CITS5501 Software Testing and Quality Assurance
System, integration and regression testing

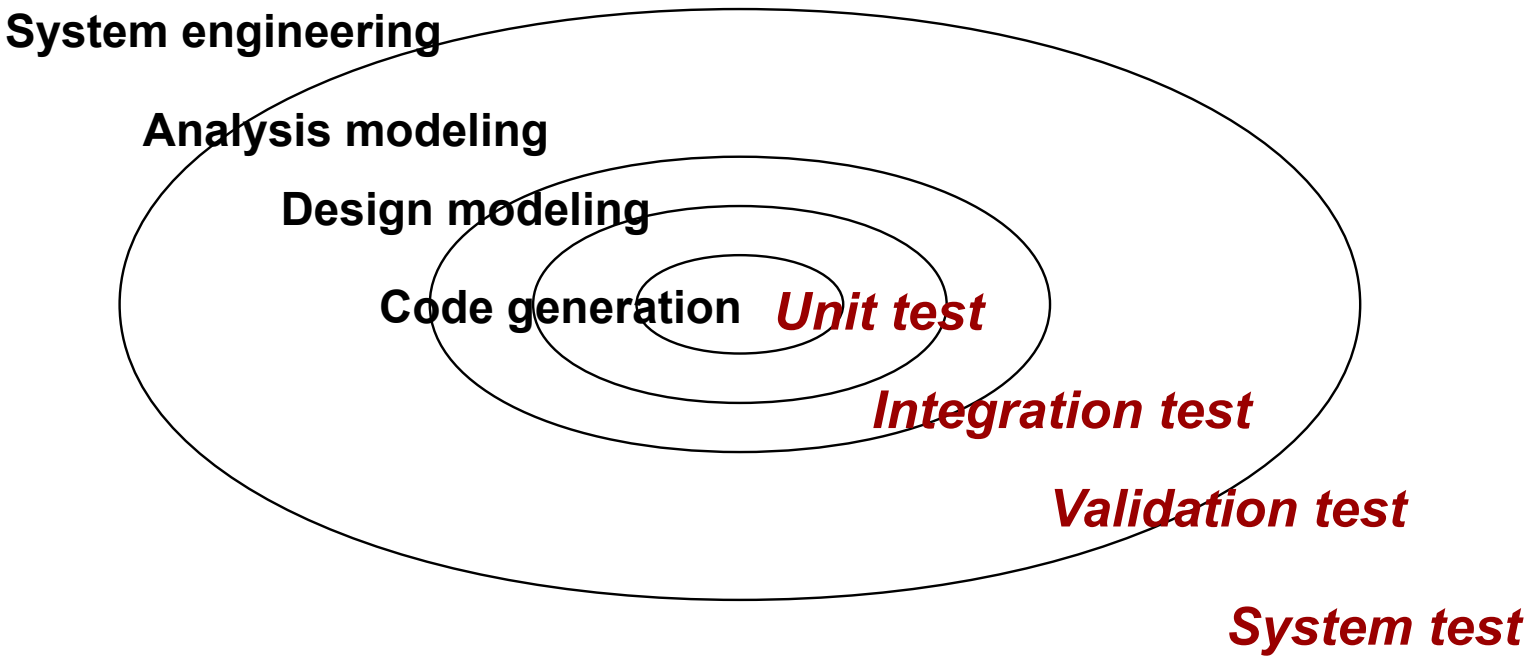
Unit coordinator: Arran Stewart

Overview

- Testing strategy
- Integration testing
- Regression testing
- “Smoke” testing
- Web testing

Testing strategy

Types of testing:



Testing strategy

- Typically, we begin by 'testing-in-the-small' and move toward 'testing-in-the-large'
  - Start with units (functions/classes)
  - Then start integrating them

Testing strategy

- While doing unit testing, we will typically make use of "mocks"/doubles in place of other units or modules
- In integration testing, we can test how units or modules work together

