team, but not using TSP. It can be seen that the number of test days per thousand lines of code (KLOC) was reduced significantly. At TIS, projects usually require that 22% of the duration of the project be spent in the final two test phases. On the TaskView 3.0 project, this percentage was reduced to 2.7%, or a schedule savings of roughly 20%. Only one high-priority defect was found in these last two test phases (candidate evaluation and system test). This was unprecedented at TIS [Webb 99, Webb 00].
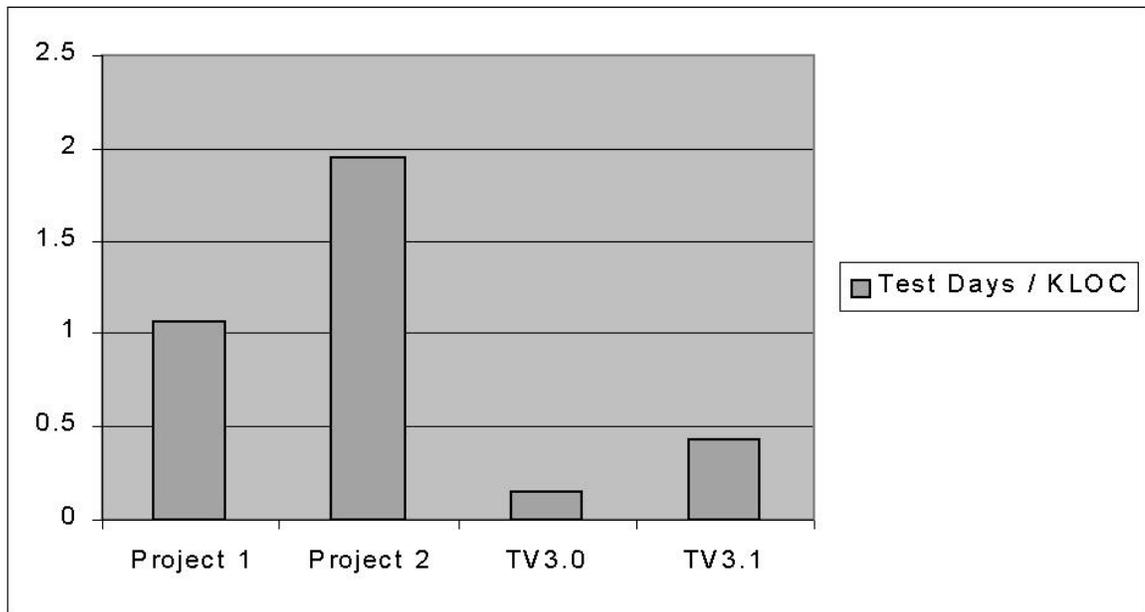
Figure 3: Hill AFB Test Duration

3.2.3 Discussion

The data shows how TIS, a CMM Level 5 organization, was able to make improvements in quality by using the TSP on the TaskView project. Data show that defect densities in acceptance test improved from prior projects that did not use the TSP.

In fact, on TaskView 3.1, the same team used the TSP and improved on its performance from TaskView 3.0. On TaskView 3.1, the team actually found more defects in integration and system test, but found fewer defects in acceptance test. This concerned the team until it began to look at the defects. It turned out that by examining only the high-priority or "hot" defects, the team realized that the increase in the number of defects in integration and system test was actually a result of its discipline in recording every defect. After using the TSP for an extended period of time, TIS became more accustomed to recording every defect. TIS reports that high-priority defects in acceptance test have become nearly non-existent. Almost all of

the defects that customers see are exclusively new requirements or limitations in the operational environment [Webb 00].

3.3 Boeing

This section summarizes data reported by John Vu at the SEPG 2000 conference in Seattle [Vu 00].

3.3.1   Company Background

The Boeing Company is a very large organization that has focused on CMM-based improvement since 1991. The company comprises more than 200 organizational units that have been assessed, and more than 10,000 people have been trained in the CMM framework. Several of Boeing's units have been assessed at high CMM maturity levels (3, 4, and 5), and the company has a well-established Software Engineering Process Group infrastructure. Boeing's process-improvement work has produced significant results that validate CMM-based methods. The company has been able to show that with advanced stages of process maturity, significant improvements can be made with respect to effort and schedule performance and improvements in productivity and cycle times, as well as improvements in quality and defect management [Vu 00].

The TSP collaboration with Boeing began in 1997. Boeing sent two engineers to the SEI PSP training course and these individuals later became certified to teach the PSP. TSP introduction started in 1998. Boeing is currently piloting the PSP and TSP. Three major pilot activities were used in the reported data.

- Pilot 1 is a major software update (new code and functions) being done by 35 software engineers, of which 32 are PSP trained.

- Pilot 2 consists of new software development being done by 36 PSP-trained software engineers.

- Pilot 3 consists of software maintenance (two releases/year), involving 22 PSP-trained software engineers.

3.3.2   Data Reported

Table 4 summarizes the results of the Boeing TSP pilots. The data show that Boeing's TSP projects obtained post-release defect densities of 0.08, 0.14, and 0 defects/KLOC. Data from Pilot 1 show how this represents improvement over prior performance. These data are illustrated in Figures 4 and 5 [Vu 00].

Table 4:   Boeing Pilot Size and Defect Data

Pilot	KLOC	Inspection Defects	Test Defects	Post-Release Defects	Total Defects
1	12.4	221	61	1	283
2	13.8	326	122	2	440
3	0.67	7	2	0	9

Figure 4 illustrates the reduction in post-release defect density on a TSP project from previous non-TSP projects. On Pilot 1, the measurement baseline was based on previous releases of the product from the same project team. Releases 6, 7, and 8 were done without using the TSP, and release 9 was done using the TSP. Figure 4 illustrates how the number of defects on the pilot decreased from the previous three releases. It illustrates that even though the size of the release grew by 2.36 times, the number of defects detected was 75% lower. This indicates a defect density reduction of (2.36/.25) = 9.44 times improvement over previous releases [Vu 00].

Figure 5 illustrates how the reduction in defect density also produced decreases in system test time. Again, even though the size of the release was 2.36 times as large, the amount of time per KLOC needed to execute system testing was reduced by 94%. This was attributed to the fact that few defects were present in system test, so the test procedures had to be executed only once [Vu 00].

