

Software Testing

Dagmawi Lemma
courses.dag@gmail.com

Software Testing
Dagmawi Lemma - 2020

1

Testing in the SDLC

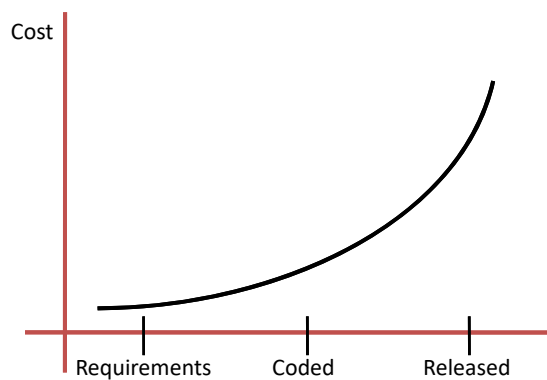
Software Testing

Software Testing
Dagmawi Lemma - 2020

120

Testing in SDLC

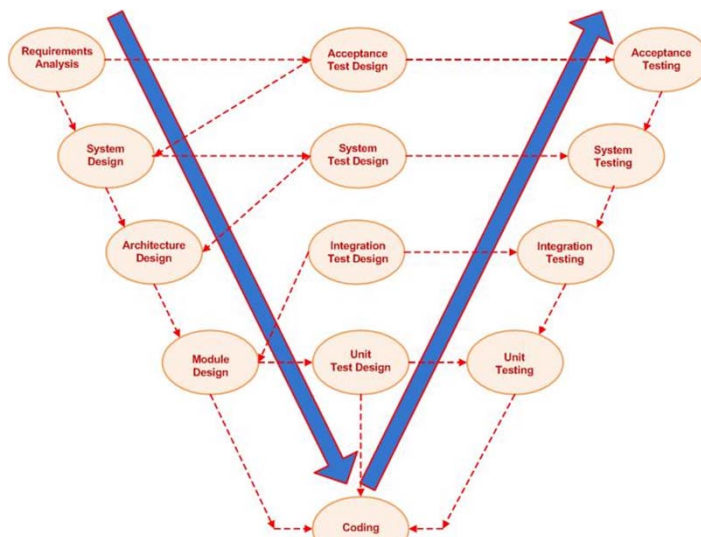
- Cost of testing escalate as we move along the SDLC phases



Software Testing
Dagmawi Lemma - 2020

121

The V-model



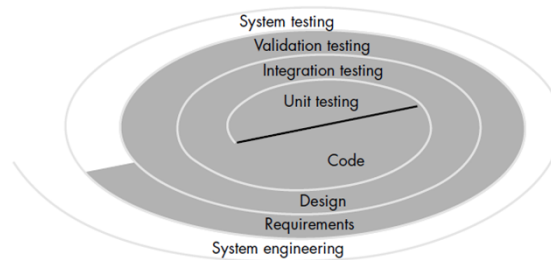
Software Testing
Dagmawi Lemma - 2020

122

Testing Classification : Level of Testing

Level of Testing

- Testing can be classified by the its level taking place in the development process
- Testing occurs at various stage of the software development process – each level with different objectives

