Software Engineering Institute

The Personal Software Process[SM] (PSP[SM]) Body of Knowledge, Version 2.0

Marsha Pomeroy-Huff
Robert Cannon
Timothy A. Chick
Julia Mullaney
William Nichols

August 2009

SPECIAL REPORT
CMU/SEI-2009-SR-018

Software Engineering Process Management Program

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

About This Report

The intent of the Personal Software Process[SM] (PSP[SM]) Body of Knowledge (BOK) contained in this report is to provide guidance to software professionals who are interested in using proven-effective, disciplined methods to improve their personal software development process. However, it is also of interest to individuals who do not develop software but work with or manage projects involving many other kinds of development. The PSP BOK can aid these individuals in determining the knowledge and skills that most professionals should possess when working in a self-directed teaming environment such the Team Software Process[SM] (TSP[SM]). Development professionals who will find the PSP BOK to be useful include but are not limited to

- senior executives of software development organizations or of companies who use software as a component in their products
- program and project managers
- members of integrated product-development teams
- professionals who give support to software and other development projects (for example, systems engineers, testers, quality assurance specialists, and technical writers)
- customers and stakeholders
- process improvement consultants

The PSP BOK can also be used by education professionals in creating or assessing instructional products, from individual courses to entire curricula. For example, it can be used by professors or course designers to ensure that the content of new courses enables students to master the knowledge and skills of each competency area or to examine existing courses and evaluate them on their coverage of the requisite competencies.

Similarly, this document can be used to create, assess, or accredit certifications or other credentials programs for PSP practitioners. Certification programs provide individuals with documentation attesting that those individuals have attained a well-defined and objectively-measured level of proficiency in a particular field or discipline, as defined by a core set of knowledge and skills. Individuals who successfully demonstrate their proficiency—usually measured by performance on a standardized assessment instrument, and recognized by awarding of a credential or certification—are regarded as competent, skilled professionals with a demonstrated level of mastery in the competencies delineated by their profession's body of knowledge.

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

Acknowledgments

In preparing this report, the authors consulted with several individuals who provided ideas and contributions to the content of the PSP BOK. In particular, we want to acknowledge our version 1.0 co-authors, Mark Seburn, Julia Mullaney, and Robert Cannon, who provided review comments for this version. Thanks to Jefferson Welch and Alan Willett for helping us out when we got stuck for inspiration, words, and/or time (sometimes all at the same moment). Our editors in SEI Technical Communications were instrumental in catching the typos, grammar errors, and other defects that eluded us, and we appreciate their willingness to do the other tedious chores (such as document formatting) that are necessary to make the document visually appealing. Other individuals also contributed to reviewing the content and clarity of the report, and we would like to thank these individuals for their time and assistance: Yoshi Akiyama, Kimberly Campbell, David Carrington, Noopur Davis, Clare Dixon, Caroline Graettinger, Tom Hilburn, Watts Humphrey, Susan Kushner, Jim Over, Hans-Peter Pfister, Mary Ellen Rich, Jim Van Buren, and Alan Willett.

Finally, the authors would like to honor the ancient tradition of "saving the best for last" by ending this section in acknowledging the enormous contributions of Watts Humphrey, not only to this team and to this effort, but to the entire field of software engineering. It is fitting that as Version 1.0 of this document was being completed in 2005, Watts received word that he had been awarded the National Medal of Technology, which is given to America's leading innovators by the President of the United States. This award is appropriate recognition for a lifetime of effort aimed at improving the software development process. The Capability Maturity Model® (CMM®) has helped many companies to achieve consistent excellence at the organizational level, and the culmination of Watts's life work – the PSP and TSP methodologies – have given that same capability to individuals and teams. Those of us who use these methods thank Watts for these innovations, which have improved not only the quality of our work and our schedule- and effort-estimating abilities, but also the quality of our work life in general.

Foreword

What do we want from software engineering? That, of course, depends on who is asking. Just about everyone wants quality software on predictable schedules and for committed costs. Those of us in the development business also want a fun job, a rewarding development experience, and the satisfaction of doing professional work. Managers and users want development professionals who are credible and professional and can be trusted to do what they commit to do. Educators are most concerned with preparing student developers to do the kinds of work that society needs.

While the software business has been troubled almost from the outset, with unpredictable schedules, missed commitments, and poor-quality products, it has also produced some remarkable innovations. Software has supported and enabled many, if not most, of the modern advances in science and technology. Software is a truly extraordinary technology. It controls, guides, or enables just about every product, tool, or support system in our modern world. Just about everything that people now do depends to some degree on the effective performance of software. Software is central to our lives, to our businesses, and, in a growing number of cases, to our very survival. It is therefore critically important that software professionals learn and consistently use the best possible methods when they do their jobs.

This body of knowledge encapsulates the basic knowledge and principles behind the PSP. While the PSP is almost certainly not the last word in software development practice, it was developed from the basic principles of science and engineering, and when developers truly follow these principles in their work, they produce quality products on predictable schedules and for their committed costs. This may sound too good to be true, but it is not really the PSP that does it. The PSP is so effective because software professionals are such extraordinarily capable people. Since the PSP is a set of practices and methods that enable software developers to control their own working lives, when competent professionals learn and consistently follow these engineering and scientific principles, and when they are empowered to manage their own work, they do an unbelievably good job.

This BOK describes the basic PSP practices and methods. From this foundation, it is our hope that the software community will develop the courses, the support tools and methods, the certification and qualification programs, and all of the other required elements to enable the widespread adoption of these methods.

Watts S. Humphrey
June 2009

Abstract

As the profession of software engineering evolves and matures, it must achieve some of the critical elements needed for recognition as a bona fide discipline. Among these elements are the establishment of a recognized body of knowledge (BOK) and certification of professional practitioners.

The body of knowledge contained in this report is designed to complement the IEEE Computer Society's *Software Engineering Body of Knowledge (SWEBOK)* [IEEE 04] by delineating the skills and concepts that compose the knowledge areas and competencies of a proven-effective process improvement method, the Personal Software Process (PSP). As adoption of the PSP methodology continues to grow, it becomes crucial to document the fundamental knowledge and skills that set PSP practitioners apart from other software professionals. The PSP BOK serves this purpose and more. It helps individual practitioners to assess and improve their own skills; provides employers with an objective baseline for assessing the personal process skills and capabilities of their product development team members; and guides academic institutions that want to incorporate PSP into their software and other engineering courses or curricula. The PSP BOK also facilitates the development of PSP certification programs that are based on a well-established, standard set of knowledge and skills.

1 Introduction

Software engineering is one of the largest and most influential industries in modern society. It has evolved from early calculation applications used only by government agencies and university think tanks to complex applications that permeate every aspect of modern life. The banking, tele-communications, travel, medical, entertainment, and even agriculture industries rely heavily on software to operate. Software affects even the most mundane aspects of our lives, from buying groceries to doing a load of laundry or filling our cars' fuel tanks with gas.

Yet, in spite of its pervasive influence, software engineering is a relatively young discipline. The term "software engineering" has been in popular use only since the late 1960s, following its introduction in the title of a NATO Science Committee conference at Garmisch, Germany [Naur 69].

One frequent criticism of the software profession is the poor quality of the products it produces. This problem has been attributed to many causes, from the way software professionals are educated to the overall problems inherent within a young profession. An article in the online encyclopedia *Wikipedia* summed up the criticism of software development as follows:

> *"In traditional engineering, there is a clear consensus how things should be built, which standards should be followed, and which risks must be taken care of; if an engineer does not follow these practices and something fails, he gets sued. There is no such consensus in software engineering: Everyone promotes their own methods, claiming huge benefits in productivity, usually not backed up by any scientific, unbiased evidence" [Wikipedia 05].*

A powerful counter to this criticism is the widespread adoption of the Personal Software Process (PSP) methodology. Developed in 1993 by Watts S. Humphrey, the PSP is a disciplined and structured approach to developing software. By using the PSP concepts and methods in their work, individuals in almost any technical field can improve their estimating and planning skills, make commitments that they can meet, manage the quality of their work, and reduce the number of defects in their products.

The effectiveness of the PSP methodology (and its companion technology, the Team Software Process[SM] or TSP[SM]) in both academic and industrial settings is documented in numerous technical reports and peer-reviewed journal articles. Since PSP relies heavily on the collection and analysis of personal data as proof of effective process implementation, the claims made in these reports and articles are supported by objective, hard-data evidence.

The concepts and methodologies of the PSP and TSP technologies have reached a level of maturity sufficient to warrant that further refinements be made by the professional community, academia, and

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

certification entities. To support this effort, further educational expansion of the PSP must be accomplished by and accepted in the community. The research performed by Ford and Gibbs revealed that as a profession advances, it must have ways to assess and assure the adequacy of education and training curricula and the competency of individual professionals to further the profession [Ford 96].

Professional PSP competency measures are needed to assess both the level of knowledge acquisition and the level of skill in applying that knowledge. Certification is one of the most widely used mechanisms that a profession employs to make explicit the core set of knowledge and skills that a professional is expected to master, to establish objective assessments of those core competencies, and to provide a foundation for continuing qualification of individual professionals.

At the core of the process of maturing a profession is the establishment of a body of knowledge (BOK). A *body of knowledge* is a document generated by experts or masters in a particular profession to identify and delineate the concepts, facts, and skills that professionals and practitioners in that profession are expected to have mastered. The Institute of Electrical and Electronics Engineers (IEEE) Computer Society has established a body of knowledge for the software engineering profession as a whole [IEEE 04]. The PSP BOK document is meant to complement and build upon that work by describing the essential skills and knowledge specific to the PSP approach to software process improvement.

It is the authors' expectation that as PSP practice becomes more widespread, there will be further evolutions to the body of knowledge, particularly in the area of process extensions. The authors invite knowledgeable users of the PSP to submit suggestions for future revisions to this body of knowledge.

1.1 Purpose

The PSP BOK is not intended to be a comprehensive overview of the entire field of software engineering, nor is it meant to exhaustively delineate every supporting detail of the various key concepts and skills that compose the PSP competency areas. Rather, the purpose of this document is to provide an overview of the competencies, knowledge areas, and key concepts and skills that constitute the core PSP body of knowledge. The main purposes of this document are to

- define the essential knowledge and skills that PSP-trained software professionals are expected to master
- characterize the standard practices of PSP-trained software professionals
- delineate the knowledge and skills that set PSP practitioners apart from common software (and other) engineering practices
- establish a baseline for developing, assessing, and accrediting PSP courses and curricula throughout academia
- facilitate the establishment of PSP certification programs that are based on an established and agreed-upon standard knowledge and skills set
- provide employers with a baseline for assessing the skills and capabilities of their product development team members
- characterize the disciplined practices used by self-directed TSP team members

Another purpose of this document is to define and delineate the baseline knowledge and skill set upon which the Carnegie Mellon® Software Engineering Institute (SEI) certification program for the SEI-Certified PSP Developer is based.

Although the principles and skills of the PSP can and should be applied to every phase of the product life cycle, the primary concentration is on improving the outputs of the development phase of software projects. Therefore, there are categories of software and other engineering knowledge that are not represented in the PSP BOK (architecture, requirements definition, hardware design, etc.), although it is assumed that a proficient software professional will have some degree of familiarity with such topics. It is also assumed that individuals who have mastered the PSP possess certain knowledge and skills, such as the ability to write software in one or more recognized programming languages, and proficiency in basic mathematics and applying statistical methods. Since these knowledge and skill areas are prerequisites for using the PSP, an exhaustive description of these areas is not included in this body of knowledge.

Similarly, although the PSP BOK is meant to guide the design, development, implementation, and assessment of courses and curricula based in part or in whole on the knowledge and skills delineated in it, the PSP BOK is not intended to be a guide for curriculum or course development. Such activities require pedagogical knowledge and expertise outside the domain of this body of knowledge; therefore, this document is meant to guide only the *content*—not the *methodology*—of PSP instruction and training.

2 Suggested Uses of the PSP BOK

The PSP BOK can be used in professional, industrial, and academic settings. For example, it can be used as a basis for certifying practitioners who have attained proficiency in all of the key concepts and skills that the body of knowledge comprises. This section discusses potential uses of the PSP BOK in detail.

2.1 Use by Software Development Professionals

The definitions of essential concepts and skills that compose the PSP BOK can assist software professionals in assessing their own skills and proficiencies, and in identifying areas in which they may need further improvement.

2.2 Use by the Software Development Industry

The PSP BOK can be used by employers who want to establish an objective baseline for assessing the skills and capabilities of their product development team members. By understanding software best practices, the industry can implement improvement efforts within its organizations, thereby achieving higher quality products and better management of costs and schedules.

2.3 Use by TSP Practitioners

The PSP is the training tool that enables TSP to be used by development teams. TSP provides an operational framework in which PSP-trained individuals combine their personal process discipline skills with proven process management techniques to plan, develop, and deliver high-quality software products within the given schedule and cost parameters. However, not all members of development teams are software developers; teams can and do include members who are document writers, testers, quality assurance specialists, and so on. The PSP BOK provides a handy one-stop reference for PSP-trained developers and can provide non-PSP-trained team members with a valuable overview of the concepts and practices used by the developers, thereby establishing a common foundation and shared vocabulary for all members of a development team.

2.4 Use by Academic Institutions

The PSP BOK can assist academic institutions in updating software engineering or computer science curricula to reflect current software development practices used in industry. As employers begin to require that newly hired developers possess the SEI-Certified PSP Developer credential, academic institutions can begin to prepare students for the certification examination. Some institutions may choose to offer a PSP course, while others may choose to integrate PSP into several of their courses. In both cases, institutions can use the guidance provided by this BOK to ensure that the topics included on the certification examination are adequately presented.

Academic institutions are expected to be innovative in developing ways to prepare students to master the PSP BOK. Whether via traditional classroom courses, distance education, or other media, the instruction that academic institutions offer will provide a benchmark for industrial or commercial training programs based on the PSP BOK. Academic instruction in the BOK competencies, knowledge areas, key concepts, and key skill areas also provides a baseline for assessing the quality of instruction offered through industrial or commercial training or other such venues.

