



Facts and Fallacies of Software Engineering



Robert L. Glass

Foreword by Alan M. Davis

Facts and Fallacies of Software Engineering

Facts and Fallacies of Software Engineering

Table of Contents

Contents

Acknowledgments

Foreword

Part I: 55 Facts

Introduction

Chapter 1 About Management

People

Fact 1. The most important factor in software work is the quality of the programmers

Fact 2. The best programmers are up to 28 times better than the worst programmers

Fact 3. Adding people to a late project makes it later

Fact 4. The working environment has a profound impact on productivity and quality

Tools and Techniques

Fact 5. Hype (about tools and techniques) is the plague on the house of software

Fact 6. New tools and techniques cause an initial loss of productivity/quality

Fact 7. Software developers talk a lot about tools, but seldom use them

Estimation

Fact 8. One of the two most common causes of runaway projects is poor estimation

Fact 9. Software estimation usually occurs at the wrong time

Fact 10. Software estimation is usually done by the wrong people

Fact 11. Software estimates are rarely corrected as the project proceeds

Fact 12. It is not surprising that software estimates are bad. But we live and die by them anyway!

Fact 13. There is a disconnect between software management and their programmers

Fact 14. The answer to a feasibility study is almost always yes

Reuse

Fact 15. Reuse-in-the-small is a well-solved problem

Fact 16. Reuse-in-the-large remains a mostly unsolved problem

Fact 17. Reuse-in-the-large works best in families of related systems

Table of Contents

Fact 18. Reusable components are three times as hard to build and should be tried out in three settings

Fact 19. Modification of reused code is particularly error-prone

Fact 20. Design pattern reuse is one solution to the problems of code reuse

Complexity

Fact 21. For every 25 percent increase in problem complexity, there is a 100 percent increase in solution complexity

Fact 22. Eighty percent of software work is intellectual. A fair amount of it is creative. Little of it is clerical

Chapter 2 About the Life Cycle

Requirements

Fact 23. One of the two most common causes of runaway projects is unstable requirements

Fact 24. Requirements errors are the most expensive to fix during production

Fact 25. Missing requirements are the hardest requirements errors to correct

Design

Fact 26. Explicit requirements explode as implicit (design) requirements for a solution evolve

Fact 27. There is seldom one best design solution to a software problem

Fact 28. Design is a complex, iterative process. Initial design solutions are usually wrong and certainly not optimal

Coding

Fact 29. Designer primitives (solutions programmers can readily code) rarely match programmer primitives

Fact 30. COBOL is a very bad language, but all the others (for business applications) are so much worse

Error Removal

Fact 31. Error removal is the most time-consuming phase of the life cycle

Testing

Fact 32. Software is usually tested at best at the 55 to 60 percent (branch) coverage level

Fact 33. One hundred percent coverage is still far from enough

Fact 34. Test tools are essential, but many are rarely used

Fact 35. Test automation rarely is. Most testing activities cannot be automated

Fact 36. Programmer-created, built-in debug code is an important supplement to testing tools

Reviews and Inspections

Table of Contents

Fact 37. Rigorous inspections can remove up to 90 percent of errors before the first test case is run

Fact 38. Rigorous inspections should not replace testing

Fact 39. Postdelivery reviews (some call them retrospectives) are important and seldom performed

Fact 40. Reviews are both technical and sociological, and both factors must be accommodated

Maintenance

Fact 41. Maintenance typically consumes 40 to 80 percent of software costs. It is probably the most important life cycle phase of software

Fact 42. Enhancements represent roughly 60 percent of maintenance costs

Fact 43. Maintenance is a solution, not a problem

Fact 44. Understanding the existing product is the most difficult task of maintenance

Fact 45. Better methods lead to more maintenance, not less

Chapter 3 About Quality

Quality

Fact 46. Quality is a collection of attributes

Fact 47. Quality is not user satisfaction, meeting requirements, achieving cost and schedule, or reliability

Reliability

Fact 48. There are errors that most programmers tend to make

Fact 49. Errors tend to cluster

Fact 50. There is no single best approach to software error removal

Fact 51. Residual errors will always persist. The goal should be to minimize or eliminate severe errors

Efficiency

Fact 52. Efficiency stems more from good design than good coding

Fact 53. High-order language code can be about 90 percent as efficient as comparable assembler code

Fact 54. There are tradeoffs between size and time optimization

Chapter 4 About Research

Fact 55. Many researchers advocate rather than investigate

Part II: 5+5 Fallacies

Introduction

Chapter 5 About Management

Table of Contents

Fallacy 1. You cant manage what you cant measure

Fallacy 2. You can manage quality into a software product

People

Fallacy 3. Programming can and should be egoless

Tools and Techniques

Fallacy 4. Tools and techniques: one size fits all

Fallacy 5. Software needs more methodologies

Estimation

Fallacy 6. To estimate cost and schedule, first estimate lines of code

Chapter 6 About the Life Cycle

Testing

Fallacy 7. Random test input is a good way to optimize testing

Reviews

Fallacy 8. Given enough eyeballs, all bugs are shallow

Maintenance

Fallacy 9. The way to predict future maintenance costs and to make product replacement decisions is to look at past cost data

Chapter 7 About Education

Fallacy 10. You teach people how to program by showing them how to write programs

Conclusions

About the Author

Index