CompSci 230
Software Design and Construction

Software Quality 2015S1

Revision

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.

Questions

- What do we mean by ‘software quality’?
- How can we achieve ‘software quality’?

What is software quality?

- Some ideas:
  - Fitness for use
    - How well does the product perform its intended functions?
    - Varies according to user group (concept of 'grade')
      - In a car, I want fuel-efficiency and safety
      - You may prefer a car that looks good and has powerful acceleration
    - For software, a user may care about, for example, security, usability, reliability, etc.

What is software quality?

- So the answer to 'what is software quality?' is 'it depends'
- Some products to consider
  - Control software for a patient monitoring system
    - reliability? accuracy?
  - Game to support children learning maths
    - usability? child-age appropriate?
  - Mobile app
    - resource usage? usability? developer can easily modify to upgrade frequently?

Quality models

- A number of quality models have been developed that describe the *quality characteristics* to be considered when developing a quality product.

- As a developer, you should understand which of these apply to the product you are developing.

Quality models

- ISO/IEC 25010 – System and software quality models

BS ISO/IEC 25010:2011
ISO/IEC 25010:2011(E)

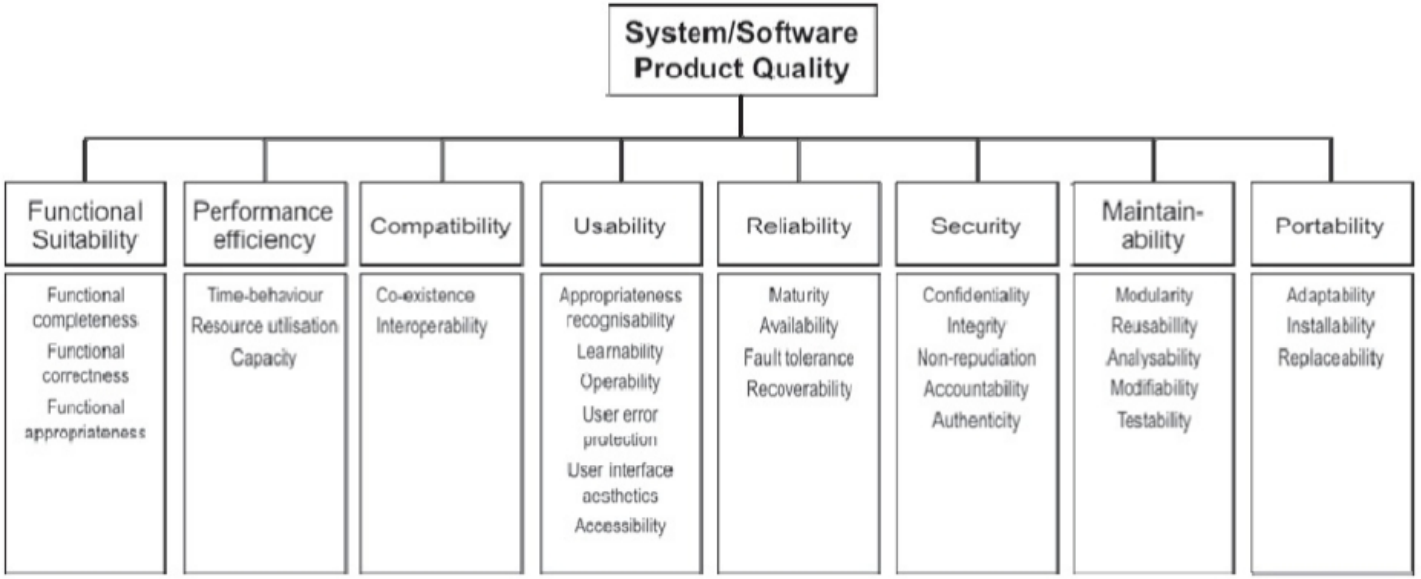


Figure 4 — Product quality model

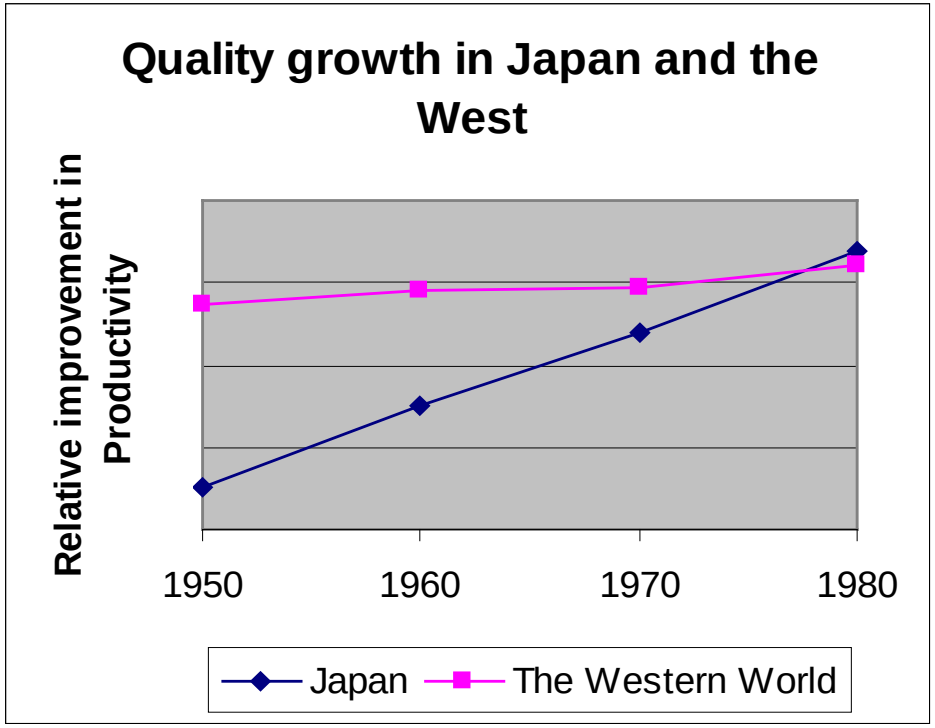
Historical perspective

- First look at where many of the software quality ideas come from.
- Origins are in manufacturing.

- Brief historical look at manufacturing:
  - US
  - Japan

Historical perspective

- Brief historical look at manufacturing:
  - adapted from Juran 1981



Quality themes

- Deming:
  - Quality is conformance to specification
    - *Get the specs right and build according to them. Basis of the 'waterfall' approach to software development.*
