Software Testing

Lecture Notes 3 (of 4)

Outline of the Lecture

- Integration Testing
  - Top-down
  - Bottom-up
  - Big-bang
  - Sandwich
- System Testing
- Acceptance Testing
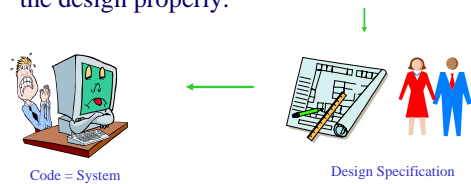- Distribution of Faults in a large Industrial Software System (ISSTA 2002)

Unit & Integration Testing

Objective: to ensure that code implemented the design properly.
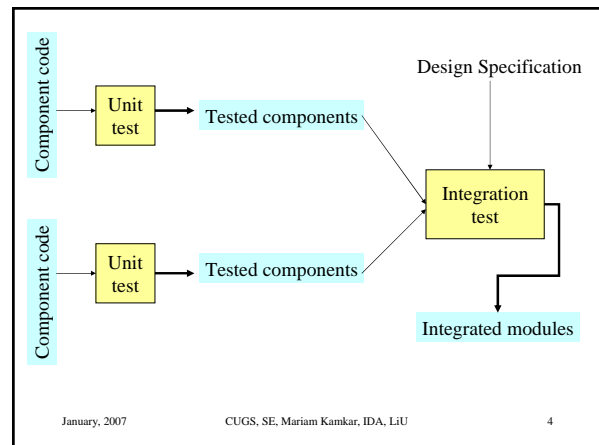
Code = System

