Advanced Software Engineering Techniques

Course 1 – October, 9, 2017

Adrian Iftene
adiftene@info.uaic.ro
ASET – overview

- SWEBOK: place and role of software engineering, knowledge areas (KAs), related disciplines
- Development and maintenance of the systems: model driven agile development, patterns of enterprise application architecture, test-driven development, refactoring: code architecture
- Object oriented design classes: SOA, object-oriented design principles
- Modeling, business modeling: BPMN, domain specific languages (DSL), frameworks: Eclipse Modeling Framework (EMF), Open Architecture Ware (OAW)
Model driven development

- Model-driven development (MDD) – software methodology focused on creating models close to a specific field than informatics concepts

- Model-driven architecture (MDA) is the best known initiative of MDD and was launched by the group OMG (Object Management Group) in 2001
Agile model driven development

- AMDD is agile version of MDD

- Identify the high-level scope
- Identify initial “requirements stack”
- Identify an architectural vision

- Modeling is part of iteration planning effort
- Need to model enough to give good estimates
- Need to plan the work for the iteration

- Work through specific issues on a JIT manner
- Stakeholders actively participate
- Requirements evolve throughout project
- Model just enough for now, you can always come back later

- Develop working software via a test-first approach
- Details captured in the form of executable specifications
Test driven development

- Test-Driven Development – TDD

**TDD steps:**

1. Add a test.

2. Execute tests; the new test will fail.

3. Add functional code such that pass all tests.

4. Run tests again.
   - If the test fails, go to 3.
   - If the tests pass successfully, we can continue with other functionality

5. Refactoring code (functional and testing)
Modeling

- IBM Rational Rose Modeler
- BPMN
- Domain specific languages (DSL)

- Working frameworks:
  - Eclipse Modeling Framework
  - Open Architecture Ware (OAW)
Business Process Modelling Notation (BPMN) is a graphical representation for specifying business processes in a workflow.
BPMN – Example

Issue Voting List [0 to 5 Issues]

Announce Issues for Discussion

Moderate E-mail Discussion

7 Days

Delay 6 days from Announcement

Check Calendar for Conference Call

Conference Call in Discussion Week?

Yes

Moderate Conference Call Discussion

No

E-Mail Discussion Deadline Warning

Evaluate Discussion Progress

This Task returns the value of the Discussion Over to True or False

Allow 1 week for the discussion of the Issues – through e-mail or calls

Calendar

Wait until Thursday, 9am

The Sub-Process will repeat of the DiscussionOver variable is False
Eclipse Modeling Framework

- EMF is an Eclipse-based modeling framework and code generation facility for building tools and other applications based on a structured data model.
AOP is a programming paradigm which isolates secondary or supporting functions from the main program’s business logic.

AOP increases modularity by allowing the separation of cross-cutting concerns.

AOP includes programming techniques and tools that support the modularization of concerns at the level of the source code.
AOP – Basic Terminology

- Cross-cutting concerns – aspects of a program which affect other concerns
- Advice – additional code
- Pointcut – point where additional code is executed
- Aspect – the combination of the pointcut and the advice
AOP Languages

- Examples:
  - AspectJ,
  - CaesarJ,
  - CLOS,
  - Compose,
  - JAsCo,
  - ObjectTeams
AOP – AspectJ – Hello World!

```java
public aspect HelloWorldBeforeAfter {
    pointcut mainMethod() : execution(public static void main(String[]));

    before() : mainMethod() {
        System.out.println("Before Hello World!");
    }

    after() returning : mainMethod() {
        System.out.println("After Hello World!");
    }
}
```

- aspect
- pointcut
- advice
AOP – AspectJ – Example 2

- **Problem**: we want to know when something changes the student (*name* or *grade*)
- **Solution**: we add a pointcut for all “set” methods
AOP – AspectJ – Example 3

- **Problem**: we want to trace our program execution
- **Solution**: we add a pointcut for all methods

```java
package student;

public aspect TraceAspect {

    // tracing
    pointcut trace () : call (* *(.) as ! within(TraceAspect));

    before () : trace () {
        System.out.println("TRACE":"+thisJoinPoint.getSignature()");
    }
}
```

- set
- println
- toString
SOA

- SOA (Service Oriented Architecture) involves distributing application functionality into smaller units, distinct – called services – which can be distributed over a network and can be used together to create complex applications.

- Services are independent functional units that solve specific problems and can be combined to solve complex problems.

- They can also be reused in different applications.
Examples of services:
- complete an application online to create an account
  view a bank statement
- make an online ticket orders
- For a robot: services for vision, hearing, moved
SOA - .NetROBOT - Tudor D.
Object–oriented design principles

- Architecture and dependencies: *When we say that we have a degraded project?*

- Design principles of classes: *responsibility, dependencies, separation*

- Architecture design principles:
  - Reuse, versioning, closing
  - Coupling, dependence

- Object–oriented design patterns:
  - Abstract server, Adapter, Observer, Bridge, Abstract Factory
Architectural degradation

- Rigid – hard to change
- Fragile – changes have undesirable effects
- Immobility – separation into components is difficult
- Viscous – things not running to properly
- Additional complexity
- Additional repetition
- Opacity – hard to understand
Refactoring

- Successive changes lead to sub-optimal code structure
  - increase the complexity
  - decreases clarity
- Refactoring is a change in the internal structure of a software product in order to make it easier to understand and modify without changing its observable behavior
- Results:
  - decreased coupling
  - increasing cohesion
Refactoring – When?

- The following situations are signals for refactoring:
  - duplicate code
  - long methods
  - large classes
  - Long lists of parameters
  - Instructions switch by type objects – is recommended polymorphism
  - Speculative Generality – Hierarchy of classes that subclasses have the same behavior
  - Intense communication between objects (strong coupling)
  - Chaining of messages
Github

- Github – code
- GoogleDrive – documents
Links

- Agile Model Driven Development (AMDD) http://www.agilemodeling.com/essays/amdd.htm
- OAW http://www.openarchitectureware.org/
Bibliography
