Integration and System Testing

1 Integration Testing

- When unit testing has demonstrated a suitable level of correctness for our components, we need to start combining these
- **Big Bang** - Stick all the components together and hope it workd
- **Phased** - Begin testing before all components are ready

1.1 Bottom-up integration

- Aim to complete unit testing for components at the lowest level of hierarchy first
- Test the next level of components, using the lowest ones
- Continue with this to complete system level

This does assume that there is a component hierarchy
Need to create a set of component drivers to test each level by providing the necessary calls
Devising an oracle for this is often relatively tractable

1.1.1 Issues

- Helps identify sources of problems quite well
- Lower level components get tested first and key ones at the top level only get tested later

1.2 Top-down integration

- Involves testing with the key components at the top of the hierarchy
- Since lower level elements may not be ready or tested, can use a stub which emulates the missing component in a simplified manner for each one.
- Testing of components in the middle may need stubs and drivers

1.2.1 Issues

- Writing the stubs and drivers may be quite complex
- Needs the support of an effective test harness to aid configuration, and also collection of test outputs. (call the correct stubs and drivers at the right time)
- Devising an oracle can be quite challenging

1.3 Sandwich integration

- Combine top down and bottom up to work from both ends, reducing the number of stubs needed

2 Continuous builds

- Maintain a single source repository
- Automate the build
- Make the build self testing
- Require everyone to commit every day
- Keep the build fast
- Ensures visibility to all participants

3 System Testing

System testing is much more concerned with conformance to the specification (requirements) than with finding bugs. Precedes and underpins UAT

- Unit testing and integration testing are concerned with whether the coding conforms to the design
- System testing is concerned with whether the design conforms to the requirements

Steps in system testing include:

1. Function testing
2. Performance testing
3. Acceptance testing
4. Installation testing

3.1 Function tests

- Driven by the list of requirements
- Can be documented in a tabular form by listing the requirement and then recording how it has been tested and the outcomes of that test
- Can form a preliminary stage for UAT, ensuring that the system works and that it conforms to the requirements as stated. UAT then assesses whether the stated requirements are the real ones

3.2 Performance tests

Address the non functional issues such as

- Security
- Speed
- Accuracy
- Reliability

The ordering here is important. It is a good principle to get the system working then address these issues.