

CSE 464 Software Quality Assurance and Testing

Introduction to Software Quality

Xusheng Xiao

Associate Professor
School of Computing and Augmented Intelligence
Arizona State University

What is Software?

- Computer **programs, procedures**, and possibly **associated documentation** and **data** pertaining to the **operation of a computer system**.

[IEEE 610.12 Standard Glossary of Software Engineering Terminology]

- Software is not just the **programs** but also all **associated documentation** and **configuration data** that is needed to make these **programs operate correctly**. [Sommerville]

Software Types

- System software: software designed to facilitate the operation and maintenance of a computer system and its associated programs
 - E.g., operating systems, assemblers, utilities
- Support software: software that aids in the development or maintenance of other software
 - E.g., compilers, loaders, and other utilities
- Application software: software designed to fulfill specific needs of a user
 - E.g., software for navigation, payroll, or process control

System Software



[IEEE 610.12 Standard Glossary of Software Engineering Terminology]

What about Software Engineering?

- (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1). [IEEE 610.12 Standard Glossary of Software Engineering Terminology]
- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification to maintaining the system after it has gone into use.

1968 NATO Conference



SOFTWARE ENGINEERING

Report on a conference sponsored by the
NATO SCIENCE COMMITTEE
Garmisch, Germany, 7th to 11th October 1968

Chairman: Professor Dr. F. L. Bauer

Co-chairmen: Professor L. Bolliet, Dr. H. J. Helms

Editors: Peter Naur and Brian Randell

- Defined best practices for software grounded in the application of engineering.
- Played a major role in gaining general acceptance for the term software engineering.

One Person vs. Multi-Person Development



Programming

VS



Software Engineering

[Parnas, 1975]

Single Version vs Multi-Version Programs



Programming

vs



Software Engineering

[Parnas, 1975]

Software Engineering

- *“A discipline that deals with the building of software systems which are so large that they are built by a team or teams of engineers.”*
[Ghezzi, Jazayeri, Mandrioli]
- *“Multi-person construction of multi-version software.”* [Parnas]

Software Engineering

- *“A discipline whose aim is the production of fault-free software, delivered on-time and within budget, that satisfies the user’s needs. Furthermore, the software must be easy to modify when the user’s needs change.” [Schach]*
- *“Difficult.” [van der Hoek]*

Key Points

- Systematic application of process
 - “sound engineering principles”
 - “the application of a systematic, disciplined, quantifiable approach”
 - “applying scientific knowledge”
- Practical, economic value
 - “obtain economically software”
 - “creates practical, cost-effective solutions”
 - “developing software systems in the service of mankind”
- Quality
 - “software that is reliable and works efficiently”

The Mythical Man-Month

by Fred Brooks (I)

- Published 1975, Republished 1995
- Experience managing the development of OS/360 in 1964-65
- Central Argument
 - Large programming projects suffer management problems different in kind than small ones, due to division of labor.
 - Critical need is the preservation of the conceptual integrity of the product itself.

Fred Brooks, 1999 Turing Award Winner

The Mythical Man-Month by Fred Brooks (II)

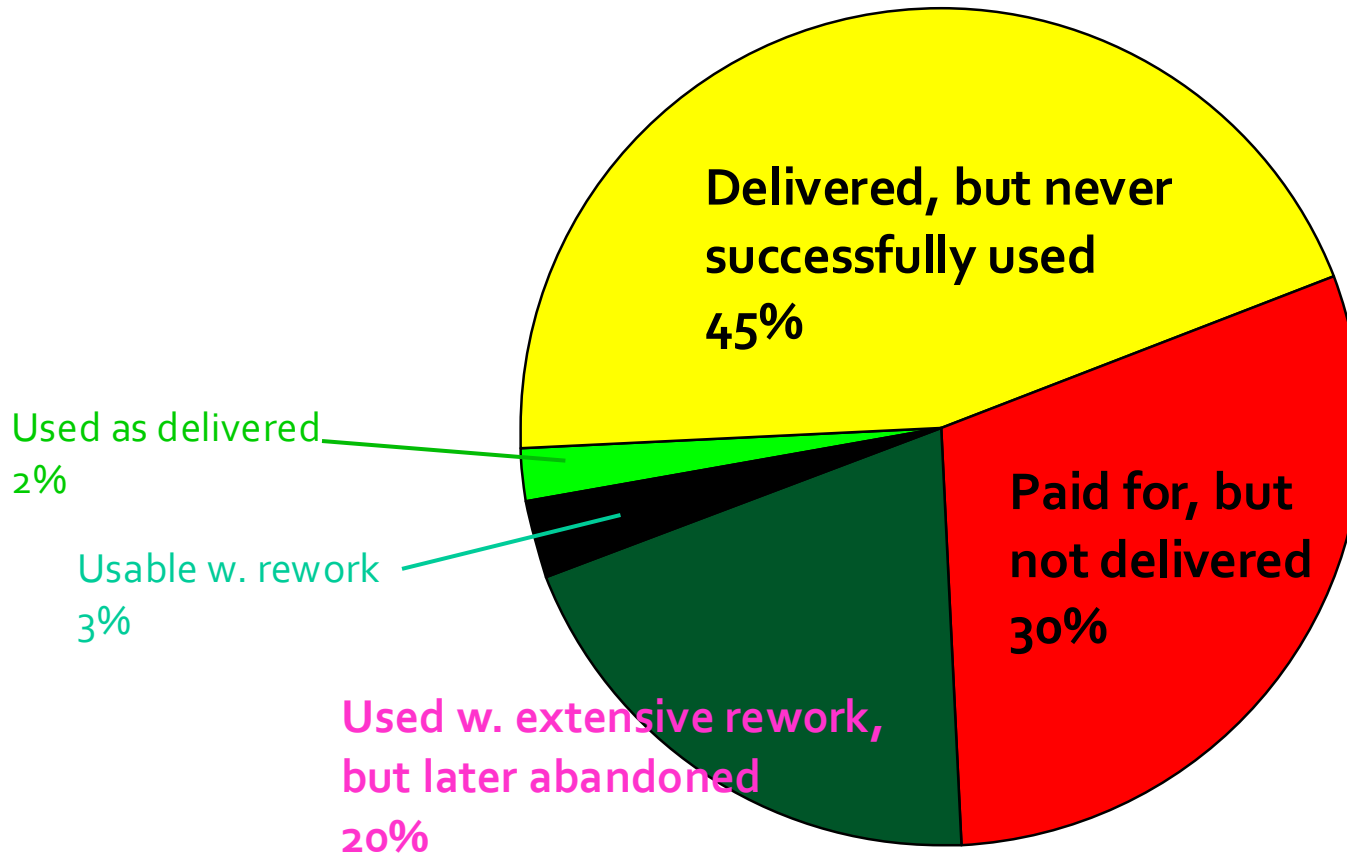
- Central Conclusions
 - Conceptual integrity achieved through exceptional designer
 - Implementation achieved through well-managed effort
 - Brooks's Law: Adding personnel to a late project makes it later

