

Syllabus: OMSE 532

Software Architecture and Domain Analysis

Spring 2006

Stuart Faulk
Dept. of Computer and Information Science
120 Deschutes Hall
University of Oregon
(541) 346-1350
faulk@cs.uoregon.edu

This syllabus is a preliminary draft based on a prior edition of this course. It can be expected to change as I better understand the background and needs of the current students.

Catalog Description

Methods and principles of the architectural design of complex, large-scale software systems to accommodate change and evolution through many product releases or versions. Survey of the major architectural styles, their strengths and weaknesses, and architectural trade-offs with respect to system goals and desired properties. Study of architectural approach to development of open systems and frameworks based on case studies. Software engineering of domain-specific software architectures for families of systems (e.g., product lines) including domain analysis, domain modeling, and design of domain specific software architectures. Relation of software architecture to requirements and its effects on downstream design and software evolution. Students examine domain analysis and the architectural design process and products in the business context including the effects of decisions on cost and schedule.

Rationale

The architectural design of complex, long-lived systems or families of systems is fundamental to efficient, reliable software production, and fundamentally different from small-grain design issues. The field of "software architecture" continues development that can be traced back at least to DeRemer & Kron's "Programming in the Large vs. Programming in the Small," Parnas's "On the Criteria for Decomposing Programs into Modules" and related papers. Today there is wide recognition of the opportunity for exploiting architectural principles to address a variety of software engineering issues including development of common platforms, reduced cycle time, improved maintainability, in decreased production cost.

Objectives

On completion of the architecture and design course, students shall understand principles of software architecture and their application, and shall:

- Understand the practical application of architectural concepts in software engineering.
- Develop a clear understanding of which concerns properly belong to software architecture (and which do not).
- Understand principles of architectural design by applying them on a small scale.
- Become familiar with common architectural styles: understand what distinguishes one style from another and the tradeoffs involved in using a particular style.
- Understand how to evaluate existing or proposed architecture for suitability to task
- Understand how to apply architectural design principles to support evolutionary system development (systematic reuse, lower-cost, faster time-to-market, higher quality).

Students finishing this course are not expected to immediately become lead software architects, but will have background knowledge and skills that will prepare them to acquire architectural expertise in a particular application domain through work experience.

Required Text

Software Architecture in Practice (Second Edition), by Len Bass, Paul Clements, and Rick Kazman. Addison-Wesley (SEI series in software engineering), 2003. ISBN 0321154959

Additional Readings

Readings are also listed below in the week-by-week class schedule. The instructors will, provide copies of the required papers at class. Small changes to the reading list are likely.

- Britton, K., and D. Parnas, *A-7E Software Module Guide*, NRL Memorandum Report 4702, U.S. Naval Research Laboratory, December 8, 1981.
- Clements, P. C., *Function Specifications for the A-7E Function Driver Module*, NRL Memorandum Report 4658, U.S. Naval Research Laboratory, November 27, 1981.
- Campbell, O'Connor, Mansour, and Turner-Harris, "Reuse in Command and Control Systems": IEEE Software, Sept. 1994.
- S. Faulk, "Software Requirements: A Tutorial," from *Software Engineering*, M. Dorfman and R.H. Thayer, eds., IEEE Press, 1996
- Kazman, R., M. Barbacci, M. Klein, S. Carriere, and S. Woods, "Experience with Performing Architecture Tradeoff Analysis," ICSE 21, Los Angeles, CA.
- D. Parnas, "On the Criteria for Decomposing Systems into Modules," CACM 15(12):1053-1058, 1972.

- D. Parnas, "On a 'Buzzword': Hierarchical Structure," Proceedings of the IFIP Congress, 74:336-390, 1974.
- D. Parnas, "Designing Software for Ease of Extension and Contraction," IEEE TSE, SE-5(2):128-137, 1979.
- D. Parnas and D. Weiss, "Active Design Reviews", Proceedings of the Eighth International Conference on Software Engineering, August 28-30, 1985, London, UK, p. 132-136.
- Shaw, M., "Comparing Architectural Design Styles," *IEEE Software*, Nov. 1995, p. 27-41
- Weiss, D., "Defining Families: The Commonality Analysis," unpublished manuscript.

Topic Outline

All readings are listed in the week they are due and should be completed before the class session unless otherwise noted. Assignments are due the following week.

Week 1 – Course Introduction and Overview

Overview of OMSE program, course content, and grading. Class discussions on “what is architecture” and why is its design a significant issue.

- Administrivia
- Course context and goals
- The Architectural Business Cycle
- What is “software architecture” – class discussion
- What and who are affected by architectural design decisions
- A ‘bird’s-eye view’ of where the course is going

Assignment: Exercise 1: Understanding Parnas in architectural terms

Readings:

- Chapters 1, 2 of Bass et al
- S. Faulk, "Software Requirements: A Tutorial" (review)

Week 2 - Design at the Architectural Level

Discussion of architectural styles and reference models (Ch. 2). Discussion of what it means to do design at an architectural level: i.e., “What are we trying to accomplish in designing an architecture?” What are the input and outputs of the design phase? What is the (idealized) process?

