Software Failures – Why we need testing

Software Testing and Quality Assurance

Joe Timoney

First a little history on the programmer

- At the beginning of the computer revolution it was women that were the first programmers
- Ada Lovelace is thought to have written the first code
- She wrote a program for Babbage's Analytical engine

ENIAC

- In 1943 the first general-purpose computer was built in the US at the University of Pennsylvania
- The programming team of six were all female
- See http://www.topsecretrosies.com/Top_Secret_Rosies/Home.html

1960s

- By the 1960s men from established fields like physics, mathematics and electrical engineering left their old professions to become programmers
- This was a new job that had no professional identity, no professional organisations, and no means of screening potential members

Early Programmers

- Programming was viewed as an art form
- The programmer often learned his craft by trial and error. There was no such thing as Stack Overflow….
- The software world was completely undisciplined

Development of the programming profession

- This changed as the user had to specify the requirements,
- And the programmer developed the programs
- A division of labour occurred as programs become more sophisticated

The Software Industry

- Throughout the 1970s there was an expansion of automated information-processing tasks in companies
- The importance of programming to companies increased and tools appeared to support the programmer's productivity
- The introduction of the personal computer and its widespread adoption after 1980 accelerated the demand for software and programming

The nature of Software

- It is abstract and intangible
- There is a lack of physical constraints
- Software is not thought to have natural limits unlike real-world materials
- It easily becomes extremely complex
- It is effort intensive (need to organize carefully)

Many ordinary software failures

- Does not solve user's problem
- There is some schedule slippage and it is not ready in time
- There can be cost over-runs and in extreme cases it becomes too costly to complete
- It has poor quality and poor maintainability

Ariane 5

- On June 4, 1996, the maiden flight of the European Ariane 5 launcher crashed about 40 seconds after takeoff. Media reports indicated that the amount lost was half a billion dollars -- uninsured.
- The CNES (French National Center for Space Studies) and the European Space Agency immediately appointed an international inquiry board who produced their report in hardly more than a month

Ariane 5

Ariane 5

- It is a remarkably short, simple, clear and forceful document. Its conclusion: the explosion was the result of a software error -- possibly the costliest in history.
- The error came from a piece of the software that was *not* needed during the crash. It has to do with the Inertial Reference System, (termed SRI in the report).
- Before lift-off certain computations are performed to align the SRI. It caused an exception, which was not caught after takeoff.

Ariane 5

- The exception was due to a floating-point error: a conversion from a 64-bit integer to a 16-bit signed integer (which should only have been applied to a number less than 2^15) was erroneously applied to a greater number, representing the "horizontal bias" of the flight.
- There was no explicit exception handler to catch the exception, so it crashed the entire software, hence the on-board computers, hence the mission.

Therac-25

- The Therac-25 was a radiation therapy machine which caused massive radiation overdoses that resulted in the serious injury and death of patients.
- Eleven Therac-25s were installed: five in the US and six in Canada. Six accidents involving massive overdoses to patients occurred between 1985 and 1987. The machine was recalled in 1987 for extensive design changes, including hardware safeguards against software errors.

Therac-25

- The overdoses have generally been attributed to the flaws in the software that would allow operators to override SW errors that would arise, many fatal to those patients being treated.
- The amount of the overdose was, more often than not, many times more than the recommended therapeutic dose that eventually culminated in severe trauma or death.

Lufthansa Airbus Crash

- On Sept. 14, 1993, a Lufthansa Airbus A320-200 was landing in bad weather at Warsaw airport, Poland. The pilots had been warned of gusting cross winds, rain and possible wind shear conditions.
- In order to compensate for the bad weather problems the crew added 20 knots of speed to their landing approach and used a standard cross wind landing technique, keeping the right wing low and landing first on the right gear.

Lufthansa Airbus Crash

- However, because of the gusting winds and heavy rains, the wheels aquaplaned during the first nine seconds on the ground. The extra wind and water combined to fool the Airbus computer, indicating the big jet had not landed.
- The computer responded by disabling the aircraft braking systems. With no brakes, the Lufthansa jet skidded off the end of the Warsaw runway and struck a hill, killing the first officer, one passenger, and injuring 45 others. The A320 was totally destroyed in the crash.

Lufthansa Airbus Crash

- The crash report that followed indicated the flight crew followed the Airbus book on how to land the big jet in bad weather. Lufthansa, in response to the crash, changed the procedures against the advice of Airbus.
- No Lufthansa A320s have crashed since that change. Airbus, of course, insists there is no problem in their control software.

AT&T

- For nine hours in January 1990 no AT&T customer could make a long-distance call.
- The problem was the software that controlled the company's long-distance relay switches—software that had just been updated.
- AT&T wound up losing $60 million in charges that day.

