Testing III

Week 16

Agenda (Lecture)

- White box testing
  - Condition coverage
  - Loop coverage
  - Path coverage

Agenda (Lab)

- Implementation
- Review of SRS/SDD documents
- Submit a weekly project progress report at the end of this week lab session

Condition Coverage

- All possible values of the predicates of compound conditions are exercised at least once
- Lazy evaluation

Test Cases for Condition Coverage

Node	Source Line
A	read a, b, c
B	type = “scalene”
C	if (a==b || b==c || a==c)
D	type = “isosceles”
E	if (a == b && b == c)
F	type = “equilateral”
G	if (a >= b + c || b > a + c || c > = a + b)
H	type = “not a triangle”
I	if (a <= 0 || b <= 0 || c <=0)
J	type = “bad inputs”
K	print type

Test Cases for Condition Coverage

if (a==b || b==c || a==c)

Combination	Possible Test Case	Branch
TXX	3,3,4	ABC-D
FTX	4,3,3	ABC-D
FFT	3,4,3	ABC-D
FFF	3,4,5	ABC-D

if (a==b && b==c)

Combination	Possible Test Case	Branch
TT	3,3,3	E-F
TF	3,3,4	E-G
FX	4,3,3	E-G

if (a >= b + c || b >= a + c || c >= a + b)

Combination	Possible Test Case	Branch
TXX	8,4,3	G-H
FTX	4,8,3	G-H
FFT	4,3,8	G-H
FFF	3,3,3	G-I

if (a <= 0 || b <= 0 || c <= 0))

Combination	Possible Test Case	Branch
TXX	0,4,5	I-J
FTX	4,-2,-2	I-J
FFT	5,4,-3	I-J
FFF	3,3,3	I-K

Condition Coverage

- Combination of edge coverage and more detailed conditions
  - Edge: Every edge of the control flow graph is executed at least once
  - Condition: All values of conditions are exercised at least once
- More precise than branch coverage

Path Coverage

- Select test cases such that **every path** in the graph is visited
- Finer than all previous criteria and an **ideal** criterion
- However, number of paths is exponential in decisions; almost infinite paths needs in arbitrary loops

Test Cases for Path Coverage

Node	Source Line
A	read a, b, c
B	type = “scalene”
C	if (a==b || b==c || a==c)
D	type = “isosceles”
E	if (a == b && b == c)
F	type = “equilateral”
G	if (a >= b + c || b > a + c || c > = a + b)
H	type = “not a triangle”
I	if (a <= 0 || b <= 0 || c <=0)
J	type = “bad inputs”
K	print type

Path Coverage

Path	T/F	Test Case	Output
ABCEGIK	FFFF	3,4,5	Scalene
ABCEGHIK	FFTF	3,4,8	Not a triangle
ABCEGHIJK	FFTT	0,5,6	Bad inputs
ABCDEGIK	TFFF	5,8,5	Isosceles
ABCDEGHIK	TFTF	3,8,3	Not a triangle
ABCDEGHIJK	TFTT	0,4,0	Bad inputs
ABCDEFGIK	TTFF	3,3,3	Equilateral
ABCDEFGHIJK	TTTT	0,0,0	Bad inputs

