Software Engineering Institute

Integrating CMMI and TSP/PSP: Using TSP Data to Create Process Performance Models

Shurei Tamura

November 2009

TECHNICAL NOTE
CMU/SEI-2009-TN-033

Software Engineering Measurement and Analysis

http://www.sei.cmu.edu

Carnegie Mellon

This report was prepared for the

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

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

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

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

Table of Contents

List of Figures

List of Tables

Acknowledgments

I want to thank the following individuals for their contributions to the technical content of this report: Robert W. Stoddard, David Zubrow, Mike Konrad, William Nichols, James McHale, James McCurley, and Rusty Young of the SEI, Jim Van Buren of the Charles Stark Draper Laboratory and David R. Webb of Hill Air Force Base. Erin Harper provided editorial assistance.

Abstract

This report describes the fundamental concepts of process performance models (PPMs) and describes how they can be created using data generated by projects following the Team Software Process (TSP). PPMs provide accurate predictions and identify factors that projects and organizations can control to better ensure successful outcomes, helping organizations move from a reactive mode to a proactive, anticipatory mode.

PPMs are fundamental to the implementation of the high maturity process areas of Capability Maturity Model® Integration and are specifically required in the Quantitative Project Management and Organizational Process Performance process areas. The three examples in this report demonstrate how data generated from projects using TSP can be combined with data from other sources to produce effective PPMs.

1 Introduction

Capability Maturity Model® Integration (CMMI®) is a process improvement framework for the development of products and services [Chrissis 07]. It consists of practices that address development and maintenance activities for the entire product life cycle, from conception through delivery and maintenance. It also contains practices to be implemented at the organizational and project levels for purposes of institutionalization and improvement.

The Team Software Process℠ (TSP℠) and the co-requisite Personal Software Process℠ (PSP℠) define a set of project practices that have been shown to produce highly desirable process performance results in terms of delivered product quality, schedule performance, and cost performance [Webb 08]. TSP and PSP provide team- and individual-oriented principles along with a specific, well-defined methodology for planning and executing software projects. A fundamental component of working on a TSP team is the collection and use of detailed measures of size, defects, effort, schedule, and rework. These measures provide valuable insight into project performance and status and serve as a basis for predicting several aspects of project completion.

The development and use of process performance models (PPMs) are identified as high maturity practices in CMMI, primarily in the process areas of Organizational Process Performance (OPP) and Quantitative Project Management (QPM). High quality data are required to develop and use PPMs effectively. The practices and procedures built into TSP for collecting well-defined, fine-grained data about process execution and product quality also result in data that serves as an excellent basis for the development and use of PPMs.

This technical note offers a description of the fundamental concepts of PPMs and the connections between them and TSP. Examples of PPMs created using data from TSP teams are provided to illustrate the concepts and highlight the connections.

2 Process Performance Models

2.1 What is a Process Performance Model?

A process performance model (PPM) is a description of the relationships among attributes of a process and its work products. PPMs are developed from historical process performance data and calibrated using collected process and product measures from the project. They can be used to predict results that will be achieved by following a process [Chrissis 07].

Process performance models

- predict future outcomes based on possible or actual changes to factors (e.g., they support "what-if" analyses)
- relate the behavior or circumstance of a process or subprocess, represented as a controllable or uncontrollable factor, to an outcome
- use factors from one or more upstream processes or subprocesses to predict downstream outcomes
- use controllable factors so projects can take action to influence outcomes (preferable)
- are statistical or probabilistic in nature rather than deterministic (e.g., models account for statistical variation and depict the uncertainty in the factors and the uncertainty or range of values in the outcome)

PPMs are useful tools for project and process management. Project managers use them to predict process performance with a known level of confidence so they can better identify and understand risks. PPMs should include at least one controllable factor so managers can perform "what-if" analyses to see what impact various courses of action might have on their projects. They can be used in a similar way for process improvement at the organizational level. When composing a project's defined process, PPMs play an important role in analyzing whether that process will be able to meet the project's quality and process performance objectives.

In the context of process management, PPMs help organizations identify and leverage important relationships among process factors and outcomes. They also provide insight into the actual and expected magnitude of variation associated with the use of a process, and they can be used to estimate the effects of alternative process changes.

2.2 Creating Process Performance Models

A number of conditions in an organization influence the nature and value associated with a PPM. Each of the following is an important input to the process of creating a PPM:

- the organizational set of standard processes and tailoring guidelines that support the establishment and maintenance of project-defined processes
- high-quality, historical process performance data
- knowledge and skills related to process performance modeling
- high-fidelity data collection, analysis, and reporting processes

- tools and techniques for developing PPMs (e.g., statistical models and Monte Carlo simulation)

As shown in Figure 1, a data repository that includes TSP data provides the information needed to create PPMs. TSP teams define the project processes that specify how and when data are gathered, and TSP team members—trained in PSP—understand and use data for planning and performing their work. Therefore, TSP teams can provide high quality, fine-grained data which can be an excellent basis for creating and using PPMs.

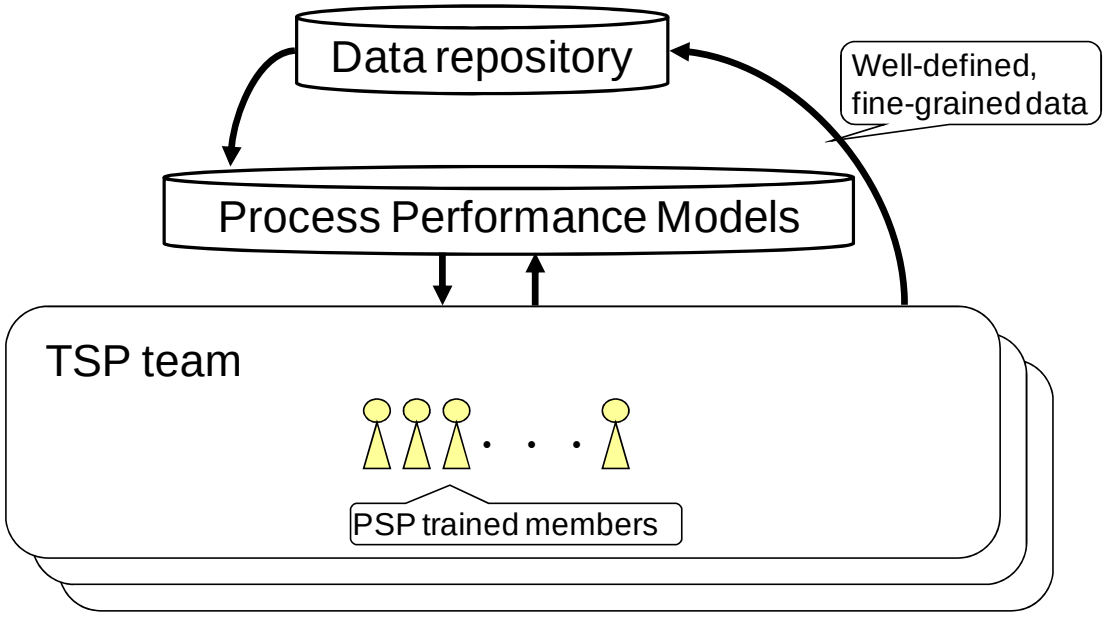

Figure 1: Process Performance Models and TSP

The principle steps used to create PPMs are described below [Zubrow 09].