  - You cannot inspect quality into a product – it must be built in from the start
    - *Inspecting (testing) finds what is wrong with the product, which then has to be reworked. Too late. For developer, it's your job to make sure what is passed on to testing team or another developer is of high quality.*
  - Most problems are a result of the system (process) – don't blame the workers, it's a management problem
    - *Consider: team of 5 developers, 4 experienced and 1 new. The new one is in charge of keeping the specification wiki up to date. Hmmm…*

Quality themes

- Juran:
  - Quality is fitness for use (issues of 'grade')
    - *Seen in agile approaches to software development i.e. work closely with the client to make sure you deliver what (s)he really wants.*
    - *Developer must understand required quality characteristics.*
  - Workers must be empowered by support of top management
    - *Basis of TQM initiatives.*
    - *HUGE problem in many (most?) software development initiatives.*

http://www.toyota.co.jp/ev/vision/production_system

Quality themes

- Pareto:
  - Only a few factors are responsible for most of the problems
  - *If you want to improve your software process, identify the small number of areas that will make the biggest difference.*

- Toyota:
  - Eliminate waste (inventory, processing steps)
  - *Basis of the lean approach to software development.*

http://www.toyota.co.jp/ev/vision/production_system

Quality themes and SE

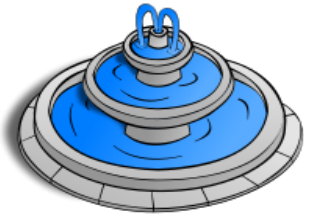
- Underlying belief is : Good process -> quality product
  - This idea is behind all the (heated) discussions about software process. For example, should an organisation implement a waterfall approach? XP? Perhaps Lean Software Development is the way to go?
  - Each of these approaches implements some of the ideas from above.
  - Current thinking is that practices must be adapted to the particular circumstances of the project.