Design Specification

Components

Slide 7

Top-down

Stubs for four components

Slide 8

Slide 9

Slide 10

A

B   C   D

E   F   G

Slide 11

Top-down

Slide 12

Top-down

Test A

Test ABCD

Test ABCDG

Test ABCDEFG

2

Modified Top-down

Modified Top-down

Test A

Test B   Test C   Test D

Test A, B, C,

E   F   G

Test A, B, C, D, E, F, G
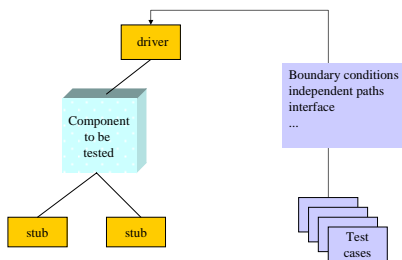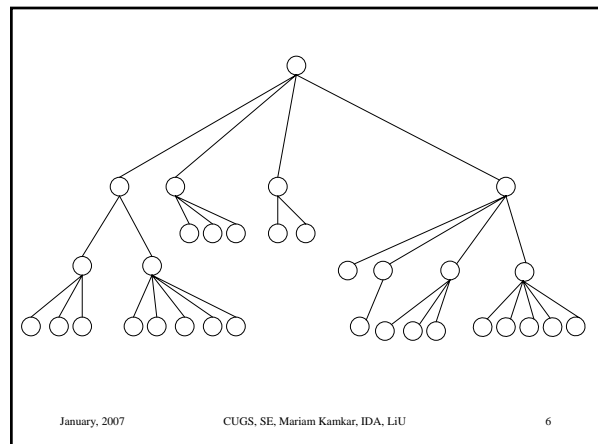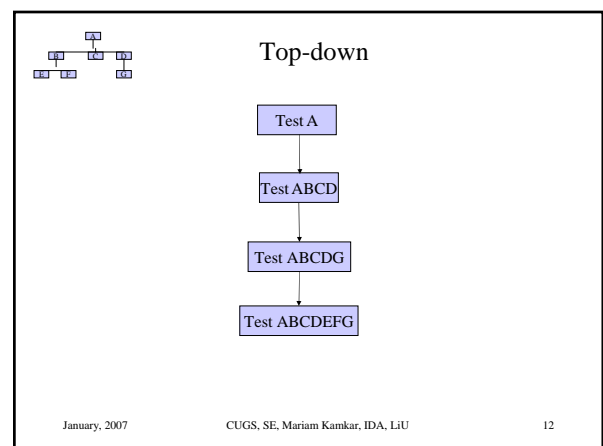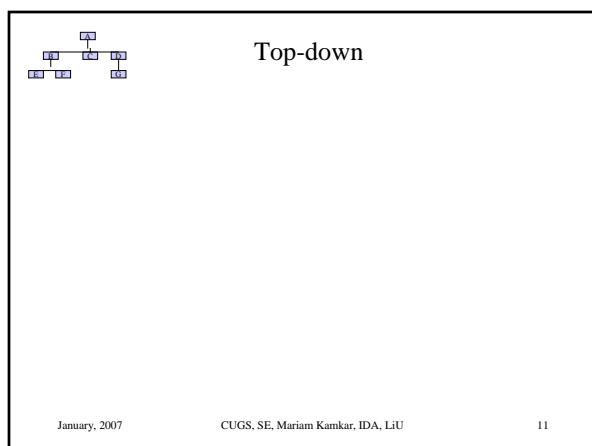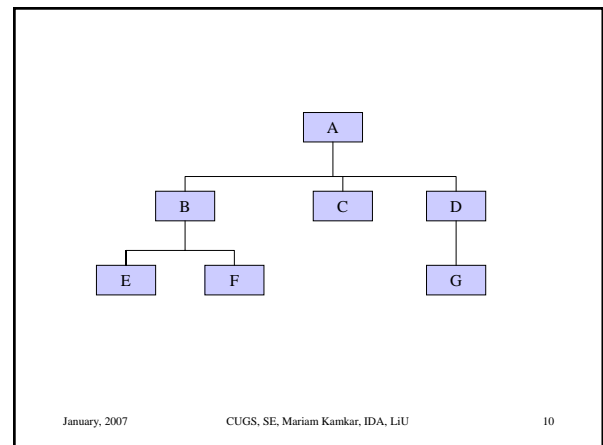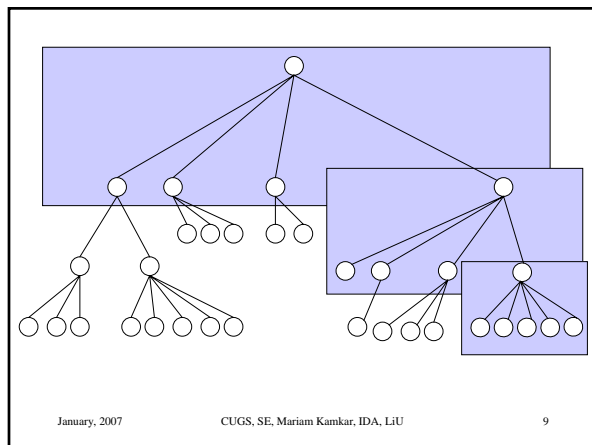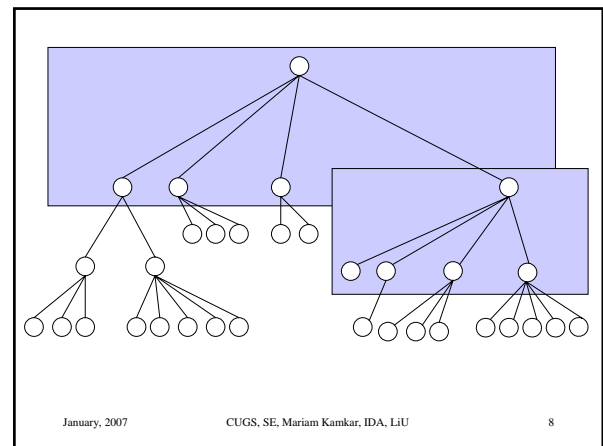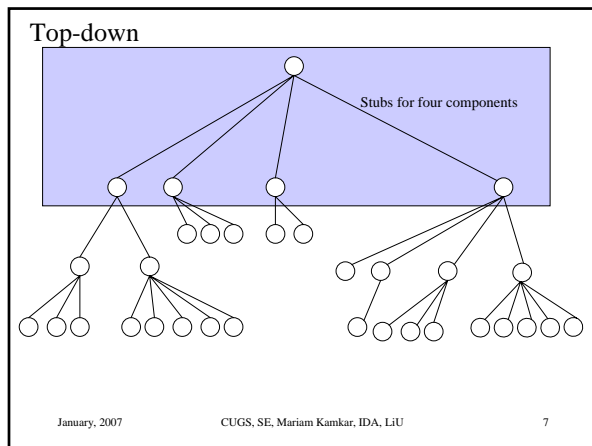
Bottom-up