1. Identify or reconfirm the project's goals of interest. The process outcome to be modeled should be aligned with the project's quality and process performance objectives.
2. Select the process or processes to analyze and model. This can be a project or an organizational process. The model can predict outcomes associated directly with the process under investigation or an outcome further downstream in the development life cycle.
3. Decide what outcome to predict. Some examples include quality, productivity, cycle time, process effectiveness, and customer satisfaction. Note that processes may have more than one type of outcome and that multiple models may be developed to address them. For instance, duration, quality, and effort consumed may all be relevant process performance outcomes.
4. Hypothesize factors to investigate in the PPM. Controllable and uncontrollable factors should be included. Root causes of outcomes, factors correlated with outcomes, and leading indicators of outcomes are good candidates for analysis.
5. Select the modeling techniques to use. This decision is driven by the measurement scale of the available data and the type of model being created. Statistical models such as regression analysis, analysis of variance (ANOVA), logistic regression, and logit analysis can be used

to create PPMs. Probabilistic modeling approaches such as Monte Carlo simulation and discrete event simulation can be also used.

6. Obtain relevant data, evaluate its quality, and document its limitations. Make decisions about how and what to sample, the conceptual validity of the data, and the reliability of the measurement process.
7. Establish statistical and business criteria for evaluating the performance or utility of the model. While statistical analysis often produces measures of how well the model fits the data, these measures do not always reflect performance in business terms. Both types of criteria should be set and considered.
8. Fit the model to the data and evaluate the result against the statistical and business criteria.
9. If the result is satisfactory, deploy the model and periodically review it in terms of its value and utility to the organization.

2.3 Relationship of CMMI Level 4 and 5 Process Areas to Process Performance Models

This section describes the relationships of four CMMI Level 4 and 5 process areas to PPMs.

Organizational Process Performance

The purpose of Organizational Process Performance (OPP) is to establish and maintain a quantitative understanding of the performance of the organization's set of standard processes in support of quality and process performance objectives and to provide the process performance data, baselines, and models to quantitatively manage the organization's projects (see Figure 2). Process performance models are established based on the organization's set of standard processes and the organization's process performance baselines (OPP SP 1.5). The PPMs can be used to establish or verify the reasonableness of the quantitative objectives for quality and process performance for the organization (OPP SP 1.3). As depicted, there are strong connections between OPP and the other high maturity process areas (QPM, CAR, and OID).

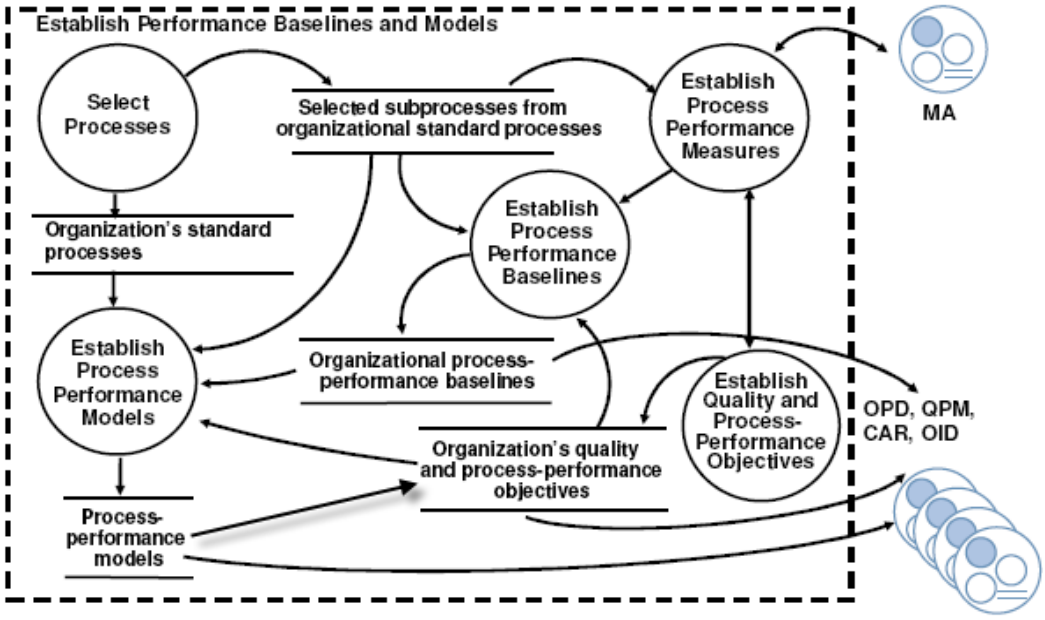

Figure 2: OPP Goal and Practices

Quantitative Project Management

The purpose of Quantitative Project Management (QPM) is to quantitatively manage the project's defined process to achieve the project's established quality and process performance objectives. This process area applies quantitative and statistical techniques to do tasks such as the following:

- evaluate candidate compositions of the project's defined process (QPM SP 1.2)
- establish or verify the reasonableness of the project's quality and process performance objectives (QPM SP 1.1)
- estimate progress toward achieving the project's quality and process performance objectives (QPM SP 1.4 Sub 4)

Causal Analysis and Resolution

The Causal Analysis and Resolution (CAR) process area provides the mechanism for the organization and projects to identify root causes of selected defects and other problems and take action to prevent them from occurring in the future.

This process area applies quantitative and statistical techniques to do tasks such as the following:

- select the defects and other problems for analysis by aiding impact, benefit, and return on investment predictions (CAR SP 1.1)
- identify potential sources of the defects and other problems (CAR SP 1.2)
- select action proposals for implementation by aiding impact, benefit, and return on investment (ROI) predictions (CAR SP 2.1)
- evaluate the effects of changes on process performance to see if it meets predicted performance (CAR SP 2.2)

Organizational Innovation and Deployment

The Organizational Innovation and Deployment (OID) process area is used to select and deploy proposed incremental and innovative improvements that improve the organization's ability to meet its quality and process performance objectives.