Key developer practices

- Two key developer practices that support quality outcomes are:
  - version control
  - testing

Version Control Best Practices

1. Complete **one change at a time** and commit it
   - If you committing several changes together you cannot undo/redo them individually
   - If you don't commit and your hard disk crashes…
2. **Don't break the build**
   - Test your changes before committing
3. Commit **only the source** files (e.g. not `.class` files)
4. **Use the log** by writing a summary for each commit
   - What has been changed and why
5. **Communicate** with the other developers
   - See who else is working on a part before changing it
   - Discuss and agree on a design
   - Follow the project guidelines & specifications



- Black box
  - Design tests from the specifications only (no knowledge of code structure)
    - Tester must understand the user perspective
    - Independent tester? Or developer with domain knowledge?
  - Techniques
    - Equivalence partitioning (split the input into partitions, where values in each partition can be viewed as being 'similar')
    - Boundary value analysis (for each partition, choose values at the boundaries over those in the middle of the partition)

- Black box

Thoughts
Perhaps if the software-under-test is an application, someone who understands the users viewpoint will be more effective?
Is this technique really appropriate for within-development? What if the software-under-test is an API? Who is the user?
In a way, Black box testing can be viewed as testing interfaces – between human user and application, system interfaces, development modules, …
Can we use only for functionality? Or can we use to test other quality characteristics (efficiency, reliability, …)?

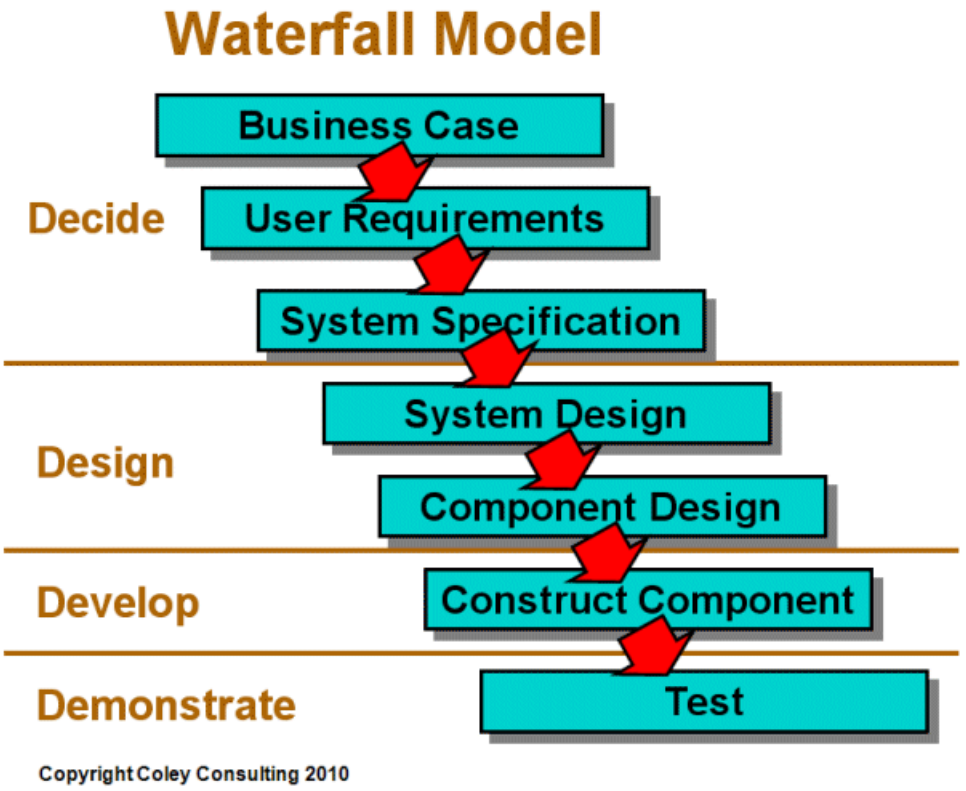
- White box
  - Design tests from a knowledge of code structure
    - Tester must be familiar with programming language
    - Developer ? (BEFORE submitting code)
  - Logic path techniques (in order of strength)
    - Statement coverage
    - Decision coverage
    - Decision/condition coverage