Driver for five components

A
B C D
E F G

Bottom-up

Bottom-up

Test E   Test F   Tets G

Test BEF   Test C   Tets DG

Tets ABCDEFG

Big-bang

Test A
Test B
Test C
Test D
Test E
Test F
Test G

Test ABCDEFG

A
B C D
E F G

Sandwich

Target level B,C,D

Slide 25 — Sandwich

Sandwich

Target level B,C,D

Test E
Test F
Test B, E, F
Test G
Test D, G
Test A
Test ABCDEFG

Slide 26 — Modified Sandwich

Modified Sandwich

Slide 27 — Modified Sandwich

Modified Sandwich

Test E
Test B
Test B, E, F
Test F
Test D
Test G
Test D, G
Test A
Test C
Test ABCDEFG

Slide 28 — Comparison of Integration Strategies
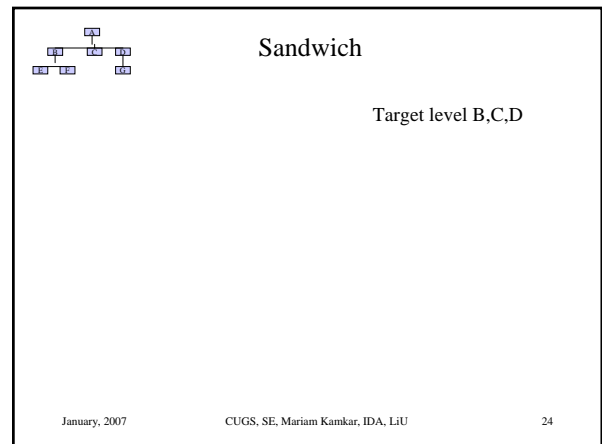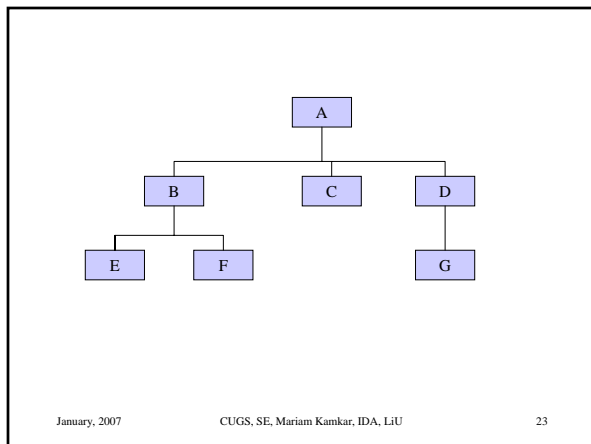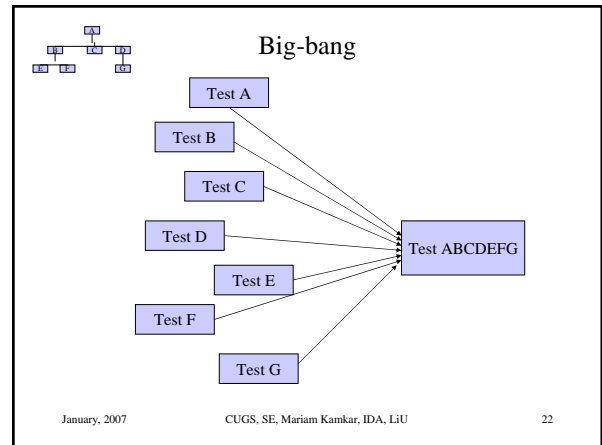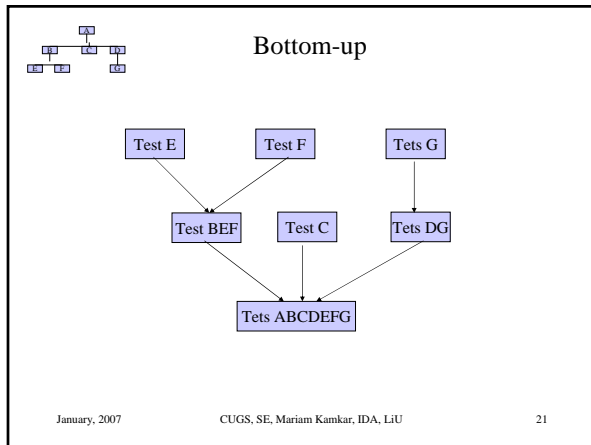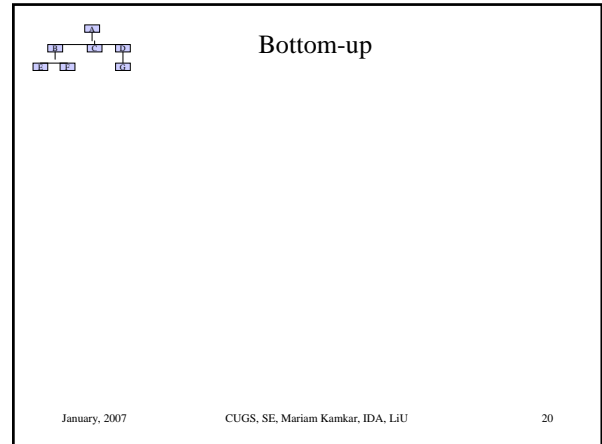