Pentium Chip

- Thanks to a programming error, Intel's famous Pentium chip turned out to be pretty bad at math. The actual mistakes it made were fairly minute (beyond the eighth decimal point) and limited to certain kinds of division problems.
- The problem became a huge public relations disaster.
- After playing down the severity of the problem, causing even more public backlash, Intel finally agreed to provide anyone who asked with a fixed chip.

Software Failures

- **Y2K (1999)**
- **Cost:** €350 billion
  - **Disaster:** One man's disaster is another man's fortune, as demonstrated by the infamous Y2K bug. Businesses spent billions on programmers to fix a glitch in legacy software. While no significant computer failures occurred, preparation for the Y2K bug had a significant cost and time impact on all industries that use computer technology.
  - **Cause:** To save computer storage space, legacy software often stored the year for dates as two digit numbers, such as "99" for 1999. The software also interpreted "00" to mean 1900 rather than 2000, so when the year 2000 came along, bugs would result.

Software Failures

- **Mars Climate *Orbiter* (1998)**
- **Cost:** €100 million
  - **Disaster:** After a 286-day journey from Earth, the Mars Climate Orbiter fired its engines to push into orbit around Mars. The engines fired, but the spacecraft fell too far into the planet's atmosphere, causing it to crash on Mars.
  - **Cause:** The software that controlled the Orbiter thrusters used imperial units for force (pound-seconds), rather than metric units (Newton-seconds, defined in the Software Interface Specification (SIS)) for thrust instructions as specified by NASA and used in the software generating the instructions on the ground – ***a metric mixup***

Windows Vista

- For 19 hours on August 24, 2007, anyone who tried to install Windows was told, by Microsoft's own antipiracy software (called Windows Genuine Advantage) that they were installing illegal copies.
- If you'd bought Windows Vista, you discovered certain features shut off as punishment. The bug this time was both human and traditional: Someone accidentally installed a buggy, early version of the Genuine Advantage software on Microsoft's servers.

Software Failures

- **Royal Bank of Scotland (2012)**
- **Cost:** unknown, still to be determined
  - **Disaster:** A software update was applied on 19 June 2012 to RBS CA-7 software which controls the payment processing system. Customer wages, payments and other transactions were disrupted. Some customers were unable to withdraw cash using ATMs or see bank account details. Others faced fines for late payment of bills because the system could not process direct debits. It took until the 16th July before it was fixed.
  - **Cause:** The software upgrade was corrupted

RBS Software Upgrade Consequences

- People could not withdraw cash from the ATMs
- Bills could not be paid because direct debits could not be processed, furthermore, customers faced fines for not having bills paid on time
- Wages were not being paid into accounts
- Social welfare payments were not going through
- Completion of new home purchases were delayed
- Others were stranded abroad

British Airways Terminal 5 Opening in 2008

- Terminal 5 opened at Heathrow airport on March 27th 2008
- During the first five days, BA misplaced more than 23,000 bags, cancelled 500 flights and made losses of £16m.
- The CEO, Willie Walsh, revealed that IT problems and a lack of testing played a large part in the trouble. But he said the airline could have coped if IT had been the only issue.

British Airways Terminal 5 Opening in 2008

British Airways Terminal 5 Opening in 2008

- BAs' written evidence showed how many IT problems staff had to contend with.
- To begin with, loading staff could not sign on to the baggage-reconciliation system to link passengers and bags. They had to reconcile bags manually, causing flight delays.
- Problems with the wireless Lan at some check-in stands meant that staff could not enter information on bags into the system using their handheld devices.

British Airways Terminal 5 Opening in 2008

- During testing on the baggage system, technicians installed software filters in the baggage system.
- Their job was to prevent specimen messages generated by the baggage system during the tests being delivered to the "live" systems elsewhere in Heathrow.
- They were accidently left in place after the terminal opened.

British Airways Terminal 5 Opening in 2008

- As a result, the Terminal 5 system did not receive information about bags transferring to British Airways from other airlines.
- The unrecognised bags were automatically sent for manual sorting in the terminal's storage facility

British Airways Terminal 5 Opening in 2008

- An "incorrect configuration" stopped the feed of data from the baggage-handling system to the baggage reconciliation system.
- On Saturday 5 April - a week and a half after opening - the reconciliation system failed for the whole day. Bags missed their flights because the faulty system told staff that they had not been security screened.

British Airways Terminal 5 Opening in 2008

- As these errors built up, more bags went unrecognised by the system, missed their flights, or had to re-booked on new flights.
- The baggage-handling system froze after becoming unable to cope with the number of messages generated by re-booking flights, forcing managers to switch off the automated re-booking system.

British Airways Terminal 5 Opening in 2008

- By 5pm on the first day of opening, British Airways could no longer accept checked baggage. It told passengers in the departure lounge they would be leaving without their luggage.
- Anyone who had not yet checked in could choose between travelling without baggage or re-booking their flight. Staff took unrecognised bags out of the system and sorted them manually - this happened every day until 31 March, during which time a total of 23,205 bags had to be manually sorted.

