Course Overview

Successful design of complex software systems requires the ability to describe, evaluate, and create systems at an architectural level of abstraction. This course introduces architectural design of complex software systems. The course considers commonly-used software system structures, techniques for designing and implementing these structures, models and formal notations for characterizing and reasoning about architectures, tools for generating specific instances of an architecture, and case studies of actual system architectures. It teaches the skills and background you need to evaluate the architectures of existing systems and to design new systems in principled ways using well-founded architectural paradigms.

Course Objectives

Successful completion of this course will provide you with the knowledge and skills to

- Understand the influence of architectural drivers on software structures.
- Understand the technical, organizational, and business role of software architecture.
- Recognize major architectural structures (styles, patterns, tactics, etc.).
- Generate architectural alternatives for a problem and choose among them.
- Understand the principles of good architectural documentation.
- Understand the affect of COTS on architectural designs.
- Understand how formal notations can be used to specify architectures.
- Evaluate the fitness of an architectural design in meeting a set of system requirements and balancing quality trade-offs.
- Be aware of future trends in software architecture.

Prerequisites

- Experience with at least one large software system, which condition could be satisfied by industrial software development experience or an undergraduate course in software engineering, compilers, or operating systems.
- Facility with a high-level, structured language, such as C++, and familiarity with object-oriented concepts and notations. The course will use Java.
- The ability to produce web pages as part of final project presentations.

Organization

The course consists of a series of lectures conducted by faculty and staff from the School of Computer Science at Carnegie Mellon University. The lectures, captured on DVDs, lead the way through a series of assignments and chatroom discussions.

Your best approach to successfully complete the course is to follow three simple steps.
1. Do the assigned readings.
2. Watch the lecture.
3. Complete the assignments.

All course materials, with the exception of the texts and DVDs, will be available on the Carnegie Mellon Blackboard System. Email with specific information about the course Blackboard site will be sent to you prior to the start of class.

Feedback and Support

Office hours and computer conferences will take place in the Virtual Classroom on the course Blackboard site. Your instructor for the course will conduct course discussions in the Virtual Classroom every week, specific time to be determined. Should your instructor decide to use another chat tool, they will notify you at the beginning of the course. Other times, your instructor will be available by email. In addition, you should feel free to post questions and comments on the course electronic bulletin board at anytime to discuss the readings, the course, and issues related to software engineering with members of your class.

Readings

Weekly readings are used to stimulate discussion and as a way to expose you to course topics not covered directly in the lectures. For most lectures, you will be assigned a few readings to complete before watching the lecture. The course also has a number of optional reference materials which are listed in the bibliography.

The required textbooks for this course are *Software Architectures: Perspectives on an Emerging Discipline* by Mary Shaw and David Garlan, *Documenting Software Architectures: Views and Beyond* by Paul Clements et al., and *Software Architecture in Practice* by Len Bass, Paul Clements, and Rick Kazman. In addition, there are two recommended texts: *Documenting Software Architectures: Views and Beyond* by Paul Clements and *Architecting Software Intensive Systems: A Practitioner’s Guide* by Anthony Lattanze.

Evaluation

The course grade will be determined by four factors.

- **Reading Questions:** Each lecture has a set of questions designed to help you focus your thoughts on the main points of the readings. Each question should be addressed in less than a page (often a single paragraph will do). Your instructor may have you submit reading questions for two lectures at once. However, it is in your best interest to complete the reading questions prior to watching the lecture.

- **Assignments:** There are four assignments. The first three are small system architecting exercises. Each of these assignments will have a group and an individual section. The purpose of these assignments is to give you some experience using architectures to design and implement real systems. To help clarify your designs, your instructor will hold a brief, ungraded design review for most assignments during a chat session a week before the assignment is due. Groups will take turns presenting their preliminary designs and getting feedback from class members and the instructor. The fourth assignment is designed to give your team a chance to analyze an established or emerging industrial architectural standard.

- **Final Project:** The course project is designed to give you experience with the architecture of a software system. You will design and analyze the architecture of a system, document your work, and present the results to the rest of the class. Projects will be drawn from various sources, as well as the MSE studio. Your grade will depend on both the quality of your design, analysis, and documentation as well as on its presentation.
**Instructor judgement:** The instructor reserves the right to raise or lower your quantitatively determined grade based on their judgement of your mastery of course material; this judgement will be based in part on your ability to participate constructively in class discussions.

