SE 471 Software Architecture (3 units)

Course Description
Introduction to the design and implementation of complete and secure software systems. Software architectures, methodologies, model representations, component-based design, patterns, frameworks, security, architectural principles and alternatives, design documentation, relationships between levels of abstraction, theory and practice of human interface design, creating systems which can evolve, choosing software sources and strategies, prototyping and documenting designs, and employing patterns for reuse. *Prerequisite: SE 451*

Course Materials
Software Architecture in Practice, 3/E, by Len Bass, Paul Clements, and Rick Kazman.

Learning Objectives
Students that successfully complete the course will be able to:

1. Identify, analyze, and apply skills and professional standards in software design.
    - Architectural design (e.g., architectural styles, patterns, and frameworks, architectural trade-offs among various attributes, hardware and systems engineering issues in software architecture, requirements traceability in architecture, service-oriented architectures, architectures for network, mobile, and embedded systems, relationship between product architecture and the structure of development organization and market, etc.)
    - Human-computer interaction design (e.g., general human-computer interaction design principles, use of modes and navigation, coding techniques and visual design, response time and feedback, design modalities, localization and internationalization, human-computer interaction design methods, interface modalities, metaphors and conceptual models, psychology of human-computer interaction, etc.)
    - Detailed design for software (e.g., database design, design of networked and mobile systems, design notations, etc.)
    - Design evaluation of software (e.g., design attributes, design metrics, formal design analysis, etc.)
2. Identify, analyze, and apply skills and professional standards in security issues.
    - Developing secure software (e.g., secure design principles and patterns, etc.)

Course Structure
A major lesson of the course is for students to learn how to provide architectural attributes in their designs, alias quality attributes, alias non-functional attributes. The textbook contains scenarios for this, how to put in place a way of expressing what is needed, which can grow into what is designed and implemented and tested, as is true for functional attributes with use cases.

What is the format of the course?
The course involves three hours of discussion/work activity a week, along with intensive after-class project activities. The goal of every class session is for individual students and teams to be able to apply software architecture requirements, design concepts, and other skills as soon as possible, with growing independence, and to learn from that application.

How are students assessed?

Team Projects – The students will being with a project and modify it with six different architectural studies (iterations) of that system, trying to improve on its quality attributes. These studies provide direct feedback on the difficulties in making such improvements after a system is already partially built. The students also create an accompanying software architecture document from scratch, after the fact, which tests their ability to capture a design (their own!) from its code. They continue with various design activities including application of patterns and frameworks, and make-versus-buy decisions. The six major studies are done in a fashion that brings out the heuristic nature of architectural choices. For example, they test their ability to make changes to their system in a study of its modifiability. Then they try to refactor the system so as to improve that attribute systematically. Finally, they make a different set of changes to see if they actually improved the efficiency of maintenance. Students will present and demonstrate these projects at the end of each of the six exercises. Their final presentation is an overall evaluation of the successes, failures, and challenges of the studies.

Journals – As an integral part of the project, students are expected to keep technical journals in which they record both team and individual activities that were a part of their work. These are intended, in particular, to demonstrate that the students did critical thinking as a part of the project problem solving. Thus, the journals are graded on this basis, and cannot simply log what happened without showing a search for root causes, development of creative ideas, reflection on teaming experiences, and how they made personal contributions to the team. Along with other ways to judge individual contributions on teams, these journals can be used as a subjective means to verify the significance of those contributions.

Homework – These assignments are primarily individual ones such as answering questions at the ends of the chapters and small/mid-size projects to test concept comprehension. The exception is the presentation of an architecture case study, counted as homework, which requires a team of two students to do a full-hour presentation on one of the case histories from Bass's Software Architecture book. This assignment is clearly an application of earlier learning.

Biweekly quizzes – These are four short essay quizzes of approximately 30 minutes duration. All are closed book, done in class. The quizzes test for broad knowledge and application of concepts. A sample question is, "Ch 8 of Bass says that 'the complexity of flight simulators grew exponentially' over a 30 year period. Why was that particularly the case for this application? How would you allow for an application that doubled in size periodically?"

Term paper – The paper allows students to find a small, applied research topic in software architecture and analyze it as a team (of 2). A sample topic is, "Describe where the boundaries are on client-server designs, and what alternative architectural styles take over at those boundary points." Student teams are allowed to come up with their own topics.

Grade Distribution

Team Project	Journal	Homework	Quiz	Term Paper
25%	15%	25%	25%	10%

SCALE:
A, 93 or more; A-, at least 90 but less than 93;
B+, at least 87 but less than 90; B, at least 83 but less than 87; B-, at least 80 but less than 83;
C+, at least 77 but less than 80; C, at least 70 but less than 77;
D, at least 60 but less than 70
F, 0 to less than 60

Course Policies

Campus Writing Requirement
The University Writing Requirement of 2,500 words in this course will be exceeded by your journal, term paper, and homework assignments.

Inform Your Instructor of Any Accommodations Needed
Students with disabilities who require reasonable accommodations must be approved for services by providing appropriate and recent documentation to the Office of Disabled Student Services (DSS). This office is located in Craven Hall 4300, and can be contacted by phone at (760) 750-4905, or TTY (760) 750-4909. Students authorized by DSS to receive reasonable accommodations should meet with me to ensure accommodations are met.

Cougar Care Network (CCN)
Our campus has implemented a new early alert system to support and promote students' academic and personal success. I may refer you through the Cougar Care Network (CCN) to get connected to campus support and resources to assist you.

Civility Campaign
The Civility Campaign, an effort led by the Dean of Students Office, defines civility to reflect the community values of CSUSM. The university strives to be a community demonstrating respect for oneself and for others, treatment of others with dignity, and behaviors which promote a physically and psychologically safe, secure and supportive climate enabling all community members to engage as full and active participants where the free flow of ideas are encouraged and affirmed.

COURSE TOPICS for SE 471

- Key issues in design and architecture fundamentals (general concepts, context, process, principles)
- Architectural structure and viewpoints, such as
    - architectural styles, patterns, and frameworks
    - architectural trade-offs among various attributes
    - hardware and systems engineering issues in software architecture
    - requirements traceability in architecture
    - service-oriented architectures
    - architectures for network, mobile, and embedded systems
    - relationship between product architecture and the structure of development organization and market
- User interface
    - Principles
    - Issues
    - User interaction modalities
    - Information presentation
    - Localization and internationalization
    - Metaphors and conceptual models
    - Psychology of human-computer interaction
    - Coding techniques and visual design
    - Response time and feedback
- Detailed design for software
    - database design
    - design notations
- Developing secure software (e.g., secure design principles and patterns, etc.)
- Creating systems which can evolve
- Documenting designs
- Software design tools
- Reinforce design issues and strategies
    - design concepts
    - process and principles
    - concurrency
    - control and handling of events
    - error and exception handling
    - interaction and presentation
    - data structure centered design
    - aspect-oriented design