CompSci 230
Software Design and Construction

Software Quality S1 2015
Famous Failures

Lecture plan

Week 1: *No class - Anzac Day*
What is software quality?
Some key developer practices (version control, testing).

Week 2: Black box testing.
White-box testing.
Myers' testing principles.

Week 3: Traditional approach to testing (Waterfall).
Agile approach to testing (XP).
Famous failures.

Learning Goals for Today

- Schadenfreude (pleasure derived from the misfortunes of others), with some lessons learned:
  - Software professionals are trusted to "do the right thing" and to "do no harm". But: we all make mistakes, and some of us are unethical.
  - Safety-critical software can fail catastrophically, even if it is carefully tested.
  - Software has complex failure modes, often with no single cause.

Case Study: "Doing a Rotorua"

- "… Leo Gao [who ran a filling station in Rotorua] and his girlfriend Kara were like millions of couples around the world
  - as they struggled to pay their bills and keep their business afloat.
- "Today [31 May 2009], they are the subject of an international hue and cry,
  - leaving lawsuits, huffing and puffing private detectives and puzzled police on two continents in their wake.
  - And all because of a mark on a computer screen one fortieth of an inch across.
- "The desktop in question was in the Christchurch offices of Westpac, a leading New Zealand bank.
  - At its keyboard sat a woman with 30 years' experience who was about to perform a routine task: formalising an overdraft facility… " for $100,000.
- "Every digit, including the two zeroes for the cents, was put in.
  - But one thing wasn't: the decimal point.
  - And its absence gave Mr Gao and friend not $100,000, but 100 times that amount.

Source: The Independent, 31 May 2009.

- (Is the bank's software partially to blame? How could it be improved?)

Case study: Novopay

- "The widely publicised 'Novopay' project, a New Zealand-wide transition from an onshore to a near-shore service provider (Datacom to Talent2), was intended to
  - update the payroll processing system of the Ministry of Education and so
  - implement a new nationwide payroll system responsible for the payment of about 110,000 teachers and education sector staff.
- "After various changes of direction and delays from its initiation in 2005,
  - the project eventually went live in a 'big bang' cutover towards the end of 2012.
- "The cutover occurred with known outstanding issues, and the result was that the bulk of the schools had to deal with
  - underpayments,
  - overpayments, or
  - non-payments
  - and a series of compounding errors."

Source: Clear et al., The Novopay Project: The Dilemmas in Global Software Outsourcing, IITP Newsline, 17 Oct 2013.

Novopay: a failure of testing?

- Novopay's test plan was carefully developed, see http://www.education.govt.nz/ministry-of-education/information-releases/novopay-information-release/novopay-test-plans/
- 5 June 2012: "the Ministry of Education has confidence in the Novopay project to go live" http://www.education.govt.nz/assets/Documents/Ministry/Information-releases/Novopay-information-release/EdReportFinalRecommendationV1.pdf
- June 2013: *Ministerial Inquiry into the Novopay Project*
  - The impacts of the well-publicised Novopay failures have reverberated across New Zealand.
  - Every state and state-integrated school in the country has been affected.
  - Dealing with the aftermath has distracted school staff, principals, boards of trustees, the Ministry of Education and Ministers from other important concerns.
  - This state of affairs and the wider disruptions that were caused were avoidable.
  - It is clear to us that important lessons from the past, in particular
    - those arising from the 1996 education payroll implementation difficulties and
    - the INCIS experience in 2000,
    - should have been learned, but were not.

Ministerial Findings

- There were many factors that contributed to the Novopay failures. It is our overall view that
  - weaknesses in project governance and leadership
  - allowed the service to go live with a number of significant risks which the Ministry and its vendors were over-confident of managing.
- When these risks resulted in service issues Post-Go Live,
  - the Ministry and its vendors were overwhelmed by their nature and scale.
- Over the course of the project,
  - Talent2 had missed agreed milestones or deadlines, which eroded trust and confidence in its ability to deliver.
- The nature of the service that the Ministry was seeking also
  - diverged from the original proposition.