3 PSP BOK Structure and Terminology

3.1 Structure

The body of knowledge delineated in this document is organized in an architectural hierarchy in which the concepts and skills of the PSP are described and decomposed into three levels of abstraction. For the purpose of this model, the term *concept* is used to describe the intellectual aspects of the PSP content; that is, the information, facts, terminology, and philosophical components of the technology. The term *skill* refers to the ability of an individual to interpret and apply concepts to the performance of a task; in this document, if individuals understand a concept, it is assumed that they also have the ability to perform the skills related to or founded upon the concept. Together, the key concepts and skills form a *knowledge area*, and related knowledge areas constitute a *competency area*.

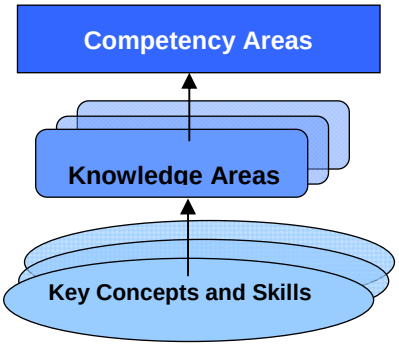

Figure 1: Architectural Hierarchy of the PSP BOK Components

3.2 Operational Definition of Terms

The PSP BOK uses the following terms to describe the categories of principles and processes it contains.

- **Competency area**: a group of closely-related knowledge areas that a practitioner is well qualified to perform intellectually or physically
- **Knowledge area**: the sum or range of specific understanding and ability gained through study of a set of concepts or through experience with a set of skills
- **Concept**: an explanatory principle applicable to a specific instance or occurrence within a particular knowledge area
- **Skill**: proficiency, facility, or dexterity that is acquired or developed through training or experience within a particular knowledge area

4 The PSP Body of Knowledge

This section contains a description of each major competency area, its supporting knowledge areas, and the key concepts and skills that compose each knowledge area. This information does not provide a detailed delineation of the PSP process but rather a high-level description of the proficiencies that a competent PSP-trained individual is expected to master. As the PSP is adopted by broader audiences throughout the world, it is expected that the content of this BOK will evolve over time with an increased range of practice in a variety of environments and cultures.

The PSP BOK is composed of seven competency areas.

- Competency Area 1: Foundational Knowledge
- Competency Area 2: Basic PSP Concepts
- Competency Area 3: Size Measuring and Estimating
- Competency Area 4: Making and Tracking Project Plans
- Competency Area 5: Planning and Tracking Software Quality
- Competency Area 6: Software Design
- Competency Area 7: Process Extensions and Customizations

The first two competency areas provide an overview of the foundation on which the PSP methods are built and an explanation of basic PSP concepts. Competency Areas 3, 4, 5, and 6 discuss more specific components such as planning, making and tracking schedules, measuring and improving product quality, and various techniques for designing software. The final competency area discusses advanced applications of the PSP by experienced practitioners.

Competency Area 1: Foundational Knowledge

The Foundational Knowledge competency area outlines the basic process definition and the knowledge and skills in statistical methods which constitute the conceptual foundation on which the PSP is built. The major knowledge areas composing the Foundational Knowledge competency area are as follows.

1.1 Process Definition – This knowledge area outlines the fundamental concepts and skills that enable engineering professionals to create, use, and stabilize the defined processes of which PSP is comprised.

1.2 Process Elements – This knowledge area delineates the components that are included in any personal process and form a framework for organizing project work.

1.3 Measurement Principles – This knowledge area describes process and product measurement and explains why measures are essential for producing high-quality work.

1.4 Statistical Elements – This knowledge area discusses the statistics which provide a foundation for planning and tracking methodologies used in the PSP, and that also provide an objective means of analyzing and improving personal processes.

References: The material covered in this competency is detailed in the following works.
[Humphrey 95, Chapters 1, 7, 11, 13, Appendix C]
[Humphrey 05a, Chapters 1, 2, 6, 8, 13]

Knowledge Area 1.1: Process Definition

The PSP is a series of defined processes that allow engineering professionals (such as software developers) to produce high-quality products on time and within budget. This knowledge area outlines the concepts and skills needed to create, stabilize, and use defined processes.

1.1.1 Process

A *process* describes the sequence of steps that a knowledgeable professional should follow to do a specified task.

1.1.2 Defined process

A *defined process* is a documented sequence of steps required to do a specific job. Processes are usually defined for jobs that are done repeatedly and need to be done in the same way each time that they are performed.

1.1.3 Benefits of defining a process

A defined process provides

- a clearly delineated framework for planning, tracking, and managing work
- a guide for doing the work correctly and completely, with the steps in the proper order
- an objective basis for measuring the work and tracking progress against goals, and for refining the process in future iterations
- a tool for planning and managing the quality of products produced
- agreed-upon, mutually-understood procedures for team members to use in coordinating their work to produce a common product
- a mechanism that enables team members to support each other throughout the course of the project

1.1.4 Process documentation

Process documentation is the act of producing a succinct written representation of a process, its entry and exit criteria, the process phases, and the process steps for each phase. The process documentation should not contain tutorial or other explanatory material typically needed by unskilled or uninformed individuals; it should provide only the necessary information that experienced practitioners require to enact the process steps.

1.1.5 Processes and plans

Whereas processes are defined sets of steps for doing a task or project, *plans* include both the process steps and other elements required for a specific instantiation of that process, such as resources needed, roles of various project members, schedules, budget, goals and objectives, commitments, and identified risks.

1.1.6 Personal processes

A *personal process* is a defined set of steps or activities that guide individuals in doing their personal work. It is usually based on personal experience and may be developed entirely "from scratch" or may be based on another established process and modified according to personal experience. A personal process provides individuals with a framework for improving their work and for consistently doing high-quality work.

1.1.7 Enactable and operational processes

An *enactable process* defines precisely *how to do* a process, and includes all of the elements required for using the process. An enactable process consists of a process definition, required process inputs, and assigned agents, resources (e.g., people, hardware, time, money), and exit criteria. An *operational process* defines precisely *what to do* by listing the required tasks in enough detail to guide a knowledgeable professional through doing that task. Operational processes provide sufficiently detailed guidance so that teams and individuals can make detailed plans for doing a project and then use the process to guide and track their work. The PSP is an example of an enactable operational process.

1.1.8 Process phases

A defined process consists of a set of steps, elements, or activities that are generally called *phases*. Simple process phases consist of steps with no further substructure. More complex processes may have phases that are themselves processes. The steps or activities in each phase are defined by a *script* (see 1.2.2). At a minimum, any process must have three phases: planning, development, and postmortem.

1.1.9 The PSP process phases

The basic PSP process has three phases.

1. **Planning**: Produce a plan to do the work.
2. **Development**: Perform the work.
   a. Define the requirements (see 4.2.2)
   b. Design the program
   c. Review the design and fix all defects
   d. Code the program
   e. Review the code and fix all defects
   f. Build or compile and fix all defects
   g. Test the program and fix all defects
3. **Postmortem**: Compare actual performance against the plan, record process data, produce a summary report, and document all ideas for process improvement.

1.1.10 Incremental development

The PSP facilitates incremental development. For larger projects, each increment can be an entire PSP project, a PSP development phase, or part of a PSP development phase, depending on the individual's needs.

- Various predefined PSP incremental development processes are available [Humphrey 05a].
- The PSP methods are used most effectively with large-scale incremental development when each increment is of high quality.

1.1.11 Process tailoring

Process tailoring is the act of customizing a process definition to support the enactment of that process for a particular purpose (see 7.1).

1.1.12 Process building and refining

Skilled PSP practitioners can use or tailor the PSP scripts to define or customize their own high-quality personal processes for building a product. Professionals should define their own processes to ensure that the processes fit their needs as closely as possible [Humphrey 95, p. 16]. As the process is enacted on various projects, the users of the process should strive for continuous refinement and improvement in both the process itself and in the quality of the products produced using that process.

Knowledge Area 1.2: Process Elements

This knowledge area describes the components that are included in any personal process and form a framework for organizing project work.

1.2.1 Process elements

Process elements are components of a process. The PSP contains four basic elements: scripts, forms, measures, and standards.

1.2.2 Scripts

Scripts are expert-level descriptions that guide personal enactment of a process. They contain references to pertinent forms, standards, checklists, sub-scripts, and measures. Scripts may be defined at a high level for an entire process or at a more detailed level for a particular process phase. A process script documents

- the process purpose or objective
- entry criteria
- any general guidelines, usage considerations, or constraints
- phases or steps to be performed
- process measures and quality criteria
- exit conditions (such as defined work products or required process data)

1.2.3 Forms

Forms provide a convenient and consistent framework for gathering and retaining data. Forms specify the data required and where to record them. As appropriate, forms also define needed calculations and data definition. Paper forms may be used if automated tools for data gathering and recording are not readily available.

In PSP, *checklists* are specialized forms used to guide personal reviews. Each checklist item verifies an aspect of the product's correctness or conformance with standards or specifications. The checklist items include the most frequently occurring defects that can be found with a review. The entire product is reviewed with a focus on only one checklist item at a time. As the review for each item is done, that item is marked complete. When the entire checklist has been completed, it serves as a record of the review.

1.2.4 Measures

Measures quantify the process and the product. They provide insight into how the process is working by enabling users to

- develop data profiles of previous projects that can be used for planning and process improvement
- analyze a process to determine how to improve it
- determine the effectiveness of process modifications
- monitor the execution of their process and make next-step decisions
- monitor ability to meet commitments and take corrective actions as needed

1.2.5 Standards

Standards provide precise and consistent definitions that guide the work and the gathering and use of data. Standards (such as coding, counting, and defect standards) enable measures to be applied uniformly across multiple projects and to be used consistently. PSP practitioners should be able to recognize areas where standards would be useful and create them when needed.

Knowledge Area 1.3: Measurement Principles

This knowledge area describes process and product measurement and explains why measures are essential for producing high-quality work.

1.3.1 The need for measures

Measures are used in the PSP so that process changes can be identified, assessed, logically implemented, and judged as effective or ineffective.

1.3.2 Measurement types

To be useful for process management, all measures should be defined, precise, accurate, and significant. There are two main types of measures used in PSP: artifact measures and process measures.

- *Artifact measures* are used to quantify product characteristics, such as product size or defects found per element.
- *Process measures* describe or quantify the development or repair process used, and are classified either as historical or current measures.
  - *Historical process measures* are used after the process has been performed to record actual data such as inspection time, test time, and so on.
  - *Current process measures* are used while the process work is being performed to record data such as duration of inspection meetings, code review time as a percentage of coding time, and the like.

Both artifact and process measures may be based on single or multiple measurements. The choice of single or multiple measures depends on the nature of the data and the use for that measure. When multiple measures are taken, a statistically sound procedure is needed to calculate the values to be used from these measures.

1.3.3 Defined measures

A *defined measure* is one that has an explicit and unambiguous meaning. For process measures, this requires the process to be precisely defined to include entry and exit criteria for every phase. The properties to be measured in the process must also be completely and explicitly defined.

1.3.4 Precise and accurate measures

A *precise measure* is one that specifies a value to a suitable level of precision, as with a specific number of digits after the decimal point. An *accurate measure* is one that correctly measures the

property being measured. Measures can be precise and accurate, precise but inaccurate, imprecise but accurate, or both imprecise and inaccurate. For process management purposes, measures should be as precise and accurate as possible.

1.3.5 Meaningful measures

To be meaningful, measures must actually represent the true value of the process or product property being measured, thus indicating that the measurement represents an objective characteristic of a real phenomenon. The measurement's significance increases with the number and consistency of the measurements that are taken.

1.3.6 Uses of process measures

Process measures can be used to evaluate product or process characteristics, to estimate product or process elements, or to predict future outcomes. They can also be used as the basis for determining improvement opportunities and their likely individual and business objectives.

Knowledge Area 1.4: Statistical Elements

Statistics are the foundation for PSP planning and tracking methodologies and also provide an objective means of analyzing and improving personal processes. (Note: PSP-specific definitions, interpretations, or application of statistical terms or elements are called out in each applicable knowledge area subsection.)

1.4.1 Distributions

A *distribution* is a set of numerical values that are generated by some common process (actual sizes of parts developed or size estimates).

1.4.2 Mean

The *mean* is the arithmetic average value of a distribution. In the PSP, the mean is typically an estimate of the mean of the distribution, not the actual mean.

1.4.3 Variance

Variance is a measure of the spread or tightness of a distribution around the mean. In the PSP, the variance is typically an estimate of the variance of the distribution, rather than the actual variance.

1.4.4 Standard deviation

Standard deviation is the square root of the variance. It is often used to characterize the expected range of deviation between an estimate and an actual value. For example, one method in PSP uses standard deviation to categorize software size into relative size tables. Standard deviation is also used as part of the calculation of prediction intervals.

1.4.5 Correlation

Correlation is a measure of the degree to which two sets of data are related. In the PSP, correlation is measured between estimated and actual size and between estimated size and actual effort.

1.4.6 Significance of a correlation

Significance measures the probability that two data sets have a high degree of correlation by chance. Estimates of size and effort in the PSP are more reliable when based on historical data that have a high degree of correlation that is significant.

1.4.7 Linear regression

Linear regression determines the line through the data that minimizes the variance of the data about that line. For example, when size and effort are linearly related, linear regression can be used to obtain effort estimates from size estimates.

1.4.8 Prediction interval

The *prediction interval* provides the range around an estimate made with linear regression within which the actual value will fall with a certain probability. For example, in PSP, the 70% prediction interval for an estimate of size or time implies a 0.7 probability that the actual value of size or time will be within the range defined by the prediction interval.

1.4.9 Multiple regression

Multiple regression is used in the PSP when estimations of size or time depend upon more than one variable. For example, if modifications to programs require much more time than additions, then "added" and "modified" can be separated into two variables for the regression calculation.

1.4.10 Standard normal distribution

The *standard normal distribution* is a normal distribution translated to have a mean of zero and standard deviation of one. The standard normal distribution is used in the PSP when constructing a size estimating table.