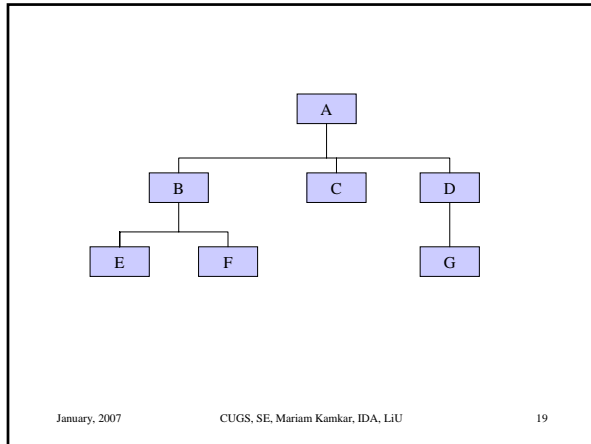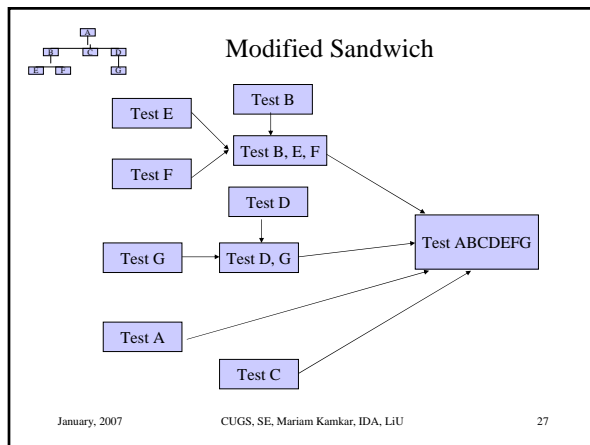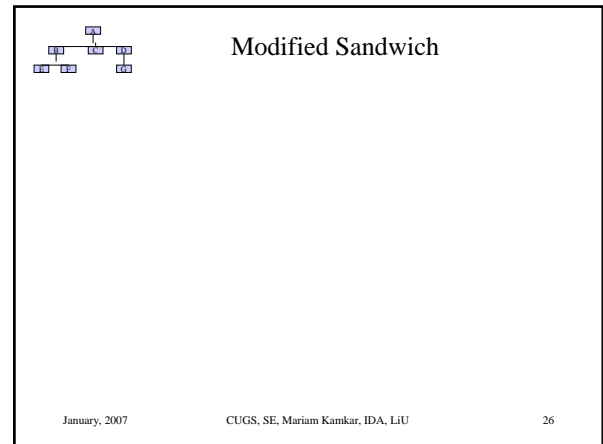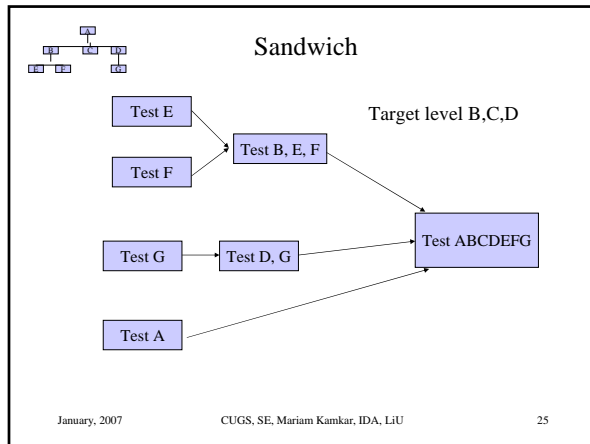
Comparison of Integration Strategies

	Top-down	Modified Top-Down	Bottom-up	Big-bang	Sandwich	Modified Sandwich
Integration	Early	Early	Early	Late	Early	Early
Time to basic working program	Early	Early	Late	Late	Early	Early
Driver needed	No	Yes	Yes	Yes	Yes	Yes
Stubs needed	Yes	Yes	No	Yes	Yes	Yes

Slide 29

- System Testing Steps
  - Function testing / Thread testing
  - Performance testing
  - Acceptance testing
  - Installation testing

- Test Automation
- Termination Problem

Slide 30 — System Testing

System Testing

Objective: to ensure that the system does what the customer wants it to do.

