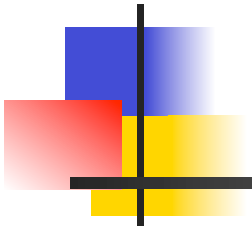
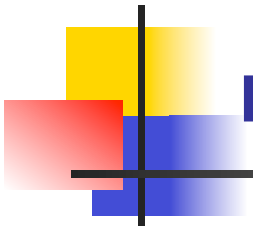


Levels of Testing

Chapter 12

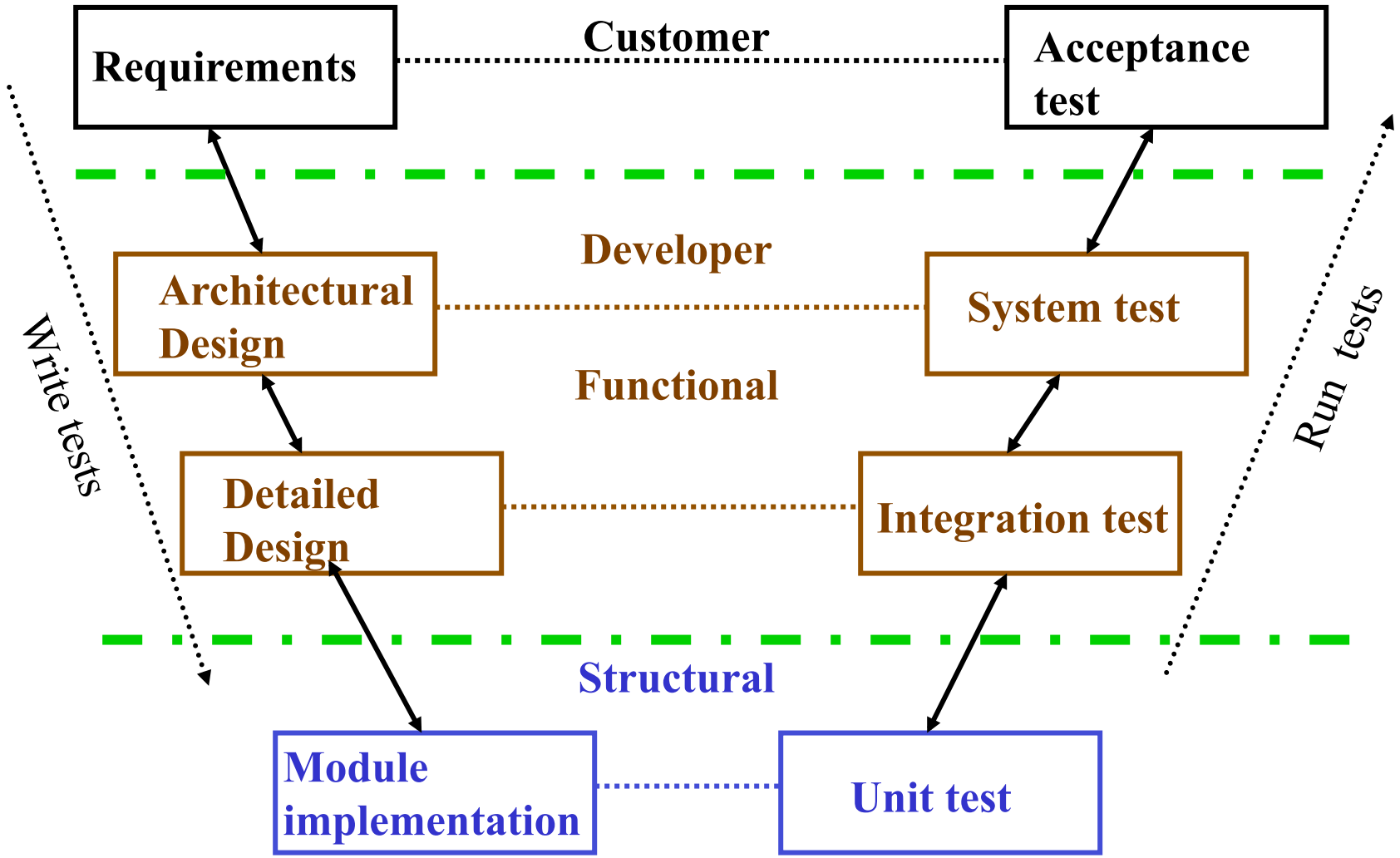
Beyond unit testing

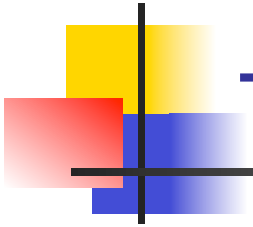


Life cycle models

- What is a life cycle model of software development?
- What is the traditional life cycle model?

