Instructor's Guide for Introduction to the Team Software Process

By
Watts S. Humphrey

With Support Tool by
James W. Over

Contents

Chapter 1
Introduction

This guide is designed to help instructors when they teach the team course defined in the text Introduction to the Team Software Process. This guide is for use with the text *Introduction to the Team Software Process,*[1] by Watts S. Humphrey, (Reading, MA: Addison Wesley, 2000). It describes some issues and answers questions instructors will have as they guide student teams in developing small to moderate-sized software products. This chapter covers the following topics:

The purpose of this guide

Teaching a TSPi (Introductory Team Software Process) course

What the TSPi is

Team selection and leadership

Using TSPi in an industrial environment

When not to use TSPi

The contents of this guide

Even though much of the material in this guide concerns classroom situations, many of the topics relate to industrial work, and almost all the software and project-related comments apply equally to both educational and industrial environments. Industrial groups, however, should be aware of the limited nature of the TSPi process. Sections 1.4 and 1.5 of this chapter discuss various aspects of using TSPi.

Before they take a TSPi course, engineers must be trained in the Personal Software Process (PSP)[SM]. Also, each team should have four to six engineers, the projects must be small, and the schedules should be short. This guide does not repeat the material given in Introduction to the Team Software Process. It assumes that you have read that book and are familiar with its contents.

---

[1] This guide is for use with the text *Introduction to the Team Software Process,* by Watts S. Humphrey (Reading, MA: Addison Wesley, 2000).

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

Introduction 1.1 ->

Contents List of Forms Tool Instructions

1.1. Teaching a TSPi Course

The TSPi course objective is to teach engineers how to conduct a team software project. It shows them how to work on a team and how to follow a disciplined process. Although producing a working program is an important course objective, the program produced by the team is a means to an end. It provides the team¹s focus and is concrete evidence that the team completed the project. The principal goal, however, is to ensure that the teams properly use the TSPi process.

To get the students to use the process, you must emphasize the importance of following every step, gathering all the data, and completing all the required forms. If the students do not follow the process, their grades must suffer, and you should tell them this in advance. Also emphasize that you will periodically look at their work to ensure that they are following the process. Software development is a complex process, and if they do not faithfully follow the TSPi while they do their work, they will be unable to fake it after the fact.

Make sure that the students know the course priorities: process fidelity, cooperative and supportive teamwork, and the product. If they do not follow the process and gather and report all the process data, they will not get a good grade, regardless of the quality of the product they produce. After they complete the course, students should believe that the process helps and that it helps them the most when they are under schedule pressure.

<- Introduction 1.1 1.2 ->

Contents List of Forms Tool Instructions

TSPi Forms

Contents Instructions for Using the TSPi Support Tool List of Forms

Instructions for Using the TSPi Support Tool

1.2. What the TSPi Is

The TSPi process provides all the forms, scripts, and standards needed for a small software team project. It also provides a procedure for launching a project, the steps for conducting a project, and a tool for gathering, analyzing, and reporting the data for this project. TSPi shows engineers how to establish goals, assign team roles, and devise a development strategy. It also walks teams through making development and quality plans. TSPi teams make detailed schedules, track their progress, and assess their status against these schedules. In addition, TSPi guides them through requirements analysis, product design, implementation, test planning, and testing. It provides an explicit process for performing team inspections, establishing and using a configuration management system, and conducting a project postmortem assessment.

The TSPi requires that students have previously been trained in the PSP. The PSP and TSPi constitute a fundamental change from traditional software development. PSP changes the engineering culture from one that concentrates on coding and testing to a planned and measured process that produces quality products on predictable schedules. The TSPi then shows engineers how to use the PSP methods while planning and doing development work. PSP and TSP go together. Without one, the other will not be effective.

[<- 1.1](#) 1.2 [1.3 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

1.3. Team Selection and Leadership

Often, in a course, you will have limited say about who participates and who does not. However, you can decide which students are on which teams and who will have which roles on these teams. These are important issues, for the ability of team members to work together is the most important single factor in determining team success. Thus, if you do not know the students personally, you should get help from someone who does know them.

Start by selecting those students you think will make the best leaders; then decide which engineers to put with each of these leaders. In doing so, try to form balanced and compatible teams. Often, in fact, the students themselves can help in deciding how to compose the teams. In this process, the most important step is to select motivated and capable team leaders. This topic is described in more detail in Chapter 5.

<- 1.2 1.3 1.4 ->

Contents List of Forms Tool Instructions

1.4. Using TSPi in an Industrial Environment

The TSPi is a scaled-down version of the Team Software Process (TSP)[SM] developed by the Software Engineering Institute (SEI). This introductory version is designed to teach teamwork principles in a graduate or upper-level undergraduate university course. It was designed for use in a supportive environment with a restricted team size and a relatively short-duration project. Most important, the instructors who guide this work must monitor and support each team.

The material in the TSPi text should help working software teams, but if the project conditions are not relatively close to the environment planned for TSPi, the process will need to be tailored. The areas likely to most need attention are the team roles, the launch process, and team management.

The most important limitation of the TSPi is that it does not address management issues. The principal problem with using TSPi in industry is that the managers want to plan the engineers' work and assign their tasks. With TSP and TSPi, the engineers do these things and the managers concentrate on coaching and supporting the teams. Without a major intervention, managers do not generally know how to support TSPi teams. Under these conditions, the process often does not work.

The SEI has developed an industrial-grade TSP for larger projects and bigger teams. This industrial-grade TSP shows engineers how to apply the PSP in an industrial environment, and it addresses the changes needed in management culture. For further information on the TSP, contact the SEI.[2]

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

[2] Information on the SEI, the PSP, and the TSP can be obtained at www.sei.cmu.edu.

<- 1.3 1.4 1.5 ->

Contents List of Forms Tool Instructions

1.5. Where Not to Use TSPi

When engineers have not been PSP-trained, neither the TSP nor the TSPi process will work. In industry, we have found that when software engineers have not been PSP-trained, they do not generally understand why or how to plan a project or how to gather and use the process data. Because much of the TSPi process deals with project planning, data gathering, and data analysis, untrained engineers are often lost. Similarly, when students have not been PSP-trained, they will have so much new material to learn that they are not likely to complete their projects.

We once attempted to introduce TSP to an industrial team that was not PSP-trained. The results were totally unsatisfactory, and we learned to never introduce the TSP to a team that is not properly prepared. This is also true for TSPi. Changing the current software culture is extremely difficult, and if the engineers and their managers have not been PSP-trained, any attempt to guide them through the TSP or TSPi, will almost certainly fail.

<- 1.4 1.5 1.6 ->

Contents List of Forms Tool Instructions

1.6. The Contents of This Guide

This guide starts with a chapter on general suggestions (Chapter 2) followed by chapters on advanced preparation, a course overview, and team selection. Chapters 6 and 7 cover the PSP refresher and the weekly team-leader meeting. Chapter 8 covers many of the issues teams will likely face as they use the process, and it gives suggestions for how to handle these issues. Chapter 9 discusses the TSPi forms, and Chapters 10, 11, and 12 deal with subsequent development cycles, team coaching, and team grading. The last chapter comments on the rewards of teaching a team course, and Appendixes A and B provide exercises designed to familiarize engineers with the PSP.

[<- 1.5](#) 1.5 [Chapter 2 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

Chapter 2
General Suggestions

This chapter provides general guidance on the TSPi process. It contains the following parts:

Explaining processes

Processes as tools

The benefits of varied experiences

Class schedule and pace: one, two, or three development cycles

Keeping it small

Tool support

<- 1.6 Chapter 2 2.1 ->

Contents List of Forms Tool Instructions

2.1. Explaining Processes

When you use a process such as TSPi, it is important to distinguish between a process and a text. A process guides you in doing a task that you know how to do. It is not intended to teach you how to do something you don't understand. A process is thus more like a shopping list than a textbook. It doesn't contain anything you didn't already know, but when you don't follow it you are likely to forget something important.

The reason it is important to understand the nature of processes is that a process is useful only if people follow it. However, people do not follow a process if it is large and wordy or contains material that seems unnecessary or repetitive. Thus students using TSPi for the first time need guidance on using the process. Although the TSPi is explained in detail in the text *Introduction to the Team Software Process*, many students do not read textbooks. Others may not fully grasp the material.

It is not surprising that students have trouble understanding the TSPi. Even though the material is straightforward, many first-time readers simply do not grasp the concepts. TSPi is a major departure from the way they have previously developed software, and most inexperienced teams need considerable support at first. Therefore, you must plan to spend time with each team each week, particularly during the first development cycle.

The teams need guidance on how to follow the process, and they need coaching to ensure that they record and use their data. When faculty insist that the process be followed, teams generally do so, at least with adequate coaching. When teams follow the process, their results are almost universally good.

If teams do not follow the process, they do not learn the lessons the course is designed to teach. They generally fall behind schedule, panic, and revert to banging out code. Under these conditions, the teams are not efficient, the work takes longer, and the products are of poor quality. Most important, the students do not learn that the fastest and most efficient way to develop quality software is to follow a defined, high-quality process.

<- Chapter 2 2.1 2.2 ->

Contents List of Forms Tool Instructions

2.2. Processes as Tools

The process is a tool, and teams should adjust the process if they find parts that do not work for them. Make sure that the students understand that the process is not a straightjacket and that they can change it if they need to. They must, however, record their data and fill out the process forms. Before they change the process, however, they must make sure that the process is the problem. Do not let the students skip the reviews and inspections, stop recording their data, or omit completing the forms.

Be pragmatic about the process and make sure that the work students are required to do really contributes to their results. Students should think of the process as a tool that helps them do better work and not as a bureaucratic set of unnecessary tasks.

<- 2.1 2.2 2.3 ->

Contents List of Forms Tool Instructions

2.3. The Benefits of Varied Experiences

There is a trade-off in teaching this course. It is designed to maximize student learning, and a broader range of experiences will increase its educational value. However, changes in the process temporarily reduce productivity, add to learning time, and cause inefficiency. Therefore, students generally resist process changes when they are given a choice.

For example, in a second or third development cycle, if you give teams the choice of keeping the same roles or switching to new ones they invariably keep the same roles. Students who have been forced to change roles, however, generally think it was a good idea. The usual exception, however, is for the team leader. If teams want to keep the same leader, permit them to do so. Even then, if there is a good alternative candidate who would like the leadership role, urge the team to consider a change.

<- 2.2 2.3 2.4 ->

Contents List of Forms Tool Instructions

2.4. Class Schedule and Pace

The TSPi process is designed for one, two, or three development cycles. Depending on the other topics in the course, there should be ample time for one and possibly two cycles in almost any semester or quarter system. With three cycles, however, a full semester is required and there is no time for any other material. Depending on the students' familiarity with the PSP, you may not even have time for three development cycles.

Students often complain that the first TSPi cycle is too leisurely and that the second cycle is too rushed. This is by design. The TSPi is designed to push the students. It teaches them the process in the first cycle, and then, in the second cycle, they use the process when under schedule pressure. Thus, even though students learn about the process in the first cycle, they do not have the conviction to use it in practice. After the second cycle, students both understand the process and have tried to use it when rushing to meet a tight schedule. Unfortunately, under these conditions, many teams fail to follow the process. Although they know many things they should have done differently, they need a third cycle to fully internalize the lessons they learned in the first two.

Software development is not a relaxed occupation, and engineers rarely have a leisurely time. Thus, without pressure, students do not get a realistic experience. If you follow the defined course strategy, the students will see the need to adapt the size of the job to the time they expect to have available. They will also see that a defined, high-quality process is the fastest and best way to do the job, even under pressure. Because the process they will use in practice is the one they fall back on when in a hurry, they need to learn this important lesson.

[<- 2.3](#) 2.4 [2.5 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

2.5. Keeping It Small

In spite of the advice in the textbook, the most common mistake teams make is trying to do too much. Even faculty report that the biggest single mistake they made was picking too big a problem. So keep the projects small. You do not want a trivial problem, but unless you have built the product before and have data on how big it will likely be, the chances are that it will take more code than you think. When the job is too large, the students either fail to produce a working product or they spend an excessive amount of time in test. This is unnecessary and unfair because students can learn the important lessons of this course even while developing small products.

[<- 2.4](#) 2.5 [2.6 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

2.6. Tool Support

Software development is a sophisticated and complex process, and a well-run TSPi project will amass a substantial amount of data. Without an orderly process, recording and using these data are not easy. Sometimes the data are lost in volumes of material, and, more often, the data are simply not recorded. The TSPi tool was developed to help students manage their process data. After students enter their time, size, and defect data, the TSPi tool automatically completes all the required forms and makes all the necessary calculations.

Although the TSPi is relatively simple, it is difficult to use without tool support. Without support, students would have to do a substantial amount of manual data collection and complete a large number of forms. A tool helps to ensure process fidelity, but it also has the disadvantage that students do not gain a full understanding of all the forms, data, and calculations. You can minimize this potential problem by designing the course for three complete development cycles. You must also require that the students use their data to thoughtfully analyze and report on their work.

<- 2.5 2.6 Chapter 3 ->

Contents List of Forms Tool Instructions

Chapter 3
Specific Course Suggestions

This chapter describes specific suggestions for conducting a TSPi course. The topics covered are as follows.

Student preparation

Instructor preparation

The weekly schedule

Facilities required

[<- 2.6](#) Chapter 3 [3.1 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

3.1. Student Preparation

The basic prerequisites for the TSPi course are explained in the textbook *Introduction to the Team Software Process.* The principal requirement is that the students have completed a PSP course, either with the textbook *A Discipline for Software Engineering* or the textbook Introduction to the Personal Software Process. The students must also be competent programmers and have a copy of *Introduction to the Team Software Process.*

In addition, it would be most helpful if the students had courses on software design and software requirements. Additional useful courses include testing, configuration management, and team inspections.

If the students have not satisfactorily completed a PSP course or are not competent programmers, you should not let them into the TSPi course. They would likely have a great deal of trouble with the material. This would be unfair both to these students and to the other members of their teams.

Before the students start using the TSPi process, they should read textbook Chapters 1, 2, and 3 and Appendix A. They should also read the two-page scripts on each team role in Appendix E. Finally, have them read the chapter or chapters in Part III (Chapters 11 through 15) on the roles that most interest them.

During the course, the students must also read about one chapter a week during the first development cycle. These chapters are specified in the DEV script in Chapter 1 of the textbook as well as in Appendix D. Then, in cycles 2 and 3, the students should read those chapters in Part III (Chapters 11 through 15) that they have not previously read. Also explain that the material in Part IV (Chapters 16, 17, and 18) deals with teamwork problems and methods. These topics are frequent sources of trouble, and most engineers find it helpful to have read these chapters early in the course. I suggest that you assign this reading before the teams start the second development cycle or earlier.

[<- Chapter 3](#) 3.1 [3.2 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

3.2. Instructor Preparation

If you are a qualified software engineering or computer science instructor and are a competent programmer, you should have no trouble teaching a TSPi course. This assumes, of course, that you are adequately prepared. The suggested instructor preparation for this course is that you have either taught or taken a PSP course, using either the PSP textbook *A Discipline for Software Engineering* or the introductory textbook *Introduction to the Personal Software Process.* Alternatively, you could have attended a qualified faculty workshop on teaching the PSP and TSP. In addition, you must have read the textbook *Introduction to the Team Software Process* as well as this Instructor's Guide. You should also be familiar with the TSPi tool used by this text. It is described in the TSPi tool instructions which are available in the supplements for the TSPi text. These supplements are described on the last page of the text.

Experience with software development, either in industry or with project team courses, would also be most helpful. In addition, you should be familiar with software process principles, understand how processes are used in software organizations, and recognize the need for process discipline. In particular, you must be willing to monitor the quality of the processes the students use. This means that you must regularly examine their process data, analyze their work, and periodically emphasize the importance of quality. You must also examine the products the students produce, but this is of secondary importance compared with the need for maintaining process discipline.

<- 3.1 3.2 3.3 ->

Contents List of Forms Tool Instructions

3.3. The Weekly Schedule

The biggest problem most students and instructors have with the TSPi course is the need to manage their time. The teams must meet for weekly meetings, strategy development, product planning, and inspections. They generally have trouble finding this time. You also must hold a weekly meeting with the team leaders, help the teams when they have problems, and review the teams' TSPi materials and data. Many of these needs can be efficiently covered if you schedule a two- to three-hour closed laboratory session every week.

The principal factor governing the success of your course is the amount of time and effort you can devote to guiding and assessing the students' work. As a rule, you should expect to spend a minimum of about an hour per week per student on this work. For very small classes, the time would probably be two or more hours per week per student. If the course has more than about 10 to 15 students, however, you will likely need a teaching assistant.

In addition, as described in Chapters 7 and 9, the quality of the teams' work largely depends on the care and regularity with which you review their data. Because few faculty have the time to do all the reviews suggested in Chapter 9, you must establish priorities. Chapter 9 discusses this subject in some detail. I suggest you consider the following general priorities for reviewing team data.

1. Make sure that the teams are recording all the required data. You can generally
2. check this by looking at the team and engineer TASK, SCHEDULE, SUMP, and SUMQ forms.
3. When in doubt, also ask to see the engineers' LOGT and LOGD forms.
4. To anticipate problems and provide timely help, track the teams' progress
5. using the team and engineer WEEK forms.
6. Try to examine each form the first time it is called for, as shown in
7. Tables 7.1 and 9.1.
8. When forms are incorrect, require that they be corrected and reviewed with you
9. again. If they are still incorrect, do it again until they are right.

This discipline can initially take a fair amount of time. When the students start properly reporting their data and completing their forms, however, the time demands of the course will be much less.

3.4. Faculties Required

The key facility needed for a TSPi course is a laboratory environment where each student can have a personal workstation. These workstations must be equipped with the development environment the teams will use and an e-mail capability for intrateam communication any time of the day or night. To use the TSPi tool, the teams also need to run Microsoft Windows or Windows NT systems as well as Excel.

The teams will also need space to work together, hold team meetings, and meet with the instructor. Teamwork requires that the team members learn to work together. To do this, they must be able to interact frequently as a team. Unless the teams have convenient facilities for doing this, they are unlikely to have a satisfactory team experience.

<- 3.3 3.4 Chapter 4 ->

Contents List of Forms Tool Instructions

Chapter 4
Course Overview

This chapter describes a suggested initial lecture to explain what the TSPi course is about and to tell students what is expected of them. The chapter also describes the principal lessons the students should take from the course. The topics covered in this chapter are as follows.

Course objectives

The principal lessons of the course

Learning from history

Team selection

The PSP refresher

Team communication

Course grading

Before you discuss the TSPi process in detail, it would be a good idea to use the DEV script to give a brief overview. This overview should include the cyclic development strategy and the phases within each cycle. Also, describe the product the teams will develop and explain where they can find more information about it. If you plan to use one of the sample need statements in Appendix A of the textbook, tell the students which one and describe where to find it.

<- 3.4 Chapter 4 4.1 ->

Contents List of Forms Tool Instructions

4.1. Course Objectives

The three principal objectives of this course are to expose students to the problems of developing software-intensive products, to guide teams through a team development process, and to demonstrate the benefits of using a defined and measured process. Although the process the students will use is an extension of the PSP, they will apply PSP methods to a larger product, and they will track and report on their work every week.

<- Chapter 4 4.1 4.2 ->

Contents List of Forms Tool Instructions

4.2. The Principal Lessons of the Course

When students follow the TSPi process, they start off thinking that data gathering is a burden. Assuming that you require them to consistently record and report all their data, data gathering soon becomes automatic. Until you get the students to this point, however, they fight the process and have trouble learning from it.

The second lesson, which many software teams must learn for themselves, is that without a disciplined process, the development and testing of even small programs can take a very long time. In fact, industrial software groups often take several weeks to test programs of the size the students will develop in this course. Because this course allows only one week for testing in the first cycle and half a week each for the second and third cycles, the teams must use a high-quality process. If they do not, they either will not produce a working product or will have to work many late nights and weekends.

The third lesson is that, with a disciplined process, the teams will finish their projects on time, the products will work, and they will take little system test time. Finally, the students will find that disciplined teamwork is predictable, rewarding, and fun.

<- 4.1 4.2 4.3 ->

Contents List of Forms Tool Instructions

4.3. Learning from History

The principal advance of the Industrial Age was the introduction of scientific and engineering methods. Technological progress was limited, however, until the early scientists began to learn from the experiences of others. As long as artisans and craftsmen could learn only by themselves or from their immediate associates, technological progress was limited. If every engineer had been forced to invent calculus, create statistics, or devise Khirchhoff's laws, we wouldn't have automobiles, computers, or jet airplanes.

Urge your students to learn from what others have painfully learned. The TSPi builds on nearly 50 years of software development experience. When engineers don't capitalize on this wealth of historical knowledge, they spend years painfully learning what others have already discovered. What is more, they will spend this time only to learn that following a disciplined process is the fastest and best way to develop quality software. Although a lifetime of learning can be rewarding, it is a tragedy to waste it learning what is already well known.

The story of one industrial software developer is illustrative. He needed to produce a small program in one week. Although he had learned the PSP, he was not convinced that all the discipline was worthwhile. He therefore decided to forget about the PSP, ignore planning, skip design and reviews, and start writing code. He wrote the program in a couple of days, and compiling took only a few hours. Then he started testing. He worked late every night and most of one weekend and, after two weeks, was finally finished. He later told his manager, "Never again!" He realized that he would have found almost every test defect in a careful code review. In this course, the students do not have time for two weeks of system testing.

<- 4.2 4.3 4.4 ->

Contents List of Forms Tool Instructions

4.4. Team Selection

The TSPi process used in this course is designed for five-person teams. You will make the team assignments and, barring unforeseen problems, these teams will stay together for the entire semester. Also, the TSPi teams have defined team-member roles, and you will assign these roles for the first development cycle. To properly make these assignments, however, you need information from the students. The INFO form, designed for this purpose, contains a wealth of useful information. One of your first actions should be to hand out copies of the INFO form and ask the students to complete and return them to you as soon as possible. Point out that if they have any team or role preferences, they should indicate them on this form.

Also suggest that before they complete the INFO form, they should read the two-page descriptions of each role in Appendix E of the text. They will also find more information on the roles in Part III of the TSPi textbook. Team selection is discussed in more detail in Chapter 5 of this guide.

<- 4.3 4.4 4.5 ->

Contents List of Forms Tool Instructions

4.5. The PSP Refresher

If you plan to give a PSP refresher lecture or two, explain what you plan to cover and why. This topic is covered in more detail in Chapter 6 of this guide.

<- 4.4 4.5 4.6 ->

Contents List of Forms Tool Instructions

4.6. Team Communication

Describe how the TSPi maintains communication among teams and between the instructor and the teams. This is done through a weekly meeting of every team and the meeting you hold with all of the team leaders every week. The purpose of the team-leader meeting is to monitor each team's use of the TSPi and to provide help and guidance. Also describe the reporting process and explain that each team will report its data to you every week. The team-leader meeting is described in Chapter 7.

<- 4.5 4.6 4.7 ->

Contents List of Forms Tool Instructions

4.7. Course Grading

The students will be most interested in the criteria you will use to determine their grades. Their principal concern will be whether they will be graded as individuals or all members of a team will get identical grades. Explain the PEER form and discuss how you will handle grading. For a further discussion of these points, see Chapter 11 in this guide and Chapter 3 in the TSPi textbook.

<- 4.6 4.7 Chapter 5 ->

Contents List of Forms Tool Instructions

Chapter 5
Team Selection

This chapter discusses team selection for a TSPi course. The selection of teams for industrial software projects is an entirely different subject and is not discussed in this guide. This is one of the areas in which TSPi differs the most from the industrial-grade TSP. The topics covered in this chapter are as follows.

Selecting team leaders

Assigning students to teams

The team-leader meeting

Team size

Role assignments

<- 4.7 Chapter 5 5.1 ->

Contents List of Forms Tool Instructions

5.1. Selecting Team Leaders

Selecting good team leaders is important. The leader makes the team, and teams with effective leaders have an enormous advantage. Conversely, if the leader has personal problems or is unwilling or unable to establish and maintain discipline, the team is likely to fail. The strategy for selecting student teams is to first select the students you think will make the best leaders and then select the members for these leaders' teams. Next, before announcing the team assignments to the class, hold a team-leader meeting to discuss the selections.

In selecting the team leaders, start by reviewing the INFO forms and looking at the students' transcripts. If you don't know all the students well, ask other faculty for advice and then interview the candidates before making the final choices. You need to understand peoples' characters to judge whether they will be effective leaders. On the other hand, there is considerable evidence that many people can be effective leaders, at least when they need to be.

The principal criteria for effective leaders are that they are objective and motivated and can deal openly with their teams. They must not get hung up on authority problems or have an outsized ego. Effective leaders must be tough when necessary as well as empathic and patient as the situation requires. The key is to select leaders who will think about the team and its needs rather than about themselves and their needs.

Leadership is particularly important when a team has multiple stars. Highly competent engineers are often creative and self-confident. When a team has several such stars, they often compete. The leader's job is to focus the team on the principal objective and to build a cooperative rather than a competitive environment. The leader's objective is to fully utilize all the team members' skills and abilities.

Above all, avoid leaders who seek power or have something personal to prove. Effective leaders build loyalty, they engender trust, and they are enthusiastic. Like the best coaches, they strive to get the best from their people. They don't make touchdowns or hit home runs, and they don't seek publicity or praise. Instead, they want these tokens of success for their teams.

<- Chapter 5 5.1 5.2 ->

Contents List of Forms Tool Instructions

5.2. Assigning Students to Teams

The students' INFO forms will contain much of what you need to know about the students' capabilities and interests. Try to honor their preferences. If someone does not want to be on the same team with another student, try to oblige. Similarly, when students wish to work together, arrange it if you can.

Try to balance the students' capabilities. Give each team a mix of skills and backgrounds. Distribute those students who have industrial experience, and do the same for any graduate students or seniors who have had the full PSP course. Mix the men and the women, age with inexperience, and strive for a cultural, racial, and ethnic balance. Although you should not put antagonistic people together, recognize that team diversity generally increases a team's creativity and resilience.

In picking teams, try to ensure that all the students know a common programming language and urge them to use a language they all know. The students have a great deal to learn in this course without also struggling to learn a new programming language. For each team, try to pick students who have compatible schedules so that they can meet and work together.

Transfer students can be a problem for two reasons: you do not know them, and they may not be adequately prepared. You may also have to accept some transfer students who have not had PSP training, even though this is not recommended. Make sure to put such students on different teams, and give them extra help with the PSP and TSPi. Because it is unfair to put students into courses for which they are ill prepared, discuss the issues with them and make sure that they understand and are prepared to handle the problems.

<- 5-1 5.2 5.3 ->

Contents List of Forms Tool Instructions

5.3. The Team-leader Meeting

After you have picked the teams, meet with the team leaders before announcing the team assignments. In this meeting, discuss the students you have assigned to each team. You might even have a free-agent trading session, giving the team leaders an opportunity to swap team members. Although they will probably not trade anyone, the opportunity to do so will involve them in the team selection process and could prevent some serious problems.

<- 5-2 5.3 5.4 ->

Contents List of Forms Tool Instructions

5.4. Team Size

Five students per team is optimum. With four or fewer members, there can be problems if a student must leave. It is always traumatic when a team loses a member, but with a small team a single loss can cause especially serious problems. On the other hand, the more students there are on a team, the harder it is to balance schedules and to run a consensus-based decision-making process. It is best to stick to five-student teams if possible.