British Airways Terminal 5 Opening in 2008

- BA puts the failure to spot the IT issues down to inadequate system testing, caused by delays to BAA's construction work.
- Construction work was scheduled to finish on 17 September 2007. The delays meant BA IT staff could not start testing until 31 October.
- Several trials had to be cancelled, and BA had to reduce the scope of system trials because testing staff were unable to access the entire Terminal 5 site.

British Airways Terminal 5 Opening in 2008

- Walsh said BA's IT staff finally removed the software filters on 31 March 2008, four days after opening.
- This was not the only IT problem to continue for a few more days though….

In September 2016…

- British Airways passengers have been hit by long delays after an IT glitch affected worldwide check-in systems.
- Angry travellers were forced queued for hours, as airport staff manually processed flight checks-ins. Some passengers posted photographs on social media of hand-written boarding passes.

BA September 2016

Just for reassurance…

- British Airways is not the only major airline to have experienced technical difficulties in 2016, as Delta Air Lines was forced to ground flights worldwide in early August 2016 after a power outage knocked out its computer systems around the world

The explanation

- "Airline computers juggle multiple systems that must interact to control gate, reservations, ticketing and frequent fliers. Each of those pieces may have been written separately by different companies. Even if an airline has backup systems, the software running those likely has the same coding flaw," he said.
- "Tracking down a software flaw can be very difficult. It's like investigating crime; there is a lot of data they've got to sift through to figure out what happened."

Software Failures

- **Excel 2007:**
  - Ask people with calculators or slide rules to multiply 850 x 77.1, and they'll answer 65,535. But in September 2007, it was discovered that Excel 2007 answered 100,000.
  - **Cause:** According to Microsoft, this bizarre rounding-up occurred only in calculations that resulted in floating point numbers near 65,535 or 65,536 – but there was a *display-only* bug. What's more, Excel actually calculated the correct answer, but the bug prevented it from displaying properly.

Excel 2007 Bug

AC1 =AA1*AB1

	AA	AB	AC
1	425	154.2	100000
2	850	77.1	100000
3	1700	38.55	100000
4	6375	10.28	100000
5	6425	10.2	100000
6	12750	5.14	100000
7	12850	5.1	100000
8	25500	2.57	100000
9	25700	2.55	100000

Many other examples exist

- In July 2001 a "serious flaw" was found in off-the-shelf software that had long been used in systems for tracking U.S. nuclear materials.
- The software had recently been donated to another country and scientists in that country discovered the problem and told U.S. officials about it

Interesting Quotes

- “If Microsoft made cars instead of computer programs, product-liability suits might now have driven them out of business.”
- if cars were like software, they would crash twice a day for no reason, and when you called for service, they’d tell you to reinstall the engine
- “Thank the programmer”, Airline pilot after a smooth landing

Interesting Quotes

- There is not now, and never will be, a language in which it is the least bit difficult to write bad programs
- Your problem is another's solution; Your solution will be their problem

(In)famous BSOD

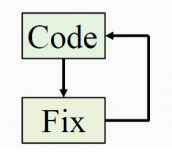
Any solutions?

- Software Engineering!!!
- The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"—IEEE Standard Glossary of Software Engineering Terminology.

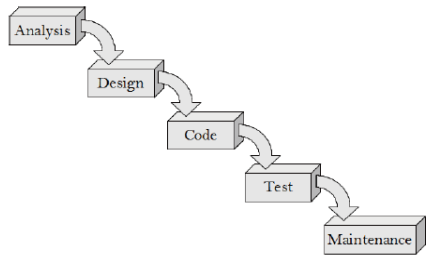
Software Development Lifecycles

- Code and Fix
- Waterfall
- V-model
- Sashmi
- Incremental development
- Extreme Programming
- SCRUM
- DevOps

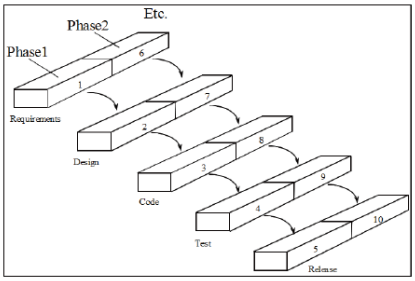
Code and Fix



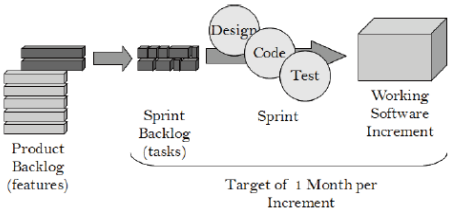
Waterfall



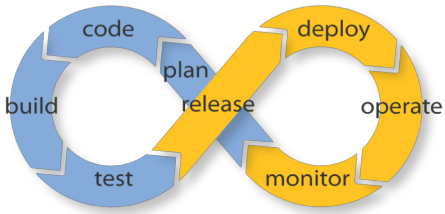
Incremental



SCRUM



DevOps



Endless Possibilities: DevOps can create an infinite loop of release and feedback for all your code and deployment targets.

