David Marca

Applying Software Engineering Principles

Little, Brown Computer Systems Series

Library of Congress Cataloging in Publication Data

Marca, David.
Applying software engineering principles.

(Little, Brown computer systems series)
1. Electronic digital computers — Programming.
I. Title. II. Series.
QA76.6.M358 1983 001.64'2 83-7964
ISBN 0-316-54574-0

(Library of Congress Catalog Card No. 83-7964

ISBN 0-316-54574-0

9 8 7 6 5 4 3 2 1

MV

Published simultaneously in Canada by Little, Brown & Company (Canada) Limited
Printed in the United States of America

Preface

This book is designed for professional programmers and professional-minded students — that is, it is designed for individuals who develop, or intend to develop, software in a professional setting.

Software engineering is the act of an individual who learns to develop software in a practical setting.

Successful software developers seem to have the ability to grow from an existing base of knowledge. They frequently modify their work habits, applying past experiences to new job situations. This book presents software engineering principles that enable programmers to share the experiences of many other professionals.

Sources of Professional Learning

Novice programmers learn about tools and techniques from courses and evaluate their effectiveness by using them on the job. Novices advance to higher levels of professionalism by learning from others. In formal reviews and walkthroughs, they learn ways to apply tools and techniques. From working in small teams, they learn to observe the work habits of others and to evaluate new concepts by placing them in an open forum.

The best professionals learn the difficult and rewarding task of stepping back and observing their own work. Learning comes easier when you know yourself — you can strengthen your abilities, compensate for deficiencies, and identify areas for improvement.

The software engineering principles in this book were developed from all these levels of professional learning. Some were learned from books or schooling, some by watching others work, and some by reflecting on my own work. This is, therefore, a book of experience woven with facts; it is an act of sharing.

Software engineering principles are the *whys* of developing software; they state the reasons behind the programmer's day-to-day work.

Learning from This Book

This book is not overly concerned with syntax, nor with writing FORTRAN statements on coding forms, nor with drawing flowcharts of programs. *This is an* advanced programming book concerned with the nature of programming once the syntax is learned.

This book covers more than what might normally be considered relevant to an advanced programming course. It describes what you should be *thinking* about when you are programming — that is, it describes how you apply software engineering principles to make a computer language represent a solution to a problem.

The principles and associated examples given throughout the book describe the paradigms I think are the most important ones for the engineering of software. The software examples, written in Microsoft's FORTRAN for the TRS–80 microcomputer, can be implemented on almost any computer.

Software engineering principles generalize the ways of writing software.

These principles and examples will, perhaps, confirm the way you engineer software. Or they may show you a different way of looking at the programming process, describe a new concept for you, or lead you to write down principles of your own. I hope that you will learn as much about the engineering of software from reading this book as I have from writing it.

Why should *you* read this book? Because it will, hopefully, make you think about the way you engineer software. All of the principles and most of the discussions apply to any computer language. So, whether you program exclusively in FORTRAN or in twenty different languages, you should find much related to your work.

David Marca

Research & Development Group
SofTech, Inc.
Waltham, MA

April, 1983

Acknowledgments

The production of a book is never the work of one person. This book is no exception, for it is the synthesis of the knowledge and the experience of many in the computing field. The most appropriate acknowledgment of their work is the collection of references at the end of this book.

I am especially grateful to all the computing professionals I have worked with and have learned from during my career. During the past twelve years, they have openly shared their ideas, listened to my views, and never hesitated to tell me when I was wrong.

Thanks also goes to Jerry Weinberg for his help in directing my writing efforts when no one else could; and to Beth Anderson, developmental editor of the second manuscript, who helped me become a better writer.

Contents

Part II Engineering with a Computer Language 45

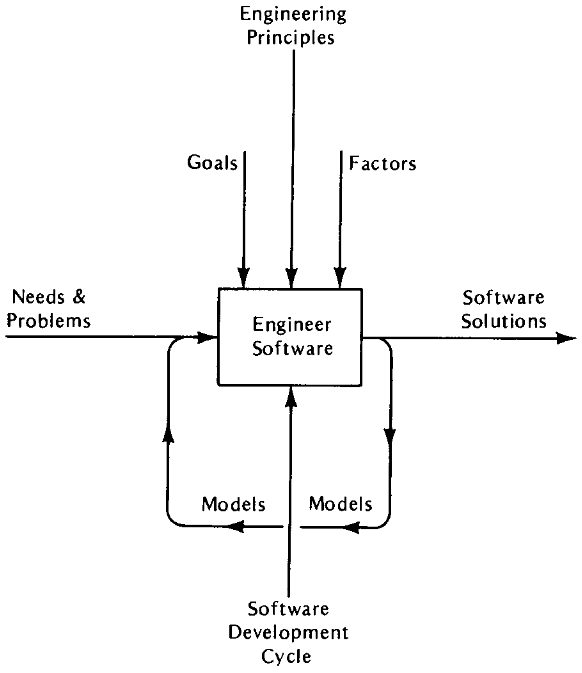
Part III Engineering with Existing Software 107

David Marca

Applying Software Engineering Principles

Little, Brown and Company
Boston Toronto

FIGURE I–1: A Model of Engineering Software



Part I

Important Aspects of Software Engineering

It is more important to understand about engineering than about coding. That is, software engineers are engineers first and builders of software second; they apply engineering concepts to their work.

Much work precedes the construction of software, though. Software engineers must establish the context for their work. And they must spend some time setting up their projects correctly.

Starting a project is hard, because you must

— define goals,
— formulate an approach to reach those goals, and
— assemble a project team to implement the approach.

To explain, making trade-offs among conflicting factors is a large part of setting goals. Also, defining cycles of software development and developing modeling guidelines help establish an approach. Finally, selecting each individual in the project team requires matching skills and personalities with the approach.

Mistakes made early in a project can jeopardize its success. So software engineers invest time in getting a good start, knowing that the success of the project may depend upon their first few decisions.

PRINCIPLE: Project Start-up
The hardest part of many projects is getting them started correctly.

1 Software Engineering Concepts

Engineering means reaching established goals, within the constraints of the real world, according to a plan. Software is the result of an engineering process in which developers use engineering concepts to build programs. These concepts can be summarized by a set of principles that can be deduced by starting with a very simple fact: Software is engineered in a practical setting [WGU82].

1.1 Practical Considerations During Software Development

Probably the most difficult aspects of building well-engineered software are identifying and assessing practical issues. Software engineers understand that they are performing an engineering activity. What separates software engineers from other programmers is their ability to make decisions with the practical issues in mind during all phases of software development.

There is a distinction between software science and software engineering. The distinction is that the results of engineering need only be useful, not perfect. Getting software to work well enough to be useful is the essence of software engineering [YE79]. But often something very useful cannot be built by proven methods. Software systems fall into this category. So software developers frequently must build systems without an exact science to back them up.

DEFINITION: *Software Engineering*
Software engineering builds useful software.

Today's software systems are considered products of engineering because they cannot be proven to operate correctly, yet they perform very useful functions. In other words, a software system need not be perfect to be useful. For example, the OS/360 project [BF75] comprised hundred of people who developed more than a million lines of code. During initial development of the software no one proved that each module operated correctly. Instead, test cases were used to verify that the operating system