1.4.11 Log-normal distribution

Many statistical operations assume that data values are normally distributed, but some PSP measures do not meet this requirement. For example, size values cannot be negative but can have small values that are close to zero. These distributions also typically have higher probability at large values than a normal distribution. When a log transformation is applied to data sets of this type, the resulting distribution may be normally distributed and, therefore, suitable for statistical analyses that assume normally distributed data. Statistical parameters for the normal distribution may be calculated and then transformed back to the original distribution. Size data in the PSP are generally log-normally distributed, so they must be transformed into a normal distribution for construction of a size estimating table.

1.4.12 Degrees of freedom

Degrees of freedom (df) measures the number of data points (n), as compared to the number of parameters (p) that are used to represent them. In linear regression, two parameters (β_0 and β_1) describe the line used to approximate the data. Since at least two points are needed to determine a line, the number of degrees of freedom is *n-2*. In general, the number of degrees of freedom is *n-p*.

1.4.13 The t-distribution

The *t-distribution* enables estimation of the variance of a normal distribution when the true parameters are not known, thus enabling calculation of statistical parameters based upon estimates from sample data. Like the normal distribution, it is bell-shaped, but it varies depending upon the number of points in the sample. For fewer data points, the distribution is short with fat tails. As the number of data points increases, the distribution becomes taller with smaller tails and approaches the normal distribution. In PSP, the t-distribution is important because it helps to determine the significance of a correlation and the prediction interval for regression, each of which is dependent upon the number of points in the sample data set.

Competency Area 2: Basic PSP Concepts

The second competency area outlines the basic process improvement concepts and skills on which the PSP is built. The major knowledge areas composing this competency area are as follows.

2.1 Process Fidelity – This knowledge area introduces the concept of process fidelity and addresses the effect of process fidelity on process quality.

2.2 Data Collection – This knowledge area addresses skills and concepts related to gathering and using process data.

2.3 Data Measures – This knowledge area describes the four basic PSP measures.

2.4 Data Analysis – This knowledge area describes the knowledge and skills needed by PSP practitioners to analyze the process data that they collect.

2.5 Process Improvement – This knowledge area describes the knowledge and skills needed by PSP practitioners to improve their own defined personal process.

References: The material covered in this competency is detailed in the following works.

[Feiler 92]
[Humphrey 95, Chapters 1, 2, 7, 13]
[Humphrey 05a, Chapters 1, 2, 13]
[Humphrey 05b, Chapters 4-9]
[Humphrey 06, Chapter 11]

Knowledge Area 2.1: Process Fidelity

This knowledge area introduces the concept of process fidelity and addresses the effect of process fidelity on process quality.

2.1.1 Process fidelity

Process fidelity (sometimes called *process discipline or process compliance*) is the degree to which individuals follow their own defined personal process. The objective of process fidelity is to improve work performance and produce higher-quality products. Unless the process is followed faithfully, process improvement is not possible.

2.1.2 Process fidelity and useful data

In order to have meaningful data for implementing and improving a personal process, the process must be followed as defined.

2.1.3 Process fidelity and product quality

The quality of the product is governed by the quality of the process used to develop it. It is not enough to define a high-quality process; individuals also must follow that process when developing the product. Creating and consistently using a high-quality process will result in the production of high-quality products. Product quality, in turn, has a direct effect on an individual's ability to meet the schedule and budgetary objectives for the product.

2.1.4 Process fidelity and planning

When a project is planned in accordance with effective and efficient processes and estimates are made based on solid data, the resultant delivery commitment date probably will be accurate. When projects are conducted according to the details contained in an accurate plan, they are delivered on schedule consistently, as long as the work is completed using the defined processes and adjustments are made to the plan to reflect changes in the project conditions. If the defined process is not followed, the plan no longer relates to what is being done, and it becomes impossible to track the progress against the plan accurately. Precise project tracking requires accurate data.

2.1.5 Process fidelity and performance improvement

A well-defined and measured process that is followed faithfully enables individuals to select the methods that best suit their particular abilities and support the tasks that they need to perform. Individuals must personally use well-defined and measured processes in order to consistently improve their performance.

Knowledge Area 2.2: Data Collection

This knowledge area addresses the skills and concepts related to gathering and using process data.

2.2.1 Collecting data

The PSP is based on data because individuals cannot improve their work processes unless they understand precisely how they work and exactly what they do. Data should be used to indicate areas for improvement and to provide a baseline for measuring the effects of changes to the process. Benefits of collecting and analyzing data include the following.

- Establishing standards for products and processes
- Determining if a specific product or process meets its defined criteria
- Precisely controlling individuals' work
- Developing indicators of individuals' performance
- Improving personal performance
- Managing the quality of the products produced
- Estimating when the work will be finished
- Precisely planning, tracking, and reporting on the work

2.2.2 Collecting useful data

To be most useful, data should be collected in accordance with the following guidelines.

- The data collection process must have specific objectives and plans.
- The actual data should be selected for its relevance in implementing a model or testing a hypothesis.
- The data should be collected by the people who are actually going to use it, and they should understand its importance and take appropriate care to gather accurate and relevant information.
- The data collection process must include consideration for the impact of data gathering on the organization and its people.
- The data-gathering plan must have management support; management must regard data collection as an investment with potentially high pay-offs in terms of being able to accurately predict product development costs and schedules, as well as providing a basis for improving an organization's efficiency and the quality of its products.

2.2.3 Collecting high-quality data

Software data are highly error-prone. The best way to ensure that data are of high quality is to train individuals in the proper methods for taking process measures and recording the data that they collect. Using automated tools for data collection, when appropriate tools are available, can help to improve data quality by providing individuals with a convenient means for capturing process information immediately after the data become available.

2.2.4 Ensuring data quality

The best way to ensure that high-quality data are collected is to require individuals to collect their own information in real time (or as soon as possible after the data are generated). However, individuals must be certain that their personal process data will not be used to assess their performance; if people fear that their data will be used to rate or punish them, they will not collect accurate data, if they collect any data at all.

2.2.5 Using data for planning purposes

High-quality data are useful for making accurate personal plans; however, any data (regardless of quality) are better than no data at all. Whenever possible, every product, job, or project should be planned using estimates that are based on analogous historical data (see 2.3 for types of data measures that are typically used for estimates).

- The best estimates are based on actual data from one or more prior products, jobs, or projects of a similar nature.
- The more similar the prior efforts are to the one being planned, the more accurate the estimate is likely to be.
- The more historical data are used when making an estimate, the more accurate the estimate is likely to be.
- Estimating a large job or an entire project as a composite of multiple smaller work products or sub-projects is more accurate than estimating the project as a single large unit.

Knowledge Area 2.3: Data Measures

This knowledge area describes the four basic PSP measures.

2.3.1 Basic PSP measures

The basic PSP measures are time, size, quality (defects), and schedule data.

2.3.2 Time measures

Time is measured in minutes and is tracked while doing the work because time recorded later is more likely to be inaccurate. Basic components are *start date, start time*, *end date, end time, interrupt time, off-task time,* and *delta time*. The *time in phase* is the planned or actual time spent in a particular phase of the process.

- *Interrupt time* is not included in the time measurement for a task or process phase. If there is an interruption during the work, that time is subtracted from the time measurement.
- *Off-task* time is the time spent doing things other than planned project tasks; generally, it is not measured or tracked, since it does not contribute to meeting the stated schedule goals. *Off-task time* includes time spent in management and administrative meetings, attending training classes, reading email, or any of the other essential activities that a team member must do. Off-task time for a given task or work period is calculated by subtracting the total delta time from the total elapsed time spent on a task.
- *Delta time* is the actual time that it took to complete a task or process phase. It is calculated as end time minus start time (less any interrupt time).

Time data are most accurate when collected using an automated tool; the tool should be able to record start and stop times and dates, calculate the elapsed time, and subtract interruption time from elapsed time to calculate the delta time. Each entry for time data should also include the names of the process phase/step, the product and element being worked on, the project task being performed, and the person doing the work.

2.3.3 Size measures

A *size measure* is used to measure how big a work product is. Size measures are selected so that they are appropriate to the work product, for example, using pages (vs. words or letters) as a measure for text pages, or taking programming tasks and language into account for software components (see Knowledge Areas 3.1 and 3.2). Size measure data should be collected in real time to the extent possible because data collected after the fact is more likely to be inaccurate. Size measures apply not only to the final deliverable products, but also to the component parts and interim versions of the product.

Size data are most accurate when collected using an automated tool that will record both the planned and actual sizes for the various product parts or components, using the size accounting measure categories described in 3.1.6. The tool must calculate totals for each category of size data or otherwise ensure the self-consistency the data being collected.

2.3.4 Quality measures (defect data)

In PSP, product quality is measured in terms of defects. A *defect* is anything in the program or software product that must be changed for it to be properly designed, developed, maintained, enhanced, or used. Defects can be in the code, designs, requirements, specifications, or other documentation. Defects should be recorded as soon as they are discovered, preferably using an automated tool. The following data should be collected for *every* defect injected: defect identifier number, date when the defect was discovered, phase when the defect was injected, phase when the defect was removed, defect type, time to find and fix the defect, and a brief description of the defect. A new defect may be injected while fixing another defect. In this case, the second defect is recorded separately, with a reference (called the *fix reference*) back to the original defect. The time required to fix each defect includes the total time required to find and fix the problem, and validate the correction. Fix time is recorded separately for each defect.

2.3.5 Defect type standard

The *defect type standard* defines categories into which similar defects can be placed. Consistent assignment of similar defects to the same defect type category is essential for process analysis.

2.3.6 Schedule measures

Schedule measures are used to plan when the project should be complete and to track progress against the plan. Schedule data are most accurate when collected using an automated tool that will record planned task names and descriptions, phases in which the work is to be done, product/element involved, applicable committed dates for completing tasks, and the dates on which tasks were completed. Schedule data should be collected in real time to the extent possible, particularly information regarding task completion dates, since this is the primary means of obtaining earned value (EV) credit that allows individuals to track their progress against the planned schedule (see 4.5).

2.3.7 Derived measures

The PSP provides a set of performance and quality measures to help individuals implement and improve their personal processes. Specific derived measures are discussed in later knowledge areas.

Knowledge Area 2.4: Data Analysis

This knowledge area describes the knowledge and skills needed by PSP practitioners to analyze the process data that they collect.

2.4.1 Measurement framework and data analysis

All of the PSP measures are related. Individuals must understand how each measure relates to the others and how they can be used to derive measures that provide information on process effectiveness.

2.4.2 Postmortem

A *postmortem analysis* of the work conducted at the completion of a phase or project provides valuable information, including

- updated project data for time, size, defects, and schedule (actual, to-date, and to-date %)
- updated calculations for quality or performance data
- a review of performance against the plan
- updated historical databases for size and productivity
- process adjustments needed, based on personal data (notes made on process improvement proposal (PIP) forms, changes in design or code review checklists indicated by defects that escaped a phase, and so on)

2.4.3 Performance measures

The key performance measures of the personal process are

- ability to meet schedule commitments for delivery of promised content
- quality of delivered content
- project-specific measures

2.4.4 Performance baselines

Before individuals can improve their performance, they first must understand the level of their current performance. After gathering enough project data to provide a meaningful amount of information for analysis, individuals should conduct a baseline analysis of their to-date performance and formulate appropriate process changes to improve their performance in problem areas.

2.4.5 Combined measures

Measures can be combined to provide useful data for future project plans and process improvements. For example, measures from multiple projects can be combined to create a chart showing trends in estimated size vs. actual size to provide data for future size estimates.

2.4.6 Analyzing historical data

Data should be examined to determine whether they are appropriate for analysis. For example, data from projects based on the C# language may not provide an appropriate correlation for analyzing projects based on the C++ language. Historical data also should be examined to determine whether the correlation is adequate and significant as a basis for project and process measurement and analysis.

2.4.7 Analyzing size-estimating accuracy

Historical personal process data for estimated size vs. actual size can be analyzed as a way to determine possible causes for misestimates. Consider the following questions.

- How often is the estimate vs. actual within the 70% prediction interval?
- Is there a tendency to miss parts in the conceptual design?

- What could be done to improve estimates?
- Are the size estimates biased in any way?
- Is there a tendency to misjudge relative sizes of parts?
- Are the size estimates improving over time?

2.4.8 Analyzing effort-estimating accuracy

Historical personal process data for estimated effort vs. actual effort can be analyzed to determine possible causes for misestimates. Consider the following questions.

- How often is the estimate vs. actual within the 70% prediction interval?
- Do the size estimate errors correlate with the effort estimate errors?
- Do underestimated projects correlate with a larger percentage of rework?
- Are the effort estimates improving?
- What could be done to improve estimating accuracy?

2.4.9 Analyzing size and time relationships

Historical personal process data can be analyzed to determine any relationship between size and effort. Consider the following questions.

- Is productivity stable? Why or why not?
- Are there any quantitative differences between higher and lower productivity projects? If so, what might explain these quantitative differences?

2.4.10 Analyzing phase yields

Historical personal process data for phase yields can be analyzed to identify problems and to generate PIPs for possible improvements. Consider the following questions.

- Is there a relationship between yield and review rate (size reviewed per hour) for design and code reviews?
- Are sufficient defects being found in the proper phases?
- Are reviews being conducted effectively?
- What are the personal defect-removal leverages for various appraisal/failure phase combinations? How might these leverages be improved?

2.4.11 Analyzing defects injected per phase

A Pareto analysis of defect types is a useful tool for analyzing historical personal process data for defects injected per phase. Consider the following issues.

- Determine which types of defects occur most often.
- Determine which types of defects take longest to find and fix.
- Analyze the per-phase and overall trends for defects injected per size unit.
- Analyze the per-phase and overall trends for defects injected per hour.

2.4.12 Determining the cost of rework

Data can be analyzed to determine the cost of rework. Consider these aspects when performing an analysis.

- Determine the percentage of PSP project time that defect-free testing would take.
- Determine how long testing takes for PSP projects.
- Determine what types of defects cost the most in terms of time to find and fix (per phase and per project).
- Determine the types of defects most commonly found in personal compiling and testing.
- Determine the types of defects most commonly found in product testing and in the delivered product.
- Generate a Pareto analysis to identify the phases in which the defects found in the product were injected.

Knowledge Area 2.5: Process Improvement

This knowledge area describes the knowledge and skills needed by PSP professionals to improve their own defined personal process.

