COMP 4004
Quality in OO Software Development

Course Outline
Learning Objectives

- Understand the pros and cons of TDD and of refactoring in Java
- Gain a basic understanding of metrics, xUnit patterns and Model Based Testing (MBT)
- Gain experience with JUnit and Cucumber
- Understand the pros and cons of state-based MBT (e.g., studying SpecExplorer)
- Understand the basics of scenario testing
  - Time permitting, understand the basics of scenario modeling using ACL and of corresponding scenario monitoring in Java.

Activities, Process, and Models
Requirements
Models

Requirements Doc.
Use Cases
Domain Objects

Analysis
Models

Objects
-structure
-scenario
-behavior

Design
Models

Objects
-structure
-scenario
-behavior

Implementation
Models

Code

Test Models

Test Plan, Test Cases
Test Drivers & Results

OO Development Overview

A Certification Iteration

Phase 1:
Creation of models, test models, and code
(LHS is top-down)

Phase 2:
Execution of test cases
(RHS is bottom up)

Customer feedback

Development

Validation & Verification

Product Test

Requirements

Analysis

Design

Construction

Consumption

Decomposition:
From complex system capabilities to individual classes

Incremental Delivery:
From simple classes to complex system capabilities

Risks, Goals and Requirements
Waterfall Development

Activities carried out one after the other as steps

- Requirements Capture
- Analysis
- Design
- Implement
- Testing

“The Big Bang approach to software development”

Incremental–Iterative Development

The Spiral Model

- Requirements Capture
- Analysis
- Design
- Implement
- Testing

Can have macro and micro iterations but needs to converge towards a solution
Traceability is required to achieve convergence:
- We must document the *continuity* that must exist between the work-products of different activities.
  - At least required for regression testing
  - In turn, continuity enables completeness and consistency checks.

Within a particular activity, the work-products must be *consistent*:
- e.g., the structural, scenario and behavior models must be consistent
  - For example, if an interaction diagram shows an object receiving a message, then the FSM of this object must reflect this possibility

Work-products must also be *complete* with respect to the current requirements.
A Scenario Driven Modeling Approach

Problem Description

Use Case 1

MSCs

Reqs

UCMs

Use Case 2

Inter-scenario relationships

Use Case 3

FSMs and/or code

Is it traceable?

The Agile Manifesto—a statement of values

<table>
<thead>
<tr>
<th>Individuals and interactions</th>
<th>over</th>
<th>Process and tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working software</td>
<td>over</td>
<td>Comprehensive documentation</td>
</tr>
<tr>
<td>Customer collaboration</td>
<td>over</td>
<td>Contract negotiation</td>
</tr>
<tr>
<td>Responding to change</td>
<td>over</td>
<td>Planning over commitment</td>
</tr>
</tbody>
</table>

Source: www.agilemanifesto.org

Page 6
The Core of Agile Modeling

Some Core Principles
◦ Assume Simplicity
◦ Expect Incremental Change
◦ Enabling the Next Effort is Your Secondary Goal
◦ Model With a Purpose
◦ Use Multiple Models
◦ Maximize Stakeholder Investment
◦ **Value Quality first**
◦ Get Rapid Feedback
◦ **Software Is Your Primary Goal**

Some Core Practices
◦ Active Stakeholder Participation
◦ Apply the Right Artifact(s)
◦ **Collective Ownership**
◦ Create Several Models in Parallel
◦ Create Simple Content
◦ Depict Models Simply
◦ Display Models Publicly
◦ **Model in Small Increments**
◦ Model With Others
◦ Prove it With Code
◦ Use the Simplest Tools

Scrum

Sprint 2-4 weeks

24 hours

Sprint goal
Return

Cancel
Coupons
Gift wrap
Product backlog

Sprint backlog

Potentially shippable product increment