Novopay: A Critical Evaluation of Failure…

- "Setting the broader historical context to the [Novopay] project …
  - the (now) Ministry of Education … had experienced a major and embarrassing payroll project failure some twenty years earlier.
  - The project was referred to as 'the failed implementation of a centralised payroll system for the New Zealand Education Department' [Myers, 1995].
- "Symptoms included
  - thousands of teachers who found they had not been paid correctly, and
  - hundreds who did not get paid at all on 8 February (the first pay day of 1989);
  - 'relief teachers and some part-time teachers had not been paid by mid-April' [Myers, 1995].
- "Yet by June 1989 the Education Department's Director of Management Services 'was able to announce publicly that the…computerized payroll system was
  - on target to meet its objective of saving the Government millions of dollars' …
  - Despite this positive perspective, 'less than six months later the centralised payroll processing was scrapped by the government' [Myers, 1995].
- "… in 1996 history seemingly repeated itself for the Ministry…
  - [Novopay] demonstrates some surprising similarities with the earlier projects… perhaps sadly illustrating that …
  - generational knowledge in implementing software systems does not exceed a ten year timespan?"
- Clear's analysis (in brief): the Novopay project encountered a similar set of "dilemmas, tensions or contradictions" between stakeholders as did the two earlier projects.
  - These problems were inadequately addressed, perhaps because they were unrecognised or discounted.

Source: Clear et al., "A critical evaluation of failure in a nearshore outsourcing project: What dilemma analysis can tell us", *8th IEEE Conf. on Global Software Engineering*, 2013, 179-187.

Tensions between Stakeholders

- "Given the mixed views of parties and interests in any large-scale project, questions arise of:
  - who is a stakeholder, and what influence does each have on the outcome?
  - Whose concerns are most likely to be taken into account in the implementation of a new system, and at what stage do they become salient?" [Clear, 2013]
- "Stakeholders of a computer system have been defined as:
  - 'People who will be affected in a significant way by or have material interests in the nature and running of the new computerised system'
- Clear et al. [2013] identify 26 stakeholders in the 1989 payroll system:
  - "Educators, Financial Organizations, Government Departments – National and Regional, non-salaried educators, Payroll operational staff, Payroll units, Political, The Press, School Principals and Regional Representatives, Senior Management at National and Regional levels, Teachers' Unions, and the Vendor."

Top Ten Costliest Software Bugs

- Not a reliable source, but a fun list!
  http://top-10-list.org/2010/05/03/ten-costliest-software-bugs/

1. Mars Climate Orbiter Crashes
   - "The contractor who was given the responsibility of planning the navigation system got the specifications from NASA but
     - instead of using the metric system,
     - he carried out measurements using imperial units.
   - "What happened was that the space craft crashed into Mars and over 125 million dollars were lost."
2. Ariane 5 Flight 501 (more on this later)

Top Ten Costliest Software Bugs (cont.)

3. EDS Fails Child Support
   - "About 6 years back, EDS created an IT system that was quite complex and presented it to the CSA or the Child Support Agency in U.K… The cost has been estimated at 1 billion dollars till date."
   - http://en.wikipedia.org/wiki/Electronic_Data_Systems:
     - In 2004, EDS was criticised by the UK's National Audit Office for its work on IT systems for the UK's Child Support Agency (CSA), which ran seriously over budget causing problems which led to the resignation of the CSA's head, Doug Smith on 2004-11-27.
     - An internal EDS memo was leaked that admitted that the CSA's system was "badly designed, badly tested and badly implemented".

Current score: Tech Errors 2, Mgmt Errors 1. Lowest score "wins" ;-)

Top Ten Costliest Software Bugs (cont.)

4. **Soviet Gas Pipeline Explosion**

- "A CIA operation to sabotage Soviet industry by duping Moscow into
  - stealing booby-trapped software was spectacularly successful …
- "… the operation caused 'the most monumental non-nuclear explosion and fire ever seen from space' in the summer of 1982."