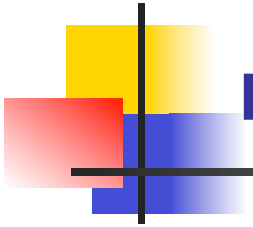
V-Model – development & testing





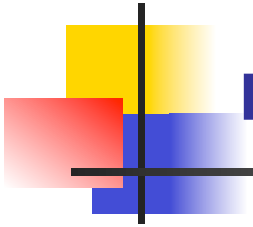
Traditional model

- Waterfall
  - **Levels correlate with levels of testing**
  - **Functional testing is implied**
  - **Bottom up testing is implied**



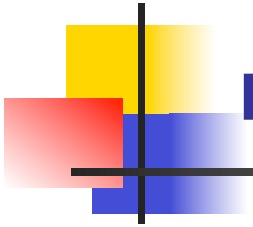
Developer – unit testing stage

- Testing of individual components
- Unit is best understood
- Have both functional and structural testing



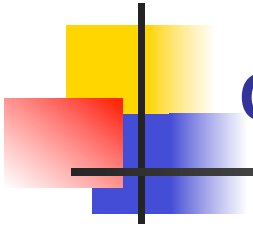
Developer – integration testing stage

- Testing to expose problems arising from the combination of components
- Bottom up
  - **Combine smaller units into larger ones, until system level is reached**



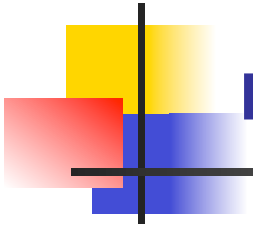
Developer – system testing stage

- Testing the complete system prior to delivery
- Functional testing
- No good structural notation for descriptions



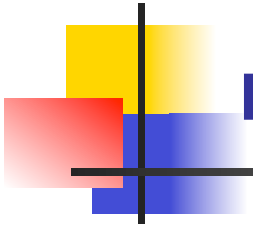
Customer testing stage

- Acceptance testing
  - **Testing by users to check that the system satisfies requirements. Sometimes called alpha testing**



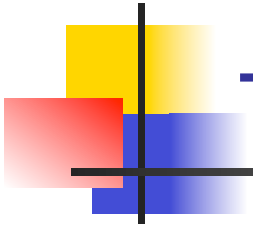
Basic development methods

- **What are the three basic methods that can be used to develop a system?**



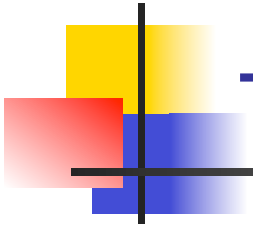
Basic development methods – 2

- Top down
- Bottom up
- Big Bang



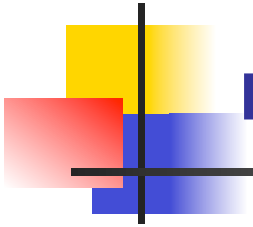
Top-down development

- How does it work?



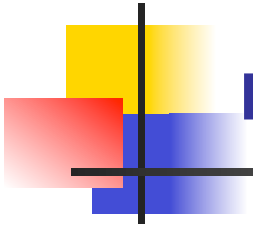
Top-down development – 2

- Build upper level
- Test using stubs
  - **Throw away**



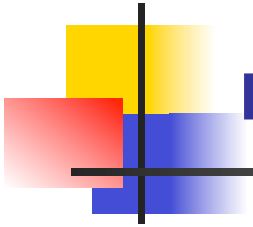
Bottom-up development

- How does it work?



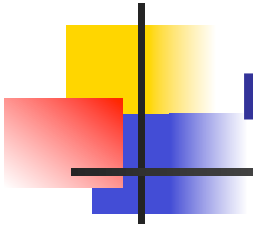
Bottom-up development – 2

- Build lower levels
- Test with drivers
  - **Throw away**



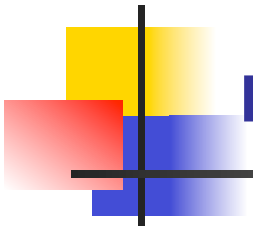
Big bang development

- How does it work?



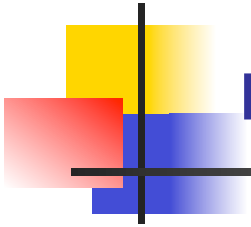
Big bang development

- Build everything
- No stubs or drivers
- Then test



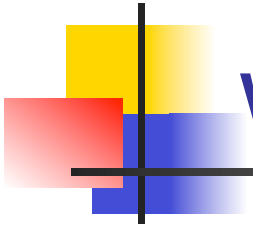
Problems with waterfall model

- What are they?



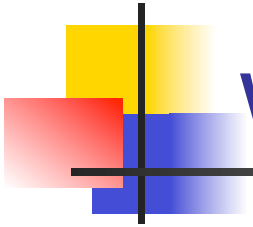
Problems with waterfall model – 2

- 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.