Software Testing and Debugging

- Software testing is concerned with confirming the presence of errors
- Debugging is concerned with locating and repairing these errors

Static Testing

- Static Verification (or Static Analysis) can be as straightforward as having someone of training and experience reading through the code to search for faults.
- It could also take a mathematical approach consisting of symbolic execution of the program

Static Testing

- Can use modelling such as UML
- Can apply formal methods
- Tools such as Spec# exist

Spec#

- Spec# is a formal language for API contracts (influenced by JML, AsmL, and Eiffel), which extends C# with constructs for non-null types, preconditions, postconditions, and object invariants.
- Spec# comes with a sound programming methodology that permits specification and reasoning about object invariants even in the presence of callbacks and multi-threading.

Dynamic Testing

- Dynamic Verification (or Software Testing) confirms the operation of a program by executing it.
- Test Cases are created that guide the selection of suitable Test Data (consisting of Input values and Expected Output values).
- The Input values are provided as inputs to the program during execution
- The Actual Outputs are collected from the program, and then they are compared with the Expected Outputs.

Black and White box Testing

- Black Box testing is based entirely on the program specification and aims to verify that the program meets the specified requirements
- White box testing uses the implementation of the software to derive the tests. The tests are designed to exercise some aspect of the program code

InternationalComparisons

2003 IEEE Software

- **Survey:** Completed in 2002-2003
- **Objective:** Determine usage of iterative versus Waterfall-ish techniques, with performance comparisons
  - 118 projects plus 30 from HP-Agilent for pilot survey
- **Participants**

  India: Motorola MEI, Infosys, Tata, Patni

  Japan: Hitachi, NEC, IBM Japan, NTT Data, SRA, Matsushita, Omron, Fuji Xerox, Olympus

  US: IBM, HP, Agilent, Microsoft, Siebel, AT&T, Fidelity, Merrill Lynch, Lockheed Martin, TRW, Micron Tech

  Europe: Siemens, Nokia, Business Objects

"Conventional" Good Practices

	India	*Japan*	*USA*	*Europe etc*	*Total*
Number of Projects	*24*	*27*	*31*	*22*	*104*
Architectural Specs %	83%	70%	55%	73%	69%
Functional Specs %	96%	93%	74%	82%	86%
Detailed Design %	100%	85%	32%	68%	69%
Code Generators -- Yes	63%	41%	52%	55%	52%
Design Reviews - Yes	100%	100%	77%	77%	88%
Code Reviews -- Yes	96%	74%	71%	82%	80%

"Newer" Iterative Practices

	India	Japan	USA	Europe etc	Total
No. of Projects	*24*	*27*	*31*	*22*	*104*
Subcycles -- Yes	79%	44%	55%	86%	64%
Beta tests -- Yes	67%	67%	77%	82%	73%
Pair Testing -- Yes	54%	44%	35%	32%	41%
Pair Programmer -- Yes	58%	22%	36%	27%	35%
Daily Builds at project start	17%	22%	36%	9%	22%
In the middle	13%	26%	29%	27%	24%
At the end	29%	37%	36%	41%	36%
Regression test each build	92%	96%	71%	77%	84%

"Crude" Output Comparisons

		India	Japan	USA	Europe etc.	TOTAL
Projects		24	27	31	22	***104***
LOC/ Month	median	209	469 cf. *389* in 1990	270 cf. *245* in 1990	436	***374***
faults/ 1000 LOC	median	.263	**.020** cf. .20 in 1990	.400 cf. .80 in 1990	.225	***.150***

Observations

- Most projects (64%) are not pure waterfall; 36% were
- Mix of "conventional" and "iterative" common
  use of functional specs, design & code reviews, but with subcycles, regression tests on frequent builds
- Customer-reporting of defects improved
  over past decade in US and Japan; LOC "productivity" may have improved a little
- Japanese still report best quality & productivity
  but what does this mean? Preoccupation with "zero defects"? Need more lines of code to write same functionality per day as US & Indian programmers?
- Indian projects strong in process and quality

Observations

- Best "nominal" quality from traditional "waterfall" (fewer cycles & late changes implies less bugs)
- Best balance of quality, flexibility, cost & speed from combining conventional & iterative practices
- However, differences in quality between waterfall & iterative disappear if a bundle of techniques of used:
  1. Early prototypes (get customer feedback early)
  2. Design reviews (continuously check quality of design)
  3. Regression tests on each build (continuously check quality of code and functional status)