No Silver Bullet by Fred Brooks

- **Essence:** the difficulties inherent in the nature of the software
- **Accidents:** those difficulties that today attend its production but that are not inherent
- **Solution (?):** Grow Great Designers

Why Software Engineering?

*9 software projects totaling \$96.7 million: Where The Money Went
[Report to Congress, Comptroller General, 1979]*



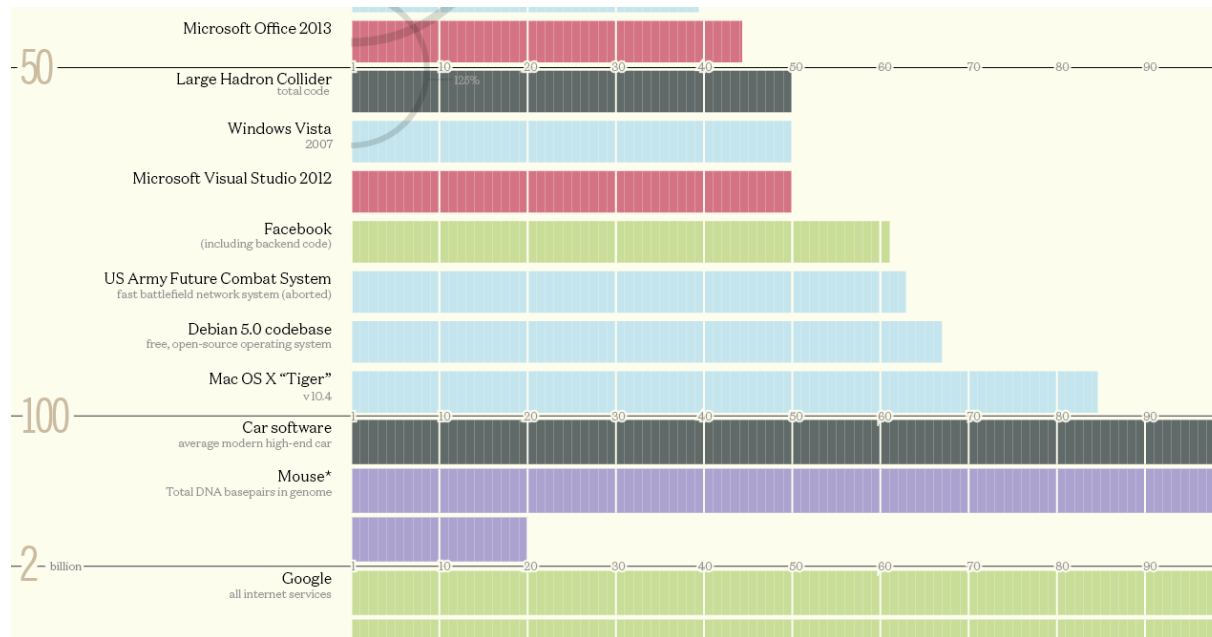
Take a look at the Standish Report (The "Chaos" Report)

Another Useful Definition

- Software engineering is a discipline that integrates:
 - Process
 - Provides a framework for software development
 - Methods
 - Provide “how to’s” for building software
 - Tools
 - Provide automated or semi-automated support for the process and the methods

Software Complexity

- Large software systems are among the most complex artifacts ever produced by man.

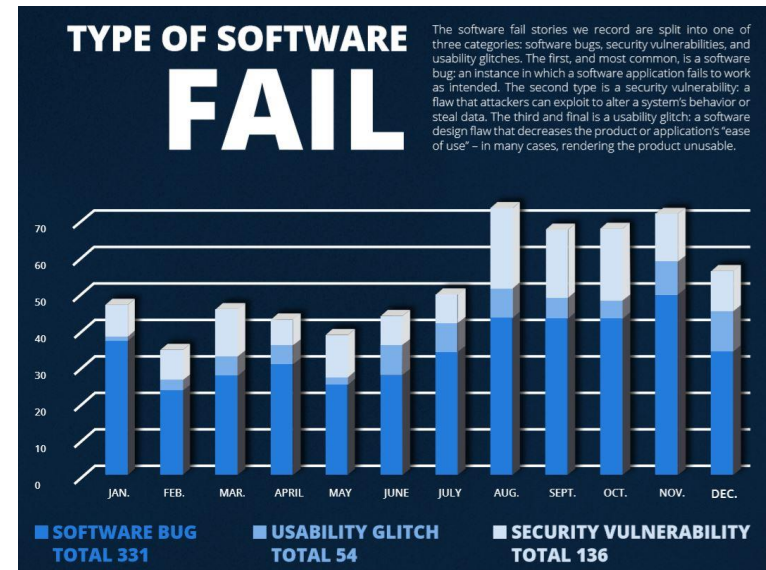


<https://www.visualcapitalist.com/millions-lines-of-code/>

- Examples: The Windows 10 binaries are about 3GB in size. Google (all services) consists of ~ 2 billion SLOC.

