COURSE SYLLABUS

Catalogue Description:

Software architecture is an increasingly important industry practice and research area within the software engineering community. Software architecture as a discipline is gaining an ever increasing prominence as software intensive systems have become a critical and indispensable component of most companies and business activities across all industries. As software systems become ever more sophisticated, complex and interconnected, applying well-understood and rigorous software architecture methods, practices and techniques is no longer optional or desirable, but necessary and required for most if not all software implementations. Solid software architecture can be traced to be at the heart of most well-engineered and successful software system.

Software architecture deals with the high level building blocks that represent an underlying software system. These building blocks are the components (units of computation in a system), the connectors (models of the interactions between software components), and the configurations (arrangements of software components and connectors, and the rules that guide their composition). Software architectures that are found particularly useful for families of systems are often codified into architectural styles.

Per the definition by the Software Engineering Institute (SEI) at Carnegie Mellon University: Architecture serves as the blueprint for both the system and the project developing it, defining the work assignments that must be carried out by design and implementation teams. The architecture is the primary carrier of system qualities, such as performance, modifiability, and security, none of which can be achieved without a unifying architectural vision. Architecture is an artifact for early analysis to make sure that the design approach will yield an acceptable system. Architecture holds the key to post-deployment system understanding, maintenance, and mining efforts. In short, architecture is the conceptual glue that holds every phase of the project together for all its many stakeholders.
The course provides the student with a complete treatment of software architecture, its foundation, principles, and elements, including those described above. The class is centered around reading assignments, and homework that will test understanding of the course material. A class project will require the student to leverage the architectural techniques learned during the course (e.g., architectural recovery, architectural styles, domain specific software architectures) to design and implement a real-world software system.

This course will afford the student a complete treatment of software architecture, its foundation, principles, and elements, including those described above. The class is centered around reading assignments, and homework that will test understanding of the course material. A class project will require the student to leverage the architectural techniques learned during the course to reconstruct, assess, evaluate, or design a real-world software system.

In addition to foundations, and practical experience with software architectures, the class will also introduce the student to the state-of-the-art in software architecture research, future trends and state-of-the-practice.

Prerequisites:

Comp 170, Comp 163 and Comp 271. Students will be expected to understand and write programs in Java and similar languages.

Special Course Requirements*:

1. The course may include some assignments done with another person or a team of students. Success or failure of this group work will affect your grade and leaning in the course.
2. The course will use Blackboard to organize materials. Ask for help if you are not familiar with Blackboard (and see http://www.luc.edu/blackboard/).

Course Objectives and Goals

Upon the successful conclusion of the course, the student will be able to:

1. Understand the theory and motivation of modern approaches to software architecture
2. Analyze, assess, and evaluate the appropriateness of the software architecture of a software intensive system; assess system conformance to the software architecture
3. Elicit, capture, and formally document architectural requirements and quality attributes
4. Analyze, define and document the appropriate software architecture for a software intensive system based on architectural requirements, constraints, and quality attributes
5. Incorporate cost benefit analysis methodology, as well as financial, organizational, and business context considerations in analyzing and defining software architectures
6. Use modern tools to support and implement software architecture.
**Textbooks**

**Required:**

1. “Software Architecture: Foundations, Theory, and Practice”  
   Richard N. Taylor, University of California, Irvine  
   Nenad Medvidovic, University of Southern California  
   Eric Dashofy, Aerospace Corporation  
   Wiley Publishing ©2009

2. “Software Architecture in Practice”  
   Len Bass, Paul Clements, Rick Kazman  
   SEI Series in Software Engineering, Carnegie Mellon University  
   Addison-Wesley Professional ©2003

**Recommended Additional Reading:**

3. “Software Architecture: Perspectives on an Emerging Discipline”  
   Mary Shaw, Carnegie Mellon University  
   David Garlan, Carnegie Mellon University  
   Prentice-Hall ©1996

4. “Essential Software Architecture”  
   Ian Gorton  
   Springer ©2006

   Anthony J. Lattanze, Carnegie Mellon University  
   Auerbach Publications ©2008

   Nick Rozanski, Eoin Woods  
   Addison-Wesley Professional ©2005

7. “A Software Architecture Primer”  
   John Reekie, Rohan McAdam  
   Angophora Press ©2006
8. “Software Architecture and Design Illuminated”  
   Kai Qian, PhD, Southern Polytechnic State University  
   Xiang Fu, Hofstra University  
   Lixin Tao, Pace University  
   Chong-wei Xu, Kennesaw State University  
   Jorge Diaz-Herrera, Rochester Institute of Technology  
   Jones & Bartlett Publishers ©2009

The instructor will provide additional reading materials, papers and handouts.

Assignments and Projects
There will be a required homework assigned at the end of each class session that will be due at the beginning of the following class. Late homework will not be accepted.

Class Participation and Attendance
Attendance will be mandatory for all class sessions. Class attendance and participation will contribute to your final grade.

Course Grading
Your grade will consist of these components with relative weights as follows (I reserve the right to adjust the percentages in your favor if circumstances warrant). See in addition the sections on Timely Completion and Academic Honesty.
   1. Individual Homework Assignments and Projects (10%). Homework assignments, including individual projects.
   2. Participation (5%). All students are expected to attend all class sessions in person for the full time period. The participation grade will be based upon attendance and contributions to discussions in class, and attendance and full, enthusiastic involvement in projects and assignments. Attendance will be taken in class.
   3. Team Assignments and Software Architecture Projects (20%). Software architecture analysis, evaluation, design, and documentation assignments for group projects.
   4. Course Presentations and Reports (15%). Students will be required to make presentations and reports in class on their work. Details in the class schedule. Grade will depend on both technical content and quality of the presentation and/or written report.
   5. Exams (50%).

