Levels of Testing

Chapter 12

Beyond unit testing

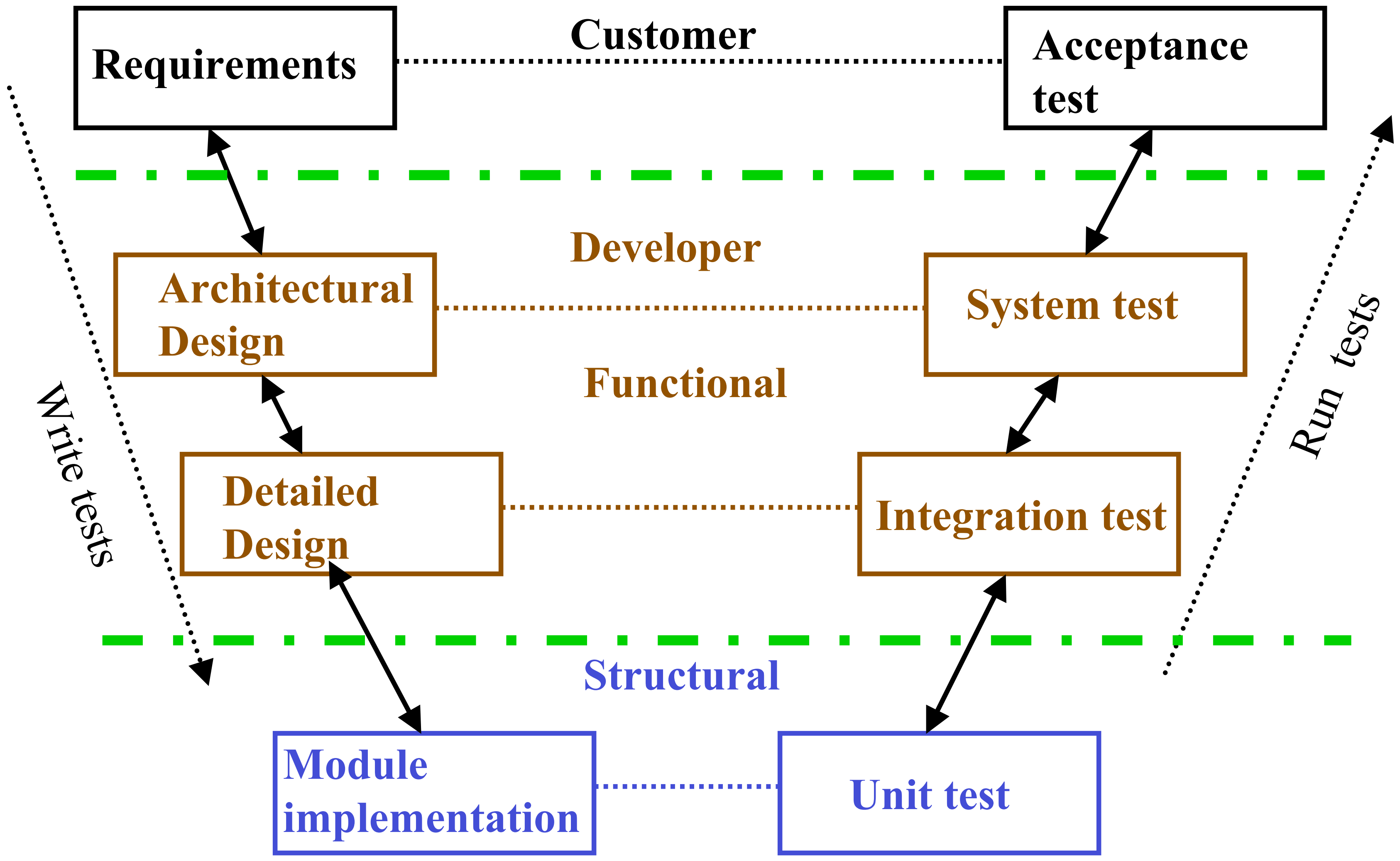
Testing stages

- Developer
  - Unit testing
    - Testing of individual components
  - Integration testing
    - Testing to expose problems arising from the combination of components
  - System testing
    - Testing the complete system prior to delivery
- Customer
  - Acceptance testing
    - Testing by users to check that the system satisfies requirements. Sometimes called alpha testing