Integration testing

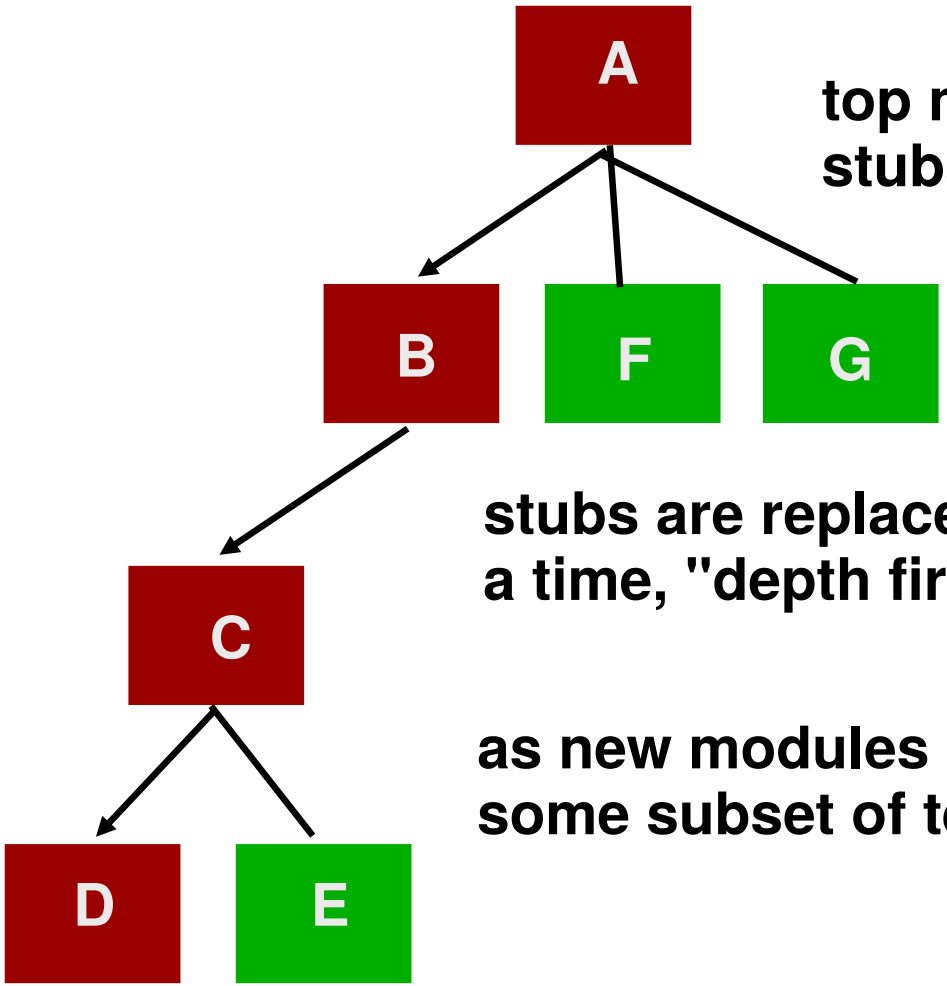
- The entire system is viewed as a collection of subsystems (sets of classes) determined during the system and object design.
- The order in which the subsystems are selected for testing and integration determines the testing strategy

Integration testing strategies

Main options:

- Big bang integration (nonincremental)
- Bottom up integration
- Top down integration
- Sandwich testing
- Variations of the above

