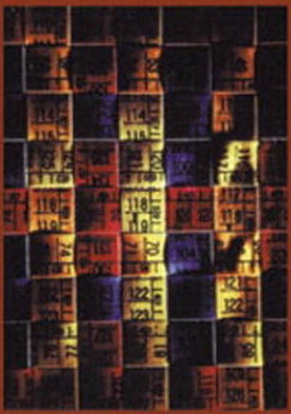
SEI SERIES IN SOFTWARE ENGINEERING

Measuring the Software Process



Statistical Process Control *for* Software Process Improvement

William A. Florac

Anita D. Carleton

Foreword by Watts S. Humphrey

Measuring the Software Process

Statistical Process Control for Software Process Improvement

William A. Florac
Anita D. Carleton

ADDISON-WESLEY

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sidney • Tokyo • Singapore • Mexico City

Software Engineering Institute

The SEI Series in Software Engineering

Special permission to use *Practical Software Measuremnet: Measuring for Process Management and Improvement,* CMU/SEI-97-HB-003, and *Goal-Driven Software Measurement—A Guidebook,* CMU/SEI-96-HB-002, by Carnegie Mellon University, in *Measuring the Software Process: Statistical Process Control for Software Process Improvement,* is granted by the Software Engineering Institute.

Library of Congress Cataloging-in-Publication Data

Florac, William A.
Measuring the software process : statistical process control for software process improvement / William A. Florac, Anita D. Carleton.
p. cm. -- (The SEI series in software engineering)
Includes bibliographical references.
ISBN 0-201-60444-2
1. Software measurement. 2. Computer software--Quality control--Statistical methods. I. Carleton, Anita D. II. Title.
III. Series.
QA76.76.S65F56 1999
005.1'4--dc21 99-20519
C I P

ISBN 0-201-60444-2
Text printed in the United States on recycled paper at Courier in Westford, Massachusetts.
7th Printing, May 2008

To Marilyn, my wife and best friend for over fifty years.

—Bill

I dedicate this book to my Denny, Neil, Kellen, and my parents—the source of all of my inspiration and support for everything I do.

—Anita

This page intentionally left blank

Contents

This page intentionally left blank

Figures

This page intentionally left blank

This page intentionally left blank

Preface

This book is about using statistical process control (SPC) and control charts to measure and analyze software processes. It shows how characteristics of software products and processes can be measured and analyzed using statistical process control so that the performance of activities that produce the products can be managed, predicted, controlled, and improved to achieve business and technical goals.

If you are a software manager or practitioner who has responsibilities for product quality or process performance, and if you are ready to define, collect, and use measurements to manage, control, and predict your software processes, then this book is for you. It will put you on the road to using measurement data to control and improve process performance. Not only will the discussions here introduce you to important concepts, but also they will introduce you to tried-and-true process measurement and analysis methods as set forth in a software environment.

On the other hand, if your organization does not yet have basic measurement processes in place, you should make establishing measures for planning and managing projects your first priority. Handbooks such as *Practical Software Measurement: A Foundation for Objective Project Management* [McGarry 1998] and *Goal-Driven Software Measurement* [Park 1996] make excellent starting points, as do the examples and advice found in books by people such as Watts S. Humphrey and Robert B. Grady [Humphrey 1989; Grady 1987, 1992].

This book is an extension and elaboration of the Software Engineering Institute (SEI) guidebook *Practical Software Measurement: Measuring for Process Management and Improvement* [Florac 1997]. The guidebook grew out of a collaborative effort with the authors of *Practical Software Measurement: A Foundation for Objective Project Management* [McGarry 1998]. Both publications were written to encourage and guide software organizations to use measurements to quantitatively manage software projects and processes.