Reductions in system test duration translate to reductions in cycle time. Because fewer days are required for testing, the product is ready for delivery sooner. In the pilot, cycle time was reduced by almost one month because of reductions in system test time alone [Vu 00].
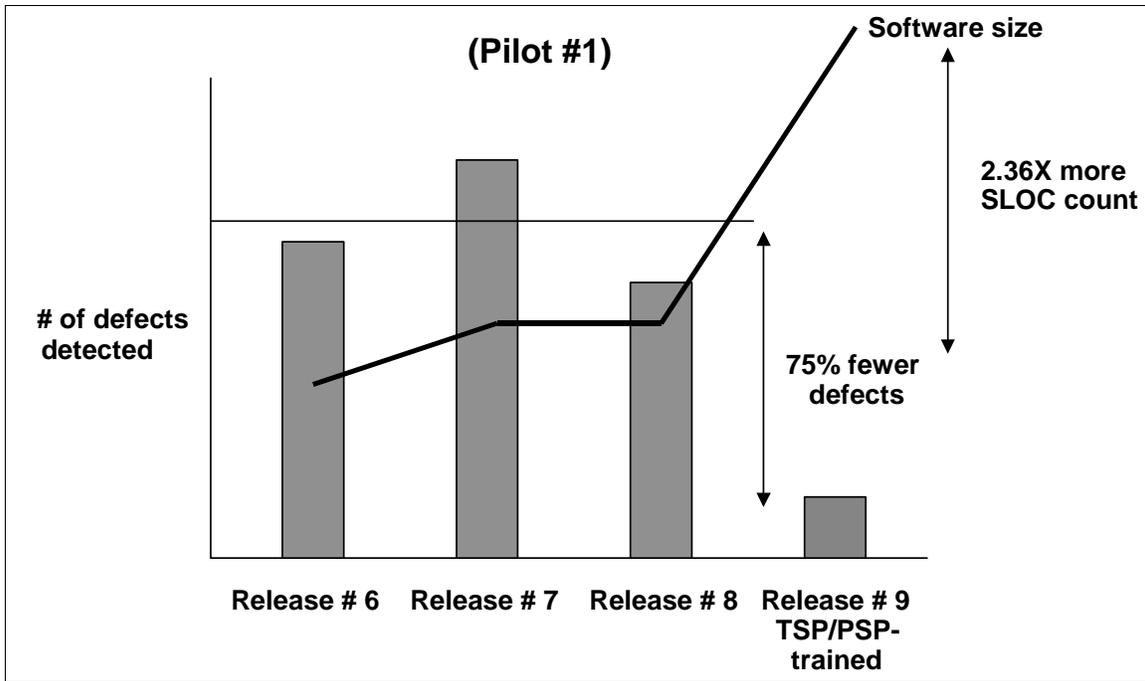
Figure 4:   Boeing Defect Data for Pilot 1
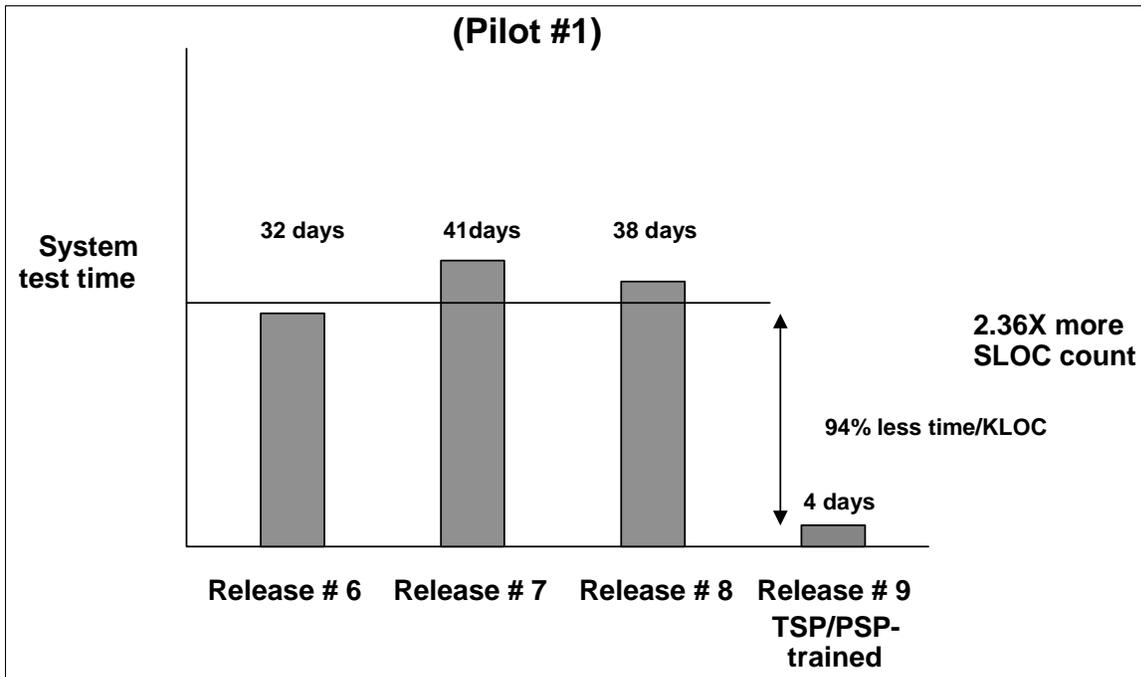
Figure 5:   Boeing Test Time Data for Pilot 1

3.3.3 Discussion

The data reported by Boeing show improvements in quality and reductions in system test time. Data were not reported on defect densities in other defect-removal phases, such as compile and personal reviews. These data were recorded by the individual and were not summarized for the team. As part of the internal negotiations to conduct this pilot, the team agreed that it would not publicly report on personal data. However, the team coach did work with individuals on their personal data to ensure that TSP quality methods were being used.

Boeing reports that it has made significant improvements using the TSP technology. As a mature organization with many initiatives underway, Boeing views the TSP as a means of implementing CMM-based improvement, and reports that TSP is able to help organizations achieve higher maturity levels. A major factor in the success of the TSP is the ability to change individual behaviors and attitudes about using disciplined methods. Boeing reports that collecting measurement data enables its engineers to focus on quality and process improvement, while the teamwork approach helps them to strategically build successful project teams.

3.4 Advanced Information Services

This section summarizes data reported by Pat Ferguson at the Software Engineering Symposium 1999 conference in Pittsburgh [Ferguson 99]. As noted earlier in this report, Advanced Information Services (AIS) developed its own team process based on the PSP approach. AIS's work with disciplined engineering methods pre-dates the availability of the TSP in its current form, so AIS developed its own team process that uses the PSP and is consistent with the TSP. All work currently done by AIS software engineers uses these disciplined methods.

3.4.1 Company Background

AIS is an independent software contracting organization based in Peoria, IL, and Chennai, India. AIS builds custom development projects, almost all of which involve new technologies and application domains. AIS builds systems that are client–server, embedded, mainframe, and Internet/intranet. The company has a long history of process improvement and working with the SEI.