Top Down Integration

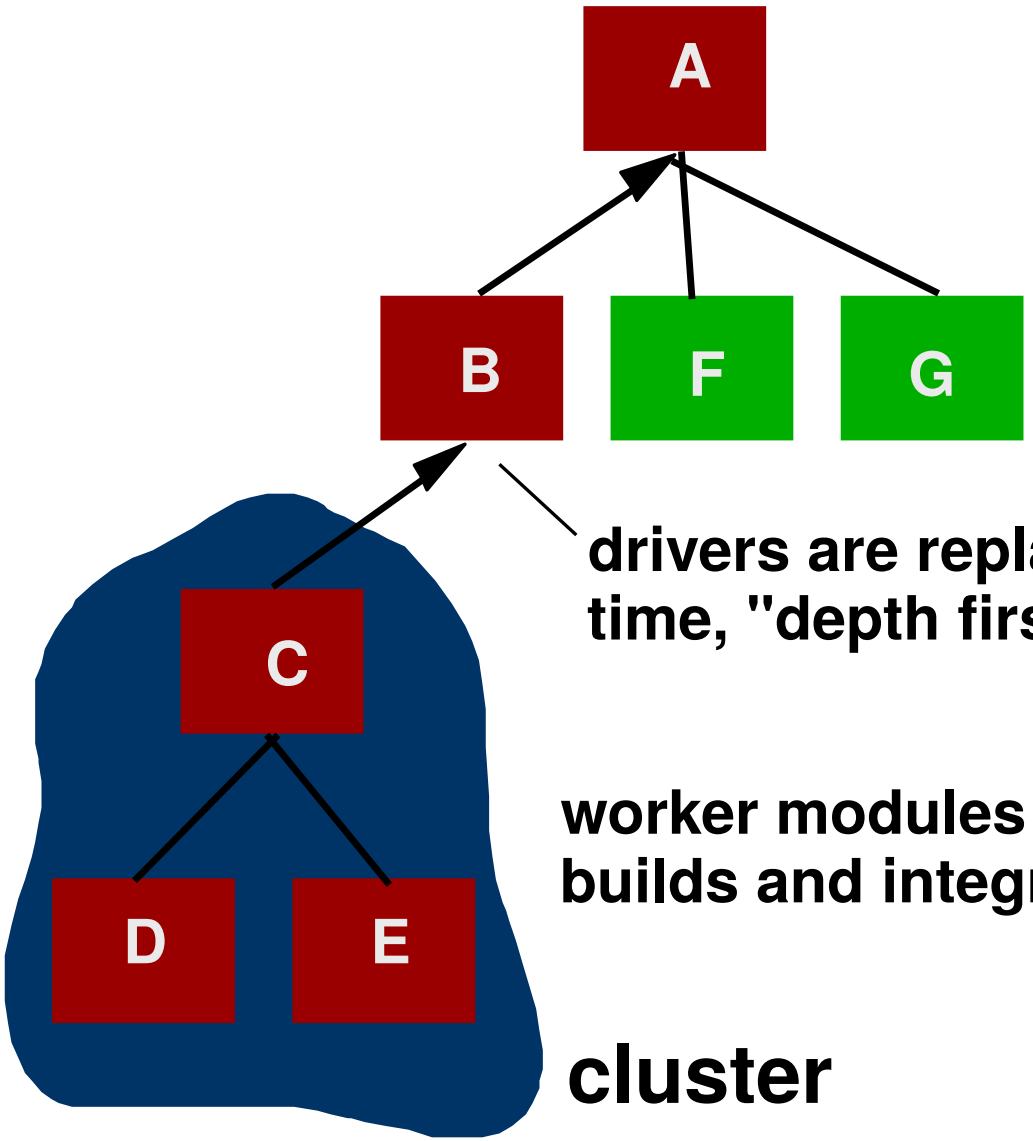


top module is tested with stubs

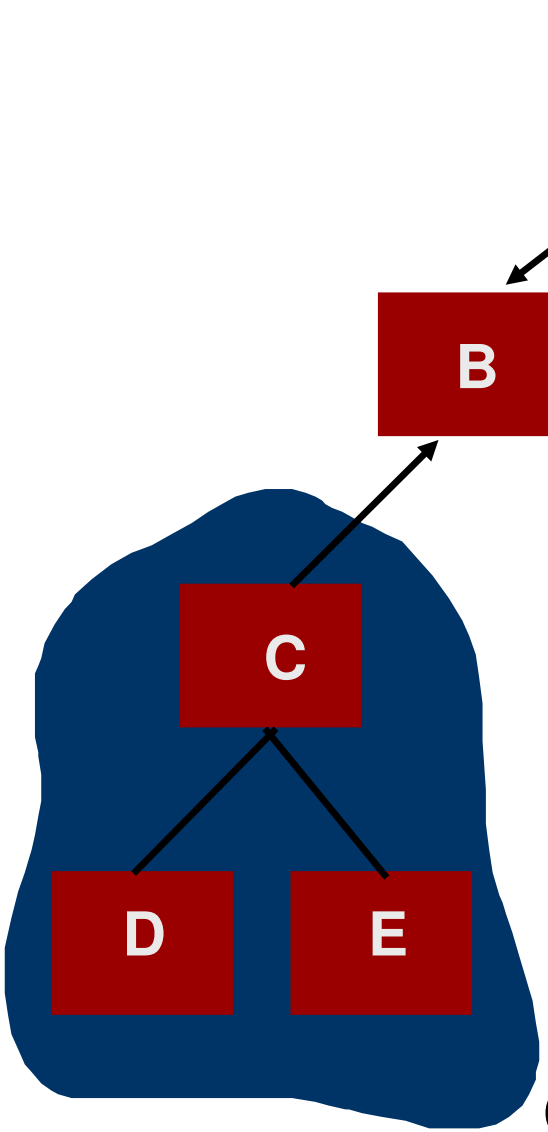
stubs are replaced one at a time, "depth first"

as new modules are integrated, some subset of tests is re-run

Bottom-Up Integration



Sandwich Testing



Top modules are tested with stubs

Worker modules are grouped into builds and integrated

cluster

Pros and cons of bottom up integration testing

- Pro: Systems tested as they are ready
- Con: Typically tests one important subsystem (UI) last