This book is organized into eight chapters. The focus of Chapter 1 is to introduce you to the primary concepts associated with managing, measuring, controlling, and improving software processes. The motivation for using statistical process control is also discussed—that is, utilizing control charts for making process decisions and for predicting process behavior. This chapter begins by characterizing the term *software process,* especially as it is used in SPC applications. Issues of process performance, stability, compliance, capability, and improvement are briefly introduced (and elaborated throughout the book) since these form the basis for improving process performance. A section on measuring process behavior then follows. A framework for measuring process behavior is presented next and

serves as the guiding structure for the rest of the book. The remaining chapters follow this framework with more detailed discussions, expanding on the activities associated with using statistical process control techniques for improving the software process.

The focus of Chapter 2 is to discuss the activities associated with measuring the software process. They include identifying process management issues, selecting and defining the measures, and integrating the measurement activities with the organization's processes. The idea here is to understand what you want to measure and why and to select appropriate measures that will provide insight into your issues.

In Chapter 3, we discuss the specifics associated with collecting software process data. The principal tasks include designing methods and obtaining tools for data collection, training staff to execute the data collection procedures, and capturing and recording the data. Additionally, there is a discussion of many of the important tools available to analyze, understand, and explain causal relationships to the process performance data.

In Chapter 4, we embark on the initial discussion of analyzing process behavior with Shewhart's control charts by graphically illustrating the concepts of process variation and stability. The basics of constructing control charts, calculating limits, and detecting anomalous process behavior are given to provide a basis for the ensuing chapters.

Chapter 5 is dedicated to providing the information to construct and calculate limits for the several different control charts applicable to software processes. Examples of the calculations and charts are set in familiar software settings.

Chapter 6 discusses a number of topics that arise when using control charts. Guidelines are offered for how much data is necessary for control charting, recognizing anomalous process behavior patterns, rational subgrouping, aggregation of data, and insufficient data granularity.

Chapter 7 provides insight on what actions to take after you have plotted your data on process behavior charts. The actions involve removing assignable causes of instability, changing the process to make it more capable, or seeking ways to continually improve the process.

The book concludes with Chapter 8. It provides ten steps for getting started using statistical process control, cites the experiences by some of those who have used statistical process control in a software environment, and addresses a number of frequently asked questions.

The appendixes include several of the more commonly used tables for calculation of control chart limits and a special topics section that contains detailed discussions addressing statistical process control fundamentals. For those who wish to learn more about the topics addressed in this book, we have included an extensive list of references following the appendixes.

Everything in this book has its roots in experience—often that of the authors, at times that of others. Much of the experience comes from software settings, while other lessons are adapted from other service- and industrial-oriented environments. Some of the numerical examples are composed rather than factual; in others, the

names of organizations providing the data are disguised to maintain confidentiality. We have tried to ensure that the examples represent reasonable extensions of practices that have been demonstrated to be meaningful and successful in software settings.

The focus, then, is on the acquisition of quantitative information and the use of statistical process control methods to help you to reliably identify the problems (and the opportunities) present in the processes you operate. When the methods are properly used, you can confidently use the results to control and predict your process behavior and guide your improvement actions. We recognize that much more can be said about the use of measurements and statistical methods for controlling and improving processes than this book addresses. On the other hand, we have striven to provide the information, ideas, and methods that are sufficient for you to get started using statistical process control to better understand the behavior of your software processes. We trust that it will encourage you and others to begin applying the concepts that it explains and illustrates.

This page intentionally left blank

Acknowledgments

This book represents the aggregation of many years' lessons and experiences in undertaking software process improvement initiatives and from implementing software measurement practices. Throughout the years, we have learned a great deal from a great many—what software process improvement entails, what it means to implement a measurement program, what to measure, how to measure, and how to use measurements to manage projects. We are indebted to Watts S. Humphrey for influencing our thinking on software process improvement and quantitative software process management.

These experiences, in turn, have led us to ask more questions. How can we use measurements to control, improve, and, better yet, predict? This questioning brought us to the use of statistical techniques, specifically to the world of statistical process control (SPC). This is where our diagnostic journey began.

We learned about basic statistical process control issues and techniques from Donald J. Wheeler, Douglas C. Montgomery, Thomas Pyzdek, Gerald J. Hahn, and the authors of the Western Electric *Statistical Quality Control Handbook*. We are indebted to them all for presenting SPC concepts in a very understandable and pragmatic way.

