Software Engineering Design and Construction

159.251

Revision lecture

Amjed Tahir

a.tahir@massey.ac.nz

- Exam: Wednesday 28 November 2017
  - **Venue**: 6-304 (South Campus)
  - **Time:** 9am-12pm (3 hours)

Spend time based on the mark given to the question!

- Exam structure:
  - 52 marks in total
  - Divided equally between the 2 parts of the paper
  - Answer all questions

Topics covered in this paper

- Part 1: effective software development
  - DRY and duplication in code and design
  - Separation of concerns and orthogonally
  - Organizing your code
  - Software testing
  - Design Patterns, anti-patterns and refactoring.

Topics covered in this paper

- Part 1: collaborative software development
  - Scripting (Bash)
  - Agile Software Development
  - Version control – using git as an example
  - Software Configuration Management -issue tracking
  - Software Measurements and metrics
  - Continuous development
  - Software Licensing

Evil of Duplication

- Copy and past programming
- The DRY principle
- Layered model:
  - UI, Domain and Persistency.
- Types of duplications:

  Imposed (no choice), inadvertent (not realizing it), impatient (the dirty way!), inter-developer (communication issues)
- JAXB to preserve persistency.

Separation of Concerns and Orthogonality

- Complexity of systems – dependency between modules.
  - Coupling and cohesion
- Logging in Java – log4j
- Liskovs Substitution Principle (LSP)
-
- Dependency Injection and Service Locator.

Organising Code

- Comments – auto-generated comments
- Comments to manage Technical debt (TODO, FIXME)
- Javadoc
- Formatting your code for better readability.
- Code Beaufitiers, Uglifiers and Obfuscators
- Velocity templetes etc..
- Naming convention
- Using refelction

Testing

- Semantic testing vs Compilation.
- Assertion vs assumption
- Testing level: unit, integration and system.
- Test oracle, test fixture and test termination.
- Full test, boundary tests, critical test etc..
- Code/test coverage metrics (explain also in slide 56 in Lecture 5: metrics):
  - Statement (Line), class, branch etc
  - Full coverage test vs good test.
- TDD
  - Test first vs test last

Design Patterns, Antipatterns and Refactoring

- SOLID principle
- Design Patterns from the GangOf4
  - Singleton: there is only one instance of the object.
  - Adapter: wrapping - using delegation
  - Factory: decouples client applications from service provides
  - Lazy Initialization vs Eager Initialization .
- Anti-patterns:
  - Subtype Knowledge, Swiss Army Knife, Spaghetti Code
- Code smells:
  - God class, Feature Envy, type checking, long methods
- Test smells:
  - Eager Test, Sensitive Equality, Assertion-free
- Refactoring

Agile Software Development

- What's agile software development?
- Agile vs Traditional software development.
- Team work over individuality.
- Agile methods:
  - XP, Scrum, TDD, Kanban etc..

Version control

- What's version control?
- what is meant by a "***commit***" in the context of version control systems?
- Differences between *centralised* and *decentralised* models with examples.
- Explain the idea of ***branch*** and ***merge*** in a version control system.
- What is **.gitignore** used for?

Software Configuration Management

- Managing issues e.g., bugs.
- Managing change
- Issue tracking, why are they important?
- How does an issue tracker system work?

Software Metrics

- What's a software metrics?
- Why do we measure our software?
- Process vs. products metrics!
- Static vs. dynamic!
- Cyclomatic complexity and LOC
- Measuring dependency: OO metrics DIT, CBO (IC, EC)

Software Licensing

- Explain the main differences between the following open source licences:
  - MIT,BSD, GPL v2 & v3,Apache, Eclipse
- Even if your happy to give your software away, adding a licence is still a good idea. Why?

Continuous Integration and Continuous Development

- CI in Agile software development.
- What do you need in a CI and CD process?
  Toolset..
- What's the difference between Continues Deployment and Continuous Delivery?

What to study?

- Understand well what's DRY principle and how it works – practice examples!
- Understand how logging frameworks work.
- Understand what Convention over Configuration means (getter, setter etc..)
- How to perform testing, testing levels, coverage etc.. And what make a good test…
- Understand assertion and assumption.
- Automation tools - maven, ant etc…
- Specific patterns with example. Use java to explain your answer

- Agile software development – why it is different than previous methods.
- Understand different agile methods and their practices: only the ones that we covered in the lectures.
- CI and CD
- Using shell scripts – understand different commands as explained in your lectures.
- Metrics- size and CC metrics.
- Versions control – using git as an example
- Configuration management – issue tracking
- Licensces: permissive (BSD, and MIT) and copyleft (GPL)