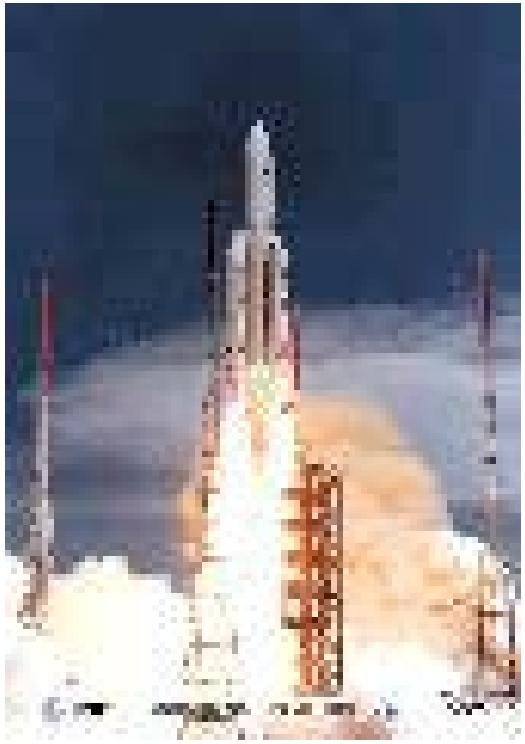
The Telegraph, 28 Feb 2004. Available: http://www.telegraph.co.uk/news/worldnews/ northamerica/usa/1455559/CIA-plot-led-to-huge-blast-in-Siberian-gas-pipeline.html

- *I'll let you "score" this one.*
- *Was it an ethical error, a management error, or a technical error for the Soviets? Was it a success in all of these ways for the CIA?*

Ariane 5 analysis, by Sommerville

- The following slides are from a case study by Ian Sommerville,
  - Author of *Software Engineering*, 9th Edition, Addison-Wesley, 2010.
- I have corrected a few typos, reformatted, added some colour (for emphasis), and added a few comments (in red).
- His original slides are available for download:
  - http://www.cs.st-andrews.ac.uk/~ifs/Books/SE9/CaseStudies/Ariane5/index.html
  -

Ariane 5



- A European rocket designed to launch commercial payloads (e.g. communications satellites) into Earth orbit
- Successor to the successful Ariane 4 launchers
- Ariane 5 can carry a heavier payload than Ariane 4
- YouTube Video of the first launch (25 seconds). Longer video (2 min.)

The problem

- The attitude and trajectory of the rocket are measured by a computer-based inertial reference system.
  - This transmits commands to the engines to maintain attitude and direction.
  - The software failed and this system and the backup system shut down.
- Diagnostic commands were transmitted to the engines which interpreted them as real data
  - and which swivelled to an extreme position resulting in unforeseen stresses on the rocket.

Software failure

- Software failure occurred when
  - an attempt to convert a 64-bit floating point number to a signed 16-bit integer caused the number to overflow.
- There was no exception handler associated with the conversion
  - so the system exception management facilities were invoked.
  - These shut down the software.
- The backup software was a copy and
  - behaved in exactly the same way.

Avoidable failure?

- The software that failed was reused from the Ariane 4 launch vehicle.
  - The computation that resulted in overflow was not used by Ariane 5.
- Decisions were made
  - Not to remove the facility as this could introduce new faults;
  - Not to test for overflow exceptions because the processor was heavily loaded.
    - For dependability reasons, it was thought desirable to have some spare processor capacity.

Why not Ariane 4?

- ... Ariane 4 (a smaller vehicle) ... has a lower initial acceleration and build up of horizontal velocity than Ariane 5.
- The value of the variable [that overflowed on Ariane 5] ... could never reach a level [on Ariane 4] that caused overflow ...

Validation failure

- As the facility that failed was not required for Ariane 5,
  - there was no requirement associated with it.
- As there was no associated requirement,
  - there were no tests of that part of the software and hence no possibility of discovering the problem.
- During system testing,
  - simulators of the inertial reference system computers were used.
  - These did not generate the error as there was no requirement!

Review failure

- The design and code of all software should be reviewed for problems during the development process
- Either
  - The inertial reference system software was not reviewed because it had been used in a previous version;
  - The review failed to expose the problem or that the test coverage would not reveal the problem;
  - The review failed to appreciate the consequences of system shutdown during a launch.