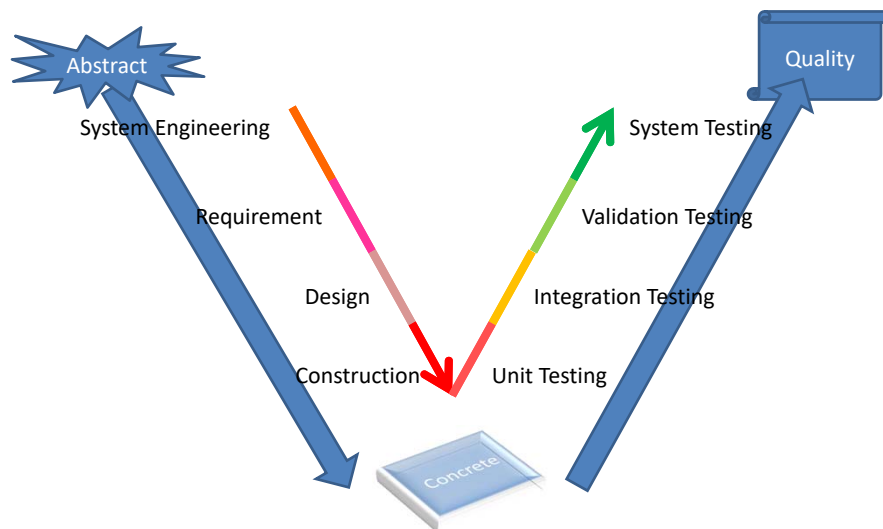


Software Testing
Dagmawi Lemma - 2020

123

Testing Classification : Level of Testing

Testing Strategies



Software Testing
Dagmawi Lemma - 2020

124

Testing Classification : Level of Testing

Unit Testing

- The traditional approach is to write unit test cases after the code is written
- Test-Driven Development on the other hand is an approach that encourages writing unit test cases before the code is written
 - Writing test cases first forces you to think about the requirements and design before writing the code, which leads to a better design
 - Since your test cases will fail at first, they help you know when the code is done (i.e., when the tests work, you're done)
 - Writing test cases before the code doesn't take any more effort than writing them after the code
 - Writing test cases first will help you detect and remove bugs sooner

Software Testing
Dagmawi Lemma - 2020

125

Testing Classification : Level of Testing

Unit Testing

- Focuses verification effort on the smallest unit of software design
 - Class, module, component, ...
- White-box oriented
 - However if the strategy focus on the behavioral or functional aspect of the component, it can be Black-Box
- Can be conducted in parallel for multiple components
- Tests that occur as part of unit
 - Interface
 - Local data structures
 - Boundary conditions
 - Independent paths
 - Error handling paths

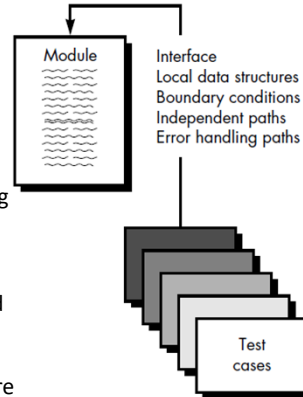
Software Testing
Dagmawi Lemma - 2020

126

Testing Classification : Level of Testing

Unit Testing

- Interface
 - The module (test component) interface is tested to ensure that information properly flows into and out
- Local data structure
 - The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution
- Boundary conditions
 - Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing
- Independent paths
 - Through the control structure are exercised to ensure that all statements in a module have been executed at least once
- Error handling paths
 - Error handling paths are tested

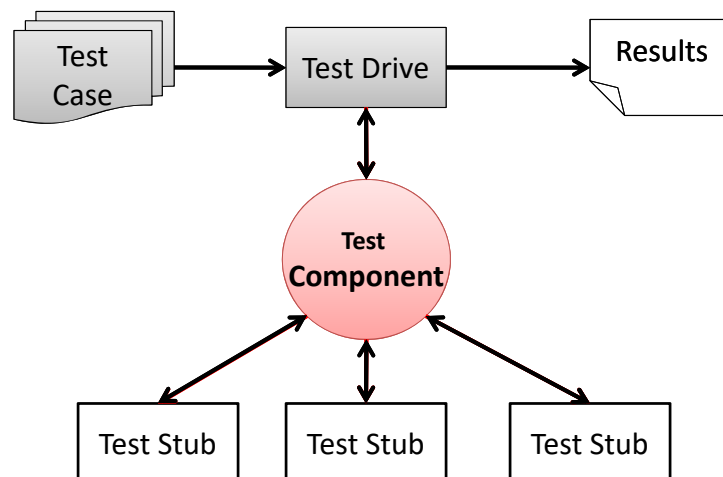


Software Testing
Dagmawi Lemma - 2020

127

Testing Classification : Level of Testing

Unit Testing



Software Testing
Dagmawi Lemma - 2020

128

Testing Classification : Level of Testing

Unit Testing

- Unit testing is normally considered as an adjunct to the coding step
- driver and/or stub software must be developed for each unit test
 - In most applications a driver is nothing more than a "main program" that accepts test case data, passes such data to the test component
 - Stubs serve to replace modules that are subordinate (called by) the test component
- In some cases, complete testing can be postponed until the integration test step due to high overhead that may occur due to drivers and stub

Software Testing
Dagmawi Lemma - 2020

129

Testing Classification : Level of Testing

Integration Testing

- Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing
- Why do we need it?
 - Data can be lost across an interface
 - one module can have an unintended, adverse affect on another;
 - Sub-functions, when combined, may not produce the desired major function;
 - Global data structures can present problems
 - ...