Software Quality Matters

- Software failure caused **\$1.7 trillion** in financial losses in 2017
- Software failures are mostly caused by **bugs** and **vulnerabilities**



<https://www.globenewswire.com/news-release/2018/01/24/1304535/0/en/Tricentis-Software-Fail-Watch-Finds-3-6-Billion-People-Affected-and-1-7-Trillion-Revenue-Lost-by-Software-Failures-Last-Year.html>



What is Software Quality?

- Answer to this question depends on whom you ask the question, under what circumstance, for what kind of software systems ...so on.



How do You Compare the Quality of These Cars?

- Different stake-holders of the software system are looking for different quality factors.

Watts S. Humphrey's View of Better Software



Good software, in Humphrey's view, "is usable, reliable, defect free, cost effective and maintainable and software now is none of those things."

You can't take something out of the box and know it's going to work."

Many consider him as the "father of [software quality](#)."

Software Functional Quality

- Software functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications.

Learning from history: The case of Software Requirements Engineering – Requirements Engineering Magazine.
Retrieved 2021-02-25.

- It can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile product.

Pressman, Roger S. (2005). Software Engineering: A Practitioner's Approach (Sixth International ed.). McGraw-Hill Education. p. 388. ISBN 0071267824.



NEW YORK (CNNMoney)

When it comes to lethal bugs, the computer glitch that set fire to \$440 million of Knight Capital Group's funds last Wednesday ranks right up there with the tsetse fly.

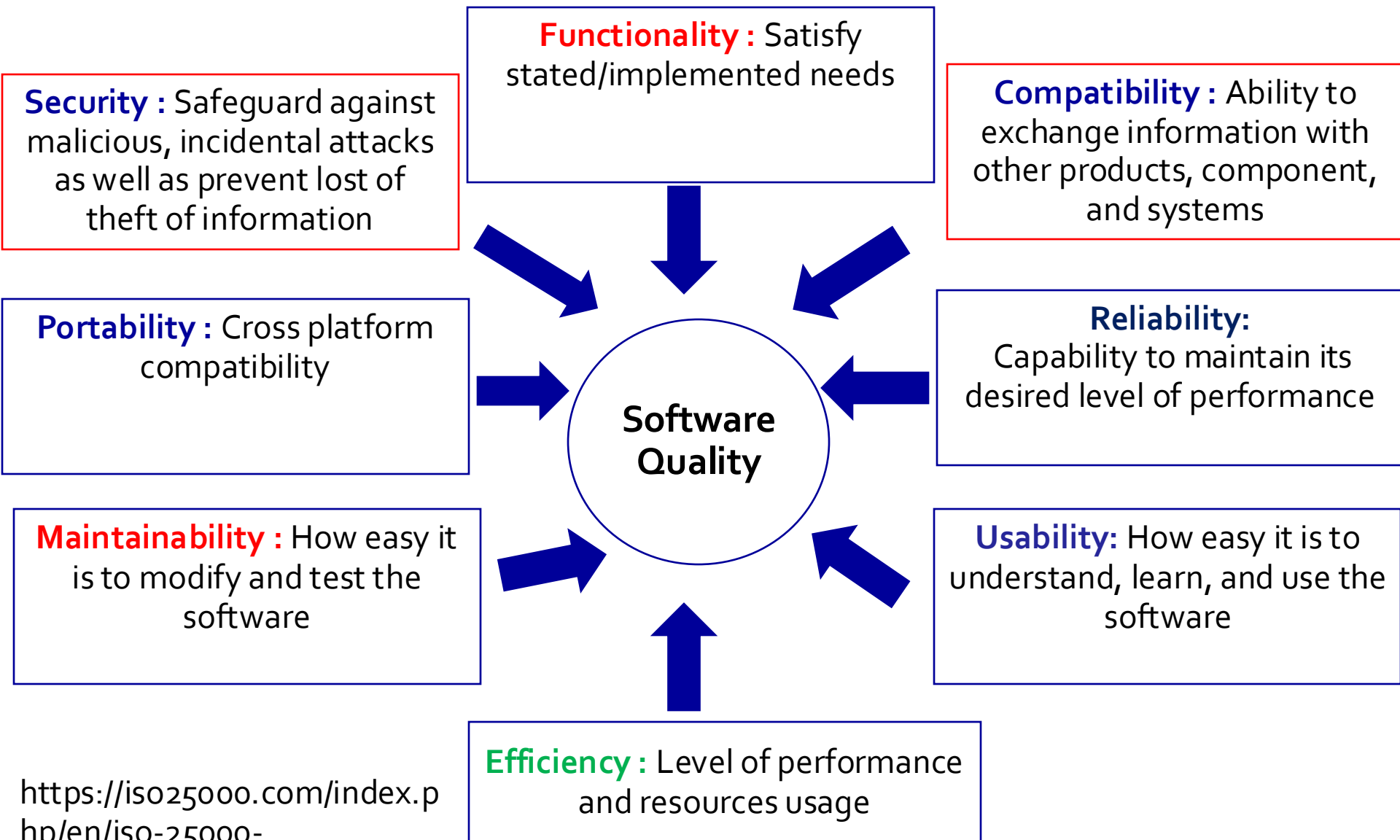
Software Structural Quality

- **Software structural quality** refers to how software meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability
- **Non-functional requirements or NFRs** are a set of specifications that describe the system's operation capabilities and constraints and attempt to improve its functionality.
- Types of Non-functional Requirement : Scalability, Reliability, Regulatory, Maintainability, Serviceability, Utility, Security, Manageability, Data integrity, Capacity, Availability, Usability, Interoperability, Environmental
- Security - **critical** non-functional software quality

4. People willingly downloaded more than 2 billion mobile apps that steal their personal data

Attackers used social media threats and mobile apps, not just email, to trick users into infecting their own systems. One in five clicks on malicious URLs occurred off the network, many of them from social media and mobile devices. Malicious mobile apps are no longer corner cases—they're real-world threats. Our analysis of authorized Android app stores discovered more than 12,000 malicious mobile apps— capable of stealing information, creating backdoors, and other functions—accounting for more than 2 billion downloads.