In general, class size will not be an even multiple of five. Under these conditions, make a few six-engineer teams instead of more four-engineer groups. For example, with 36 students, I suggest six teams of five and one of six instead of four teams of four students and four teams of five.

<- 5-3 5.4 5.5 ->

Contents List of Forms Tool Instructions

5.5. Role Assignments

Based on the information in the students' INFO forms, you should be able to make team assignments that give most students the roles they request. For larger or smaller teams, however, you must make some role adjustments. When assigning roles for four-student teams, distribute the support manager role among the team members. For six-student teams, split the quality/process manager role into a quality manager and a process manager.

When teams are larger than six, the role adjustments are more significant. You can, for example, establish roles for each process step, such as a requirements manager, design manager, and so on. These roles then maintain the focus on their areas of responsibility throughout the process. A test manager, for example, ensures that the team considers testing issues, even during the requirements phase.

Another approach is to establish backup responsibilities; students act as alternates to the member who has the primary role responsibility. It is then important, however, to ensure that the students with shared roles really share the work. One approach is to have them divide the activities and responsibilities shown in the scripts for that role. They give you a copy and put a copy in the team's project notebook. Because the role workload is heaviest for the team leader, the planning manager, and the quality/process manager, assign backups for these roles first.

<- 5.4 5.5 Chapter 6 ->

Contents List of Forms Tool Instructions

Chapter 6
The PSP Refresher

This chapter describes the suggested contents of some brief refresher lectures to give at the beginning of the course. These lectures are particularly important when a number of the students have not recently used the PSP. The topics in this chapter review subjects covered in the textbook Introduction to the Personal Software Process. The review topics and the chapters in this introductory PSP text that cover them are as follows.

When a refresher is needed

Metrics review (Chapter 19)

Tracking time (Chapter 3)

Defects, code reviews, and checklists (Chapters 12, 13, and 14)

Program size (Chapter 6)

Task and schedule planning (Chapter 9)

Estimating project completion (Chapter 9)

After a review of the first two topics, the recommended approach is to have the students complete an exercise and then to review their results in class. These exercises are about the PSP and are based on the process described in Introduction to the Personal Software Process. Because the TSPi process used in this course is more sophisticated, emphasize that the students must read the textbook for this course (Introduction to the Team Software Process) and familiarize themselves with the TSPi tool that supports it. All the exercises are described in the exercise packages given in Appendix A of this guide. The exercise answers are given in Appendix B.

If many students took their PSP courses more than a semester earlier or if they have not used the PSP in one or more recent courses, start the course with one or two refresher lectures. It is also a good idea to have the students do the exercises described in this chapter and even write a simple program using the PSP.

<- 5.5 Chapter 6 6.1 ->

Contents List of Forms Tool Instructions

6.1. Quality Metrics

The basic measures in the PSP and TSPi are time, size, and defects. If the students gather data on these measures for every process phase, every product element, and each team member, they can calculate all the measures they will need for this course. Although the PSP courses cover the measures used by the TSPi, it would be a good idea to review some of the principal measures students will be using. I recommend discussing KLOC, defects/KLOC, yield, A/FR and LOC/hour. All these are described in the PSP texts as well as in the TSPi textbook.

<- Chapter 6 6.1 6.2 ->

Contents List of Forms Tool Instructions

6.2. Tracking Time

This lecture is based on the material in Chapter 3 of Introduction to the Personal Software Process. If any students are confused or want a further review of this material, suggest that they read this chapter. To track time, students need the following:

- A defined process or plan to follow
- A time tracking log or tool for recording the time
- Some way to tell time
- Work to do while tracking their time

After reviewing the basic concepts of time tracking, hand out the Exercise 1 package in Appendix A of this guide and have the students complete the time log (LOGT) entries for paragraphs 2 and 3 of the exercise scenario. A completed LOGT form with the correct entries is given in Appendix B of this guide under Exercise 1 Answer.

Have the students take a few minutes to complete the LOGT entries and then go over the correct answers with the class. If anyone got an incorrect answer, make sure that he or she understands the reason for the error before you proceed.

<- 6.1 6.2 6.3 ->

Contents List of Forms Tool Instructions

6.3. Defects, Code Reviews, and Checklists

The material in this refresher lecture comes from Chapters 12, 13, and 14 in the textbook *Introduction to the Personal Software Process.* If any students are confused or want a further review of this material, suggest that they read these chapters. In this lecture, briefly discuss the following topics with the class.

- Defects and quality
- Major and minor defects
- Defect types
- The defect recording log
- Code reviews
- The code-review checklist
- Using a code-review checklist
- A LOGD form and the LOGD instructions
- The defect-type standard
- A code-review script
- A code-review checklist (language-dependent)
- A small sample program with seeded defects (language-dependent)

After this discussion, give the students the Exercise 2 package in Appendix A of this guide and have them complete it. This package contains the source code for a small program, which they will review. While the students are doing the code review, they should record their defects on a LOGD form. Also have them use the code review checklist and the defect-type standard. Copies of all these materials are in the Exercise 2 package.

If the students know the Pascal programming language, you can use the exercise package in this guide. If they do not, you will have to prepare exercise materials in a language the students know. For a new exercise, you will need to provide the following materials.

For this new exercise package, you can use items 1, 2, and 3 from the Exercise 2 package but need to replace items 4 and 5 using a language the students know. You also need to prepare a LOGD form and a program listing with the seeded defects identified. You should test the program to make sure it works.

After the students have completed the exercise, hand out a copy of the correct answer and go over it with the class. Have each student calculate the yield he or she achieved in the review. Note that this simple program has several seeded defects, and anyone who makes a reasonable effort should find at least 50% of them. A competent review would find 70% or more, and a 90% or higher yield would be excellent.

Also emphasize the importance of doing careful reviews. Point out how much time the students will waste in test if they do not find most of the defects in the reviews and that their work will be much more efficient and predictable when they do. Emphasize that in the TSPi projects, the class schedule does not provide enough time for them to find all or even most of their defects in compile and test. Thus, if they do not do competent reviews, they will likely have to work several late nights in the final week of each development cycle. The answers to the Pascal version of Exercise 2 are given in Appendix B of this guide under Exercise 2 Answer.

<- 6.2 6.3 6.4 ->

Contents List of Forms Tool Instructions

6.4. Program Size

- The principles of size measurement

- How LOC are counted in the PS

After giving students a few examples, have them count the program in the Exercise 3 package in Appendix A of this guide. This program is also shown in Appendix B of this guide with each LOC numbered. Again, after the students complete the exercise, distribute the correct answer and ask whether everyone got the correct answer and whether there are any questions.

The material in this review segment is based on Chapter 6 of *Introduction to the Personal Software Process.* If any students are confused or want a further review of this material, suggest that they read this chapter. In this lecture, you should review

<- 6.3 6.4 6.5 ->

Contents List of Forms Tool Instructions

6.5. Task and Schedule Planning

The material in this lecture is based on Chapter 9 of *Introduction to the Personal Software Process.* If any students are confused or want a further review of this material, suggest that they read this chapter. Because the introductory PSP course does not require that students use earned value to track their projects, many will not know how to do it. It is therefore important that you take enough time to completely cover the material in this and the next section. I suggest you cover the following topics.

- How and why earned value is used
- The task plan
- The schedule plan
- Calculating planned value

Next, hand out the Exercise 4 package and have the students complete it. In this exercise, the students complete a task plan with task hours along with a schedule plan with weekly hours. They also make the planned-value calculations. Then they must show the planned value for each TASK on the task template and the planned value for each week on the SCHEDULE template. To complete this exercise, the students need a hand calculator or a computer with spreadsheet capability.

After the students have completed the exercise, hand out the exercise answer and go over the correct answers. If anyone got an incorrect answer, make sure he or she understands the reason for the error before you proceed. The correct answer for this exercise is given in Appendix B of this guide.

<- 6.4 6.5 6.6 ->

Contents List of Forms Tool Instructions

6.6. Estimating Project Completion

The material in this section is taken from Section 9.6 in *Introduction to the Personal Software Process.* If any students are confused or want a further review of this material, suggest that they read this section. The topics I suggest you cover are

- Calculating earned value
- Estimating project completion

After the lecture, give the students copies of the Exercise 5 package and ask them to determine the earned value achieved to date. Also, have them estimate the week when the project will be completed. After they have completed the exercise, review the correct results and check that all the students either got the correct answer or understand where they made a mistake. The correct results are given in Appendix B of this guide.

<- 6.5 6.6 Chapter 7 ->

Contents List of Forms Tool Instructions

Chapter 7
The Weekly Team-leader Meeting

This chapter covers the weekly team-leader meeting. By holding this meeting with the team leaders every week, you will be able to monitor team status, assess the teams' work, and identify problems. The topics in this chapter are as follows.

<- 6.6 Chapter 7 7.1 ->

Contents List of Forms Tool Instructions

7.1. Why Have a Weekly Meeting?

The objective of the weekly team-leader meeting is to have the team leaders help you manage the course. Unless this is a very small class or you have several teaching assistants, you will not have time to review all the work of every engineer on every team. You need the help of the team leaders to ensure that all the members of every team follow the TSPi. Ask the team leaders to alert you when anyone is having problems and rely on their judgment in handling problems. Explain to the team leaders that their job is to help you run this course. Although this puts them in the position of representing the team to you and you to the team, that is the nature of leadership. To be effective leaders, they must be able to handle such dual roles.

The team leaders must prepare for the weekly meetings with you. When they are prepared, the team status reviews should be quick and efficient. Then you can spend the bulk of the time discussing the teams' issues and problems. If the team leaders are not prepared, the meeting will take a great deal longer and will not be very productive. So insist that the team leaders be prepared. It will save a lot of time.

During these meetings, look at the team and individual data and assure yourself that the teams are following the process. Use Chapter 9 in this guide to decide which forms to examine and what questions to ask. Then tell the team leaders what you plan to cover next week. This means, of course, that you must also prepare for the team-leader meeting. After you have reviewed the data, discuss any teamwork problems or any other topics that the team leaders wish to raise.

Team courses provide a marvelous perspective on teamwork principles and issues. Strive to keep the meetings relaxed and open, and encourage the team leaders to participate in all the discussions. Ask for their comments and suggestions, and seek to learn from everyone's experience. This approach will make the meetings more productive and will provide the team leaders with a richer learning experience. It should also be more fun.

<- Chapter 7 7.1 7.2 ->

Contents List of Forms Tool Instructions

7.2. Why Review Team Data?

The principal reason to review the team's data is to ensure that all the students are following the process and that they are not having any serious problems. If the students suspect that no one is looking at their data, they will stop recording the data. Emphasize that to get a good grade in the course, they must gather and report all the required TSPi data.

As noted in Chapter 3, you will probably not have time to do all the reviews suggested in Chapter 9. If you start the team-leader reviews properly, however, you can have them do much of the work. Again, the guidelines suggested in Chapter 3 are as follows.

1. Make sure the teams are recording all the required data. You can generally check this by looking at the team and engineer TASK, SCHEDULE, SUMP, and SUMQ forms. When in doubt, also ask to see the engineers' LOGT and LOGD forms.

2. To anticipate problems and provide timely help, track the teams' progress using the team and engineer WEEK forms.

3. Try to examine each form the first time it is called for, as shown in Tables 7.1 and 9.1.

4. When forms are incorrect, require that they be corrected and reviewed with you again. If they are still incorrect, do it again until they are right.

Although recording and reporting data takes effort, the students should have learned how to do this in the PSP course. They should also know why the data are needed and how the data can help them to track and manage their work. If the students are using the TSPi tool, it will handle much of the work of gathering and reporting the data. If, for some reason, they are not using this tool, they need some automated support or else the volume of data will quickly become unmanageable. Some teams have developed a simple spreadsheet tool in only a few hours and found it to be very helpful.

<- 7.1 7.2 7.3 ->

Contents List of Forms Tool Instructions

7.3. Conducting the Team-leader Meeting

In addition to the team's and every engineer's WEEK forms, have each team leader bring an updated copy of the team's project notebook to the weekly meeting. Also ask them to bring copies of the team's time logs, at least for the first few weeks. Based on your plans for the meeting, they should also bring copies of the other materials you requested.

Meeting Roles

At the opening of each meeting, agree on who will be the meeting recorder. Someone can volunteer to do this every week, or the team leaders can choose to rotate the responsibility. The TSPi calls for every meeting to have a recorder and a timekeeper. You will presumably handle the timekeeper role, and the recorder's job is to note the key meeting results and produce a brief meeting report. This report should summarize only the action items and key decisions.

The Meeting Agenda

For the weekly team-leader meeting, I suggest the following standard agenda.

- Meeting roles
- The meeting agenda
- Outstanding items from the previous meeting
- Team-leader status reports
- Special issues or problems
- Topics for the next weekly meeting
- Outstanding action items

If the weekly meeting has a standard agenda, the team leaders will find it easier to prepare. They should know in advance that you will review the outstanding action items from the prior meeting, what data you wish to review, and which special issues or problems you plan to discuss. If you or any of the team leaders wishes to include additional agenda items, you should do so at the beginning of the meeting.

Outstanding Items from the Previous Meeting

Before the team leaders review their reports, review the outstanding items that you plan to cover. The team leaders should then cover their follow-up items during their team reports. Check each outstanding item, however, to ensure that none are forgotten.

Team Leader Status Reports

Each team leader should provide you copies of the forms specified for that week's review. The forms to review each week are shown by the E and T marks in Table 7.1. The E refers to the individual engineer or student data, and the T refers to the team data. The data to review for each form are discussed in Chapter 9.

The first week a form is called for, examine the data closely to ensure that the teams are completing the forms properly. If any are incorrect, have the team leaders get them corrected and, in the next meeting, review the corrected forms as well as the updated versions for the next week. After the team leaders understand what is wanted, their teams will start submitting correct data. When they do, you can stop reviewing those forms. You should, however, ask the team leaders every week whether they have examined all the team's forms and can attest that every engineer is properly recording and reporting all his or her data. You should also make an occasional spot check to ensure that they are continuing to do so. When teams do not record and track their data, it is generally because the team leader did not check. Make sure they know that they must check the data every week.

Table 7.1. Phases when TSPi Forms Should Be Completed

Form	Abbreviation		Process Phases							
		Engineer Or Team	**LAU**	**STRAT**	**PLAN**	**REQ**	**DES**	**IMP**	**TEST**	**PM**
Configuration Change Request	CCR	E[1]				X[2]	X	X	?[3]	
Configuration Status Report	CSR	T				X	X	X	X	
Student Information Sheet	INFO	E	X							
Inspection Report	INS	T				X	X	X		
Issue Tracking Log	ITL	T		X	X	X	X	X	X	
Defect Recording Log	LOGD	E/T				X	X	X	X	
Time Recording Log	LOGY	E		X	X	X	X	X	X	X
Test Log	LOGTEST	E						X	X	
Team Peer Evaluation	PEER	E								X
Process Improvement Proposal	PIP	E		?	?	?	?	?	?	?
Schedule Planning	SCHEDULE	E/T			X	X	X	X	X	X
Strategy Recording	STRAT	T		X			X			
Defects Injection Summary	SUMDI	T								
Defects Removed Summary	SUMDR	T								
Program Plan Summary	SUMP	E/T			X	X	X	X	X	X

Quality Plan	SUMQ	E/T			X	X	X	X	X	X
Size Summary	SUMS	E/T			X	X	X	X	X	X
Time Summary	SUMT	T								
Task Summary	SUMTASK	E								
Task Planning	TASK	E/T			X	X	X	X	X	X
Weekly Status	WEEK	E/T			X	X	X	X	X	X

Note 1: E means the form is completed with data for a single engineer, T means the form has data for the team, and E/T means the form is used for both engineer and team data.

Note 2: The X means that these forms should be completed in that phase and be available in the team's project notebook the next week.

Note 3: The ? means that these forms may or may not be completed in these phases.

Special Issues or Problems

If the team leaders have any special issues or concerns, they should cover them during their reports. If many team leaders have the same issue, however, you may wish to hold a more general discussion of the issue after all the team-leader reports.

Topics for the Next Weekly Meetin

At this point, use Table 7.1 to identify the forms you wish to review in the next weekly meeting. Also, tell the team leaders what you intend to ask about on each form so that they can be prepared. The key is for the team leaders to know that you will look at the team's data and that you will ask detailed questions about the data. If they are prepared, the reviews will take very little time. The items to review for each form are described in more detail in Chapter 9.

Summary of Action Items

At the close of the meeting, summarize the action items from the meeting. Do not rehash the issues; just summarize what the action item is, who will handle it, and when. Then have the meeting recorder note these items for the meeting report. Add any other significant items and provide copies of the report to all the attendees. The report should be brief and cover only the outstanding items, important decisions, and any key facts or data. Also, if there are more than a few action items, consider using the ITL form and an ITL tracking system to track them.

7.4. The First Weekly Meeting

You will hold the first team-leader meeting during team selection. Then, depending on the time taken by the PSP reviews, you may hold another team-leader meeting before the teams start on the projects. At the first meeting after the projects start, the teams will have completed only the LAU and STRAT phases. Because they will have limited data at this point, it is a good time to review the purpose of the weekly meetings and to discuss what you expect of the team leaders. You should also describe the data they need to bring to the next meeting.

The key data to review in the first weekly meeting are the engineers' time logs. Examine the logs and make sure that every engineer is properly entering his or her time. Discuss any problems and make sure that the team leaders know how the forms should be completed. If any forms were not correct, have the team leaders return them to the engineers to get them corrected. Then look at these corrected forms again the next week. Also look at the next week's LOGT forms for those same engineers to ensure that they are now completing them correctly. By taking a little time in the first meeting you can demonstrate to the class that you will not accept incorrect work.

When the students know you will be looking at their time logs, they are most likely to complete them properly. This is also true for all the other forms. Check each form carefully when it is first called for by the TSPi. When you find problems, have the team leaders take those forms back to the engineers to be corrected. Then review the corrected forms in the next meeting. List these incorrect forms in the action item list and make sure you check each one the next week.

In making the corrections, some engineers will complain that they cannot remember what the entries should have been. Although this is likely true, don't accept it as an excuse for not correcting the forms. Have them make their best estimate of the data they do not remember and label these data as estimated. Then insist that the students complete the rest of the form correctly. If you do not settle for incorrect work, the students will quickly learn not to submit it. As soon as they learn that, course management and grading will be much easier.

The other forms to review in the first week are the teams' STRAT forms and the ITL logs. Encourage the team leaders to use the ITL forms to track issues and then review these ITLs every week. In this way, you can see which teams are managing their problems and issues and which are not. You will also gain considerable insight into each team's performance.

<- 7.3 7.4 Chapter 8 ->

Contents List of Forms Tool Instructions

Chapter 8
The Process Phases

This chapter reviews the TSPi process phases and discusses some problems and questions students will likely have in each phase. There is one section in this chapter for each major phase.

- Launching teams
- The development strategy
- Team planning
- The requirements process
- Designing with teams
- Implementation
- Test
- The postmortem

The chapter concludes with a discussion of the final team report.

<- 7.4 Chapter 8 8.1 ->

Contents List of Forms Tool Instructions

8.1. Launching Teams

In the launch phase, the likely questions concern roles, goals, the team meeting, and data requirements.

Team Roles

During the launch phase, you allocate the students to teams and give them their role assignments. Although the roles are intended to be flexible, students occasionally view them as restrictive. Every project involves unknowns, and it is important that every team member be willing to take responsibility for handling unforeseen tasks. If the engineers think it is somebody else's responsibility, they should check with that person and see whether he or she needs help. The key requirement for a high-performance team is that everyone does his or her own job plus something extra. It is that something extra that makes great teams.

Team Goals

Most engineers would rather skip the subject of goals and get on with the job. Although this is understandable, teams need goals to provide an agreed basis for settling issues. Teams should review their goals, make sure they understand them, and then use these goals to guide their work. Also, suggest that the teams periodically reexamine their goals to make sure they have not lost sight of their overall objectives.

The Team Meeting

The team meeting is the principal communication mechanism among team members. It provides the framework for coordinating the team's work, and it provides the material used by the team leaders to report progress in your weekly meeting with them. The teams hold their first weekly meeting during the launch phase. In this meeting, the first order of business is to decide where and when to hold their regular weekly meetings.

Data Requirements

Before to each weekly meeting, every team member provides a completed personal copy of form WEEK to the planning manager. From these status reports, the planning manager prepares the team's WEEK report and distributes it to the team at their weekly meeting. During the launch phase, the team members discuss how to complete these forms and agree on when they will provide them to the planning manager.

8.2. The Development Strategy

Students are often confused about what a strategy is and why one is needed. Discuss these points and explain the criteria used for evaluating alternative strategies.

Producing the Strategy

In the strategy phase, the teams first create a conceptual design for their planned product. In doing so, they should consider the work to be done in all development cycles. Then they decide how to divide the work among the cycles and document the strategy on the STRAT form. Finally, suggest that they attach a brief description of the conceptual design to this STRAT form.

Workload Balance

Although teams should not attempt to develop too large a product in the first development cycle, they should also realize that the subsequent cycles will have much shorter schedules. In the standard course schedule, for example, cycle 1 is allocated six weeks, and cycles 2 and 3 are each allocated four weeks. Teams should be cautious about the amount of work they allocate to cycles 2 and 3.

System Infrastructure

Need statements typically define the functions users will see but say little about the system facilities needed to provide these functions. There is, however, a lot more to systems than the user-visible behavior. Examples of these system facilities are system control, I/O control, error handling, and file management. Although many teams fail to plan for this system infrastructure, it can add up to a lot of code.

The Configuration Management Plan

The TSPi configuration management process is described in Appendix B of the textbook. Unless the team modifies this process, the configuration management plan need only consist of naming the configuration control board (CCB) members, establishing CCB meeting guidelines, and ensuring that the team understands the configuration change request (CCR). The support manager also needs to produce a configuration status report (CSR) every week.

8.3. Team Planning

The planning process is discussed at length in Chapter 5 of the TSPi textbook. All the students should read this chapter before they start the planning phase. One issue that is often confusing is the relationship between the conceptual design produced in the strategy phase and the conceptual design used in the planning phase. They are the same thing. The teams produce the conceptual design in the strategy phase and use it in the planning phase.

A second possible area of confusion is the quality plan. The software process has many interrelated elements, and during quality planning students will begin to see how changes in phase time or defect-removal yield can affect product quality. A principal reason for producing a quality plan is to illustrate the importance of an early focus on quality. At the end of the planning phase, review each team's completed SUMQ form to make sure the data look reasonable. The quality standard in Appendix G of the TSPi textbook can help you in this evaluation. If the plans look neither realistic nor sufficiently aggressive, have the teams fix their plans and review them with you again the following week. Remember, if teams do not plan to do quality work, they probably won't.

A third common area of confusion is workload balancing. This is not something the teams should do once and forget. Depending on the accuracy of the team's plan, the engineers might even have to rebalance their workload every week. If someone falls behind, for example, the entire team should consider what it can do to help, even if it means reallocating everyone's work.

Finally, teams commonly either forget to schedule rework time after reviews and inspections or they do not schedule enough time. Check that they have made suitable allowances when they first make their plans.

<- 8.2 8.3 8.4 ->

Contents List of Forms Tool Instructions

8.4. The Requirements Process

Students are often confused about the need for a requirements document. They assume that the need statement describes what is wanted and wonder why a software requirements specification (SRS) is needed. The need statement, however, describes what the customer wants, whereas the SRS specifies what the team intends to build. The process of producing the SRS also helps the team agree on what it plans to do.

The System Test Plan

When teams wait to produce the system test plan until after the product is designed, they tend to test what they are building and not what the requirements specify. Such tests generally fail to identify missing functions or do not reveal major design errors. If you make the system test plan during the requirements phase, these problems are less likely.

Inspecting the Requirements and Test Plan

Because many students may not be familiar with the inspection process, have them read Appendix C in the textbook before they do the requirements inspection. You might also devote some class time to discussing the inspection process and explaining why inspections are important. The key point is to make sure that the teams inspect the requirements and that they also inspect the system test plan at the same time.

Requirements Baseline

After the SRS has been inspected, reviewed with the instructor, and approved, make sure that it is baselined. Also, check that the teams use the CCR procedure for every change to the baselined requirements and report these activities in the CSR for that week.

<- 8.3 8.4 8.5 ->

Contents List of Forms Tool Instructions

8.5. Designing with Team

In producing the first-cycle design, the teams should anticipate the enhancements planned for subsequent cycles. Although they should not yet design these latter functions, they should provide hooks for their later inclusion. Thus, in the initial cycle, teams define the overall product architecture but design only the first-cycle functions. Because finding and fixing high-level design mistakes during implementation and test can be extraordinarily difficult, each team should do a very careful design inspection.

Design Standards and Methods

If the students have not had a design course, they will need guidance on how to produce and document a design. Also, explain what a design standard is, why it is needed, and how to produce one. In addition, mention that they can use a standard that another team has developed as long as all team members agree to follow it.

Most students, and even many working engineers, do not truly believe that design is important. They may produce vague design notes or brief sketches or even start coding directly from the requirements. Insist that they produce a documented design before they start coding. The TSPi process will work with any design methodology, but if the teams have not been taught a specific design format, explain the four PSP design templates and require their use. These templates are described in Chapter 10 of A Discipline for Software Engineering.

Inspecting the Design and Test Plan

Again, make sure that the students have read Appendix C in the textbook before they do the inspections.

Design Baseline

Reemphasize to the teams that they must baseline their product designs after they have completed and inspected them. They must not then change these designs without using the CCR procedure.

<- 8.4 8.5 8.6 ->

Contents List of Forms Tool Instructions

8.6. Implementation

Just as in the PSP, the students should make personal implementation-phase plans for each program before they start its detailed design. They then document these plans on the SUMP and SUMQ forms. In doing this, they should follow the quality-planning methods described in Chapter 5 in the TSPi textbook and use the TSPi tool to produce the quality plan.

Detailed Design

In detailed design, the teams should use their agreed design standard. Again, stress the need for a complete and documented design and tell them that you will look at their design documents.

Implementation Standards

If the teams have not already produced their coding and LOC counting standards, they should produce them now and review them with all team members. To ensure that they produce these standards, ask the team leaders to bring copies to your next weekly meeting. Also, point out that if several teams want to share the same standards, they can do so. They should not waste time developing a new standard if they can find an acceptable one that someone else has produced.

The Unit Test Plan

The unit test plans need not be elaborate, but they should explain what tests will be run. For example, the plans should identify the loops to be tested, the variable values to be used, and the error conditions to be checked. They should test all variables and parameters at nominal values, at limit values, and outside these limits, and, where appropriate, they should test for overflows, underflows, zero values, empty and full states, date sensitivity, and so forth.

Test Development

For TSPi, the test development work will likely be modest, but the teams should still use disciplined methods. They should record the test development hours in the appropriate phase, but they should not count the LOC or defects against the system components. One approach is to name one or more test components and to record all the test development work against them.

Design and Code Reviews

In the implementation phase, require that the students do personal design and code reviews. Although you should look at how the engineers do these reviews, do not set numerical yield or rate targets because this could bias the data. Look at the SUMQ forms to see whether the students

are doing the reviews and whether they are getting reasonable yields. If they are not, again stress the importance of careful reviews and urge the teams to do better on the next development cycle.

Detailed Design and Code Inspections

Again, make sure that the students have read Appendix C in the textbook before they do these inspections.