Software Testing
Dagmawi Lemma - 2020

130

Testing Classification : Level of Testing

Integration Testing

- non-incremental integration
 - “big bang” approach
 - components are combined in advance
 - entire program is tested as a whole
- Incremental integration
 - Add a piece, retest the system, Add a piece, retest the system, ...
 - If it breaks, you know what caused the problem (i.e., the last piece you added)
- Integration Testing is testing that's done during integration to ensure that the system continues to work each time a new piece is added

Software Testing
Dagmawi Lemma - 2020

131

Testing Classification : Level of Testing

Integration Testing

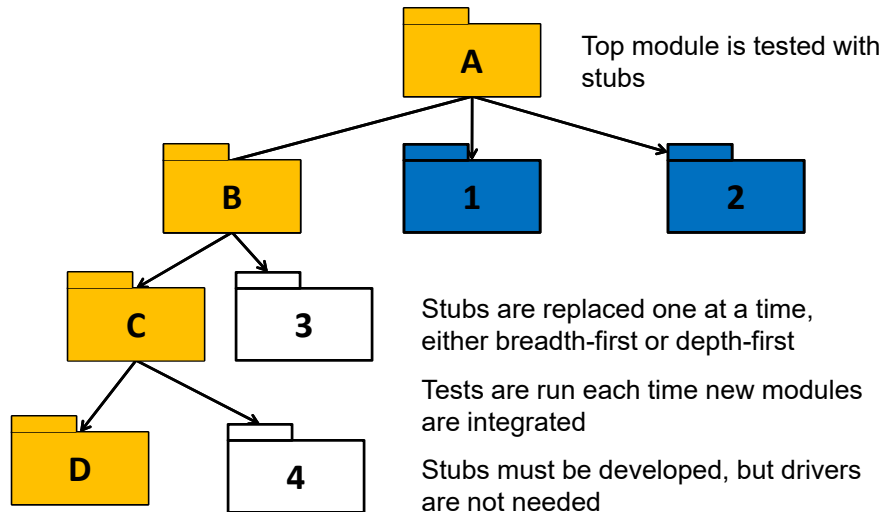
- Incremental integration strategies
 - Top-down Integration
 - Bottom-up Integration
 - Regression Testing
 - Smoke Testing

Software Testing
Dagmawi Lemma - 2020

132

Testing Classification : Level of Testing

Top-Down Integration Testing



Software Testing
Dagmawi Lemma - 2020

133

Testing Classification : Level of Testing

Top-Down Integration Testing

- Incremental approach to construction of program structure
- **Depth-first integration** would integrate all components on a major control path of the structure
 - Selection of a major path is somewhat arbitrary and depends on application-specific characteristics
- **Breadth-first integration** incorporates all components directly subordinate at each level, moving across the structure horizontally

Software Testing
Dagmawi Lemma - 2020

134

Testing Classification : Level of Testing

Top-Down Integration Testing

- The integration process
 1. The main control module is used as a test driver and stubs are substituted for all components directly subordinate to the main control module.
 2. Depending on the integration approach selected (i.e., depth or breadth first), subordinate stubs are replaced one at a time with actual components.
 3. Tests are conducted as each component is integrated.
 4. On completion of each set of tests, another stub is replaced with the real component.
 5. Regression testing (we will see it in the coming slide) may be conducted to ensure that new errors have not been introduced.

Software Testing
Dagmawi Lemma - 2020

135

Testing Classification : Level of Testing

Top-Down Integration Testing

- Benefits
 - verifies major control or decision points early in the test process
 - well-factored program structure, decision making occurs at upper levels in the hierarchy and is therefore encountered first
 - If major control problems do exist, early recognition is essential
 - Early demonstration of functional capability is a confidence builder for both the developer and the customer