Life cycle models

- Traditional waterfall
  - **Levels correlate with levels of testing**
  - **Functional testing is implied**
  - **Bottom up testing is implied**
- Unit, integration, system
  - **Unit is best understood**
    - **Have both functional and structural testing**
  - **System is understood**
    - **Functional testing**
      - **No good structural notation for descriptions**
  - **Integration**
    - **Bottom up – combine smaller units into larger ones, until system level is reached**

V-Model – development & testing



Development styles

- Top-down
  - **Build upper level**
  - **Test using stubs**
    - **Throw away**
- Bottom-up
  - **Build lower levels**
  - **Test with drivers**
    - **Throw away**
- Big bang
  - **Build everything**
  - **No stubs or drivers**
  - **Then test**

Problems with waterfall model

- Waterfall
  - Too slow
  - Too rigid
  - Too focused on top-down functional development and bottom-up testing
  - Not the way people work
  - Staffing levels of different types batched at different times with the levels requiring large resource shifts from low to high and back.

Waterfall spin-off models

- Development in stages
  - **Level use of staff across all types**
  - **Testing now entails both**
    - **Regression**
    - **Progression**
- Main variations involve constructing a sequence of systems
  - **Incremental**
  - **Evolutionary**
  - **Spiral**
- Waterfall model is applied to each build
  - **Smaller problem than original**
  - **System functionality does not change**

Waterfall spin-off models – 2

- Incremental
  - Have high-level design at the beginning
  - Low-level design results in a series of builds
    - Incremental testing is useful
    - System testing is not affected
  - Level off staffing problems
- Evolutionary
  - First build is defined
  - Priorities and customer define next build
  - Difficult to have initial high-level design
    - Incremental testing is difficult
    - System testing is not affected

Waterfall spin-off models – 3

- Spiral
  - **Combination of incremental and evolutionary**
  - **After each build assess benefits and risks**
    - **Use to decide go/no-go and direction**
  - **Difficult to have initial high-level design**
    - **Incremental testing is difficult**
    - **System testing is not affected**
- Advantage of spin-off models
  - **Earlier synthesis and deliverables**
  - **More customer feedback**

Rapid prototyping

- Specification based life cycle model
- Build quick and dirty system
  - **Good for risk analysis**
  - **Customer feedback**
- System testing is difficult
  - **Where is the specification?**
- Good for acceptance testing
  - **Emphasis is behaviour, not structure**

Executable specifications

- Specification based life cycle model
- Extension of rapid prototyping
- Specific behaviourial models are build and executed
  - **Statecharts**
  - **Finite state machines**
  - **Petri nets**
- Customer feedback as for rapid prototyping

Integration & system testing

- Need to know difference between integration and system testing
  - **Avoid testing gaps and redundancies across levels**
  - **Set testing goals appropriate for each level**
- Structural & behavioural views separate integration and system testing goals

Threads

- Threads
  - **Use cases, describe behaviour**
  - **Have at different levels**
- System level
  - **Data context and sequence of port events**
- Integration
  - **Path in state machine**
- Unit
  - **Path in program graph**

Structural insights

- Integration testing
  - Assumes unit level testing completed
  - Can be seen as interface testing
    - What about algorithms at higher levels?
  - Uses preliminary design
- System testing
  - Requirements level
  - What is the difference between requirements and preliminary design?
  - What-how and other definitions too vague
    - Inevitability of intertwining specification and design

Behavioural insights

- System level
  - Deals with port boundaries
    - What the user sees and does
  - Sequences of integration-level threads
- Integration level
  - Deals with boundaries between port and unit
    - Within the system
  - Sequences of unit-level threads