Syllabus: CSCI 630 Software Design & Maintenance

Kevin Buffardi

October 2018

Overview

A study of designing and maintaining complex software. The course builds upon fundamental software engineering skills with an emphasis on: object-oriented software design patterns, anti-patterns, code review and refactoring, and tools for evaluating code quality. Students practice maintaining software by collaborating on a large-scale open source project using automated development operation (DevOps) tools.
Prerequisites: CSCI 430 and classified graduate standing.

Instructor

Kevin Buffardi, kbuffardi@csuchico.edu, Office hours: To be annouced.

Required Materials

Laptop computer; No textbook required
Suggested reading:

"Design Patterns: Elements of Reusable Object-Oriented Software"
(Gamma, Helm, Johnson, Vlissides)
ISBN: 0-201-63361-2

"The Mythical Man-Month: Essays on Software Engineering"
(Brooks)
ISBN: 0-201-83595-9

Schedule

This is the tentative semester schedule, subject to change.

1. Course & project introduction, accelerated review of advanced version control (*Git & GitHub*)
2. Interfaces & Advanced Object-Oriented Design principles (*Java*)
3. Introduction to design patterns and anti-patterns
4. Composition pattern designs
5. Implementing Composite, Adapter, & Decorator patterns
6. Creational pattern designs
7. Implementing Singleton & abstract factory patterns
8. Behavioral pattern designs
9. Implementing Iterator, Observer, & Strategy patterns
10. Design Patterns review and exam
11. Bug tracking, code review, & refactoring (*Java, GitHub*)
12. Accelerated review of unit testing (*JUnit*)
13. Build Automation & Continuous Integration (*Gradle, Jenkins*)
14. Static and Coverage Analysis (*PMD, Cobertura*)
15. Project review

Learning Outcomes

Learning comes in different forms. From this course, students are expected to *minimally* gain the following learning outcomes, with **Core Body of Knowledge** topics from ACM Curriculum Guidelines for Graduate Degree Programs in Software Engineering

- *Comprehension and Application* of **Software Design Fundamentals** including: general design concepts; context of software design; and software design process
- *Application* of **Key Issues in Software Design** including: distribution of components; interaction and presentation; and data persistence
- *Application and Analysis* of **Software Structure and Architecture** including architectural styles (macro architectural patterns) and design patterns (micro architectural patterns)
- *Application* of **Software Design Quality Analysis and Evaluation** including: quality attributes; quality analysis and evaluation techniques; and measures
- *Application* of **Software Design Notations** including both structural (static) descriptions and behavioral (dynamic) descriptions
- *Application and Analysis* of **Software Design Strategies and Methods** with an emphasis on Object-oriented design
- *Application and Analysis* of **Testing Techniques** with a focus on unit testing and **Test-Related Measures** by evaluation of tests with code coverage
- *Comprehension* of **Software Maintenance Fundamentals** including: definitions and terminology; nature of maintenance; and need for maintenance
- *Application* of **Key Issues in Software Maintenance, the Maintenance Process, and Techniques for Maintenance** including: technical and management issues; software maintenance measurement; maintenance activities; program comprehension; and reengineering

These outcomes are categorized according to Bloom's Taxonomy of the Cognitive domain, as *italicized*.