2.5.1 Rationale for process improvement

The reasons for implementing process improvements are to improve the predictability and quality of delivery, reduce cycle time, and maintain or improve productivity.

2.5.2 Scope for process improvement

Many kinds of processes can and should be used, including personal, team, and organizational processes.

- Although the people involved in process improvement will vary with the process type, the principles and methods are identical for all process types.
- The people who should perform the improvement work are the people who use the process: team members, teams, or even entire organizations. People who are not currently using the process are typically incapable of defining usable and helpful improvements for those who are.
- Large process improvement breakthroughs are rare, but small changes can be made almost every time a process is used.

2.5.3 Benchmarks for process improvement

Benchmarks can help individuals in motivating and guiding their process improvement efforts. The general strategy for obtaining and using process benchmarks is as follows.

- Identify one or more projects doing similar work.
- Establish benchmarking agreements with the individual(s) doing the similar work. In doing so, consider
  - similarity of the work
  - opportunities for the teams to interact and share relevant data
  - confidential material

  - disclosure provisions
  - data release and/or publication
  - management review and oversight
- Select best-of-class benchmarks from among the cooperating projects.
- Regularly establish and update benchmark goals for cost, schedule, and quality performance.

2.5.4 Set performance improvement goals based on data

Before implementing any process changes, PSP practitioners should analyze historical process data to determine root causes of past performance problems. Performing an analysis on their performance baselines should help individuals to determine the most important areas for improvement. Once potential changes have been identified, it is important to set measurable performance improvement goals (e.g., "reduce cost of rework by 20%") to know when the desired improvement has been achieved.

2.5.5 Record process improvement suggestions

The PSP uses a Process Improvement Proposal (PIP) form to capture problems with using the process and suggestions for improving or modifying it. Keep the PIP form at hand at all times for recording insights into opportunities for process improvement before those insights are lost.

2.5.6 Implement highest payoff improvements first

Personal data analysis generates many PIPs. Practitioners should choose to implement the PIPs that offer the highest potential improvement in comparison to the effort required to make the changes.

2.5. 7 Measure process changes

Because PSP professionals use personal processes as the basis for doing their work, practitioners must understand how to update their processes to reflect any changes made to those processes. They should also be aware of the impact that changes may have on the applicability of their historical process data to future work based on the altered process.

2.5.8 Monitor performance results

To determine if implemented process improvements have been effective, PSP practitioners should periodically repeat the steps for baselining their work processes and compare their baseline performance to previously established improvement goals. When so doing, practitioners should be careful to avoid the complications of bolstering and clutching.

- *Bolstering* is the selective recall of only those results that reinforce an opinion or belief, usually manifest by forgetting failures and remembering only successes. Use of *all* PSP data from all projects should preclude bolstering.
- *Clutching* is the tendency to perform badly when under pressure or when a good outcome is especially critical, thereby negating successful performance on past projects when using the same processes. By following established processes and using data (rather than instinct) as a basis for instantiating process changes, clutching can be minimized or avoided.

2.5.9 Watch for improvement opportunities

When working on PSP projects, practitioners should watch for new problem areas and be aware of ideas for continued improvement.

Competency Area 3: Size Measuring and Estimating

This competency area describes the size measurement and estimating concepts on which the PSP is built. The essential elements of size measurement and estimating are the ability to define appropriate size measures and to use disciplined methods and historical data to estimate size. The major knowledge areas composing this competency area are as follows.

3.1 Size Measures – This knowledge area outlines the objectives of measuring size, the criteria for selecting a size measure, and the PSP size accounting system.

3.2 Size Data – This knowledge area discusses the primary ways that size data are used in the PSP.

3.3 Size Estimating Principles – This knowledge area discusses the principles upon which the PSP size estimating process is based. The PSP supports many size estimating methods, but all methods must adhere to these principles.

3.4 Proxies – This knowledge area discusses selecting and organizing proxy data.

3.5 The PROBE Estimating Method – The PSP uses a defined estimating process called *PRO*xy *B*ased *E*stimating (PROBE). This method is used to estimate both size and effort. This knowledge area defines how size estimates are made using the PROBE method.

3.6 Combining Estimates – This knowledge area discusses the various ways that estimates can be combined

3.7 Size Estimation Guidelines – This knowledge area discusses the limitations of size estimating.

References: The material covered in this competency is detailed in the following works.

[Humphrey 95, Chapters 4, 5, Appendix A]
[Humphrey 97, Chapters 6, 10]
[Humphrey 00]
[Humphrey 05a, Chapters 3, 5, 6]
[Humphrey 06]

Knowledge Area 3.1: Size Measures

This knowledge area outlines the objectives of measuring size, the criteria for selecting a size measure, and the PSP size accounting system.

3.1.1 Rationale for using size measures

Objectives for using size measures include

- achieving consistency in describing size
- normalizing time and defect data
- making better size estimates and plans

3.1.2 Types of measures

Measures may be categorized as

- absolute or relative
- explicit or derived
- objective or subjective
- dynamic or static
- predictive or explanatory

3.1.3 Criteria for size measures

Useful size measures must be

- related to development effort
  - Does the size of the product statistically correlate with development effort?
  - Does time spent on development of the measured part of the product represent a significant part of the project's work?
- precisely defined
- directly countable
- suitable for early planning

3.1.4 Counting standards

Counting standards provide guidance that is

- precise about what to count
- application/language specific
- invariant, providing the same outcome each time the standard is applied

3.1.5 Physical and logical size

A *physical size measure* provides information about the size of a physical entity (the actual number of occurrences of an item in some product). A *logical size measure* also provides size information but relies on counting groupings of physical entities that can logically be grouped together. Physical size measures are based on a simple objectively described standard – a number that is arrived at no matter who is counting. The logical size measure of a physical entity does not necessarily correspond to the physical size measure of that same entity, depending on the counting standard defined for the logical measurement.

3.1.6 Size accounting

PSP size accounting methods for planned, actual, and to-date size define the measures for

- **base (B)**: the unmodified program to which subsequent enhancements are added
- **added (A)**: code that is added to the base code
- **modified (M)**: the part of the base code that is changed
- **deleted (D)**: the part of the base code that is subsequently removed
- **reused (R)**: an existing part or item that is copied unchanged from a source other than the base
- **added and modified (A&M)**: all added and modified code
- **new reusable (NR)**: a part or item that is developed with the intention of later reusing that part or item
- **total (T)**: the size of the entire program

3.1.7 Using the size measure selection procedure

Steps for selecting size measures are as follows.

1. Gather product development data (resources required, product characteristics measures, any special development conditions, etc.).
2. Rank the products by required resources.
3. Identify the characteristics that distinguish the products that took the greatest effort from those that required the least effort.
4. Select a size measure or size measures. For the candidate size measure(s) determine correlation between size and required resources. If there is no correlation, repeat steps 3 and 4 for other candidate size measures.

Some typical size measures include the following.

- **Database elements**: A count of the fields, queries, or other commonly used elements in a database product
- **Lines of code (LOC)**: A count of the logical lines of code in a product.
- **Screen elements**: A count of the elements in a user interface or other GUI product.
- **Document size**: A count of document pages, lines, words, or characters.
- **Design size**: A count of the classes, data definitions, interface specifications, or GUI characteristics defined.
- **Requirements size**: A count of requirements pages, shall statements, or function points.

Knowledge Area 3.2: Size Data

This knowledge area discusses the primary ways that size data are used in the PSP.

3.2.1 Size data help to make better plans

Size and time are often correlated, and when they are, size estimates can be used to estimate effort. Plans can then be created based on the size and effort estimates.

3.2.2 Size data are useful for tracking development effort

Size and time are often correlated, and when they are, the size estimates can be used to track effort.

3.2.3 Size data help in assessing program quality

Normalizing defect data based on size permits determining the

- quality of all or some part of the development process
- relative defect content of some parts of large programs
- future workload for maintenance and support

Knowledge Area 3.3: Size Estimating Principles

This knowledge area discusses the principles upon which the PSP size estimating process is based. The PSP supports many size estimating methods, but all methods must adhere to these principles.

3.3.1 Estimating is uncertain

No one knows how big the product will be, and the earlier in the process that the estimate is made, the less is known. Estimating can be biased by business needs and other pressures.

3.3.2 Estimating is a learning process

Estimating improves with experience and with data.

3.3.3 Estimating is a skill

Some people will be better at estimating than others. Most people improve at estimating with deliberate practice.

3.3.4 Strive for consistency

The objective of the size estimating process is to follow a process that consistently produces unbiased estimates. Doing so will balance the over-estimates and under-estimates.

3.3.5 Use defined methods for making estimates

Using a defined size estimating process facilitates learning, provides a framework for using historical data, establishes a baseline against which improvement can be measured, and helps to remove bias from the process.

3.3.6 Estimates are subject to error

The accuracy of estimates will fluctuate around some mean. Estimates may also have some bias.

3.3.7 Estimate in detail

When estimating in parts, the total error will be less than the sum of the part errors, assuming that the parts are estimated independently. Estimating in detail also helps to ensure that the estimate is complete.

3.3.8 Use historical data to make estimates

When making size estimates, find a way to use whatever historical data are available.

Knowledge Area 3.4: Proxies

This knowledge area discusses selecting and organizing proxy data.

3.4.1 Using proxies instead of a size measure

Most size measures that meet the required criteria are not available during planning. A *proxy* is a stand-in measure that relates product size to planned function and provides a means in the planning phase for judging (and therefore, of estimating) a product's likely size.

3.4.2 Criteria for choosing a proxy

The criteria for good proxy are as follows.

- The proxy size measure should closely relate to the effort required to develop the product and correlate with development costs.
- The proxy content of a product should be directly countable.
- The proxy should be easy to visualize at the beginning of a project.
- The proxy should be customizable to the needs of each project and individual
- The proxy should be sensitive to implementation variations that affect cost or effort.

3.4.3 Using relative size tables

Relative size tables are used to organize proxy data so that historical proxy data can be used to estimate the size of similar new parts.

3.4.4 Building a relative size table

The PSP defines two procedures for building a relative size table from historical data: the *sort* method and the *standard deviation* method. Other methods may be used, but they must adhere to the size estimating principles.

3.4.5 Building a relative size table with the *sort* procedure

When using the sort procedure for building a relative size table, the parts are separated into functional categories such as calculation, text, data, etc. The table is populated by completing the following steps for each category.

1. Sort the size data.
2. Pick the smallest value as very small (VS).

3. Pick the largest value as very large (VL).
4. Pick the median value as medium (M).
5. For large (L) and small (S), pick the midpoints between M and VL, and M and VS, respectively.

3.4.6 Building a relative size table with the *standard deviation* procedure

When using the standard deviation procedure for building a relative size table, the parts are separated into functional categories such as calculation, text, data, etc. The table is populated by completing the following steps for each category:

1. If the data are log-normally distributed (as is usually the case with program size data), transform the data into a normal distribution by calculating the natural log of each datum; else skip this step.
2. Calculate the mean (avg) and standard deviation (σ) of the data set.
3. Calculate the size midrange points by assigning VS = avg–2σ; S = avg–σ; M = avg; L = avg+σ; VL = avg+2σ.
4. If the original data were log-normally distributed, apply the inverse transformation by calculating the anti-log of each of VS, S, M, L, and VL; else do nothing.

Knowledge Area 3.5: The PROBE Estimating Method

The PSP uses a defined estimating process called PROxy-Based Estimating (PROBE). This method is used to estimate both size and effort. This knowledge area defines how size estimates are made using the PROBE method.

3.5.1 What is PROBE?

PROBE is a procedure for estimating size and effort. The overall procedure is as follows.

1. Develop the conceptual design (see 3.5.2).
2. Identify and size the proxies.
3. Estimate other elements.
4. Estimate program size. (Select the appropriate PROBE method, as described in 3.3.5.)
5. Calculate prediction intervals (for methods A and B only) (see 3.5.8).

3.5.2 Conceptual design

The *conceptual design* is a high-level postulation of the product elements and their functions. The conceptual design subdivides a desired product into its major parts. The conceptual design is used solely as a basis for producing size and effort estimates (see 4.2.4) and may not necessarily reflect how the actual product is designed and built.

3.5.3 Formulate size estimates for proxies

Compare the size of new parts in the conceptual design against similar parts in the historical database to judge type and relative size. Use the number of items per part and historical size/part data to estimate proxy size.

3.5.4 Formulate estimates for various types of program elements

- Count base size (B).
- Estimate modifications (M).
- Estimate deletions (D).
- Estimate base additions (BA).
- Estimate parts additions (PA).
- Estimate reused (R).
- Estimate planned new reusable (NR).

3.5.5 Select the appropriate PROBE method

1. Check to see if method A can be used by ensuring that the data meet the criteria below, and assessing correlation, β_0, and β_1.
   - There are three or more data points (estimated E and actual A&M) that correlate.
   - The absolute value of β_0 is less than 25% of the expected size of the new program.
   - β_1 is between 0.5 and 2.

   If PROBE method A can be used, then calculate the projected size as $y = \beta_0 + \beta_1(E)$, where
   - y = projected added and modified size
   - E = estimated proxy size
   - β_0 and β_1 are calculated using estimated proxy size and actual added and modified size

2. If method A cannot be used, check to see if method B can be used.
   - There are three or more data points (plan A&M and actual A&M) that correlate.
   - The absolute value of β_0 is less than 25% of the expected size of the new program.
   - β_1 is between 0.5 and 2.

   If PROBE method B can be used, then calculate the projected size as $y = \beta_0 + \beta_1(E)$, where
   - y = projected added and modified size
   - E = estimated proxy size
   - β_0 and β_1 are calculated using plan added and modified size, and actual added and modified size

3. If methods A and B cannot be used and there are historical data, use method C. Calculate project size as $y = \beta_0 + \beta_1(E)$, where
   - y = projected added and modified size
   - E = estimated proxy size
   - $\beta_0 = 0$
   - $\beta_1 = \underline{ActualTotalAdded\&ModifiedSizeToDate}$
   - *PlanTotalAdded&ModifiedSizeToDate*

4. If there are no historical data, use method D, which is to use engineering judgment to estimate added and modified size.