The process-improvement history at AIS can be broken into three periods. During the initial period, from 1986 to 1992, AIS had no focused effort on process improvement. Data from projects during this period are reported as a baseline for comparison to the subsequent periods. During this period, schedule and effort were unpredictable and quality was not measured. The organization struggled with rework, change control, and other typical problems that a CMM Level 1 organization faces. Its average schedule deviation was 112%, and effort deviation was 85%. Quality was rarely measured because defects were not always recorded and tracked reliably. Also, AIS's acceptance test phase was not always distinct, as is often the case when there is little or no focus on quality.

In 1992, AIS recognized the need to improve. Its business required that it deliver defect-free software products and satisfy customers' increasingly demanding time-to-market goals. It also needed to improve the profitability of development projects by meeting cost estimates and schedule commitments with reasonable consistency. Subsequently, AIS embarked on a period of CMM-based improvement, lasting from 1992 to 1996. During this period, the company focused on doing assessments, developing action plans, and implementing training to align existing processes with the CMM framework. AIS was successful, reducing schedule deviations to an average of 41% and effort deviations to 17%. The company was also able to concentrate on measuring quality and began predicting and tracking quality in acceptance test [Ferguson 99].

In 1996, as a result of a CMM-based assessment, AIS introduced the PSP to the organization. This began the most recent period of process improvement, in which the PSP technology was used in conjunction with a CMM-based improvement strategy. During this period AIS, which has three SEI-authorized PSP instructors to train its engineering staff, was able to make even more improvements in its process, as indicated by Figures 6–9. Based on these results, the AIS development group was recognized by the community of software professionals with the 1999 Institute of Electrical and Electronics Engineers (IEEE) Computer Society Software Process Achievement Award [Ferguson 99].

3.4.2 Data Reported

The following figures illustrate the improvements that AIS has achieved in predictability for effort and schedule. These graphics were presented by AIS in an SEI-published report and at the 1999 Software Engineering Symposium in Pittsburgh [Ferguson 99].

**Effort Deviation Individual Value Control Chart -
Commercial Systems**
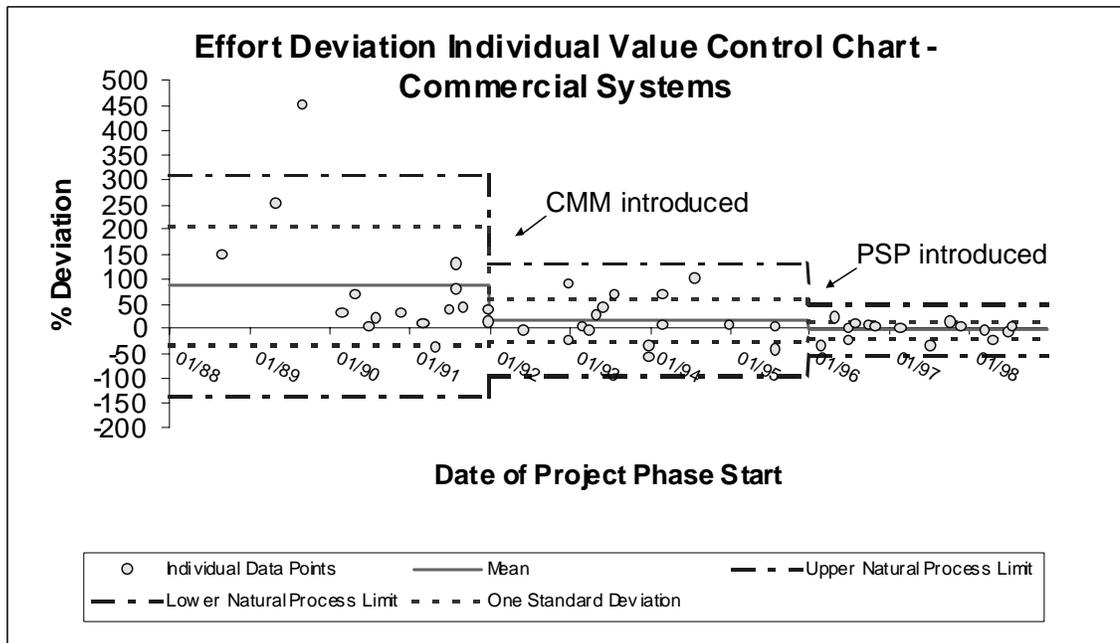
Figure 6:   AIS Effort Deviation

The first range on Figures 6 and 7 is from 1988 to 1992, the "no-process" period. The second range is from 1992 to 1995, the CMM period. The third range is from 1996 to present, the PSP period. The percentage of deviation-from-plan is on the vertical axis. As shown, the results in the early years were unpredictable and projects were frequently late. In the CMM period, project-management processes helped teams meet commitments more frequently, and schedule predictability improved [Ferguson 99].
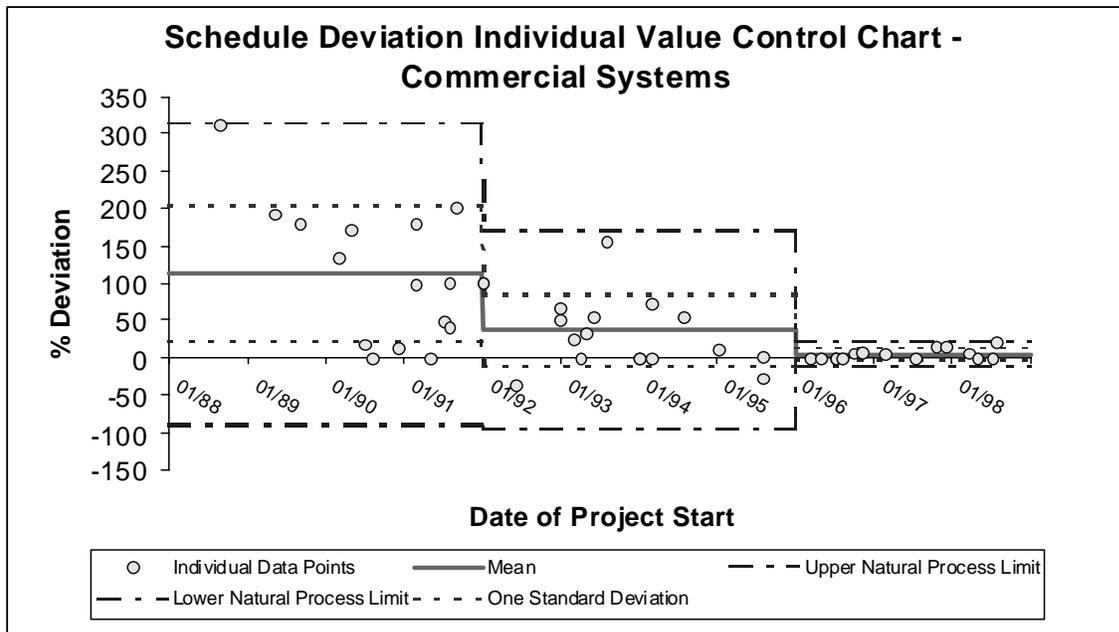
Figure 7: AIS Schedule Deviation

After the implementation of the PSP approach, two things enabled AIS to improve schedule predictability even further. AIS reports that individuals' schedules for their components are more accurate, and there is more commitment to those schedules. The quality of work is reportedly better, and the corresponding reduction in rework helps engineers meet planned schedules [Ferguson 99].

