CSC 331/631
Software Engineering
T-Th 2:00 - 3:15pm, Spring 2018, Manchester 241

Instructor
Dr. V. Paúl Pauca
office: 235 Manchester Hall
tele: 336.758.5454
email: paucavp@wfu.edu
course web: http://www.cs.wfu.edu/~pauca/CSC331_631.html

Course
Textbook:
Software Engineering, 9th Edition by I. Sommerville (suggested)
Javascript: The Good Parts by D. Crockford (suggested)

Grading:
5 quizzes ................. 20%
4 project iterations ......... 60%
Poster review & presentation . 10%
Customer exit interview .... 5%
2 screencasts ............... 5%

Graduate students registered in CSC 631 will be expected to do review of a software engineering paper assigned by the instructor.

Major Topics to be Covered
• Software processes
• Software project management
• Tools and environments
• Requirements engineering
• Software design
• Software construction
• Software verification and validation

Primary Learning Outcomes: By the end of the course students will able to

• Software Processes
  – Describe how software can interact with and participate in various systems including information management, embedded, process control, and communications systems. [Familiarity]
  – Describe the relative advantages and disadvantages among several major process models (e.g., waterfall, iterative, and agile). [Familiarity]
  – Describe the different practices that are key components of various process models. [Familiarity]
Differentiate among the phases of software development. [Familiarity]

Describe how programming in the large differs from individual efforts with respect to understanding a large code base, code reading, understanding builds, and understanding context of changes. [Familiarity]

Explain the concept of a software lifecycle and provide an example, illustrating its phases including the deliverables that are produced. [Familiarity]

Compare several common process models with respect to their value for development of particular classes of software systems taking into account issues such as requirement stability, size, and non-functional characteristics. [Usage]

**Software project management**

Discuss common behaviors that contribute to the effective functioning of a team. [Familiarity]

Create and follow an agenda for a team meeting. [Usage]

Identify and justify necessary roles in a software development team. [Usage]

Understand the sources, hazards, and potential benefits of team conflict. [Usage]

Apply a conflict resolution strategy in a team setting. [Usage]

Use an ad hoc method to estimate software development effort (e.g., time) and compare to actual effort required. [Usage]

List several examples of software risks. [Familiarity]

Describe the impact of risk in a software development lifecycle. [Familiarity]

Describe different categories of risk in software systems. [Familiarity]

**Tools and Environments**

Describe the difference between centralized and distributed software configuration management. [Familiarity]

Describe how version control can be used to help manage software release management. [Familiarity]

Identify configuration items and use a source code control tool in a small team-based project. [Usage]

Describe how available static and dynamic test tools can be integrated into the software development environment. [Familiarity]

Describe the issues that are important in selecting a set of tools for the development of a particular software system, including tools for requirements tracking, design modeling, implementation, build automation, and testing. [Familiarity]

Demonstrate the capability to use software tools in support of the development of a software product of medium size. [Usage]

**Requirements Engineering**
– List the key components of a use case or similar description of some behavior that is required for a system. [Familiarity]
– Describe how the requirements engineering process supports the elicitation and validation of behavioral requirements. [Familiarity]
– Interpret a given requirements model for a simple software system. [Familiarity]
– Describe the fundamental challenges of and common techniques used for requirements elicitation. [Familiarity]
– List the key components of a data model (e.g., class diagrams or ER diagrams). [Familiarity]
– Identify both functional and non-functional requirements in a given requirements specification for a software system. [Usage]
– Conduct a review of a set of software requirements to determine the quality of the requirements with respect to the characteristics of good requirements. [Usage]

• Software Design

– Articulate design principles including separation of concerns, information hiding, coupling and cohesion, and encapsulation. [Familiarity]
– Use a design paradigm to design a simple software system, and explain how system design principles have been applied in this design. [Usage]
– Construct models of the design of a simple software system that are appropriate for the paradigm used to design it. [Usage]
– Within the context of a single design paradigm, describe one or more design patterns that could be applicable to the design of a simple software system. [Familiarity]
– For a simple system suitable for a given scenario, discuss and select an appropriate design paradigm. [Usage]
– Create appropriate models for the structure and behavior of software products from their requirements specifications. [Usage]
– Explain the relationships between the requirements for a software product and its design, using appropriate models. [Assessment]
– For the design of a simple software system within the context of a single design paradigm, describe the software architecture of that system. [Familiarity]
– Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-server. [Familiarity]
– Investigate the impact of software architectures selection on the design of a simple system. [Assessment]
– Apply simple examples of patterns in a software design. [Usage]
– Describe a form of refactoring and discuss when it may be applicable. [Familiarity]
– Select suitable components for use in the design of a software product. [Usage]
– Explain how suitable components might need to be adapted for use in the design of a software product. [Familiarity]
– Design a contract for a typical small software component for use in a given system. [Usage]

