



## 8. Unit and Integration Testing

Andrea Polini

Software Engineering II – Software Testing  
MSc in Computer Science  
University of Camerino

# Testing Phases and Objectives

- ▶ Unit test: each program unit functions as **expected**
- ▶ Integration test: integration of more units interact as **expected**
- ▶ System test: the whole systems behave as **expected**

# Unit testing

## Main characteristics

- ▶ generally performed by the developer of the unit
- ▶ test could be defined before writing the code
- ▶ white box adequacy criteria should be defined and targeted

## Testing activities

- ▶ Test plan
- ▶ Test design
- ▶ Test codification
- ▶ Test environment setting
- ▶ Test execution
- ▶ Test result analysis

# Unit testing strategies

## Test derivation

**Random testing** is a strategy often used to select runs to execute (test cases). In some situations it **behaves rather well**, in comparisons with more complex strategies. On the other hands there are situations in which performances are not so good. **The well-known triangle problem.**

## What's that

**J**Unit is the implementation for the Java world of the xUnit framework. This is a framework that support the phases of:

- ▶ coding
- ▶ environment setting
- ▶ execution
- ▶ analysis

# Mocks and Stubs

Unit to test could depends on “server” units

- ▶ **stubs**: simple software elements that provide dummy responses to requests from a unit under test
- ▶ **mocks**: more complex software elements that provide responses to requests from a unit under tests and can also check correctness of invocations

# Integration testing

## Integration testing rationale

Integration testing aims at showing **mismatches in the interactions among components**

It is possible that faults are discovered in relation of uncorrect behaviour of integrated units nevertheless this is **not the objective of integration testing**

The Mars orbiter example

# Integration testing

## Questions to answer

- ▶ Integration hierarchy
- ▶ Sequencing and stubbing
- ▶ Test generation

**Assumption:** all components have been tested well during unit testing



# Integration sequence

## Dependencies

The definition of an integration strategy should consider **dependency (relations) among units**.

- ▶ data dependency
- ▶ functional dependency
- ▶ control dependency

Dependencies have an impact on the integration sequence. **Object Relation Diagram** permit to capture dependencies among components

# Integration sequence

Which are the information that can be used for deciding integration strategies?

## Programming paradigm

- ▶ OO Programming
  - inheritance, association, aggregation/composition, dependencies (uses, calls, parameter, send, instantiates)
  - In OO programming particularly complex is the management of polymorphism, that creates dynamic dependencies
- ▶ Procedural programming
  - call graph

# Integration sequence

Which are the information that can be used for deciding integration strategies?

## Programming paradigm

- ▶ OO Programming
  - inheritance, association, aggregation/composition, dependencies (uses, calls, parameter, send, instantiates)
  - In OO programming particularly complex is the management of polymorphism, that creates dynamic dependencies
- ▶ Procedural programming
  - call graph

# Deciding on the integration strategy

## Integration hierarchy

- ▶ **Big bang**
  - not the same as system integration
  - can lead to difficulties in locating bugs
  - applicable when few components need to be integrated
  - no stubs or mocks need to be created
- ▶ **Top-down**
- ▶ **Bottom-up**

# Deciding on the integration strategy

- ▶ Availability of code
- ▶ Difficulty or ease of constructing stubs (strictly coupled classes)
- ▶ difficulty or ease of constructing drivers
- ▶ size of the test team
- ▶ need of showing partially working programs

# Optimal test order

The problem of finding an optimal integration sequence is **NP-complete** in literature different approaches to derive sub optimal solutions have been proposed.

- Tai-Daniels
- Traon-Jeron-Jezequel-Morel
- Briand-Labiche-Wang

Strategies foresees the analysis of the ORD to identify **Strongly Connected Components** (SCC) and then the removal of dependencies so to generate DAG with no cycles. Then the analysis give the number of stubs needed to proceed with integration testing.

# Test derivation and adequacy

- ▶ What are the techniques to derive tests for integration purpose?
- ▶ What are the adequacy criteria that can be used?