This process area applies quantitative and statistical techniques to do tasks such the following:

- analyze the costs and benefits of process- and technology-improvement proposals (OID SP 1.1 Sub 2)
- analyze potential innovative improvements to understand their effects on process elements and predict their influence on the process (OID SP 1.2 Sub 3)
- determine whether the ability of the defined process to meet quality and process performance objectives is adversely affected by changes to the process (OID SP 2.2 Sub 9)
- analyze the progress of the deployed process and technology improvements toward achieving the organization's quality and process performance objectives to determine whether the predicted impacts are achieved (OID SP 2.3 Sub 4)

3 Examples of Process Performance Models

This section presents examples of PPMs that could be created using data from TSP teams. In the examples, additional variables are included to demonstrate how data generated from TSP teams can be combined with data from other sources to produce effective PPMs. The models and their uses are described in the context of high maturity process areas.

The first two examples illustrate applications within QPM. The first involves tracking against a project's quality and process performance objectives and the second involves composing the project's defined process. The third example describes a case of problem solving that might be done within the context of CAR or OID. Example outcomes and factors that might be included in PPMs, which are likely to be measured in organizations that have applied TSP, are described in Appendix A.

3.1 Example 1: QPM – Meeting the Project Objective for Product Quality

This example illustrates the use of a PPM to track against a project's quality and process performance objectives. In this example, an organization analyzed historical TSP team data to develop a PPM predicting defect density in system test. The controllable x factors included in the model were

- prototype: dummy variable reflecting if a prototype will be developed (1=Yes, 0=No)
- requirements inspection rate (pages/Hr)

Defect density in system test and the requirements inspection rate are recorded as part of routine TSP data collection, but use of a prototype is not. The development approach in TSP teams sometimes includes building a prototype, however.

Figure 3 shows the results of a dummy variable regression using the controllable factors to predict the outcome y, defect density (Defects/KLOC), in system test. The results indicate that the model provides a good fit to the data and that the included x factors are statistically significant. That is, the coefficient of determination (R-squared) is high, .828, and its corresponding p-value is below 0.05. Similarly, the p-value for each of the x factors is less than 0.05. Further investigation of the residuals provides confidence in the usability of the results. The adjusted R-squared of this model is higher than the adjusted R-squared of the model without the prototype. This means the inclusion of the prototype variable significantly improves the fit of the model.

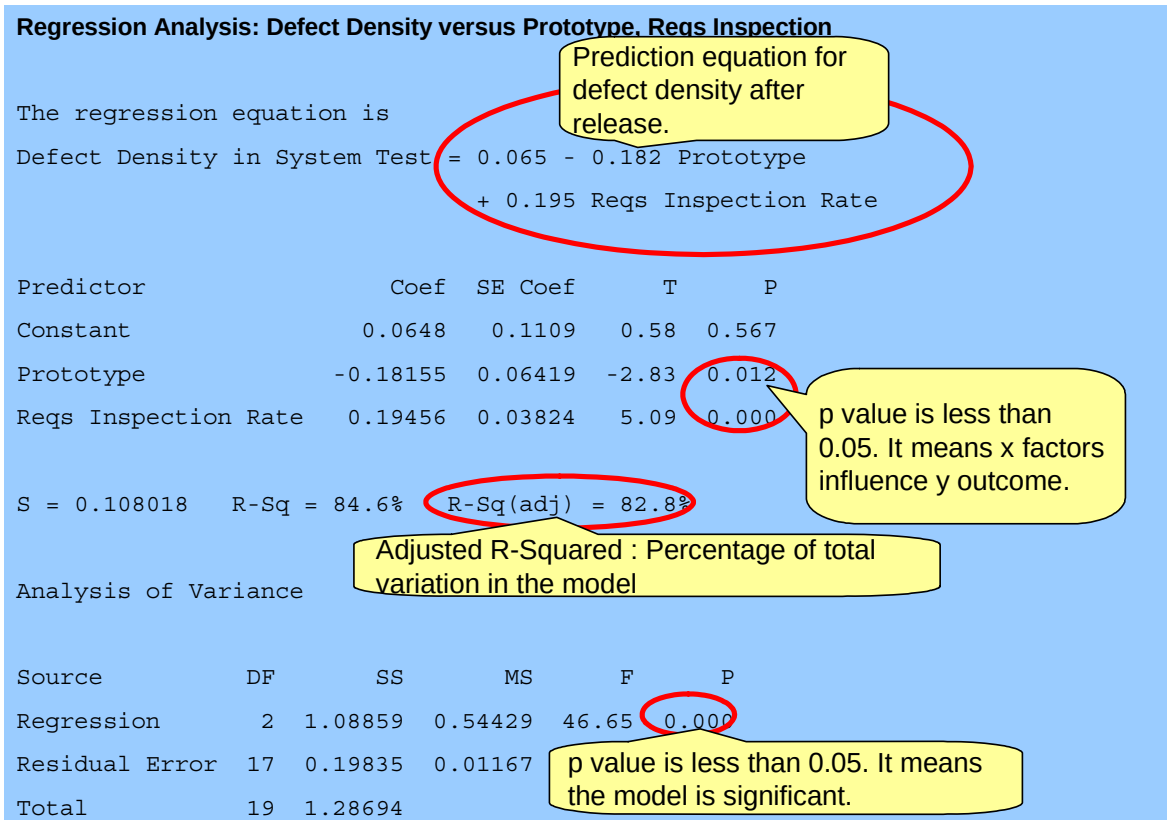

Figure 3: Defect Density vs. Prototype, Requirements Inspection Rate[1]

This model can be interpreted as follows:

- if prototyping is conducted, the estimated defect density in system test will decrease by 0.182 defects/KLOC, on average
- for the requirements inspection review rate, for each additional page that will be inspected per hour, the estimated rate of defect density in system test will increase by 0.195 defects/KLOC, on average

The use of this equation for an individual project would require the computation of a prediction interval. A prediction interval establishes the range within which a predicted value will occur with a stated degree of confidence or probability [Zubrow 09]. Prediction intervals are important because they reflect the underlying variation in performance and provide a manager with a realistic range for the expected outcome. This provides a manager with greater insight into risk and uncertainty than a simple point prediction. To minimize risk, the upper value of the prediction interval would be less than the target system test, defect density value.

3.1.1 Uses of This PPM

This model can be used at the following project stages:

[1] Minitab® statistical software was used to produce the charts and graphs in this section.