3.5.6 Estimate program size

- Calculate estimated proxy size, E = BA + PA + M.
- Calculate projected A&M size, P = $\beta_0 + \beta_1$ (E), for methods A, B, and C.
  For method D, P = professional judgment.
- Calculate planned added size, A = A&M – M.
- Calculate planned total size, T = P + B – D – M + R.

3.5.7 Count and calculate actual data for various program elements

- Count BA, PA, M, D, R.
- Calculate actual proxy size, E = BA+PA+M.
- Count actual total size, T.
- Calculate actual added size, A = T-B+D-R.
- Calculate actual added and modified size, A&M = A+M.
- Count actual new reusable, NR.

3.5.8 Prediction interval definition

The *prediction interval* is used in PROBE methods A and B. A prediction interval is

- the range within which the actual size is likely to fall 70% of the time
- not a forecast
- applicable only if the estimate behaves like historical data

Knowledge Area 3.6: Combining Estimates

This knowledge area discusses the various ways that estimates can be combined.

3.6.1 Combine independent estimates

Use this method to combine independent estimates.

1. Make separate linear regression projections.
2. Add projected sizes.
3. Add the squares of the individual ranges and calculate square root to calculate prediction interval.

3.6.2 Use multiple proxies

Use multiple regression when there is (*a*) correlation between development time and each proxy, and (*b*) the proxies do not have separate size/hour data.

1. Identify and size each proxy.
2. Use multiple regression to project program size.
   $y = \beta_0 + x_1\beta_1 + x_2\beta_2 + \ldots x_m\beta_m$
3. Calculate prediction intervals.
   UPI = projected size + range (70%)
   LPI = projected size – range (70%)

Knowledge Area 3.7: Size Estimation Guidelines

This knowledge area describes the limitations of size estimating.

3.7.1 Clustered or grouped data

For data that are clustered or grouped, size estimates may not be very useful for estimating effort. However, the size estimate still may be useful in estimating average effort.

3.7.2 Extreme data points

Extreme data points can lead to erroneous β_0 and β_1 values, even with high correlation. Estimates made for points outside the range of the data used to calculate β_0 and β_1 are likely to be seriously in error.

3.7.3 Unprecedented products

Resist making an estimate until the completion of a feasibility study and development of prototypes. Do not confuse making an estimate with guessing.

3.7.4 Data range

Estimates made for points outside the range of the data used to calculate β_0 and β_1 are likely to be seriously in error.

Competency Area 4: Making and Tracking Project Plans

This competency area discusses the ability to use an estimate of software size to plan and track a software project. Essential parts of project planning are the ability to construct a schedule, define tasks, plan tasks to conform to the schedule, and to track task completion against the plan. The major knowledge areas composing the competency area are as follows.

4.1 PSP Planning Principles – This knowledge area delineates the principles upon which the PSP planning framework is based.

4.2 The PSP Planning Framework – This knowledge area delineates the framework that integrates PSP planning tasks, historical databases, and tracking activities. It also addresses using PROBE to generate overall resource estimates.

4.3 Software Size and Effort – Project planning requires an estimate of software size (see Competency Area 3). This knowledge area describes the relationship between size and effort.

4.4 Task and Schedule Planning – This knowledge area describes how to use an overall resource estimate to create a schedule that defines the tasks to be completed and the expected completion dates.

4.5 Schedule Tracking with Earned Value – The PSP earned value (EV) system is used to track the progress of work completed against the schedule plan. This knowledge area discusses calculating EV, using the EV to determine work progress against the plan, and revising the planned schedule based on average EV earned to-date on the project.

4.6 Planning and Tracking Issues – Management must be kept informed of project status. Projects that will not be completed on schedule may need to be replanned.

Reference: The material covered in this competency is detailed in Chapters 4 and 7 of [Humphrey 05a].

Knowledge Area 4.1: PSP Planning Principles

This knowledge area delineates the principles upon which the PSP planning framework is based.

4.1.1 Plan the work

The people who do the work are best suited to plan the work.

- Individuals should always develop a plan for the work before committing to or starting a project. When individuals are involved in developing the plan, they are most likely to be committed to that plan.

- Plans should be based on a defined process and historical data, and made at a level of detail that is appropriate to the job to be done.
- When it is difficult to make an accurate plan, start with a preliminary plan and replan often.
- When the plan does not fit the work, revise the plan.

4.1.2 What is a PSP plan?

A PSP plan

- defines the work and how it will be done
- is a basis for agreeing on the cost, schedule, and resources for a project
- is an organizing structure for doing the work
- is a framework for obtaining the required resources
- provides a record of what was initially committed

4.1.3 Detailed plans

Detailed plans guide individuals' work and allow them to precisely track their progress. Detailed plans are more accurate, provide more precise measures, and give better guidance than high-level plans. Detailed plans also enable individuals to make accurate projections and realistic commitments, be more productive, do higher-quality work, and maintain their motivation to meet other plan goals.

Knowledge Area 4.2: The PSP Planning Framework

This knowledge area delineates the framework that integrates PSP planning tasks, historical databases, and tracking activities. It also addresses using PROBE to generate overall resource estimates.

4.2.1 Software product plan components

Components of a software product plan include the following.

- Project sizing: How large is the project and how much time is needed to do the whole project?
- Project structure: How will the work be accomplished? How should the tasks be sequenced?
- Project status: What is the status of the project at any given time? How can the completion date be estimated?
- Assessment: Compare the actuals to estimates. How good was the plan? How can the plan be improved next time?

4.2.2 PSP planning framework

The PSP planning framework consists of seven basic tasks.

1. Define the requirements (see 4.2.3).
2. Produce the conceptual design (see 4.2.4).
3. Produce the product size estimate (see 3.5.5).
4. Produce the resource estimate (see 4.2.6).
5. Produce the schedule (see 4.2.10 and 4.5).
6. Develop the product (see 4.2.11).
7. Analyze the process (see 4.2.12 and 2.3.2).

4.2.3 Requirements definition

Start by defining the work to be done, in as much detail as possible. The accuracy of the plan is dependent on how much the individuals know about the work to be done at the time when the work is being planned.

4.2.4 Produce the conceptual design

The conceptual design (see 3.5.2) is a preliminary approach to the product that names the expected objects and their functions. When making a conceptual design, several alternative approaches might be considered in order to choose the optimal approach to doing the development work. For larger products, several steps may be needed to produce a conceptual design.

- Produce a preliminary list of product objects and their expected functions.
  - Start with a system- or high-level product design.
  - Subdivide the resulting parts to a level of detail that corresponds to existing elements in the historical database (if any).
- Use the appropriate PROBE method to produce size and resource estimates.
- Total the element estimates to produce the product estimates.

4.2.5 Use PROBE for size and resource estimation

The PROBE method is used to estimate the size of the product and the time required to do the work (see 3.5.5 and 4.2.6).

4.2.6 Select the appropriate PROBE method for resource estimation

1. Check to see if method A can be used.
   - There are three or more data points (estimated E and actual development time) that correlate.
   - The absolute value of β_0 is near 0.
   - β_1 is within 50% of 1/(historical productivity).
2. If method A cannot be used, check to see if method B can be used.
   - There are three or more data points (plan A&M and actual development time) that correlate.
   - The absolute value of β_0 is near 0.
   - β_1 is within 50% of 1/(historical productivity).
3. If method B cannot be used and there are historical data, use method C.
4. If there are no historical data, use method D.

4.2.7 To-date time in phase

To-date time in phase is the sum of the actual time in a given phase for a particular project plus the to-date times in phase from historical projects.

4.2.8 To-date percent time in phase

To-date percent time in phase is the percentage of to-date time in each phase.

4.2.9 Distributing time across phases

Planned time is distributed across phases using historical to-date percent for time in phase.

4.2.10 Schedule projection

An earned value schedule provides a projection of the project completion date (see 4.5).

4.2.11 Product development

Product development is guided by the defined personal process used to generate the plan. As the work is done, individuals gather and record data.

4.2.12 Process analysis

At the end of a project, the gathered data are analyzed (see 2.3.2).

4.2.13 Cost performance index (CPI)

The cost performance index (CPI) is calculated as

$$\frac{\textit{planned total development time to date}}{\textit{actual total development time to date}}$$

Knowledge Area 4.3: Software Size and Effort

Project planning requires an estimate of software size (see Competency Area 3). This knowledge area describes the software relationship between size and effort.

4.3.1 Size and effort correlation

Larger programming projects require more effort. Accurately estimating programming effort requires use of a size measure that has a significant correlation with effort. Size data are suitable for planning purposes if the r^2 value is greater than 0.5 and if the tail area in the significance calculation is ≤ 0.05.

4.3.2 Productivity

Productivity is the ratio of a product's size to the time expended to develop that product, generally measured as size measure per hour.

Knowledge Area 4.4: Task and Schedule Planning

This knowledge area describes how to use an overall resource estimate to create a schedule that defines the tasks to be completed and the expected completion dates.

4.4.1 Project plan characteristics

A project plan must be

- accessible: easy to locate and reference

- clear: straightforward and easy to read
- specific: responsibilities and costs identified
- precise: appropriate level of precision
- accurate: based on relevant data and an unbiased estimating process

4.4.2 Period plans and project plans

A *period plan* covers a specific unit of time, such as a week or month. A *project plan* describes all efforts and costs for developing a product.

4.4.3 Task hours and working hours

Task hours is a measure of the time spent working on defined project tasks. *Working hours* includes task hours and accounts for non-task activities such as time reading and answering e-mail, attending meetings, etc.

4.4.4 Milestones

Milestones are key indicators of project progress. Their completion dates can be estimated so that progress against them can be tracked and risks to their completion can be addressed before the project goes seriously off-schedule.

4.4.5 Schedule plan requirements

Required elements for producing a schedule plan are

- a calendar of available time
- the order in which the tasks are to be completed
- estimated effort for each task

4.4.6 Task order

Task order is driven by the development strategy.

- Each task needs completion criteria.
- Task interdependencies must be defined.

4.4.7 Estimated task time

The time needed for task completion is estimated in one of several ways, using

- the size of the product produced by the task and historic product data from similar tasks
- an overall estimate based on the to-date percent data from similar completed processes
- the appropriate PROBE estimating technique

4.4.8 PSP schedule plans

Schedule plans are produced for PSP projects by following these three steps.

1. Pick an appropriate time period (e.g., three to six months from the planned start date).
2. Distribute the estimated available task time across the duration of the project schedule.
3. Calculate the cumulative planned schedule hours to the end of the project period.

4.4.9 PSP task plans

Task plans are produced for PSP projects by following these four steps.

1. Estimate the task time in hours (see 4.4.7).
2. Calculate the sum of the total planned hours.
3. Calculate the plan time period in which each listed task will be completed, based on the schedule plan (see 4.4.8).
4. Calculate the planned schedule completion date of the project.

Knowledge Area 4.5: Schedule Tracking with Earned Value

The PSP earned value system is used to track the progress of work completed against the schedule plan. This knowledge area discusses calculating EV, using the EV to determine work progress against the plan, and revising the planned schedule based on average EV earned to-date on the project.

4.5.1 Planned value (PV)

The *planned value* of a task is equal to its planned time expressed as a percentage of the total planned time for the project. For example, a 5-hour task in a 50-hour project would have a PV of 10.

4.5.2 Earned value (EV)

Earned value is a method used for tracking the actual progress of completed work against the overall project plan. As each task is completed, its PV is added to the cumulative EV for the project. Partially-completed tasks do not contribute to the EV total.

4.5.3 Using EV measures

When using EV, keep these limitations in mind.

- The EV method assumes that the rate of task completion in the future will be roughly the same as it was in the past. If this is not the case, the EV projections will not be accurate.
- The EV method measures progress relative to the plan. If the plan is inaccurate, the EV projections are also likely to be inaccurate.
- The EV method assumes that the project's resources are uniform. If the staffing level increases, the EV projections will be pessimistic, and if the staffing is cut, the projections will be optimistic.

4.5.4 EV as a measure of actual progress relative to planned progress

At any time during a project, the sum of value earned for completed tasks represents the percentage of work that has been completed. A comparison of the cumulative EV to the cumulative PV at a given time indicates progress of the work against the planned schedule.

- PV is the same as EV: work is on schedule
- EV is larger than PV: work is ahead of schedule
- PV is larger than EV: work is behind schedule

4.5.5 Project tracking with EV

During planning, the total PV for project tasks can be computed for each time period. Likewise, adding up the EVs for completed tasks at any time period in a project determines the percentage of completed work to-date for the project. At any point in the project, the EV can be compared to the cumulative PV to determine if the project is on schedule, behind schedule, or ahead of schedule (see 4.5.4 above).

4.5.6 Calculating PV for each task

PV for a task is calculated by dividing the estimated time ("planned time") for that task by the total planned time for all tasks, then multiplying the quotient by 100.

4.5.7 Calculating PV for each time period

PV for a time period is calculated by adding the PVs for all tasks that are planned to complete during that time period.

4.5.8 Calculating cumulative PV for a given time period

The cumulative PV to-date for a given time period is calculated by adding the PVs for all preceding time periods to the PV for the given time period.

4.5.9 Calculating EV to-date against PV to-date

EV for a given time period and the cumulative EV for that time period can be calculated by using the same procedure for calculating PV. The cumulative EV can be compared to cumulative PV to determine if the project is on schedule (see 4.5.4 above).

4.5.10 Estimating the project completion date

The estimated project completion date can be calculated by computing the average EV per week to-date and then using the average value for EV per week to compute the time necessary to complete the remaining planned value. This assumes that the project continues to earn the average EV rate as before.

Knowledge Area 4.6: Planning and Tracking Issues

Management must be kept informed of project status. Projects that will not be completed on schedule may need to be replanned.

4.6.1 Informing management of issues

Keep management informed of results from earned value analyses and inform them about schedule problems. Data about project status may be helpful in obtaining management assistance.

4.6.2 When to adjust a plan

A plan should reflect the way that the individual is actually working. If it does not, the plan should be revised. When work methods or processes are revised, the entire plan should be re-examined.

4.6.3 Handling part-time assignments

Part-time assignments may be troublesome because task hours are divided among several projects. Frequent switching between tasks makes work on any one task difficult, and hampers coordination with other team members on a project.

