Facts and Fallacies of Software Engineering

Robert L. Glass
Foreword by Alan M. Davis

Facts and Fallacies of Software Engineering

This page intentionally left blank

Facts and Fallacies of Software Engineering

Robert L. Glass

Addison-Wesley

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

Library of Congress Cataloging-in-Publication Data

Glass, Robert L., 1932–
Facts and fallacies of software engineering / Robert L. Glass.
p. cm.
Includes bibliographical references and index.
ISBN 0-321-11742-5 (alk. paper)
1. Software engineering I. Title.

QA76.758 .G52 2003
005.1'068'5--dc21 2002027737

ISBN 0-321-11742-5

Text printed in the United States on recycled paper at RR Donnelley Crawfordsville in Crawfordsville, Indiana.

9th Printing July 2008

This book is dedicated to the researchers
who lit the fire of software engineering and
to the practitioners who keep it burning.

This page intentionally left blank

Contents

This page intentionally left blank

Acknowledgments

To Paul Becker,
now of Addison-Wesley, who has been the editor for nearly all of my non-self-published books, for his belief in me over the years.

To Karl Wiegers,
for his contributions of frequently forgotten fundamental facts and for the massive job of reviewing and massaging what I wrote.

To James Bach, Vic Basili, Dave Card, Al Davis, Tom DeMarco, Yaacov Fenster, Shari Lawrence Pfleeger, Dennis Taylor, and Scott Woodfield,
for the hugely important task of helping me identify appropriate citations for the sources of these facts.

This page intentionally left blank

Foreword

When I first heard that Bob Glass was going to write this book and model it after my *201 Principles of Software Development,* I was a bit worried. After all, Bob is one of the best writers in the industry, and he would provide tough competition for my book. And then, when Bob asked me to write his foreword, I became even more worried; after all, how can I endorse a book that seems to compete directly with one of mine? Now that I have read *Facts and Fallacies of Software Engineering,* I am pleased and honored (and no longer worried!) to have the opportunity to write this foreword.

The software industry is in the same state of affairs that the pharmaceutical industry was in during the late nineteenth century. Sometimes it seems that we have more snake-oil salespeople and doomsayers than sensible folks practicing and preaching in our midst. Every day, we hear from somebody that they have discovered this great new cure for some insurmountable problem. Thus we have oft heard of quick cures for low efficiency, low quality, unhappy customers, poor communication, changing requirements, ineffective testing, poor management, and on and on. There are so many such pundits of the perfunctory that we sometimes wonder if perhaps some portion of the proclaimed panaceas are possibly practical. Whom do we ask? Whom in this industry can we trust? Where can we get the truth? The answer is Bob Glass.

Bob has had a history of providing us with short treatises on the many software disasters that have occurred over the years. I have been waiting for him to distill the common elements from these disasters so that we can benefit more easily from his many experiences. The 55 facts that Bob Glass discusses in this wonderful book are not just conjectures on his part. They are exactly what I have been

waiting for: the wisdom gained by the author by examining in detail the hundreds of cases he has written about in the past.

The 55 facts that follow are likely not to be popular with all readers. Some are in direct opposition to the so-called modern ways of doing things. For those of you who wish to ignore the advice contained within these covers, I can wish you only the safest of journeys, but I fear for your safety. You are treading on well-trod territory, known to be full of mines, and many have destroyed their careers trying to pass. The best advice I can give you is to read any of Bob Glass's earlier books concerning software disasters. For those of you who wish to follow the advice contained herein, you too are following a well-trod path. However, this path is full of successful testimonies. It is a path of awareness and knowledge. Trust Bob Glass because he has been there before. He has had the privilege of analyzing his own successes and failures along with hundreds of others' successes and failures. Stand on his shoulders, and you will more likely succeed in this industry. Ignore his advice, and be prepared for Bob to call you in a few years to ask you about your project—to add it to his next compilation of software disaster stories.

Alan M. Davis
Spring 2002

Author's Addendum:
I tried to get Al Davis to tone down this foreword. It is, after all, a bit syrupy sweet. But he resisted all of my efforts. (I really *did* try! Honest!) In fact, in one such exchange, he said, "You deserve to be on a pedestal, and I'm happy to help you up!" My experience with being on pedestals is that, inevitably, you fall off, and when you do, you break into Humpty-Dumpty-like insignificant fragments.

But regardless of all that, I cannot imagine greater and more wonderful sentiments than the ones Al bestows on me here. Thanks!

Robert L. Glass
Summer 2002

PART 1

55 FACTS

This page intentionally left blank

Introduction

This book is a collection of facts and fallacies about the subject of software engineering.

Sounds boring, doesn't it? A laundry list of facts and fallacies about building software doesn't sound like the kind of thing you'd like to kick back and spend an hour or two with. But there's something special about these facts and fallacies. They're fundamental. And the truth that underlies them is frequently forgotten. In fact, that's the underlying theme of this book. A lot of what we ought to know about building software we don't, for one reason or another. And some of what we think we know is just plain wrong.