Software Testing
Dagmawi Lemma - 2020

136

Testing Classification : Level of Testing

Top-Down Integration Testing

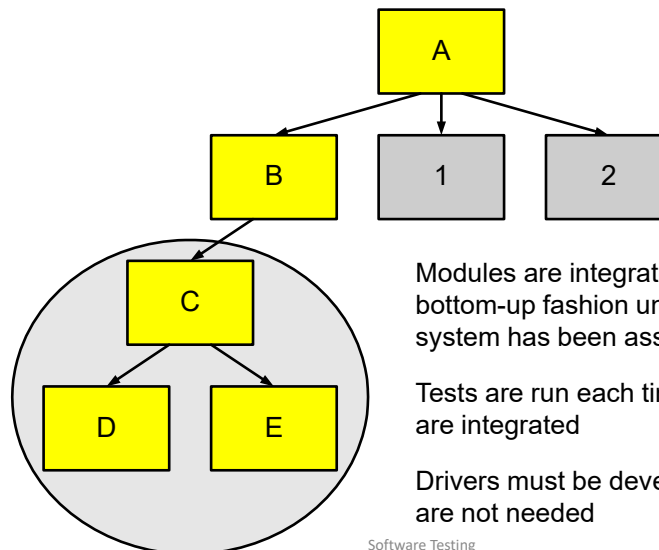
- Challenges
 - relatively uncomplicated (at least ideal)
 - in practice, logistical problems can arise.
 - What if processing at low levels in the hierarchy is required to adequately test upper levels?
 - The tester is left with three choices:
 1. Delay many tests until stubs are replaced with actual modules,
 2. Develop stubs that perform limited functions that simulate the actual module,
 3. Integrate the software from the bottom of the hierarchy upward (this takes us to bottom-up)

Software Testing
Dagmawi Lemma - 2020

137

Testing Classification : Level of Testing

Bottom-Up Integration Testing



Modules are integrated in a bottom-up fashion until the entire system has been assembled

Tests are run each time new modules are integrated

Drivers must be developed, but stubs are not needed

Software Testing
Dagmawi Lemma - 2020

138

Testing Classification : Level of Testing

Bottom-Up Integration Testing

- Begins construction and testing with atomic modules
- Need for stubs is eliminated
- Drivers are required

Software Testing
Dagmawi Lemma - 2020

139

Testing Classification : Level of Testing

Bottom-Up Integration Testing

- The integration process
 1. Low-level components are combined into clusters (sometimes called builds) that perform a specific software sub-function
 2. A driver (a control program for testing) is written to coordinate test case input and output
 3. The cluster is tested
 4. Drivers are removed and clusters are combined moving upward in the program structure

Software Testing
Dagmawi Lemma - 2020

140

Testing Classification : Level of Testing

Sandwich Integration Testing

- Combination of top-down and bottom-up integration
- Integrate from the top and from the bottom as it makes sense, and meet somewhere in the middle

Software Testing
Dagmawi Lemma - 2020

141

Testing Classification : Level of Testing

Unit Testing or Integration Testing?

- Sometimes people who claim to be doing unit testing are actually doing bottom-up integration testing
- How do you tell the difference?
 - Unit testing uses stubs to isolate the module under test from its dependencies
 - Bottom-up integration testing does not (i.e., modules call their “real” dependencies)

Software Testing
Dagmawi Lemma - 2020

142

Testing Classification : Level of Testing

Regression Testing

- Each stage of integration takes the software into different level of version
 - new data flow paths are established,
 - new I/O may occur, and
 - new control logic is invoked
- These changes may cause problems with functions that previously worked flawlessly

Software Testing
Dagmawi Lemma - 2020

143

Testing Classification : Level of Testing

Regression Testing

- **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
- Three different classes of test cases:
 - A representative sample of tests that will exercise all software functions.
 - Additional tests that focus on software functions that are likely to be affected by the change.
 - Tests that focus on the software components that have been changed.

Software Testing
Dagmawi Lemma - 2020

144

Testing Classification : Level of Testing

Smoke Testing

- The smoke test should exercise the entire system from end to end
- The smoke testing approach encompasses the following activities:
 1. 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
 2. 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 errors that have the highest likelihood of throwing the software project behind schedule.
 3. The build is integrated with other builds and the entire product (in its current form) is smoke tested daily

Software Testing
Dagmawi Lemma - 2020

145

Testing Classification : Level of Testing

Validation Testing

- Reasonable expectations are defined in the *Software Requirements Specification*
 - *But is it valid?*
 - *ARE WE BUILDING THE CORRECT PRODUCT?*
- A series of black-box tests required to demonstrate software validation
 - conformity with requirements

Software Testing
Dagmawi Lemma - 2020

146

Testing Classification : Level of Testing

Validation Testing

- a test plan: outlines the classes of tests to be conducted
- a test procedure: defines specific test cases that will be used to demonstrate conformity with requirements
- ensuring
 - all functional requirements are satisfied
 - all behavioral characteristics are achieved
 - all performance requirements are attained
 - documentation is correct
 - design goals are met