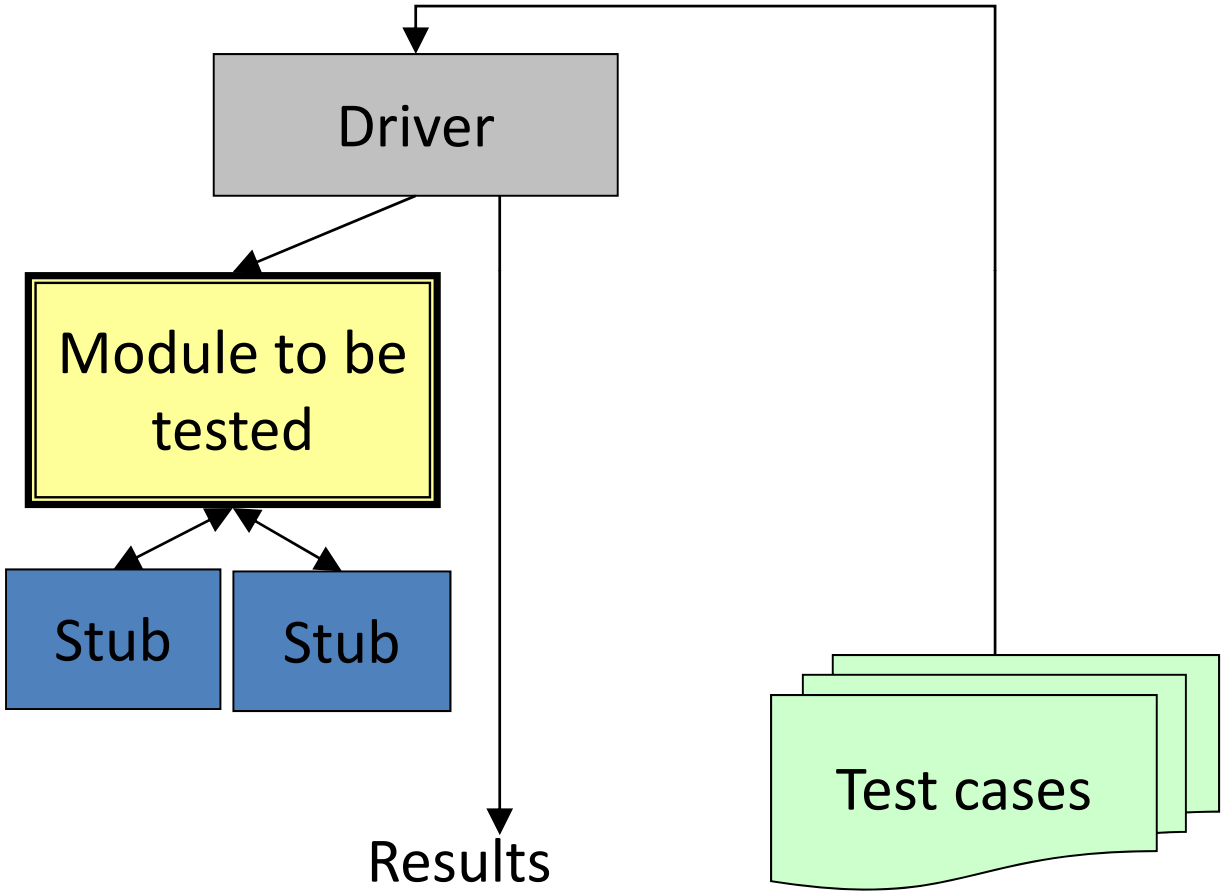
Testing Levels – Custom Software

- Unit testing
- Integration testing
- Product testing
- Acceptance testing
- Regression Testing

Testing Levels – COTS

- Unit testing
- Integration testing
- Product testing
- Alpha testing
- Beta testing
- Regression Testing