Competency Area 5: Planning and Tracking Software Quality

This competency area describes the need to produce products that satisfy users' needs, ways to measure the degree to which user needs are met, and ways to produce high-quality products. The major knowledge areas composing this competency area are as follows.

5.1 PSP Quality Principles – This knowledge area outlines the principles upon which the PSP quality framework is based.

5.2 Quality Measures – PSP data enable determining measures of product and process quality and the effectiveness of the process at removing defects.

5.3 Quality Methods – Personal reviews are an effective and efficient way to improve product quality and individual productivity. Various review methods are effective in different situations.

5.4 PSP Code Reviews – Code reviews should follow a defined process and employ checklists constructed from personal defect data. Consistency in following a review strategy based on experience can make reviews more efficient and effective.

5.5 PSP Design Reviews – Design reviews should follow a defined review process, including appropriate design analyses, using checklists that are built on sound design principles. Consistency in following a review strategy based on measured experience can make reviews more efficient and effective.

5.6 Review Issues – Reviews can be very effective if they are conducted using guidelines that are based on extensive and quantified experience.

References: The material covered in this competency is detailed in Chapters 8 and 9 of [Humphrey 05a].

Knowledge Area 5.1: PSP Quality Principles

This knowledge area outlines the principles upon which the PSP quality framework is based.

5.1.1 Personal responsibility

To produce quality products, individuals must feel personally responsible for the quality of their products (see 7.3). In order to build quality products consistently, individuals must be disciplined in making and following plans, tracking and managing their personal time, and maintaining quality as the top priority.

5.1.2 The economics of quality

- It costs less to find and fix defects earlier in a process, rather than later.
- The longer a defect remains in a product, the greater the cost to remove it.
- Testing is an inefficient and ineffective way to remove defects.
- It is more efficient to prevent defects than to find and fix them.
- The right way is always the fastest and cheapest way to produce a high-quality outcome.
- Reviews are fundamentally more efficient than testing for finding and fixing defects.

5.1.3 Product quality

A quality product is one that satisfies the customer. Such a product must satisfy a minimum threshold of functionality and usefulness. The product should also satisfy user expectations with respect to a number of other criteria.

- The product must work, i.e., perform with reasonable consistency. If this goal is not achieved, then nothing else is important. Additional user concerns might include
  - performance
  - safety
  - security
  - usability
  - functionality
- The product must provide functionality that the user needs and at the time the user needs it. In many development projects, users' perceptions of quality are frequently overlooked because individuals spend most of their time finding and removing defects.

5.1.4 Process quality

A quality process must meet the needs of its users to produce quality products efficiently. A quality process must

- produce a quality product consistently
- be usable and efficient
- be easy to learn and adapt to new circumstances

Knowledge Area 5.2: Quality Measures

PSP data enable determining measures of product and process quality and the effectiveness of the process at removing defects.

5.2.1 Personal defect data

Personal defect data are useful in understanding and refining the personal process. Analysis of these data provides a valuable resource for constructing personal review checklists.

5.2.2 To-date defects injected and removed

To-date defects injected and removed is the sum of the actual defects injected and removed for each project phase, plus the to-date defects injected and removed per phase from historical projects.

5.2.3 To-date percent defects injected and to-date percent defects removed

To-date percent defects injected is the percentage of to-date defects injected in each phase. *To-date percent defects removed* is the percentage of to-date defects removed in each phase.

5.2.4 Yield

Yield is the percentage of defects in the program that are removed in a particular phase or group of phases. A yield measure can be calculated for any individual phase or group of phases.

5.2.5 Phase yield

Phase yield is the percentage of defects removed during a phase.

5.2.6 Process yield

Process yield is the percentage of defects removed prior to entering the compile phase (or before entering unit test if there is no compile phase).

5.2.7 Review yield

Review yield is the percentage of defects in the program found during the review.

5.2.8 Percent appraisal cost of quality (COQ)

Percent appraisal COQ is the percentage of development time spent in design and code review.

5.2.9 Percent failure COQ

Percent failure COQ is the percentage of development time spent in compile and test.

5.2.10 Cost of quality (COQ)

Cost of quality is the percentage of time spent performing appraisal and failure tasks. COQ defines quality issues in management and business terms. The principal COQ measures are as follows.

- **Performance costs**: the costs of doing the job in the first place
- **Appraisal costs**: the costs of examining a product to determine its quality
- **Failure costs**: the costs of fixing a defective product, including all the attendant costs of the product's failure
- **Prevention costs**: the costs of devising and implementing measures to prevent failures

5.2.11 COQ appraisal to failure ratio (COQ A/FR)

COQ A/FR is the ratio of time spent in appraisal tasks to time spent in failure tasks.

5.2.12 Defect density

Defect density is the number of defects found per size measure. It is normalized for product size to enable comparison of various products and the processes that produced them.

5.2.13 Process quality index (PQI)

The *process quality index (PQI)* is a derived measure that characterizes the quality of a software development process. The PQI value is the product of five quality profile component values.

1. *Design quality* is expressed as the ratio of design time to coding time.
2. *Design review quality* is the ratio of design review time to design time.
3. *Code review quality* is the ratio of code review time to coding time.
4. Code quality is the ratio of compile defects to a size measure.
5. Program quality is the ratio of unit test defects to a size measure.

The PQI components are normalized to [0, 1] such that zero represents poor practice and one represents desired practice. The ratios are plotted on the axes of a pentagon with scale [0, 1]. The resulting polygon can be compared with the containing pentagon to determine the quality of the process. Recommended values for each PQI component are as follows.

- Design quality is the minimum of 1.0 or the time spent in detailed design divided by the time spent in coding.
- Design-review quality is the minimum of 1.0 or 2 times the time spent in detailed design review divided by the time spent in detailed design).
- Code-review quality is the minimum of 1.0 or 2 times the time spent in code review divided by the time spent in coding.
- Code quality is the minimum of 1.0 or 20/(10+Defects/KLOC in compile).
- Program quality is the minimum of 1.0 or 10/(5+Defects/KLOC in unit testing).

5.2.14 Calculating values for the PQI components

To calculate and interpret PQI values, do the following.

- Multiply the five PQI element measures together to give a number between 0.0 and 1.0.
- Values below 0.5 indicate that the product is likely to be of poor quality. The lower the value, the poorer the quality is likely to be.

5.2.15 Composite PQI

A *composite PQI* measure represents the overall process quality for a project that produced multiple programs. This composite PQI can be calculated in three ways, each of which has advantages and disadvantages.

1. The *PQI product measure* is calculated by taking the product of all of the PQI for the component programs.
   a. Advantage: This measure will quickly indicate that a product has components with low PQI values.
   b. Disadvantage: For large systems, the values are likely to be too low to be useful in managing system quality.
2. The *overall PQI measure* is determined by using the overall values for all of the programs for calculating the quality profile component values. For example, review time would be the sum of the review times for all the program elements and the unit test defects would be the total defect density for all of the combined programs.

a. Advantage: This measure has the advantage of being easy to calculate and providing a general indicator of overall product quality.
b. Disadvantage: A few poor quality components will be masked by the larger number of high quality components.

3. The *minimum PQI measure* is calculated by using the PQI value for that program component that had the minimum PQI value.
   a. Advantage: This measure has the advantage of rapidly pinpointing any poor-quality component.
   b. Disadvantage: The measure does not indicate anything about the quality of the overall program.

Since no single composite measure is best for all purposes, composite PQI measures should be used with care and their meaning thoroughly explained.

5.2.16 Phase defect removal rate

For each phase of a process, the *phase defect removal rate* is the number of defects found per hour in that phase.

5.2.17 Review rate

Review rate refers to the size of product reviewed per hour. This rate is calculated for both review and inspection phases (see 5.3.3).

5.2.18 Defect-removal leverage (DLR)

Defect-removal leverage is a measure of the relative effectiveness of defect removal for any two process phases. For example, the DRL for design review relative to unit test would be defined as "DRL(DR/UT) = defects per hour in design review divided by defects per hour in unit test."

Knowledge Area 5.3: Quality Methods

Personal reviews are an effective and efficient way to improve product quality and individual productivity. Various review methods are effective in different situations.

5.3.1 Personal reviews

A *personal review* is conducted by the individual who examines his or her own product with the goal of finding and fixing as many defects as possible. Personal reviews should precede any other activity that uses the product (coding, compiling, testing, inspecting, etc.).

5.3.2 Personal review principles

The following principles should be followed when individuals examine their own products during personal reviews.

- Find and fix all defects in the work.
- Use a checklist derived from personal defect data.
- Follow a structured review process.
- Follow sound review practices.

- Measure the reviews.
- Use data to improve the reviews.
- Produce reviewable products.
- Use data to identify where and why defects were injected and then to change the process to prevent similar defects in the future.

5.3.3 Inspections

An *inspection* is a structured team review of a component or product. The object of an inspection is to identify problems in the product. Inspections are conducted according to a defined procedure with attendees filling established roles. In a properly-run inspection, the participants do not discuss the problems identified, nor do they attempt to solve those problems.

5.3.4 Walkthroughs

A *walkthrough* is less formal than an inspection. A product, such as a design or a code segment, is presented to an audience that raises issues and asks questions.

5.3.5 Relationship between reviews and inspections

A personal review should precede any inspection. A review before inspection assures inspectors that they are looking for more subtle issues, rather than obvious mistakes.

5.3.6 Conducting effective personal reviews

For effective and efficient reviews, these practices should be followed.

- Take a break between working and reviewing.
- Review products in hard copy form, rather than electronically.
- Check off each item as it is completed.
- Update review checklists periodically to respond to changes in personal data.
- Build and use a different checklist for each design method, programming language, or product type.
- Thoroughly analyze and verify every non-trivial design construct (see 6.6).

Knowledge Area 5.4: PSP Code Reviews

Code reviews should follow a defined process and employ checklists constructed from personal defect data. Consistency in following a review strategy based on experience can make reviews more efficient and effective.

5.4.1 Code review checklist

A code review checklist is specific to an individual's coding process. It lists defect categories that have caused problems in the past so that these defects are checked for during the review.

5.4.2 PSP code review process

The PSP code review process is as follows.

1. Obtain the code review checklist, coding standard, and defect standard.

2. Print a copy of the code to be reviewed.
3. For each defect category on the checklist, make a complete pass over the code and check off each item as it is completed.
4. Correct all defects and check each defect fix for correctness.

5.4.3 Code review strategy

A review strategy should be based on data that show the strategy to be effective. An initial strategy is to examine lower-level modules first so that procedural abstractions are reviewed and understood before higher-level ones. After a strategy has been determined to be effective, it should be followed consistently. Depending on the type of product and the individual's knowledge of its design, different review strategies may be appropriate.

5.4.4 Review against a coding standard

Code reviews should check for compliance with coding standards. Applicable coding standards should be referenced in the code review checklist.

Knowledge Area 5.5: PSP Design Reviews

Design reviews should follow a design review process, including appropriate design analyses, based on checklists that are built on sound design principles. Consistency in following a review strategy based on measured experience can make reviews more efficient and effective.

5.5.1 Design review principles

- Produce designs that can be reviewed.
- Follow an explicit review strategy.
- Review the design in stages.
- Verify that logic correctly implements the requirements.
- Check for safety and security issues.

5.5.2 Design review checklist

A design review checklist is specific to an individual's design process. It is based on personal defect data and lists defect categories that have caused problems in the past so that these defects are checked for during the review.

5.5.3 PSP design reviews

The PSP design review process is as follows.

1. Obtain the design review checklist and design and defect standards.
2. Print a copy of the design to be reviewed, if appropriate.
3. For each defect category on the checklist, make a complete pass over the design and check off each item as it is completed.
4. Correct all defects and check each defect fix for correctness.
5. Analyze all complex design constructs to verify their correctness (see 6.6).

5.5.4 Design review strategy

A review strategy should be based on data that show the strategy to be effective. After a strategy has been determined to be effective, it should be followed consistently.

Knowledge Area 5.6: Review Issues

Reviews can be very effective if they are conducted using guidelines that are based on extensive and quantified experience.

5.6.1 Review efficiency

Design and code reviews find defects directly, helping the reviewer to gain a mental picture of the intended behavior of the program. In large system-development processes, design and code reviews are especially important because effective scale-up of PSP methods requires that all increments be of high quality. To ensure that large-scale systems achieve the same high quality as smaller systems, the PSP scripts should be followed, and each module and/or increment should be subjected to design reviews, code reviews, and regression testing to ensure that new increments do not cause problems with previously-tested and accepted functioning modules.

5.6.2 Reviewing before or after compiling

Many development environments use automatic code analyzers and/or compilers that are quite helpful; their use is not discouraged. However, to be the most effective, the review should be performed before using the code analyzer or compiler. Code reviews should be performed prior to testing.

5.6.3 Review objectives

Properly conducted code reviews significantly reduce testing time and produce high-quality results. Unless the individual is committed to producing high-quality products, the review process is likely to be ineffective. Individuals whose objective is to begin testing as soon as possible rarely perform code reviews or perform them so poorly that they are a waste of time.

Competency Area 6: Software Design

The Software Design competency area describes the ability to incorporate design and design verification activities into a personal software development process. The major knowledge areas composing this competency area are as follows.

6.1 Software Design Principles – This knowledge area describes software design principles as incorporated in the PSP.

6.2 Design Strategies – This knowledge area addresses the design strategies used in the PSP.

6.3 Design Quality – This knowledge area describes the key characteristics that can be used to assess the quality of a software design.

6.4 Design Documentation – Software designs must be documented, along with the related requirements, constraints, and rationale. This knowledge area discusses the design documentation that is the responsibility of the individual.

6.5 Design Templates – The PSP does not specify design techniques, but does provide a set of templates as a frame for design documentation. The templates help to ensure that a system and its modules are completely and precisely implemented.

6.6 Design Verification – To be effective, design reviews must go beyond simply reading through a design document. The PSP provides a number of design verification techniques that can be used to identify errors and omissions in software designs.

References: The subject matter covered in this competency is detailed in the following works.
[Humphrey 95, Chapters 10, 12]
[Humphrey 05a, Chapters 10-12]

Knowledge Area 6.1: Software Design Principles

This knowledge area describes software design principles as incorporated in the PSP.

