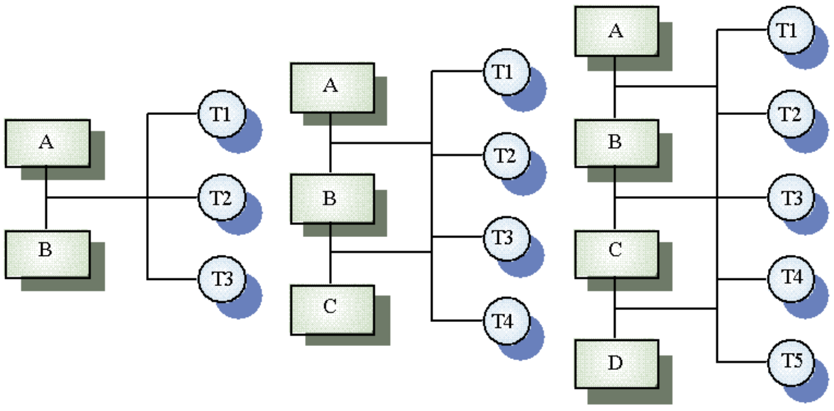
Integration Testing

& other aspects of testing

Integration testing

- Tests subsystems composed of integrated groups of components
  - Eventually the complete system
- Integration testing should be black-box testing
  - Internal structure too complex for white box
  - With tests derived from the specification
- Main difficulty is localising errors
  - The fault could be anywhere in the group
- Careful ***incremental*** *integration testing* reduces this problem
  - Systematically grow the groups being tested
- Again, since we are not testing a *complete system*, we must execute the *components* under test in a *test harness:*
  - Stubs and drivers again

Incremental integration testing



Test sequence 1

Test sequence 2

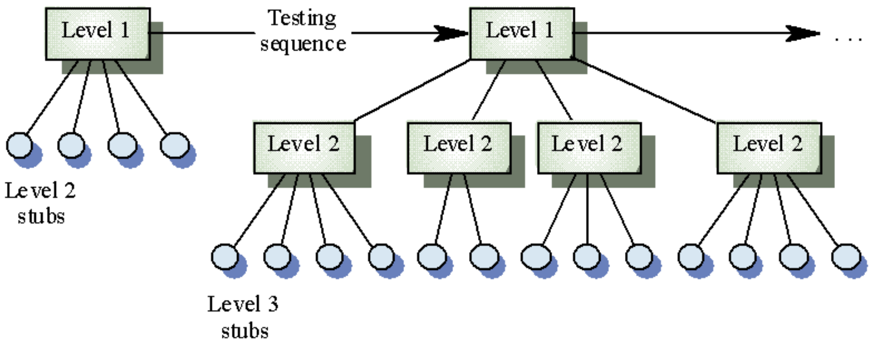
Test sequence 3

Diagrams from slides accompanying Somerville's text book

Approaches to integration testing

- Top-down testing
  - Start with high-level (main, dependent) components and form groups from the top-down
  - Test harness: Individual lower level (depended on) components are replaced by *stubs* to enable this
- Bottom-up testing
  - Aggregate lower level *individual components* into levels until the complete system is created
  - Test harness: Higher level *drivers* must be written to carry out the tests
- In practice integration testing can involve a combination of these strategies
- Bottom-up integration testing is a practical approach
  - Only drivers needed, no stubs