Software Testing
Dagmawi Lemma - 2020

147

Testing Classification : Level of Testing

Validation Testing

- After each validation test case has been conducted, one of two possible conditions exist
 - The function or performance characteristics conform to specification and are accepted
 - OR
 - A deviation from specification is uncovered and a deficiency list is created

Software Testing
Dagmawi Lemma - 2020

148

Validation Testing

- Configuration Review
 - An important element of the validation process
 - The intent of the review is to ensure that all elements of the software configuration have been properly developed, are cataloged, and have the necessary detail to bolster the support phase of the software lifecycle
 - complements the formal technical review by assessing a configuration object for characteristics that are generally not considered during review

Validation Testing

- Configuration Review asks and answers the following questions:
 1. Has the change specified in the ECO (engineering change order) been made? Have any additional modifications been incorporated?
 2. Has a **formal technical review** been conducted to assess technical correctness?
 3. Has the software process been followed and have software engineering standards been properly applied?

Testing Classification : Level of Testing

Validation Testing

- Configuration Review asks and answers the following questions: *(continued)*
 4. Has the change been "highlighted" in the SCI (*SCI is a software entity that has been established as a configuration item*)? Have the change date and change author been specified? Do the attributes of the configuration object reflect the change?
 5. Have SCM (Software Configuration Management) procedures for noting the change, recording it, and reporting it been followed?
 6. Have all related SCIs been properly updated?

Software Testing
Dagmawi Lemma - 2020

151

Testing Classification : Level of Testing

Validation Testing

- Can you foresee how the customer will really use a program?
 - Misuse of instruction
 - Strange combination of input
 - Misunderstanding of output
 - ...

Software Testing
Dagmawi Lemma - 2020

152

Testing Classification : Level of Testing

Acceptance Testing

- For custom software – acceptance testing
 - i.e. customer validate all requirements
 - Testing is conducted by the end-user rather than software engineers
- acceptance testing can be conducted over a period of weeks or months, thereby uncovering cumulative errors that might degrade the system over time

Software Testing
Dagmawi Lemma - 2020

153

Testing Classification : Level of Testing

Acceptance Testing

- Alpha and Beta Testing
 - formal acceptance testing is impractical to apply for software developed to be used by many customers (e.g. commercial off-the-shelf software)
 - alpha and beta testing can be used to uncover errors that only the end-user seems able to find
 - alpha test is conducted at the developer's site by a customer
 - beta test is conducted at one or more customer sites by the end-user

Software Testing
Dagmawi Lemma - 2020

154

Testing Classification : Level of Testing

System Testing

- Software is only one element of a larger computer-based system
- **System integration** and **system validation** tests are conducted; fall outside the scope and are not conducted solely by software engineers
 - the **software design and testing** can impact the success of software integration

Software Testing
Dagmawi Lemma - 2020

155

Testing Classification : Level of Testing

System Testing

- System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system
- Each test has a different purpose
 - however, each works to verify that system elements have been properly integrated and performs allocated functions
 - Recovery Testing
 - Security Testing
 - Stress Testing
 - Performance Testing

Software Testing
Dagmawi Lemma - 2020

156

Testing Classification : Level of Testing

System Testing

- Recovery Testing
 - a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed
 - Automated Recovery
 - Re-initialization, check-pointing mechanisms, data recovery, and restart are evaluated for correctness
 - Manual Recovery
 - The mean-time-to-repair (MTTR) is evaluated to determine whether it is within acceptable limits
 - How long would it take to revert the system to its correct state?

Software Testing
Dagmawi Lemma - 2020

157

Testing Classification : Level of Testing

System Testing

- Security Testing
 - verifies that protection mechanisms built into a system will, in fact, protect it from improper penetration
 - How about social engineering?
 - It is a system, so do we have safe procedures, policies, guidelines, training, training, training ...
 - Can we trace back security breaches if it occurs?

Software Testing
Dagmawi Lemma - 2020

158

Testing Classification : Level of Testing

System Testing

- Stress testing
 - A system test that executes a system in a manner that demands resources in abnormal quantity, frequency, or volume.
 - For example,
 - interrupts occurring with high frequency
 - special tests may be designed that generate ten interrupts per second, when one or two is the average rate,
 - input data rates may be increased by an order of magnitude to determine how input functions will respond,
 - test cases that require maximum memory or other resources are executed,
 - test cases that may cause thrashing in a virtual operating system are designed,
 - test cases that may cause excessive hunting for disk-resident data are created.
 - Essentially, the tester attempts to break the program.

