CompSci 230
Software Design and Construction

Software Quality 2015S1

Traditional approach to testing (Waterfall)

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

- Have a working understanding of :
  - the waterfall model for software development
  - testing in the waterfall model
  - iterative, incremental and evolutionary development

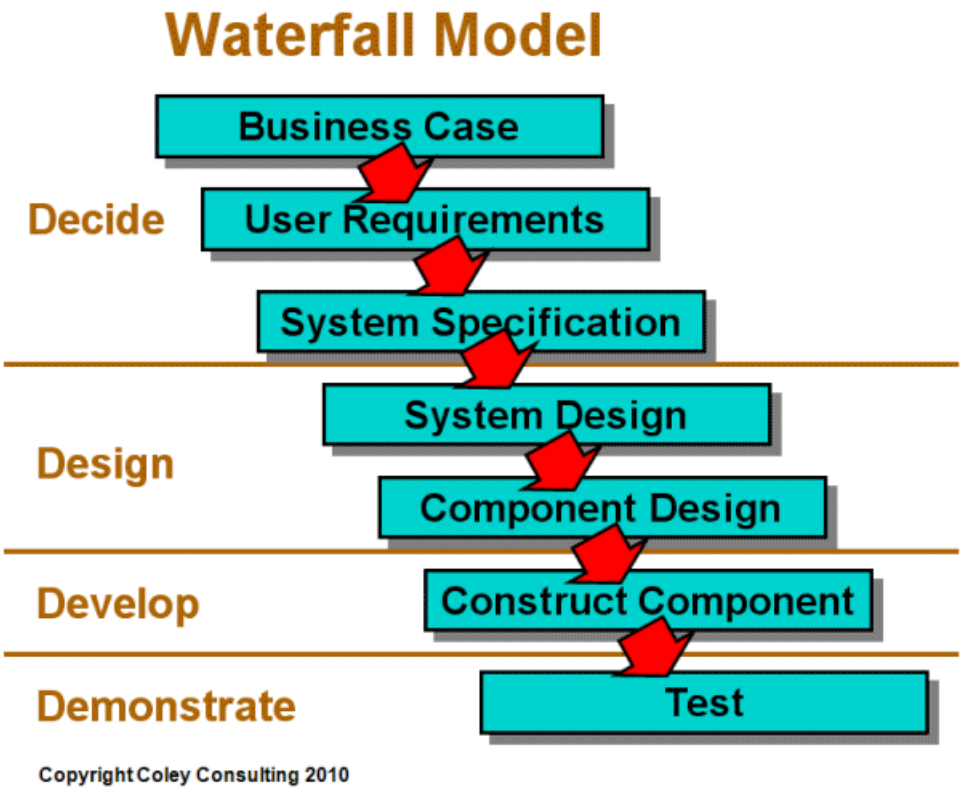
- Discuss:
  - limitations of the waterfall model
  - agile alliance and manifesto

Brief history of waterfall

- 1960s : programming is an 'art'
  - practitioners receive no formal training
  - serious concerns about quality as projects became larger
- 1968 : conference organised by the NATO Science Committee
  - discussed issues with 'software manufacture'
  - coined the term 'software engineering'
  - introduced standard development model (waterfall) based on staged manufacturing process
- Note : many 'current' issues discussed (need to iterate, obtain feedback from customer, reuse, product architecture) BUT appears to have been little attempt to reframe software development as anything other than a manufacturing process.

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
  - Process – reviews, walkthroughs, inspections

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

Waterfall model - documents

- Documents play a critical role in the verification process.
- Document standards :
  - *IEEE Recommended Practice for Software Requirements Specifications (SRS)*
  - etc.

IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications

Waterfall model - requirements

- A template from the IEEE SRS.
- Note that, in addition to the features, you must consider external product interfaces and non-functional requirements.