As shown in Figures 6 and 7, the average effort and schedule deviations were reduced throughout the three periods. More important than the averages are the ranges. Even during the CMM period (before PSP introduction), projects ranged from –50% to +150% for schedule deviation, and from –60% to +100% in effort deviation. These ranges made it difficult for AIS to make reliable commitments. During the PSP period, the range of schedule deviations narrowed to between 0 and 20%, and the effort-deviation range narrowed to plus or minus 25% [Ferguson 99].

Currently, AIS uses the natural process limits determined by the performance of its process to estimate future work with confidence. Consequently, the company is able to make accurate schedule commitments to its customers. Even though one or two PSP projects were off by 20%, AIS can look at the circumstances surrounding those projects to attribute the cause of the deviation. Then, the company can look at a proposed project and determine if the same circumstances are present and account for this range when making commitments [Ferguson 99].

Figure 8 shows reductions in system test defect densities on AIS projects. In Figure 8, the horizontal axis represents a range of project start dates. Each point on the chart is a completed/delivered project. The scale on the vertical axis is defects per KLOC [Ferguson 99].
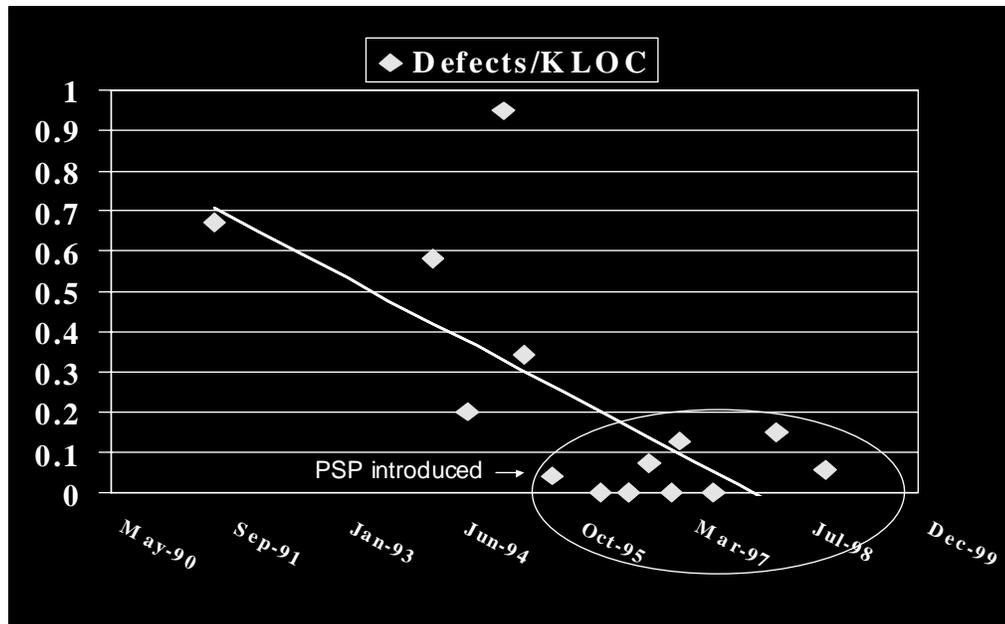
Figure 8:   System Test Defect Densities

The earliest quality data available are from 1991, but not from all projects. Only in the past few years have quality data been available from all projects. In the early years, AIS did not track defects, nor did it measure size. However, AIS reports that during the early years, customer acceptance was a lengthy process because of rework. If points were plotted for those early years, they would most likely be off of the graph. Data gathered since the introduction of PSP in 1995 suggests an increase in product quality [Ferguson 99].

Data in Figure 9 show that for the projects reported in Figure 8, lower defect densities resulted in fewer days of system test. Again, this reduces the project cycle time and saves the effort that would typically be expended to fix problems and rerun test procedures [Ferguson 99].
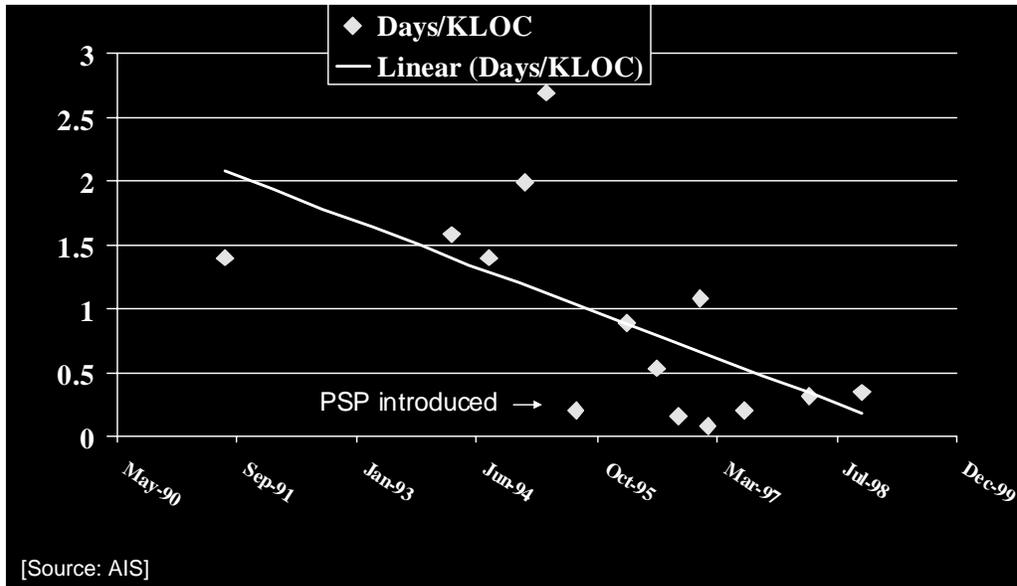
Figure 9:   AIS System Test Duration

3.4.3 Comparing PSP and Non-PSP Engineers

AIS has some limited data comparing PSP and non-PSP engineers. On a 1995 PSP pilot project, some of its software engineers had been trained in PSP and some had not. The experience and education levels of these engineers were similar. Acceptance test defects were tracked back to the components developed by each engineer, and results showed that the PSP-trained software engineers produced a higher quality product. Also on this pilot project, the PSP-trained engineers were reportedly more productive than non-PSP-trained engineers. At AIS, this is the best and only comparison available because AIS cannot do a comparison like this at the present time—for a very good reason: all software engineers are now PSP trained and PSP practices are integrated into the AIS defined process [Ferguson 99].

