Software Testing

Overview

Learning objectives

- Be able to describe the role of software quality in the specification, design/construction, and operation of software.
- Be able to outline the main software verification and validation activities.
- Be able to identify some key criteria for selecting and combining different techniques used in software quality assurance.

Engineering processes

- Focus of Engineering Processes
  - Ensuring the product is fit for purpose in use
  - Ensuring the process is reproduceable
  - Reflecting on and improving the process
- Engineering Processes
  - Aim to build the right thing (validation) – meets the needs of use
  - Aim to build the thing right (verification) – the thing you built does what you want
  - Collect data throughout specification, design/construction, operation to: improve the product, improve the engineering process
- Software Engineering:
  - Works with products that are deeply embedded in organisations and people's lives, so:
    - Needs change continuously in many contexts
    - Processes need to be responsive: Agile, DevOps, Continuous Integration/Deployment,

Software Quality in the Engineering Process

- Software Quality Assurance (SQA) is increasingly embedded in all aspects of the Engineering process because the later quality problems are recognized the more costly it is to fix them.
- This embedding is in practices such as:
  - Test driven development: tests are the specification (roughly speaking)
  - DevOps uses data collected from operation to:
    - Continuously validate the requirements and recognize requirement change.
    - Identify software quality issues.
- The software engineering process involves:
  - Identifying and validating requirements using operational data and other sources
  - Constructing the means to verify the product meets the requirement (e.g. tests)
  - Developing a verified product (e.g. Test Driven Design)
  - Deploying the product and gathering operational data to inform decisions on development (and more)

Example proto-AirBnB

- This is intended to be realistic without being a historically accurate account:
  - Proto-AirBnB was mainly aimed at sofa surfers of people who wanted a very cheap room and the user community had quite high trust of each other.
  - Proto-AirBnB expanded to a much larger population with a wider range of accommodation.
  - Scammers started to appear, and operational data suggested that this was increasing.
  - Validation against operational data identified a missing requirement, that people needed to be accurately identified.
  - Users are required to be fully identified – spec developed – tests developed
  - New version with this feature is deployed – operational data sees decline in scams.
  - [Do you see any issue in requiring people to upload a passport image and have it verified?]

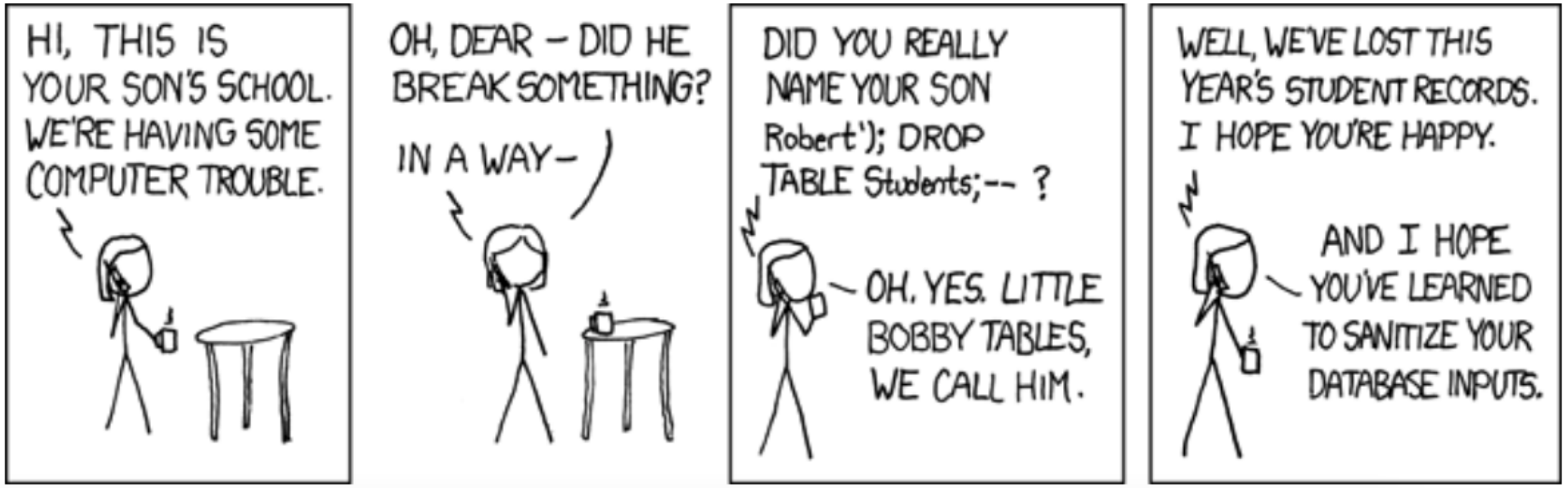
Challenges of Software

- For many systems there are also many stakeholders often with competing and conflicting requirement (can you think of examples?)
- Systems might depend on unreliable infrastructure (e.g. libraries, operating systems, comms, …)
- Software often has complex, exploitable, errors e.g. buffer overruns, “one off” issues in the use of arrays (e.g. UoE ISG mail problem)
- Software is often ”open” to a wide range of potential users and abusers.

