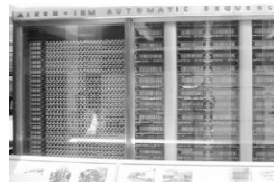


# Introduction to Software Testing

## What is a computer bug?

- In 1947 Harvard University was operating a room-sized computer called the Mark II.
  - mechanical relays
  - glowing vacuum tubes
  - technicians program the computer by reconfiguring it
  - Technicians had to change the occasional vacuum tube.
- A moth flew into the computer and was zapped by the high voltage when it landed on a relay.
- Hence, the first computer bug!



## Bugs a.k.a. ...

- Defect
- Fault
- Problem
- Error
- Failure
- Inconsistency
- Product Incidence
- Feature

## Defective Software

- We develop programs that contain defects
  - How many? What kind?
- Hard to predict the future, however... it is highly likely, that the software we (including you!) will develop in the future will not be significantly better.

## Sources of Problems

- **Requirements Definition:** Erroneous, incomplete, inconsistent requirements.
- **Design:** Fundamental design flaws in the software.
- **Implementation:** Mistakes in chip fabrication, wiring, programming faults, malicious code.
- **Support Systems:** Poor programming languages, faulty compilers and debuggers, misleading development tools.

## Sources of Problems (Cont'd)

- **Inadequate Testing of Software:** Incomplete testing, poor verification, mistakes in debugging.
- **Evolution:** Sloppy redevelopment or maintenance, introduction of new flaws in attempts to fix old flaws, incremental escalation to inordinate complexity.

## Adverse Effects of Faulty Software

- **Communications:** Loss or corruption of communication media, non delivery of data.
- **Space Applications:** Lost lives, launch delays.

## Adverse Effects of Faulty Software (Cont'd)

- **Transportation:** Deaths, delays, sudden acceleration, inability to brake.
- **Safety-critical Applications:** Death, injuries.
- **Electric Power:** Death, injuries, power outages, long-term health hazards (radiation).

## Adverse Effects of Faulty Software (Cont'd)

- **Money Management:** Fraud, violation of privacy, shutdown of stock exchanges and banks, negative interest rates.
- **Control of Elections:** Wrong results (intentional or non-intentional).

## Bug in Space Code

- Project Mercury's FORTRAN code had the following fault:  
    DO I=1.10 instead of ... DO I=1,10
- The fault was discovered in an analysis of why the software did not seem to generate results that were sufficiently accurate.
- The erroneous **1.10** would cause the **loop** to be executed exactly once!

## Military Aviation Problems

- An F-18 crashed because of a missing exception condition:  
**if ... then ... without the else clause**  
that was thought could not possibly arise.

## Year Ambiguities

- In 1992, Mary Bandar received an invitation to attend a kindergarten in Winona, Minnesota, along with others born in '88.
- Mary was 104 years old at the time.

## Year Ambiguities (Cont'd)

- Mr. Blodgett's auto insurance rate tripled when he turned 101.
- He was the computer program's first driver over 100, and his age was interpreted as 1.
- This is a mistake because the program's definition of a teenager is someone under 20!

## Therac-25 Radiation "Therapy"

- In Texas, 1986, a man received between 16,500-25,000 rads in less than 1 sec, over an area of about 1 cm.
- He lost his left arm, and died of complications 5 months later.
- In Texas, 1986, a man received at least 4,000 rads in the right temporal lobe of his brain.
- The patient eventually died as a result of the overdose.

## AT&T Bug: Hello? ... Hello?

- In mid-December 1989, AT&T installed new software in 114 electronic switching systems.
- On January 15, 1990, 5 million calls were blocked during a 9 hour period nationwide.

## AT&T Bug (Cont'd)

- The bug was traced to a C program that contained a **break** statement within a **switch** clause nested within a loop.
- The **switch** clause was part of a loop. Initially, the loop contained only **if** clauses with **break** statements to exit the loop.
- When the control logic became complicated, a **switch** clause was added to improve the readability of the code ...



## Bank Generosity

- A Norwegian bank ATM consistently dispersed **10 times** the amount required.
- Many people happily joined the queues as the word spread.

## Bank Generosity (Cont'd)

- A software flaw caused a UK bank to **duplicate** every transfer payment request for **half an hour**. The bank lost 2 billion British pounds!
- The bank eventually recovered the funds but **lost half a million pounds** in potential interest.

## Making Rupee!

- An Australian man purchased \$104,500 worth of Sri Lankan Rupees.
- The next day he sold the Rupees to another bank for \$440,258.
- The first bank's software had displayed a **false** exchange rate in the Rupee position!
- A judge ruled that the man had acted without intended fraud and could keep the extra \$335,758!

## Discussion ...

- Have you heard of other software bugs?
  - In the media?
  - From personal experience?
- Does this embarrass you as a future software engineer?

## Specification

“if you can’t say it, you can’t do it”

- You have to know what your product is before you can say if it has a bug.
- A *specification* defines the product being created and includes:
  - **Functional requirements** that describes the features the product will support. E.g., on a word processor
    - Save, print, check spelling, change font, ...
  - **Non-functional requirements** are constraints on the product. E.g,
    - Security, reliability, user friendliness, platform, ...

## A software bug occurs when at least one of these rules is true

- The software does not do something that the specification says it should do.
- The software does something that the specification says it should not do.
- The software does something that the specification does not mention.
- The software does not do something that the product specification does not mention but should.
- The software is difficult to understand, hard to use, slow ...

Most bugs are not because of mistakes in the code ...