3.4.4 Discussion

It should be noted that AIS presented results from the use of PSP on projects; the organization has not formally started using the TSP. However, its internal process is similar to the process presented in the TSP and includes the same measures and procedures. It is therefore comparable to the formal TSP process.

Most of the projects at AIS are done for external customers at an established price. AIS relies on its estimating ability to win projects, and project success depends on meeting commitments. The company must be confident that it can develop the products within the schedule and budget. The disciplined methods and data have been extremely helpful in enabling AIS to estimate and meet its commitments.

A major benefit reported by AIS is that the organization and its individual staff members understand continuous process improvement. Everyone participates in proposing and implementing small changes. The value of process improvement no longer has to be explained. It is a way of life.

4 Summary of the Data

The case studies presented in Chapter 3 each provide some common ways to represent TSP results and have some unique aspects. The common elements include reductions in defect densities and test duration, and improvements in the accuracy of effort and schedule estimates. Some uncommon aspects reported include the comparison between PSP- and non-PSP-trained engineers (AIS), and the comparison between estimated and actual project size and effort (Teradyne). Another aspect that could be considered uncommon is the definition of the levels of testing among the four organizations. Based on these similarities and differences in the data, it can be misleading to combine these results and look at aggregate data. However, to explore the opportunities for improvement that the TSP provides, this chapter looks at the effects of using TSP on projects aggregated from the four reporting organizations. Based on the results, some preliminary conclusions are drawn and assertions made to help organizations set goals for improvement.

4.1 Preliminary Data Analysis

The data presented in this chapter are based on exploratory analysis of the data from each of the four organizations reporting. Assumptions for exploratory data analysis are not as rigorous as for establishing statistical significance. For example:

- It is not assumed that the data come from homogenous samples.
- The population is not assumed to be normally distributed.
- The number of data points may be relatively small.

Looking at data across organizations under these conditions can be misleading. However, the use of statistical tests in this situation can provide insight into patterns within the data that might otherwise be misinterpreted. Table 5 simply indicates ranges of values reported so that limited trends can be seen within the data. The ranges do not include data from every project, as not all organizations reported comparable information.

Table 5:   Summary Results of Using the TSP

Category	Without TSP	With TSP
Effort deviation	16%	-4%
(% average, range)	-60% to +100%	-36% to +26%
Schedule deviation (% average, range)	33%	6%
	-33% to +157%	-8% to 23%
System test defect density (defect/KLOC)	1 to 8	0.1 to 1.1
Acceptance test and release defect density (defects/KLOC)	0.2 to 1.9	0 to 0.35
Duration of system test (days/KLOC)	1 to 7.7	0.1 to 1.1

Analysis of variance (ANOVA) tests of effort and schedule deviations do not indicate statistical significance between treatments. However, since visual examination clearly shows a pronounced difference in the ranges, an F-test was done on the ratio of the variances of this data. This test suggests that the variation in the effort and schedule deviation data is reduced significantly, as is discussed in Section 4.2. The ANOVA tests done on the defect density and test duration data do indicate a meaningful difference. The data suggest that projects using the TSP reduce defect densities and test duration. Additional comparable data from more projects are required to examine statistically these suggestive results.

Note that the ranges shown in Table 5 overlap in some cases. This can be attributed to the fact that these ranges represent data from four different organizations and that the operational definitions for each of the parameters may be slightly different for each organization. This overlap tends to dilute some of the significant results obtained at each organization. For example, see Figure 10, which shows the individual improvements in average acceptance-test defect density across the four organizations. Defect densities were reduced by 70% to 95% across the four organizations.
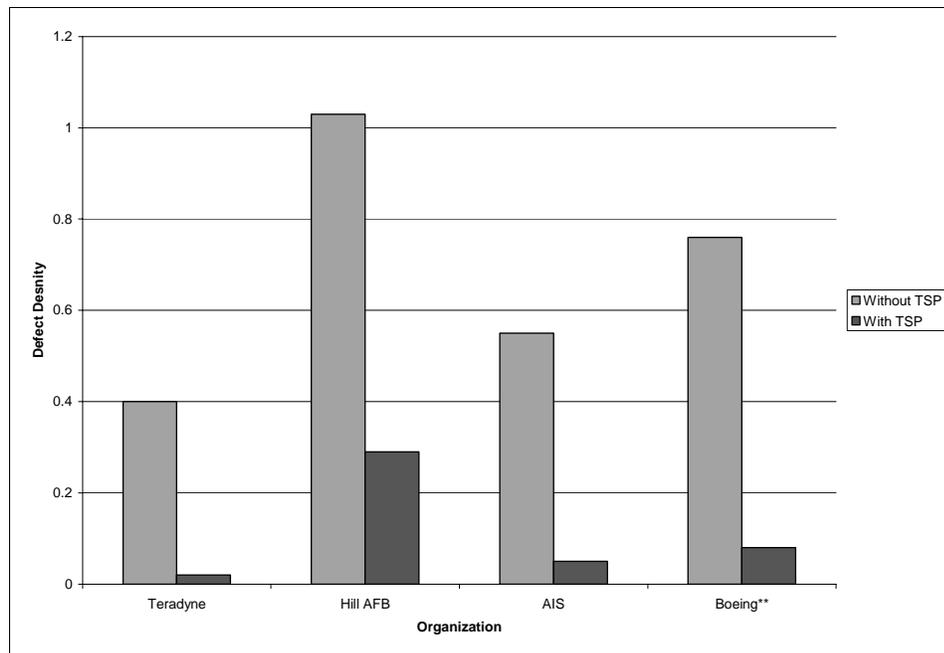
** Represents post-release data

Figure 10: Improvements in Acceptance Test

The next several sections will discuss how organizations have, through the use of disciplined methods, been able to achieve improvements in estimation accuracy, defect density, and ultimately productivity. Detailed case studies that supplied this data are included in Sections 3.1–3.4. For further detail, see the references cited in each case-study section.

4.2 Estimation Accuracy: On-Track Costs and Schedules

Cost and schedule problems often begin when engineers make commitments that are based on inaccurate size and resource estimates. With the TSP approach, engineers use the estimating methods they learned in the PSP training. They have historical data on their own personal productivity from the PSP class assignments, and the TSP provides additional planning and quality guidelines to help them make reliable estimates for size, effort, and schedule. However, just as important as the historical data are the methods that the individuals follow to develop conceptual designs for planning, to break the system down into parts that can be more reliably estimated, and to develop detailed, earned-value plans that can be tracked as the project is implemented. As a result of the detailed plans, TSP teams are able to accurately track progress and make adjustments to their detailed plans when necessary so that their overall commitment is maintained. Data from projects seem to indicate that although individual task and part estimates may deviate from a plan, teams are able to hold to their overall schedules, usually at the predicted cost.