We would also like to acknowledge the support and contributions of our colleagues and friends at the SEI and in the software community. First, we are grateful to our friend and colleague Robert E. Park. Bob was our coauthor on *Practical Software Measurement: Measuring for Process Management and Improvement*, the SEI guidebook that served as a starting point for our work. We appreciated Bob's tenacity in researching the "roots" of SPC and his meticulous attention to all of the details for producing the SEI guidebook. Thanks to Bob, our work had a solid foundation from which to go forward. We also appreciate the enthusiastic support and encouragement from Dave Zubrow and Bill Peterson at the SEI. We also are grateful for the insights, discussions, and review comments from our colleagues associated with the Practical Software Measurement Project: Jack McGarry, Dave Card, Pam Geriner, and Betsy Clark. We would also like to thank Julie Barnard, Betsy Clark, Watts S. Humphrey, Dan Nash, Dan Paulish, Mark Paulk, Ed Weller, and Dave Zubrow for reviewing the manuscript and offering many helpful comments. We also want to especially acknowledge Julie Barnard, Calvin Armstrong, Ricardo de la Fuente, Ted Keller, and Tom Peterson involved with the Space Shuttle Onboard Software Project for engaging in a collaboration with us to learn practical applications of SPC. We appreciated their enthusiasm, willingness, and openness in working with their data and with experimenting and learning with us. All of these folks encouraged our work and provided support in many ways for which we are very grateful.

Special thanks go to Peter Gordon, Helen Goldstein, Marty Rabinowitz, and Katherine Pizzuti at Addison Wesley Longman, Inc., and to copyeditor Jean Peck and compositor Rob Mauhar for their help in producing this book. We appreciated their advice and guidance in this process.

Finally, we want to recognize the support and enthusiasm of our respective spouses, Marilyn Florac and Dennis Carleton. Somehow, you knew just when we needed some words of encouragement, some quiet moments, or a cup of hot chocolate. Our deepest thanks for your patience, love, and understanding.

William A. Florac and Anita D. Carleton

1

Managing and Measuring Process Behavior

We can, however, no longer muddle through on intuition; the process is too complex and the products too important.

Watts S. Humphrey 1989

Every organization asks the question, Are we achieving the results we desire? This question is expressed in many ways: Are we meeting our business objectives? Are our customers satisfied with our products and services? Are we earning a fair return on our investment? Can we reduce the cost of producing these products or services? How can we improve the response to our customers' needs or increase the functionality of our products? How can we improve our competitive position? Are we achieving the growth required for survival? For many software organizations, the answers to questions like these are rooted in the relationships that exist within the organization's basic makeup, its overall philosophy, its people, its management, and the facilities and operational processes used by the organization to deliver its products and services.

The understanding and the appreciation of this relationship have not come easily to organizations in the software industry, be they in the defense sector or the commercial sector. After years of software product development characterized by missed schedules, overexpenditures, burnt-out engineers, and poor-quality products, competitive pressure and the increasing demand for more software products, and for more reliable ones, made it imperative to find a better way to produce software products and services. Initially, new technology innovations and improvements (languages, compilers, CASE tools, and so forth) were viewed as the "silver bullet." For the most part, the technology helped, but it did not provide the breakthrough desired.

In the meantime, other industries were also finding it necessary to find a better way to compete and produce quality products due to worldwide competition. They began to adopt the approach advocated by W. Edwards Deming and successfully implemented by many competitive industries in Japan. Deming's approach, greatly influenced by the work of Walter A. Shewhart, deals with the notions of process

management and continual improvement. Deming's approach contends that to be competitive, improve quality, and increase productivity, the following actions are required:

- Focus on the processes that generate the products and services to improve quality and productivity. Consider the task of building the product or providing the service as a series of integrated and interconnected processes.
- Ensure that the processes are properly supported.
- Manage poorly behaving processes by fixing the process, not blaming the people.
- Recognize that variation is present in all processes and that existence of variation is an opportunity for improvement. Improvement comes from reducing variation.
- Take variation into account in the decision-making process. Management action uses data from the process to guide decisions.

