CS 6301

(Advanced Topics in Software Engineering)

Instructors

Alessandro Orso

Recommended textbook

No required text. Students should have access to a Software Engineering introductory text, such as the books by Pressman or Sommerville.

Prerequisites

- Students are expected to have taken an undergraduate software engineering course or have some experience in industry.
- Students should already know how to program.
- Students should have graduate-level standing or permission of the instructor.

Learning Objectives

Upon successful completion of this course, students will be able to:

- Describe the way software is developed, verified, and deployed in modern companies.
- Use tools and technologies commonly utilized in today's development organizations.
- Analyze, explain, and apply cutting edge research techniques aimed at developing better quality software.
- Apply the concepts learnt in the classroom in practice in real-world projects that involve teamwork and collaboration among teams.

Academic Integrity

Academic dishonesty will not be tolerated. This includes cheating, lying about course matters, plagiarism, or helping others commit a violation of the Honor Code. Plagiarism includes reproducing the words of others without both the use of quotation marks and citation. Students are reminded of the obligations and expectations associated with the Georgia Tech Academic Honor Code and Student Code of Conduct, available online at http://www.honor.gatech.edu.

Learning Accommodations

If needed, we will make classroom accommodations for students with documented disabilities. These accommodations must be arranged in advance and in accordance with the Office of Disability Services (http://disabilityservices.gatech.edu).

Excused Absence Policy

See http://www.catalog.gatech.edu/rules/4/

Topics Covered

• Introduction and overview • Software processes • Integrated development environments • Version control systems • Requirements engineering • Software prototyping • Software architecture • Software design	• Design patterns • Refactoring • Software testing • Software maintenance • Static analysis techniques and tools • Various modern and cutting edge technologies (e.g., Android OS, Android Wear, Google APIs, Continuous delivery, REST, BDD, PaaS, Cloud DBs)

In addition, student will have the opportunity to listen from and interact with real-world software engineers during engaging guest lectures.

Detailed Syllabus

Introduction and overview

- History of software engineering
- Software process and software phases
- Tools of the trade

Software processes

- Traditional software phases
  - Requirements engineering
  - Design
  - Implementation
  - Testing
  - Maintenance
- Lifecycle documents
- Main types of lifecycle models, from waterfall to agile
- Classic mistakes

Integrated development environments

- Editing, compiling, running, and testing code within an IDE

Version control systems

- General concepts
- Centralized VCS (e.g., CVS, SVN)
- Distributed VCS (e.g., Mercurial, GIT)
- Git and GitHub

Requirements engineering

- Definition and properties of requirement
- Functional and non-functional requirements
- User and system requirements
- Definition of requirements engineering and why it is needed
- Requirements engineering activities
- Software requirements specification
- Prototypes and requirements

Software architecture

- What is software architecture?
- Architectural evolution
- Software architecture’s elements: components, connectors, and configurations
- Architectural styles
- Architecture in action

Software design

- From architecture to design
- Example of high- and low-level design
    - Class diagrams
    - Component diagrams
    - Deployment diagrams
- Design principles
    - Coupling
    - Cohesion

Design patterns

- History of design patterns
- Format and essential elements
- Choosing a pattern
- Detailed discussion of a few well-known patterns

Refactoring

- History of refactoring
- Examples of refactorings
- Refactoring within IDEs
- Refactoring guidelines
- Bad smells

Software testing

- Overview and terminology
- Verification techniques
- Granularity levels
- Functional (black-box) testing
    - Random testing
    - Equivalence partitioning
    - Boundary testing
    - Category partition method
    - Model-based testing
- Structural (white-box) testing
    - Test requirements, test specifications, and test cases
    - Main coverage criteria and their limitations

Software maintenance

- Configuration management
- Regression testing
- Debugging

Static analysis techniques and tools

- Static versus dynamic verification
- Strengths and weaknesses of static analysis
- Static analysis techniques and tools

Various modern and cutting edge technologies

Some examples (the list will be updated over the years):

- Android OS
- Google APIs
- Continuous Integration
- Continuous Delivery
- REST
- BDD
- PaaS
- Cloud DBs