- before a TSP launch, to train team members to monitor their own review rate for requirements inspections
- during a TSP launch, to
  - establish or verify the feasibility of a target defect density in system test as the project's quality and process performance objective
  - decide if the project should conduct prototyping
  - decide or verify the target requirements inspection rate
  - plan for the needed number of requirements inspections, given the target review rate and the total number of pages of material to be reviewed
  - identify and evaluate risk if the project does not conduct prototyping
- during the development phase or cycle, to manage the performance of requirements inspection by adjusting the review rate in order to keep predicted defect density in system test at the targeted level
- during the project postmortem, to determine if the actual defect density in system test met the target

This demonstrates how routinely collected TSP data and additional measures of the development process like a prototype can be analyzed together to create an enriched PPM.

3.2 Example 2: QPM – Process Composition Regarding Review Type and Allowable Review Rate

This example illustrates an application of a PPM that could be created using data from TSP teams on the project's defined processes within QPM. In this example, each TSP team analyzed historical TSP team data to develop a PPM to establish a target code review rate. Code review yield (the percentage of defects present that are removed by the review) and code review rate (the number of lines of code reviewed per hour) are routinely recorded when TSP is being used. As shown in Figure 4, the linear regression provides information about how code review rate (the x controllable factor) predicts code review yield (the outcome). Furthermore, the statistical output provides information that can be used to create confidence and prediction intervals around the yield values. Knowledge of the underlying distribution and intervals around the mean can be valuable for planning and executing the project.

Note that the model makes explicit the tradeoff between quality (i.e., yield) and effort (i.e., review rate). In addition, training and review techniques such as using a review checklist are important in order to conduct effective reviews. Individuals who lack knowledge about how to conduct reviews often have a higher review rate. PSP training teaches effective review techniques. After training they read code or other review materials carefully and effectively, which takes more time but reduces escaped defects.

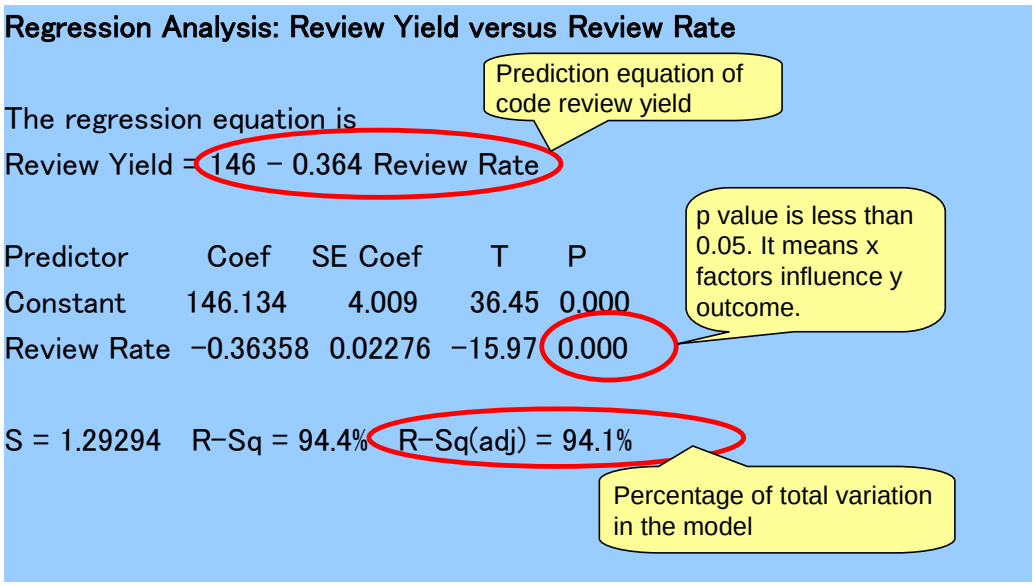

Figure 4: Code Review Rate vs. Review Yield

3.2.1 Uses of This PPM

This model can be used at the following project stages:

- before a TSP launch, to
  - train team members on the influence of review rate on yield
- during a TSP launch, to
  - establish a suitable code review rate
  - make a detailed plan for the type, number of code reviews, and the amount of material to be covered during a review
  - establish the timing and schedule for code reviews
- during the development phase or cycle, to
  - manage the performance of code reviews by using code review rate as an entry criterion for conducting code reviews
  - understand the implications for testing and rework as a result of violating the targeted review rate
- during the project postmortem, to
  - evaluate if the actual code review yield met the target
  - evaluate if the PPM continued to accurately predict yield

The data needed for the development of this model is routinely collected by TSP teams. Furthermore, TSP provides precise operational definitions for these measures. Providing clear data collection practices and standard definitions helps to increase the accuracy and reliability of the data gathered and, in turn, enhances the quality of the results that come from the analysis of the data. This reduces the variability and increases the confidence in the relationships identified as part of the PPM.

3.3 Example 3: CAR/OID – Process Improvement

In this example, an organization chose to improve its testing strategy by using and updating a PPM.

The organization had a PPM that predicted escaped unit test defects/KLOC. As shown in Figure 5, the linear regression provides information about how testing coverage, the "x" controllable factor, predicts integration test defects density, the outcome. The adjusted R-squared is high and statistically significant and the residuals are normally distributed and random. These results indicate the model is a good fit for the data.

Escaped unit test defects/KLOC is recorded as part of routine TSP data collection, but testing coverage is not. Test coverage in this case was measured as statement coverage, the percentage of lines of code that are executed during integration testing. While most TSP teams in the organization wanted to decrease their escaped unit test defects, the as-is process performance did not meet their objective. Through analysis of historical defect data, it was discovered that low integration testing coverage was associated with high escaped unit test defects. As a result of analyzing historical process improvement proposals from TSP teams and from surveys of effective testing techniques and methodologies, the organization decided to pilot a non-TSP testing approach, introducing the unit test support tool with functions creating test cases and measuring test coverage to improve test coverage [Glass 02].