Unit Testing

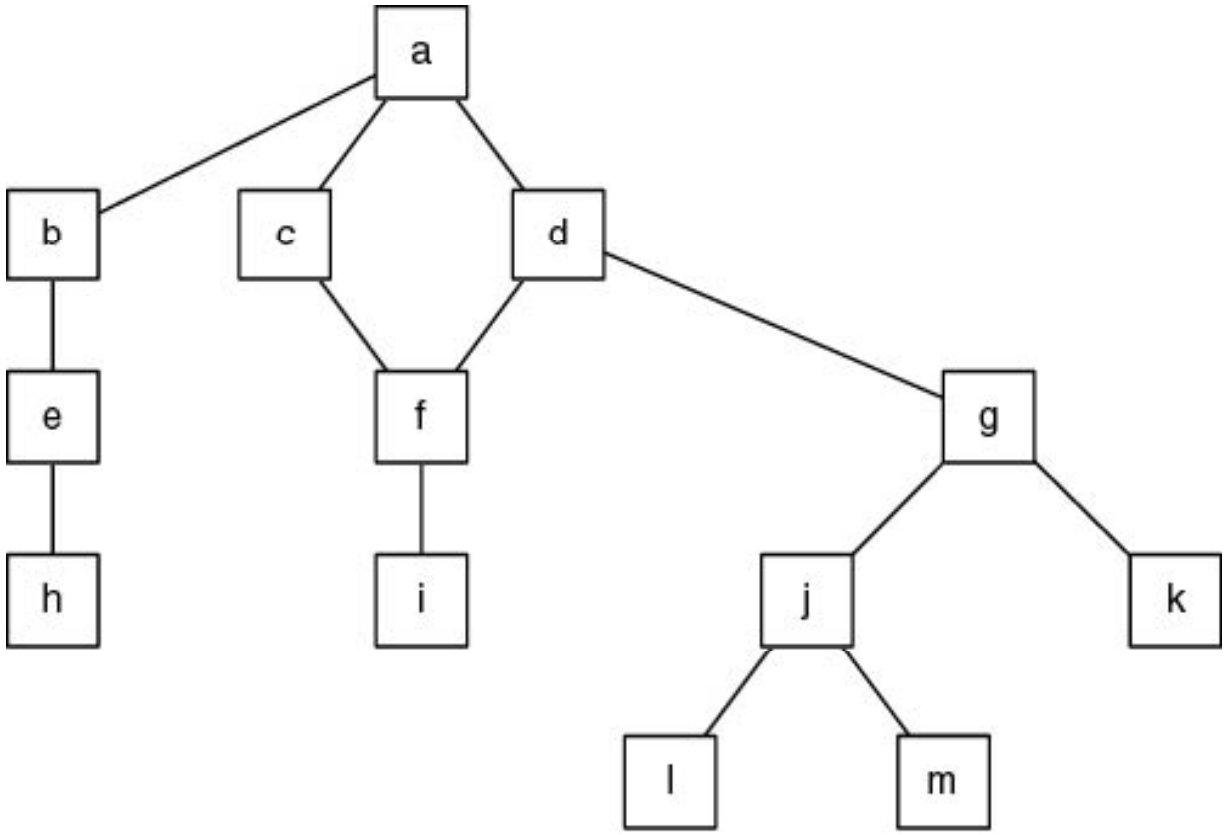


Unit Test Procedures

- Develop source code
- Review source code
- Verify the code with design
- Design test cases
- Develop driver and stub

