Testing OO Software - What's different?

- Less complexity in procedures
  Short methods
- Complexity is relocated
  to the connections among components
- Less problems
  based on intra-procedural and control flow
- More problems
  related to interaction between classes
- Less static determinism
  many faults can now only be detected at runtime

OO Mutation

Information Hiding
AMC - Access Modifier Change

Inheritance
HVD - Hiding Variable Deletion
HVI - Hiding Variable Insertion
OMD - Overriding Method Deletion
OMM - Overridden Method Moving
OMR - Overridden Method Rename
SKR - Super Keyword Deletion
PCD - Parent Constructor Deletion

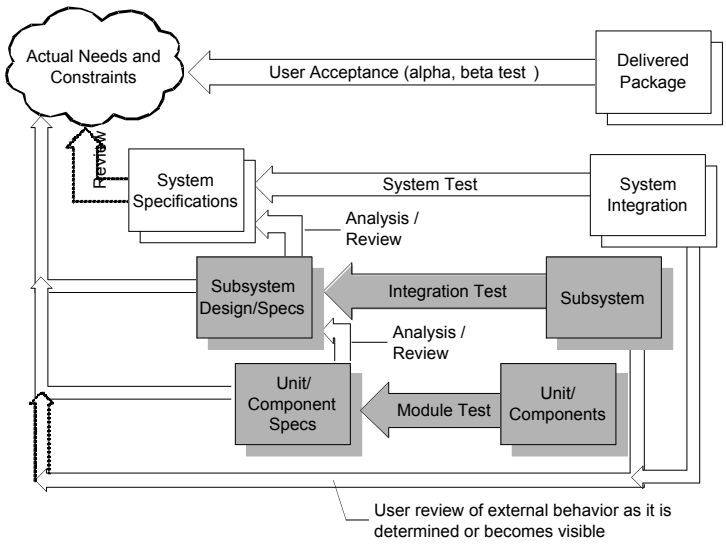
Polymorphism
ATC - Actual Type Change
DTC - Declared Type Change

RTC - Reference Type Change
PTC - Parameter Type Change

Overloading
OMC - Overloading Method Change
OMD - Overloading Method Deletion
AOC - Argument Order Change
ANC - Argument Number Change

Java Specific
TKD - this Keyword Deletion
SMV - Static Modifier Change
VID - Variable Initialization Deletion
JDC - Java Supported Default Constructor

Mutation testing defines a set of operators that simulate typical defects in OO systems. These mutation operators generate far less mutants than classical mutation operators, but also a higher percentage of equivalent mutants. Test cases to kill these mutants have to exercise classes in ways to distinguish them from ancestors or other polymorphic types, for example.



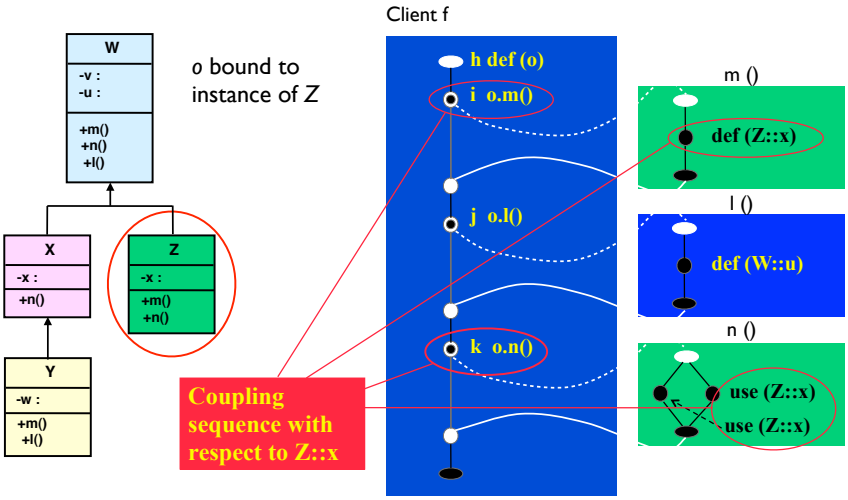
Object-oriented software only affects the module and integration test levels, the remaining test levels are independent of the type of software.

OO Testing Levels

- Intra-method testing : Testing individual methods within classes
- Inter-method testing : Multiple methods within a class are tested in concert
- Intra-class testing : Testing a single class, usually using sequences of calls to methods within the class
- Inter-class testing : More than one class is tested at the same time (integration)

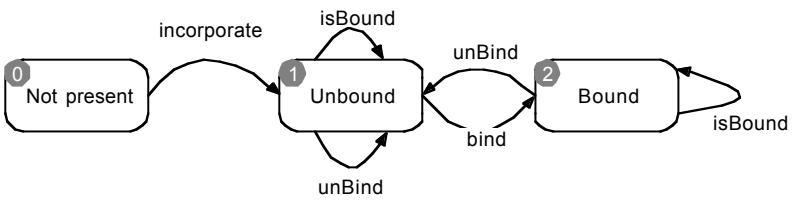
Test cases for intra- and inter-method testing are method calls, but for intra- and inter-class testing a test case consists of a sequence of method calls. In addition, setup code to get objects into the necessary states is required.

Example Coupling Sequence



A coupling sequence is a sequence of method calls in which a variable shared by a common instance context is defined in one method and used in another method.

Deriving an FSM and test cases



- TC-1: incorporate, isBound, bind, isBound
- TC-2: incorporate, unBind, bind, unBind, isBound

A simple analysis of the informal specification of class Slots allows to identify states and transitions. This state machine allows to derive sequences of method calls for intra-class testing, for example by choosing method call sequences that cover all states or transitions in the FSM.