Software Testing
Dagmawi Lemma - 2020

159

Testing Classification : Level of Testing

System Testing

- Performance Testing
 - designed to test the run-time performance of software within the context of an integrated system (i.e. for system testing)
 - but occurs throughout all steps in the testing process
 - may couple with stress testing
 - often necessary to measure resource utilization

Software Testing
Dagmawi Lemma - 2020

160

Object Oriented Testing

- OOA testing is not as such different from conventional testing
 - strives to find error
 - test drives and stubs are still useful
 - System testing focus on the requirements (as in conventional)
- OOA testing strategies
 - Focus is given to classes (obviously)
 - Unit testing: class level (methods)
 - Integration testing: communication between classes

Software Testing
Dagmawi Lemma - 2020

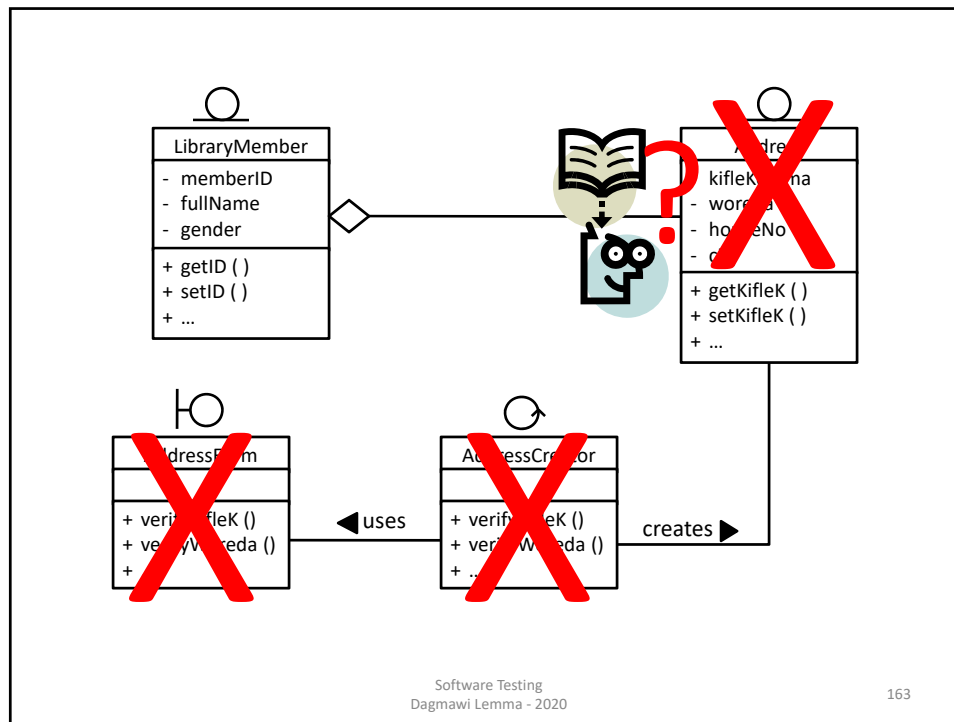
161

Object Oriented Testing

- Level of importance
 - consider a class in which a number of attributes are defined during the first iteration of OOA. If an extraneous attribute is appended to the class, two operations (at least, set and get) are then possibly specified to manipulate the attribute
- OR
- If the extraneous element is a class which is later found to be error as a review is conducted and a domain expert points it out

Software Testing
Dagmawi Lemma - 2020

162



Object Oriented Testing

- By eliminating such extraneous attribute, class, relationship, etc. at this early stage, the following problems and unnecessary effort may be avoided during analysis:
 1. Extra work/effort has been avoided
 2. incorrect or extraneous class relationships
 3. Improper behavior or business rules can be avoided

Object Oriented Testing

- On the other hand, if the error is not uncovered during analysis and propagated further :
 1. Improper allocation of the class to subsystem and/or tasks may occur during system design.
 2. Unnecessary design work may be expended to create the procedural design for the operations that address the extraneous attribute.
 3. The messaging model will be incorrect (because messages must be designed for the operations that are extraneous).