My notes:

- *We could make a professional judgement, by reference to the ACM guidelines on harm avoidance (on the next slide).*

ACM Guidelines on 1.2 Harm Avoidance

- "… Well-intended actions, including those that accomplish assigned duties, may lead to harm unexpectedly.
  - "In such an event the responsible person or persons are obligated to undo or mitigate the negative consequences as much as possible. …
  - "To minimize the possibility of indirectly harming others, computing professionals must minimize malfunctions by following generally accepted standards for system design and testing. …
  - "Furthermore, it is often necessary to assess the social consequences of systems to project the likelihood of any serious harm to others. …
- "In the work environment the computing professional has the additional obligation to report any signs of system dangers that might result in serious personal or social damage.
  - "If one's superiors do not act to curtail or mitigate such dangers, it may be necessary to 'blow the whistle' to help correct the problem or reduce the risk.
  - "However, capricious or misguided reporting of violations can, itself, be harmful. Before reporting violations, all relevant aspects of the incident must be thoroughly assessed. In particular, the assessment of risk and responsibility must be credible.
  - "It is suggested that advice be sought from other computing professionals. See principle 2.5 regarding thorough evaluations."

Lessons learned (according to Somerville)

- Don't run software in critical systems unless it is actually needed.

Good idea... but I think it was not standard practice until it was learned the "hard way" on Ariane 5.

- As well as testing for what the system should do, you may also have to test for what the system should not do.

Yes, of course... this was well-known, but ... how can we test for all the things a system "should not do"?

- Do not have a default exception handling response which is system shut-down in systems that have no fail-safe state.

Obvious in retrospect... but I think it was not standard practice until it was learned the "hard way" on Ariane 5. It is now standard practice to insist that every safety-critical system has a fail-safe state that it will reliably reach. Anyway: the Ariane 5 rocket failed, but this did not cause the mission control system to fail, and the rocket was destroyed without harming anyone!

Lessons learned (cont.)

- In critical computations, always return best effort values even if the absolutely correct values cannot be computed.

Good idea... but I think it was not standard practice until it was learned the "hard way" on Ariane 5.

- Wherever possible, use real equipment and not simulations.

Yes, of course... but eventually you have to "go live" on a test launch!

- Improve the review process to include external participants and review all assumptions made in the code.

Good idea! It is now routine, in critical design, to review assumptions "before it is too late".

Process improvement (to avoid making the same mistake again) is – I think – a very appropriate response when it's impossible to "undo" a mistake. Do you agree?

Avoidable failure

- The designers of Ariane 5 made a critical and elementary error.
- They designed a system where a single component failure could cause the entire system to fail.
- As a general rule, critical systems should always be designed to avoid a single point of failure.

This is very harsh criticism from Sommerville. Do you think it is justified?

Therac-25

- "Between June 1985 and January 1987, a computer-controlled radiation therapy machine, called the Therac-25, massively overdosed six people. …
  - Error messages provided to the operator were cryptic, and some merely consisted of the word MALFUNCTION followed by a number from 1 to 64 denoting an analog/digial channel number. …
- "An operator involved in one of the accidents testified that she had become insensitive to machine malfunctions.
  - "Malfunction messages were commonplace and most did not involve patient safety. …
  - "The operator further testified that during instruction she had been taught that there were 'so many safety mechanisms' that she understood it was virtually impossible to overdose a patient."

Source: N. Leveson, "Medical Devices: The Therac-25", excerpt from her book on *Safeware: System Safety and Computers,* Addison-Wesley, 1995. Available http://sunnyday.mit.edu/papers/therac.pdf.

Causal Factors of the Therac-25 Accidents

- Overconfidence in software
  - The first safety analysis on the Therac-25 did not include software – although nearly all responsibility for safety rested on it.
- Confusing reliability with safety
  - The software was highly reliable. It worked tens of thousands of times before overdosing anyone.
- Lack of defensive design
  - The software did not contain self-checks or other error-detection and error-handling features…
- Failure to eliminate root causes
  - … the tendency to believe that the cause of an accident had been determined (e.g. a microswitch failure …) without adequate evidence…
- Complacency
- Unrealistic risk assessments
- [Leveson identifies five other causal factors!]

Source: N. Leveson, "Medical Devices: The Therac-25", excerpt from her book on *Safeware: System Safety and Computers,* Addison-Wesley, 1995. Available http://sunnyday.mit.edu/papers/therac.pdf.

Learning Goals for Today

- Schadenfreude (pleasure derived from the misfortunes of others), with some lessons learned:
  - Software professionals are trusted to "do the right thing" and to "do no harm". But: we all make mistakes, and some of us are unethical.
  - Safety-critical software can fail catastrophically, even if it is carefully tested.
  - Software has complex failure modes, often with no single cause.