As you may surmise, the preceding "process-thinking" principles require a unique relationship within the organization, involving its management and management philosophy, its people, and the management of the processes used to produce products and provide services. These principles are embodied in Shewhart's continual improvement cycle [Shewhart 1939], popularized by Deming [Deming 1986], which has been characterized as "Plan, Do, Check (or Study), Act" (PDCA). It is an iterative learning cycle that allows one to continually improve processes by learning, evaluating and working on improvement one step at a time.

These notions of process thinking were adapted to the development of software by Watts S. Humphrey in his book *Managing the Software Process* [Humphrey 1989]. Humphrey related this process thinking to the software development process by constructing a framework of software process maturity levels having the objective of achieving a controlled and measured process as a foundation for continual improvement.

Over the past decade, the concepts, methods, and practices associated with process management and continual improvement have gained wide acceptance in the software community. These concepts, methods, and practices embody a way of thinking, a way of acting, and a way of understanding the data generated by processes that collectively result in improved quality, increased productivity, and competitive products.

The acceptance and the use of this process-thinking approach by much of the software industry have required many of those charged with collecting and analyzing software data to consider ways to measure software processes that are responsive to questions relating to process performance (such as effectiveness, efficiency, timeliness, predictability, improvements, and product quality) in quantitative terms. Traditional software measurement and analysis methods, those that provide status at a point in time and compare against the "plan," are not sufficient for determining past process performance or for predicting process performance.

Other disciplines have addressed this issue by using statistical process control methods, specifically using Shewhart's control charts. Managers and engineers

within the manufacturing, chemical, petroleum, food-processing, electronics, and aerospace industries have used control charts for many years. They have come to realize that control charts provide the basis for making process decisions and predicting process behavior.

However, there are those in the software community who look upon the use of control charts as an analysis tool unique to manufacturing or high-quantity, repetitive processes. Since they view the software development process as a highly human-centric, intellectual, design-oriented process, they reason that statistical process control and control charts are not applicable and cannot (or should not) be used to measure software processes. Needless to say, our experience does not support such reasoning and is, in fact, quite the contrary.

We have come to appreciate the value added when control charts are used to provide engineers and managers with quantitative insights into the behavior of their software development processes. In many ways, the control chart is a form of instrumentation–like an oscilloscope, or a temperature probe, or a pressure gauge–providing data to guide decisions and judgment by process-knowledgeable engineers and managers.

We recognize that statistical process control principles and control charts have received little use as of yet in most software organizations. Nevertheless, we refuse to let the current state of software measurement stop us from promoting their use. The benefits of the empirical methods associated with statistical process control (SPC) have been so evident in other development, production, and service activities that it would be foolish to ignore their potential for improving software products and services.

1.1 What Is a Software Process?

The term *process* means different things to different people, so it is important to clearly define what we mean when we use the word, especially in the context of a software development or support environment.

> *A process can be defined as the logical organization of people, materials, energy, equipment, and procedures into work activities designed to produce a specified end result.*
>
> Gabriel A. Pall 1987

Pall's definition of *process* is illustrated in Figure 1.1. It differs from the definitions of *process* and *software process* given in the Capability Maturity Model (CMM) for Software[1] in that it includes people, materials, energy, and equipment within the scope of *process*. Version 1.1 of the CMM, by way of contrast, views a process as a "sequence of steps" performed by people with the aid of tools and equipment to transform raw material into a product [Paulk 1993b, 1995].

[1] Capability Maturity Model and CMM are service marks of Carnegie Mellon University.