A.5 Template of SRS Section 3 organized by feature

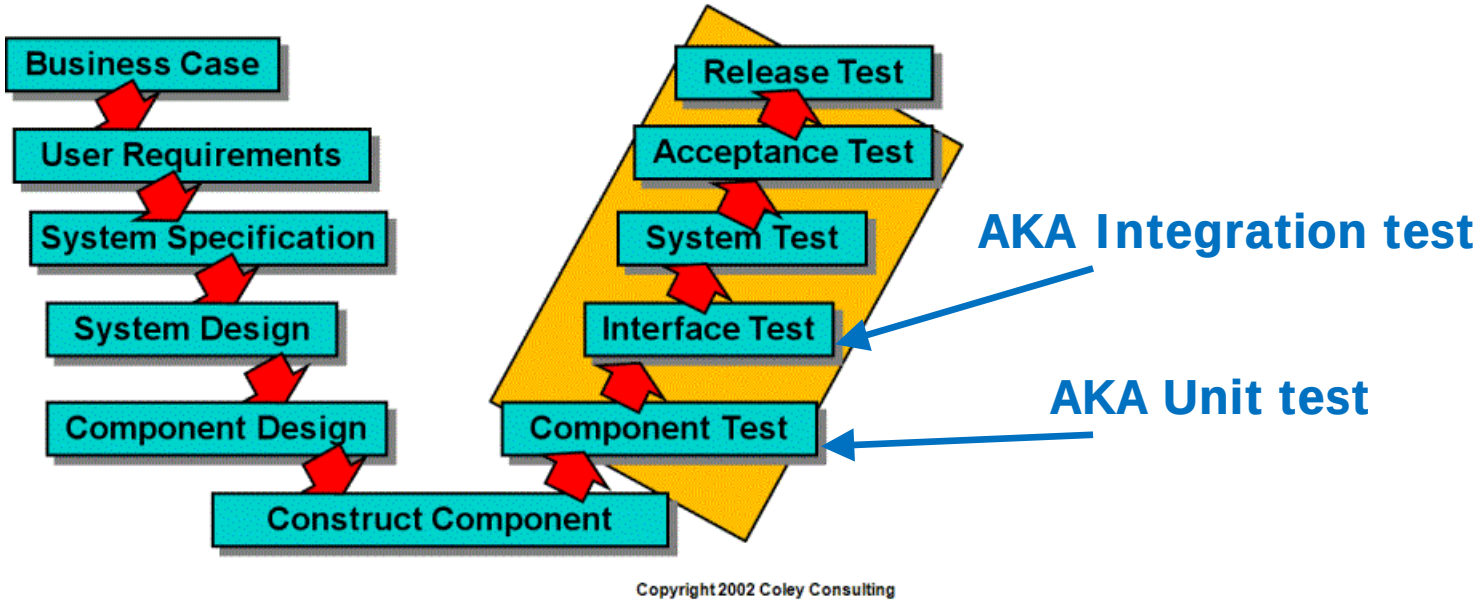
3. Specific requirements

- 3.1 External interface requirements
  - 3.1.1 User interfaces
  - 3.1.2 Hardware interfaces
  - 3.1.3 Software interfaces
  - 3.1.4 Communications interfaces
- 3.2 System features
  - 3.2.1 System Feature 1
  - 3.2.1.1 Introduction/Purpose of feature
  - 3.2.1.2 Stimulus/Response sequence
  - 3.2.1.3 Associated functional requirements
  - 3.2.1.3.1 Functional requirement 1
  - .
  - 3.2.1.3.n Functional requirement n
  - 3.2.2 System feature 2
  - .
  - 3.2.m System feature m
  - .
- 3.3 Performance requirements
- 3.4 Design constraints
- 3.5 Software system attributes
- 3.6 Other requirements

IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications

V-model for testing

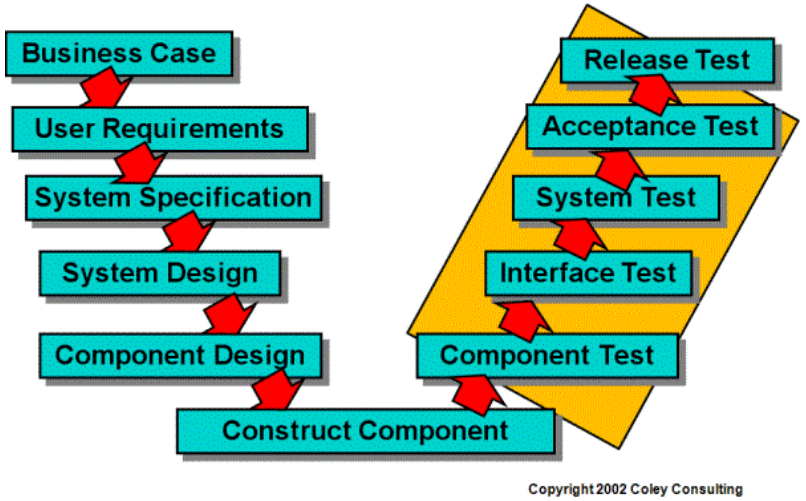
- Staged model testing : Each stage
  - has a corresponding kind of test



