Department of Computer Science and Software Engineering

Software Architecture and Design I

CSSE 374 Winter Term 2010-11

Class Times: 1st Period in Olin 101

Instructor: Shawn Bohner
Email: bohner@rose-hulman.edu
Phone(s): 812-877-8685 (office)   812-877-3708 (home)
Office: Moench Hall, Room F212
Office Hours: MTRF (Most afternoon except when Senior projects are meeting)
Wednesdays: (Most times accept when Senior projects are meeting)
Or by Appointment
Course Assistants/Advisors (for Sections 1 & 2): Tim Ekl, Eric Stokes, and Sam Varga

Course Description: Introduction to the architecture and design of complete software systems, building on components and patterns. Topics include architectural principles and alternatives, design documentation, and relationships between levels of abstraction.

Overview: Designing software is an integral part of engineering a software product. Poorly designed software exposes systems and the organizations they serve to many risks ranging from innocent defects injected during development to more sinister assurance vulnerabilities that might compromise the systems' mission. Effectively designed software ensures a product that evolves with its changing environment, providing the requisite functionality within the expected performance parameters.

Design is both a process and a product – a creative process producing a software artifact that models key aspects of the system. This course presents software design principles in the context of software engineering that enables software engineers to produce and deliver software on-time, on-budget, and on specification. Students will be exposed to software engineering teams, the Unified Modeling Language (UML) and its application to analysis and design, and a range of software design activities and artifacts. Students will participate in a team project to gain an appreciation of the practical aspects of software engineering and an understanding of the interdisciplinary nature of design. Topics to be covered include:

1. Basic principles of Software Design
2. Moving from problem to solution, from *what* to *how*, from analysis to design
3. Problem Domain Modeling
4. Structure and Behavior Modeling
5. Class and Object Design
6. Software Architectures and Styles
7. Gang of four Design Patterns
8. GRASP Principles
9. Architecture & Design Refinement

Learning Outcomes: Upon successfully completing this course, a student should be able to.

1. Work effectively with a team of software project stakeholders, including customers and members of the development team.
2. Demonstrate object-oriented design basics like domain models, class diagrams, and interaction (sequence and communication) diagrams.
3. Recognize the differences between problems and solutions and deal with their interactions.

4. Use fundamental design principles, methods, patterns and strategies in the creation of a software system and its supporting documents.
5. Identify criteria for the design of a software system and select patterns, create frameworks, and partition software to satisfy the inherent trade-offs.
6. Analyze and explain the feasibility and soundness of a software design.

Prerequisites: Pre-requisite with CSSE 371. (*Software experience, and an ability write and communicate effectively will make this course more meaningful.)*
Textbooks: "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3ed)" by Craig Larman; Printice Hall PTR (2004). ISBN 13: 978-0131489066
Readings may also be assigned from other relevant technical publications.

Course Evaluation and Feedback: Please feel free to provide feedback about the course at any time. If you feel uncomfortable talking with me directly, there is an anonymous feedback box under the ANGEL account for this course where you can provide feedback throughout the term; I typically check it once a day and will try respond to feedback in a reasonable time. There will also be two anonymous plus-delta evaluations of the course where you can offer suggestions on how to improve the course and its delivery.

Grading:

Examination(s)	30%	Quizzes	5%
Homework/Case Studies	20%	Team Project Deliverables	35%
Project Mtgs./Participation	10%		

Expectations: Students will be expected to attend and participate in class. Students will be required to use the CSSE374 course website on Angel to obtain relevant information, and interact with instructor and other students. Announcements and assignments will be conveyed via Rose-Hulman email addresses and/or posted on the course website. Students will be expected to work on some assignments with other team members.

Assignments: Homework and junior project assignments will be assigned regularly. Unless otherwise requested, please post these on Angel in the associated drop boxes. Homework and projects are necessary instruments for tracking progress of students. A typical student will work approximately 9 hours outside of lectures each week on this course (depending on background). This is a demanding course covering a great deal of material -- please avoid falling behind on the assignments. While this course is demanding, it is also rewarding for those that want strong understanding of software engineering as a discipline.

Case Studies: We may be doing case studies in class. You should prepare (and submit) a brief write-up of your understanding and opinion of the case study. The write-up should be no more than 1/2 page in length and will be collected in class. If you have an excused absence from class, this can be sent to me via email the day of the class.

Late Submissions: Please note that homework and project deliverables will be due at the specified time on the specified day. Late quizzes will not be accepted. Homework assignments, and project deliverables will also not be accepted late, with the following exception: You have three "late day" credits. You may use one of them on any homework or project assignment, which will allow you to submit that assignment up to 24 hours after the due time. Homework's or project assignments, which are more than 24 hours late, will receive a deduction of at least 10% per late day (or not be accepted at all), depending on the circumstances and the degree of lateness. You may earn a maximum of one additional "late day" by submitting an assignment or a project deliverable 24 hours before the due date. Please send me an email alerting me to the same to obtain the "late day" credit. If you submit something late for which late day credits are allowed, I will assume that you want to use one of your credits unless you tell me otherwise.

Academic Integrity: CSSE Honesty Policy (see http://www.cs.rose-hulman.edu/index.php/courses-mainmenu-28/82-honesty-policy.html) governs class and performance. Joint study is allowed (even encouraged) on some items as expressed by the instructor; however, each student must produce his or her solutions individually. Students must not collaborate on tests or homework that is passed in unless directed by the instructor.

Attendance Policy: Attendance is mandatory (unless with a legitimate excused absence such as illness). If you cannot make it to class or lab, you are still responsible for all materials covered in class as well as all announcements. Up to 2 unexcused absences are permitted. In accordance with the Rose-Hulman attendance policy, additional unexcused absences may result in you receiving a failing grade for the course.

Laptop Policy: You may need to use your laptops during some portion of the class period. Please be sure to bring your laptop, a power brick, and a network cable to class. During class discussion, please do not use your laptops. Laptop use during discussions can be distracting to your classmates, the instructor, and may also keep you from focusing on the material. If you typically use your laptop for note taking, please get permission from the instructor in advance so arrangements can be made.

Writing: Written communication is important in CSSE 374. Remember that a software document has several unique and important characteristics:

1. Technical documents are often the result of group authorship, thus it requires planning and final tweaking.
2. Specificity and organization are more important than flow; hence technical documents are often ordered around lists and tables rather than paragraphs.
3. Documents are often the reader's only source of information on the particular subject or product; hence they must be thorough and complete.
4. Documents are often used to answer specific questions; hence, they should facilitate finding specific pieces of information (navigation).
5. Documentation must bridge from general specifications to particulars of implementation and operation, hence it must make abstract concepts concrete and make concrete facts fit generalized concepts.
6. Documentation can be presented in many forms: online via HTML, MS help files, just plain text, and on paper as reference manuals, tutorial, quick reference guides, etc. It is important to choose the correct medium and even more important to write to fit the medium.

You can always drop by my office or consult with your project manager, if you have any questions regarding your document. We would be happy to look at it and suggest some changes. You should also be aware of the service provided by the Learning Center.