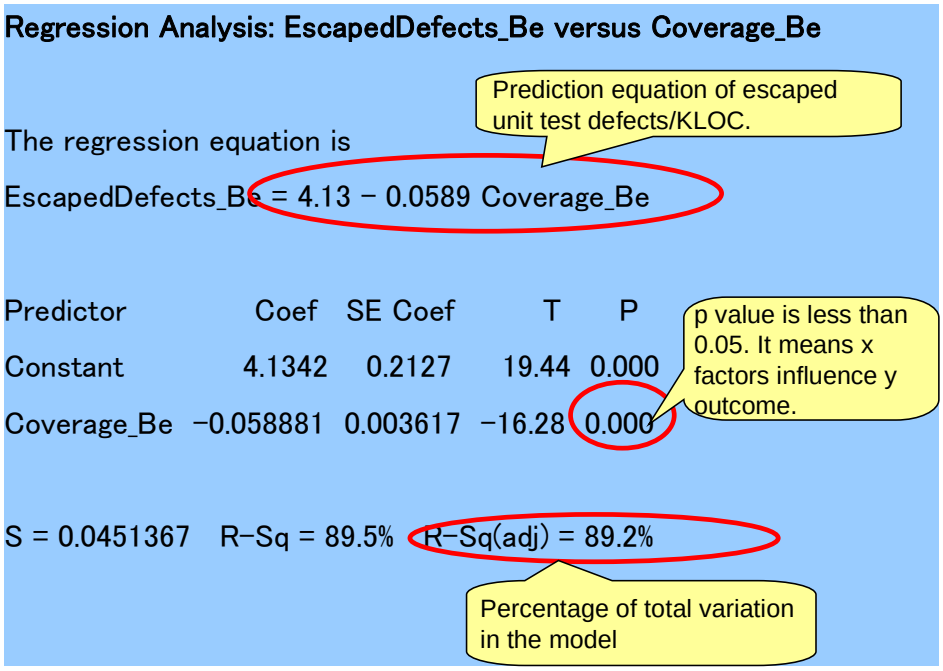

Figure 5: Escaped Unit Test Defects/KLOC vs. Coverage Before Pilot

Three pilot TSP teams with ten components used the unit test support tool. Figure 6 shows the results of the two-sample t-tests for comparing the means of escaped unit test defects/KLOC before pilot projects with projects in pilot. Each data set followed a normal distribution. The box plots from the two-sample t-test are shown in Figure 7. The null hypothesis was rejected and the organization concluded that using the unit test support tool did decrease escaped unit test de-

fects/KLOC to a level close to its objective. Based on these results, the decision was made to use the unit test support tool across the organization.

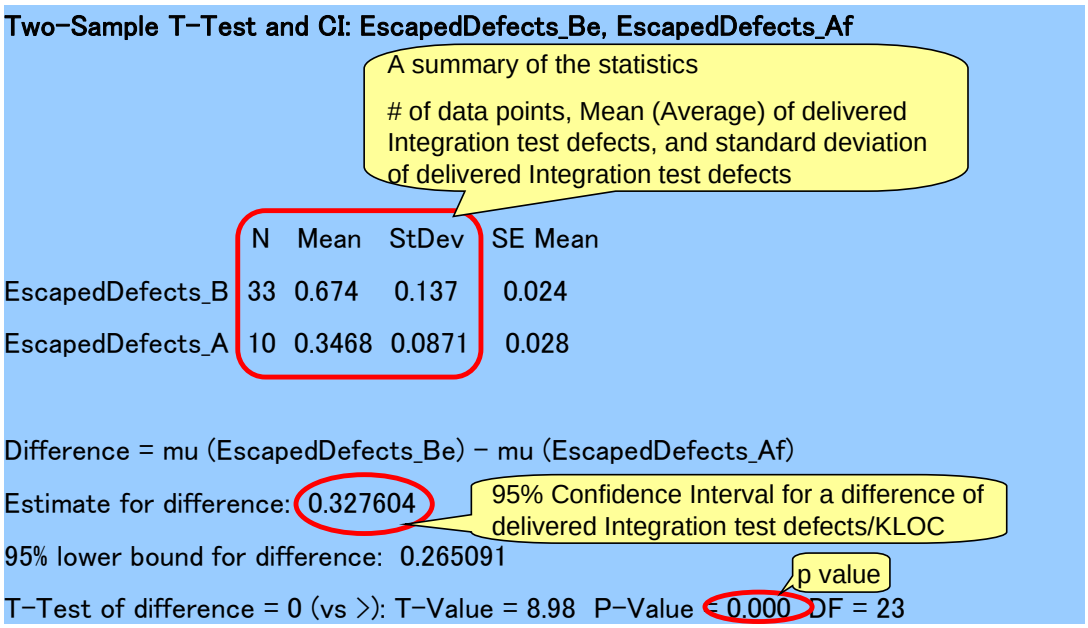

Figure 6: The Result of the Two Sample T-Test

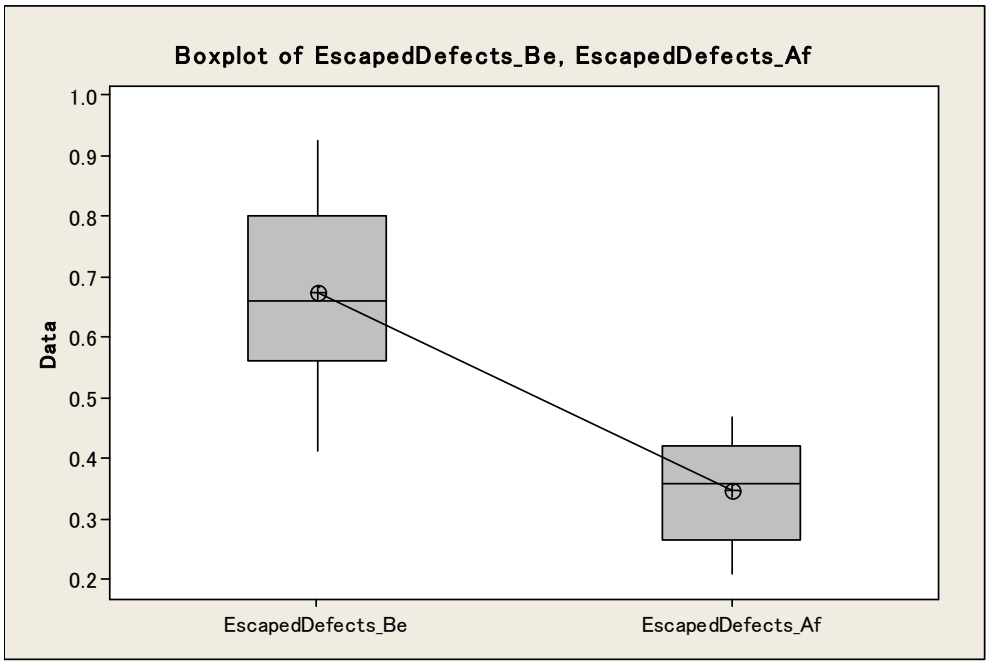

Figure 7: Box Plots from the Two Sample T-Test

As shown in Figure 8, the linear regression for predicting escaped unit test defects/KLOC is updated based on data from pilot projects. Again, the R-squared was high and the residuals were normally distributed and random. The model for the prediction of defect density after unit test

based on planned testing coverage provides a good fit to the data. This new model can be used to predict escaped unit test defects/KLOC in future TSP teams using the unit test support tool.