Component Quality Review

The component quality review is the final quality check before integration and system test. Before a module or component is baselined, the quality/process manager examines the engineer's data and attests that the work was properly done. If there are any problems, he or she should review the situation with the CCB. If the quality problems appear serious, the CCB should decide on any remedial action and report the problem to the team leader and the instructor. The quality review is important because any single poor-quality component can often delay the entire team while they try to clean it up in system test.

Component Release and Baseline

Again, emphasize that the teams must baseline their products after they have completed and inspected them. They must not then change these products without using the CCR procedure.

<- 8.5 8.6 8.7 ->

Contents List of Forms Tool Instructions

8.7. Test

If the students have developed thorough test plans, carefully followed these plans, and observed the CCR process for all changes, they should have few problems in the test phase. There is, however, a chance that some teams will spend too much effort on the user documentation. They need write only enough so that a typical user could follow the documentation to install and use the product.

<- 8.6 8.7 8.8 ->

Contents List of Forms Tool Instructions

8.8. The Postmortem

At the end of each cycle, each team writes a brief summary report on its work. To ensure that all the teams produce comparable reports, suggest some basic report contents. At a minimum, the teams should compare their actual performance with their cycle goals and use TSPi data to justify their conclusions. They should also provide summary plan versus actual comparisons for

- Product size
- Development hours
- LOC/hour
- PDF
- Yield before compile
- Yield before system test
- Defect levels in compile
- Defect level in all test phases

The teams should also report their review and inspection rates and ratios and relate these values to the yields they achieved in these reviews and inspections. Also suggest that they show the defect-removal profiles for each system and for every component.

Preparing Role Evaluations

Discuss with the class the role evaluations and ask them to point out specific ways to improve the way each role was performed. Explain how you will use the evaluations in grading, and tell them that you will remove any identifying comments from the evaluations and make them available to help this and other teams in the future.

<- 8.7 8.8 8.9 ->

Contents List of Forms Tool Instructions

8.9. The Final Report

The teams should prepare and present a brief final report at the end of the course. In these reports, they should review their results across all development cycles and explain any significant variations. Have them briefly summarize their results, discuss how these results changed from cycle to cycle, and explain what caused the variations. Finally, ask them to summarize the key lessons they took from the course and what they would do differently on the next project.

<- 8.8 8.9 Chapter 9 ->

Contents List of Forms Tool Instructions

Chapter 9
Checking Forms and Data

This chapter discusses the forms to check at various TSPi phases and the ways to check them. After some general comments, we discuss the items to check on each form. For each form, the topics covered are things to check, interpreting the data, common reporting mistakes, and general comments.

<- 8.9 Chapter 9 9.1 ->

Contents List of Forms Tool Instructions

9.1. The Forms to Check

Most of the forms need to be checked at least once, and a few forms need to be checked every week. The forms to check every week are the WEEK form, the TASK and SCHEDULE forms, and, after the requirements phase, the SUMP and SUMQ forms. Once you know that the engineers can submit each form correctly, let the team leaders do the bulk of the reviews. However, make sure to ask the team leaders every week whether they have checked each engineer's forms and whether these forms are complete and correct. You should also occasionally do a spot check to ensure that the team leaders are doing these weekly reviews.

Table 9.1 is the same as Table 7.1, and it shows the phase when each form is used. The first form we discuss is the WEEK form. After that, the forms are discussed in the alphabetical order of their abbreviations.

CCR: the configuration change request

CSR: configuration status report

INS: inspection report

LOGD: defect log

LOGT: time log

LOGTEST: test log

SCHEDULE: schedule planning

STRAT: strategy recording form

SUMP: program plan summary

SUMQ: quality summary

SUMS: size summary

TASK: task planning

Table 9.1. Phases When TSPi Forms Should Be Completed

Form	Abbreviation		Process Phases

		Engineer Or Team	LAU	STRAT	PLAN	REQ	DES	IMP	TEST	PM
Configuration Change Request	CCR	E[1]				X[2]	X	X	?[3]	
Configuration Status Report	CSR	T				X	X	X	X	
Student Information Sheet	INFO	E	X							
Inspection Report	INS	T				X	X	X		
Issue Tracking Log	ITL	T		X	X	X	X	X	X	
Defect Recording Log	LOGD	E/T				X	X	X	X	
Time Recording Log	LOGY	E		X	X	X	X	X	X	X
Test Log	LOGTEST	E						X	X	
Team Peer Evaluation	PEER	E								X
Process Improvement Proposal	PIP	E		?	?	?	?	?	?	?
Schedule Planning	SCHEDULE	E/T			X	X	X	X	X	X
Strategy Recording	STRAT	T		X			X			
Defects Injection Summary	SUMDI	T								
Defects Removed Summary	SUMDR	T								
Program Plan Summary	SUMP	E/T			X	X	X	X	X	X
Quality Plan	SUMQ	E/T			X	X	X	X	X	X
Size Summary	SUMS	E/T			X	X	X	X	X	X
Time Summary	SUMT	T								
Task Summary	SUMTASK	E								
Task Planning	TASK	E/T			X	X	X	X	X	X
Weekly Status	WEEK	E/T			X	X	X	X	X	X

Note 1: E means the form is completed with data for a single engineer, T means the form has data for the team, and E/T means the form is used for both engineer and team data.

Note 2: The X means that these forms should be completed in that phase and be available in the team's project notebook the next week.

Note 3: The ? means that these forms may or may not be completed in these phases.

The INFO form is not discussed because it is used only at the beginning of the course and need not be reviewed. Similarly, the PEER form is used only at the end of each cycle. The teams should be encouraged to submit PIP forms to help them remember issues to discuss during the postmortem. It is also a good idea to check whether the PIP form is being used. Finally, the SUMDI, SUMDR, and SUMT forms are not reviewed because they summarize data that already appear on the SUMP forms. These forms are included in the TSPi process only as an aid for people who do not use the TSPi tool.

9.2. The WEEK Form

The first time teams can complete the WEEK form is after they have completed the PLAN phase. Each team must then complete and submit a WEEK form every week, and each student must submit a WEEK form to his or her planning manager each week. The planning manager then consolidates these data to produce the team's WEEK form.

In the beginning of the course, review the WEEK forms carefully to make sure they are correct. As soon as the teams know how to complete this form, continue to review it as a way to monitor the teams' progress.

Things to Check: WEEK Form

Check first that you have a WEEK form from each team and from each engineer on each team. Then check that all the plan and actual entries are completed for the weekly data and that the plan and actual hours and earned values are reported properly for each engineer. Also check that the earned values are rolled up correctly for the team and that the team's earned value is consistent with the tasks the engineers showed as complete in their personal WEEK forms.

The earned-value check can be tricky because the engineers' WEEK forms report EV against their personal plans, whereas the team's WEEK form reports EV against the team's plan. To total the engineers' EV, you must first convert them to team EV. Do this by finding the total hours on each engineer's TASK form and adding these hours to give the total planned team hours. Then, for each engineer, take the ratio of his or her total planned hours to total team planned hours and use that ratio to adjust that engineer's EV.

For example, with five engineers, where A planned a total of 35 hours, B planned 37.5, C planned 40, D planned 42.5, and E planned 45, the total team planned hours for this cycle are 200. Engineer A's EV is thus worth 35/200 or 0.175 of the team EV. Thus, if engineer A reported 16 EV in one week, then his or her contribution to team EV that week would be 16*0.175 = 2.8.Although the TSPi tool will make these calculations, it is a good idea to check to make sure that the tool is being used properly. If it is, and if the values are correct, you need not check this again as long as the numbers continue to look reasonable.

In addition to EV, check the items in the Issue/Risk Tracking section of the WEEK form to see whether the team is tracking and managing its risks and issues.

Interpreting the Data: WEEK Form

After satisfying yourself that the data are complete and accurate, look at the team's EV versus PV. If EV is running late, probe for the causes. Generally, there are two cases.

1. The hours are greater than or near to the plan and the EV are less than the plan. This is

usually because the team has underestimated the work and is trying to do too much. If the engineers are putting in a reasonable amount of time each week, suggest that the team reduce the size of the job they are trying to do.

2. The hours and EV are both less than the plan. Here, there are two cases: Either the team members planned to spend more time than they really did or the workload is unbalanced.

If the plan is too ambitious, the team should cut back on the job. To check the balance of the team's workload, compare the hours worked by each engineer in the Team Member Weekly Data section of the WEEK form. If one or two engineers are working much more or much less time than the others, ask the team leader to explain. Unbalanced workload is an early indicator of teamwork problems. Also, see whether any engineers have much more or much less EV than the others. Again, probe this to see whether someone is running into trouble. A useful early indicator of trouble is that an engineer works a reasonable number of weekly hours but does not earn much EV. To check this, look at the hours per EV for each engineer on the team.

Common Reporting Mistakes: WEEK Form

The most common mistake on the WEEK form is leaving out data. Engineers also often calculate the EV incorrectly or forget to note when tasks were completed. They also occasionally check that the wrong tasks were done. This error would show up as incorrect EV totals for the engineer and the team.

General Comments: WEEK Form

It often takes several weeks before teams start submitting correct WEEK forms. Keep checking the engineers' and teams' WEEK forms every week until they are correct. Thereafter, use the WEEK forms as a way to check on team progress.

[<- 9.1](#) 9.2 [9.3 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

9.3. The Configuration Change Request: CCR

The CCR form should first be used when the SRS is baselined. Thereafter, it should be used every time a new product is baselined and every time a baselined product is changed.

Things to Check: CCR Form

Check that the header and product information are correct and that the status and approval sections are complete. The teams need not complete the change information for new product submissions, but they should provide this information for every change.

Interpreting the Data: CCR Form

There is not much on the CCR form to interpret. Either the appropriate data are submitted and are correct, or they are not. If there are errors or omissions, get the form corrected and look at it again the next week.

Common Recording Mistakes: CCR Form

The most common mistakes with the CCR form are not submitting change requests for baseline changes or not completing the data for product size, owner's name, or backup address. For changes to baselined products, the principal omission will probably be the change information. The bulk of the TSPi change activity will probably be changes during integration or system test. Often the only information needed for these changes is that the change is needed to fix a defect found in integration or system test. If, however, the change requires correction of the requirements or design materials, then the change impact and change description information should describe what must be done to the SRS and SDS. It is then important to ensure that the SRS, SDS, detailed design, and code are all updated to reflect the change. Similarly, there may also have to be changes in test cases and test plans.These impacts should also be mentioned.

Note that if not all the required items are corrected for every change, there will likely be problems in subsequent development cycles. That is why the teams need to religiously use their SCM process.

General Comments: CCR Form

Teams often argue that the change control procedure is unnecessarily complex and that they should not have to follow it for the simple changes they have to make in test. If the changes are truly simple, however, the change control board (CCB) procedure would also be simple. The reason for the procedure is that seemingly simple changes often turn out not to be very simple when someone looks at them. All the SCM procedure does is to make sure that someone looks at them.

9.4. The Configuration Status Report: CSR

The CSR form is first used to report configuration status after the SRS document has been developed, inspected, and baselined. This is done in the requirements phase. The SRS will be the first product to be baselined. Until it is, there is nothing to report on the CSR form.

Things to Check: CSR Form

The principal thing to check is that the team has baselined the SRS document and that the support manager has recorded the number of text pages. Thereafter, check each week to make sure that all appropriate new items were added when they were completed and inspected. This would be the SDS at the end of the design phase and all the source code at the end of implementation. In addition, make sure that all changes to baselined products are approved with a CCR, that the proper INS forms are attached to the CCR, and that all changes are reported on the CSR.

Interpreting the Data: CSR Form

For the modest-sized TSPi projects, you will not generally be able to draw much useful information from the CSR reports. With larger projects, the CSR data can indicate when the requirements have settled down and when a product is stabilizing in integration or system test. These data can also show code growth during implementation as well as the decline of change activity as testing nears completion. These analyses, however, require trend data over an extended period. With TSPi, there is not enough time to get trend data. You can look at the volume of test defects and compare the results among teams or development cycles, but because the test phase will takes one week or less, there will be not be weekly trend data.

Common Reporting Mistakes: CSR Form

The principal mistakes are that teams either do not baseline all their products or they do not properly track their configuration and change activity and status.

General Comments: CSR Form

Again, for the small TSPi products, the software configuration management process need not be overly formal. The students should recognize that the CCR process is a mechanism to ensure proper consideration of every change. Teams need to learn to manage all their baseline changes, not only to control the projects but also to maintain control of quality from the beginning of the job.

9.5. The Inspection Report: INS

The INS form is used to report the results of all inspections. You need to get at least one INS form at the end of the requirements phase for the SRS inspections and get one at the end of the design phase for the SDS inspection. You also need at least two INS forms for every component after implementation: one for the detailed design inspection and a second one for the code inspection. In general, it is a good idea to inspect products in segments that can be covered in about two hours or less. Longer inspections tend to have much lower yields. Thus, for large SRS or SDS documents and large modules, you should get multiple INS forms for each inspection.

Things to Check: INS Form

Check that the moderator has completed all the INS-form entries, that all the major defects are reported, and that the engineers who found each major defect are checked off. In addition, make sure that the inspection summary data are properly calculated and complete.

Interpreting the Data: INS Form

The first thing to evaluate with each INS form is whether all the engineers made a reasonable preparation effort, as shown by their preparation rates. You can compare these rates with the QUAL standard in the textbook (in Appendix G) to see whether they are reasonable. Second, see if the inspection achieved a reasonable yield. With care, PSP-trained engineers should reach yields of 80% or higher. For experienced engineers, an inspection yield of less than 70% is unacceptable and calls for a reinspection. For students, however, the defect levels will likely be higher and the yields lower. The yield estimates should still give you a useful perspective on the team's and each engineer's inspection performance.

Third, compare the defect levels for several teams or prior courses to see whether the results look reasonable. If any results appear unusually high or low, ask the team leaders to have the quality/process managers make an analysis for the next meeting. Often the problem is caused by inclusion of minor defects or by large numbers of coding errors by inexperienced programmers. If, however, the numbers are unusually low, it is likely that the engineers did superficial reviews or did not record all the defects.

Finally, when the inspection yields are too low, check that the teams are reinspecting their products. Often, in fact, student yields will be 30% or less. When they are this low, have the students do a reinspection and make sure they do it properly. They will almost certainly find more defects.

Common Reporting Mistakes: INS Forms

Check that the yields and summary data are calculated correctly. A common error is that the A and B columns in the Defect Data section are incorrect. Also, students may misunderstand the meaning of unique defects or they may include minor defects when they should use only major defects for estimating the remaining defects.

General Comments: INS Form

The inspection process provides the best check on the quality of the engineers' work. If teams find more defects in the code inspection than the developer found in the code review, product quality is probably poor. Similarly, engineers should find many more defects in detailed-design reviews than are found in the detailed-design inspections. When this is not the case, the product will likely have many remaining design defects.

If inspection yields are less than 70% in the implementation phase, try to find out why they are low. If the problem is too high review rates, suggest that the team do a reinspection. Usually, the engineers will not spend enough time in preparation. Also, they often do not use an up-to-date checklist. Suggest that they look at the defects found in the last inspection and make sure that no similar defects remain. Have them use a trace table on every loop structure and check that every state machine is complete and orthogonal.[1] Emphasize the importance of improving the yields in the next development cycle. If this is a common problem, give a lecture on yield to the entire class.

---

[1] For further information on state machine verification and trace tables, see Humphrey 1995, pages 335 and 400.

<- 9.4 9.5 9.6 ->

Contents List of Forms Tool Instructions

9.6. The Defect Recording Log: LOGD

The first time the LOGD form is completed is by the inspection moderator in the requirements inspection. Another LOGD form is also completed by the moderator for each inspection thereafter. Note that in TSPi, the quality/process manager normally acts as the inspection moderator.

The engineers should not use the LOGD forms to report the defects they find in inspections. Otherwise, when several engineers found the same defect it would be counted multiple times by the TSPi tool. To avoid this problem, the quality/process manager completes one LOGD form at each inspection and makes sure that each major defect is entered only once in the TSPi tool.

The engineers start reporting the defects they find on the LOGD form during the implementation phase. Look at these LOGD forms to ensure that everyone is completing them properly.

Things to Check: LOGD Form

First, check that all required data are entered for every defect. This includes defect number, type, phase injected, phase removed, fix time, and comments. For a fix defect, the engineers should also complete the fix defect box.

Interpreting the Data: LOGD Form

It is most important to check that the students record all their defects. Although there is no way to know this precisely, several consistency checks can generally indicate when students are not recording all their defect data. First, determine the defect-removal rates for each defect-removal phase and see whether they are reasonable. For example, compare the number of compile defects with the time spent in compile. In general, engineers will find between 10 and 20 defects per hour while compiling. When engineers report one or two defects in 45 minutes of compiling, they are almost certainly not reporting all the defects found. When engineers do not report all the compile defects, they are probably not reporting their other defects. The QUAL standard in Appendix G in the textbook provides guidelines on reasonable defect-removal rates.

Also check that the total defects/KLOC are reasonable. Although there can be wide fluctuation in defect levels among engineers and among programs for the same engineer, engineers with consistently low defect levels often have a distinctive process. They generally have high design-to-code time ratios, low compile and test times, and good review practices. If the engineers' processes do not have these characteristics and if they also have low total defects/KLOC, it generally means they are not reporting all the defects.

When you discuss this problem with the class, explain the importance of recording all the defects and the fact that a number of students are not doing so. Do not, however, explain how you determined this. If you give reasonableness criteria for defects/KLOC, removal rates, or time ratios, students are likely to adjust their time or defect data to meet your criteria. This would be the worst possible case. You would not be

able to tell who was or was not reporting defects.

Common Reporting Mistakes: LOGD Form

The typical errors on LOGD forms are either incomplete data on the defects reported or defects that are not reported. For example, every defect report must indicate the product element in which the defect was found. During the requirements and design phases, this is "system." In implementation and test, it is the specific program module or component.

Another common mistake is to incorrectly report the phases. The phase removed should not be before the phase injected. Also, the total defect fix times should be consistent with the time-in-phase reported on the LOGT form. Finally, the comment section should describe every defect in enough detail that you can understand what it was.

General Comments: LOGD Form

When students are sloppy in their quality practices, they usually start by skipping entries in the LOGD form, generally during compile. It is important to check this form carefully and to make sure that the team leaders and quality/process managers check every engineer's LOGD forms every week.

<- 9.5 9.6 9.7 ->

Contents List of Forms Tool Instructions

9.7. The Time Recording Log: LOGT

Engineers start using the LOGT form at the very beginning of the TSPi process. You should get LOGT forms in the first meeting after the strategy and planning phases.

Things to Check: LOGT Form

In checking the LOGT forms, make sure that the engineers track all the time they work on the project and that they properly report this time against the process phases and the product components. In the STRAT phase, for example, their time should be recorded in the strategy phase and rolled up to the strategy and planning entry on the SUMP form. During the strategy, planning, requirements, and design phases, the product entry should be "system."

Interpreting the Data: LOGT Form

LOGT contains pure data, so there is not much to interpret.

Common Reporting Mistakes: LOGT Form

The most common mistake with the LOGT form is that students leave out important information. With the TSPi tool, this should not be a problem. However, even with the tool, students may report incorrect phase information or incorrectly name the product. When they do this, the TSPi tool cannot properly summarize the data.

General Comments: LOGT Form

As long as you start the course requiring that the LOGT form be properly completed, it should not be a problem. You can best ensure proper reporting by looking at the LOGT forms the very first week and insisting that any problems be corrected and resubmitted. Because correcting incorrect or incomplete LOGT forms on the TSPi tool is quite inconvenient, students will find it much easier to enter their time data correctly when they do the work rather than try to fix it later.

[<- 9.6](#) 9.7 [9.8 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

9.8. The Test Log: LOGTEST

The TSPi process calls for the engineers to complete LOGTEST forms for all the tests they run. Thus, LOGTEST is first completed for the unit tests in the IMP phase and then for the integration and system tests in the test phase.

Things to Check: LOGTEST Form

Check that all entries in the LOGTEST form are completed and that the values entered make sense and are consistent with the team's other data. For example, when an engineer spends several hours in test, there were probably several unsuccessful test runs. All these should be reported along with the final correct run.

Interpreting the Data: LOGTEST Form

A useful measure of process quality is the percentage of the total development schedule spent in test. With the data reported in the LOGTEST form, you can determine this percentage. Software organizations typically spend about 40% to 60% or more of total development time in test. Organizations with very high quality processes spend about 20% or less of their time in test. With care, TSPi teams should spend 20% or less of their time in test.

Common Reporting Mistakes: LOGTEST Form

Typically, engineers either do not complete LOGTEST for every test or they leave out key entries. Examples are not identifying the test that was run, omitting the test phase (such as system test), not describing the product configuration being tested, or forgetting to note whether the test was successful. Also, in the comments section, they should reference where any other key information can be found. Another common error is to enter obviously incorrect data. For example, one obviously incorrect report would show a several-hour test with no defects found, or a test run with defects and an incorrect result but no report of a correct run for that test. Did they get the defect fixed and not bother to rerun the test, did they ignore the problems, or did they fail to report the correct run? Although the latter case is most likely, you cannot tell without a properly completed form.

General Comments: LOGTEST Form

Although useful in unit testing, the LOGTEST form is particularly important in the test phase. For example, when teams develop products in several cycles, they will want to know which tests were run and which product versions were tested. Without these data, efficient regression testing is not possible. Similarly, the LOGTEST data can help to determine the degree to which each team's product was tested.

9.9. The Schedule Planning Template: SCHEDULE

Because the SCHEDULE form is first completed in the planning phase, you should examine both the teams' and the engineers' SCHEDULE forms right after that phase.

Things to Check: SCHEDULE Form

For both the teams and the engineers, check that the planned weekly hours are reasonable, the weekly planned values are correctly calculated, and the weekly planned values match the TASK form planned values.

Interpreting the Data: SCHEDULE Form

The principal things to check with the SCHEDULE form are that the weekly hours per engineer are reasonable and that the weekly hours for each engineer on a team are reasonably balanced.

Common Reporting Mistakes: SCHEDULE Form

A common mistake engineers make during planning is to assume that all the time they work on the project will be productively used in completing tasks. The students will spend a significant amount of time attending team meetings and classes, and performing such miscellaneous tasks as getting supplies, setting up computing systems, or consulting with instructors and teaching assistants. To address this problem, you can either have the students include a management and miscellaneous (M&M) task in each phase to cover this time or have them reduce their overall scheduled time and not track this time. Either approach will work, but it is usually wise to record all the time data because it will help the students make more-accurate plans. You should, however, have all teams handle this issue in the same way. Otherwise, the team data will not be comparable.

The engineers should expect to spend more time on the project than is required for their development tasks. In industrial teams, we call this *task hours*; most teams are unable to average even 20 task hours in a standard 40-hour work week. Although student teams generally have fewer overhead tasks than industrial teams, teams should be cautious about committing too many task hours, at least until they have some data to guide their planning.

General Comments: SCHEDULE Form

The SCHEDULE and TASK forms are key in tracking team progress. It is important that the teams learn how to complete them correctly and to ensure that the engineers update their SCHEDULE and TASK forms every week. Have the team leaders check these forms every week, and look at them yourself to monitor team progress.

9.10. The Strategy Recording Form: STRAT

The STRAT form is first used in the strategy phase, and it is updated during design.

Things to Check: STRAT Form

During the strategy phase, check that the teams have identified all the functions they plan to implement and that each function has a need-statement reference. The teams should also define the cycle when each function will be implemented, estimate the LOC for each function, and total these LOC for each phase. Then, during the design phase, the teams will refine these data and identify which components will implement each function. The teams may choose to update the LOC for each function, but that is not necessary unless the plan has changed significantly.

Interpreting the Data: STRAT Form

Although the STRAT form is quite simple, there are several things to look for. Teams should not plan to develop too much product function in the first cycle. Without historical data, they are likely to try to do too much. Because the schedule for cycle 1 is considerably longer than for cycles 2 or 3, however, teams must also be careful not to defer too much function to latter cycles.

Teams also commonly forget about the system infrastructure or glue. They generally plan for all the functions defined in the need statement, but they often make little or no allowance for the code to control the system, handle I/O, manage files, generate error messages, and provide the user interface. Because these functions can take substantial code, teams frequently end up with much more work than they had anticipated. You can generally identify this problem from the team's STRAT form.

Common Reporting Mistakes: STRAT Form

In the strategy phase, the most common STRAT form mistakes are omitting system functions, leaving out LOC or hours entries, or not providing a need-statement reference for each function. During design, the students may drop need-statement traceability or not identify the modules that provide each function.

General Comments: STRAT Form

The STRAT form provides a record of a team's strategic decisions and a way to track what the engineers did versus what they planned.

9.11. The Program Plan Summary: SUMP

The SUMP form should be completed as soon as teams start producing products. Thus, you should look at SUMP forms at the end of the requirements phase and at every phase thereafter. Then, during implementation, ask to see completed SUMP forms for every module or component as well as for the total system.

Things to Check: SUMP Form

When teams use the TSPi tool, it automatically generates the SUMP form from the engineers' time, size, and defect data. The principal omissions are to forget to enter size data for the requirements and design documents or to fail to record inspection data. Assuming that the engineers use the tool to record their time, size, and defect data and that they report these data properly against the products and phases, the SUMP forms should be correct.

Interpreting the Data: SUMP Form

The SUMP form provides an overall view of the project, and it is the easiest place to check the completeness of the engineers' data. The principal items to examine are the size data and the times spent in the process phases. Also check that defects are recorded and that the total defects injected equal the total defects removed.

Common Reporting Mistakes: SUMP Form

If the teams use the TSPi tool, there should be no reporting or calculation mistakes on the SUMP forms. If they are not using the tool, the principal problems are in accurately summarizing the data by phase and product. Unless they do this religiously every week, teams will likely have trouble finding all the data. TSPi teams generate substantial volumes of information, and unless they have a tool to handle these data, updating can be a substantial chore. Most of the data will simply get lost in the mass of project information. Thus, if the teams are not using the TSPi tool, check the SUMP forms carefully every week to ensure that the teams are keeping them up-to-date.

General Comments: SUMP Form

The SUMP forms summarize the teams' and each engineer's work. They provide an overall view of each team's results and are the best reference for assessing a team's or engineer's data.

9.12. The Quality Plan: SUMQ

Like the SUMP form, SUMQ should be completed as soon as teams start producing product elements, and it should be updated every week thereafter. The engineers should complete a SUMQ form for the overall system and one for every component or module.

Things to Check: SUMQ Form

If the teams are using the TSPi tool, the SUMQ form will be completed automatically, and the engineers will not have to make any further calculations or enter any further data.

Interpreting the Data: SUMQ Form

The SUMQ form provides the information needed to check on the quality of the team's and the engineers' work. Review this form weekly to determine whether the engineers are doing reviews and inspections, are spending enough time in these reviews and inspections, are achieving reasonable review and inspection yields, and are reporting appropriate defect levels. All these questions can be answered with the data on the SUMQ form. The QUAL standard in Appendix G of the TSPi textbook provides guidelines for assessing the SUMQ data.

Common Reporting Mistakes: SUMQ Form

When students complete the SUMQ form manually, they will likely make many calculation errors. Initially check all the calculations and then have the team leaders check them weekly thereafter and have any problems corrected. When the same problems are repeated among several teams, discuss the topic with the entire class.

