
Software Testing and Maintenance

Introduction

Jeff Offutt

2018

“Traditional” Quality Attributes (1980s)

1. Efficiency of process (time-to-market)
2. Efficiency of execution (performance)

**This is what we teach is important in
undergraduate computer science classes ...**

It was true .. in 1980

Modern Quality Attributes

1. Reliability
2. Usability
3. Security
4. Availability
5. Scalability
6. Maintainability
7. Performance & Time to market



**Why the
change ?**

**All of these factors
(sometimes called “-ilities”)
are important in the 2000s**

Based on an informal survey of around a dozen web software development managers, 2000.

Software Projects in the 1960s

- In the 1960s we built tiny *log cabins* ...
- Single-programmer
- Not much complexity
- No process needed
- Design could be kept in short term memory



Software Projects in the 1970s

- In the 1970s we built bigger **houses** ...
- Still **single**-programmer – focus on **algorithms** and **programming**
- A little **more complex**
- We had to start **thinking** harder
- The lack of process led to some **disasters**
- For most of the industry, **quality** did not affect the bottom line
- But **costs** were starting to **increase** ...



Software Projects in the 1980s

- In the 1980s we built *office buildings* ...
- We needed teamwork – and communication
- A lot more complex – data abstraction
- We needed to write down requirements and design
- Poor process and ignorance of need for process created spectacular failures
- We no longer had the skills and knowledge for successful engineering



Software Projects in the 1990s

- In the 1990s we built **skyscrapers** ...
- We needed **more** than teamwork and communication
- We needed totally **new technologies** – languages, modeling techniques, processes
- Software development **changed** completely
- New languages (Java, UML, etc) led to **revolutionary procedures**
- Education fell **behind** ...



Software Projects in the 2000s

- In the 2000s we build integrated collections of continuously *evolving cities* ...
- Algorithm design and programming is no longer the primary focus of software development
- CS education fell so far behind it is almost obsolete
- New applications (web, embedded) is making quality crucial
- Developers learn more from training courses than they did in college
- Not much new development



Pace of Change is Exhilarating

- We have gone from ...
 - Log cabins ... to houses ... to office buildings ...to skyscrapers ... to building the most complicated engineering systems in human history
- In just half a career !!
- Civil engineers took thousands of years for this kind of change
 - And the most complicated civil engineering products pale in comparison the complexity of a modern IT system
- Electrical engineers took a couple of centuries

No way researchers, educators, or engineers could keep up !

Theory, Practice and Education

- What have you learned in college ?

How to build houses

- General software engineering courses, such as SE introduce a few concepts about buildings

The way we build software has changed dramatically since the CS curriculum stabilized in 1980 !!!!

- Very little new development is being done
- Maintenance ... evolution ... re-engineering ... maintainability ... being “agile”

What Can You Do ?

- As a **developer** ...
 - Program very **neatly**
 - **Design** to make change easy
 - Follow **processes** that make change easy
- As a **professional** ...
 - **Listen** to your colleagues when they teach you things you didn't learn in college
 - Take **training classes** eagerly (in the next 20 years, you should spend more time in training than you spent in college CS courses)
 - Further your **education** (MS degree)

Goals of This Class

1. **Reliability / Testing**
2. Usability
3. Security
4. Availability
5. Scalability
6. **Maintainability**
7. Performance & Time to market

Current Reality

- Most software development is actually maintenance
- Maintenance is no longer as boring as it was in the 1980s
- “We have as many testers as we have developers. And developers spend half their time testing. We’re more of a testing organization than we’re a software organization.”
 - Bill Gates of Microsoft

This class teaches modern methods for the two dominant portions of software development