For team or group assignments the final grade for each student will be calculated using the Individual Contribution Index (ICI) method described in a separate document. If for any reason you do miss a class session, it is your responsibility to determine what you missed, locate any handouts, determine any changes in assignments, course plans, or schedules, etc. It is not my obligation to help you make up for missing class.
I will not always be covering items from the web or book in class; additional materials will be added and additional guidance will be given. Information and activities in class that are not in the book will be on exams, quizzes, and helpful for your assignments and programming projects.

The course will be graded on a curve based on class performance. There is no set mapping from number scores to the letter grade.

Please do not ask for “extra credit” to improve your grade as this is neither practical in the course nor fair to your fellow students. I will be happy to discuss your performance in the course with you at any time, including discussing possible grade based on performance to date and ways to improve your performance during the remainder of the course.

Individual Homework Assignments and Projects: 10%
Participation: 5%
Team Assignments and Projects: 20%
Course Presentation and Report: 15%
Exams: 50%
   Quiz 1: (TBD)  5%
   Midterm: (TBD) 20%
   Final exam: (TBD) 25%

**Academic Integrity**

Students are expected to have read the statement on academic integrity available at http://www.luc.edu/academics/catalog/undergrad/reg_academicintegrity.shtml. This policy applies to the course. The minimum penalty for academic dishonesty is a grade of F for that assignment. Multiple instances or a single severe instance on a major exam or assignment may result in a grade of F for the course. All cases of academic dishonesty will be reported to the department office and the relevant college office where they will be placed in your school record.

Academic dishonesty includes, but is not limited to, working together on assignments that are not group assignments, copying or sharing assignments or exam information with other students except in group assignments, submitting as your own information from current or former students of this course, copying information from anywhere on the web and submitting it as your own work, and submitting anything as your own work which you have not personally created for this course. If you do wish to use materials that are not your own, please check with me ahead of time and cite your source clearly. When in doubt, ask first!

**Timely Completion**

The student is expected to complete all assignments, readings, and projects on time. In computer systems in the “real world”, there is always strong emphasis on getting projects done on time. Use class to develop your own skills at timely completion.

Personal and group projects and other assignments will be due as described at time of assignment. See the class schedule for advanced planning.

Late assignment submission is discouraged. Late projects and other assignments will be accepted only if you have a written agreement with me at least 24 hours before the assignment is due. (This possibility will be reserved for rare and cases of extreme need).
You are welcome to ask questions on all assignments and course work, seek additional information on the assignments, and offer observations on the assignments to me either in or outside of class. To discourage procrastination, no questions on the assignment will be answers on the date the assignment is due – please plan your work ahead and do not wait for the last minute to begin work!

**Class Schedule by Week** (schedule will be updated until the beginning of the course)

**Week 1:** September 4, 2010 – No Class - Labor Day weekend

No Class - Labor Day weekend

**Week 2:** September 11, 2010

Course Overview and Administrative Clarifications
The Big Idea, Architecture in Context
“Software Architecture: Foundations, Theory, and Practice” – Chapters 1, 2
“Software Architecture in Practice” – Chapter 1

**Week 3:** September 18, 2010

Basic Concepts, Introduction to Design
“Software Architecture: Foundations, Theory, and Practice” – Chapters 3, 4 (begin)
“Software Architecture in Practice” – Chapter 2

**Week 4:** September 25, 2010

Architectural Styles, Styles and Greenfield Design
“Software Architecture: Foundations, Theory, and Practice” – Chapter 4 (continue and finish)
“Software Architecture in Practice” – Chapters 2, 7

**Week 5:** October 02, 2010

Software Connectors, Software Connectors in Practice
“Software Architecture: Foundations, Theory, and Practice” – Chapter 5
“Software Architecture in Practice” – Chapter 7

**Week 6:** October 09, 2010 - Quiz

Introduction to Modeling, Modeling
“Software Architecture: Foundations, Theory, and Practice” – Chapter 6

**Week 7:** October 16, 2010

Modeling and Notations
“Software Architecture: Foundations, Theory, and Practice” – Chapter 6
**Week 8:** October 23, 2010 – Midterm Exam

Analysis of Software Architectures
“Software Architecture: Foundations, Theory, and Practice” – Chapter 8

**Week 9:** October 30, 2010

Architectural Analysis Techniques
Architectural Tradeoff Analysis Method (ATAM)
“Software Architecture: Foundations, Theory, and Practice” – Chapter 8
“Software Architecture in Practice” – Chapter 11

**Week 10:** November 06, 2010

Implementing Architectures, Implementation
Software Product Lines
“Software Architecture: Foundations, Theory, and Practice” – Chapter 9
“Software Architecture in Practice” – Chapter 14

**Week 11:** November 13, 2010

Deployment and Mobility, Applied Architectures
“Software Architecture: Foundations, Theory, and Practice” – Chapter 10, 11

**Week 12:** November 20, 2010

Designing for NFP, Quality Attributes
“Software Architecture: Foundations, Theory, and Practice” – Chapter 12, 13
“Software Architecture in Practice” – Chapters 4, 5

**Week 13:** November 27, 2010 – Thanksgiving Break – No Classes

Thanksgiving Break – No Classes

**Week 14:** December 04, 2010

Documenting Software Architectures
Economics of Software Architecture and Engineering, Cost Benefit Analysis Method (CBAM)
“Software Architecture in Practice” – Chapters 9
“Software Architecture in Practice” – Chapter 12

**Week 15:** December 11, 2010 – Final Project – Team Presentations
Final Project – Team Presentations

**Week 16:** December 18, 2010 – Final Exam

Final Exam

_Last update: April 25, 2010_