General Comments: SUMQ Form

SUMQ is a very important form. It provides an overall picture of the quality of the teams' and each engineer's work, and it provides the data to determine how faithfully the engineers are following the TSPi process.

[<- 9.11](#) 9.12 [9.13 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

9.13. The Size Summary: SUMS

The SUMS form is first used during the planning phase to record the estimated sizes of the products each team plans to develop. Thereafter, the SUMS form is used to record the actual sizes of the products produced in every phase.

Things to Check: SUMS Form

The principal thing to check with SUMS is that the size data are entered for all the products produced and that these data are correctly recorded with appropriate size units. Also make sure that the engineers properly enter the size data in the base, deleted, modified, added, reused, new and changed, and total columns.

Interpreting the Data: SUMS Form

Because the SUMS form contains raw data, there is nothing to interpret.

Common Reporting Mistakes: SUMS Form

The most common mistakes with SUMS are forgetting to enter size data or entering these data in the wrong place or with the wrong size units.

General Comments: SUMS Form

Although the SUMS form is important, it does not provide any information that you cannot obtain from the SUMP forms. In fact, it is by completing the SUMS form that the teams provide the size data needed to generate the SUMP and SUMQ forms.

<- 9.12 9.13 9.14 ->

Contents List of Forms Tool Instructions

9.14. The Task Planning Template: TASK

The TASK form is first completed in the planning phase and updated every week thereafter.

Things to Check: TASK Form

The first thing to check on the TASK form is that the engineers have produced a sufficiently detailed plan. No engineer should have tasks that are planned to take more than about ten hours. If any do, suggest ways to subdivide these tasks and have the team leader get the forms corrected and review them the next week.

If the team is using the TSPi tool, much of the work of completing the TASK form will be done by the tool. Then all you need to check are that the tasks are properly entered and that the proper products and phases are identified for each task. Also, examine the actual hours for each completed task to ensure that the engineers have been properly entering the tasks on the LOGT forms. Finally, make sure that the engineers entered the week when each task was done.

Interpreting the Data: TASK Form

From the TASK form, you can determine whether the teams have a reasonably balanced workload or one or more of the engineers is overloaded. After the teams have worked for a week or so, examine the updated TASK forms to see which students have completed their tasks on time and which ones are falling behind. The data on the TASK form also show how the team is progressing and can help you anticipate problems. A typical problem is that one or two engineers are consistently late and are delaying the entire team. The team leader should then consider rebalancing the team's workload to reduce the work for these engineers.

Common Reporting Mistakes: TASK Form

If the teams are not using the TSPi tool, they must complete the TASK form manually. This requires that the engineers obtain their time data from the time logs and accumulate these data on the TASK form against the proper tasks and products. They must also note the week each task was completed and determine the earned value for each completed task. They generally have a substantial volume of data, so this is a highly error-prone activity.

General Comments: TASK Form

The TASK form provides a key tool for tracking team progress. It is important that the engineers complete it correctly and that they update it every week. Examine the TASK forms every week to ensure that the teams are updating them properly and that the work is proceeding roughly as planned.

Chapter 10
Later Development Cycles

This chapter reviews some issues that will likely arise in the second and third development cycles. These issues are covered in the order of the phases in which they occur.

<- 9-14 Chapter 10 10.1 ->

Contents List of Forms Tool Instructions

10.1. Launch Issues

It is generally not a good idea to change team members in the middle of the course. It takes time to build effective teams, and once a team starts developing effective working relationships, you should let this process continue. Occasionally, however, changes are necessary. Some students may have family or health problems, and others may not be able to do the work. In some cases students cannot work effectively on a team. For a discussion of these teamwork issues, see Chapter 12 in this guidebook.

Changing Team Roles

If you let the students decide, they will probably not want to switch roles. However, because students will learn more by switching roles, you should urge them to do so. It is generally best, however, to leave the final decision up to the teams.

Goals

In the second and third cycles, the teams should review how they have performed against their current goals and establish new goals for the next cycle. Starting with the second cycle, teams can use their prior experience to set more-realistic goals. Then have them document their new goals, give you a copy, and put a copy in the project notebook. Make a point of looking at these goals and asking questions about them. The only way to get teams to treat their goals seriously is to take them seriously yourself. Then, at the end of the cycle, have the teams report performance against their goals.

<- Chapter 10 10.1 10.2 ->

Contents List of Forms Tool Instructions

10.2. Strategy Issues

In later development cycles, teams occasionally want to change their development strategies. If they built a system that works, however, they should build on what they have done. There will always be better ways to build systems, and teams are most likely to think of these better ideas right after they have completed the first cycle. If they keep switching to the next best idea, however, they will never get anything built. The time for great new ideas is in the first development cycle. In the second and third cycles, the teams need to concentrate on finishing the job they have started.

<- 10.1 10.2 10.3 ->

Contents List of Forms Tool Instructions

10.3. Planning Issues

If the teams underestimated the job on the first cycle, they should not adjust their second-cycle estimate to compensate. Although they should use prior-cycle data to make better estimates, they should not make bias adjustments without a substantial amount of data. After they have data on three or more project cycles, they should use a statistically sound method such as PSP PROBE to make any bias adjustments [Humphrey 1995]. If they make bias adjustments based on one project's data, however, they will often make the estimates worse.

<- 10.2 10.3 10.4 ->

Contents List of Forms Tool Instructions

10.4. Requirements Issues

The requirements problems in the second and third cycles are much like the strategy issue discussed earlier. Teams will have many ideas on better ways to build this product. Although they should take advantage of their new knowledge, they should build on what they have done and not start on a new direction.

<- 10.3 10.4 10.5 ->

Contents List of Forms Tool Instructions

10.5. Design Issues

Assuming that the prior-cycle design worked, the principal later-cycle design issues concern modification. Each cycle tests the earlier design and, if the prior-cycle enhancement hooks were well thought out and properly implemented, the enhancement work should be relatively straightforward. Inexperienced teams, however, often make major design mistakes. If the original design had fundamental flaws, the team may have to make major changes to get the system to work.

To address this question, the engineers should first try to fix the known problems without a major redesign. If this is not possible, they should then consider adjusting the requirements. Finally, if it appears that the first-cycle design is fundamentally flawed, suggest that the team treat it as a prototype. They should then throw it away and produce a new and properly designed first-cycle product. By doing this, they will produce a better product, their work will be more efficient, and they will learn more than they would have by reworking the old design.

<- 10.4 10.5 10.6 ->

Contents List of Forms Tool Instructions

10.6. Implementation Issues

The principal implementation issues in the second and third development cycles concern modification quality and configuration control.

Modification Quality

There is substantial evidence that engineers make many more errors when modifying programs than they make in new development. Two common reasons for this are that the original design was not clear or that the modifications themselves were not completely designed. By using sound design practices, the team will provide a quality base for future enhancements. Then, in subsequent cycles, the teams should update these designs and do thorough design inspections. This approach will improve the quality of the current work and greatly simplify the work in subsequent cycles.

Configuration Control

A principal reason for having a software configuration management (SCM) system is to maintain control of product versions. When teams have sound SCM practices, they rarely have control problems. If, however, they have not used proper SCM practices, they are likely to misplace component versions, lose track of defect fixes, or put out-of-date modules into test. When teams have such problems, urge them to discuss these problems and their causes in the final report presentations. One team's experiences will have more impact than a semester of faculty lectures.

<- 10.5 10.6 10.7 ->

Contents List of Forms Tool Instructions

10.7. Test Issues

Cyclic development is highly effective with high-quality products. The reason is that testing costs increase exponentially with product size. Thus, if teams develop their products in small increments and if each increment is built on a high-quality base, testing for the new increments can concentrate on the new code. When teams do this, they have fewer defects to fix in test. Moreover, the time to find and fix these defects will be sharply reduced.

With poor-quality products, it can take many hours to find and fix each defect in system test. The reason is that the defects found while testing a new product enhancement could be anywhere in the total system. This is why it is important for teams to carefully review and inspect every product element. When they do, a few carefully designed tests will generally find the bulk of the remaining defects. This then produces a high-quality base for the next development cycle. This approach not only improves the quality of the products being developed but also reduces the testing time for all subsequent product versions, often by as much as ten times.

Regression Testing

As part of the system test plan, make sure that the teams regression test the prior-cycle functions. To construct these regression tests, they should use the prior-cycle LOGTEST data to select the most effective tests. The tests that found the most defects in prior cycles are usually the best ones to use for regression testing.

<- 10.6 10.7 10.8 ->

Contents List of Forms Tool Instructions

10.8. Postmortem Issues

Although the postmortem issues are generally the same from cycle to cycle, this is the time to think about improvement. If you have the teams focus on improvement at the end of the course, hopefully they will realize that self-improvement should not stop at the end of the semester.

[<- 10.7](#) 10.8 [Chapter 11 ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

Chapter 11
Grading Team Courses

A great deal could be said about grading. This chapter addresses only the special issues of grading team courses. These issues concern teamwork, measurement, and overall effectiveness.

<- 10.8 Chapter 11 11.1 ->

Contents List of Forms Tool Instructions

11.1. Teamwork

Although the students must be competent software engineers, and although they need to build quality products, a principal course lesson concerns working with other engineers. With the TSPi, students should learn how to use a defined and measured process to cooperatively develop quality products. There are thus three elements in evaluating student performance:

1. How well the student worked with the team
2. How well the student followed the TSPi process
3. The student's overall effectiveness in producing quality products on the committed schedules

All these elements are important and should be considered in establishing grades. The approach I suggest is as follows.

- Rank the student as excellent if he or she did well in all three respects.
- Rank the student good if he or she did at least adequately on all the criteria
- Rank the student fair if he or she did poorly on any of the criteria.
- In other cases, rank the student as poor.

The following cases explore some possible situations.

- Case 1. Peter was a brilliant loner who could not work with his team. He was design manager and had strong views on how to build the product. Whereas the other team members tried to contribute, Peter was impatient and largely ignored them. Because Peter was such a good engineer, however, he produced the entire design himself and wrote almost all the code. In the end, Peter, almost single-handedly, produced a working product.

Even though he did superior technical work, Peter clearly did not work with his team. Although it is not clear how to grade the rest of the team, Peter should get no better than a fair grade, and I would even consider ranking him as poor.

- Case 2. Joyce should not have been in the class. She had not taken a PSP course and was not a competent programmer. In spite of her lack of qualifications, however, she was made the quality/process manager and worked hard on every task she was given. She took the lead in producing the user documentation, she handled the team's data, and she produced the team's final report. Overall, with Joyce's help, the team produced a working product and presented an outstanding final report.

Joyce did well against two of the three grading criteria and adequately on the third. Even though she did not write any part of the program code, she did produce several parts of the SRS and user documentation and did well on everything else she attempted. Software teams need a range of capabilities, and Joyce could be invaluable, even without programming skills, so I would rank her as good.

- Case 3. Greg worked precisely the hours he had committed, but nothing extra. Although the rest of the team worked several late evenings and weekends, Greg never did. He attended the team meetings but did not meet his other commitments. On one occasion, Greg was supposed to give part of a class presentation but did not prepare and did a poor job. Although the team produced an adequate product Greg delivered one program module of marginal quality, he did not work well with the team, did not adequately follow the process, and produced marginal work. I would give him no better than a fair grade and would even consider ranking him as poor.

<- Chapter 11 11.1 11.2 ->

Contents List of Forms Tool Instructions

11.2. Measurement

Because the PSP and TSPi courses generate a great deal of student data, it is tempting to base students' grades on these data. You should carefully review the students' data, but you must not establish any data-based grading criteria. For example, if you tell the class that a 50% or better yield-before-compile is required to get an excellent grade, everybody is likely to have a 50% or better yield. Although these yield values could be honest, high yields can be achieved in many ways. Smart students will immediately realize that the easiest way to get a high yield is to find large numbers of trivial mistakes in the SRS document. A less honest but equally effective approach would be to stop reporting defects when the yield approaches 50%.

A principal objective of the PSP and TSPi courses is to demonstrate that using a defined and measured process will help engineering teams do better work. If you derive grades from the students' data, you are more likely to teach them that personal measurements are dangerous and should be avoided.

<- 11.1 11.2 11.3 ->

Contents List of Forms Tool Instructions

11.3. Overall Effectiveness

On balance, you need to consider how each student performed as an engineer. Did he or she follow directions, deliver work on time, and do thorough and complete work? When late, did he or she warn you in advance or miss the commitment without any notice? Although it is relatively easy to grade those students who do very little or those who do everything well, it is the ones in the middle who are the problem. To help in assigning fair and consistent grades, it is important to consider all the important aspects of the work.

In evaluating effectiveness, consider assigning weights to the various product and process artifacts the students produce. For example, consider the relative importance of the SRS, SDS, user documentation, test plans, and finished programs. On the process side, assign values for correctly recording time, size, and defects and properly competing the SUMP, SUMQ, SCHEDULE, TASK, and WEEK forms. Similarly, you could give weights to doing reviews, holding inspections, and following the SCM process.

Finally, on the management side, did the engineer and his or her team produce sound plans? Was their progress tracked? Were issues managed? Was the work completed as committed? Note, however, that there can be many reasons for technical work being late, and many are not under the engineer's control. You should not mark the students grades down too severely for being late, as long as they tell you in advance and deal responsibly with any resulting problems.

<- 11.2 11.3 Chapter 12 ->

Contents List of Forms Tool Instructions

Chapter 12
Handling People Problems

This chapter describes some people issues you will likely encounter in team courses. There are many possible people issues. This chapter deals with the following:

The shrinking violet

The drone

The hard head

Removing a team member

Losing a member

Losing multiple team members

Team leader problems

<- 11.3 Chapter 12 12.1 ->

Contents List of Forms Tool Instructions

12.1. The Shrinking Violet

Shrinking violets are generally unsure of their own abilities and are reluctant to take on unfamiliar challenges. Unless they can overcome this problem, however, they should reconsider their decision to be software engineers. This is a stressful enough business without the added burden of insecurity. Shrinking violets lack self-confidence, but they generally also want to be accepted and respected. Such people do not believe they can do jobs before they tackle them, and they are reluctant to try anything new because they are afraid to fail.

Usually, the most successful way to handle these people is to give them an opportunity to excel. Assign them to jobs that push them, and then get their teams to support and assist them. Generally, people are capable of far more than they think, and the only way they will learn to appreciate their own capabilities is to succeed at a challenging assignment.

To get a shrinking violet to accept a challenge, build on his or her need for acceptance and respect. Shrinking violets will generally step up to a challenge when the team asks for help. They will be nervous about the job but flattered to be asked. This could, in fact, be one of the first times anyone has shown enough confidence in them to ask for their assistance.

When team leaders tell you about shrinking violets, help them find tasks these engineers can likely accomplish. Then have the team leader explain the job and that the team needs their help. Pick the job with care but try to pick one that can grow. Then, as the person accomplishes each part, add to it. One example is test planning. Start with the system test plan and then, when this task is done, add build and test planning. Then perhaps extend the job to the user documentation.

Generally, shrinking violets will not cause teamwork problems. They can, however, limit a team's capability . If you can challenge shrinking violets to perform, you can give them the courage to fully support their team, and this would be rewarding for everyone involved.

<- Chapter 12 12.1 12.2 ->

Contents List of Forms Tool Instructions

12.2. The Drone

Drones do as little as they can get by with. The most obvious symptoms are that they don't work as many hours as the rest of the team, they are frequently late for meetings, and they rarely meet their commitments. What is generally most annoying is that they don't feel guilty when they inconvenience everyone else.

These people have been coasting all their lives. They have developed thick skins and have learned to ignore peer pressure. Generally, in fact, the only way to handle drones is to apply substantial pressure.

In industry, drones can generally be handled with a combination of team and management pressure. When peer pressure is coupled with the threat of job termination, almost everyone will respond. But in a university course, such threats will not work. In the absence of a credible threat, however, you probably cannot change this person's habits. It is thus unlikely that student drones will perform, almost regardless of what you or the team does.

You should not, however give up without making an effort. First find out whether you are really dealing with a drone. Ask the team leader to talk to this person and start on the assumption that there is a valid reason for this person's apparent irresponsible behavior. Perhaps he or she has a heavy workload or has a job. There could also be health or family problems.

If the team leader has explored these possibilities and has concluded that this person is a drone, offer to talk to this person yourself. In this conversation, discuss the student's problems and see whether you can find out why he or she is unwilling to fully support the team. Unless there is a valid explanation, point out that if this behavior continues his or her grade will suffer. You will not likely accomplish much, but you must make the effort.

Nobody likes to be taken advantage of, and drones can be a serious annoyance, but they are rarely disabling. Generally, however, even if the team wants to drop this member, it is best not to do so. Suggest that the team essentially ignore the problem and act as if this person is only working part-time or possibly not even on the team. Although the team's energy and enthusiasm will likely suffer, try to get them to concentrate on the job and ignore this annoyance.

<- 12.1 12.2 12.3 ->

Contents List of Forms Tool Instructions

12.3. The Hard Head

Occasionally, you will find someone who is incapable of working with the team. This person may be highly opinionated, insist on taking charge, or have a strong bias against a team member. Sometimes such people feel threatened, are unwilling to compromise, or have an overblown opinion of their own abilities. These people are generally difficult to handle, both because they are very sensitive to even implied criticism and because they are often seriously unhappy with themselves and with everyone around them. In general, one such person can incapacitate a team. Although such people are relatively rare, they do show up, and they can be a problem for everyone involved.

When the team leader brings you such a problem, the first step is to have him or her talk to this person and find out the source of the problem. Hard heads can be very sensitive, however, and any hint that the team leader is being critical or taking a position could stop all communication. Counsel the team leader to focus on teamwork issues and not to get into blame or personality discussions. Just ask for this person's views and focus on the job to be done.

If the team leader cannot establish some degree of rapport with this person, meet with him or her yourself. When you do, focus on listening and trying to understand this person's concerns. Remember, however, that such people are often very bright and will act surprised that you feel there is a problem. They will be very reasonable and agree to work with the team. However, their inability to compromise or to follow anyone else's lead will largely incapacitate the team.

If, in the last analysis, neither you nor the team leader is successful in getting this person to participate as a regular team member, you may have to remove this person from the team.

<- 12.2 12.3 12.4 ->

Contents List of Forms Tool Instructions

12.4. Removing a Team Member

Removing a team member is probably the most sensitive action you can take. If the team is being destroyed, however, you may have to do it. If you can catch such problems early, you could have this person drop the course without a grade. Before taking this action, consider two choices:

1. Removing the person now or delaying removal until the end of the current development cycle

2. Having this person drop the course or making another team assignment

Before making another team assignment, however, realize that this person will likely be a problem for any other team. Also, recognize that this is a course about teamwork, and if some student is incapable of learning the teamwork lessons, that student has failed.

In removing a team member, first discuss the issues individually with every member of the team, including the difficult person. Make it clear that you want to understand the issues and see whether you can help to resolve them. If there does not appear to be a reasonable solution, and if the rest of the team wants this member removed, then discuss the problem with this person. He or she will likely have several complaints and feel wronged. So make sure you really listen to and understand the complete story before making any decision.

In these discussions, consider the following steps.

1. Stay impartial and never give the impression that you are speaking for the team or are critical of this person. Just ask and listen.

2. Spend enough time talking to the troublesome person to thoroughly understand his or her concerns.

3. Talk to all the other team members to ask them what they think should be done.

4. Keep notes on these meetings and make sure that, if everyone wants this person dropped from the team, you have a clear record of who said what and when.

5. Talk to the troublesome person again and suggest that you and the team are concerned about the pressures he or she is facing.

6. See whether you can get this person to be the first to mention resigning from the team. When the subject of resignation comes up, however, point out that the team would be willing to have him or her resign if that would help.

7. Above all, try to get this person to resign voluntarily. If you try to remove this person without prior agreement, you could become involved in a time-consuming and contentious complaint to the university or even to a government agency.

8. Make very detailed notes of all the discussions.

<- 12.3 12.4 12.5 ->

Contents List of Forms Tool Instructions

12.5. Losing a Team Member

Losing a team member is always a traumatic experience both for the team and for the member who left. Thus, you should do what you can to prevent such losses. If they occur, however, help the team overcome the resulting stresses. If the person had to leave for sickness, family, or workload reasons, the problems will not be severe. The principal issue will be the loss of a valued team member.

If, on the other hand, the person resigned because of teamwork problems, the recovery problems will likely be more difficult. The team members will almost certainly feel rejected. Did this person quit because of something they did? Whose fault is it that this happened? Can they pull themselves together to get the job done? In these cases, you should meet with the remaining team members as a group to discuss their concerns. In this meeting, do a lot of listening. Explain that such things often happen and that it is generally no one's fault. Then, after they have thoroughly discussed their concerns, turn the conversation to the work they have ahead of them and how they might handle their increased workload. The principal therapy in these cases is to focus on the work to be done and not to dwell on the past.

<- 12.4 12.5 12.6 ->

Contents List of Forms Tool Instructions

12.6. Losing Multiple Team Members

On occasion, you may have a team leader who cannot handle the job. The textbook has an extensive discussion on the traits and duties of team leaders, and most people can, when thrust into such positions, do a creditable job. However, you will occasionally select an ineffective team leader. Sometimes this person may not be suited to a leadership role, but more often the team will have problems that this leader is not equipped to handle. In general, however, unless the problems are severe, do not change the team-leader role during a development cycle.

<- 12.5 12.6 12.7 ->

Contents List of Forms Tool Instructions

12.7. Team-leader Problems

On occasion, you may have a team leader who cannot handle the job. The textbook has an extensive discussion on the traits and duties of team leaders, and most people can, when thrust into such positions, do a creditable job. However, you will occasionally select an ineffective team leader. Sometimes this person may not be suited to a leadership role, but more often the team will have problems that this leader is not equipped to handle. In general, however, unless the problems are severe, do not change the team-leader role during a development cycle.

A leadership change is traumatic for the deposed leader, for the person taking over the leadership role, and for the rest of the team. Thus, unless the team revolts, do not make a change in the middle of a development cycle. The only exceptions are that the current leader decides that the role selection was a mistake, there is another team member who is willing to take over, and the rest of the team agrees.

If the team revolts, however, you will have to do something. Then you should go through many of the steps discussed earlier for removing a team member. Remember, however, that the team leader may not be a hard head but rather may be having trouble with this assignment.

Generally, when leaders are rejected by their teams, they will not be able to stay on the same team. If they and everyone else on the team agrees, however, just reassign roles and leave the team in place. Otherwise, put the removed team leader on another team or devise some face-saving assignment that he or she can do for the rest of the cycle or semester.

<- 12.6 12.7 Chapter 13 ->

Contents List of Forms Tool Instructions

Chapter 13
The Rewards of Teaching Teams

In many years of software work, I have found that almost everyone I have taught or worked with is remarkably capable. Also, just like other types of engineers, they like to build things. I have also found, however, that software work is often hectic, generally unsuccessful, and largely unrewarding.

It doesn't have to be this way. When teams are properly trained and guided, they can do extraordinary work. In fact, we don't really know the limits of what such teams can do. What is more, neither do they. Generally, however, software people are capable of doing far more than they or anyone else believes they can do. And finding out what they can do is intensely rewarding, both for them and for those of us who are fortunate enough to coach and support them.

What is most rewarding is to help teams to behave professionally. When teams know how to plan a job and when they take the time to produce a thoughtful plan, their plans are remarkably good. Then, when they work to these plans, these teams invariably meet their commitments while producing quality products. But what teams are most surprised to discover is that while they are developing plans, they also develop the conviction to defend the plans. And when they defend their plans to management, management invariably accepts them.

Finally, when engineers work to rational plans, they are remarkably creative and productive. What is more, when they work on capable teams, they enjoy the work and they produce extraordinary products. What I have found most rewarding is teaching these remarkable people how to take charge of their work and how to enjoy what they do.

<- 12.7 Chapter 13 PSP Familiarization Exercises ->

Contents List of Forms Tool Instructions

Appendix A
PSP Familiarization Exercises

This appendix contains five PSP familiarization exercises. These exercises are designed to familiarize students with various aspects of the PSP but not to replace a PSP course. Thus, if students have not taken a PSP course, merely completing these exercises will not provide the background needed for the TSPi course. The exercises are for students who took a PSP course several semesters earlier and have not used it since. The final two exercises will also be useful for students who have just completed a PSP course. The reason is that, although the PSP teaches about earned value, the students do not generally use it.

The five exercises in this section are reviews of topics covered in the textbook Introduction to the Personal Software Process [Humphrey 1997]. The review topics and the chapters in this introductory PSP text that cover them are as follows:

Tracking time (Chapter 3)

Defects, code reviews, and checklists (Chapters 12, 13, and 14)

Program size (Chapter 6)

Task and schedule planning (Chapter 9)

Estimating project completion (Chapter 9)

The recommended way to use each of these exercises is to first describe the material in class, then have the students complete the exercise, and, finally, hand out the exercise answer and discuss it. In particular, make sure that students who got an incorrect answer know what they did wrong and how to do it correctly. The exercise answers are given in Appendix B.

<- Chapter 13 PSP Familiarlzation Exercises Exercise #1: Tracking Time ->

Contents List of Forms Tool Instructions

PSP Familiarization Exercise 1: Tracking Time

Purpose

The objective of this exercise is to show you how to track your time using the LOGT form.

Contents

This exercise package contains an exercise scenario, a partially completed time recording log (LOGT) form, and the LOGT form instructions.

Directions

Using the data given in the following scenario, complete the attached LOGT form. The first day of the scenario has already been completed, and you should complete days 2 and 3. After you are finished, the instructor will hand out the exercise answer and check to see whether you got the correct result. If you made mistakes, make sure you understand the problems and then complete day 4 of the exercise.

Exercise Scenario

Day 1: Joyce started planning her work on the Calc program module on Monday, June 6, at 8:37 AM (Plan). She finished the plan at 9:26 AM and immediately started on the detailed design (Design). She worked for nearly an hour, took a 30-minute break, and then continued until lunchtime, at 11:45. After lunch, Joyce got back to work at 12:48 and worked until 1:31, when she finished the design. She then did her e-mail and checked on a planned trip until she started on the detailed design review at 2:18 (DLDR). Some time later, she got a phone call and stopped for a break. After this 27-minute interruption, she continued with the design review until 4:56, when she quit for the day. Before leaving, however, she asked Pete and Craig to help her with a design inspection the next morning at 10:00 AM.

Day 2: On Tuesday at 8:15, Joyce started working on the unit test plan (Test plan). A little later, Pete stopped in to ask whether they could delay the inspection until 1:30. They checked with Craig, who agreed to the new time. After this 21-minute interruption, she continued with test planning until 9:39, when she started on unit test development (Test dev). This took until 12:46, with three interruptions totaling 28 minutes. After a hurried lunch, Joyce got her notes and started the inspection briefing at 1:30 (DLDI). This meeting took only until 1:57. After the briefing, Pete and Craig reviewed the program while Joyce did other work until 3:50. She then met with them in the design inspection meeting until 5:28 (DLDI).

Day 3: The next morning, Joyce took from 8:38 to 9:56 to fix the problems found in the inspection (DLDI). In this time, she got one phone call that took 17 minutes. After Joyce did her e-mail and read a report, she started to write the program source code at 11:18 (Code). During coding, she was interrupted

for 23 minutes by an engineer who needed to ask her some questions. Joyce then continued coding until 12:06 when she quit for lunch. After lunch, she put a Do-Not-Disturb sign on her door and forwarded her phone to the reception center. Joyce then worked from 1:11 until 2:16, when she completed coding. After a break, she started on the code review at 2:26, and worked until 3:54 (CDR). She then called Craig and Pete to see how soon they could do a code inspection. Before starting on another task, Joyce sent them copies of her source code and asked them to review it before they met first thing in the morning for the inspection.