Pros and cons of top-down integration testing

Pro:

- Test cases can be defined in terms of the functionality of the system (functional requirements)

Cons:

- Writing stubs can be difficult: Stubs must allow all possible conditions to be tested.
- Possibly a very large number of stubs may be required, especially if the lowest level of the system contains many methods.
- One solution to avoid too many stubs: Modified top-down testing strategy
  - Test each layer of the system decomposition individually before merging the layers
  - Disadvantage of modified top-down testing: Both stubs and drivers are needed

Steps in integration testing

1. Based on the integration strategy, select a component to be tested. Unit test all the classes in the component.
2. Put selected component together; do any preliminary fix-up necessary to make the integration test operational (drivers, stubs)
3. Do functional testing: Define test cases that exercise all uses cases with the selected component
4. Do structural testing: Define test cases that exercise the selected component
5. Execute performance tests
6. Keep records of the test cases and testing activities.
7. Repeat steps 1 to 7 until the full system is tested.

   The primary goal of integration testing is to identify errors in the (current) component configuration.

Which integration strategy should you use?

- Factors to consider
  - Amount of test harness (stubs &drivers)
  - Location of critical parts in the system
  - Availability of hardware
  - Availability of components
  - Scheduling concerns

Which integration strategy should you use?, cont'd

- Bottom up approach
  - good for object oriented design methodologies
  - Test driver interfaces must match component interfaces
  - Top-level components are usually important and cannot be neglected up to the end of testing
  - Detection of design errors postponed until end of testing

Which integration strategy should you use?, cont'd

- Top down approach
  - Test cases can be defined in terms of functions examined
  - Need to maintain correctness of test stubs
  - Writing stubs can be difficult

Regression testing

- Mentioned in previous lectures:
  - Regression testing is the re-execution of some subset of tests that have already been conducted, to ensure that changes have not propagated unintended side effects
- Whenever software is corrected, some aspect of the software configuration (the program, its documentation, or the data that support it) is changed.
- Regression testing helps to ensure that changes (due to testing or for other reasons) do not introduce unintended behavior or additional errors.
- Regression testing may be conducted manually, by re-executing a subset of all test cases or using automated tools.

Smoke Testing

A common approach for creating "daily builds" for product software
Smoke testing steps:

- Software components that have been translated into code are integrated into a "build."
  - A build includes all data files, libraries, reusable modules, and engineered components that are required to implement one or more product functions.
- A series of tests is designed to expose errors that will keep the build from properly performing its function.
  - The intent should be to uncover "show stopper" errors that have the highest likelihood of throwing the software project behind schedule.
- The build is integrated with other builds and the entire product (in its current form) is smoke tested daily.
  - The integration approach may be top down or bottom up.

WebApp Testing - I

- The content model for the WebApp is reviewed to uncover errors.
- The interface model is reviewed to ensure that all use cases can be accommodated.
- The design model for the WebApp is reviewed to uncover navigation errors.
- The user interface is tested to uncover errors in presentation and/or navigation mechanics.
- Each functional component is unit tested.

WebApp Testing - II

- Navigation throughout the architecture is tested.
- The WebApp is implemented in a variety of different environmental configurations and is tested for compatibility with each configuration.
- Security tests are conducted in an attempt to exploit vulnerabilities in the WebApp or within its environment.
- Performance tests are conducted.
- The WebApp is tested by a controlled and monitored population of end-users. The results of their interaction with the system are evaluated for content and navigation errors, usability concerns, compatibility concerns, and WebApp reliability and performance.

Other sorts of testing

- Validation testing
  - Focus is on software requirements
- System testing
  - Focus is on integration of sub-systems
- Alpha/Beta testing
  - Focus is on customer usage
  - Alpha testing = done by employees of development organisation, simulates typical use tasks
  - Beta testing = done by releasing to a limited number of real users

Other sorts of testing, cont'd

- Recovery testing
  - forces the software to fail in a variety of ways and verifies that recovery is properly performed
- Security testing
  - verifies that protection mechanisms built into a system will, in fact, protect it from improper penetration
- Stress testing
  - executes a system in a manner that demands resources in abnormal quantity, frequency, or volume
- Performance Testing
  - test the run-time performance of software within the context of an integrated system