ISO - 25010



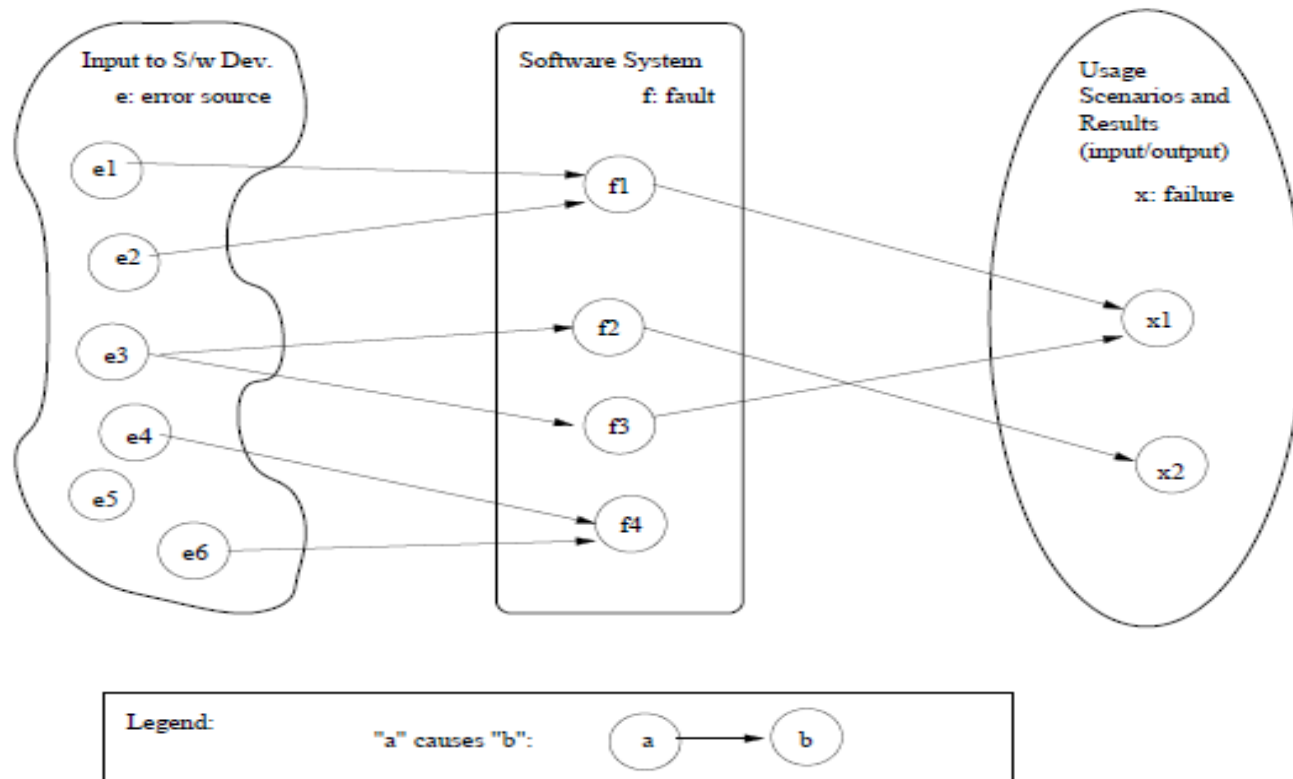
<https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>

Software Defects and Quality

Defect (or Bug): is a problem with the software, either with its external behavior or with its external characteristics.

Correctness, Defects (Bugs) and Quality

Good Quality = > Less Defects



- Relations: errors \Rightarrow faults \Rightarrow failures

Error, Fault, and Failure

Error : A human action that produces an incorrect result

Fault: An incorrect step, process, or data definition in a program/ design/other software engineering artifact.

Failure: The inability of a system or a component to perform its required functions within specified performance requirements

Defect (or Bug): is a problem with the software, either with its external behavior or with its external characteristics. **Defect is an instance of a failure.**

```
if(x < 5)
cout<< "X is 5 or less;
else
cout<<"X is greater than 5";
```

Mistake, Fault, Error, Failure

What does *Bug* mean in "*Bug Report*"?

Developer makes a **mistake**.

Fault (defect, bug) introduced in the software.

Fault remains undetected during testing

i.e., run **test inputs** & **check outputs**

The program **fails** (based on **test oracles**) during execution

i.e., it behaves unexpectedly

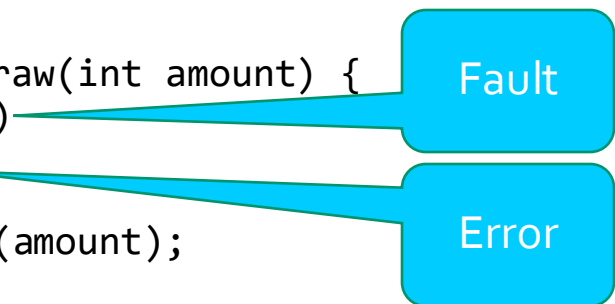
Error: difference between (1) computed, observed, or measured value/condition &
(2) true, specified, or theoretically correct value/condition

Where is Fault, Error, or Failure?