- White box

Thoughts
Developer unwillingness to find defects in his or her own work (Myers' psychology of testing).
What if the code is wrong in the first place (developer didn't understand the specification)? Tests will pass when carried out by developer, but QA may later reject code when testing from the specs.
It might be difficult to find inputs that will force a test to take a specific path. Even more difficult to be sure every path is taken (recognising 'dead code' is surprisingly difficult in a procedural programming language such as Java or C).
Aim to exercise the most likely usage paths? Is the developer the best person to know this? Is this where Black box testing comes in?

Waterfall model

- Staged model. Each stage :
  - implemented by different people with different skill sets
  - must be completed and 'signed off' before the next begins
  - **verified** against the previous stage before sign-off



http://www.coleyconsulting.co.uk/from-waterfall-to-v-model.htm

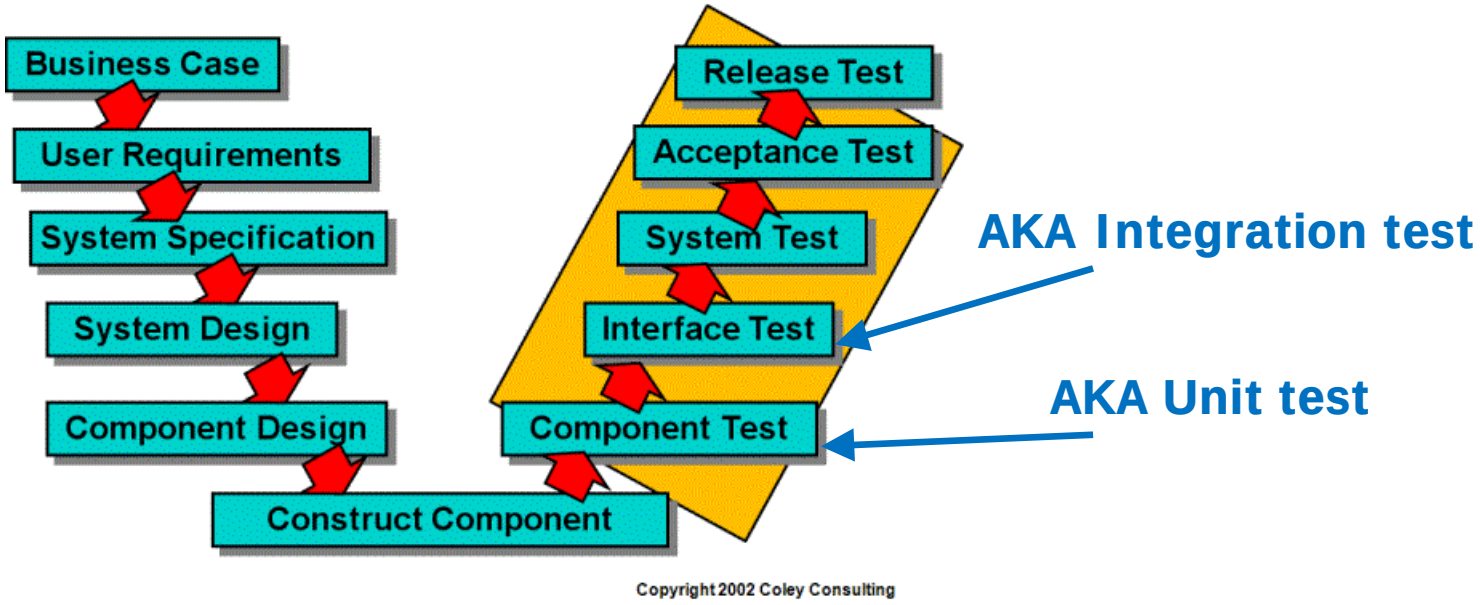
Waterfall model - verification

- At each stage, what is needed to **verify** that the product is being built according to what is stated in the previous stage (*Are we building the product right*)?
  - Documents – requirements specs, design specs, code, test cases
    - Stringent specification includes quality requirements
  - Process – reviews, walkthroughs, inspections