- The Bass, et al. definition of architecture
- Terminology: architectural styles and models
- What does it mean to “design and architecture?” – i.e., what decisions should be made during architectural design
- Parnas papers: software engineering origins of architectural design

- Key architectural concepts: Abstraction and Information Hiding – what they mean, how they differ, and when to use each

Assignment: The Spam Filter Architecture: Part 1

Readings:

- D. Parnas, "On the Criteria for Decomposing Systems into Modules,
- D. Parnas, "Designing Software for Ease of Extension and Contraction,"

Week 3 – A Case Study in Software Architecture – the A-7E Operational Flight Program

Overview of the A-7E case study from Chapter 3 of the text and class handouts.

Discusses each of the system's three primary architectures in terms of the elements of architectural design.

- Why the A-7E is worth studying.
- One system, many useful views of architecture: the Module Structure, Uses Structure, and Process Structure
- For each structure: discussion of the design goals, components and relations used, and types of decisions represented in the structure.
- Class discussion: Spam Filter Architecture: Part 1

Assignment: The Spam Filter Architecture: Part 2

Readings

- A7E case study (Chapter 3 from Bass et al)
- A-7E Software Module Guide (handout)
- Function Specifications for the A-7E Function Driver Module (handout)

Week 4 – Architectural Quality Attributes

Students will present their solutions to the Spam Filter Architecture problem in class for open discussion. Lecture and discussion on architectural quality attributes (Chapters 4-5 of text) and Lampson paper.

- Presentation and discussion of Spam Filter Architectures
- Quality attributes in practice

Assignment: A-7E/SAAM Exercise

Readings:

- Chapters 4, 5 of Bass et al

Week 5 – Evaluating Architecture

Issues in and methods for evaluating software architectures to determine “goodness” relative to requirements and design goals. Class lecture and discussion of the strengths and weaknesses of the architectural evaluation method presented in the text (SAAM) followed by discussion of how we can alter the SAAM method to get more precise and

accurate results. Lecture on an effective peer-review method appropriate for architectural issues (among others) – Active Design Reviews.

- Software Architecture Evaluation Method (SAAM)
- Variations on SAAM – can we make the process more precise and accurate?
- Active Design Reviews – an alternative to traditional (e.g. Fagan) inspections

Assignment: **Take-home Midterm Exam**

Readings:

- Chapter 9 of Bass et al
- Kazman, R., M. Barbacci, M. Klein, S. Carriere, and S. Woods, “Experience with Performing Architecture Tradeoff Analysis”
- Parnas, D. and D. Weiss, “Active Design Reviews”
-

Week 6 - Alternative Views of Software Architecture

A look at some other views of Software Architecture (i.e., differing from that given in the text). We examine what other people might mean by “architecture,” and how these relate to the view discussed in class.

- Review of Midterm
- Lecture/discussion on some other views of software architecture

Readings:

- Shaw, M., “Comparing Architectural Design Styles,” *IEEE Software*, Nov. 1995, p. 27-41
- Garlan, “Architectural Styles, Design Patterns, and Objects”
-

Week 7 – Open Architectures and Formal Representations

A look at architecture as standards or common platforms facilitating open development. What properties distinguish successful architectures from unsuccessful ones?

- Open Systems
- World Wide Web as Architecture

Reading:

- Chapter 7 of Bass et al (WWW case study) and 8 (CORBA)
- Garlan, “Architectural Mismatch”
-
-

Week 8 – Architectures for Families of Systems

Class lecture and discussion on the concept and use of program families. We discuss why viewing programs as families is important to developing architectures with certain critical properties including maintainability, ease of change, and robustness.

- What is a program family?

- Why develop programs as families?
- The role of architecture in developing a family of systems.
- Designing program families.
- Commonality Analysis – a process for analyzing and characterizing a program family.

Assignment: Commonalty analysis for FWSOS Project

Reading

- D. Parnas, "On the Design and Development of Program Families," IEEE TSE SE-2(1):1-9, 1976.
- Weiss, D., "Defining Families: The Commonality Analysis"

Weeks 9 – Architectural Approach to Software Product Lines

Class lecture and discussion of software product lines and the role of architecture in developing product lines. Approach to taking a strategic view of software development. We close the loop of the Architectural Business Cycle by showing the relationship between strategic business goals and software product lines.

- What is a software product-line?
- The Domain Engineering process and its products
- The Application Engineering Process and its products
- Product-line case studies: Celsius Tech and Rockwell Command and Control
- Discussion of Commonalty Analysis Exercise

Assignment: Module Guide for FWSOS

Reading

- Chapter 16: Celsius Tech
- Campbell, O'Connor, Mansour, and Turner-Harris, "Reuse in Command and Control Systems": IEEE Software, Sept. 1994.

Week 10 – Review

Students will present the results of their design for the FWSOS project. Lecture will review course material to provide oversight and provide review for the take home final exam.

- Project Reviews and presentations.
- Course Review for final

Assignment:

- Take home final

Summary of Assignments

Students should come to class prepared to discuss the readings assigned for that class. Students may be asked, with no prior warning, to answer questions about or lead a discussion of any of the assigned readings.

Students must also complete a number of small exercises. An assignment will exercise will typically be given out during one class session and due one or two weeks later.

In the second half of the course, students will participate in adding to a software architecture for a small system. Students go through the steps of extending a domain, analyzing commonality, and applying principles of architectural design to extend an existing architecture to encompass the evolving product domain.

A midterm examination will be given in approximately week 5. It will be a take-home examination, due one week after it is handed out. A final examination, also take-home, will be given the last week of class.

Resource requirements

Students are expected to have access to the internet, including electronic mail and the worldwide web. Students should read electronic mail regularly. Some course materials will be distributed using electronic mail.