Example

- Should not allow withdrawal when there is a balance of 100 or less

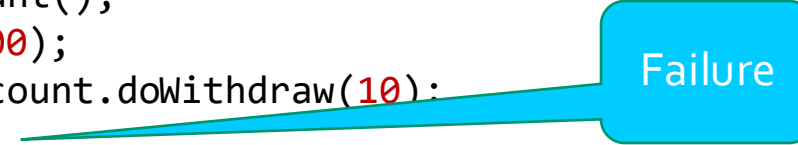
```
public boolean doWithdraw(int amount) {  
    if (Balance < 100)  
        return false;  
    else  
        return withDraw(amount);  
}
```



Fault

Error

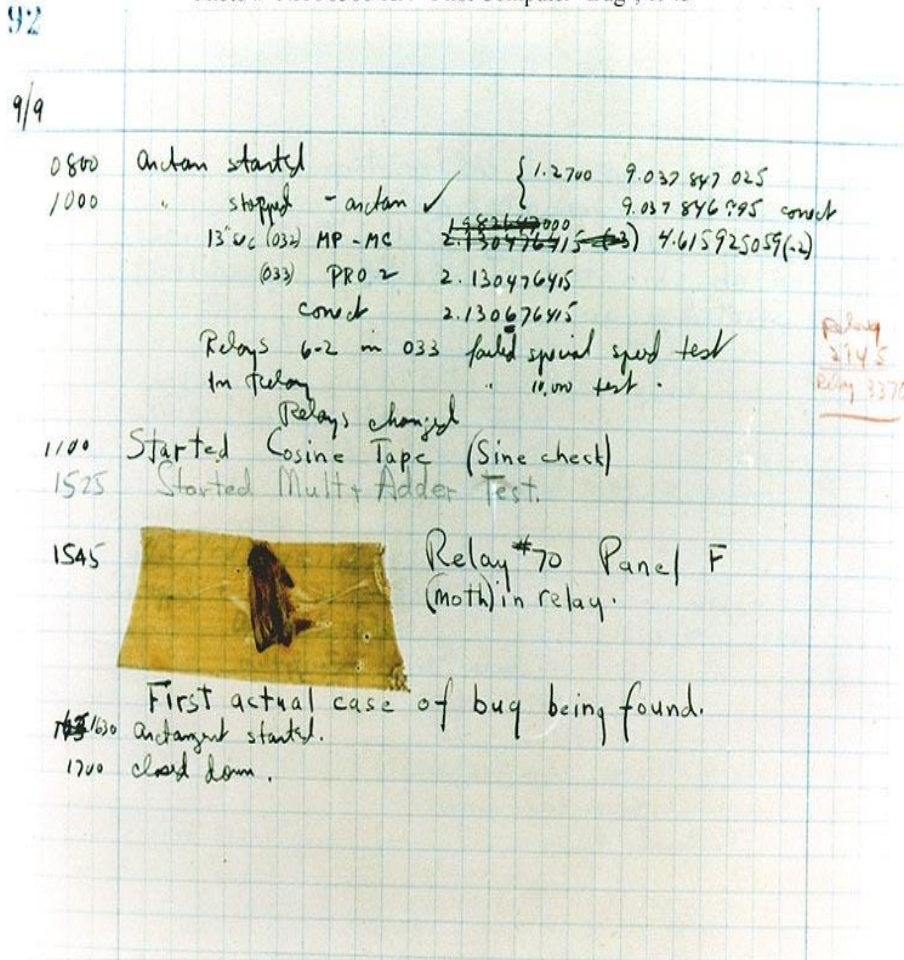
```
void testWithdraw() {  
    Account a = new Account();  
    Account.setBalance(100);  
    boolean success = Account.doWithdraw(10);  
    assertTrue(!success);  
}
```



Failure

The First Computer Bug?

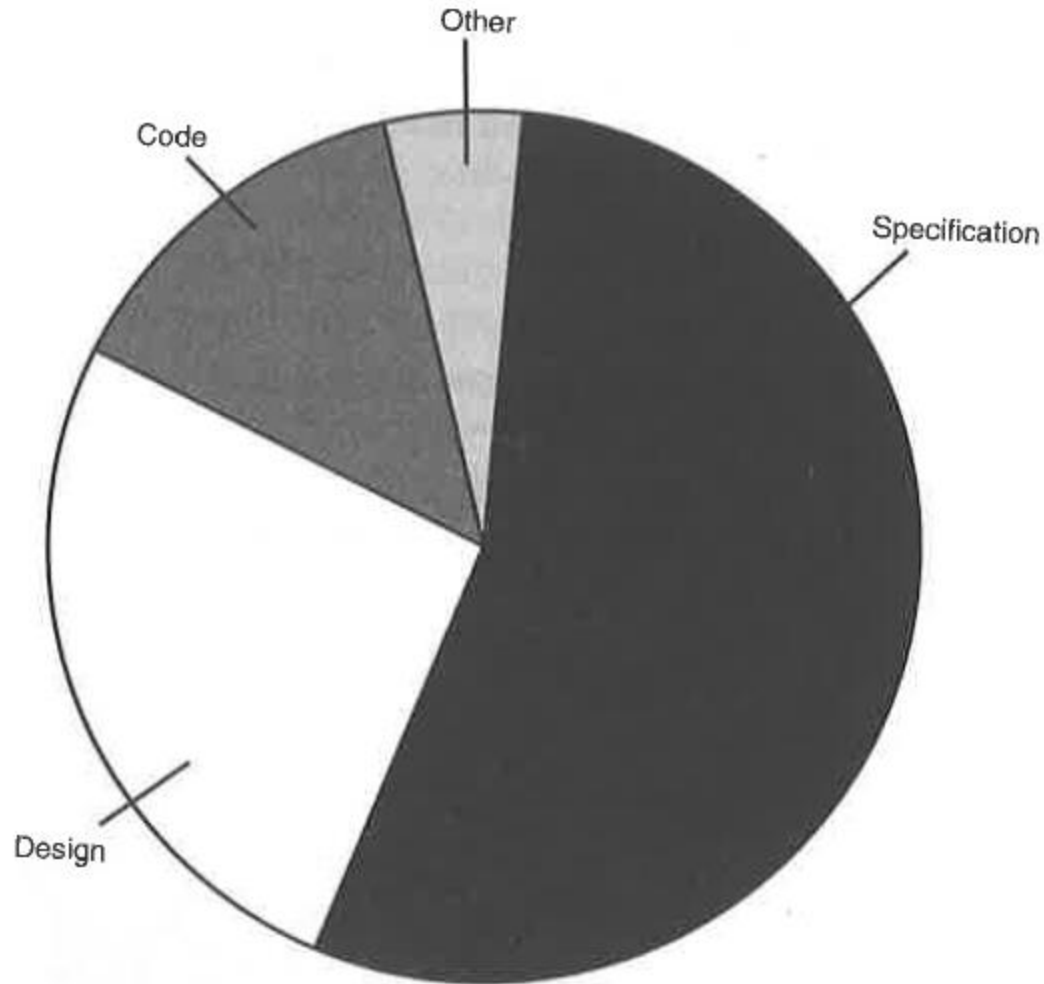
Photo # NH 96566-KN First Computer "Bug", 1945