Figure 11 shows charts generated from the SPSS tool using the effort and schedule deviation data provided by AIS and Teradyne. The charts show the median deviation within a box that encloses the 25th and 75th percentiles. In other words, 50% of the projects reporting fall within the box. The high and low bars on the chart represent the maximum and minimum data points. As can be seen, the variation in the data is reduced considerably with the use of the TSP.[1]
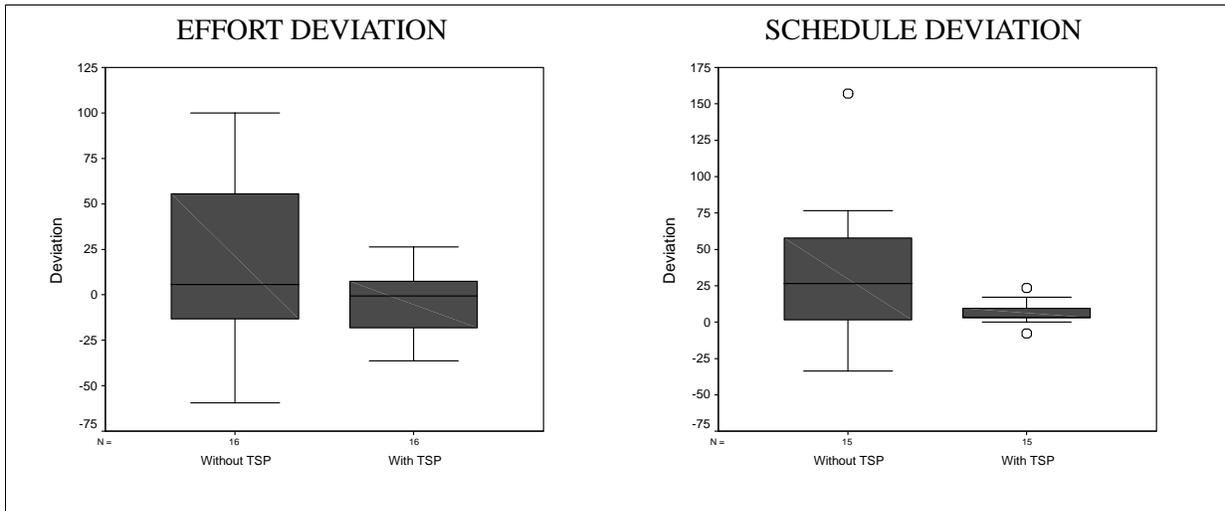

Figure 11: Effort and Schedule Deviation

The ranges indicated in the table do not come from data on all projects. It should be noted that there are often overriding factors with software projects that cause the actual costs and schedules to deviate from plan. In many cases, the customer may ask for more functionality or change the delivery dates based on revised needs. This was the case with some of the projects in this report. As a consequence of these changing customer needs, cost and schedule estimates were not emphasized or tracked and were considered less important to report on than the resulting product quality.

With disciplined methods, the TSP teams are better able to manage changing customer needs. When a customer asks for additional functionality, the team is able to assess immediately and accurately how this change affects the detailed plan. The team can make adjustments to its plans, if necessary, to accommodate the customer request. The team can also judge whether the change indicates a change in cost or schedule, so that the change can be negotiated with the customer. Without detailed plans, teams are often forced to take on additional work, without relief on either cost or schedule.

---

[1]   Using the F-test for variances, the calculated test statistic for the effort deviation data was 6.3 versus a critical value of 2.46. The calculated test statistic for schedule deviation data was 45.6 versus a critical value of 2.55. These data suggest that the variation in the effort and schedule deviation data was reduced significantly.

4.3 Defect Density: Higher-Quality Products

Figure 12 illustrates the reductions in system and acceptance test defect densities through the use of disciplined methods.[2] Before the TSP launch, engineers learn from the PSP training how to manage defects in their personal processes. The PSP teaches engineers to remove defects early, near the time when the defects are injected, rather than later when software products are tested and shipped to the customer. The PSP also teaches engineers how much the defects cost them and that defects found later in the development process cost significantly more to fix than those found early. In the TSP, quality management methods are defined so that individuals can practice these defect-management techniques in a team environment. The team estimates the number of defects that it expects will be injected and removed throughout the process. With these estimates, team members are able to set quality goals for desired target defect densities in finished products. Then, team members track detailed data against these goals and make project and process decisions along the way using these data. For example, team members are able to answer questions such as: "Is the product ready for a team inspection?" "Is the product ready to ship?" "How many defects remain and how will this affect our field support effort?"
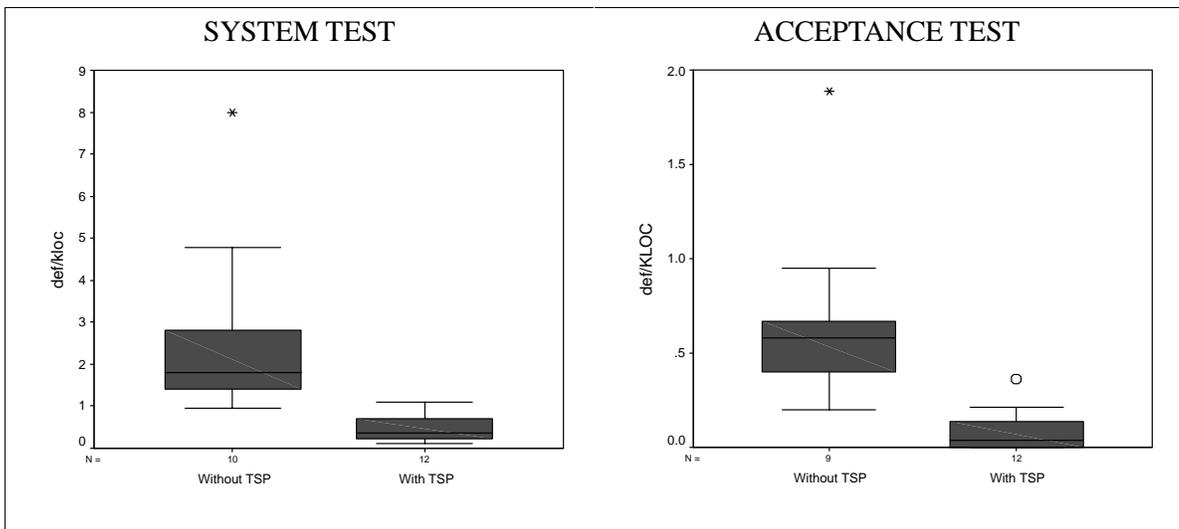

Figure 12: Reductions in Test Defect Densities

On TSP teams, individuals are responsible for the quality of the products they produce. The team is responsible for the quality of the products that they deliver. With this responsibility comes a sense of ownership, whereby the team is committed to using the quality methods. All products are personally reviewed. After the personal reviews are successfully completed, code is compiled and inspected and documents are inspected by the team using disciplined, measured processes. Using these methods, teams are able to deliver products that are relatively free of defects. As the data show, even with limited pilot usage, the TSP has been suc-

---

[2]   ANOVA tests for these data resulted in calculated test statistics of 11.8 for system test and 17 for acceptance test versus a critical value of 4.3. These data suggest that there is a significant difference in the results.

cessful in producing software products with extraordinarily high quality. It should also be noted that some data come from an organization that was already at CMM Level 5; even that team was able to improve upon the quality and defect density of delivered products.