Day 4: On Thursday at 7:45, Joyce, Craig, and Pete began the code inspection and worked until 9:11 (Code I). She then took a break and worked on fixing the inspection defects from 9:19 until 9:43 (Code I). At 9:43, Joyce started compiling her program. She had to repair only two defects, so she was finished at 9:55 (Comp). She was so pleased with the way compiling went that she decided to start immediately on unit testing (UT). She ran into a design problem that took more than half an hour to fix, and she found and fixed one other problem in less than a minute. She completed unit testing at 11:39. In this time she had two interruptions of 11 and 26 minutes. After taking a break for lunch, Joyce worked from 12:41 to 1:28 to complete the postmortem on her work (PM).

[<- PSP Familiarlzation Exercises]

Exercise 1: Tracking Time

[Exercise 1: Time Recording Log: Form LOGT ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Time Recording Log: Form LOGT

Name	Joyce	Date	6/6
Team	Team 6	Instructor	WSH
Part/Level	Calc/Module	Instructor	1

Date	Start	Stop	Interruption Time	Delta Time	Phase / Task	Comp-onent	Comments
Day 1							
6/6	8:37	9:26		49	Plan	Calc	Implementation
	9:26	11:45	30	109	Design	"	"
	12:48	1:31		43	Design	"	"
	2:18	4:56	27	131	DLDR	"	"

<- Exercise 1: Tracking Time

Exercise 1: Time Recording Log: Form LOGT

Exercise 1: Time Recording Log Instructions: Form LOGT ->

Contents List of Forms Tool Instructions

TSPi Time Recording Log Instructions: Form LOGT

Purpose	• Use this form to record the time spent on each project task General
General	• Either keep one log and note the task and product element for each entry or keep separate logs for each major task. • Record all the time you spend on the project. • Record the time in minutes. • Be as accurate as possible. • If you forget to record the starting, stopping, or interruption time for a task, promptly enter your best estimate.
Header	• Enter your name, date, team name, and instructor's name. • Name the part or assembly and its level. • Enter the cycle number.
Date	• Enter the date when you made the entry. • For example, 10/18/99.
Start	• Enter the time when you start working on a task. • For example, 8:20.
Stop	• Enter the time when you stop working on that task. • For example, 10:56.
Interruption Time	• Record any interruption time that was not spent on the task and the reason for the interruption. • If you have several interruptions, enter their total time. • For example, 37‹took a break
Delta Time	• Enter the clock time you actually spent working on the task, less the interruption time. • For example, from 8:20 to 10:56 less 37 minutes is 119 minutes.
Phase/Task	• Enter the name or other designation of the phase or task you worked on. • For example, planning, code, test and so on.
Component	• If the task was for a unique component, enter the name of the component.
Comments	• Enter any other pertinent comments that might later remind you of any unusual circumstances regarding this activity. • For example, had a requirements question and needed help

<- Exercise 1: Time Recording Log: Form LOGT

Exercise 1: Time Recording Log Instructions: Form LOGT

Exercise 2: Defects, Code Reviews, and Checklists ->

Contents List of Forms Tool Instructions

PSP Familiarization Exercise 2: Defects, Code Reviews, and Checklists

Purpose

This exercise shows you how to conduct an effective code review.

Contents

This exercise package contains the PSP defect type standard, a blank LOGD form, the LOGD form instructions, a code-review script, a Pascal code-review checklist, and the source code for a small Pascal program.

Directions

Follow the code-review script and use the code-review checklist to find as many of the program's defects as you can. When you find a defect, clearly mark it on the program listing and record it on the LOGD form, together with a brief defect description that includes the defect line number.

Follow the code-review script and use the code-review checklist to find as many of the program's defects as you can. When you find a defect, clearly mark it on the program listing and record it on the LOGD form, together with a brief defect description that includes the defect line number.

After you have completed the code review and turned in your LOGD form, the instructor will hand out the exercise answer. He or she will then review the program one line at a time and discuss each defect. At the end, when you know how many defects were in the program, calculate your personal review yield.

[<- Exercise 1: Time Recording Log Instructions: Form LOGT]

Exercise 2: Defects, Code Reviews, and Checklists

[Exercise 2: Defect Types -Standard DEFECT ->]

[Contents] [List of Forms] [Tool Instructions]

The TSPi Defect Types - Standard DEFECT

Type Number	Type name	Description
10	Documentation	Comments, messages
20	Syntax	Spelling, punctuation, typos, instruction formats
30	Build, package	Change management, library, version control
40	Assignment	Declaration, duplicate names, scope, limits
50	Interface	Procedure calls and references, I/O, user formats
60	Checking	Error messages, inadequate checks
70	Data	Structure, content
80	Function	Logic, pointers, loops, recursion, computation, function defects
90	System	Configuration, timing, memory
100	Environment	Design, compile, test, or other support system problems

<- Exercise 2: Defects, Code Reviews, and Checklists

Exercise 2: Defect Types - Standard DEFECT

Exercise 2: Defect Recording Log: Form LOGD ->

Contents List of Forms Tool Instructions

Defect Types	
10 Documentation	60 Checking
20 Syntax	70 Data
30 Build, Package	80 Function
40 Assignment	90 System
50 Interface	100 Interface

TSPi Defect Recording Log: Form LOGD

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: ______________________________

____

____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: ______________________________

____

____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: ______________________________

____

____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: ______________________________

____

____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: ______________________________

____

____

<- Exercise 2: Defect Types - Standard DEFECT

Exercise 2: Defect Recording Log: Form LOGD

Exercise 2: Defect Recording Log Instructions: Form LOGD ->

Contents List of Forms Tool Instructions

TSPi Defect Recording Log Instructions: Form LOGD

Purpose	• Use this form to hold data on the defects you find and correct.
General	• Keep a separate log for each program component. • Record each defect separately and completely. • If you need additional space, use another copy of the form.
Header	• Enter your name, date, team name, and instructor's name. • Name the part or assembly and its level. • Enter the cycle number.
Date	• Enter the date you found the defect.
Number	• Enter the defect number. • For each program component (or module), use a sequential number starting with 1 (or 001 and so on).
Type	• Enter the defect type from the defect type list summarized in the top left corner of the log form. • See Appendix G in the textbook for the defect type specification. • Use your best judgment in selecting which type applies.
Inject	• Enter the phase when this defect was injected. • Use your best judgment.
Remove	• Enter the phase during which you fixed the defect. • This is generally the phase when you found the defect.
Fix Time	• Enter the time you took to fix the defect. • This time can be determined by stopwatch or by judgment.
Fix Defect	• If you or someone else injected this defect while fixing another defect, record the number of the improperly fixed defect. • If you cannot identify the defect number, enter an X in the Fix Defect box.
Description	• Write a succinct description of the defect that is clear enough to later remind you about the error and help you to remember why you made it. • If the defect was injected in a prior cycle, give the cycle number and put an * by the phase injected (such as code* or design*).

<- Exercise 2: Defect Recording Log: Form LOGD

Exercise 2: Defect Recording Log Instructions: Form LOGD

Exercise 2: Code Review Script ->

Contents List of Forms Tool Instructions

Code Review Script

Purpose		• To assist engineers in producing quality software products • To help them conduct efficient and high-yield code reviews
Entry criteria		• The program requirements and design • The program source code • The coding standards • The code review checklist • A defect recording log (LOGD) form
General		• Do the code review with a source code listing, never on the screen. • In following the checklist, review the entire program completely for one checklist item and then review it again for the next item. • Continue reviewing in this way until all checklist items are done. • When finished, do one general review of the program for any problems that might not have been covered by the checklist. • During the review, note every defect on the program source code listing and record the defect on the LOGD form • After completing the code review, correct all the defects. • After correction, produce a new program listing and review all the corrections to ensure that they are correct. • If there were many corrections, review the complete program again.
Step	Activities	Description
1	Completeness	• Check the program against the requirements specification (SRS) to ensure that it includes all required functions. • Check the program against the design specification (SDS) to ensure that all required program functions are implemented.
2	Checklist	• Follow the code review checklist to find all the defects in the program. • In doing the review, mark the defects on the source code listing and record them on the defect recording form (LOGD).
3	Corrections	• Following the code review, correct all the defects.
4	Rereview	• After completing the corrections, produce a source program listing for the corrected program. • Review all the corrections to ensure that they are correct.
5	Final check	• Correct any remaining defects. • Check these corrections for possible mistakes and correct and rereview any defects that are found.
Exit Criteria		• The program has been completely reviewed. • All the defects found have been corrected. • The corrections have been checked for correctness. • All defects are recorded on the defect recording log (LOGD). • Total review time is recorded on the LOGT form.

<- Exercise 2: Defect Recording Log Instructions: Form LOGD

Exercise 2: Code Review Script

Exercise 2: Code Review Checklist: Pascal ->

Contents List of Forms Tool Instructions

Code Review Checklist: Pascal

Purpose	To guide engineers in conducting effective code reviews.	
General	As you complete each review step, note the number of defects found in the box to the right. If none, put a check in the box. Complete the checklist for one program, function, or procedure before starting to review another.	
Step	Description	Defects
Complete	Verify that all functions in the design are coded and that all necessary functions and procedures have been implemented.	
Includes	Check that all needed with statements are properly defined and used.	
Logic	Verify that the program flow and all procedure and function logic is consistent with the detailed design.	
Loops	● Do a hand simulation of all loops and recursive procedures that were not simulated in the detailed design review or inspection. ● Ensure that every loop is properly initiated and terminated. ● Check that every loop is executed the correct number of times.	
Calls	Check every function and procedure call to ensure that it exactly matches the definition for formats and types.	
Declarations	Verify that each variable and parameter ● Has exactly one declaration ● Is used only within its declared scope ● Is spelled correctly wherever used	
Initializations	Check that every variable is initialized.	
Limits	Check all variables, arrays, and indexes to ensure that their use does not exceed declared limits.	
Begin-end	Check all begin-end pairs, including cases where nested ifs could be misinterpreted.	
Boolean	Check Boolean conditions and correct use of = and :=.	
Format	Check every program line for instruction format, spelling, and punctuation.	
Pointers	Check that all pointers properly use new, dispose, and nil and are properly referenced.	
Strings	Verify that all characters and strings are bracketed by ' '.	
Input-output	Double check all input-output formats.	
Spelling	Check that every variable, parameter, and key work is properly spelled.	
comments	● Ensure that all commenting is accurate and according to standard. ● Verify that every comment is enclosed in { and } and does not contain a }.	

<- Exercise 2: Code Review Script

Exercise 2: Code Review Checklist: Pascal

Exercise 2: Pascal Source Program ->

Contents List of Forms Tool Instructions

Pascal Source Program

```
1  {Spelling procedure; WSH; 06/29/91}
2  {Purpose: This procedure converts positive or zero integers to strings.}
3  {Usage instructions:
4  Call: SPELLING (YourNumber: integer; YourResult: string; YourError: 0)
5  Return: YourNumber: unchanged
6          If YourNumber negative, YourResult: blank character string
7                  YourError = 1
8          If YourNumber >0 or 0, YourResult: string of characters for the number
9                  YourError = 0}
10  {Implementation for the spelling procedure.}
11  PROCEDURE SPELLING(var InputInteger: integer; var SomeString: string;
12                  var SomeError: integer);
13   var i, {index}
14      Length, {maximum number length}
15      SomeInteger, {a trial number}
16      N, {tens variable}
17      Offset: integer; {offset value to convert digits to ASCII characters}
18      Flag: boolean; {to show leading 0s}
19      SpelledNumber : string; {trial string, becomes output}
20   begin
21      YourError = 0;
22      SomeInteger := InputInteger;
23      SomeString := '';
24      Test := 0;
25      SpelledNumber := '';
26      >N := 1000;
27      Length := 6;
28      Offset := 48;
29      i := 1;
30      Flag := false;
31      {Next, cycle through the integer from highest-order to lowest-order digits to
32      determine digit's value and add its character to SpelledNumber.}
33      if SomeInteger < 0 {Does not work for negative integers.}
34        then
35          SomeError := 1
36        else
37          repeat {until i := Length}
38            if SomeInteger > N {Is first digit >= 1?}
39              then
40                begen
41                  Test := SomeInteger;
42                  Test := Test div N; {Find the leading digit value.}
```

```
43              SomeInteger := SomeInteger mod(Test*N); {Eliminate the
44                   leading digit.}
45              Test := Test + Offset; {Set digit to its ASCII value.}
46              SpelledNumber := SpelledNumber + chr(Test); {Add the
47                   character to end of SpelledNumber.}
48            end
49           else {Check for zero characters.}
50            if Flag
51              then {If an internal character was 0, insert '0'.}
52                SpelledNumber := SpelledNumber + '0'
53              else {If 0 or no character in leading positions, leave blank.}
54                SpelledNumber := SpelledNumber + ' ';
55        i := i + 1; {Step to the next digit.}
56        N := N div 10
57      until i = Length; {Continue until all digits tested}
58      SomeString := SpelledNumber; {Set output to resulting string}
59  end; {PROCEDURE SPELLING }
```

[<- Exercise 2: Code Review Checklist: Pascal]

Exercise 2: Pascal Source Program

[Exercise 3: Program Size ->]

[Contents] [List of Forms] [Tool Instructions]

PSP Familiarization Exercise 3: Program Size

Purpose

This exercise shows you how to count lines of code.

Contents

The exercise package contains a line-of-code counting standard and the source code for a small Pascal program.

Directions

First, read the line-of-code counting standard and mark every line that should be counted on the listing. Then count and number the lines you have marked. When you are finished, the instructor will hand out the exercise answer and check to see whether everyone got the correct answer. Ask about any differences and make sure you know how to count lines of code.

Instructor's Note:

If the students do not know the Pascal language, you will have to provide a LOC counting standard and sample program listing in a language they know.

Line-of-Code Counting Standard: Pascal

Purpose	To guide engineers in determining the size of Pascal source programs.
General	● Line-of-code (LOC) counting standards are arbitrary and language-dependent. ● For consistent size measurement, all software team members should use a single LOC counting standard.
Item	Description
Parentheses	No statement or punctuation marks within '(Š)' are counted.
Brackets	No statement or punctuation marks within '{ Š }' are counted.
Semicolon	Every occurrence of a ';' is counted once.
Period	Every occurrence of a '.' that follows a terminating END statement is counted once.
Key Words	Every occurrence of the following selected key words is counted once: BEGIN, CASE, DO, ELSE, END, IF, RECORD, REPEAT, THEN
Special	Where there is no ';' at a line end, every statement preceding the following key words is counted once (if not already counted): ELSE, END, UNTIL
Comma	Every occurrence of a ',' in the USES or VAR portions of the program is counted once.

[<- Exercise 2: Pascal Source Program]

Exercise 3: Program Size

[Exercise 3: Pascal Source Program ->]

[Contents] [List of Forms] [Tool Instructions]

Pascal Source Program

```
1  {Spelling procedure; WSH; 06/29/91}
2  {Purpose: This procedure converts positive or zero integers to strings.}
3  {Usage instructions:
4  Call: SPELLING (YourNumber: integer; YourResult: string; YourError: 0)
5  Return: YourNumber: unchanged
6          If YourNumber negative, YourResult: blank character string
7                  YourError = 1
8          If YourNumber >0 or 0, YourResult: string of characters for the number
9                  YourError = 0}
10  {Implementation for the spelling procedure.}
11  PROCEDURE SPELLING(var InputInteger: integer; var SomeString: string;
12                  var SomeError: integer);
13   var i, {index}
14      Length, {maximum number length}
15      Test, {a trial number}
16      SomeInteger, {a trial number}
17      N, {tens variable}
18      Offset: integer; {offset value to convert digits to ASCII characters}
19      Flag: boolean; {to show leading 0s}
20      SpelledNumber : string; {trial string, becomes output}
21   begin
22      YourError = 0;
23      SomeInteger := InputInteger;
24      SomeString := '';
25      Test := 0;
26      SpelledNumber := '';
27      N := 10000;
28      Length := 6;
29      Offset := 48;
30      i := 1;
31      Flag := false;
32      {Next, cycle through the integer from highest-order to lowest-order digits to
33      determine digit's value and add its character to SpelledNumber.}
34      if SomeInteger < 0 {Does not work for negative integers.}
35        then
36          SomeError := 1
37        else
38          repeat {until i := Length}
39            if SomeInteger > N {Is first digit >= 1?}
40              then
41                begin
42                  Test := SomeInteger;
```

```
43              Test := Test div N; {Find the leading digit value.}
44              SomeInteger := SomeInteger mod(Test*N); {Eliminate the
45                   leading digit.}
46              Test := Test + Offset; {Set digit to its ASCII value.}
47              SpelledNumber := SpelledNumber + chr(Test); {Add the
48                   character to end of SpelledNumber.}
49              Flag := true; {Set flag to take care of non-leading zeros.}
50            end
51          else {Check for zero characters.}
52            if Flag
53              then {If an internal character was 0, insert '0'.}
54                SpelledNumber := SpelledNumber + '0'
55              else {If 0 or no character in leading positions, leave blank.}
56                SpelledNumber := SpelledNumber + ' ';
57        i := i + 1; {Step to the next digit.}
58        N := N div 10
59      until i = Length; {Continue until all digits tested}
60      SomeString := SpelledNumber; {Set output to resulting string}
61  end; {PROCEDURE SPELLING }
```

<- Exercise 3: Program Size

Exercise 3: Pascal Source Program

Exercise 4: Task and Schedule Planning ->

Contents List of Forms Tool Instructions

PSP Familiarization Exercise 4: Task and Schedule Planning

Purpose

In this exercise, you learn how to calculate planned value and how to determine the dates when you will complete each task.

Contents

The exercise package contains partially completed TASK and SCHEDULE templates together with their instructions.

Directions

Complete the following TASK and SCHEDULE templates, giving the planned value for each task, the week each task is expected to be completed, and the total planned value anticipated for each week.

Instructor's Note:

Because many students may not be familiar with the reasons and methods for estimating and tracking earned value, start with a brief lecture on the subject. Then discuss the steps required to complete this exercise.

Because this exercise is designed to illustrate the planned-value calculations, only part of the TSPi TASK form is used.

[<- Exercise 3: Pascal Source Program]

Exercise 4: Task and Schedule Planning

[Exercise 4: Example Task Planning Template: Form TASK ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Example Task Planning Template: Form TASK

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Line-of-Code Counting Standard: Pascal

Task		Plan Hours		Plan Size/Value			Actual		
Phase	Task Name	Total Team Hours	Cummulative Hours	Week No.	Planned Value	Cumulative PV	Hours	Cumulative Hours	Week No.
Strategy	Produce team strategy	7.00	7.00						
Plan	Overall planning	7.75	14.75						
Plan	Detailed plan	7.50	22.25						
Requ.	Need statement review	3.75	26.00						
Requ.	Produce SRS	4.50	30.50						
Requ.	Inspect SRS	5.00	35.50						
Requ.	Produce design standards	2.00	37.50						
Design	Produce SDS	6.25	43.75						
Design	SDS inspection	6.25	50.00						
Design	Coding standards	2.00	52.00						
Imp.	Implementation planning	3.25	55.25						
DLD	Detailed design	15.00	70.25						
DLDR	Detailed design review	10.00	80.25						
DLDI	Detailed design inspection	18.00	98.25						
Code	Code	5.00	103.25						
CDR	Code review	5.00	108.25						
Comp	Compile	2.00	110.25						
CDI	Code inspection	7.50	117.75						

Unit test	Unit test	7.50	125.25						
Imp.	Quality review	4.00	129.25						
TD	Test plan and development	5.50	134.75						
B&I	Build and integration	1.50	136.25						
ST	System test	18.00	154.25						
Doc	Documentation	2.50	156.75						
PM	Postmortem	4.00	160.75						
		160.75							

<- Exercise 4: Task and Schedule Planning

Exercise 4: Example Task Planning Template: Form TASK

Exercise 4: Task Planning Template Instructions: Form TASK ->

Contents List of Forms Tool Instructions

TSPi Task Planning Template Instructions: Form TASK

Purpose	● TSPi Task Planning Template Instructions: Form TASK ● To compute the planned value for each project task ● To estimate the planned completion date for each task ● To provide a basis for tracking schedule progress even when the tasks are not completed in the planned order
General	● Expand this form or use multiple pages as needed. ● Expand this form or use multiple pages as needed. ● Use task names and numbers that support the activity and are consistent with the project work breakdown structure. ● Where possible, use the TSPi tool for planning.
Produce the Task Plan	Use the TSPi tool to produce the task plan. ● Start with the list of tasks given by the TSPi process. ● If you feel additional tasks are needed, insert them ● Estimate the sizes of the products involved with each task (use data from form SUMS where appropriate). ● Estimate the engineering hours for each task. ● Enter these data as described in the rest of this instruction. If you do not have the TSPi tool, you must make the calculations yourself, as described below and in Chapter 5.
Update the Task Plan	To update the task plan during and after development ● Enter the task times from the TSPi time logs (LOGT). ● Record the week number when you finish a task. ● The TSPi tool will complete the calculations for you. If you do not have the TSPi tool, you must make the calculations yourself, as described below and in Chapter 5.
Header	● Enter your name, date, team name, and instructor's name. ● Name the part or assembly and its level. ● Enter the cycle number.
Phase	Enter the TSPi process phase in which the task will be done. ● For example, requirements, code inspection, compile, etc.
Part	Enter the name of the part with which the task is associated. ● For a requirements inspection, the part would be SRS. ● For coding, the part would be the program or module.
Task Name	Enter a task number and name. ● List the tasks in the order in which you expect to complete them. Select tasks that have explicit completion criteria. ● For example, planning completed, program compiled and all defects corrected, testing completed and all test cases run, etc.
#Engineers	Enter the number of engineers who will participate in this task.

Plan-Hours	Enter the planned hours for each task. ● Enter the hours planned for each engineer. ● Enter the total hours for all engineers on the team.
Plan-Cumulative Hours	● Enter the cumulative sum of the plan hours down through each task.
Size Units	Enter the size unit. For example, ● For a program the size unit is typically LOC. ● For the SRS, the size unit is typically pages.
Size	Enter the size of the product involved in that task.
Week No.	● For each cumulative hours entry, find the plan cumulative hours entry on the SCHEDULE form that equals or just exceeds it. ● For each task, calculate the percent its planned hours are of the total planned hours. ● If several weeks on the SCHEDULE form have the same cumulative value, enter the earliest date. ● Pick the plan date as the Monday of the week during which completion for that task is planned.
Planned Value	● Total the planned hours for all the tasks. ● Enter the week number from the row of the SCHEDULE form as the plan date on the TASK form. ● Enter this percent as the planned value for that task. ● The total planned value should equal 100.
Cumulative Planned Value	● Enter the cumulative sum of the planned values down through each task. ● Complete the SCHEDULE form down through Plan-Cumulative Hours before proceeding. ● Then complete the SCHEDULE and TASK forms together.
Actual Hours	As each task is completed, enter the total actual hours expended by the team on that task.
Cumulative Hours	As each task is completed, total the hours for all completed tasks to date.
Week No.	Enter the week number when the task was completed.
Earned Value	When a task is completed, enter the following data in your personal form WEEK for that week: ● The task name and its planned and actual hours ● The task earned value (from the planned value column for That task in form TASK) ● The planned week number for the task Also enter the earned value for all tasks completed in the week on form SCHEDULE.

[<- Exercise 4: Example Task Planning Template: Form TASK]

Exercise 4: Task Planning Template Instructions: Form TASK

[Exercise 4: Schedule Planning Template: Form SCHEDULE ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Schedule Planning Template: Form SCHEDULE

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

		Plan			Actual			
Week No.	Date	Direct Hours	Cumulative Hours	Cumulative Planned Value	Team Hours	Cumulative Hours	Week Earned Value	Cumulative Earned Value
1		20	20					
2		20	40					
3		20	60					
4		20	80					
5		20	100					
6		20	120					
7		20	140					
8		20	161					

<- Exercise 4: Task Planning Template Instructions: Form TASK

Exercise 4: Schedule Planning Template: Form SCHEDULE

Exercise 4: Schedule Planning Template Instructions: Form SCHEDULE ->

Contents List of Forms Tool Instructions

TSPi Schedule Planning Template Instructions: Form SCHEDULE

Purpose	● To record estimated and actual hours expended by week ● To show the cumulative planned value by week ● To track earned value versus planned value as tasks are completed
General	● Expand this template or use multiple pages as needed. ● Complete in conjunction with the TASK form. ● Where possible, use the TSPi support tool for planning. ● If you use the TSPi support tool, it will complete all the calculations for the TASK and SCHEDULE forms. ● If not, you will have to do the calculations yourself.
Header	● Enter your name, date, team name, and instructor's name. ● Name the part or assembly and its level. ● Enter the cycle number.
Week No.	● From the cycle start, enter a week number, starting with 1.
Date	● Enter the calendar date for each week. ● Pick a standard day in the week, for example, Monday.
Plan - Direct Hours	● Enter the number of hours you plan to work each week. ● Consider nonwork time such as vacations, holidays, etc. ● Consider other committed activities such as classes, meetings, and other projects.
Plan - Cumulative Hours	● Enter the cumulative planned hours through each week.
Plan - Cumulative Planned Value	For each week, take the plan cumulative hours from the SCHEDULE form and ● On the TASK form, find the task with nearest equal or lower plan cumulative hours and note its cumulative PV ● Enter this cumulative PV in the SCHEDULE form for that week ● If the cumulative value for the prior week still applies, enter it again
Actual	During development, enter the actual hours, cumulative hours, and cumulative earned value for each week. ● To determine status against plan, compare the cumulative planned value with the actual cumulative earned value. ● Also compare cumulative planned hours with cumulative actual hours. ● If you are behind schedule and actual hours are below the plan, you are not spending enough time. ● If you are behind schedule and actual hours are equal to or above the plan, the problem is poor planning.

<- Exercise 4: Schedule Planning Template: Form SCHEDULE

Exercise 4: Schedule Planning Template Instructions: Form SCHEDULE

Exercise 5: Estimating Project Completion ->

Contents List of Forms Tool Instructions

PSP Familiarization Exercise 5: Estimating Project Completion

Purpose

In this exercise, you learn how to calculate earned value and how to estimate the likely date when you will finish the project.

Contents

This exercise package contains partially completed TASK and SCHEDULE templates.

Directions

The following TASK and SCHEDULE templates show the planned tasks for a project together with the estimated times, earned values, and completion week for each task. These forms also show the actual task completion week and actual hours spent for the first two weeks of the project. With these data, calculate the earned value (EV) for each week so far on the project. Then, based on the earned value rate to date, estimate the week number when you will complete the project and the projected EV for each week until you are finished. The instructor will then hand out the exercise answer and review it with the class. If you got any incorrect results, find out the nature of your mistakes and make sure you know how to get the correct answers.