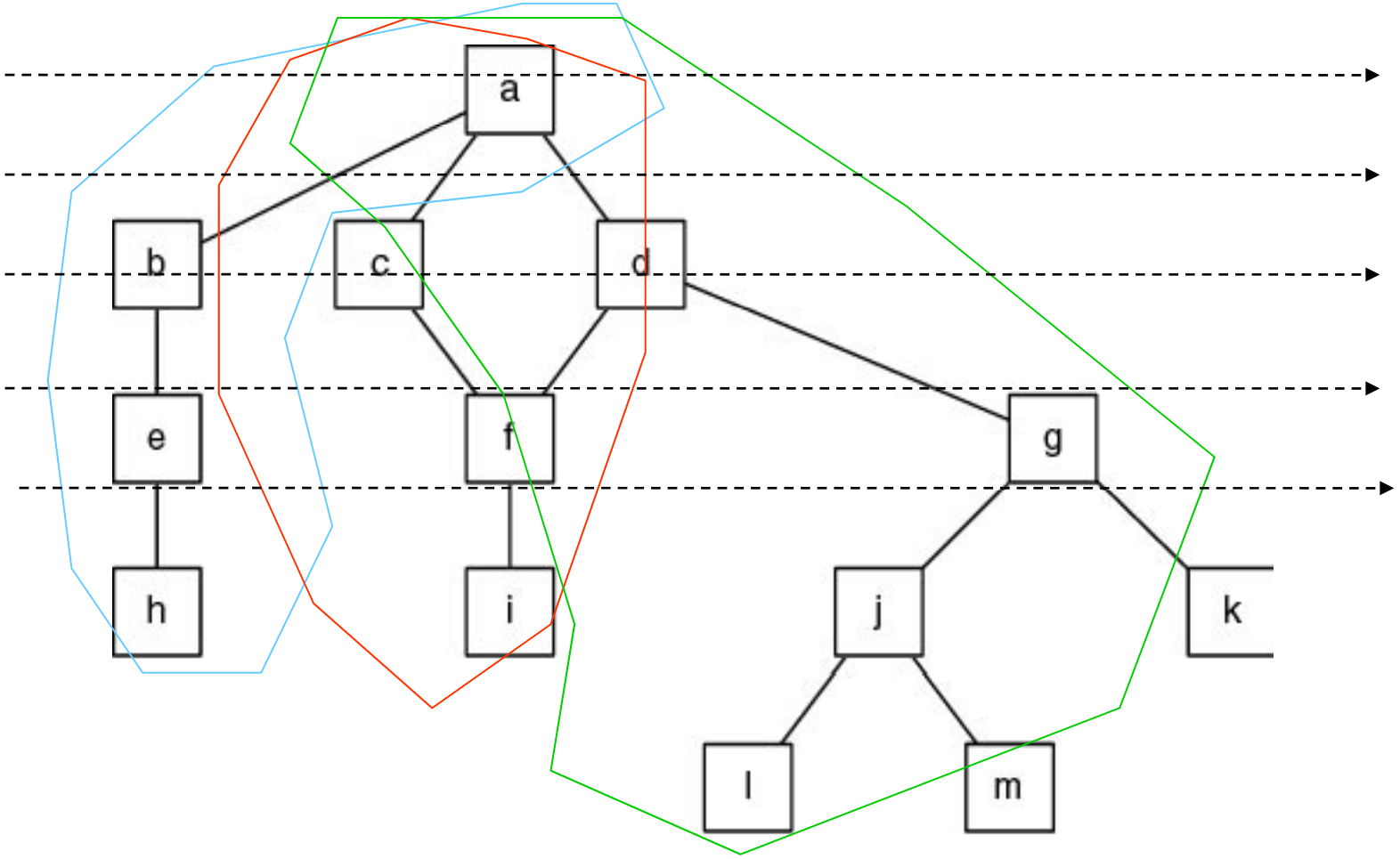
Integration Testing

- Top-down implementation, integration, and testing
- Bottom-up implementation, integration, and testing
- Sandwich implementation, integration, and testing