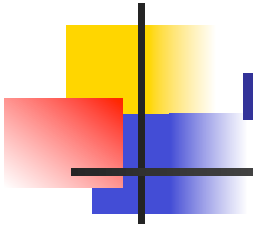
Waterfall spin-off models

- Development in stages
  - **Level use of staff**
  - **Testing now entails both**
    - **Regression**
    - **Progression**



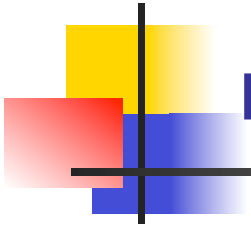
Waterfall spin-off models – 2

- 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 during a build**



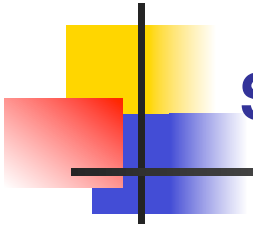
Incremental model

- 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



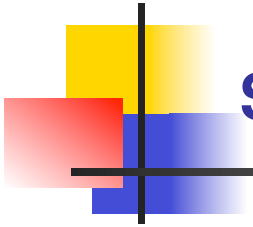
Evolutionary model

- 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**



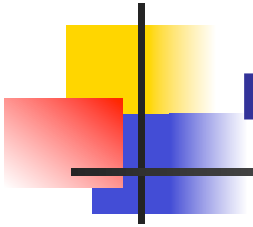
Spiral model

- 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**



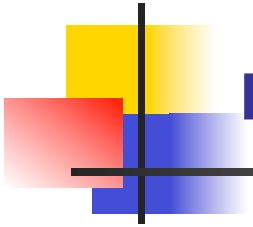
Spiral model – advantage

- Earlier synthesis and deliverables
- More customer feedback
- Risk/benefit analysis is rigorous



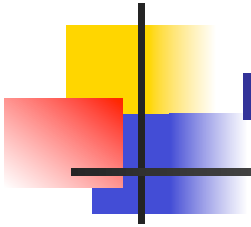
Rapid prototyping

- 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**



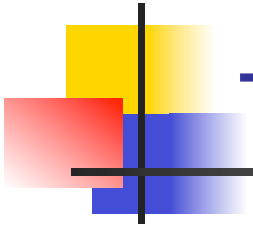
Executable specifications

- Specification based life cycle model
- Extension of rapid prototyping
- Specific behavioural models are built and executed
  - **Statecharts**
  - **Finite state machines**
  - **Petri nets**
  - **Z specification language**
- Customer feedback as for rapid prototyping



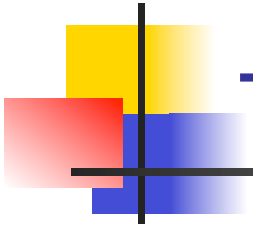
Integration & system testing

- 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

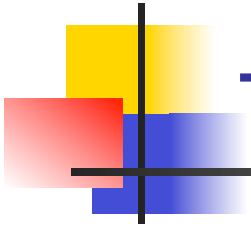
- **What are they?**



Threads – 2

- Use cases
- Describe behaviour

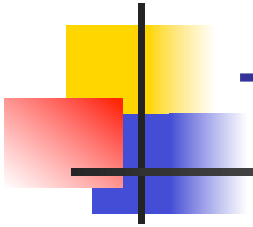
- Have threads at different levels
  - **What are the levels?**



Threads – 3

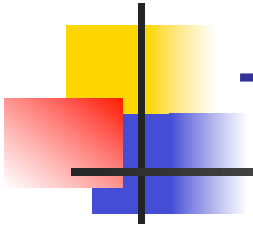
- Use cases
- Describe behaviour

- Have threads at different levels
  - **What are the levels?**
    - **System**
    - **Integration**
    - **Unit**



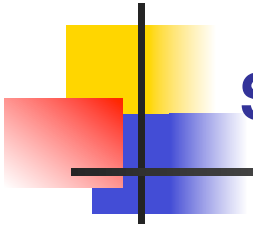
Thread levels

- What are the threads at each level?
  - System
    - ???
  - Integration
    - ???
  - Unit
    - ???



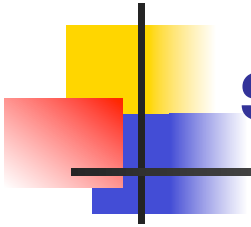
Thread levels

- System level
  - **Data context and sequence of port events**
- Integration
  - **Path in a finite state machine**
- Unit
  - **Path in a program graph**



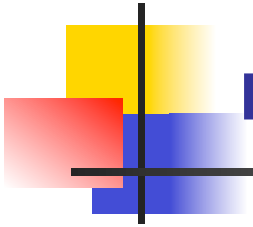
Structural insights – integration testing

- Assumes unit level testing completed
- Can be seen as interface testing
  - **What about algorithms at higher levels?**
- Uses preliminary design



Structural insights – system testing

- Requirements level
- What is the difference between the following?
  - **requirements**
  - **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**