- In 1947, Grace Murray Hopper was working on the Harvard University Mark II Aiken Relay Calculator (a primitive computer).
- On the 9th of September, 1947, when the machine was experiencing problems, an investigation showed that there was a **moth trapped between the points of Relay #70, in Panel F.**
- The operators removed the moth and affixed it to the log. The entry reads: "First actual case of bug being found."
- The word went out that they had "debugged" the machine and the term "debugging a computer program" was born.

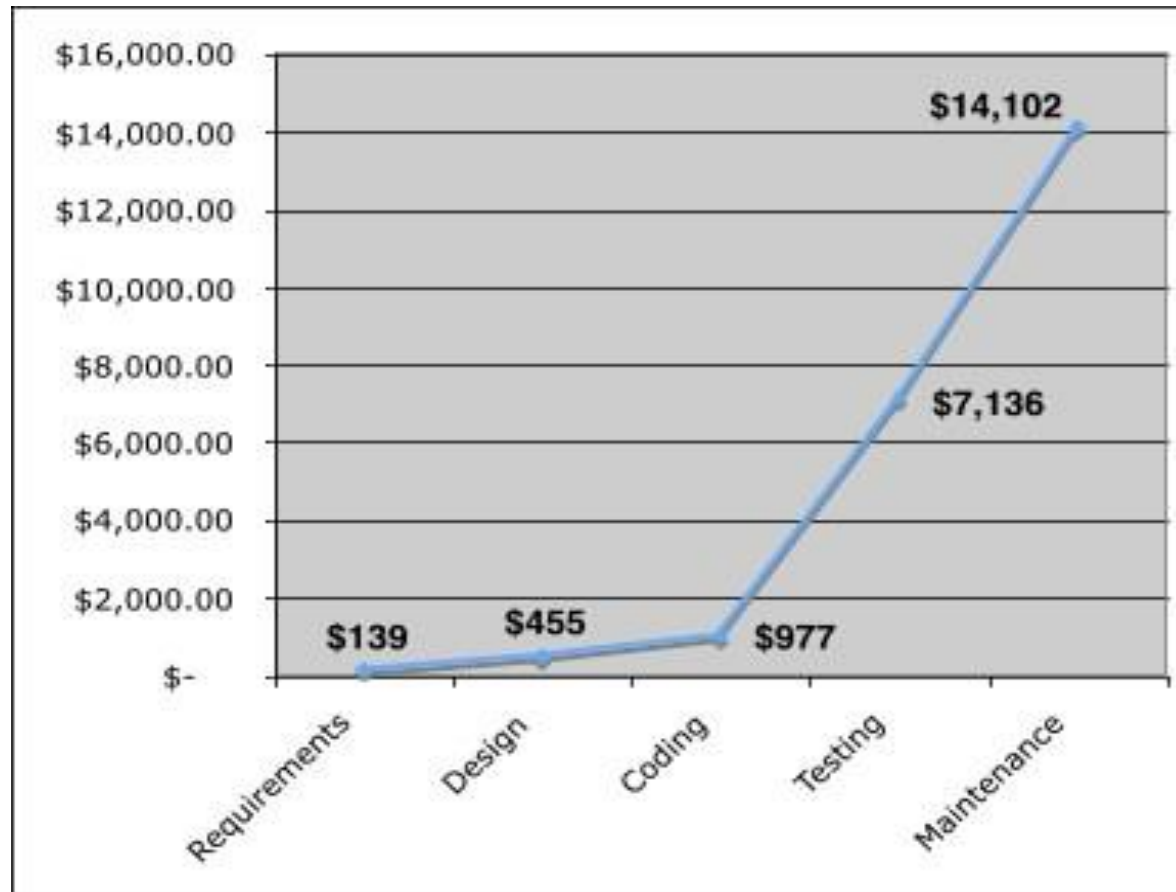
<http://www.history.navy.mil/photos/images/h96000/h96566kc.htm>