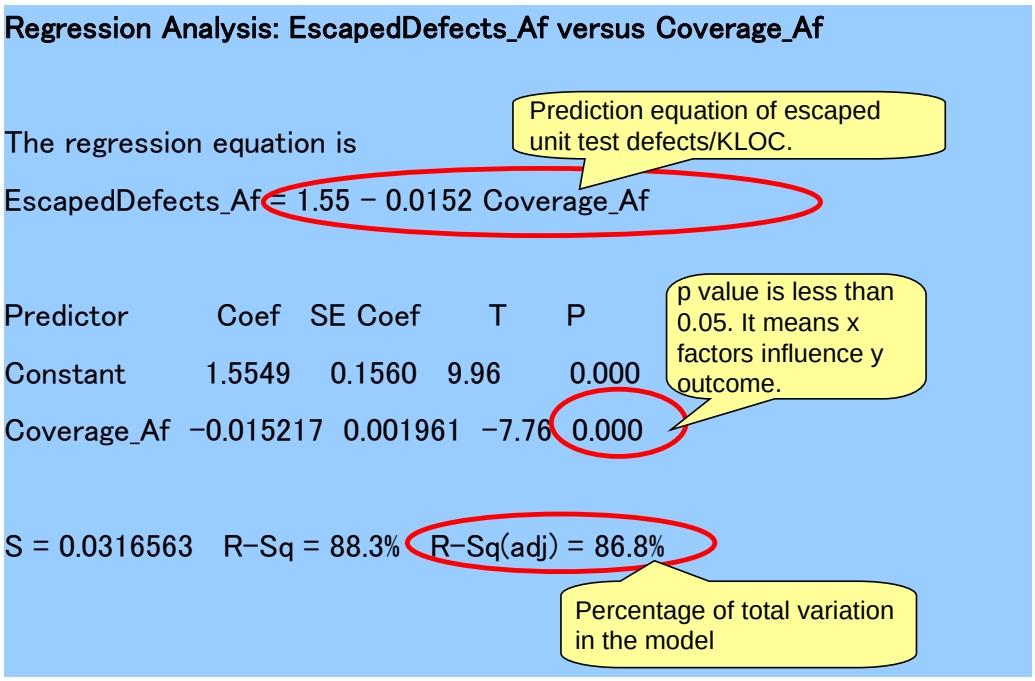

Figure 8: Escaped Unit Test Defects/KLOC vs. Coverage in Pilot Projects

Although using sound design methods reduces the defect injection rate and sound review and inspection methods remove defects before unit test, an effective testing strategy is still important to eliminate more defects before delivery. Testing is also necessary to verify the effectiveness of the upstream defect removal activities. That is, even if testing removes very few defects, it is necessary for verification. In fact, we have seen products developed with essentially no defects removed in system test. Only confidence in the effectiveness of system test provides confidence that defects will not be found by the end user.

3.3.1 Uses of This PPM

After establishing a model through pilots it can be used during the following project stages:

- before a TSP launch, to train team members on the importance of test coverage and the new unit testing strategy
- during a TSP launch, to
  - plan the testing strategy to achieve the target test coverage
  - investigate tradeoffs between test coverage and unit test defect density
- during the development phase or cycle, to
  - monitor the impacts of actual test coverage achieved
  - identify future defect density and its implications for downstream quality and activities based on expected test coverage to be achieved
- during the project postmortem, to evaluate and validate if the PPM continued to accurately predict escaped unit test defects/KLOC

4 Summary

TSP teams collect and use detailed measures of size, defects, effort, schedule, and rework. These measures provide valuable insight into project performance and status and serve as a basis for predicting several aspects of project completion. Organizations using TSP can use the data they collect, along with additional variables not recorded as part of routine TSP data collection, to produce highly effective PPMs.

The examples in this report show that organizations implementing TSP already have the data they need to create PPMs. The first two examples illustrate applications within QPM: tracking against a project's quality and process performance objectives and composing the project's defined process. The third example describes a case of problem solving that might be done in the context of Causal Analysis and Resolution (CAR) or Organization Innovation and Development (OID). TSP teams can use the PPMs they create to enrich their quantitative management and implement the high maturity process areas of CMMI.

Appendix A: Example Outcomes and Factors

This section provides example outcomes and factors that can be used to create PPMs in organizations applying TSP. The tables list outcome data and factors currently collected by TSP and other factors that could be collected by the organization to create a PPM. As mentioned in Section 2.1, some of the factors of the PPM should be controllable, with uncontrollable factors working as constraints.

These examples assume the TSP team uses the typical TSP development process illustrated below in Figure 9 and that the principal work products in each development phase are created, shown in Table 1.

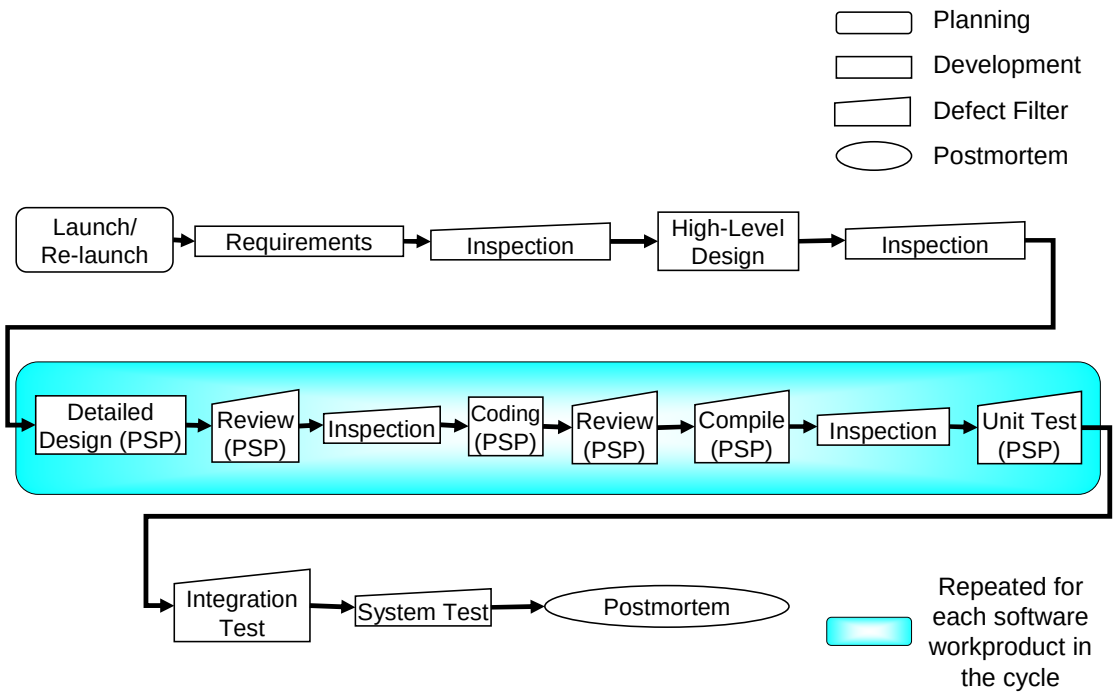

Figure 9: Development Process

Table 1: Principal Work Product in Each Development Phase

Development phase	**Principle work products**
Requirements	Requirements documents
High-level design	High-level design documents
Detailed design	Detailed design documents
Coding	Source code