Who is the *we* in that previous paragraph? People who build software, of course. We seem to need to learn the same lessons over and over again, lessons that these facts—if remembered—might help us avoid. But by *we* I also mean people who do research about software. Some researchers get mired so deeply in theory that they miss some fundamentally important facts that might turn their theories upside-down.

So the audience for this book is anyone who's interested in building software. Professionals, both technologists and their managers. Students. Faculty. Researchers. I think, he said immodestly, that there's something in this book for all of you.

Originally, this book had a cumbersome, 13-word title: Fifty-Five Frequently Forgotten Fundamental Facts (and a Few Fallacies) about Software Engineering was, well, excessive—or at least those responsible for marketing this book thought so. So cooler heads prevailed. My publisher and I finally settled on Facts and Fallacies of Software Engineering. Crisp, clear—and considerably less colorful!

I had tried to shorten the original long title by nicknaming it the F-Book, noting the alliteration of all the letter *F*s in the title. But my publisher objected, and I suppose I have to admit he was right. After all, the letter F is probably the only dirty letter in our alphabet (*H* and *D* have their advocates, also, but *F* seems to reach another level of dirtiness). So the F-Book this is not. (The fact that an early computer science book on compiler-writing was called the Dragon Book, for the sole reason that someone had [I suppose arbitrarily] put the picture of a dragon on its cover, didn't cut any ice in this particular matter.)

But in my defense, I would like to say this: Each of those F-words was there for a purpose, to carry its weight in the gathering meaning of the title. The *55*, of course, was just a gimmick. I aimed for 55 facts because that would add to the alliteration in the title. (Alan Davis's wonderful book of 201 principles of software engineering was just as arbitrary in its striving for 201, I'll bet.) But the rest of the *F*s were carefully chosen.

Frequently forgotten? Because most of them are. There's a lot of stuff in here that you will be able to say "oh, yeah, I remember that one" and then muse about why you forgot it over the years.

Fundamental? The primary reason for choosing this particular collection of facts is because all of them carry major significance in the software field. We may have forgotten many of them, but that doesn't diminish their importance. In fact, if you're still wondering whether to go on reading this book, the most important reason I can give you for continuing is that I strongly believe that, in this collection of facts, you will find the most fundamentally important knowledge in the software engineering field.

Facts? Oddly, this is probably the most controversial of the words in the title. You may not agree with all of the facts I have chosen here. You may even violently disagree with some of them. I personally believe that they all represent fact, but that doesn't mean you have to.

A few fallacies? There are some sacred cows in the software field that I just couldn't resist skewering! I suppose I have to admit that the things I call fallacies are things that others might call facts. But part of your fun in reading this book should be forming your own opinion on the things I call facts—and the things I call fallacies.

How about the age of these facts and fallacies? One reviewer of this book said that parts of it felt dated. Guilty as charged. For facts and fallacies to be forgotten frequently, they must have been around for awhile. There are plenty of golden oldies in this collection. But here I think you will find some facts and fallacies that

will surprise you, as well—ideas that are "new" because you're not familiar with them. The point of these facts and fallacies is not that they are aged. It's that they are ageless.

In this part of the book, I want to introduce the facts that follow. The fallacies will have their own introduction later in the book. My idea of an introduction is to take one last trip through these 55 frequently forgotten fundamental facts and see how many of them track with all of those F-words. Putting on my objectivity hat, I have to admit that some of these facts aren't all that forgotten.

- Twelve of the facts are simply little known. They haven't been forgotten; many people haven't heard of them. But they are, I would assert, fundamentally important.
- Eleven of them are pretty well accepted, but no one seems to act on them.
- Eight of them are accepted, but we don't agree on how—or whether—to fix the problems they represent.
- Six of them are probably totally accepted by most people, with no controversy and little forgetting.
- Five of them, many people will flat-out disagree with.
- Five of them are accepted by many people, but a few wildly disagree, making them quite controversial.

That doesn't add up to 55 because (a) some of the facts could fit into multiple categories, and (b) there were some trace presences of other categories, like "only vendors would disagree with this." Rather than telling you which facts fit into which of those categories, I think I'll let you form your own opinion about them.

There's controversy galore in this book, as you can see. To help deal with that, following each discussion about a fact, I acknowledge the controversies surrounding it. I hope, by doing that, I will cover your viewpoint, whether it matches mine or not, and allow you to see where what you believe fits with what I believe.

Given the amount of controversy I've admitted to, it probably would be wise of me to tell you my credentials for selecting these facts and engaging in this controversy. (There's a humorous bio in the back of the book, so here I'll make it quick.) I've been in the software engineering field for over 45 years, mostly as a technical practitioner and researcher. I've written 25 books and more than 75 professional papers on the subject. I have regular columns in three of the leading