Customer　　　　Developer

Requirements definition
Requirements specification

Functional requirements
Nonfunctional requirements

5
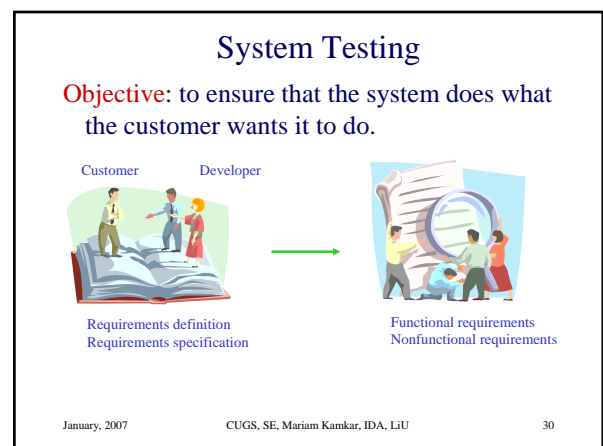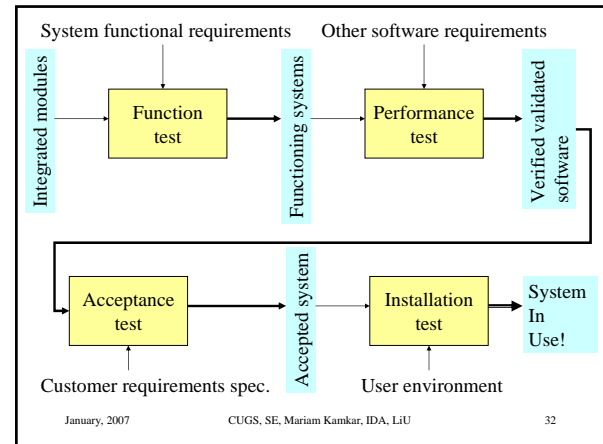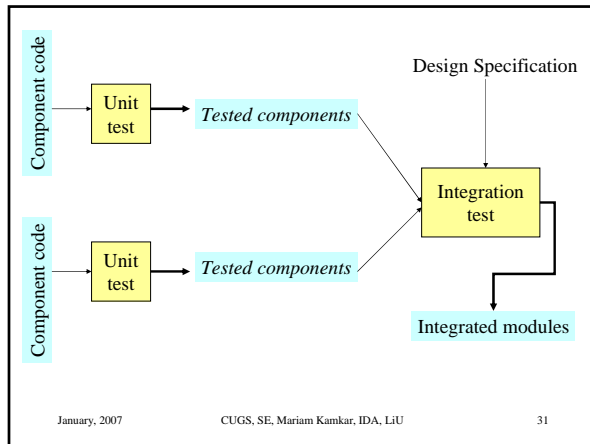
Function testing/Thread testing
(testing one function at a time)
functional requirements

- Threads:
  - A scenario of normal usage
  - A stimulus/response pair
  - Behavior that results from a sequence of system-level input
  - An interleaved sequence of port input and output events
  - A sequence of atomic system functions (ASF)
- ASF: an atomic system function is an action that is observable at the system level in terms of port input and output events

---

A function test checks that the integrated system performs its function as specified in the requirement

- Guidelines
  - use a test team independent of the designers and programmers
  - know the expected actions and output
  - test both valid and invalid input
  - never modify the system just to make testing easier
  - have stopping criteria

Cause-and-Effect-Graph
(test case generation from req.)