Example

- If an elevator can safely carry a load of 1000 kg - it can also safely carry any smaller load
- a service may work perfectly most of the time but:
  - It might fail because some service it uses has failed or is insufficiently resourced
  - It might fail because of a denial of service attack.
  - It may not adequately check for SQL injection and will fail under some circumstances

Software can have users with conflicting needs



New technologies

- New Technologies
  - Can reduce error e.g. Natural Language AI models can do very accurate autocomplete so avoiding a range of issues
  - But can lead to errors of over reliance or introduce new issues e.g. some of the Natural Language models can be quite racist depending on their training.
- New development approaches can introduce new kinds of faults

examples

  - Machine Learning introduces a wide range of ethical issues.
  - Poorly designed web services can be subject to a wide range of resource issues.

Choosing Approaches

- Particular domains will have established practice and often will have standards or regulation to comply with.
- Test designers must
  - Design the testing process
    - To assure the required level of quality
    - Meet the cost constraints (notice that this may be impossible)
  - Design a specific solution that suits
    - the problem
    - the requirements
    - the development environment
- Have a look at the risks Forum: http://catless.ncl.ac.uk/risks/ for example see this: http://catless.ncl.ac.uk/Risks/33/44#subj6

Five Basic Engineering Questions

1. What is the role of software quality assurance in development processes?
2. Software quality assurance has a range of techniques – how do we choose where to use them?
3. What does it mean for a product to be “ready”?
4. How do we maintain the quality of the product as it evolves in line with the contexts of use?
5. Why and how can the process be improved?
   - Identifying process issues
   - Fixing process issues

Q1:What is the role of SQA in development processes?

- SQA starts as soon as we start to consider making a software product
  - The quality we aim for has an impact on cost, time to completion, customer satisfaction, product liability, … so we must start looking at SQA from when we initiate a project
- SQA is necessary through whole lifetime of the product, including any decommissioning activity (decommissioning can be tricky) because:
  - The world changes and so the specification changes to adapt the behaviour of the product to its environment.
  - The infrastructure underpinning the production process and the product in operation will change over time and we need to assure the quality of the product is maintained.

Early start: from feasibility study

- The feasibility study of a new project must take into account the required qualities and their impact on the overall cost
- At this stage, quality-related activities include
  - risk analysis
  - measures needed to assess and control quality at each stage of development.
  - assessment of the likely evolution of the product and its quality
  - contribution of quality control activities to development cost and schedule.

Continuous Integration and Deployment

- Many systems evolve continuously and are deployed continuously to add new features to the system of to fix issues identified in operation or by SQA activities. Typical activities include:
  - analysis of customer requests, operational data, to identify new requirements
  - Mapping those requirements to SQA elements like tests
  - Embed the new SQA elements into the development process
    - For example, if a new feature has been identified it may be appropriate to branch the development, extend the requirements and corresponding tests and develop until the system passes all tests and then merge back into the main development.

2: Software quality assurance has a range of techniques – how do we choose where to use them?

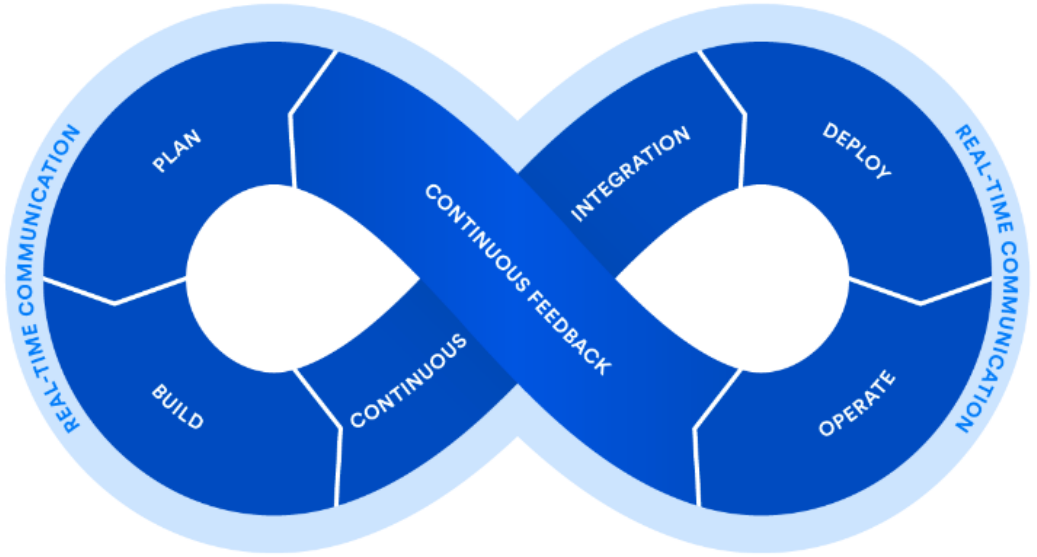
No single SQA technique can serve all purposes