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

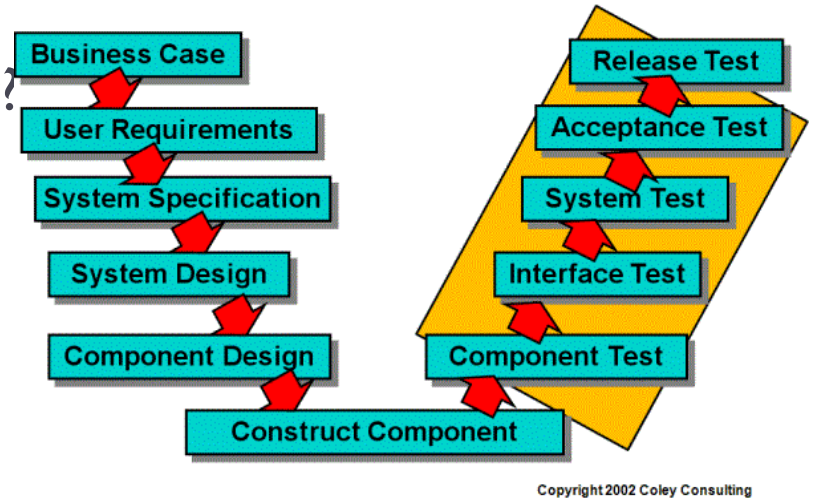
V-model for testing

- Component (unit) test
  - Does the component implement the design?
- Performed by
  - developer OR
  - independent tester
- Issues
  - Cannot catch every bug in a component. Impossible to test
    - every combination of inputs (black box)
    - every execution path (white box)



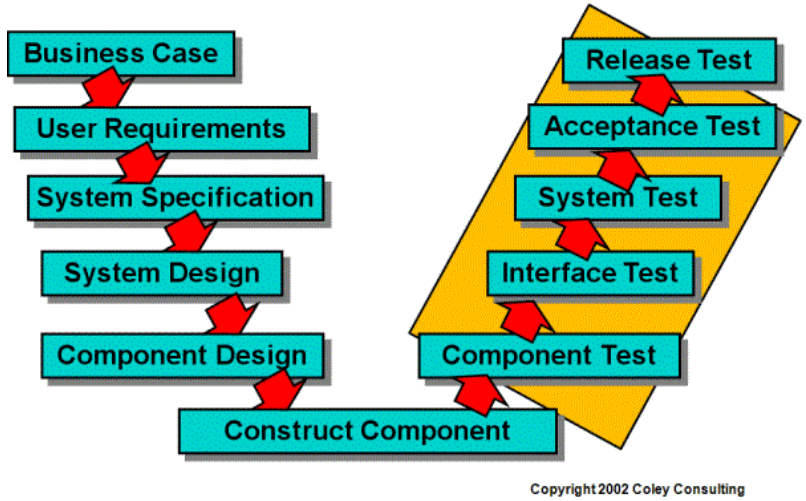
V-model for testing

- Interface (integration) test
  - Do the components work with each other?
- Performed by
  - developer OR
  - independent build person
- Issues
  - Many integration issues for large application
    - interfaces
    - misunderstanding about functionality