Typical interconnection diagram

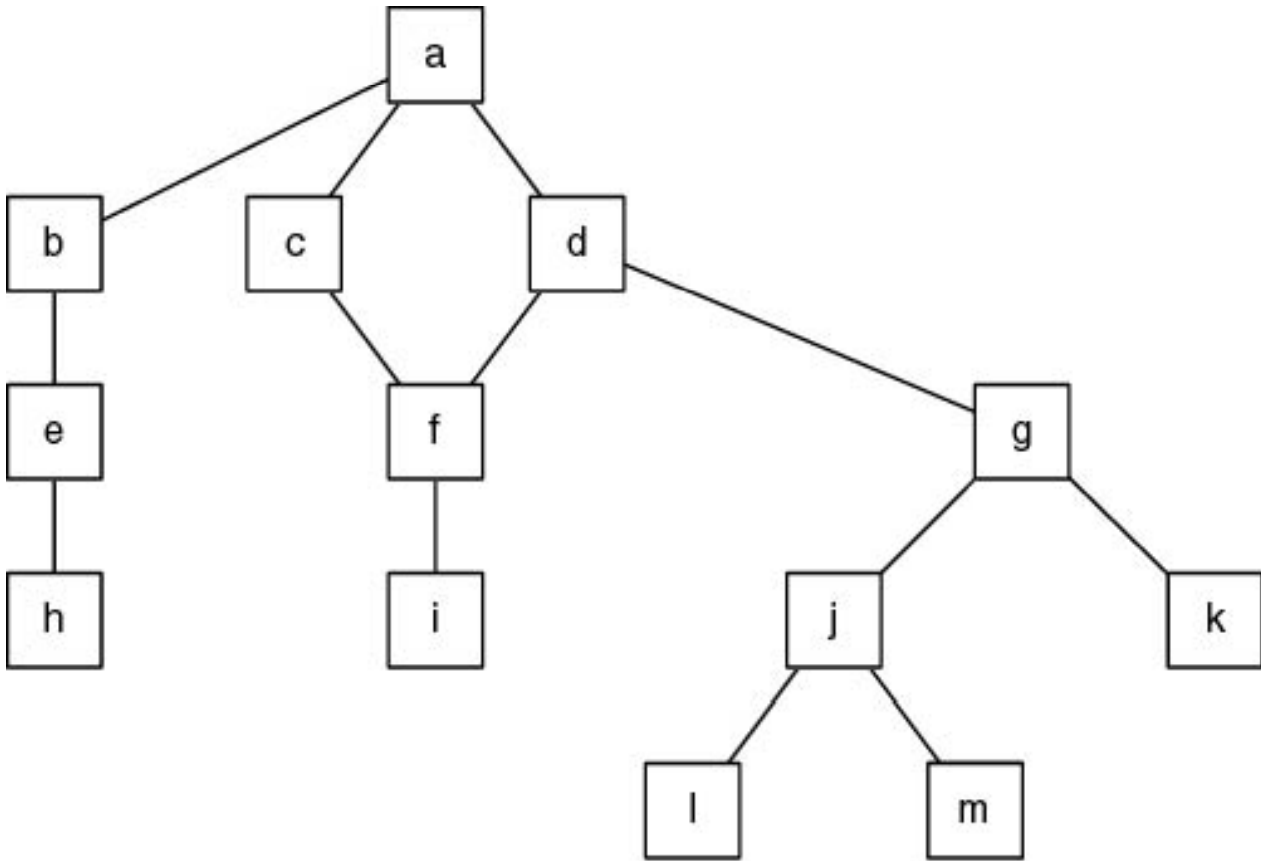
Top-Down



Top-Down (cont'd)

- Advantages
  - Fault isolation
  - Stubs not wasted
  - Major design flaws show up early
- Disadvantages
  - Reusable modules are not properly tested
  - Lower level (operational) modules are not tested frequently