Instructor's Note:

Because many students may not know how to use earned value to estimate project completion, review the earned-value method before handing out the exercise. The topics to discuss are

- How to calculate the earned value for the tasks completed to date
- How to determine the earned value for each project week
- How to calculate the likely earned value to be gained in each future week
- How to estimate the week number when the project will be completed

<- Exercise 4: Schedule Planning Template Instructions: Form SCHEDULE

Exercise 5: Estimating Project Completion

Exercise 5: Schedule Planning Template: Form SCHEDULE ->

Contents List of Forms Tool Instructions

TSPi Example Task Planning Template: Form TASK

TSPi Example Task Planning Template: Form TASK

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Task		Plan Hours		Plan Size/Value			Actual		
Phase	Task Name	Total Team Hours	Cumulative Hours	Week No.	Planned Value	Cumulative PV	Hours	Cumulative Hours	Week No.
Strategy	Produce team strategy	7.00	7.00	1	4.35	4.35	3.8	3.8	1
Plan	Overall planning	7.75	14.75	1	4.82	9.18	9.6	13.4	1
Plan	Detailed plan	7.50	22.25	2	4.67	13.84	11.2	24.6	2
Requ.	Need statement review	3.75	26.00	2	2.33	16.17	5.3	29.9	2
Requ.	Produce SRS	4.50	30.50	2	2.8	18.97	6.1	36.0	2
Requ.	Inspect SRS	5.00	35.50	2	3.11	22.08			
Requ.	Produce design standards	2.00	37.50	2	1.24	23.33	1.4	37.4	2
Design	Produce SDS	6.25	43.75	3	3.89	27.22			
Design	SDS inspection	6.25	50.00	3	3.89	31.10			
Design	Coding standards	2.00	52.00	3	1.24	32.35			
Imp.	Implementation planning	3.25	55.25	3	2.02	34.37			
DLD	Detailed design	15.00	70.25	4	9.33	43.70			
DLDR	Detailed design review	10.00	80.25	5	6.22	49.92			
DLDI	Detailed design inspection	18.00	98.25	5	11.20	61.12			

Code	Code	5.00	103.25	6	3.11	64.23			
CDR	Code review	5.00	108.25	6	3.11	67.34			
Comp	Compile	2.00	110.25	6	1.24	68.58			
CDI	Code inspection	7.50	117.75	6	4.67	73.25			
Unit test	Unit test	7.50	125.25	7	4.67	77.92			
Imp.	Quality review	4.00	129.25	7	2.49	80.40			
TD	Test plan and development	5.50	134.75	7	3.42	83.83			
B&I	Build and integration	1.50	136.25	7	0.93	84.76			
ST	System test	18.00	154.25	8	11.20	95.96			
Doc	Documentation	2.50	156.75	8	1.56	97.51			
PM	Postmortem	4.00	160.75	8	2.49	100.00			
		160.75			100.0				

<- Exercise 5: Estimating Project Completion

Exercise 5: Schedule Planning Template: Form SCHEDULE

Exercise 5: Schedule Planning Template Instructions: Form SCHEDULE ->

Contents List of Forms Tool Instructions

TSPi Schedule Planning Template: Form SCHEDULE

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Line-of-Code Counting Standard: Pascal

		Plan			Actual			
Week No.	Date	Direct Hours	Cumulative Hours	Cumulative Planned Value	Team Hours	Cumulative Hours	Week Earned Value	Cumulative Earned Value
1		20	20	19.18	19.2	19.2		
2		20	40	23.33	22.6	41.8		
3		20	60	34.37				
4		20	80	43.70				
5		20	100	61.12				
6		20	120	73.25				
7		20	140	84.76				
8		21	161	100.00				

<- Exercise 5: Schedule Planning Template: Form SCHEDULE

Exercise 5: Schedule Planning Template Instructions: Form SCHEDULE

Exercise 1 Answer: Tracking Time ->

Contents List of Forms Tool Instructions

TSPi Time Recording Log: Form LOGT

Name	Joyce	Date	6/6
Team	Team 6	Instructor	WSH
Part/Level	Calc/Module	Instructor	1

Date	Start	Stop	Interruption Time	Delta Time	Phase / Task	Comp-onent	Comments
Day 1							
6/6	8:37	9:26		49	Plan	Calc	Implementation
	9:26	11:45	30	109	Design	"	"
	12:48	1:31		43	Design	"	"
	2:18	4:56	27	131	DLDR	"	"
Day 2							
6/7	8:15	9:39	21	63	Test Plan	Clac	Implementation
	9:39	12:46	28	159	Test Dev	"	"
	1:30	1:57		27	DLDI	"	"
	3:50	5:28		98	DLDI	"	"
Day 3							
6/8	8:38	9:56	17	61	DLDI	"	Implementation
	11:18	12:06	23	25	Code	"	"
	1:11	2:16		65	Code	"	"
	2:26	3:54		88	CDR	"	"
Day 4							
6/9	7:45	9:11		86	Code I	"	Implementation

	9:19	9:43		24	Code I	"	"
	9:43	9:55		12	Comp	"	"
	9:55	11:39	37	67	UT	"	"
	12:41	1:28		47	PM	"	"

<- Exercise 5: Schedule Planning Template Instructions: Form SCHEDULE

Exercise 1 Answer: Tracking Time

Exercise 2 Answer: Defects, Code Reviews, and Checklists ->

Contents List of Forms Tool Instructions

PSP Familiarization Exercise 2 Answer: Defects, Code Reviews, and Checklists

The corrected Pascal source program is shown next. Each program line that has a defect is marked with an * and enclosed in braces. The corrected line then follows, also marked with an *. Where a line was missing from the program, the correct line is inserted, and the line number is noted with an *. There are a total of eight seeded defects in this program, as shown. You should typically have found four or more of these defects in the review. Six defects would have reflected a good review, and seven or eight would be outstanding. Note that because Pascal does not permit a } within a comment, the erroneous closing brace in line 4 is indicated with an *.

Pascal Source Program

```
 1  {Spelling procedure; WSH; 06/29/91}
 2  {Purpose: This procedure converts positive or zero integers to strings.}
 3  {Usage instructions:
 4*  Call: SPELLING (YourNumber: integer; YourResult: string; YourError: 0*
 4*  Call: SPELLING (YourNumber: integer; YourResult: string; YourError: 0)
 5  Return: YourNumber: unchanged
 6          If YourNumber negative, YourResult: blank character string
 7                      YourError = 1
 8          If YourNumber >0 or 0, YourResult: string of characters for the number
 9                      YourError = 0}
 10  {Implementation for the spelling procedure.}
 11  PROCEDURE SPELLING(var InputInteger: integer; var SomeString: string;
 12                     var SomeError: integer);
 13    var i, {index}
 14        Length, {maximum number length}
14*        Test, {a trial number}
 15        SomeInteger, {a trial number}
 16        N, {tens variable}
 17        Offset: integer; {offset value to convert digits to ASCII characters}
 18        Flag: boolean; {to show leading 0s}
 19        SpelledNumber : string; {trial string, becomes output}
 20    begin
 21        YourError = 0;
 22        SomeInteger := InputInteger;
 23        SomeString := '';
 24        Test := 0;
 25        SpelledNumber := '';
26*        {N := 1000;}
26*        N := 10000;
```

```
 27      Length := 6;
 28      Offset := 48;
 29      i := 1;
 30      Flag := false;
 31      {Next, cycle through the integer from highest-order to lowest-order digits to
 32      determine digit's value and add its character to SpelledNumber.}
 33      if SomeInteger < 0 {Does not work for negative integers.}
 34        then
 35          SomeError := 1
 36        else
 37          repeat {until i := Length}
 38            if SomeInteger > N {Is first digit >= 1?}
 39              then
40*               {begen}
40*               begin
 41                Test := SomeInteger;
42*               {Test := Test div N; {Find the leading digit value.)}
42*                Test := Test div N; {Find the leading digit value.}
 43               SomeInteger := SomeInteger mod(Test*N); {Eliminate the
 44                     leading digit.}
45*               {Test = Test + Offset; } {Set digit to its ASCII value.}
45*                Test := Test + Offset; {Set digit to its ASCII value.}
 46               SpelledNumber := SpelledNumber + chr(Test); {Add the
 47                     character to end of SpelledNumber.}
47*                Flag := true; {Set flag to take care of non-leading zeros.}
 48             end
 49            else {Check for zero characters.}
 50              if Flag
 51                then {If an internal character was 0, insert '0'.}
52*                 {SpeledNumber := SpelledNumber + '0'}
52*                 SpelledNumber := SpelledNumber + '0'
 53               else {If 0 or no character in leading positions, leave blank.}
 54                 SpelledNumber := SpelledNumber + ' ';
 55        i := i + 1; {Step to the next digit.}
 56        N := N div 10
 57      until i = Length; {Continue until all digits tested}
 58      SomeString := SpelledNumber; {Set output to resulting string}
 59  end; {PROCEDURE SPELLING }
```

<- Exercise 1 Answer: Tracking Time

Exercise 2 Answer: Defects, Code Reviews, and Checklists

Exercise 2 Answer: Defect Recording Log: Form LOGD ->

Contents List of Forms Tool Instructions

Defect Types	
10 Documentation	60 Checking
20 Syntax	70 Data
30 Build, Package	80 Function
40 Assignment	90 System
50 Interface	100 Interface

TSPi Defect Recording Log: Form LOGD

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: Line 4 has a closing P{ instead of a closing).

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: Line 14: The declaration for Test is missing

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: On line 26, N should start at 10,000 not 1,000.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: On line 40, begin is misspelled.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: On line 42, there is a closing) instead of a closing }.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: On line 45, = should be :=.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: After line 47, there should have been an instruction to reset Flag.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: On line 52, SpelledNumber is misspelled.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

[<- Exercise 2 Answer: Defects, Code Reviews, and Checklists]

Exercise 2 Answer: TSPi Defect Recording Log: Form LOGD

[Exercise 3 Answer: Program Size ->]

[Contents] [List of Forms] [Tool Instructions]

PSP Familiarization Exercise 3 Answer: Program Size

This exercise shows you how to count lines of code

Pascal Source Program

```
 1  {Spelling procedure; WSH; 06/29/91}
 2  {Purpose: This procedure converts positive or zero integers to strings.}
 3  {Usage instructions:
 4  Call: SPELLING (YourNumber: integer; YourResult: string; YourError: 0)
 5  Return: YourNumber: unchanged
 6          If YourNumber negative, YourResult: blank character string
 7                  YourError = 1
 8          If YourNumber >0 or 0, YourResult: string of characters for the number
 9                  YourError = 0}
10  {Implementation for the spelling procedure.}
11  PROCEDURE SPELLING(var InputInteger: integer; var SomeString: string;
1 12                  var SomeError: integer);
2 13    var i, {index}
3 14        Length, {maximum number length}
4 15        Test, {a trial number}
5 16        SomeInteger, {a trial number}
6 17        N, {tens variable}
7 18        Offset: integer; {offset value to convert digits to ASCII characters}
8 19        Flag: boolean; {to show leading 0s}
9 20        SpelledNumber : string; {trial string, becomes output}
10 21     begin
11 22        YourError = 0;
12 23        SomeInteger := InputInteger;
13 24        SomeString := '';
14 25        Test := 0;
15 26        SpelledNumber := '';
16 27        N := 10000;
17 28        Length := 6;
18 29        Offset := 48;
19 30        i := 1;
20 31        Flag := false;
   32       {Next, cycle through the integer from highest-order to lowest-order digits to
   33       determine digit's value and add its character to SpelledNumber.}
21 34        if SomeInteger < 0 {Does not work for negative integers.}
22 35          then
23 36            SomeError := 1
```

```
24  37        else
25  38          repeat {until i := Length}
26  39            if SomeInteger > N {Is first digit >= 1?}
27  40              then
28  41                begin
29  42                  Test := SomeInteger;
30  43                  Test := Test div N; {Find the leading digit value.}
31  44                  SomeInteger := SomeInteger mod(Test*N); {Eliminate the
    45                      leading digit.}
32  46                  Test := Test + Offset; {Set digit to its ASCII value.}
33  47                  SpelledNumber := SpelledNumber + chr(Test); {Add the
    48                      character to end of SpelledNumber.}
34  49                  Flag := true; {Set flag to take care of non-leading zeros.}
35  50                end
36  51              else {Check for zero characters.}
37  52                if Flag
38  53                  then {If an internal character was 0, insert '0'.}
39  54                    SpelledNumber := SpelledNumber + '0'
40  55                  else {If 0 or no character in leading positions, leave blank.}
41  56                    SpelledNumber := SpelledNumber + ' ';
42  57            i := i + 1; {Step to the next digit.}
43  58            N := N div 10
44  59        until i = Length; {Continue until all digits tested}
45  60        SomeString := SpelledNumber; {Set output to resulting string}
46  61    end; {PROCEDURE SPELLING }
```

[<- Exercise 2 Answer: TSPi Defect Recording Log: Form LOGD]

PSP Familiarization Exercise 3 Answer: Program Size

[Exercise 4 Answer: Task and Schedule Planning ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Example Task Planning Template: Form TASK

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Task		Plan Hours		Plan Size/Value			Actual		
Phase	Task Name	Total Team Hours	Cummulative Hours	Week No.	Planned Value	Cumulative PV	Hours	Cumulative Hours	Week No.
Strategy	Produce team strategy	7.00	7.00	1	4.35	4.35			
Plan	Overall planning	7.75	14.75	1	4.82	9.18			
Plan	Detailed plan	7.50	22.25	2	4.67	13.84			
Requ.	Need statement review	3.75	26.00	2	2.33	16.17			
Requ.	Produce SRS	4.50	30.50	2	2.8	18.97			
Requ.	Inspect SRS	5.00	35.50	2	3.11	22.08			
Requ.	Produce design standards	2.00	37.50	2	1.24	23.33			
Design	Produce SDS	6.25	43.75	3	3.89	27.22			
Design	SDS inspection	6.25	50.00	3	3.89	31.10			
Design	Coding standards	2.00	52.00	3	1.24	32.35			
Imp.	Implementation planning	3.25	55.25	3	2.02	34.37			
DLD	Detailed design	15.00	70.25	4	9.33	43.70			
DLDR	Detailed design review	10.00	80.25	5	6.22	49.92			
DLDI	Detailed design inspection	18.00	98.25	5	11.20	61.12			
Code	Code	5.00	103.25	6	3.11	64.23			
CDR	Code review	5.00	108.25	6	3.11	67.34			
Comp	Compile	2.00	110.25	6	1.24	68.58			
CDI	Code inspection	7.50	117.75	6	4.67	73.25			
Unit test	Unit test	7.50	125.25	7	4.67	77.92			
Imp.	Quality review	4.00	129.25	7	2.49	80.40			
TD	Test plan and development	5.50	134.75	7	3.42	83.83			

B&I	Build and integration	1.50	136.25	7	0.93	84.76			
ST	System test	18.00	154.25	8	11.20	95.96			
Doc	Documentation	2.50	156.75	8	1.56	97.51			
PM	Postmortem	4.00	160.75	8	2.49	100.00			
		160.75			100.0				

<- Exercise 3 Answer: Program Size

Exercise 4: Task and Schedule Planning

Exercise 4: Schedule Planning Template: Form SCHEDULE ->

Contents List of Forms Tool Instructions

TSPi Schedule Planning Template: Form SCHEDULE

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Line-of-Code Counting Standard: Pascal

		Plan			Actual			
Week No.	**Date**	**Direct Hours**	**Cumulative Hours**	**Cumulative Planned Value**	**Team Hours**	**Cumulative Hours**	**Week Earned Value**	**Cumulative Earned Value**
1		20	20	9.18	19.2	19.2	19.18	9.18
2		20	40	23.33	22.6	41.8	11.04	20.22
3		20	60	34.37				*30.33*
4		20	80	43.70				*40.44*
5		20	100	61.12				*50.55*
6		20	120	73.25				*60.66*
7		20	140	84.76				*70.77*
8		20	161	100.00				*80.88*
								90.99
								100.00

<- Exercise 4 Answer: Task and Schedule Planning

Exercise 4 Answer: Schedule Planning Template: Form SCHEDULE

Exercise 5 Answer: Estimating Project Completion ->

Contents List of Forms Tool Instructions

PSP Familiarization Exercise 5 Answer: Estimating Project Completion

The earned value for each week and the cumulative earned value are shown on the following SCHEDULE form. In addition, in the Cumulative Earned Value column, the projected EVs are given in italics for each week for the rest of the project. As is clear, at the present EV rate, the team will not complete the project until week 10.

Based on the data from the TASK and SCHEDULE templates, the team appears to have underestimated the work. The members have worked slightly more than the planned 20 hours per week but have not completed all the planned tasks. The tasks completed have a total estimated hours of 32.5 hours, and the team has taken 37.4 hours to do them. This is 15% more than the estimate.

To get back on schedule, the team must complete the remaining tasks in six weeks. These tasks were planned to take a total of 160.75 - 32.5 = 128.25 hours. If the 15% underestimate holds, this work could be expected to take 147.5 hours. The team has already done some of the work on these remaining tasks, judging by the fact that they have worked a total of 41.8 hours and completed 37.4 hours of tasks. Thus, they have worked 41.8 - 37.4 = 4.4 hours on the remaining tasks. This leaves 147.5 - 4.4 = 143.1 hours of work to complete in six weeks. This indicates that, to get back on schedule, the team will have to work 143.1/6 = 23.85 hours per week.

Although this estimate is not guaranteed, it is the most reasonable projection based on the work done to date. These data, however, are based on only two weeks of work and only 20.22% of the job, so there could be a substantial error. For example, if the team had completed the SRS inspection, which it has presumably almost finished, it would have earned 3.11 more EV, for a total of 23.33 in two weeks. At this rate, the projected job duration would be a little less than nine weeks.

[<- Exercise 4 Answer: Schedule Planning Template: Form SCHEDULE]

Exercise 5 Answer: Estimating Project Completion

[Exercise 5 Answer: Example Task Planning Template: Form TASK ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Example Task Planning Template: Form TASK

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Task		Plan Hours		Plan Size/Value			Actual		
Phase	Task Name	Total Team Hours	Cummulative Hours	Week No.	Planned Value	Cumulative PV	Hours	Cumulative Hours	Week No.
Strategy	Produce team strategy	7.00	7.00	1	4.35	4.35	3.8	3.8	1
Plan	Overall planning	7.75	14.75	1	4.82	9.18	6.6	13.4	1
Plan	Detailed plan	7.50	22.25	2	4.67	13.84	11.2	24.6	2
Requ.	Need statement review	3.75	26.00	2	2.33	16.17	5.3	29.9	2
Requ.	Produce SRS	4.50	30.50	2	2.8	18.97	6.1	36.0	2
Requ.	Inspect SRS	5.00	35.50	2	3.11	22.08			
Requ.	Produce design standards	2.00	37.50	2	1.24	23.33	1.4	34.4	2
Design	Produce SDS	6.25	43.75	3	3.89	27.22			
Design	SDS inspection	6.25	50.00	3	3.89	31.10			
Design	Coding standards	2.00	52.00	3	1.24	32.35			
Imp.	Implementation planning	3.25	55.25	3	2.02	34.37			
DLD	Detailed design	15.00	70.25	4	9.33	43.70			
DLDR	Detailed design review	10.00	80.25	5	6.22	49.92			
DLDI	Detailed design inspection	18.00	98.25	5	11.20	61.12			
Code	Code	5.00	103.25	6	3.11	64.23			
CDR	Code review	5.00	108.25	6	3.11	67.34			
Comp	Compile	2.00	110.25	6	1.24	68.58			
CDI	Code inspection	7.50	117.75	6	4.67	73.25			
Unit test	Unit test	7.50	125.25	7	4.67	77.92			
Imp.	Quality review	4.00	129.25	7	2.49	80.40			
TD	Test plan and development	5.50	134.75	7	3.42	83.83			

B&I	Build and integration	1.50	136.25	7	0.93	84.76			
ST	System test	18.00	154.25	8	11.20	95.96			
Doc	Documentation	2.50	156.75	8	1.56	97.51			
PM	Postmortem	4.00	160.75	8	2.49	100.00			
		160.75			100.0				

<- Exercise 5 Answer: Estimating Project Completion

Exercise 5: Example Task Planning Template: Form TASK

Exercise 5: Schedule Planning Template: Form SCHEDULE ->

Contents List of Forms Tool Instructions

TSPi Schedule Planning Template: Form SCHEDULE

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

		Plan			Actual			
Week No.	**Date**	**Direct Hours**	**Cumulative Hours**	**Cumulative Planned Value**	**Team Hours**	**Cumulative Hours**	**Week Earned Value**	**Cumulative Earned Value**
1		20	20	9.18	19.2	19.2	9.18	9.18
2		20	40	23.33	22.6	41.8	11.04	20.22
3		20	60	34.37				*30.33*
4		20	80	43.70				*40.44*
5		20	100	61.12				*50.55*
6		20	120	73.25				*60.66*
7		20	140	84.76				*70.77*
8		20	161	100.00				*80.88*
9								*90.99*
10								*100.00*

<- Exercise 5 Answer: Example Task Planning Template: Form TASK

Exercise 5 Answer: Schedule Planning Template: Form SCHEDULE

References ->

Contents List of Forms Tool Instructions

References

Watts S. Humphrey, *A Discipline for Software Engineering.*
Reading, MA: Addison-Wesley Publishing Co., 1995.

Watts S. Humphrey, *Introduction to the Personal Software Process.*
Reading, MA: Addison-Wesley Publishing Co., 1997.

Watts S. Humphrey, *Introduction to the Team Software Process.*
Reading, MA: Addison-Wesley Publishing Co., 2000.

[<- Exercise 5: TSPi Schedule Planning Template: Form SCHEDULE]

References

[Contents] [List of Forms] [Tool Instructions]

TSPi Forms

This supplement contains copies of the TSPi forms. The following table lists these forms in alphabetical order by their abbreviations. It shows the pages where the form appears in the book *Introduction to the Team Software Process* (TSPi) and the page in this supplement where they appear. The instructions for these forms are contained in the TSPi textbook. Additional information on use of these forms is also contained in my earlier textbook, *A Discipline for Software Engineering.*[1] Watts S. Humphrey, A Discipline for Software Engineering. Reading, MA: Addison-Wesley Publishing Company, 1995.

Form	Abbreviation	Reference	Page
Configuration Change Request	CCR	376, 454	2
Configuration Status Report	CSR	380, 456	3
Student Information Sheet	INFO	49, 458	4
Inspection Report	INS	398, 403, 405, 460	5
Issue Tracking Log	ITL	462	6
Defect Recording Log	LOGD	464	7
Time Recording Form	LOGT	466	8
Test	LOGTEST	197, 468	9
Team and Peer Evaluation	PEER	224. 470	10
Process Improvement Proposal	PIP	472	11
Schedule Planning Template	SCHEDULE	94, 474	12
Strategy Form	STRAT	67, 152, 476	13
Defects Injected Summary Form	SUMDI	478	14
Defects Removed Summary Form	SUMDR	480	15
Plan Summary	SUMP	102, 482	16
Quality Plan	SUMQ	98, 181, 485	18
Size Summary	SUMS	89, 489	20
Time Summary Form	SUMT	491	21
Task Summary Form	SUMTASK	493	22
Task Planning Template	TASK	92, 495	23
Weekly Status Report	WEEK	53, 498	24

[1]Watts S. Humphrey, *A Discipline for Software Engineering.* Reading, MA: Addison-Wesley Publishing Company, 1995.

TSPi Configuration Change Request: Form CCR

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Product Information

Product Name _____ Product Owner _____

Product/Change Size _____ Size Measure _____

Recent Inspection _____ Moderator _____

Change Information

Reason for Change: _____

Change Benefits: _____

Change Impact: _____

Change Description: (For source code, attach listing; for code changes, include defect number (if any) and listing of changed and unchanged program segment)

Status

_____

Approved: ☐ Additional Information: ☐ Disapproved: ☐

Information needed: _____

_____

_____

Approvals

Product Owner	_____	Date:	_____
Quality/Process manager	_____	Date:	_____
CCB	_____	Date:	_____

[<- TSPi forms]

Configuration Change Request

[Configuration Status Report ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Configuration Status Repart: Form CSR

Name		Date	
Team		Instructor	
Part/Level		Cycle	

Configuration Change Process: Activity

	Current Week	Cycle to Date
CCRs submitted		
CCRs approved		
CCRs rejected		
CCRs defered		
CCRs outstanding		
CCRs reversed		

Configuration Change Process: Status

Product volume under SCM control

	Current Week	Cycle from Prior Week
Text pages		
Design pages		
Pseudocode Lines		
LOC - total		
LOC - new and changed		
Test case LOC		
Test material pages		
Test results pages		

Other items _____ _____

Other items _____ _____

Comments _____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

[<- Configuration Change Request]

Configuration Status Report

[Student Information Sheet ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Student Information Sheet: Form INFO

Name _____ Instructor _____

Date _____ Number of College Credits _____

Major _____ Expected Graduation Date _____

Briefly describe your relevant experience and interests:

_____

_____

_____

Briefly describe your work on other team projects:

_____

_____

_____

Briefly describe any leadership or management positions you have held (at work or in clubs/organizations):

_____

_____

_____

State your team preferences, if any:

_____

_____

_____

List your class schedule and other times when you have scheduled activities such as work, ROTC, clubs, sports teams, etc.

Time	Mon.	Tue.	Wed.	Thu.	Fri.	Sat.	Sun.
800-900							
915-1015							
1030-1130							
1145-1245							
1300-1400							

1415-1515							
1530-1630							
1645-1745							

Rank from 1 (least) to 5 (most) your preferences for serving in the following					
Team Leader	1	2	3	4	5
Development Manager	1	2	3	4	5
Planning Manager	1	2	3	4	5
Quality/Process Manager	1	2	3	4	5
Support Manager	1	2	3	4	5

<- Configuration Status Report

Student Information Sheet

Inspection Report ->

Contents List of Forms Tool Instructions

TSPi Inspection Report: Form INS

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Name	Defects[1]		Preparation Data			Est.
	Major	Minor	Size	Time	Rate	Yield
Totals:						

Defect Data

No	Defect Description	Defects		Engineers (finding major defects)					
		Maj	Min					A	B

Totals									
Unique Defects									

Inspection Summary Product Size _____ Size Measure _____

Total defects for A: _____ Total defects for B: _____ C (#common) _____

Total Defects (AB/C): _____ Number Found (A+B-C): _____ Number Left _____

Meeting Time: _____ Total Inspection Hours: _____ Overall Rate: _____

---

[1]Major defects either change the program source code or would ultimately cause a source code change if not fixed in time; all other defects are considered minor.