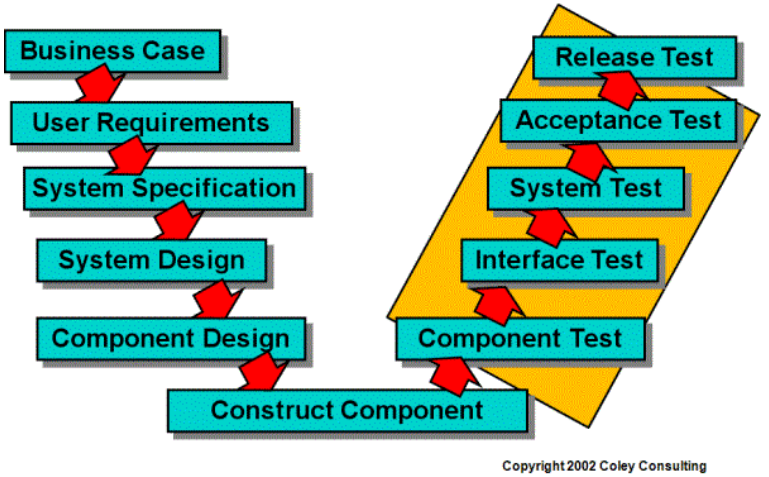
V-model for testing

- System test
  - Does the software deliver to the specification (functional and non-functional requirements)?
- Performed by
  - specialised test team
- Issues
  - Can be difficult to replicate user's (production) environment



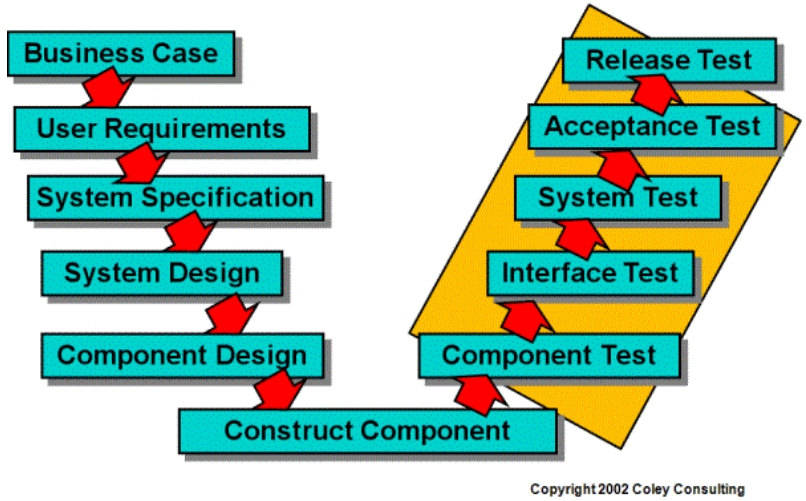
V-model for testing

- Acceptance test
  - Does the software deliver what the customer wanted?
- Performed by
  - specialised QA team and/or
  - customer (alpha releases)
- Issues
  - Can be difficult to replicate user's environment
    - use low spec machines to test e.g. latency issues



V-model for testing

- Release test
  - Does the software work in the existing business environment?
- Performed by
  - operations team and/or
  - customer (beta releases)
- Issues
  - Customers can be busy and don't want to be interrupted
  - If beta testing, need committed users



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/

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 - issues

- Practitioners uncovered some serious issues when implementing a waterfall approach :
  - During projects lasting several years, clients often changed their minds about what was required. The wrong product was delivered.
    - changes in environment
    - introduction of new technologies
  - The need for extensive documentation resulted in documents not being kept up-to-date.
    - e.g. during design phase, mistake in requirements document is discovered – fixed in design doc but not in requirements doc.
  - No communication between practitioners from different phases meant that tacit knowledge wasn't shared. Coders often didn't really grasp what was wanted.

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/

Agile alliance

- Mission

  "We support those who explore and apply Agile principles and practices to make the software industry productive, humane and sustainable."

- Manifesto

  "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

  ***Individuals and interactions** over processes and tools*
  ***Working software** over comprehensive documentation*
  ***Customer collaboration** over contract negotiation*
  ***Responding to change** over following a plan.*

  That is, while there is value in the items on the right, we value the items on the left more."