A.1 Phase Yield

Factors in ***red italics*** are routinely collected and used by TSP/PSP. Factors in black are other possible controllable factors.

y: outcome	requirements inspections yield
x: factors	***requirements inspection rate, requirements inspection/requirements time*** domain, requirements volatility, quality attributes, readability of documents, domain experiences, requirements methods and tools, requirements inspection checklist, requirements skills and experiences with methods and tools used, requirements inspection skills and experiences, quality of reused requirements documents
process/subprocess	requirement, requirements inspection

y: outcome	code review yield
x: factors	***requirement inspections rate, high-level design inspections rate, detailed design review rate, detailed design inspection rate, code review rate, code review/coding time*** code complexity, encapsulation, program language & tools, code review checklist, coding skills and experiences with the program languages and tools used, code review skills and experiences, quality of reused source code
process/subprocess	requirements, requirements inspection, high-level design inspection, detailed design review, detailed design inspection, coding, code review

y: outcome	integration test yield
x: factors	***requirement inspections rate, high-level design inspections rate, detailed design review rate, detailed design inspection rate, code review rate, code inspection rate, # of test cases in unit test, # of test cases in integration test*** requirements volatility, integration test methods and tools, Integration test skills and experiences with methods and tools used, quality of reused test cases
process/subprocess	requirements, requirements inspection, high-level design, high-level design inspection, detailed design, detailed design review, detailed design inspection, coding, code review, compile, code inspection, unit test, integration test

A.2 Process Yield

y: outcome	yield before unit test
x: factors	***requirement inspections rate, high-level design inspections rate, detailed design review rate, detailed design inspection rate, code review rate*** domain, requirements volatility, quality attributes, readability of documents, architecture measures, code complexity, encapsulation, requirements methods and tools, requirements inspection checklist, high-level design methods and tools, high-level design inspection checklist, detailed design methods and tools, detailed design review/inspection checklist, program language & tools, code review/inspection checklist, domain experiences, requirements skills and experiences with methods and tools used, requirement inspection skills and experiences, high-level design skills and experiences with the methods and tools used, high-level design inspection skills and experiences, detailed design skills and experiences with the methods and tools used, detailed design review/inspection skills and experiences, coding skills and experiences with the program languages and tools used, code review/inspection skills and experiences, quality of reused principal work products
process/subprocess	requirements, requirements inspection, high-level design, high-level design inspection, detailed design, detailed design review, detailed design inspection, coding, code review, compile, code inspection

A.3 Defect Density

y: outcome	defects/ KLOC in compile
x: factors	***requirement inspections rate, high-level design inspections rate, detailed design review rate, detailed design inspection rate, code review rate,*** domain, requirements volatility, quality attributes, readability of documents, architecture measures, code complexity, encapsulation, requirements methods and tools, requirements inspection checklist, high-level design methods and tools, high-level design inspection checklist, detailed design methods and tools, detailed design review/inspection checklist, program language & tools, code review checklist, domain experiences, requirements skills and experiences with methods and tools used, requirement inspection skills and experiences, high-level design skills and experiences with the methods and tools used, high-level design inspection skills and experiences, detailed design skills and experiences with the methods and tools used, detailed design review/inspection skills and experiences, coding skills and experiences with the program languages and tools used, code review skills and experiences, quality of reused principal work products
process/subprocess	requirements, requirements inspection, high-level design, high-level design inspection, detailed design, detailed design review, detailed design inspection, coding, code review, compile

y: outcome	defects/ KLOC in integration test
x: factors	***requirement inspections rate, high-level design inspections rate, detailed design review rate, detailed design inspection rate, code review rate, # of test cases in unit test, # of test cases in integration test*** domain, requirements volatility, quality attributes, readability of documents, architecture measures, code complexity, encapsulation, requirements methods and tools, requirements inspection checklist, high-level design methods and tools, high-level design inspection checklist, detailed design methods and tools, detailed design review/inspection checklist, program language & tools, code review checklist, unit test methods and tools, unit test methods and tools, integration test methods and tools, domain experiences, requirements skills and experiences with methods and tools used, requirement inspection skills and experiences, high-level design skills and experiences with the methods and tools used, high-level design inspection skills and experiences, detailed design skills and experiences with the methods and tools used, detailed design review/inspection skills and experiences, coding skills and experiences with the program languages and tools used, code review/inspection skills and experiences, unit test skills and experiences with the methods and tools used, integration test skills and experiences with the methods and tools used, quality of reused principal work products
process/subprocess	requirements, requirements inspection, high-level design, high-level design inspection, detailed design, detailed design review, detailed design inspection, coding, code review, compile, code inspection, unit test, integration test

y: outcome	defect/KLOC in system test
x: factors	***requirement inspections rate, high-level design inspections rate, detailed design review rate, detailed design inspection rate, code review rate, # of test cases in unit test, # of test cases in integration test, # of test cases in system test*** domain, requirements volatility, quality attributes, readability of documents, architecture measures, code complexity, encapsulation, requirements methods and tools, requirements inspection checklist, high-level design methods and tools, high-level design inspection checklist, detailed design methods and tools, detailed design review/inspection skills and experiences, program language & tools, code review checklist, unit test methods and tools, integration test methods and tools, system test methods and tools domain experiences, requirements skills and experiences with methods and tools used, requirement inspection skills and experiences, high-level design skills and experiences with the methods and tools used, high-level design inspection skills and experiences, detailed design skills and experiences with the methods and tools used, detailed design review/inspection skills and experiences, coding skills and experiences with the program languages and tools used, code review/inspection skills and experiences, unit test skills and experiences with the methods and tools used, integration test skills and experiences with the methods and tools used, system test skills and experiences with the methods and tools used, quality of reused principal work products

process/subprocess	requirements, requirements inspection, high-level design, high-level design inspection, detailed design, detailed design review, detailed design inspection, coding, code review, compile, code inspection, unit test, integration n test, system test