- **Specification** ( $\sim$  55%)
- **Design** ( $\sim$  25%)
- **Code** ( $\sim$  15%)
- **Other** ( $\sim$  5%)

## Relative cost of bugs

“bugs found later cost more to fix”

- Cost to fix a bug increases exponentially ( $10^x$ )
  - i.e., it increases tenfold as time increases
- E.g., a bug found during specification costs \$1 to fix.
- ... if found in design cost is \$10
- ... if found in code cost is \$100
- ... if found in released software cost is \$1000

## Bug Free Software

- Why can't software engineers develop software that just works?
  - As software gets more features and supports more platforms it becomes increasingly difficult to make it create bug-free.

## Discussion ...

- Do you think bug free software is unattainable?
  - Are there technical barriers that make this impossible?
  - Is it just a question of time before we can do this?
  - Are we missing technology or processes?

## Goal of a software tester

- ... **to find bugs**
- ... as *early* in the software development processes as possible
- ... and make sure they get *fixed*.

## The Software Development Process

## Software is ...

- requirements specification documents
- design documents
- source code
- test suites and test plans
- interfaces to hardware and software operating environment
- internal and external documentation
- executable programs and their persistent data

## Software effort

- Specification
- Product reviews
- Design
- Scheduling
- Feedback
- Competitive information acquisition
- Test planning
- Customer surveying
- Usability data gathering
- Look and feel specification
- Software architecture
- Programming
- ...

## Customer requirements

- *The software development team must determine what the customer wants.*
- How can you do this?
  - Guess?
  - Collect detailed information from surveys?
  - Get feedback from a previous version of the software?
  - Read reviews in magazines?
  - Get information about the competition?
  - Other ways?
- The collected data is used to guide the specification effort.

## Specification

**"If you don't know where you're going any road will take you there"**

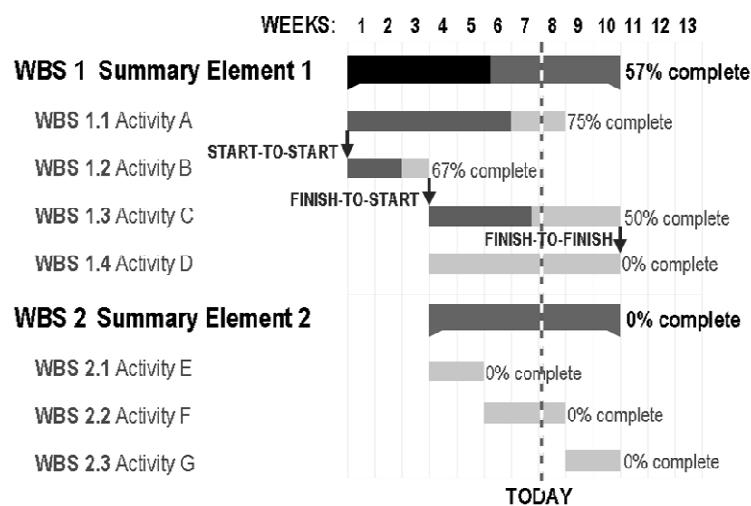
- The specification takes the data from the customer requirements and other sources and defines:
  - The features of the software (functional requirements).
  - The constraints on these features (non-functional requirements).
- Specifications can be:
  - formal (e.g., aerospace industry), rigid
  - informal (e.g., a .com start up), on a napkin or a whiteboard.



## Schedules

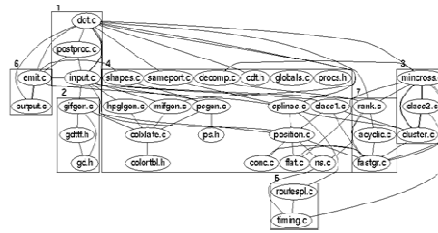
- The goals of scheduling are to know:
  - **What** work needs to be completed?
  - **How** much work is left to do?
  - **When** will the work be finished?
  - **Who** will finish each task?
  - **Other** measurable queries.
- A **Gantt chart** is a popular type of bar chart that illustrates a project schedule.

## Gantt Chart



## Design

- Before coding begins on non-trivial software projects, a set of design documents are created to serve as blueprints.
  - Software Architecture
  - Data flow diagram
  - State transition diagram
  - Flowchart
  - Commented code



## Source code ... of course

- The ultimate specification of the software!
- 'Code is king' philosophy still prevalent.
- Many programming languages and tools to choose from.

```
/**
 * Simple HelloButton() method.
 * @version 1.0
 * @author john doe <doe.j@example.com>
 */
HelloButton()
{
    JButton hello = new JButton( "Hello, wor
    hello.addActionListener( new HelloBtnList

    // use the JFrame type until support for
    // new component is finished
    JFrame frame = new JFrame( "Hello Button"
    Container pane = frame.getContentPane();
    pane.add( hello );
    frame.pack();
    frame.show(); // display the fra
}
```

## Test documents

- Test plan
  - Quality objectives, resource needs, schedules, assignments, methods, etc.
- Test cases
  - Inputs and expected outputs.
- Bug reports
  - E.g., the Bugzilla web-based bug tracker.
- Test tools and automation
- Metrics, statistics, and summaries
  - Number of unresolved bugs, mean time to repair a bug, etc.

## Software Project Staff

- Project managers
  - Write product specification, manage the schedule, critical decision tradeoffs
- Software architects, system engineers
  - Design the software, work closely with programmers
- Programmers, developers, coders
  - Write code, fix bugs
- Testers, quality assurance staff
  - Find bugs, document bugs, track progress of open bugs
- Technical writers
  - Write manuals, on line documentation
- Configuration managers, builders
  - Packaging and code, documents, and specifications

# Software Development Lifecycles

- Waterfall
- Spiral

