CS 575 – Software Design

Instructor: Giuseppe (Peppo) Valetto
E-mail: valetto@cs.drexel.edu
Office: University Crossings 106
Phone: 215-895-1518
Office Hours: Mon 4-6 and Fri. 2-3 University Crossings #106
Class Hours: Thurs. 6:00 pm-8:50 pm
Classroom: University Crossing 153

Introduction
CS 575 is a graduate-level course that introduces and analyzes in depth key principles and methodologies for software design. In this course, students will learn and discuss about foundational concepts in the discipline of software design, and will experiment and reflect upon ways in which design decisions can be expressed and implemented in a software system.
The course material is drawn from canonical software engineering texts, as well as state-of-the-art research papers.

Class goal
The main goal of CS 575 is for students to gain a thorough understanding of what it means to “design a software system”, with respect in particular to the challenges that must be faced and the techniques that must be used in professional design of complex, industry-grade software.

Class Objectives
• Reach a thorough understanding of fundamental principles of software design
  o understand the underpinning of “good” design
  o be able to evaluate and make informed decisions when faced with multiple design options
• Understand and be able to apply state-of-the-art modeling abstractions, techniques and tools
• Acquire experience in how design models and decisions are translated into the implementation of a working software system
• Get exposed to seminal research in this branch of software engineering
• Get an outlook on fundamental open problems in the discipline of software design, as well as on approaches that have been used to address them

Prerequisites
The course is open to Engineering graduate students - in particular (but not limited to) Computer Science and Software Engineering students - as well as senior undergraduate students with adequate GPA and background.
Although there are no official prerequisites, to do well in the course, the following must be true:
• you need to have an understanding of the software engineering life cycle and related practices;
• you need to have acquired substantial programming experience in some advanced object-oriented language, such as Java, C# (C++ is a viable alternative, but will possibly require more work on your part);
• you have substantial system-building experience;
• you ideally have some degree of familiarity with the UML design modeling language.

In case you want to discuss where you stand with respect to any of the above, please come see the instructor

Topics

Lectures are intended to be strongly interactive, class participation is required and strongly encouraged, and critical thinking and personal contribution to the in-class discussion are part of what is expected of students.

Major topics discussed in this course include the following:
• Design process
• Design principles
• Software architecture
• Component-based design and Service Oriented Architectures (SOAs)
• Modularization techniques
• Object-oriented design and UML
• Alternative modularization techniques, such as Aspect-oriented development
• Agile and emergent design
• Socio-technical concerns in software design

This list is tentative, and subject to change, depending on class needs.

Textbook

“Software Fundamentals – Collected Papers by David L. Parnas"
by: D.M. Hoffman and D.M. Weiss (eds.)
Publisher: Addison Wesley (2001)
ISBN: 0201703686

The textbook shall be available through the Drexel bookstore. Supplemental readings from technical and scientific papers will be indicated by the instructor in class.

Class organization

A significant portion of each class will be devoted to presentation and discussion of relevant subjects, driven by students and facilitated by the instructor. The instructor will also introduce new subjects and explain particularly important and/or complex aspects of the covered material.

Course components

1. Reading assignments:
Students are supposed to carry out preparatory readings from the textbook and other sources indicated by the instructor as part of their assignments, before the class meets. Their reflections on the material will fuel the in-class discussion. Each week, and for each of the reading assignments, a student will be chosen as the Designated Presenter, and will assume the responsibility to prepare an in-class presentation about that particular reading item.

2. **Class participation:**
   Engaging discussion of the lecture and reading material is considered a fundamental component of this class.
   Participation in the discussion on reading assignments, specifically, occurs through two different: in-class discussion, as well as online discussion, through the class discussion board. With respect to online discussions, student contribution to the online threads will be graded on a 0-3 scale: 0 means no participation; 1 means marginal contribution; 2 means satisfactory contribution; 3 means exceptional contribution. Satisfactory participation requires not simply posting comments, but stimulating the discussion by proposing questions, answering questions and replying to other’s comments, pointing additional or related material, etc. The Designate Presenter is supposed to integrate the above contributions in her presentation.

3. **Group project:**
   A substantial part of the course work is a multi-week collaborative project. Students will work in small teams on a project that will focus in particular on design problems, and is intended to bring together several of the aspects discussed in class. Further information in the Class Project section below.

4. **Group position paper:**
   The same groups working on the project will compose a short technical paper, on a topic related to some of the more conceptual aspects of software design discussed in class. The group will present their position paper in class as part of their final requirements, aiming at stimulating exchange of ideas with the rest of the class and the instructor on the chosen topic.

5. **Take-home exams:**
   The course will include at least one (midterm or final) individual exam. Those exams will take the form of take-home questions and essays, which will require of the students some independent research of prominent literature in software design, and expanded exploration of issues covered in class.

6. **Quizzes and in-class exercises:**
   In-class practical and problem-solving exercises will be carried out from time to time, with the goal to highlight and explicate specific issues that have been addressed during that class. Some of those exercises and problems will have a programming connotation: bring your laptops! Other in-class activities may include quizzes. The work done during those in-class activities will be rewarded with credit: students who are absent or unprepared to carry out those activities will not receive credit.

7. **Collaboration assessment:**
   Given the importance of team work in this course, and to ensure all members of a group responsibly and actively work towards the team’s objectives, coordination
Credit will be accorded (10% of the final grade). During the last week, each student will review him/herself as well as his/her team co-workers in terms of responsibility and contribution to the project and the position paper. The reviews of each group will be used by the instructor to relatively assess the contribution of all team members, and assign collaboration credits.

**NOTE:** in case you run into any kind of coordination and teaming problems at any time during the term, the best strategy is to report them to the instructor right away.

**Class Project**

The group project can take two forms:

1. *Canonical choice:* this is an application design and development project. The team will be asked to design, implement and test a distributed application using a Service Oriented Architecture, and related techniques and tools. The team will be able to submit an application of their interest. The instructor will also provide examples of reasonable SOA-based application that teams may want to embrace. Deliverables for this kind of projects will follow a typical (although shortened!) cycle of software production, with – of course – a strong emphasis on design deliverables and their quality.

2. *Non-canonical choice:* for teams with a substantial previous experience of SOA, or who might want try their hand at something else, or would like to explore advanced and/or research-oriented topics in software design. Those teams can propose a different kind of project, which must be centered upon some interesting design ideas. The instructor reserves a right to accept such a proposal, upon proper brainstorming with the team. Deliverables for this kind of projects will be jointly decided by the team and the instructor, depending on the nature and the objective of each project.

**Grading**

- Reading assignments: 10%
- Student participation: 10%
- Group project: 30%
- Group position paper: 20%
- Take-home exams: 20%
- In class exercises and quizzes: 10%

**Class Policy**

- **Do’s and Don’t’s**
  - Be punctual
  - Switch off cell phones
  - No laptops for personal use

- **Adhere to standard academic honesty rules.**
  - See: [http://www.drexel.edu/provost/policies/academic_dishonesty.asp](http://www.drexel.edu/provost/policies/academic_dishonesty.asp)

- **Late policy**
You will submit all assignments through the BB Vista WWW facilities. Submissions with a timestamp later than the due date and time stated in the assignment specifications are deemed late. They will be dealt with as follows:

- Graded with a 20% penalty if submitted within 1 day from the official assignment deadline
- Not graded if submitted later than 1 day

*Force majeure* exceptions (e.g. for documented illness) will be considered and an extension can be negotiated with the instructor for those cases. In those cases, contact the instructor BEFORE the official assignment deadline, if at all possible.