• **Software Construction**
– Describe techniques, coding idioms and mechanisms for implementing designs to achieve desired properties such as reliability, efficiency, and robustness. [Familiarity]
– Build robust code using exception handling mechanisms. [Usage]
– Describe secure coding and defensive coding practices. [Familiarity]
– Select and use a defined coding standard in a small software project. [Usage]
– Compare and contrast integration strategies including top-down, bottom-up, and sandwich integration. [Familiarity]
– Describe the process of analyzing and implementing changes to code base developed for a specific project. [Familiarity]
– Describe the process of analyzing and implementing changes to a large existing code base. [Familiarity]

• **Software Verification and Validation**
– Distinguish between program validation and verification. [Familiarity]
– Describe the role that tools can play in the validation of software. [Familiarity]
– Undertake, as part of a team activity, an inspection of a medium-size code segment. [Usage]
– Describe and distinguish among the different types and levels of testing (unit, integration, systems, and acceptance). [Familiarity]
– Describe techniques for identifying significant test cases for integration, regression and system testing. [Familiarity]
– Create and document a set of tests for a medium-size code segment. [Usage]
– Describe how to select good regression tests and automate them. [Familiarity]
– Use a defect tracking tool to manage software defects in a small software project. [Usage]
– Discuss the limitations of testing in a particular domain. [Familiarity]

---------------------------------------------------------------

**Course Organization:** This course is a hands-on introduction to fundamental topics in software engineering. It follows roughly the teaching philosophy underlying engineering software as a service developed by professor Armando Fox at Berkeley. As such, the course emphasizes web application development using Node.js and Javascript (instead of Ruby on Rails), agile processes and collaboration via Github, Pivotal Tracker, automated code review, cloud hosting via Heroku, and interaction with real customers in our local community.
**Video Lectures:** During the semester, students will be expected to watch a number of video lectures produced by Professor Alex Orso at Georgia Tech, which are also part of his course in Udacity. Professor Orso has graciously agreed to share this videos with us. Students are expected to watch the assigned videos outside the classroom and be prepare to discuss them in class.

**Project:** The project is the main avenue for learning software engineering concepts. We will form groups of 4 or 5 people and work with customers in our local community throughout the semester. Student teams are expected to be fully engaged in the process and all members are expected to contribute a roughly equal share. The project will take place in four iterations. The goal of the multiple iterations is to grade progress being made, the process, as well as the deliverables.

**Quizzes:** There will be 5 short quizzes spread out through the semester based primarily on material presented in the lectures. Specific dates are shown in the tentative course schedule in the course website. A make-up quiz will be administered only under exceptional circumstances.

**Poster review & presentation:** Poster presentations will be done on the day scheduled by the university for final exams. For us, this is May 2 at 2pm. Groups will be required to do a brief review of their posters the week prior to the poster presentations. Customers, and other people external to the course will be invited to the poster presentations.

**Customer exit interviews:** Customers will be briefly interviewed in person by the instructor to get their feedback relative to performance of each team. This will take place sometime after the poster presentations.

**Screencasts:** Students will be required to produce short videos of their interviews with the customer. This does not require any fancy equipment. Phone cameras and simple video editing will suffice.

**Academic Integrity:** Group assignments require collaboration as well as online research. Students are expected to carry their own weight in group work and behave professionally in all group meetings and customer meetings. While groups are expected to use online material, copying of partial or complete work without properly reference to the author will not be tolerated.

**Special Needs:** If you have a disability that may require an accommodation for taking this course, then please inform Professor Pauca and contact the Learning Assistance Center (758-5929) within the first two weeks of the semester.

**Course Plan in the Event of Closure of the University:** In the event that the University closes due to pandemic or other disaster, the course will be continued through the internet or by postal mail, if the former is not available. Professor Pauca will distribute class notes, weekly lab material, and homework through the course webpage ([www.cs.wfu.edu/~pauca/CSC331_631.html](http://www.cs.wfu.edu/~pauca/CSC331_631.html)) or by postal mail in the case of internet service failure. Class notes will contain extensive lecture material as well as short exercises designed to test the students comprehension of the
material. Office hours will be held through the internet using Webex. Students will be required to turn in assignments electronically or by postal mail. Examinations will be distributed by internet, email, or postal mail, as needed. Return date and time for examinations will be clearly specified. In addition Professor Pauca will be accessible by email through any of the following addresses: paucavp@wfu.edu and paulpauca@gmail.com.