6.1.1 Definition of software design

A *software design* transforms an ill-defined requirement into an implementable product specification.

6.1.2 The design process

The *design process* is the set of steps used within a design methodology to create a design. The design process should result in an overall view of the requirement solution, which is unobscured by low-level

implementation details. It does not construct the solution but explores the potential solution space and makes decisions about the structure and behavior of the intended product.

6.1.3 The role of design in the overall software development process

Software design links the requirements for a system to its implementation. By appropriate use of abstraction, it manages complexity and ensures that the system components work together to produce the desired results.

6.1.4 The "requirements uncertainty principle"

Because a new system affects the users and changes their needs, the requirements for a software system are often not completely known until after the completed product is put to use. The design process must provide a stable basis for this ongoing evolution.

6.1.5 The role of design in PSP

Well-designed components are critical to the success of the larger systems that utilize them. Individuals must employ design practices that can meet the demands of complex and dynamically evolving systems.

6.1.6 Design methodology in PSP

The PSP does not prescribe the use of any specific design methodology, but does it does define the requirements for design documentation.

6.1.7 Design specification structure

The elements of a complete design can be specified using the following specification structure.

- External-static (inheritance, class structure)
- External-dynamic (services, messages)
- Internal-static (attributes, program structure, logic)
- Internal-dynamic (state machine)

6.1.8 Need for design precision

A design specification should be precise. The lack of a precise design is the source of many implementation errors. For best design precision, specify and document design decisions before beginning the coding step of the process.

Knowledge Area 6.2: Design Strategies

This knowledge area addresses the design strategies used in the PSP.

6.2.1 The need for design strategies

Design is a complex intellectual process that cannot be automated, reduced to a routine procedure, or precisely controlled or predicted; however, some guidelines and strategies can be helpful in separating routine and creative activities, in ensuring that the design work is performed properly, and in identifying effective design tools and methods.

6.2.2 Nature of the design process

Design is a learning process that commonly requires moving between design levels and from one part of the system to another.

6.2.3 Design process guidelines

- Where practical, complete higher-level designs first.
- Record all assumptions, outstanding issues, questions, and problems.
- Where appropriate, use prototyping or experimentation to reduce uncertainty before designing.
- Do not consider a design complete until the designs for all interdependent components are also completed.
- Document all designs (see 6.6).

6.2.4 Types of design strategies

Design strategies may include the following.

- Progressive
- Functional enhancement
- Fast-path
- Dummy

Knowledge Area 6.3: Design Quality

This knowledge area describes the key characteristics that can be used to assess the quality of a software design.

6.3.1 Design precision

Designs should be concise and unambiguous. The design should contain sufficient detail for all intended uses of the design documentation.

6.3.2 Design completeness

- All relevant details should be included, without any unnecessary redundancy.
- The design documentation should not be limited to individual component designs, but should also document system-wide or emergent concerns.
- It is helpful to include the rationale for design decisions; it is often helpful to document alternatives that were not chosen.

6.3.3 Design usability

The design must be accessible to and understandable by all its users.

Knowledge Area 6.4: Design Documentation

Software designs must be documented, along with the related requirements, constraints, and rationale. This knowledge area discusses the design documentation that is the responsibility of the individual.

6.4.1 The need for software design documentation

Software designs must be documented, along with the related requirements, constraints, and rationale, because designs for all but the simplest programs are needed by people who will be involved with the eventual products. Examples include the following.

- The individual: to facilitate program implementation, verification, and test
- Team members: to enable design inspections and design coordination
- Testers: to enable test planning
- Maintainers: to facilitate product enhancement and repair
- Documenters and users: to enable others to understand what the product does and how it works

6.4.2 Overall design documentation concerns

To ensure that the design documentation continues to represent the product, the design documentation must be self-consistent, and changes must be managed and properly documented.

6.4.3 Common types of design documentation

The individual produces design documentation covering

- program context
- program structure
- related components
- external variables, calls, references
- detailed program logic description for design decisions; it is often helpful to document alternatives that were not chosen

6.4.4 Design visibility

The design documentation provides the visible representation of a design used for review and verification. The design is recorded using an appropriate design notation (see 6.5.1).

6.4.5 Design documentation practice

A useful practice when implementing a design is to start with the full program's design and, as each design section is implemented, encapsulate that design segment in a comment immediately before the implementation.

Knowledge Area 6.5: Design Templates

The PSP does not specify design techniques, but does provide a set of templates as a frame for design documentation. The templates help to ensure that a system and its modules are completely and precisely implemented. The templates are useful in guiding the individuals in producing simple, precise, and complete design documentation.

6.5.1 Design notation

A notation based on mathematical logic can assist in producing concise and unambiguous design documentation. Examples are pseudocode and Zed. The following criteria should be followed when selecting an appropriate design notation.

- The design notation should be capable of precisely and completely representing the design.
- It must be understandable and usable to the people who will use and/or implement the design.
- It should help the designer to produce a high-quality design.
- It should be compatible with the implementation language that will be used.
- It should allow variable degrees of implementation dependence.

6.5.2 PSP design templates

The PSP design templates represent the static structure and the dynamic behavior of a software system, capturing both the externally visible characteristics and the internal details (see 6.1.6). A complete PSP design should contain the following four categories of design elements.

- *External-dynamic*: Use the operational specification template (OST) and the functional specification template (FST) to record this information (see 6.5.3 and 6.5.4).
- *External-static*: Use the functional specification template (FST) to record this information (see 6.5.4).
- *Internal-dynamic*: Use the state specification template (SST) to record this information (see 6.5.5).
- *Internal-static*: Use the logic specification template (LST) to record this information (see 6.5.6).

6.5.3 Operational specification template (OST)

The OST documents the external-dynamic characteristics of a part of a software system. It describes one or more scenarios involving the part and the actors (e.g., users or other systems) that interact with it. Each OST has a unique ID, a user objective, and a scenario objective. For each step in a scenario, the OST lists the following elements.

- Source (system or specified actor)
- Step number
- Action
- Comments

6.5.4 Functional specification template (FST)

The FST documents a part (e.g., a class) of a software system, its external-static relationships, and its externally visible attributes. The FST also documents the external-dynamic characteristics of a part. It describes actions (e.g., class methods) that the part makes available for external use; this description includes the defined interface for each action, including arguments, constraints, and returned results.

6.5.5 State specification template (SST)

The SST documents the internal-dynamic behavior of a software system and its parts (e.g., classes) when that behavior is represented as a set of states, transitions between states, and actions associated with the transitions. The SST can be supplemented by a separate state diagram that graphically depicts the states, transition conditions, and actions. An SST contains

- state names and descriptions
- functions and internal parameters used in transition conditions
- details of state transitions
  - current state
  - next state
  - transition condition (predicate)
  - action performed when transition occurs

6.5.6 Logic specification template (LST)

The LST documents the internal-static characteristics of a part of a software system. It describes the internal logic of the part, using pseudocode to clearly and concisely explain its operation. Note that the LST information may be embedded as comments in the program source code, rather than using a separate form, as long as it is clear and sufficiently detailed.

6.5.7 Template usage

The PSP design templates may be

- used to document a design produced using various design techniques
- used to document the design of an existing software system, to support redesign or verification
- supplemented or partially replaced by other design documentation techniques (e.g., the Unified Modeling Language), as long as equivalent design information is captured in an easily usable form
- applied at different levels of the design hierarchy

Knowledge Area 6.6: Design Verification

To be effective, design reviews must go beyond simply reading through a design document. The PSP provides a number of design verification techniques that can be used to identify errors and omissions in software designs.

6.6.1 Design standards

Software designs can be verified against standards, which promote consistency and quality. These standards may include

- product conventions
- product design standards
- reuse standards

6.6.2 Verification methods

Software verification methods include

- execution table verification
- trace-table verification
- state-machine verification
- loop verification
- other analytical verification methods

6.6.3 Choosing the appropriate design verification method

- Analyze personal defect data to determine which design aspects are most defect-prone. It is not a prudent use of time to verify design aspects where make few (if any) defects are made.
- Evaluate the effectiveness of current verification methods. Identify a family of effective techniques and use them, even on small programs.
- Consider the economics of current verification techniques. Choose the verification methods that are most effective personally and that best apply to the conditions of the design.

6.6.4 Using execution table verification

- Identify loops and complex routines for verification.
- Choose order of analysis (e.g., top down or bottom up).
- Construct an execution table with program steps and relevant variable values, using multiple copies for loop iterations
- Verify execution results against the requirements specification.

6.6.5 Using trace-table verification

- Identify representative logical cases for analysis.
- For each logical case, verify using an execution table.

6.6.6 Execution table verification vs. trace-table verification

Differentiate between execution table and trace-table verification and know when to use each one.

6.6.7 Using state-machine verification

- Check the state-machine structure to ensure it has no hidden traps or loops, using a state diagram if practical.
- Examine each state and verify that the set of transitions from that state is complete (defined for all possible transition condition values).
- Examine each state and verify that the associated state transitions are orthogonal (only one transition defined for each set of transition condition values).

6.6.8 Using loop verification

Verify loop initiation, incrementing, and termination, using the verification methods appropriate to the type of loop.

- For-loop verification
- While-loop verification
- Repeat-until verification

Competency Area 7: Process Extensions and Customization

The Process Extensions and Customization competency area describes the modifications to the PSP that are required when scaling up from smaller programs to larger ones, when working with unfamiliar situations or environments, or when moving to team-based development instead of working alone. The major knowledge areas composing this competency area are as follows.

7.1 Defining a Customized Personal Process – A defined process should not be regarded as "one size fits all." This knowledge area addresses situations in which processes must be tailored to meet changes in needed outputs or developed from the ground up to address new situations or environments.

7.2 Process Evolution – A process cannot be evolved to fit changing needs or situations until the current process accurately represents what is actually done when using that process. This knowledge area addresses the activities involved with incrementally evolving an initial process into one that is an accurate and complete description of the actual process.

7.3 Professional Responsibility – Exceptional work requires responsible behavior on the part of a professional. This knowledge area describes some of the practices of responsible professionals.

References: The material covered in this competency is detailed in the following works.

[Horn 90]

[Humphrey 95, pp. 483-485, 725-740]

[Humphrey 05a, Chapter 13]

Knowledge Area 7.1: Defining a Customized Personal Process

A defined process should not be regarded as "one size fits all." This knowledge area addresses situations in which processes must be tailored to meet changes in needed outputs or developed from the ground up to address new situations or environments.

7.1.1 When to define a new or customized process

Different situations call for different methods: what works well in one environment may not be effective in another. For example, simple programming tasks may require little or no design time. However, larger systems or high-security systems (regardless of size), require a thorough design. A process without a design phase may require customization to include this activity when tailoring an existing process to fit a new situation, when the process scalability changes, or when security requirements change.

7.1.2 How to define a new or customized process

Defining a new or customized personal process follows the same principles as those for software development: start with user needs, and end with final test and release. There are eight general steps for tailoring or creating a personal process.

1. Determine personal needs and priorities.
2. Define process objectives, goals, and quality criteria.
3. Characterize the current process.
4. Characterize the target process.
5. Establish a process development strategy.
6. Define the initial process.
7. Validate the initial process.
8. Enhance the process.

7.1.3 Using information mapping for documenting a new or customized process

When tailoring an existing process (or developing scripts and forms from scratch), follow the following principles of information mapping [Horn 90].

- *Chunking*: Organize information into groups that are manageable to read and/or to accomplish.
- *Relevance*: Group "like things" together and exclude unrelated items from each chunk.
- *Labeling*: Provide the user with a label for each chunk of information.
- *Consistency*: Use consistent terms within each chunk of information, between the chunk and the label, in organizing the information, and in formatting the document or instrument in which the information is recorded.
- *Integrate graphics*: Use tables, illustrations, and diagrams as an integral part of writing.
- *Accessible detail*: Write at the level of detail that makes the document usable for all readers.
- *Hierarchy of chunking and labeling*: Group small chunks around a single relevant topic and provide each group with a label.

Knowledge Area 7.2: Process Evolution

A process cannot be evolved to fit changing needs or situations until the current process accurately represents what is actually done when using that process. This knowledge area addresses the activities involved with incrementally evolving an initial process into one that is an accurate and complete description of the actual process.

7.2.1 Initial process definition

Initial process descriptions are seldom accurate, due to a phenomenon analogous to the Heisenberg Uncertainty Principle: the act of defining a process changes that process. The initial description of the process usually contains omissions, idealizations, and other inaccuracies. The process of accurately describing what really happens often affects the process during the very act of defining it.

7.2.2 Refining a personal process

1. Start with a characterization of the process as currently used.
2. Define the target or ideal process.
3. Define the steps needed to move from the current process to the target process.
4. Develop the necessary scripts, forms, standards, and measures to use in the process.
5. Review the process as it is being implemented and correct any identified errors or omissions.

Knowledge Area 7.3: Professional Responsibility

Exceptional work requires responsible behavior on the part of a professional. This knowledge area describes some of the practices of responsible professionals.

7.3.1 Use effective methods for producing good work

Good practices are straightforward, but few people consistently use them. The dedicated professional finds effective methods for consistently producing high-quality work and then uses those methods.

7.3.2 Use data to discover strengths and weaknesses

Use the postmortem analysis of personal data to build an understanding of what is done well and areas where improvement is called for. Focus on making small improvements regularly, and major changes will take care of themselves.

7.3.3 Practice

The key to improving the quality of work products is to practice skills on the job to the maximum extent possible.

7.3.4 Learn from others, and pass it on

Talk to colleagues and review the literature to learn about new techniques and to learn from the mistakes of others. Share what is learned with others. Take advantage of benefits gained and contribute what is learned.

7.3.5 Find and learn new methods

Watch for innovations that are pertinent to personal needs. Allocate time for skill building whenever possible. Keeping up to date makes employees more attractive to their current employer (and to future employers) as a desirable and competent professional.

Conclusion

The PSP BOK is a three-tiered hierarchical model delineating the competencies, knowledge areas, and key concepts and skills that compose the essential proficiencies to be mastered by a PSP-trained professional. As PSP adoption continues to grow, it is expected that the PSP BOK will evolve to reflect further process extensions and a deeper understanding of the application of the key concepts and skills in actual on-the-job practice.