With the PSP and TSP approaches, defect data are collected at the personal level from the very beginning of the process. Defect data are estimated and tracked for requirements, design, and code activities. However, none of the projects reporting data in this report have reported data on these early stages of the projects. The major benefits in reported quality are in the form of reduced defect densities in system and acceptance test, as well as in field use.

4.4 Productivity: Reduced Cycle Times

In the PSP training, individual engineers learn how to gather the process data needed to minimize cycle times. Their data helps them to formulate accurate plans, eliminate rework, and reduce testing by as much as four to five times [Ferguson 99]. Using the TSP approach, teams are able to gather this same data and manage quality so that the team can also experience the benefits of reduced testing.

Figure 13 represents a timeline that illustrates this reduction. With TSP teams, the emphasis is on a shift in effort to the earlier phases of the project. On TSP projects, more time may be spent looking at requirements and high-level design, and reviewing and inspecting those designs. The expected savings eventually come in later phases. Even though development time may be the same, system test time is reduced on TSP projects, as is subsequent field support effort. This is a direct result of having fewer defects in system test and delivery.

Typical team

| Reqts | Design | Implement | System test |

TSP team

| Reqts | Design | Implement | System test |

Time savings
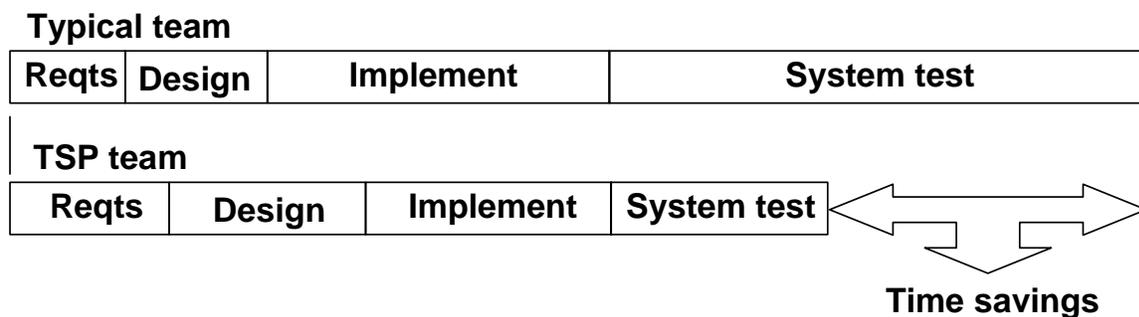
Figure 13: Reducing Cycle Time

Organizations reporting data to this point have not reported much information on changes in productivity, either with or without TSP. However, teams regularly report that testing times are significantly reduced, as shown in Figure 14.[3] In at least one case, reduced test times are a result of the fact that test procedures must be executed only once. Because there were no defects, the tests were completed rather quickly. It is expected that as more projects use the TSP, additional data will be available to allow comparisons in productivity between TSP and non-TSP teams.

COST OF SYSTEM TEST
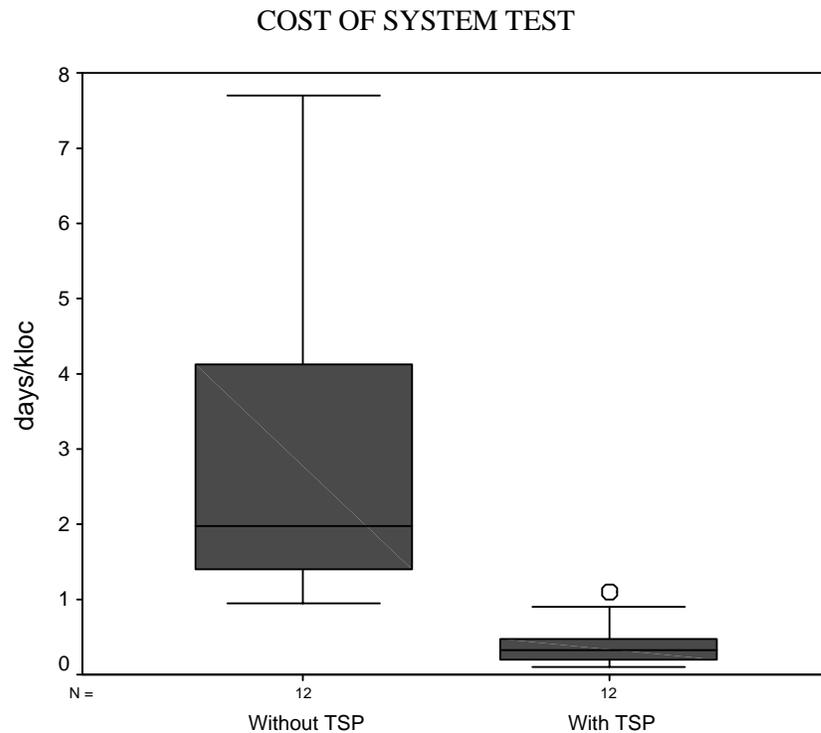
Figure 14: Reductions in Test Duration

4.5 Moving Ahead

Admittedly, the data thus far are limited. In this section, some assertions are made based on the reported results as a way to guide teams in setting goals and collecting data on future projects. It is a goal of the SEI's TSP Initiative to get more teams involved with the TSP and to collect more data on the use of TSP methods. Future reports can then provide additional details on TSP use and can enable statistically sound analyses.

With the use of the TSP, well-defined data are available consistently for projects to use to manage work. In addition, process experts and researchers can examine and analyze the data.

---

3   ANOVA tests for these data resulted in a calculated test statistic of 6.8 versus a critical value of 4.3. These data suggest that there is a significant difference in the results.