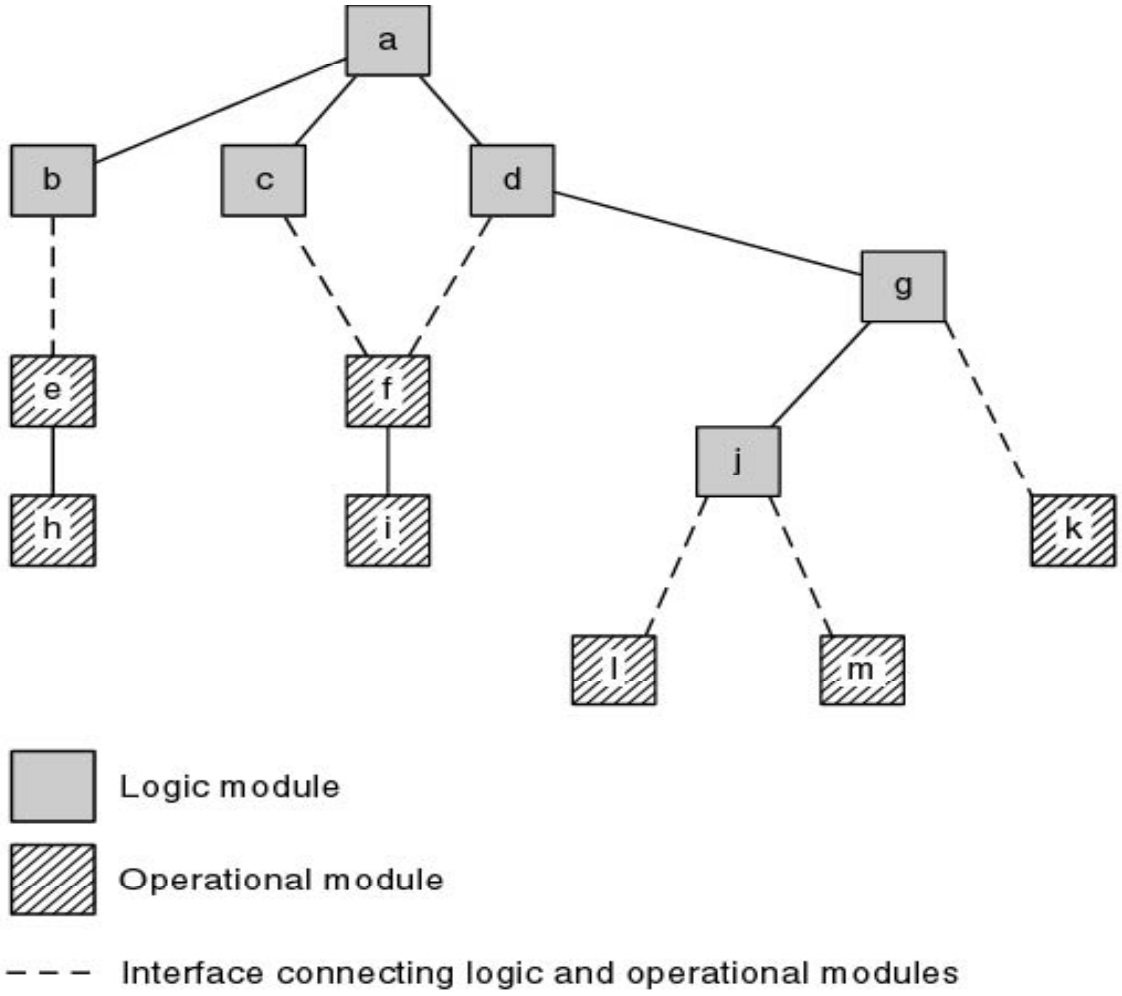
Bottom-Up



Bottom-Up (cont'd)

- Advantages
  - Operational modules thoroughly tested
  - Operational modules are tested with drivers, not by fault shielding, defensively programmed calling modules
  - Fault isolation
- Disadvantages
  - Major design faults are detected late in the integration phase

Sandwich Implementation and Integration



Sandwich Implementation and Integration (cont'd)

- Advantages
  - Major design faults are caught early
  - Operational modules are thoroughly tested
  - They may be reused with confidence
  - There is fault isolation at all times

Summary of Integration Approaches

Approach	Strengths	Weaknesses
Implementation then integration	—	No fault isolation Major design faults show up late
Top-down implementation and integration	Fault isolation Major design faults show up early	Potentially reusable modules are not adequately tested
Bottom-up implementation and integration	Fault isolation Potentially reusable modules are adequately tested	Major design faults show up late
Sandwich implementation and integration	Fault isolation Major design faults show up early Potentially reusable modules are adequately tested	—

Product Testing

- Validate all functional requirements
- Check all non-functional constraints
- Review all documentation to be handed over to the client

Acceptance Testing

- Test software with the client's environment and real data
- Performed by the SQA team in the presence of client representatives, or an independent SQA team hired by the client

Alpha Testing

- Simulated or actual operational testing by potential users/customers or an independent test team at the developers' site

Beta Testing

- Released a limited number of users outside of the company
- Sometimes, beta versions are made available to the public

Regression Testing

- Post-delivery maintenance
- Each time a module is added, the software changes
  - These changes may cause problems with functions that previously worked flawlessly
- Regression testing is …
  - The execution of tests that have already been conducted to ensure that changes do not create unintended side effects

Current and Emerging Software Engineering Technologies

- Software factory
- Component based software engineering
- Aspect oriented program
- Application frameworks
- Web engineering
- Open source software engineering
- Mobile software engineering
- API (application programming interface)
- Library or toolkits
- Design patterns
- Architecture patterns
- Model-view-controller (MVC) architecture pattern
- Service-oriented architecture
- Software product lines