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

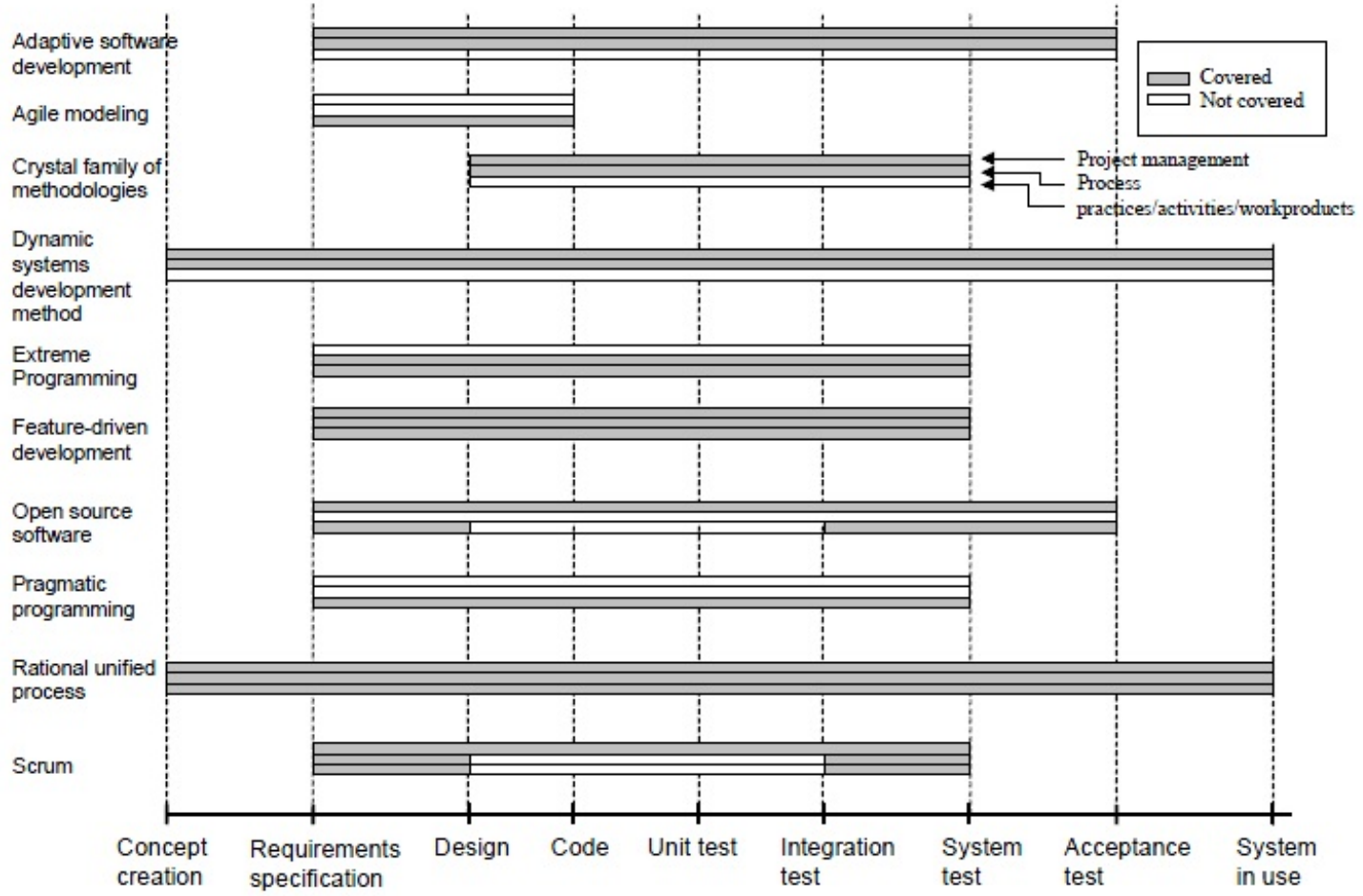
Agile alliance

- Principles:

1. Customer satisfaction by rapid delivery of useful software
2. Welcome changing requirements, even late in development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Self-organizing teams
12. Regular adaptation to changing circumstances

http://www.agilealliance.org/

Agile methods



http://en.wikipedia.org/wiki/Agile_software_development

Agile approach

- Agile approach is based on a software-as-a-service paradigm.
- Communication is largely face-to-face.
- Software is delivered frequently to customers.
- The agile methods are quite different from one another but have the Principles in common.
- Next session, we will study one agile method, eXtreme Programming (XP), in greater detail.