Software Testing
Dagmawi Lemma - 2020

165

Object Oriented Testing

Testing OOA and OOD models

- Correctness of OOA and OOD Models
 - The notation and syntax used to represent analysis and design models
 - syntactic correctness is judged on proper use of the symbology
 - each model is reviewed to ensure that proper modeling conventions have been maintained
 - semantic correctness
 - Problem domain experts shall examine the class definitions and hierarchy for omissions and ambiguity
 - model's conformance to the real world problem domain
 - accurately reflecting the real world

Software Testing
Dagmawi Lemma - 2020

166

Object Oriented Testing

Testing OOA and OOD models

- Consistency of OOA and OOD Models
 - An inconsistent model has representations in one part that are not correctly reflected in other portions of the model
 - each class and its connections to other classes should be examined
 - class-responsibility-collaboration model and
 - object-relationship diagram

Software Testing
Dagmawi Lemma - 2020

167

Object Oriented Testing

- Syntax of the model:
 - Are UML and ADT notation used correctly?
- Semantic correctness:
 - Does the model reflect the real world problem?
 - Is UML used as intended by its designers?
 - Is the ADT design complete (capturing all the classes and operations in UML diagram) and understandable?
- Testing for consistency:
 - An inconsistent model has representations in one part that are not reflected in other portions of the model

Software Testing
Dagmawi Lemma - 2020

168

Object Oriented Testing

Testing Strategies

- Unit Testing in the OO Context
 - the smallest testable unit is the encapsulated class or object
 - no longer test a single operation in isolation, rather as part of a class
 - Unlike unit testing of conventional software, which tends to focus on the algorithmic detail of a module and the data that flow across the module interface, class testing for OO software is driven by the operations encapsulated by the class and the state behavior of the class

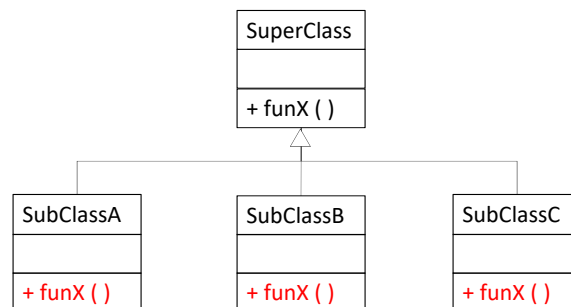
Software Testing
Dagmawi Lemma - 2020

169

Object Oriented Testing

Testing Strategies

- Unit Testing in the OO Context



Software Testing
Dagmawi Lemma - 2020

170

Object Oriented Testing

Testing Strategies

- Integration Testing in the OO Context
 - Two different strategies
 - Thread-based testing: Integrates the set of classes required to respond to one input or event for the system
 - Each thread is integrated and tested individually
 - Regression testing is applied to ensure that no side effects occur.
 - Use-based testing:
 - begins the construction of the system by testing those classes (called independent classes) that use very few (if any) of server classes. After the independent classes are tested, the next layer of classes, called dependent classes, that use the independent classes are tested

Software Testing
Dagmawi Lemma - 2020

171

Object Oriented Testing

Testing Strategies

- Integration Testing in the OO Context
 - Cluster testing
 - is one step in the integration testing of OO software
 - a cluster of collaborating classes is exercised by designing test cases that attempt to uncover errors in the collaborations

Software Testing
Dagmawi Lemma - 2020

172

Object Oriented Testing

Test Case for OO Software

- Approach to OO test case
 - uniquely identified test cases and explicitly associated with the class to be tested
 - purpose of the test case shall be stated
 - test steps shall be defined to learn about (find errors in)
 - specified **states** for the test component (class/object)
 - messages and operations that will be exercised
 - exceptions that may occur
 - external conditions
 - supplementary information that will aid in understanding or implementing the test

Software Testing
Dagmawi Lemma - 2020

173

Object Oriented Testing

Test Case for OO Software

- Conventional Vs OO test case design

Conventional Testing	OO Testing
driven by input-process-output view of software or the algorithmic detail of individual modules	focuses on designing appropriate sequences of operations to exercise the states of a class

Software Testing
Dagmawi Lemma - 2020

174

Object Oriented Testing

Test Case for OO Software

- Testing requires reporting on the concrete and abstract state of an object
 - encapsulation can make this information somewhat difficult to obtain
 - probably stubs can be used as built-in operations to report the values for class attributes

SavingAccount
-acNumber
-balance
-...
+ withdraw ()
+ <i>showBalance()</i>

*In the actual system
this could be provided
through the interface
of a boundary class*

Software Testing
Dagmawi Lemma - 2020

175