We choose a technique appropriate for the quality we are trying to assure:

- Effectiveness for different features
  example: statistical testing for bias in machine learning versus functional testing to assure safety
- Applicability at different points in a project
  example: analysis of operational data to identify missing functions
- Differences in purpose
  example: statistical testing to measure reliability
- Tradeoffs in cost and assurance
  example: for key features it may be necessary to commit high levels of resource to assure high confidence that the system has the feature (e.g. for some avionic systems the standard requires fewer than one failure in 10^9 flying hours)

SQA in the lifecycle

- Fig 1.1 of Y&P gives a detailed breakdown of activities across a classical lifecycle.
- DevOps is far more common now.
- How do you think the “map” provided by Fig 1.1 translates onto a DevOps lifecycle?
- We will discuss next lecture.



A DevOps lifecycle
Source: Atlassian

3: What does it mean for a product to be "ready"?

- SQA is aimed at assuring that the product is fit for purpose:
  - Software for an autonomous vehicle has different criteria from
  - A simple game for a phone
- Software can deviate from the specification in a range of ways e.g. it may fail in some way, it may fail to meet a requirement on some measurable feature (e.g. performance, mean time to failure, mean time to repair, …)
- For example, software contains a collection of "faults" that are caused by incorrect programming or other issue, if they are executed, they may result in in an erroneous state and after some time the system may fail as a consequence of the erroneous state.
- SQA is resource bounded in most cases, so this places limits on Quality
- In a DevOps environment the test set typically includes tests corresponding to ALL requirements (including measure on attribute) so passing the tests is the stage when the product is ready.

Specifications often include dependability

- Availability measures the quality of service in terms of running versus down time
- Mean time between failures (MTBF) measures the quality of the service in terms of time between failures
- Reliability indicates the fraction of all attempted operations that complete successfully

Dependability – how to define?

- In a Web application users take 50 interactions to reach the final point where they authorize a credit card charge.
- The software always operates flawlessly up to the point of authorization, but on half the attempts it charges the wrong amount.
- What is the reliability of the system?
- If we count the fraction of individual interactions that are correctly carried out, only one operation in 100 fail: The system is 99% reliable.
- If we count entire sessions, only 50% reliable, since half the sessions result in an improper credit card charge

4: How do we maintain the quality of the product as it evolves in line with the contexts of use?

- Software products operate for many years, and may change on a daily basis (e.g. Microsoft Teams…):
  - They adapt to environment changes
  - They evolve to serve new and changing user requirements.
- In DevOps processes
  - The QA is continually update to be in line with the specification
  - The DevOps process should define what data should be collected and how it needs to be interpreted.
  - The QA mechanisms to some extent “document” the software.
  - The QA mechanisms could be adequate to demonstrate the software complies with regulations

5: How can the development process itself be improved?

- Why change the development process? If in multiple projects the product is demonstrably not meeting the specification.
- How might this arise?
  - The translation of a specification to some QA technique is flawed.
  - There are issues in the implementation of the QA process
  - There are resource issues and insufficient resource is being devoted to QA processes.
- Also, the development process should be reviewed to eliminate unnecessary resource use e.g. duplication, use of a high resource technique where a lower resource approach would be adequate,…

Improving a DevOps process

1. Monitoring will typically include anomaly reporting alomg with a range of other behavioural data.
2. Analyze monitoring data to identify recurring anomalies that impair the deployed version of the system.
3. Analyze the gathered data on the anomalies – if the data does not allow the cause of the anomalies, analyze the situation and change the monitoring data and analysis process.
4. Implement the change of process and monitor it retains effectiveness on past cases.

An example of process improvement

1. During operation monitoring identifies attempts at SQL injection by a range of users.
2. Analysis identifies specific circumstance where this could lead to loss of data
3. Action plan: Identify components that could be vulnerable to SQL injection and require that systems using these components include mandatory SQL injection tests.

Summary

- The quality assurance process has three different goals:
  - Improving a software product
  - assessing the quality of the software product
  - improving the quality process
- We need to combine several SQA techniques through the software process
- Choice of techniques depend on organization and application domain.
- Cost-effectiveness depends on the extent to which techniques can be re-applied as the product evolves.
- Planning and monitoring are essential to evaluate and refine the quality process.