[<- Student Information Sheet]

Inspection Report

[Issue Tracking Log ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Issue Tracking Log: Form ITL

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

_____

_____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

_____

_____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

_____

_____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

_____

_____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

_____

_____

[<- Inspection Report]

Issue Tracking Log

[Defect Recording Log ->]

[Contents] [List of Forms] [Tool Instructions]

Defect Types	
10 Documentation	60 Checking
20 Syntax	70 Data
30 Build, Package	80 Function
40 Assignment	90 System
50 Interface	100 Interface

TSPi Defect Recording Log: Form LOGD

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description: _____

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

<- Issue Tracking Log

Defect Recording Log

Time Recording Form ->

Contents List of Forms Tool Instructions

TSPi Time Recording Form: Form LOGT

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Part/Level _____ Cycle _____

Date	Start	Stop	Interuption Time	Delta Time	Phase/Task	Component	Comments

<- Defect Recording Log

Time Recording Form

Test ->

Contents List of Forms Tool Instructions

TSPi Test Log: Form LOGTEST

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Part/Level _____ Cycle _____

Date	Phase	Product	Start	Stop	Interuption Time	Delta Time	Problems	Comments

<- Time Recording Form

Test

Team and Peer Evaluation ->

Contents List of Forms Tool Instructions

TSPi Team and Peer Evaluation: Form PEER

Name _____ Team _____ Instructor _____

Date _____ Cycle No _____ Week No. _____

For each role, evaluate the work required and the relative difficulty in % during this cycle.		
Role	Work Required	Role Difficulty
Team Leader		
Development Manager		
Planning Manager		
Quality/Process		
Support Manager		
Total Contribution (100%)		

Rate the overall team against each criterion. Circle one number from 1 (inaduquate) to 5 (superior).					
Team spirit	1	2	3	4	5
Overall effectiveness	1	2	3	4	5
Rewarding experience	1	2	3	4	5
Team productivity	1	2	3	4	5
Process quality	1	2	3	4	5
Product quality	1	2	3	4	5

Rate role for overall contribution. Circle one number from 1 (inadequate) to 5 (superior).					
Team Leader	1	2	3	4	5

Development Manager	1	2	3	4	5
Planning Manager	1	2	3	4	5
Quality/Process Manager	1	2	3	4	5
Support Manager	1	2	3	4	5

Rate each role for helpfulness and support. Circle one number from 1 (inadequate) to 5 (superior).					
Team Leader	1	2	3	4	5
Development Manager	1	2	3	4	5
Planning Manager	1	2	3	4	5
Quality/Process Manager	1	2	3	4	5
Support Manager	1	2	3	4	5

Rate each role for how well it was performed. Circle one number from 1 (inadequate) to 5 (superior).					
Team Leader	1	2	3	4	5
Development Manager	1	2	3	4	5
Planning Manager	1	2	3	4	5
Quality/Process Manager	1	2	3	4	5
Support Manager	1	2	3	4	5

<- Test

Team and Peer Evaluation

Process Improvement Proposal ->

Contents List of Forms Tool Instructions

TSPi Process Improvement Proposal: Form PIP

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Process _____ Phase _____

PIP Number _____ Priority _____

Problem Description

Briefly describe the problem encountered and its impact

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Proposal Description

Describe suggested changes as completely as possible, including affected forms, scripts, and so on.

_____

_____

_____

_____

_____

Submit complete PIP to the quality/process manager and keep a copy.

Do not write below this line

PIP Control # _____ Organization _____

Received _____ Acknowledged _____

Updated _____ Closed _____

Changes _____

<- Team and Peer Evaluation

Process Improvement Proposal

Schedule Planning Template ->

Contents List of Forms Tool Instructions

TSPi Schedule Planning Template: Form SCHEDULE

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Line-of-Code Counting Standard: Pascal

		Plan			Actual			
Week No.	Date	Direct Hours	Cumulative Hours	Cumulative Planned Value	Team Hours	Cumulative Hours	Week Earned Value	Cumulative Earned Value

<- Process Improvement Proposal

Schedule Planning Template

Strategy Form ->

Contents List of Forms Tool Instructions

TSPi Strategy Form: Form STRAT

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Reference	Functions	Cycle LOC			Cycle Hours		
		1	2	3	1	2	3

Totals							

[<- Schedule Planning Template]

Strategy Form

[Defects Injected Summary Form ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Defects Injected Summary Form: Form SUMDI

Plan _____ Assembly _____ Actual _____

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Parts	Strategy and Planning	Requirements	System test Plan	Requirements Inspection	High-Level Design	Integration Test Plan	HLD Inspection	Detailed Design	Detailed Design Review	Test Development	Detailed Design Inspection	Code	Code Review	Compile	Code Inspection	Unit Test	Build and Integration	System Test
Total																		

<- Strategy Form

Defects Injected Summary Form

Defects Removed Summary Form ->

Contents List of Forms Tool Instructions

TSPi Defects Removed Summary Form: Form SUMDI

Plan _____ Assembly _____ Actual _____

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Parts	Strategy and Planning	Requirements	System test Plan	Requirements Inspection	High-Level Design	Integration Test Plan	HLD Inspection	Detailed Design	Detailed Design Review	Test Development	Detailed Design Inspection	Code	Code Review	Compile	Code Inspection	Unit Test	Build and Integration	System Test
Total																		

<- Defects Injected Summary Form

Defects Removed Summary Form

Plan Summary ->

Contents List of Forms Tool Instructions

TSPi Plan Summary: Form SUMP

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Cycle _____

Product Size	**Plan**	**Actual**
Requirements pages (SRS)		
Other text pages		
High-level design pages (SDS)		
Detailed design lines		
Base LOC (B) (measured)		
Base LOC (B) (measured)	(Estimated)	(Counted)
Modified LOC (M)	(Estimated)	(Counted)
Added LOC (A)	(N-M)	(T+B+D-R)
Reused LOC (R)	(Estimated)	(Counted)
Total New and Changed LOC (N)	(Estimated)	(A+M)
Total LOC (T)	(N+B-M-D+R)	(Measured)
Total new Reuse LOC		
Estimated Object LOC (E)		
Upper Prediction Interval (70%)		

Lower Prediction Interval (70%) _____

Time in Phase (hours)	Plan	Actual	Actual %
Management and miscellaneous			
Launch			
Strategy and planning			
Requirements			
System test plan			
Requirements inspection			
High-level design			
Integration test plan			
High-level design inspection			
Implementation Planning			
Detailed design			
Detailed design review			
Test development			
Detailed design inspection			
Code			
Code review			
Compile			
Code inspection			
Unit test			
Build and integration			

System test			
Documentation			
Postmortem			
Total			

Total Time UPI (70%)	
Total Time LPI (70%)	

Defects Injected	**Plan**	**Actual**	**Actual %**
Strategy and planning			
Requirements			
System test plan			
Requirements inspection			
High-level design			
Integration test plan			
High-level design inspection			
Detailed design			
Detailed design review			
Test development			
Detailed design inspection			
Code			
Code review			
Compile			
Code inspection			

Unit test			
Build and integration			
System test			
Total Development			

Defects Removed	**Plan**	**Actual**	**Actual %**
Strategy and planning			
Requirements			
System test plan			
Requirements inspection			
High-level design			
Integration test plan			
High-level design inspection			
Detailed design			
Detailed design review			
Test development			
Detailed design inspection			
Code			
Code review			
Compile			
Code inspection			
Unit test			
Build and integration			

System test	_____	_____	_____
Total Development	_____	_____	

<- Defects Removed Summary Form

Plan Summary

Quality Plan ->

Contents List of Forms Tool Instructions

TSPi Quality Plan: Form SUMQ

Name		Date	
Team		Instructor	
Part/Level		Cycle	

Summary Rates	**Plan**	**Actual**
LOC/hour		
% Reuse (% of total LOC)		
% New Reuse (% of N&C LOC)		
Percent Defect-Free (PDF)		
In compile		
In unit test		
In build and integration		
In system test		
Defect/page		
Requirements inspection		
HLD inspection		
Defects/KLOC		
DLD review		
DLD inspection		
Code review		
Compile		
Code inspection		

Unit test		
Build and integration		
System test		
Total development		
Defects Ratios		
Code review/Compile		
DLD review/Unit test		
Development time ratios (%)		
Requirements inspection/Requirements		
HLD inspection/HLD		
HLD inspection/HLD		
DLD/code		
DLD review/DLD		
Code review/code		
A/FR		
Review rates		
DLD lines/hour		
Code LOC/hour		
Inspection rates		
Requirement pages/hour		
HLD pages/hour		
DLD lines/hour		
Code LOC/hour		

Defect-injection Rates (Defects/Hr.)		
Requirements		
HLD		
DLD		
Code		
Compile		
Unit test		
Build and integration		
System test		
Defect-removal Rates (Defects/Hr.)		
Requirements inspection		
HLD inspection		
DLD review		
DLD inspection		
Code review		
Compile		
Code inspection		
Unit test		
Build and integration		
System test		
Phase Yields		
Requirements inspection		
HLD inspection		

DLD review		
DLD inspection		
Code review		
Compile		
Code inspection		
Unit test		
Build and integration		
System test		
Process Yields		
% before compile		
% before unit test		
% before build and integration		
% before system test		
% before system delivery		

<- Plan Summary

Quality Plan

Size Summary ->

Contents List of Forms Tool Instructions

TSPi Size Summary: Form SUMDI

Plan _____ Assembly _____ Actual _____

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Product or Part Names and/or Numbers	Size Measure	Base	Deleted	Modified	Added	Reused	New and Changed	Total	Total new Reuse

Totals									

<- Quality Plan

Size Summary

Time Summary Form ->

Contents List of Forms Tool Instructions

TSPi Time Summary Form: Form SUMT

Plan _____ Assembly _____ Actual _____

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Parts	Strategy and Planning	Requirements	System test Plan	Requirements Inspection	High-Level Design	Integration Test Plan	HLD Inspection	Detailed Design	Detailed Design Review	Test Development	Detailed Design Inspection	Code	Code Review	Compile	Code Inspection	Unit Test	Build and Integration	System Test	Postmortem

<- Size Summary

Time Summary Form

Task Summary Form ->

Contents List of Forms Tool Instructions

TSPi Task Summary Form: Form SUMTASK

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Products of the Task _____

_____

Task users/customers _____

_____

_____

Overall requirements/messages _____

_____

_____

_____

_____

Specifications/constraints _____

_____

_____

Product elements	Size Unit	Plan Units	Plan Rate	Plan Time	Actual Units	Actual Time	Actual Rate

Total							

Time in Phase	Plan	Actual	%	Base Times
Planning				
Research				
Outline/High-level design				
Peer review and correct				
Draft/Detailed-level design				
Design review				
Peer review and correct				
Implementation/Rewrite				
Review				
Peer review and correct				
Test/Trial use				
Postmortem				
Total time (minutes)				
Total time (hours)				

<- Time Summary Form

Task Summary Form

Task Planning Template ->

Contents List of Forms Tool Instructions

TSPi Task Planning Template: Form SUMTASK

Name _____ Date _____

Team _____ Instructor _____

Part/Level _____ Instructor _____

Task				Plan Hours							Plan Siz/Value					Actual		
Phase	Part	Task Name	# Engineers	Team Leader	Development Manager	Planning Manager	Qual./Proc. Manager	Support Manager	Total Team Hours	Cumulative Hours	Size Units	Size	Week No.	Planned Value	Cumulative PV	Hours	Cumulative Hours	Week No.

[<- Task Summary Form]

Task Planning Template

[Weekly Status Report ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi Weekly Status Reports: Form WEEK

Name _____ Team _____ Instructor _____

Date _____ Cycle No _____ Week No. _____

Weekly Data	**Planned**	**Actual**
Project hours for this week		
Project hours for this cycle to date		
Earned value for this week		
Earned value for this cycle to date		
Total hours for the tasks completed this phase to date		

Team Member Weekly Data	**Hours Planned**	**Hours Actual**	**Planned Value**	**Earned Value**
Team Leader				
Development Manager				
Planning Manager				
Quality/Process Manager				
Support Manager				
Totals				

Development Tasks Completed	**Hours Planned**	**Hours Actual**	**Earned Value**	**Planned Week**

Totals				

Issue/Risk Tracking

Issue/Risk Name	Status

Other Significant Items

[<- Task Planning Template](#)

Weekly Status Report

[Contents](#) [List of Forms](#) [Tool Instructions](#)

Overview

Introduction

The TSPi support tool, TSPi.xls, was developed to provide automated support for the Team Software Process course described in the companion textbook, Introduction to the Team Software Process.

The support tool is implemented as a Microsoft Excel workbook.

Project team support

The TSPi support tool helps the project team

- Plan the project
- Track the project schedule
- Track the project quality plan

Minimum system configuration

TSPi.xls requires Microsoft Excel, version 97SR-1 or a later version.

Disk space requirements will depend on the amount of data entered. A minimum allocation of 3MB for each copy of the tool is recommended.

Each project team will require one copy for each team member and two copies for the team.

Installation

To install the workbook, load the file TSPi.xls on your hard drive.

Startup

Open the file TSPi.xls to startup the support tool.

Step-by-step instructions

Step-by-step instructions for using the support tool can be found in this document and in the TSPi support tool on the worksheet tab labeled Instructions.

Forms overview

- LOGD: Defect Recording Log
- LOGT: Time Recording Log
- SCHEDULE: Schedule Planning Template
- SUMQ: Quality Plan Summary
- SUMP: Program Plan Summary
- SUMS: Size Summary
- TASK: Task Planning Template
- ROLE: Team Roles

Planning overview

Plans are produced at the beginning of each cycle during the planning phase of the TSPi.

Three types of plans are produced during the planning phase.

1. A top-down team plan for the cycle. The top-down plan starts the planning process and provides a basis for creating the individual plans and the consolidated team plan.
2. Individual plans for each team member. Individual plans are based on the top-down plan tasks that are assigned to each team member. They support planning and tracking at the individual level.
3. A consolidated team plan that integrates the individual team member plans. The consolidated plan reflects the plans and commitments of each team member. The consolidated plan is used for tracking.

Tracking overview

Project status is reported at the weekly status meeting. Status is based on data gathered by each team member. Team members record these data in their own copy of the support tool workbook.

The data are:

1. Size: The measured size of each product is entered on the worksheet SUMS.
2. Time: Development time is entered on the worksheet LOGT.
3. Defects: Defects are entered on the worksheet LOGD.
4. Task Completion: The completion date for each task is entered on the TASK worksheet

These data are then automatically integrated into the consolidated team plan to produce team status. The planning manager is responsible for

consolidation.

Project status charts

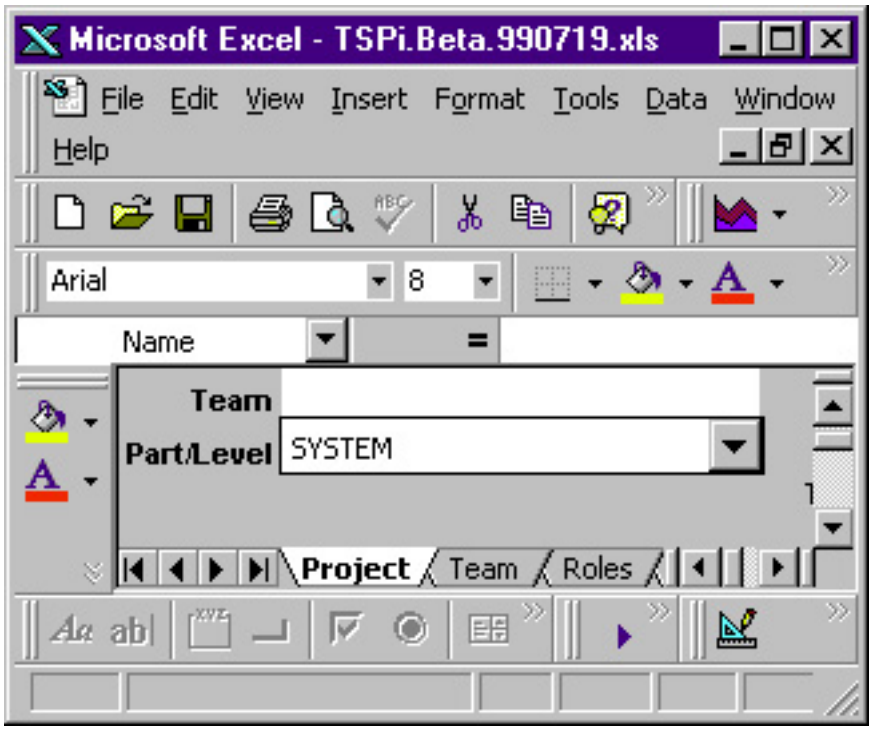
The project worksheet includes five analysis charts.

- Earned Value: Plots cumulative planned, earned, and predicted value for each project week.
- Plan versus Actual Hours: Plots cumulative planned and cumulative actual hours for each project week.
- Defect Removal Profile: Plots defects removed per KLOC, for each defect removal phase, for any assembly.
- Quality Profile: Plots performance against five quality benchmarks for any assembly.
- Percent Defect Free: Plots the percentage of the system that is free of defects at key points in the process.

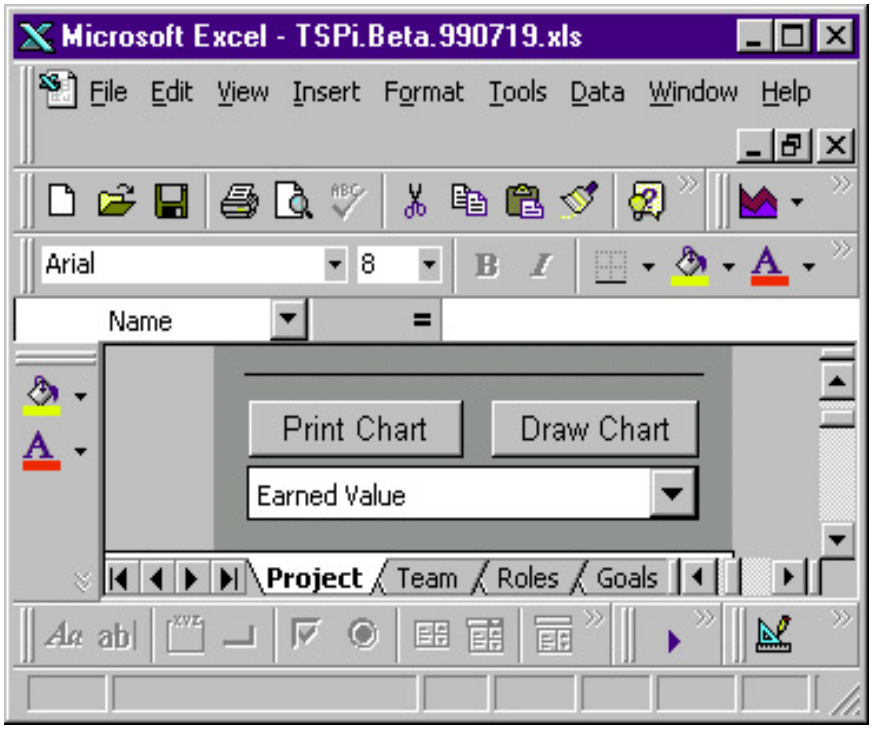
Displaying and printing the analysis charts

A set of four controls‹two drop-down menus and two buttons‹on the PROJECT worksheet control the analysis charts.

- Part/Level drop-down menu: Selects the part or assembly to display, or all parts and assemblies if SYSTEM is selected.



- Chart drop-down menu: Selects one of the five charts.
- Draw Chart button: Forces the selected chart to be redrawn.
- Print Chart button: Prints the selected chart.



Overview

Project Planning ->

Contents List of Forms Tool Instructions

Project Planning

When are project plans produced?

Project plans are produced during the TSPi development strategy and planning sessions at the start of the project.

During the first development cycle, the STRAT1 and PLAN1 scripts are used. For subsequent cycles the STRATn and PLANn scripts are used.

Since tool use is identical in both cases, first-cycle scripts are referenced hereafter. Each subsequent cycle is treated as a new project.

How are project plans produced?

The process scripts STRAT1 and PLAN1 describe how plans are produced. The tool facilitates this process by providing support for key planning forms, such as TASK and SCHEDULE.

(A detailed description of how to use the TSPi.xls workbook to support planning is included below and in the workbook. See the worksheet labeled Instructions.)

Starting the plan

Before the first team session (STRAT1 meeting) the team leader should prepare a workbook for the team.

1. Select the PROJECT worksheet and enter your name, the project name, the team name, and the Monday date for the starting week in start date, the instructor name, and the cycle number.
2. Next, select the TEAM worksheet and enter each team member's name, initials, phone number, and e-mail address.
3. Next, select the ROLES worksheet and enter the role assignments for the current cycle.

Updating the plan during the STRAT1 meeting

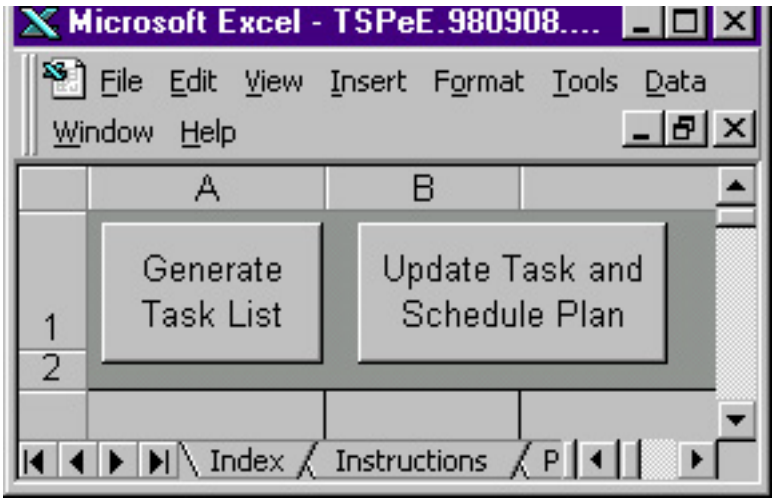
During the STRAT1 meeting, the planning manager updates the workbook as the planning proceeds.

1. Enter the assembly names, parts names, and size estimates on the SUMS worksheet.

Creating the team TASK and SCHEDULE plan

During the PLAN1 meeting, the planning manager enters task and schedule information on the TASK and SCHEDULE plan worksheets in the team¹s workbook.

1. First, select the TASK worksheet and click the Generate Task List button to create a default task list from the defined assemblies and the standard TSPi process steps.



1. , revise the task list as needed and make a size estimate for tasks where appropriate.
2. Next, estimate the hours for each task and, where known, the hours for each engineer under his or her assigned role.
3. Now select the SCHEDULE worksheet and enter the total available hours for the team, by week.
4. Next, select the TASK worksheet and click the Update Task and Schedule Plan button to generate a plan.

Producing the quality plan

Once the team task and schedule plan has been completed, the team produces the quality plan. (Review the quality standard in the textbook before entering the quality plan data.)

1. First, select the SUMQ worksheet and enter the planned defect injection rates for each phase.
2. Next, enter estimated phase yields for each phase.
3. Now review the calculated defect removal rates and defect densities (Defects/KLOC) against the Quality Standard. If necessary, adjust planned time (TASK worksheet), defect injection rates (SUMQ), or defect removal yields (SUMQ).

Creating individual TASK and SCHEDULE plans

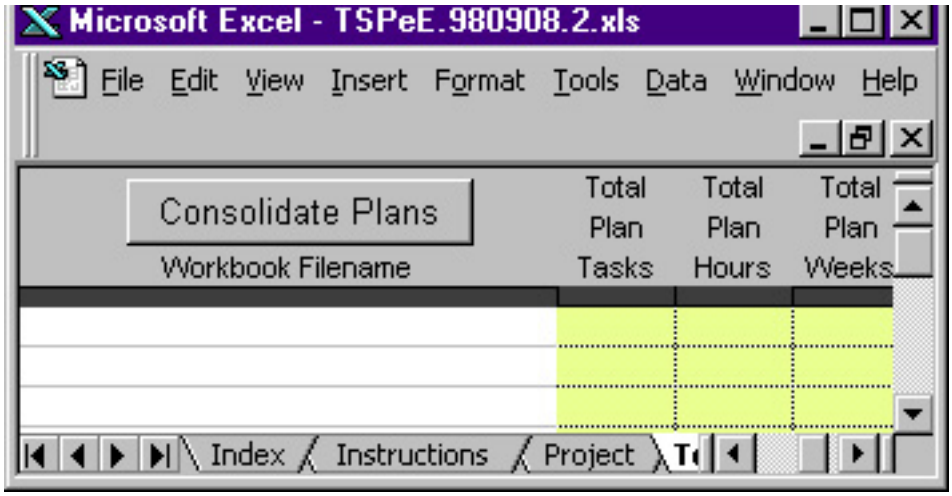
After the team task and schedule plan has been created, each team member prepares a personal task and schedule plan.

1. The team leader makes one copy of the team workbook for each team member, enters the filenames of these workbooks in the Workbook Filename column on the TEAM worksheet, and distributes the copies to the team members.
2. Next, each team member enters his or her name in the PROJECT worksheet of his or her personal workbook.
3. Next, each team member edits the task list in his or her personal workbook, deleting tasks not assigned to him or her, entering 0 hours for other engineers assigned to the same tasks, and adding and estimating any additional tasks he or she believes are required.
4. Finally, each team member revises the SCHEDULE worksheet to reflect his or her available hours and then clicks the Update Task and Schedule Plan button on the TASK worksheet to generate the schedule.

Creating the consolidated plan

After individual task and schedule plans have been created, the team prepares a consolidated team task and schedule plan.

1. The team reviews the task and schedule plans for all team members to see if they are balanced. If they are not balanced, reassign tasks among the engineers and repeat steps 3 and 4 of the previous section, Creating individual TASK and SCHEDULE plans.
2. When the individual plans are balanced, the planning manager makes a copy of the team's top-down plan workbook for use in producing the consolidated plan.
3. The team member¹s individual worksheets are then loaded into the same folder or directory as the team's consolidated plan.
4. Next, open the workbook for the consolidated team plan and click the Consolidate Plans button on the TEAM worksheet to build the consolidated plan.



1. Check again to ensure that the plan is balanced, and, if necessary, reassign tasks and repeat steps 3 and 4 of the previous section, Creating individual TASK and SCHEDULE plans. Then return to step 3 of this section.
2. If the planned time for any task is changed, repeat step 3 of the section, Producing the quality plan.

<- Overview

Project Planning

Tracking the Project Schedule ->

Contents List of Forms Tool Instructions

Tracking the Project Schedule

When is the schedule updated?

The project schedule is updated weekly before the weekly project meeting.

Which charts and worksheets are used to track schedules?

Schedules are tracked with the following charts and worksheets.

- Earned Value chart
- Planned and Actual Hours chart
- PROJECT worksheet
- TASK worksheet
- SCHEDULE worksheet
- SUMP worksheet
- SUMQ worksheet
- SUMS worksheet

How do team members prepare their individual status reports?

Use the following procedure to generate team member status.

1. Ensure that all time entries in the LOGT worksheet for the current period are complete and accurate.
2. 2Ensure that the completion dates for all completed tasks have been entered in the TASK worksheet.
3. 3Ensure that actual size measures for all assemblies in the SUMS worksheet are complete and accurate.
4. Click the Update Project button on the PROJECT worksheet.
5. Select the Earned Value chart on the PROJECT worksheet. Click the Draw button, and then click the Print button.
6. Select the Planned and Actual Hours chart on the PROJECT worksheet. Click the Draw button, and then click the Print button.
7. Print the TASK and SCHEDULE worksheets.

How is the team's schedule status report prepared?

Use the following procedure to generate the team's status.

1. Ensure that all team members have updated their individual workbooks using the above procedure.
2. Ensure that the updated team member workbooks are saved in the same directory as the team workbook.
3. Click the Consolidate button on the TEAM worksheet.
4. Select the Earned Value chart on the PROJECT worksheet. Click the Draw button and the Print button.
5. Select the Planned and Actual Hours chart on the PROJECT worksheet. Click the Draw button and the Print button.
6. Print the TASK and SCHEDULE worksheets.
7. the SUMP worksheet and print each assembly, including SYSTEM.

<- Project Planning

Tracking the Project Schedule

Tracking the Project Quality Plan ->

Contents List of Forms Tool Instructions

Tracking the Project Quality Plan

When is the quality plan updated?