### Schedule

<table>
<thead>
<tr>
<th>Lecture</th>
<th>General Topic</th>
<th>Subtopic</th>
<th>Readings</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>Overview and Software Architecture Defined</td>
<td>[Shaw 96] Ch 1; [Bass 03] Ch 1, Ch 2 (skim); [Lattanze 08] Ch 3</td>
<td>Questionnaire</td>
</tr>
<tr>
<td>2</td>
<td>Architectural Drivers</td>
<td></td>
<td>[Shaw 96] Ch 2; [Lattanze 08] Ch 4</td>
<td>RQ1</td>
</tr>
<tr>
<td>3</td>
<td>Structures</td>
<td>Structures and Views</td>
<td>[Shaw 96] Ch 2; [Lattanze 08] Ch 4</td>
<td>RQ1</td>
</tr>
<tr>
<td>4</td>
<td>Dataflow Systems</td>
<td>Dataflow Styles</td>
<td>[Shaw 96] § 3.2, 4.2.1, 4.3.1</td>
<td>RQ2</td>
</tr>
<tr>
<td>5</td>
<td>Documentation 1</td>
<td></td>
<td>[Clements 03] Prologue, Ch. 1, 2, 3, 5 (skim)</td>
<td>RQ3</td>
</tr>
<tr>
<td>6</td>
<td>Case Study</td>
<td></td>
<td>[Lattanze 99]; [Shaw 96] § 3.1, 3.1.4</td>
<td>RQ3</td>
</tr>
<tr>
<td>7</td>
<td>Call-Return Systems</td>
<td>Call-Return Styles</td>
<td>[Parnas 72]; [Clements 03] § 2.4; [Shaw 96] § 3.1; [Dvorak 02]</td>
<td>A1</td>
</tr>
<tr>
<td>8</td>
<td>Client-Server and Tiered Architecture</td>
<td></td>
<td>[STR 00]</td>
<td>Acme Lab; RQ4</td>
</tr>
<tr>
<td>9</td>
<td>Middleware</td>
<td></td>
<td>[Vinoski 02]</td>
<td>RQ5</td>
</tr>
<tr>
<td>10</td>
<td>Event Based Systems</td>
<td>Event Styles</td>
<td>[Shaw 96] § 2.4, 2.6, 2.7, 7.3</td>
<td>RQ5</td>
</tr>
<tr>
<td>11</td>
<td>Shared Info Systems</td>
<td>Shared Information Styles</td>
<td>[Nii 86]; [Sumner 04]; [eXtropia 09]</td>
<td>RQ6</td>
</tr>
<tr>
<td>12</td>
<td>Techniques and Methods</td>
<td>Guidance for the Architect</td>
<td></td>
<td>RQ6</td>
</tr>
<tr>
<td>13</td>
<td>Design by Selection</td>
<td></td>
<td></td>
<td>Final Project Descriptions; A2</td>
</tr>
<tr>
<td>14</td>
<td>Architectural Evaluation</td>
<td></td>
<td>[Bass 03] Ch 11; [Jones 01]</td>
<td>RQ7</td>
</tr>
<tr>
<td>15</td>
<td>Strategic Reuse: Product Lines</td>
<td></td>
<td>[Bass 03] Ch 14 &amp; 15</td>
<td>RQ7</td>
</tr>
<tr>
<td>16</td>
<td>Product Lines Case Studies</td>
<td></td>
<td>[Shaw 96] § 3.2; [Garlan 05]</td>
<td>RQ8</td>
</tr>
<tr>
<td>19</td>
<td>ACDM</td>
<td></td>
<td>[Lattanze 07]</td>
<td>RQ10</td>
</tr>
<tr>
<td>20</td>
<td>Architecture Conformance</td>
<td></td>
<td>[Schmerl 06]; [Bass 03] Ch 10</td>
<td>RQ10</td>
</tr>
</tbody>
</table>
Architecting for X | Performance | [Sha 90]; [Muller 10]  
---|---|---  
Availability | [Sha 95] | RQ11  
SOA, Web Services | [Papazoglou 08]; [Bagwell 07] | A4  
Security | [Benzel 05] | RQ12  
Advanced Topics | Organizational Alignment |  
Cloud Computing | [Erdogmus 09] | RQ13  
Research Directions | [Garlan 02]; [Garlan 07]; [Garlan 09] | Final Project Reports  

Bibliography


[Clements 03] Clements, Paul; Bachmann, Felix; Bass, Len; Garlan, David; Ivers, James; Little, Reed; Nord, Robert; & Stafford, Judith A. Documenting Software Architectures: Views and Beyond. Reading, MA: Addison-Wesley, 2003.