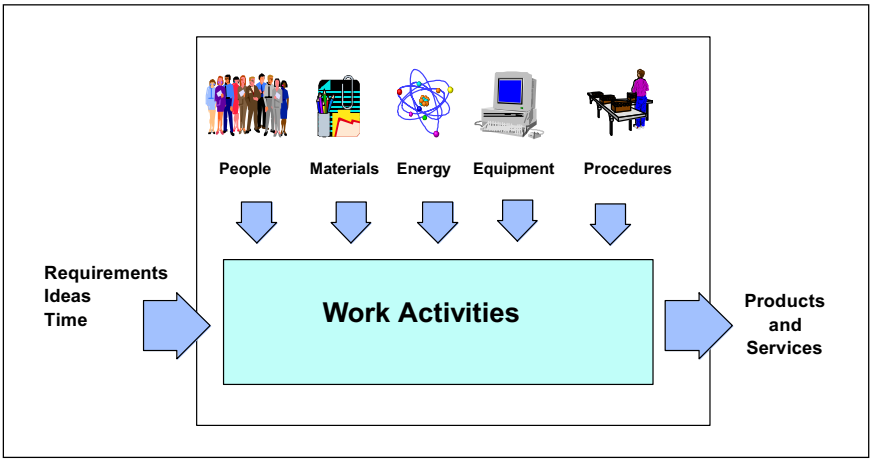


FIGURE 1.1 Definition of *Process*

Including people, materials, energy, and tools within the concept of process becomes important when we begin to apply the principles of statistical process control to improve process performance and process capability. Here, we use measures of variability to identify opportunities for improving the quality of products and the capability of processes. In searching for causes of unusual variation, it would be a mistake to exclude people, materials, energy, and tools from the scope of the investigation. This view of process, as enunciated by Pall, has been at the very heart of statistical process control since its founding in the 1920s [Shewhart 1931; Western Electric 1958; Juran 1988].

In keeping with the broad view of process, we use the term *software process* in this book to refer, not just to an organization's overall software process, but to any process or subprocess used by a software project or organization. In fact, a good case can be made that it is only at subprocess levels that true process management and improvement can take place. Thus, readers should view the concept of software process as applying to any identifiable activity that is undertaken to produce or support a software product or service. This includes planning, estimating, designing, coding, testing, inspecting, reviewing, measuring, and controlling, as well as the subtasks and activities that comprise these undertakings.

1.2 What Is Software Process Management?

Software process management is about successfully managing the work processes associated with developing, maintaining, and supporting software products and software-intensive systems. By successful management, we mean that the products and services produced by the processes conform fully to both internal and external customer requirements and that they meet the business objectives of the organization responsible for producing the products.

This concept of process management is founded on the principles of statistical process control. These principles hold that, by establishing and sustaining stable levels of variability, processes will yield predictable results. We can then say that the processes are *under statistical control*. This was first enunciated by Shewhart as follows:

> *A phenomenon will be said to be controlled when, through the use of past experience, we can predict, at least within limits, how the phenomenon may be expected to vary in the future.*
>
> Walter A. Shewhart 1931

Controlled processes are stable processes, and stable processes enable you to predict results. This, in turn, enables you to prepare achievable plans, meet cost estimates and scheduling commitments, and deliver required product functionality and quality with acceptable and reasonable consistency. If a controlled process is not capable of meeting customer requirements or other business objectives, the process must be improved or retargeted.

At the individual level, then, the objective of software process management is to ensure that the processes you operate or supervise are predictable, meet customer needs, and (where appropriate) are continually being improved. From the larger, organizational perspective, the objective of process management is to ensure that the same holds true for every process within the organization.

1.3 The Role of Software Process Management

In this section, we identify four key responsibilities of software process management and show how they relate to measurement of process performance. Our discussion of the activities and issues associated with the responsibilities will be neither all encompassing nor definitive, but it will provide a starting point from which process measurement methods and techniques can be developed with something more substantial than just abstract terms. The four responsibilities that are central to process management are as follows:

1. Define the process.
2. Measure the process.
3. Control the process (ensure that variability is stable so that results are predictable).
4. Improve the process.

These responsibilities are analogous to Shewhart's continual improvement cycle [Shewhart 1939], popularized by Deming [Deming 1986], and characterized as "Plan, Do, Check (or Study), Act" (PDCA). As an iterative learning cycle, it allows you to improve processes by learning, evaluating, and working on improvement in step-by-step fashion and, by continuing with this cycle, to reduce process variation.