The project quality plan is updated weekly before the weekly project meeting.

Which charts and worksheets are used to track quality?

Quality is tracked with the following charts and worksheets.

- Defect Removal Profile chart
- Quality Profile chart
- Percent Defect Free chart
- SUMP worksheet
- SUMQ worksheet
- SUMS worksheet

How do team members prepare their quality status report?

Quality status is reported for each product assembly. Use the following procedure to generate a team member status report for all assigned assemblies.

1. Ensure that all defect entries in the LOGD worksheet for the current period are complete and accurate.
2. Ensure that actual size measures for all assemblies in the SUMS worksheet are complete and accurate.
3. Click the Update Project button on the PROJECT worksheet, then repeat steps 4 to 7 for each assembly assigned to the team member.
4. Select the Defect Removal Profile chart on the PROJECT worksheet. Click the Draw button and the Print button.
5. Select the Quality Profile chart on the PROJECT worksheet. Click the Draw button and the Print button.
6. Print a SUMP worksheet for each assembly.
7. Print a SUMQ worksheet for each assembly.

How is the team's quality status report prepared?

Quality status is reported for each product assembly. Use the following procedure to generate the team's status.

1. Ensure that all team members have updated their individual workbooks using the above procedure and that the updated team member workbooks are saved in the same directory as the team workbook.
2. Click the Consolidate button on the TEAM worksheet, and then repeat steps 3 to 7 for each assembly.
3. Select the Defect Removal Profile chart on the PROJECT worksheet. Click the Draw button and the Print button.
4. Select the Quality Profile chart on the PROJECT worksheet. Click the Draw button and the Print button.
5. Select the Percent Defect Free chart on the PROJECT worksheet. Click the Draw button and the Print button.
6. Print a SUMP worksheet for each assembly.
7. Print a SUMQ worksheet for each assembly.

[<- Tracking the Project Schedule]

Tracking the Project Quality Plan

[User Interface Features ->]

[Contents] [List of Forms] [Tool Instructions]

User Interface Features

The user interface

The user interface incorporates several custom features of Microsoft Excel.

- Cell protection
- Background color
- Default values
- Automatically calculated cells
- Shortcut menus
- Chart Wizard
- Visual Basic (VB) macros

Review the following topics before using the TSPi workbook.

Worksheet protection and locked (protected) cells

Worksheet cells that should not be changed by the user are locked wherever possible. This prevents the user from inadvertently modifying cells that should not be changed.

Worksheet protection can be turned off temporarily using the *Tools/Protection* menu item, but on some worksheets a VB macro may turn protection back on whenever a cell is changed or a new cell is selected.

Avoid changing worksheet protection or the protected status of cells without a clear understanding of the possible side effects.

Cell background color

Cell background colors are used to indicate which cells the user must complete and which cells contain calculated values or formulas.

- Cells with a white background contain values the user must supply.
- Cells with a light-yellow background contain calculated values or formulas.
- Cells with a blue background are labels or form areas. Do not type in these cells.

Cell formulas and links

Quality status is reported for each product assembly. Use the following procedure to generate the team's status.
Most worksheet calculations are handled by VB macros instead of Excel worksheet formulas. Some cell formulas or links are actually inserted by VB macros. Changing cell formulas may not produce a long-term change, because the VB macros may overwrite any changes you insert.

Default values

A default value is a predefined value that is inserted into an empty cell when you select the cell. In general, this action is triggered only if valid data have been entered in the preceding rows and the preceding cells in the current row.

Example 1. Selecting the date cell in a LOGD worksheet row automatically enters today¹s date.

Example 2. Selecting an empty assembly, phase, or task cell in a LOGD worksheet row automatically inserts, as a default value, the corresponding value from the same column in the previous row, but only if the preceding items (assembly, phase, and task) are not blank.

Overriding defaults. Type over the value in the cell.

Restoring defaults. Delete the value in the cell, then reselect the cell.

Individual worksheet instructions provide specific descriptions of the default values for each worksheet cell.

Automatically calculated cells

An automatically calculated cell is a cell whose value can be computed based on other cells in the same row.

Example 1. Delta Time on the LOGT worksheet can be calculated from the values in cells Start, Interrupt, and Stop.

Delta Time = Stop - Start - Interrupt

This feature is not the same as an Excel cell formula. Whenever the source values change, a procedure is called to recalculate the affected cell.

Example 2. Changing Start, Stop, or Interrupt will cause Delta Time to be recalculated.

Do not try to override automatically calculated cells.

Shortcut menus

Context-sensitive shortcut menus are provided for entering lists of data, such as defect types and task names.

Shortcut menus are activated by a click of the right mouse button. Shortcut menus contain context-sensitive actions for the item underneath the cursor.

<- Tracking the Project Quality Plan

User Interface Features

User Interface Contraints ->

Contents List of Forms Tool Instructions

User Interface Contraints

Impact on the Excel user interface

Some custom features of the user interface affect the standard user interface for Excel.

- Copy and Paste
- Multiple selections
- Inserting, deleting, copying, or moving rows
- Inserting, deleting, copying, or moving columns
- Customizing/using TSPi.xls worksheet data

Copy and Paste

Copy and Paste may not work between worksheets or workbooks. Event-driven VB macros triggered by changing the displayed workbook may cause the area selected for Copy to change, which will affect the Copy and Paste commands.

Copy and Paste within a worksheet may also be affected.

Multiple selection

Editing operations involving multiple selections (more than one worksheet cell) may produce different results or may not be allowed.

Also, automatic calculations may not work when a range larger than a single cell is changed. Each cell may need to be edited separately.

Inserting, deleting, copying, or moving rows

Editing by inserting, deleting, copying, or moving entire rows is allowed on some worksheets. Excel row and column headings are not displayed when row editing is not allowed; they are displayed when row editing is permitted.

See the detailed instructions for the worksheet.

Inserting, deleting, or moving columns

Inserting, deleting, or moving columns will destroy the workbook and are not recommended.

See Customizing/using TSPi.xls worksheet data, below.

Customizing/Using TSPi.xls worksheet data

Modifying TSPi.xls worksheets may cause the custom workbook software to fail and/or corrupt the data in the worksheets.

To access data in the workbook, create a new workbook and use cell references to the TSPi.xls worksheets to access the data required for the enhancement.

Example: To add a feature that shows the ratio of interrupt time to elapsed time worked for a LOGT worksheet entry, create a new workbook and a formula like the one below. Refer to the TSPi data.

=[TSPi.xls]LOGT!F2/[TSPi.xls]LOGT!H2

[<- User Interface Features]

User Interface Contraints

[TSPi.xIs worksheets ->]

[Contents] [List of Forms] [Tool Instructions]

TSPi.xls worksheets

Worksheet forms support

TSPi.xls includes these TSPi forms implemented as worksheets.

1. Defect Recording Log (LOGD)
2. Time Recording Log (LOGT)
3. Schedule Plan Template (SCHEDULE)
4. Program Plan Summary (SUMP)
5. Quality Plan Summary (SUMQ)
6. Size Summary (SUMS)
7. Task Plan Template (TASK)

Other worksheets included with TSPi.xls, and their purposes, are as follows.

1. Instructions: step-by-step instructions for planning and tracking
2. Project: a project summary
3. Team: team member information
4. Roles: role assignments
5. Defect Types: the defect types allowed on the LOGD worksheet
6. QprofParam: the parameters used for calculating the five points on a quality profile chart

[<- User Interface Contraints]

TSPi.xIs worksheets

[The LOGT Worksheet ->]

[Contents] [List of Forms] [Tool Instructions]

The LOGT Worksheet

Overview

The LOGT worksheet implements the TSPi LOGT form.

Data entered by the engineer on this form are rolled up to worksheets TASK, SCHEDULE, and SUMP.

Delta time

Delta time is automatically calculated when start time, interrupt time, and stop time have been entered.

Delta time is recalculated when any of these values is changed.

Shortcut menu

Use the shortcut menu (accessed by clicking the right mouse button) to enter assembly names, phase names, and task names.

Using the shortcut menu to enter a task name will enter the assembly name and phase name automatically if these fields are blank.

Application note

To enter time for a task that is already completed, such as a task that is conducted away from your computer (for example, an inspection meeting),

1. enter the assembly, phase, and task;
2. enter the desired date, start time, and stop time in these fields; include any interruption time;

The Delta time will be calculated automatically.

Fields

The following table describes the fields on the LOGT worksheet.

Field	Contents	Default Value or Calculated Value
Assembly	Assembly name (use the shortcut menu)	Defaults to value in prior row
Phase	Phase name (use the shortcut menu)	Defaults to value in prior row
Task	Task name (use the shortcut menu)	Defaults to value in prior row
Date	The start date	Defaults to the current date
Start	The start time	Defaults to the current time
Int.	The total minutes of interruption time; any number <= (stop - start)	
Stop	The stop time	Defaults to the current time
Delta	Elapsed minutes - interrupts	Calculated as *stop-start-interrupts*
Comments	Any comments	

[<- TSPi.xIs worksheets]

The LOGT Worksheet

[The LOGD Worksheet ->]

[Contents] [List of Forms] [Tool Instructions]

The LOGD Worksheet

Overview

The LOGD worksheet implements the TSPi LOGD form.

Data entered by the engineer on this form are rolled up to the SUMS, SUMP, and SUMQ worksheets.

Shortcut menu

Use the shortcut menu (accessed by clicking the right mouse button) to enter assembly names, phase names for phase injected and phase removed, and defect types.

Controls

None

Fields

The following table describes the fields on the LOGD worksheet.

Field	Contents	Default Value or Calculated Time
Date	A valid date	Defaults to the current date
Num		Calculated sequence of defect numbers
Type	A defect type (use the shortcut menu)	Defaults to value in prior row
Assembly	An assembly name (use the shortcut menu)	Defaults to value in prior row
Injected	A phase name (use the shortcut menu)	Defaults to value in prior row
Removed	A phase name (use the shortcut menu)	Defaults to value in prior row
Fix Time	A numeric value	
Fix Ref.	A valid defect number	
Description	A description of the defect	

<- The LOGT Worksheet

The LOGD Worksheet

The TASK Worksheet ->

Contents List of Forms Tool Instructions

The TASK Worksheet

Overview

The TASK worksheet implements the TSPi Task Planning template.

This worksheet is used in conjunction with the SCHEDULE worksheet to produce the TSPi task and schedule plans.

Size and time estimates

Size and time estimates are entered on the TASK worksheet.

1. Enter the Size, Size Measure, and Rate, and the spreadsheet will calculate Estimated Hours. *Estimated Hours = Size / Rate*
2. Enter the number of hours for each engineer by role to calculate planned hours.
3. Alternatively, enter the plan hours directly.

Application note

Because a task¹s plan hours are the sum of the planned hours by role, an engineer¹s task plan should include only his or her planned hours on a task.

Task completed

When tasks are 100% complete, enter the actual date, and the worksheet will calculate the actual week.

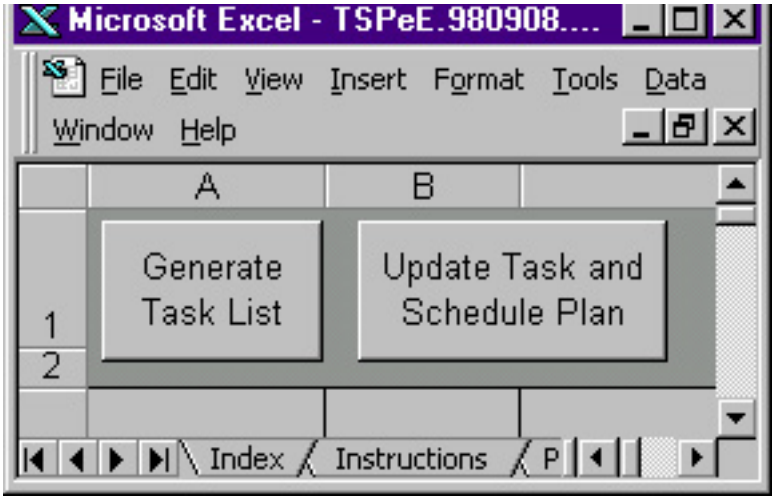
Shortcut menu

Use the shortcut menu (accessed by clicking the right mouse button) to enter assembly names and phase names.

Controls

The TASK worksheet has two controls.

1. Generate Task List: This control builds a default list of tasks on the TASK worksheet. The task list is derived from the TSPi process and the assemblies on the Assembly worksheet.
2. Update Task and Schedule Plan: This control updates the TASK and SCHEDULE worksheets with development time data from LOGT, recalculates earned value based on the date tasks were finished (Actual Date), and updates the predicted earned value.



Fields

The following table describes the fields on the TASK worksheet.

Field	Contents	Default Value or Calculated Value
Assembly	Phase name (use the shortcut menu)	Defaults to value in prior row
Phase	Phase name (use the shortcut menu)	Defaults to value in prior row
Task	The name of the task	Defaults to value in prior row
Plan Hours by Role	The plan hours per engineer, by role	
Est. Size	The start time	
Size Measure	Text Pages, Requirements Pages, HLD Pages, DLD Lines, LOC, or a user-defined measure	
Rate	A development or production rate, for example, 20 LOC/Hour	
Estimated Hours	A calculated or user-entered value	Calculated as *Size / Rate*

Plan Hours	A calculated or user-entered value	*Sum of plan hours for each role*
Actual Hours	Actual hours for this task from the LOGT worksheet	Calculated from LOGT as *Sum(DeltaTime) for the task*
Actual Date	The date the task was completed	Defaults to the current date

Calculated fields

The following table describes the calculated fields on the TASK worksheet. Do not enter data in calculated fields.

Field	Contents	Calculated Value
Actual Week	The project week for Actual Dat	Calculated based on Start Date and Actual Date
EV	Earned value	Based on PV
Cum EV	Cumulative EV value	Cumulative total
Cum Plan Hours	Cumulative plan hours	Cumulative total
PV	Plan value	Plan Hours/Total Plan Hours
Plan Week	Planned week	Based on Plan Value
Plan Date	Planned date	Based on Plan Value

[<- The LOGD Worksheet]

The TASK Worksheet

[The SCHEDULE Worksheet ->]

[Contents] [List of Forms] [Tool structions]

The SCHEDULE Worksheet

Overview

The SCHEDULE worksheet implements the TSPi Schedule Planning template.

This worksheet is used in conjunction with the TASK worksheet to produce the TSPi task and schedule plans.

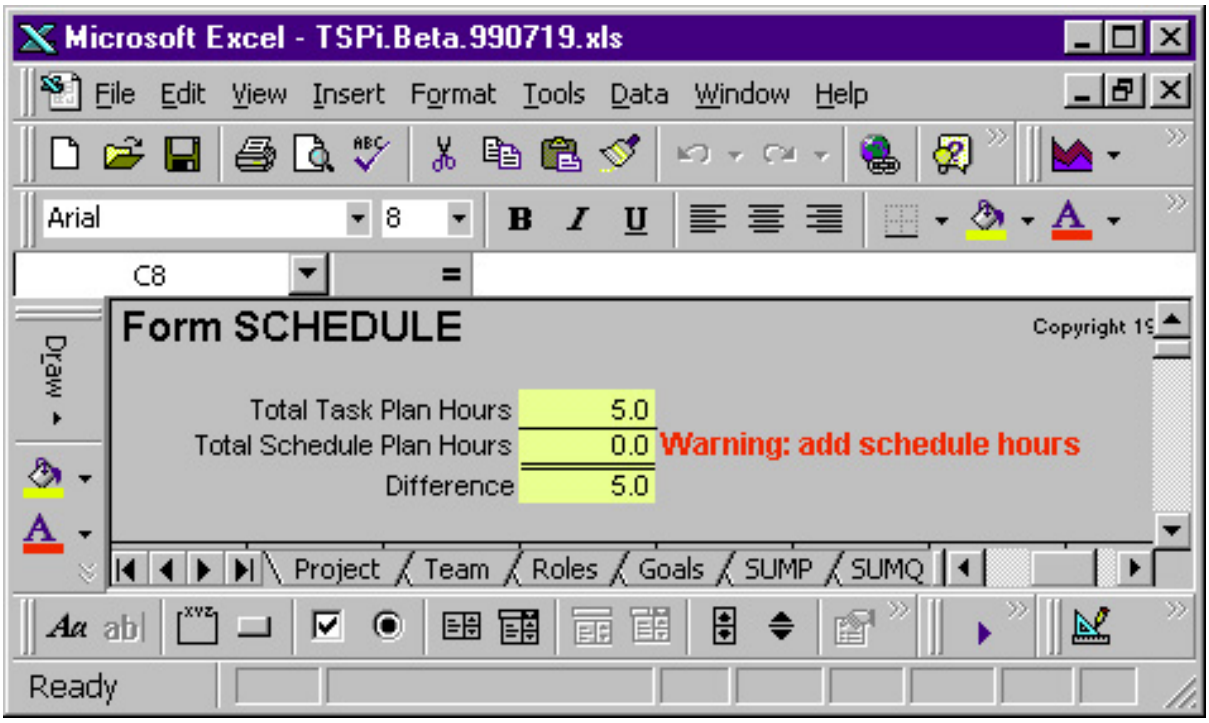
Plan Hours

To set up the SCHEDULE worksheet, enter the plan hours for each week of the schedule.

The SCHEDULE worksheet will automatically enter the date and week number as plan hours are entered.

The total plan hours on the SCHEDULE worksheet must be greater than or equal to the total plan hours on the TASK worksheet.

A summary of task plan hours and schedule plan hours at the top of the SCHEDULE worksheet shows the difference.



Deleting SCHEDULE weeks

To delete weeks on the SCHEDULE worksheet you must delete the plan hours for the weeks, one week at a time.

Fields

The following table describes the fields on the SCHEDULE worksheet.

Field	Contents	Default Value or Calculated Value
Date	The Monday date for the project week	Calculated based on project start date
Week	The project week	Calculated for each week
Plan Hours	Total hours planned for the week	Defaults to value in prior row

Calculated fields

The following table describes the calculated fields on the Schedule worksheet. Do not enter data in these fields.

Field	Contents	Default Value or Calculated Value
Cum Plan Hours	Cumulative plan hours	
Actual Hours	Actual hours for week from the LOGT	Sum(DeltaTime) for dates within the week
Cum Actual Hours	Cumulative actual hours	
PV	Plan value	For tasks planned for completion in week
Cum PV	Cumulative plan value	For the tasks actually completed in this week
EV	Earned value	
Cum EV	Cumulative earned value	
Predicted Hours		
Cum Predicted Hours		
Predicted EV	Predicted earned value	Based on average weekly EV
Cum Predicted EV	Cumulative predicted earned value	

<- The TASK Worksheet

The SCHEDULE Worksheet

The SUMP Worksheet ->

Contents List of Forms Tool Instructions

The SUMP Worksheet

Overview

The SUMP worksheet implements the TSPi SUMP Program Plan Summary form.

SUMP displays summary size, time, and defect data for any assembly or for the SYSTEM.

SUMP also includes productivity, CPI (cost-performance index), %Reuse, and %New Reuse.

Fields

The following table describes the data that are entered on the SUMP worksheet.

Data	Contents
%Reuse	Plan and actual %Reuse *%Reuse = Reuse LOC/Total LOC*
%New Reuse	Plan and actual %New Reuse *%New Reuse = New Reuse LOC/New and Changed LOC*

Calculated fields

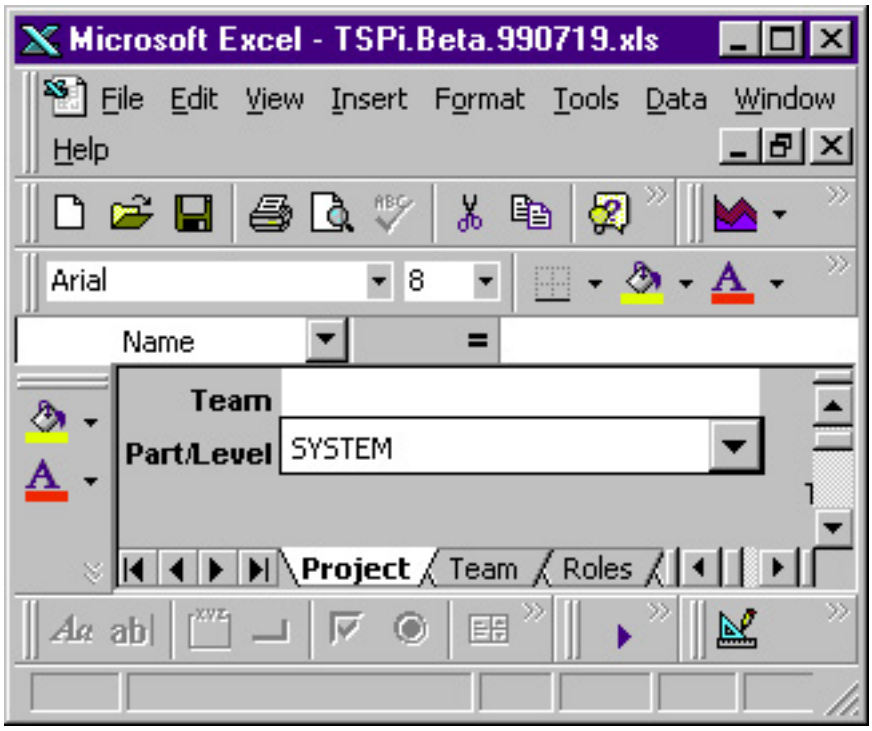
The following table describes the calculated data on the SUMP worksheet. Do not enter data in calculated fields.

Data	Contents
Program Size	Plan and actual size summary data ● requirements pages ● high-level design pages ● detailed design lines ● lines of code
Time in Phase	Plan and actual time for each TSPi phase
Defects Injected	Plan and actual defects injected in each TSPi phase from Planning through System Test
Defects Removed	Plan and actual defects removed in each TSPi phase from Planning through System Test

Productivity	Plan and Actual new and changed LOC per hour
CPI	Cost performance index *CPI = plan hours/actual hours*

Controls

The SUMP worksheet has one control, the Assembly Selector Control, which selects the assembly to display on the SUMP worksheet. The list of possible selections includes all the assemblies on worksheet SUMS and the predefined assembly SYSTEM.



[<- The SCHEDULE Worksheet](#)

The SUMP Worksheet

[The SUMQ Worksheet ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

The SUMQ Worksheet

Overview

The SUMQ worksheet implements the TSPi SUMQ Quality Plan Summary form.

SUMQ displays summary quality data for any assembly or for the SYSTEM.

Planned defect injection rates and planned phase yields are entered on this form.

Fields

The following table describes the data that are entered on the SUMQ worksheet.

Data	Contents	Default Value or Calculated Value
Plan Defect Injection Rate	Enter the planned defect injection rates for each phase.	
Plan Phase Yield	Enter the planned phase yields for each phase.	

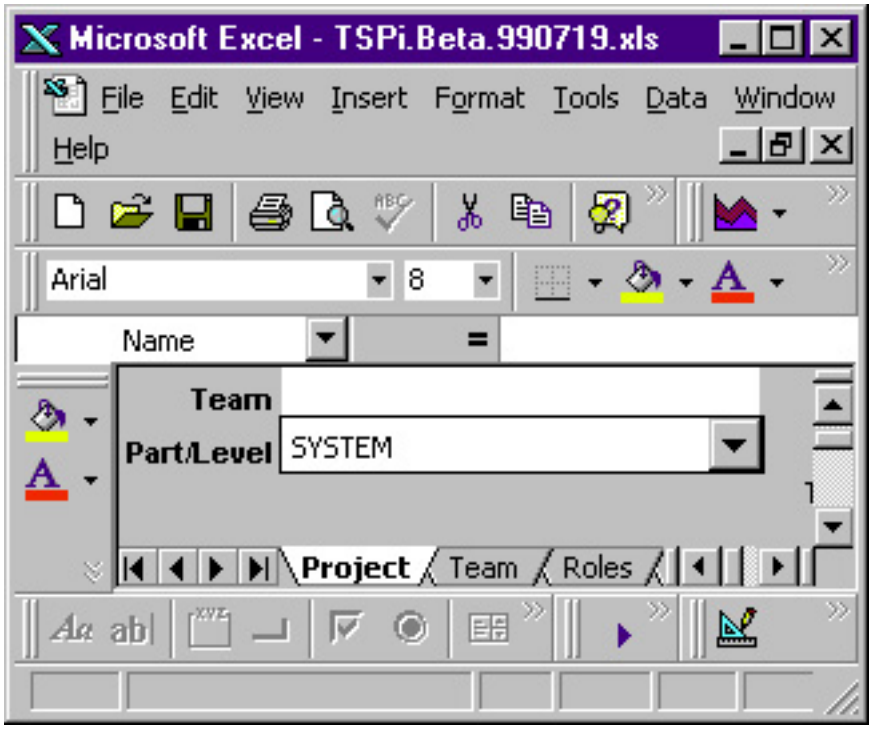
Calculated fields

The following table describes the calculated data on the SUMQ worksheet. Do not enter data in these fields.

Data	Contents
Percent Defect Free (plan and actual)	Percent of system assemblies free of defects by phase
Defects per Page (plan and actual)	Defects per page for requirements and high-level design documents
Defects per KLOC (plan and actual)	Defects per KLOC for the software by phase
Defect Ratios (plan and actual)	Defect ratios for design review vs. unit test and code review vs. compile
Development Time Ratios (plan and actual)	Development time ratios for phases: ● Requirements inspection/Requirements ● …High-level design inspection/High-level design ● Detailed design review/Detailed design ● Detailed design/Code ● Code review/ Code
Inspection and Review Rates (plan and actual)	LOC inspected or reviewed per hour
A/F Ratio (plan and actual)	Ratio of time spent in inspections and review phases vs. time spent in compile and testing phases
Phase Yield (actual only)	Yield by phase
Process Yield (plan and actual)	Cumulative yield for the process
Defect Injection Rates (actual only)	Defects injected per hour for each phase
Defect Removal Rates (plan and actual)	Defects removed per hour for each phase

Controls

The SUMQ worksheet has one control, the Assembly Selector Control, which selects the assembly to display on the SUMP worksheet. The list of possible selections includes all the assemblies on worksheet SUMS and the predefined assembly SYSTEM.



[<- The SUMP Worksheet](#)

The SUMQ Worksheet

[The SUMS Worksheet ->](#)

[Contents](#) [List of Forms](#) [Tool Instructions](#)

The SUMS Worksheet

Overview

The SUMS worksheet implements the TSPi SUMS Size Summary form.

SUMS contains the plan and actual size for each assembly and part. Size estimates and the measured size of each part are entered on SUMS

Fields

The following table describes the data that are entered on the SUMS worksheet.

Data	Contents	Default Value or Calculated Value
Part ID	A unique ID number for each part and assembly	Calculated
Part Name	The name of the part or assembly	
Assembly or Part	A = assembly P = part	Defaults to A
Part of	The parent assembly of this part or assembly	
Owner	Initials of product owner	
Size Measure	The size measure unit for the part or assembly	Valid values are • Text pages • Req pages • HLD pages • DLD lines • LOC
Planned Size	Planned Base, Deleted, Modified, Added, and Reused Size are entered.	Planned New and Changed and Total Size are calculated.
Actual Size	Actual Base, Deleted, Modified, Added, and Reused Size are entered. Actual New and Changed and Total Size are calculated.	

<- The SUMQ Worksheet

The SUMS Worksheet

Contents List of Forms Tool Instructions