Bugs are coming from



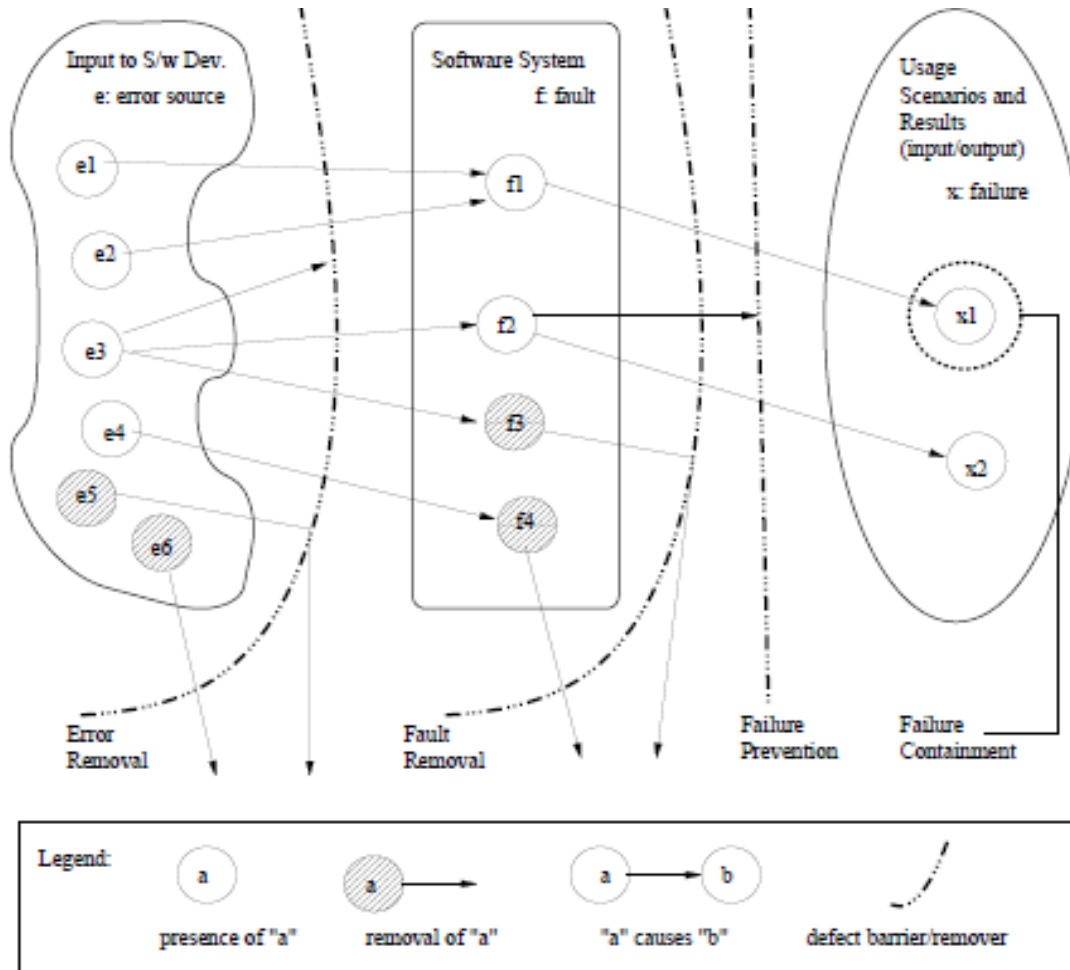
Cost of Defect (Bug) Identification and Repair

The relative costs to find and repair an error or defect increase dramatically as we go from prevention to detection to internal failure to external failure costs.



*Taken from
"Software
Testing" by
Ron Patton*

Software Quality Assurance: Towards High Quality Software



Three generic categories

Defect Prevention :

Avoid error sources

Defect Reduction:

Software Inspection, Testing, Risk management

Defect Containment:

Software fault tolerance ...etc

Software Quality Assurance Stages

Software QA as dealing with defects

Defect prevention through error blocking

- Mitigating human errors by education and training (Product and domain specific knowledge, software development knowledge, knowledge about development methodology, technology and tools, development process knowledge)
- Better manage software process
- Using appropriate tools
- Using better software design/architecture

Defect reduction through fault detection and removal

- Inspection (code/design/requirement/test plan/test cases ... etc)
- Testing : Failure observation and fault removal