As more data become available from projects and more analysis is done, the TSP can evolve and become even more effective. Hayes reports that some important hypotheses were tested and supported on the PSP class and individual data [Hayes 97]. These hypotheses must be tested for TSP as more data become available. The PSP hypotheses were as follows:

1. **Size estimation** (actual and planned LOC). As engineers progress through the training, size estimates get closer to the actual size of the programs.

2. **Effort estimation** (actual and planned hours). As engineers progress through the training, effort estimates gradually get closer to the actual effort expended.

3. **Defect density** (defects/LOC in test and total). As engineers progress through the training, the number of defects injected and removed per thousand lines of code decreases. With the introduction of design and code reviews in PSP Level 2, the defect densities of programs entering compile and test phases decrease significantly.

4. **Process** (yield). As engineers progress through the training, their yield increases significantly, with the introduction of design and code reviews.

5. **Productivity** (LOC/hour). As engineers progress through the PSP training, their productivity increases.

With the exception of number 5 above, each of these hypotheses was supported [Hayes 97]. In the case of productivity, it was determined that no significant changes, either positive or negative, were made in productivity. That information itself was positive, as the amount of process change and data collection required by the PSP approach had no significant effect on the productivity of the individual. It is hypothesized that real productivity improvements come at the project/team level, with reduction of rework and system test times.

It is anticipated that the hypotheses tested on PSP class data can be extended to the team level because TSP measures are derived from PSP measures. Therefore, the same results are expected at the team level. In fact, the organizations reporting data in this report have already obtained results that support assertions based on the PSP hypotheses. Table 6 lists some assertions that are being made based on results of the PSP class data and initial results of TSP use reported in Chapter 3 of this report [Hayes 97]. This report presents these assertions and asks that teams who decide to use the TSP to look at their data and try to prove or refute these assertions.

To prove or refute the assertions presented in Table 6, there must be more data gathered on the consistent and rigorous use of the TSP approach, so that the software process can be measured using statistical methods. This can only come over time and with continued use. It is not expected that the software process will ever become as predictable as a manufacturing process. However, there are some key parameters in the software process that can be quantified and analyzed, and the results should prove to be significant.

Table 6:   Assertions with the TSP Approach

Area	Assertions (measures and/or considerations in parentheses)
Size estimate	Using the TSP approach, size estimates are closer to the actual size of products compared to teams that do not use the TSP approach (actual and planned LOC, pages).
Effort estimate	Using the TSP approach, effort estimates are closer to the actual effort expended compared to teams that do not use the TSP approach (actual and planned hours).
Schedule estimate	Using the TSP approach, schedule estimates are closer to the actual schedule compared to teams that do not use the TSP approach (actual and planned commitment dates and milestones).
Defect density	Using the TSP approach, the number of defects per KLOC found in testing phases will decrease with respect to teams that do not use the TSP approach (defects/KLOC in system test, acceptance test, and field use).
Process	Using the TSP approach, process yield will increase with respect to teams that do not use the TSP approach (process yield). Using the TSP approach, the amount of testing needed will decrease with respect to teams that do not use the TSP approach (number of hours or days per KLOC needed for testing). Using the TSP approach, the amount of time dedicated to testing activities will decrease (percentage of hours and/or days in test).
Productivity	Using the TSP approach, productivity (in LOC/hour) will improve over teams that do not use the TSP approach.
Data collection	With continued use of the TSP approach, individuals and teams will collect more accurate data (absence or presence of data, correlation and meaningfulness of the data).
Process fidelity	As teams gain experience with the TSP approach, the fidelity with which they use the process improves. As team process fidelity improves, planning, quality, and productivity also improve.

The initial data reported on the use of the TSP approach is compelling. The TSP has been shown to provide teams with immediate direction, focus, and clear plans for meeting their commitments. Teams have effectively and repeatedly delivered on these commitments. More important, the quality levels achieved even by beginning TSP users are exceptional. This has been observed in lower system-level defect densities in products produced by TSP teams. As a result of lower defect densities in test, test times and overall cycle times have also been reduced. This can be attributed to the fact that in testing, the test procedures may only need to be executed once, and with fewer defects the fix times and rework are reduced. With few defects remaining in the product, test time is minimized and products are delivered sooner. Based on these initial results, the assertions in Table 6 are presented so that organizations and teams can begin to establish goals for process improvement.

References

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

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

[Humphrey 95]   Humphrey, W. *A Discipline for Software Engineering*. Reading, Mass.: Addison-Wesley Publishing Company, Inc., 1995.

[Musson 99]   Musson, R. "The Results of Using the TSP on Small Teams." *Proceedings of the 1999 Software Engineering Symposium*. (Pittsburgh, Pa., Software Engineering Institute, September 1999).

[Vu 00]   Vu, J. "Process Improvement in the Boeing Company." *Proceedings of the 2000 Software Engineering Process Group (SEPG) Conference*. Seattle, Wa., Software Engineering Process Group, March 2000.

[Webb 99]   Webb, D. and Humphrey, W. "Using the TSP on the TaskView Project." *Crosstalk: The Journal of Defense Software Engineering 12*, 2 (February 1999). <http://stsc.hill.af.mil/CrossTalk/1999/feb/webb.asp>.

[Webb 00]   Webb, D. "Managing Risk with TSP." *Crosstalk: The Journal of Defense Software Engineering13*, 6 (June 2000). <http://stsc.hill.af.mil/CrossTalk/2000/jun/webb.asp>.

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE November 2000 (Revised October 2001)	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE The Team Software Process^{SM} (TSP^{SM}): An Overview and Preliminary Results of Using Disciplined Practices		5. FUNDING NUMBERS F19628-00-C-0003
6. AUTHOR(S) Donald R. McAndrews		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213		8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2000-TR-015
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116		10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2000-015
11. SUPPLEMENTARY NOTES		
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS		12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

The Software Engineering Institute has developed the Team Software Process^{SM} (TSP^{SM}) to help engineering teams more effectively build software-intensive products. The TSP addresses many of the current problems of building software-intensive products and shows teams and managers explicitly how to address these problems.

This report describes the TSP technology as an implementation strategy for teams that are attempting to apply disciplined software process methods. It provides some of the background and rationale for the TSP approach, as well as an overview of the technology. Then, the report presents initial results of the use of the TSP technology in four different organizational settings. In each of these organizations, the data show that defect densities found in system-level test activities and the actual duration of these system-level tests were reduced significantly with the use of the TSP. In addition, the accuracy of software estimates improved, and the variation in estimation accuracy was significantly reduced. Based on the analysis of these results, some assertions are made to help organizations set goals for improvement.

^{SM} Team Software Process and TSP are service marks of Carnegie Mellon University.

14. SUBJECT TERMS software process, Personal Software Process, PSP, Team Software Process, TSP, results	15. NUMBER OF PAGES 52
16. PRICE CODE	

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