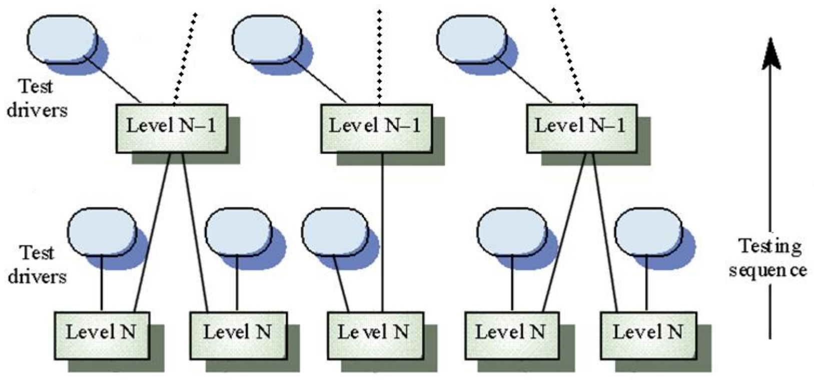
Top-down integration testing



CSCU9P6 Implementation: Testing
© University of Stirling 2019

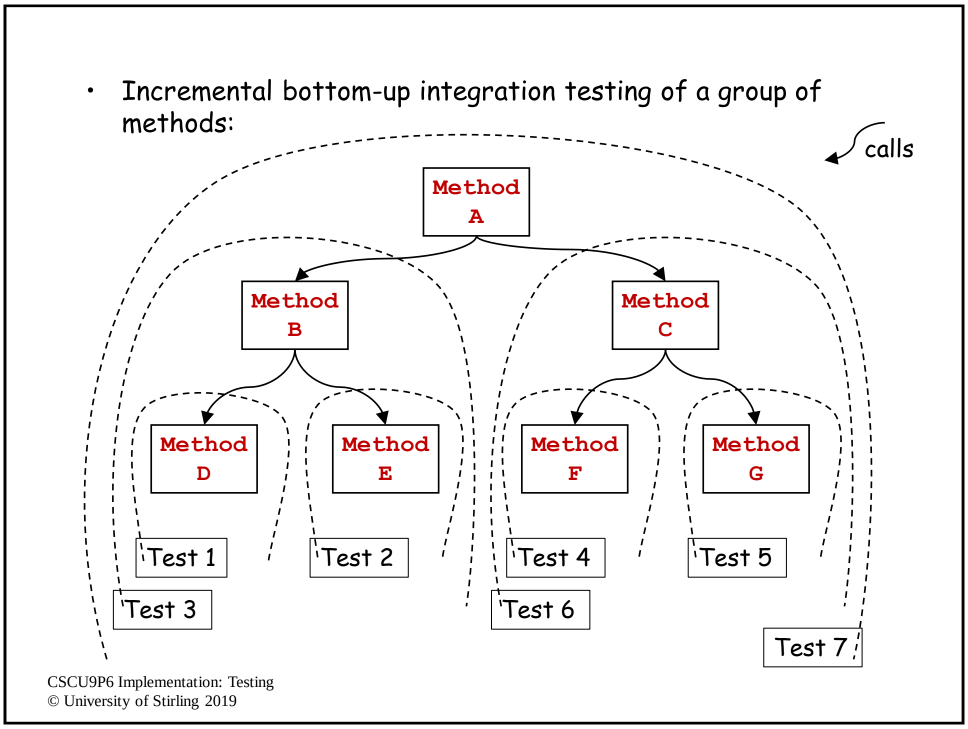
5

Bottom-up integration testing

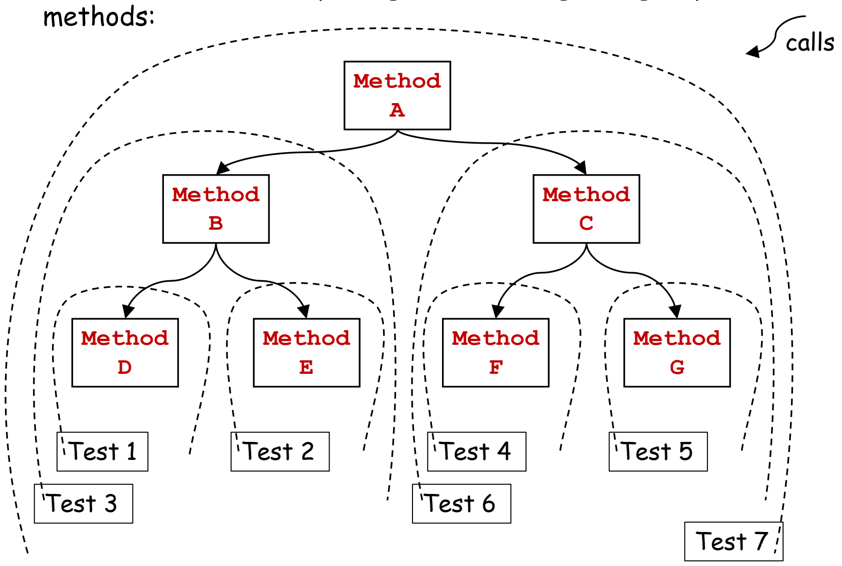


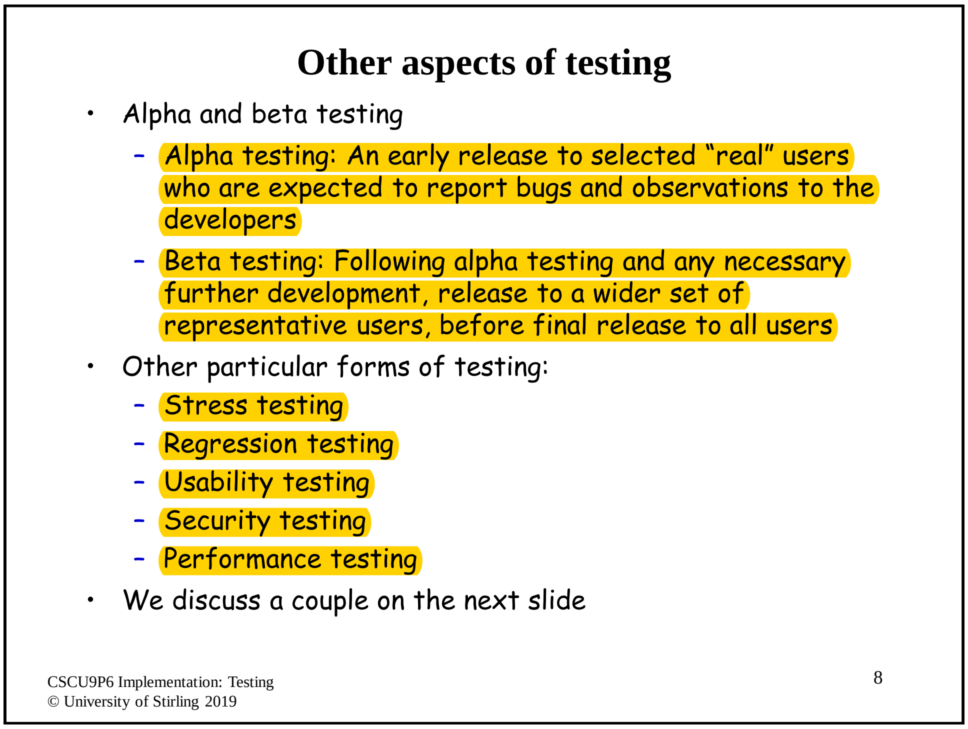
CSCU9P6 Implementation: Testing
© University of Stirling 2019

6



- Incremental bottom-up integration testing of a group of methods:





Other aspects of testing

- Alpha and beta testing
  - Alpha testing: An early release to selected "real" users who are expected to report bugs and observations to the developers
  - Beta testing: Following alpha testing and any necessary further development, release to a wider set of representative users, before final release to all users
- Other particular forms of testing:
  - Stress testing
  - Regression testing
  - Usability testing
  - Security testing
  - Performance testing
- We discuss a couple on the next slide

Stress testing

- Exercises the system beyond its maximum design load
  - Stressing the system tests failure behaviour
  - Systems should not fail catastrophically
  - Stress testing checks for unacceptable loss of service or data
- Stressing the system often causes defects to come to light
- Particularly relevant to distributed systems which can exhibit severe degradation as a network becomes overloaded

Regression testing

- Re-testing the system *following maintenance* to ensure continued correctness
  - Comprehensive test documentation is essential
  - Automated test harnesses and support are valuable

End of lecture