- causes: inputs
- effects: outputs and transformations
- causes-and-effect graph:
  - boolean graph reflecting causes and effects relationships
  - is a formal language into which a natural language specification is translated

---

Basic cause-effect graph symbols

Identity: if *a* then *b*

And: if (*a* and *b*) then *c*

Or: if (*a* or *b* or *c*) then *d*

Identity: if (not *a*) then *b*
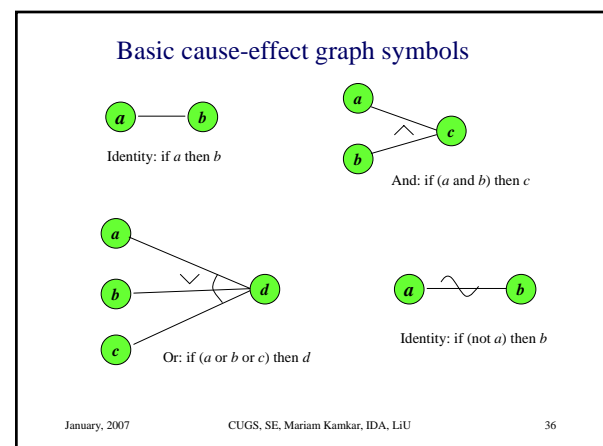
Specification: the character in column 1 must be an "A" or a "B". The character in column 2 must be a digit. In this situation, the file update is made. If the first character is incorrect, message X12 is issued. If the second character is not a digit, message X13 is issued.

Causes	**Effects**
C1: character in column 1 is "A"	E1: update made
C2: character in column 1 is "B"	E2: message X12 is issued
C3: character in column 2 a digit	E3: message X13 is issued

Sample cause-effect graph

Constraint symbols

E cause-constraint: at most one of *a* or *b* can be true

O cause-constraint: one, and only one, of *a* and *b* must be true

I cause-constraint: at least one of *a*, *b* and *c* must always be true

R cause-constraint: for *a* to be true, *b* must be true

M effect-constraint: If effect *a* to is true, Effect *b* is forced to be false

---
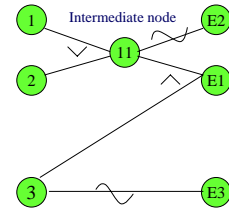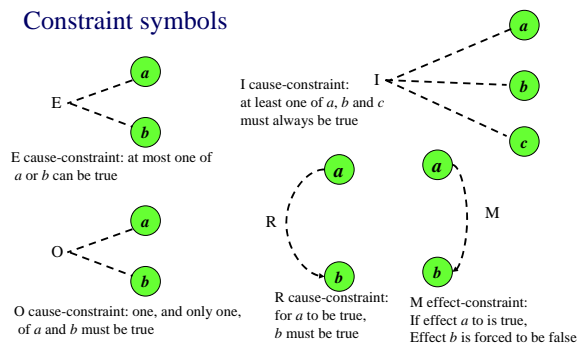
Sample cause-effect graph with "exclusive" constraint
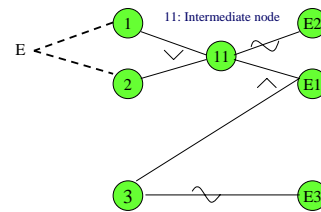
Decision table for cause-and effect graph

	Test 1	Test 2	Test 3	Test 4
Cause 1	1	0	0	X
Cause 2	0	1	0	X
Cause 3	1	1	X	0
Effect E1	1	1	0	0
Effect E2	0	0	1	0
Effect E3	0	0	0	1

Performance Testing
nonfunctional requirements

- Stress tests
- Volume tests
- Configuration tests
- Compatibility tests
- Regression tests
- Security tests
- Timing tests
- Environment tests
- Quality tests
- Recovery tests
- Maintenance tests
- Documentation tests
- Human factors tests / usability tests

Acceptance Testing
customers, users need

- Benchmark test: a set of special test cases

- Pilot test: everyday working
  - Alpha test: at the developer's site, controlled environment
  - Beta test: at one or more customer site.

- Parallel test: new system in parallel with previous one

Installation Testing
users site

Acceptance test at developers site
→ installation test at users site,
otherwise may not be needed!!

Test Automation

- Automating parts of the testing process can provide long-term benefits to organization, such as:
  - reducing the amount of time it takes to execute a suite of tests
  - reducing the tester's involvement in executing tests
  - facilitating regression testing
  - allowing for the simulation of hundreds of users
  - avoiding human mistakes by having tools control repetitive and tedious tasks

- Test automation refers to two key testing activities:
  - Executing the tests
  - Evaluating the output

Automated Testing Tools

- Code Analysis tools
  - Static, Dynamic

- Test execution tools
  - Capture-and-Replay
  - Stubs & Drivers
  - Comparators

- Test case generator

Termination Problem
How decide when to stop testing

- The main problem for managers!

- Termination takes place when
  - resources (time & budget) are over
  - found the seeded faults
  - some coverage is reached