Integration testing

- Bottom-up
- Top-down
- Big-bang
- Sandwich testing
- Modified top-down
- Modified sandwich
Bottom-up Integration Testing

- Uses Component Drivers

Top-down Integration Testing

- Uses component stubs

Modified Top-down
Big Bang Integration Testing

- Not advised:
  - need many drivers
  - need many stubs
  - hard to isolate faults

Sandwich Testing

Modified Sandwich Testing
Table 8.7: Comparison of integration strategies.

<table>
<thead>
<tr>
<th>Integration</th>
<th>Bottom-up</th>
<th>Top-down</th>
<th>Modified top-down</th>
<th>Big-bang</th>
<th>Sandwich</th>
<th>Modified sandwich</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to first testing program</td>
<td>Early</td>
<td>Early</td>
<td>Early</td>
<td>Late</td>
<td>Early</td>
<td>Early</td>
</tr>
<tr>
<td>Component drivers needed</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Stub needed</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Work parallelism when beginning</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Ability to test one particular path</td>
<td>Easy</td>
<td>Hard</td>
<td>Easy</td>
<td>Easy</td>
<td>Medium</td>
<td>Easy</td>
</tr>
<tr>
<td>Ability to plan and control sequence</td>
<td>Easy</td>
<td>Hard</td>
<td>Hand</td>
<td>Easy</td>
<td>Hard</td>
<td>Hand</td>
</tr>
</tbody>
</table>

Microsoft Synch-and-Stabilize

Traditional vs. OO-Testing

CS 1530 Software Engineering Fall 2004
Test planning

- Establish test objectives
- Design test cases
- Write test cases
- Test test cases
- Execute tests
- Evaluate test results

The Test Plan

- Describes how software will be demonstrated to be correct
  - have to know requirements, specifications, & design
- Plan contents:
  - series of tests (at unit, integration, functional, acceptance & installation level)
  - how tests will be run and criteria when test is complete and satisfactory

Automated testing tools

- Code analysis
  - Static analysis
    - code analyzer
    - structure checker
    - data analyzer
    - sequence checker
  - Dynamic analysis
    - program monitor
- Test execution
  - Capture and replay
  - Stubs and drivers
  - Automated testing environments
- Test case generators
When to stop testing

- Coverage criteria
- Fault seeding
- Confidence in the software

\[
C = \begin{cases} 
1 & \text{if } n > N \\
\frac{S(N-1)}{S(N)} & \text{if } n \leq N 
\end{cases}
\]

System testing process

- Function testing: does the integrated system perform as promised by the requirements specification?
- Performance testing: are the non-functional requirements met?
- Acceptance testing: is the system what the customer expects?
- Installation testing: does the system run at the customer site(s)?

Techniques used in system testing

- Build or spin plan for gradual testing
- Configuration management
  - versions and releases
  - production system vs. development system
  - deltas, separate files and conditional compilation
  - change control
- Regression testing
**Regression Testing**
- Identify faults after changes to system
  - because of new functionality added
  - because of fixes to other faults
- Rerun tests after every change
  - compare output or previous tests to current test
  - if there's a change, check if a fault

**Configuration Management**
- Software revision control systems
  - RCS (revision control system)
    - users have locks
  - CVS (concurrent revision control system)
    - can concurrently edit the same file
    - merges happen at updates (conflicts have to resolved manually)
- Deltas
  - = differences between previous and current version (backward delta) or current and next version (forward delta)
  - save considerable disk space

**Test team**
- Professional testers: organize and run the tests
- Analysts: who created requirements
- System designers: understand the proposed solution
- Configuration management specialists: to help control fixes
- Users: to evaluate issues that arise
Performance tests

- Stress tests
- Volume tests
- Configuration tests
- Compatibility tests
- Regression tests
- Security tests
- Timing tests
- Environmental tests
- Quality tests
- Recovery tests
- Maintenance tests
- Documentation tests
- Human factors (usability) tests

Acceptance tests

- Pilot test: install on experimental basis
- Alpha test: in-house test
- Beta test: customer pilot
- Parallel testing: new system operates in parallel with old system

Test documentation

- Test plan: describes system and plan for exercising all functions and characteristics
- Test specification and evaluation: details each test and defines criteria for evaluating each feature
- Test description: test data and procedures for each test
- Test analysis report: results of each test
INPUT DATA:
Input data are to be provided by the LIST program. The program
generates randomly a list of N words of alphanumeric characters;
each word is of length M. The program is invoked by calling
RUN LIST(N)

Case 1: Use LIST with N=5, M=5
Case 2: Use LIST with N=10, M=5
Case 3: Use LIST with N=15, M=5
Case 4: Use LIST with N=50, M=10
Case 5: Use LIST with N=100, M=10
Case 6: Use LIST with N=150, M=10

INPUT COMMANDS:
The SORT routine is invoked by using the command
RUN SORT(INBUF,OUTBUF) or
RUN SORT(INBUF)

OUTPUT DATA:
If two parameters are used, the sorted list is placed in OUTBUF.
Otherwise, it is placed in INBUF.

SYSTEM MESSAGES:
During the sorting process, the following message is displayed:
"Sorting... please wait..."
Upon completion, SORT displays the following message on the
screen:
"Sorting completed"
To halt or terminate the test before the completion message is
displayed, press CONTROL-C on the keyboard.

Step N:
Step N-1:
Screen will ask for the name of the data file.
Type right after ending: 
* add file
* modify file
* delete file
* modify file
Place cursor next for 'modify file' and press RETURN Key
Step N-2:
Screen will ask for record number. Type '4017'.
Step N-3:
Screen will fill with data fields for record 4017:
Record number: 4017
X: 0042
Y: 0036
Soil type: clay
Percolation: 4 mtrs/hr
Vegetation: kudzu
Canopy height: 25 mtrs
Water table: 12 mtrs
Construct: outhouse
Maintenance code: 3T/4F/9R

Step N-4:
Press function key B: modify.
Entries on screen will be highlighted.
Move cursor to VEGETATION field.
Type 'grass' over 'kudzu' and press RETURN Key.
Step N-5:
Entries on screen will no longer be highlighted.
VEGETATION field should now read 'grass'.
Step N-6:
Press function key C: Return to previous screen.
Step N-7:
Menu will appear, reading:
* delete file
* modify file
* rename file
To verify that the modification has been recorded, place cursor next for 'modify file' and press RETURN Key.
Step N-8:
Screen will ask for record number. Type '4017'.
Step N-9:
Screen will fill with data fields for record 4017:
Record number: 4017
X: 0042
Y: 0036
Soil type: clay
Percolation: 4 mtrs/hr
Vegetation: grass
Canopy height: 25 mtrs
Water table: 12 mtrs
Construct: outhouse
Maintenance code: 3T/4F/9R

Problem report forms
- Location
- Timing
- Symptom
- End result
- Mechanism
- Cause
Testing safety-critical systems

- Design diversity: use different kinds of designs, designers
- Software safety cases: make explicit the ways the software addresses possible problems
  - failure modes and effects analysis
  - hazard and operability studies
- Cleanroom: certifying software with respect to the specification

Table 9.6. Perspectives for safety analysis.

<table>
<thead>
<tr>
<th>Known cause</th>
<th>Unknown case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known effect</td>
<td>Description of system behavior</td>
</tr>
<tr>
<td>Unknown effect</td>
<td>Inductive analysis, including failure modes and effects analysis</td>
</tr>
</tbody>
</table>