Appendix Key Statistical Formulae and Procedures

PSP emphasizes use of statistical methods in implementing and improving personal software engineering processes. Many of the specific formulae and statistical procedures listed below are embodied within PSP support tools. The specific statistical formulae and procedures are included in this body of knowledge because they are important concepts and skills for PSP instructors and tool developers. Another reason for including the formulae is that many PSP courses and/or curricula include a requirement for learners to develop programs to implement these statistical formulae and procedures. This provides the students with data to help them understand and improve their personal processes, and gives them a deeper understanding of the mechanics behind the PSP methods and procedures. PSP practitioners are not expected to be able to recall the information in this section, but they are expected to be able to use these methods appropriately.

Numerical data for the PSP are assumed to be generated by some common process, so they can be considered to come from a distribution. Thus, statistical formulae can enable inference about the underlying distribution and the process that generated it. Therefore, the user is dealing with estimates of the distribution parameters. Although not stated for each formula, the statistics are estimates of the various statistical parameters, not their actual values.

Key Statistical Formula and Procedure	Description and Formula
A.1 Calculate the mean, μ, of a dataset $x_1,...,x_n$	$\mu = x_{avg} = \frac{1}{n}\sum_{i=1}^{n} x_i$
A.2 Calculate the variance, σ^2, about the mean for the dataset in A.1	$\sigma^2 = \frac{1}{n-1}\sum_{i-1}^{n}(x_i-\mu)^2$
A.3 Calculate the standard deviation, σ, for the variance in A.2	$\sigma = \sqrt{\sigma^2}$
A.4 Transform a dataset $x_1,...,x_n$ with mean μ and standard deviation σ into standard normal form, *i.e.*, mean = 0 and standard deviation = 1	$z_i = \frac{x_i - \mu}{\sigma}$

Key Statistical Formula and Procedure	Description and Formula
A.5 Calculate the correlation between data pairs $(x_1, y_1), \ldots, (x_n, y_n)$	$$r_{x,y} = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{\left[n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2\right]\left[n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2\right]}}$$ A value of r^2 greater than or equal to 0.5 implies a high positive linear relationship. Although r^2 may approach 1, the relationship may not be significant (see A.8).
A.6 The gamma function	$\Gamma(x) = (x-1)\Gamma(x-1)$, where, for integer values of x, $\Gamma(x) = (x-1)!$ Also, $\Gamma(1) = 1$ $\Gamma(1/2) = \sqrt{\pi}$
A.7 The t-distribution (probability density function) for degrees of freedom *df*	$$F(x) = \frac{\Gamma\left(\frac{df+1}{2}\right)}{\sqrt{df * \pi}\,\Gamma\left(\frac{df}{2}\right)}\left(1 + \frac{x^2}{df}\right)^{-\left(\frac{df+1}{2}\right)}$$
A.8 Calculate the significance of the correlation in A.5	1. Calculate $t = \frac{\lvert r_{x,y}\rvert\sqrt{df}}{\sqrt{1 - r_{x,y}^{\ 2}}}, df = n-2$ 2. Calculate the significance, *p*, of the value of *t* in step 1 by computing its probability. Begin by integrating the *t* distribution function in A.7 from -*t* to *t* for *df* = *n*–2. $$P = \int_{-t}^{t} F(x)dx$$ 3. Calculate the area under the two tails of the *t* function. Let P be the area under the t-distribution from -*t* to *t*. The "p-value" is area under the tails of the *t*-distribution, that is the sum of the integrals from -∞ to -*t* and *t* to ∞. Because the total probability (area from -∞ to ∞) is 1.0, the area under the two tails is p =1.0 – P. 4. The correlation in A.5 is significant if $p < 0.05$.

Key Statistical Formula and Procedure	Description and Formula
A.9 Calculate linear regression estimating parameters for data pairs $(x_1, y_1), \ldots, (x_n, y_n)$	Calculate the linear regression estimating parameters β_0 and β_1. $\beta_1 = \frac{\sum_{i=1}^{n} x_i y_i - n x_{avg} y_{avg}}{\sum_{i=1}^{n} x_i^2 - n x_{avg}^2}$ $\beta_0 = y_{avg} - \beta_1 x_{avg}$
A.10 Use linear regression to calculate an estimated value for *y* given an estimated value of *x* and estimating parameters from A.9	$y = \beta_0 + \beta_1 x$
A.11 Calculate multiple regression estimating parameters β_0, β_1, …, β_n for data set $(x_{11}, \ldots, x_{m1}, y_1), (x_{12}, \ldots, x_{m2}, y_2), \ldots, (x_{1n}, \ldots, x_{mn}, y_n)$	Calculate the multiple-regression estimating parameters β_0, $\beta_1, \ldots, \beta_n$. Find the beta parameters by solving the following simultaneous linear equations. $\beta_0 n + \beta_1 \sum_{i=1}^{n} x_{1i} + \beta_2 \sum_{i=1}^{n} x_{2i} + \ldots + \beta_m \sum_{i=1}^{n} x_{mi} = \sum_{i=1}^{n} y_i$ $\beta_0 \sum_{i=1}^{n} x_{1i} + \beta_1 \sum_{i=1}^{n} x_{1i}^2 + \beta_2 \sum_{i=1}^{n} x_{1i} x_{2i} + \ldots + \beta_m \sum_{i=1}^{n} x_{1i} x_{mi} = \sum_{i=1}^{n} x_{1i} y_i$ $\beta_0 \sum_{i=1}^{n} x_{2i} + \beta_1 \sum_{i=1}^{n} x_{2i} x_{1i} + \beta_2 \sum_{i=1}^{n} x_{2i}^2 + \ldots + \beta_m \sum_{i=1}^{n} x_{2i} x_{mi} = \sum_{i=1}^{n} x_{2i} y_i$ … $\beta_0 \sum_{i=1}^{n} x_{mi} + \beta_1 \sum_{i=1}^{n} x_{mi} x_{m1} + \beta_2 \sum_{i=1}^{n} x_{mi} x_{m2} + \ldots + \beta_m \sum_{i=1}^{n} x_{mi}^2 = \sum_{i=1}^{n} x_{mi} y_i$
A.12 Use multiple regression to calculate an estimated value for y given an estimated value of $(x_1, \ldots, x_m)$ and estimating parameters $\beta_0, \ldots, \beta_m$ from A.11	$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_m x_m$
A.13 Calculate the standard normal function	$F(x) = \frac{1}{\sqrt{(2\pi)}} e^{-\left(\frac{x^2}{2}\right)}$
A.14 Calculate the chi-square function for *df* degrees of freedom	$F(x) = \frac{1}{2^{\left(\frac{df}{2}\right)} \Gamma\left(\frac{df}{2}\right)} x^{\left(\frac{df}{2}\right)-1} e^{-\left(\frac{x}{2}\right)}$

Key Statistical Formula and Procedure	Description and Formula
A.15 Approximate the integral of a function *F(x)* from *a* to *b* using Simpson's Rule	$\int_a^b F(x)dx \approx \frac{W}{3}\left[F(a) + \sum_{i=1,3,5\ldots}^{N-1} 4F(iW) + \sum_{i=2,4,6\ldots}^{N-2} 2F(iW) + F(b) \right]$ where N is the number of segments (an even number) and $W = \frac{b-a}{N}$
A.16 Calculate the range around the new estimate $\bar{x}$ for the 70% prediction interval for linear regression applied to the data pairs $(x_1, y_1), \ldots, (x_n, y_n)$	$Range = t(0.70, df)\sigma\sqrt{1 + \frac{1}{n} + \frac{\left(\bar{x} - x_{avg}\right)^2}{\sum_{i=1}^{n}\left(x_i - x_{avg}\right)^2}}$ where 1. *t* is the limit of the integration of the *t*-distribution for *df* = *n-2* (see A.7) such that $\int_{-t}^{t} F(x)dx = 0.70.$ 2. $\sigma^2 = \left(\frac{1}{df}\right)\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2$ is the variance of y_i from the regression line for the data.

Key Statistical Formula and Procedure	Description and Formula
A.17 Calculate the range around the new estimate $(\bar{x}_1,\ldots,\bar{x}_m)$ for the 70% prediction interval for multiple regression applied to the data set $(x_{11}, \ldots, x_{m1}, y_1)$, $(x_{12}, \ldots, x_{m2}, y_2)$,…, $(x_{1n}, \ldots, x_{mn}, y_n)$	$$Range = t(0.70, df)\sigma\sqrt{1+\frac{1}{n}+\frac{\left(\bar{x}_1 - x_{1,avg}\right)^2}{\sum_{i=1}^{n}\left(x_{1i}-x_{1,avg}\right)^2}+\ldots+\frac{\left(\bar{x}_m - x_{m,avg}\right)^2}{\sum_{i=1}^{n}\left(x_{mi}-x_{m,avg}\right)^2}}$$ where 1. $x_{k,avg} = \frac{1}{n}\sum_{i=1}^{n} x_{ki}$, k = 1, …, m 2. *t* is the limit of the integration of the *t* –distribution function for *df = n-m-1* (see A.7) such that $$\int_{-t}^{t} F(x)dx = 0.70.$$ 3. $$\sigma^2 = \left(\frac{1}{df}\right)\sum_{i=1}^{n}\left(y_i - \beta_0 - \beta_1 x_{1i} - \ldots - \beta_m x_{mi}\right)^2$$ is the variance of y_i from the regression line for the data.

Key Statistical Formula and Procedure	Description and Formula
A.18 Use the chi-square test for normality of *n* data points	1. Convert data to standard normal form. 2. Divide the normal distribution into some number of segments *S*, such that – each segment has probability 1/*S* – $n/S \geq 5$ (an integer if possible) – $S > 3$ – $S^2 \geq n$ – when more than one value of *S* is possible, select the one such that *n* and S^2 are most nearly equal 3. Determine how many items of the ideal normal distribution would fall into each segment. This number is N_i. (If *n/S* is an integer, then $N_i = n/S$). 4. Using the same segment boundaries, determine how many items from the normalized data set fall into each segment. This number is k_i. 5. Calculate the *Q* value for the segments. $Q = \sum_{i=1}^{S} \frac{(N_i - k_i)^2}{N_i}$ 6. Calculate the probability *p* of the χ^2 distribution for S-1 degrees of freedom (*df*) by integrating the chi-square function (see A.14) from 0 to *Q*. 7. Calculate the distribution tail as *1-p*. 8. Tail areas greater than 0.2 are generally considered sufficient to accept a fit; tail areas less than 0.05 are generally considered sufficient to reject a fit; intermediate values indicate intermediate degrees of fit.

Bibliography

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

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

[Feiler 92] Feiler, Peter H. & Humphrey, Watts S. *Software Process Development and Enactment: Concepts and Definitions* (CMU/SEI-92-04). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1992. http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.004.html

[Ford 96] Ford, Gary & Gibbs, Norman E. *A Mature Profession of Software Engineering* (CMU/SEI-96-TR-004, ADA307889). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1996. http://www.sei.cmu.edu/publications/documents/96.reports/96.tr.004.html

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

[Horn 90] Horn, Robert E. *Developing Procedures, Policies, and Documentation.* Waltham, MA: Information Mapping, Inc., 1990.

[Humphrey 89] Humphrey, Watts S. *Managing the Software Process*. Reading, MA: Addison-Wesley, 1989.

[Humphrey 95] Humphrey, Watts S. *A Discipline for Software Engineering*. Reading, MA: Addison-Wesley, 1995.

[Humphrey 97] Humphrey, Watts S. *Introduction to the Personal Process*. Reading, MA: Addison-Wesley, 1997.

[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/publications/documents/00.reports/00tr022.html

[Humphrey 05a] Humphrey, Watts S. *PSP: A Self-Improvement Process for Software Engineers*. Reading, MA: Addison-Wesley, 2005.

[Humphrey 05b] Humphrey, Watts S. *PSP: A Self-Improvement Process for Software Engineers–Instructor's Guide.* Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005.

[Humphrey 06] Humphrey, Watts S. *TSP: Leading a Development Team.* Upper Saddle River, NJ: Pearson Education, Inc., 2006.

[IEEE 04] IEEE Computer Society. Guide to the Software Engineering Body of Knowledge (SWEBOK) 2004 Version. http://www.swebok.org/home.html (2004).

[Naur 69] Naur, Peter & Randell, Brian, eds. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee*. Garmisch, Germany, Oct. 7-11, 1968. Brussels, Belgium: Scientific Affairs Division, NATO, 1969.

[Webb 99] Webb, David & Humphrey, Watts S. "Using the TSP on the TaskView Project." *CrossTalk 12*, 2 (February 1999): 3-10.

[Wikipedia 05] Wikipedia, The Free Encyclopedia. *Criticism of software engineering.* http://en.wikipedia.org/wiki/Criticism_of_software_engineering (2005).

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

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

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
The Personal Software Process℠ (PSP℠) Body of Knowledge, Version 2.0	FA8721-05-C-0003

6. AUTHOR(S)

Marsha Pomeroy-Huff, Robert Cannon, Timothy A. Chick, Julia Mullaney, & William Nichols

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-SR-018

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)

As the profession of software engineering evolves and matures, it must achieve some of the critical elements needed for recognition as a bona fide discipline. Among these elements are the establishment of a recognized body of knowledge (BOK) and certification of professional practitioners.

The body of knowledge contained in this report is designed to complement the IEEE Computer Society's *Software Engineering Body of Knowledge (SWEBOK)* by delineating the skills and concepts that compose the knowledge areas and competencies of a proven-effective process improvement method, the Personal Software Process (PSP). As adoption of the PSP methodology continues to grow, it becomes crucial to document the fundamental knowledge and skills that set PSP practitioners apart from other software professionals. The PSP BOK serves this purpose and more. It helps individual practitioners to assess and improve their own skills; provides employers with an objective baseline for assessing the personal process skills and capabilities of their product development team members; and guides academic institutions that want to incorporate PSP into their software and other engineering courses or curricula. The PSP BOK also facilitates the development of PSP certification programs that are based on a well-established, standard set of knowledge and skills.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Body of knowledge, BOK, SWEBOK, software engineering body of knowledge, personal software process, PSP	92

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