y: outcome	total defects/KLOC
x: factors	***requirement inspections rate, high-level design inspections rate, detailed design review rate, detailed design inspection rate, code review rate, # of test cases in unit test, # of test cases in integration test, # of test cases in system test*** domain, requirements volatility, quality attributes, readability of documents, architecture measures, code complexity, encapsulation, requirements methods and tools, requirements inspection checklist, high-level design methods and tools, high-level design inspection checklist, detailed design methods and tools, detailed design review/inspection skills and experiences, program language & tools, code review checklist, unit test methods and tools, integration test methods and tools, system test methods and tools domain experiences, requirements skills and experiences with methods and tools used, requirement inspection skills and experiences, high-level design skills and experiences with the methods and tools used, high-level design inspection skills and experiences, detailed design skills and experiences with the methods and tools used, detailed design review/inspection skills and experiences, coding skills and experiences with the program languages and tools used, code review/inspection skills and experiences, unit test skills and experiences with the methods and tools used, integration test skills and experiences with the methods and tools used, system test skills and experiences with the methods and tools used, quality of reused principal work products
process/subprocess	requirements, requirements inspection, high-level design, high-level design inspection, detailed design, detailed design review, detailed design inspection, coding, code review, compile, code inspection, unit test, integration test, system test

A.4 Effort

y: outcome	task's effort in high-level design
x: factors	***size of high-level design documents, productivity in high-level design (e.g. size/hr)*** domain, architecture measures, high-level design methods and tools, high-level design skills and experiences with methods and tools used, quality of reused high-level design documents
process/subprocess	task in high-level design

y: outcome	task's effort in high-level design inspection

x: factors	***size of high-level design documents, high-level design inspection rate, # of inspectors in high-level design inspection, # of defects removed in high-level design inspection*** domain, architecture measures, high-level design methods and tools, high-level design inspection checklist, high-level design skills and experiences with methods and tools used, high-level design inspection skills and experiences, quality of reused high-level design documents
process/subprocess	task in detailed design

y: outcome	total effort
x: factors	***each task's effort***
process/subprocess	all processes

A.5 Schedule

y: outcome	schedule in phase, cycle, or milestone
x: factors	***each task's effort in phase, cycle or milestone, resource availability***
process/subprocess	tasks in phase, cycle or milestone

y: outcome	total schedule variance
x: factors	***schedule in each phase, cycle, or milestone***
process/subprocess	all processes

See definitions of TSP/PSP measures in *PSP: A Self-Improvement Process for Software Engineers* [Humphrey 05].

References/Bibliography

URLs are valid as of the publication date of this document.

[Chrissis 07]
Chrissis, Mary Beth; Konrad, Mike; & Shrum, Sandy. *CMMI: Guidelines for Process Integration and Product Improvement, 2nd Edition*. Addison-Wesley, 2003. http://www.sei.cmu.edu/library/abstracts/books/0321279670.cfm

[Glass 02]
Glass, Robert L. *Facts and Fallacies of Software Engineering*. Addison-Wesley, 2002.

[Humphrey 00]
Humphrey, Watts S. *The Personal Software Process (PSP)* (CMU/SEI-2000-TR-022, ADA387268). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. http://www.sei.cmu.edu/library/abstracts/reports/00tr022.cfm

[Humphrey 05]
Humphrey, Watts S. *PSP: A Self-Improvement Process for Software Engineers*. Addison-Wesley, 2005. http://www.sei.cmu.edu/library/abstracts/books/0321305493.cfm

[Humphrey 06a]
Humphrey, W. S. *TSP: Leading a Development Team*. Addison-Wesley, 2006. http://www.sei.cmu.edu/library/abstracts/books/0321349628.cfm

[Humphrey 06b]
Humphrey, W. S. *TSP: Coaching Development Teams*. Addison-Wesley, 2006. http://www.sei.cmu.edu/library/abstracts/books/201731134.cfm

[McHale 05]
McHale, James & Wall, Daniel S. *Mapping TSP to CMMI* (CMU/SEI-2004-TR-014). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/library/abstracts/reports/04tr014.cfm

[O'Toole 07a]
O'Toole, Pat. *Interpretation Issues (ATLAS #7).* 2007. http://www.pactcmmi.com/pages/atlas/index.html

[O'Toole 07b]
O'Toole, Pat. *TSP Adoption Issues (ATLAS #11).* 2007. http://www.pactcmmi.com/pages/atlas/index.html

[Stoddard 07]
Stoddard, Robert W.; Linders, Ben; & Sapp, Millee. *Exactly What are Process Performance Models in the CMMI?* SEPG Europe Conference, 2007.

[Wall 07]
Wall, Daniel S.; McHale, James; & Pomeroy-Huff, Marsha. *Case Study: Accelerating Process Improvement by Integrating the TSP and CMMI* (CMU/SEI-2007-TR-013). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2007. http://www.sei.cmu.edu/library/abstracts/reports/07tr013.cfm

[Webb 08]
Webb, David; Van Buren, Jim; Tuma, David; & Stoddard, Robert. *Uses of Monte Carlo Simulation for TSP Teams*. TSP Symposium, 2008.

[Young 08]
Young, Rawdon; Stoddard, Bob; & Konrad, Mike. *If You're Living the "High Life," You're Living the Informative Material*. SEPG North America Conference, 2008.

[Zubrow 09]
Zubrow, Dave; Stoddard, Bob; Young, Rawdon; & Schaaf, Kevin. *A Practical Approach for Building CMMI Process Performance Models.* SEPG North America Conference, 2009.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

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

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Integrating CMMI and TSP/PSP: Using TSP Data to Create Process Performance Models	FA8721-05-C-0003

6. AUTHOR(S)

Shurei Tamura

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2009-TN-033

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

11. SUPPLEMENTARY NOTES

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

13. ABSTRACT (MAXIMUM 200 WORDS)

This report describes the fundamental concepts of process performance models (PPMs) and de-scribes how they can be created using data generated by projects following the Team Software Process (TSP). PPMs provide accurate predictions and identify factors that projects and organizations can control to better ensure successful outcomes, helping organizations move from a reactive mode to a proactive, anticipatory mode.

PPMs are fundamental to the implementation of the high maturity process areas of Capability Maturity Model® Integration and are specifically required in the Quantitative Project Management and Organizational Process Performance process areas. The three examples in this report demonstrate how data generated from projects using TSP can be combined with data from other sources to produce effective PPMs.

14. SUBJECT TERMS	15. NUMBER OF PAGES
TSP, CMMI, process performance models, PPM	35

16. PRICE CODE

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