http://softwaretestingfundamentals.com/verification-vs-validation/

V-model for testing

- Staged model testing : Each stage
  - has a corresponding kind of test



http://www.coleyconsulting.co.uk/from-waterfall-to-v-model.htm

V-model for testing

- System, acceptance and release tests aim to **validate** that the software does what the customer wants it to do *(Did we build the right product?)*

- System test
  - Does the software deliver to the specification? Test team.
- Acceptance test
  - Does the software deliver what the customer wanted? Customer.
- Release test
  - Does the software work in the existing business environment? Operations team.

http://softwaretestingfundamentals.com/verification-vs-validation/

Testing quality characteristics

- Testing relating to quality characteristics:
  - Load testing – apply maximum loads to test maximum capacity.
  - Stress testing – find breaking point by applying over the maximum load.
  - Usability testing – measure how quickly users
    - learn to use the system
    - complete specific tasks
    - etc.
  - Reliability testing
  - Portability testing
  - etc.

Other testing

- Other kinds of testing :
  - Smoke testing. During integration, before the product is handed over to the test team, a superficial check is made by the build person that the product's basic features do what they are supposed to. Purpose is, of course, to not waste the test team's time.
  - Regression testing. Applied when changes to the product are needed (to fix bugs or add functionality) to make sure nothing is broken. Applied at unit, integration and system test levels,

Waterfall summary

- Waterfall is a staged approach based on a manufacturing paradigm.
- Created to address problems in large, complex development efforts.
- Communication is largely via documentation.
- Serious issues relating to documentation, communication and delivering what the customer really wanted.

Agile alliance

- Many practitioners explored ways to mitigate issues
  - Many (most?) projects actually implemented an iterative and incremental approach.
    - 1970s: Harlan Mills - upfront specification, deliver in many increments.
      - adapt designs as a result of customer feedback.
    - 1976: Tom Gilb formally introduced ideas of 'evolutionary project management'.
      - no upfront specification, rather discover requirements in an iterative way.
- In 2001, a number of separate groups working on 'agile' approaches to software development formed the **Agile Alliance**.

http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf
http://www.agilealliance.org/

XP testing

- XP is one popular agile methodology
  - Unit testing - create the unit (module) and acceptance tests before coding - *Extreme Testing* (XT)."
    - *? Isn't a more common name "Test Driven Development" (TDD) ?*
    - *? Did Kent Beck really invent the idea ?*
    - *? Was TDD originally part of XP ?*
  - Acceptance testing – carried out at the end of each iteration by the customer - *Extreme Acceptance Testing* (XAT)."
    - *? What if there's more than one customer?*
    - *? What if the stakeholders disagree about requirements?*

Myers's Summative Evaluation of XP

- "Although glamorous, XP is not for every project or every organization."
- "Proponents of XP [claim] that the chances of successful application development increase dramatically"
- "Detractors say that because XP is a process, you must do all or nothing. If you skip a practice, then … your program quality may suffer."
- "[D]etractors claim that the cost of changing a program in the future to add more features is more than the cost of initially anticipating and coding the requirement."
- "[S]ome programmers find working in pairs very cumbersome and invasive, therefore they do not embrace the XP methodology."

Where are we now?

- Many development methodologies have been proposed
  - Each tends to gather a number of zealots who:
    - advocate application of the methodology under all circumstances
    - spend significant time 'proving' the methodology works under all circumstances.
  - Architects originally stated that all practices must be carried out:
    - each supports the others
    - if you omit some, there will be gaps in the process
- The current wisdom is that no one approach is guaranteed to be effective in all cases
  - Each organisation must tailor a development process suitable for its specific contexts
  - 'Agile' has morphed to mean 'flexible'
- When the next 'silver bullet' arrives (as it surely will), you should ask 'what is the evidence?'