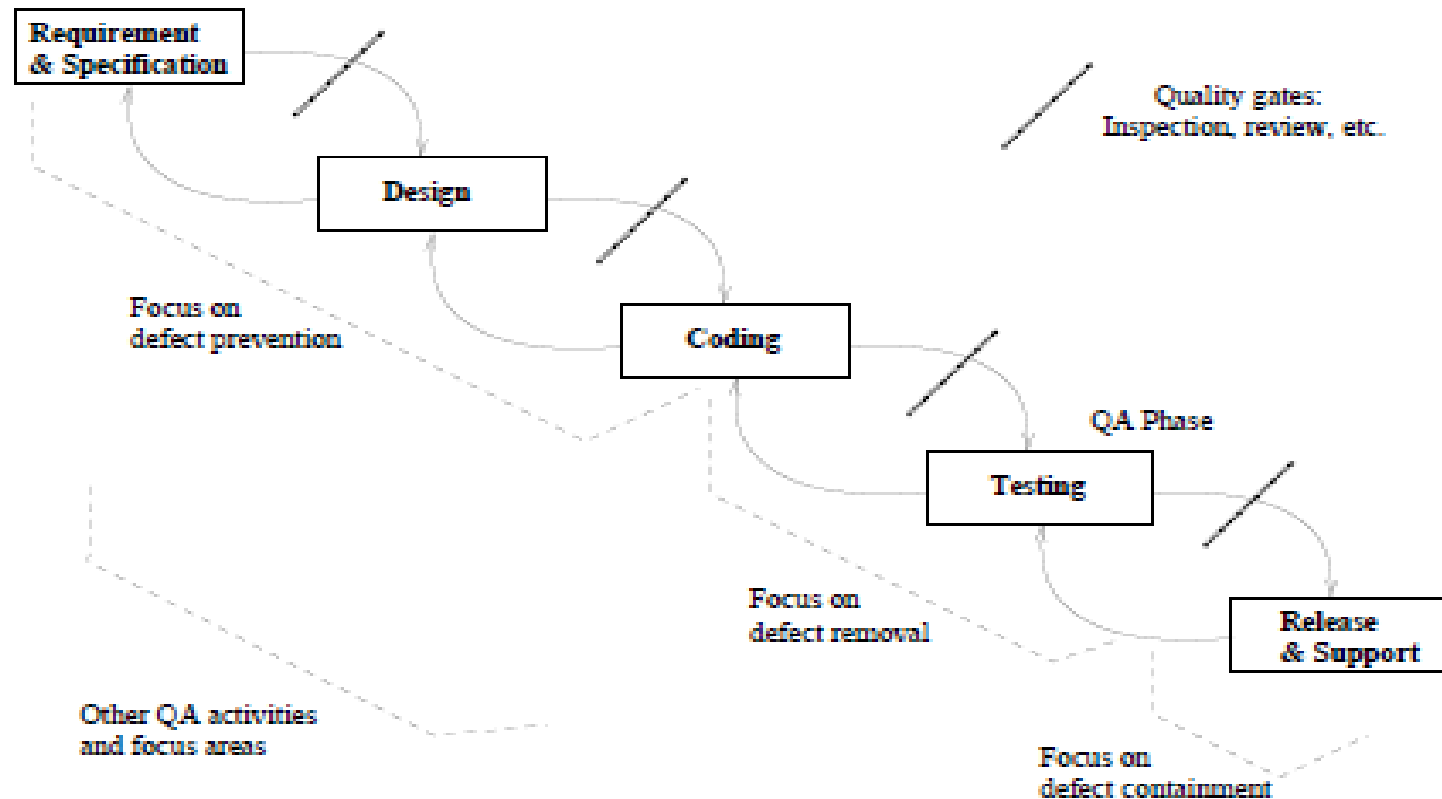
Defect containment through failure prevention and containment

- Software fault tolerance (Executing the recovery block repeatedly, Duplication/parallel redundancy)

Software Life Cycle Models and Quality Assurance

- Software life cycle for a particular project depends on the nature of the project deliverables, timelines, resources available, completeness of requirements ...etc
- Major categories of software life cycle models are **Waterfall, Incremental, Iterative, Agile**
- Aligning the SQA techniques with the software life cycle model used to develop the software is imperative in ensuring high quality software product.
- Investigating software quality assurance activities in the waterfall model allows us to get a better perspective

Quality Assurance Activities in the Waterfall Model

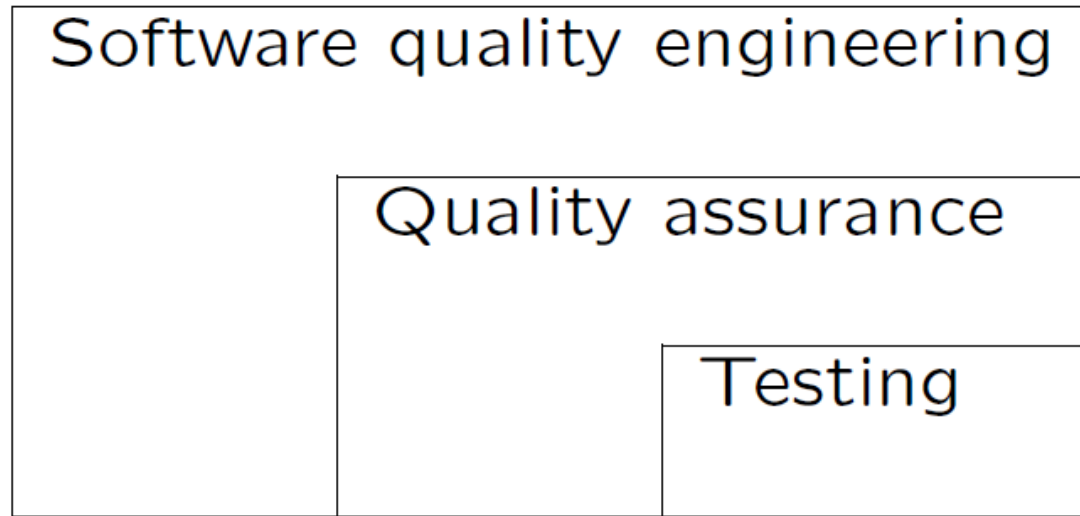


Quality Assurance Activities in the Waterfall Model

- defect prevention in early phases
 - Error sources are associated with early phases
 - Inspections and reviews can be used to detect and remove error sources
- focused defect removal in testing phase
- defect containment in late phases
- phase transitions:
 - Inspection, review...etc except from testing to release.
 - From testing to release its acceptance testing

Software Quality Assurance and Testing

Software Testing is the most important part of QA and the most commonly performed QA activity



Software Quality vs. Testing [1]

What do we test?

- In broader sense, we test if the software **does the right thing** and the software **does the things right**.
 - **Does the right thing** : based on the specification of the software
 - **Does the thing right**: perform specific tasks correctly and satisfactory

Why Do We Need Testing (Can we hire perfect software eng/programmers?)

Watts S. Humphrey at CMU conducted a study of 13000 programs and concluded that **“on average professional coders make 100 to 150 errors in every thousand line of code they write”**.

Reading assignment (posted with week 1 lectures):
Why Software is So Bad, By Charles C. Mann,
Technology Review, June 27, 2002

Thank You !



Questions ?