Design Patterns 2014

Assessment Structure

Assessment Breakdown

- 50% Continuous Assessment
- 50% Final Examination

Examination

- 2 Hour Duration
- Problem Oriented
- Given a problem specification - develop narrative for outline solution.
    - Narrative to include
        - Structure / design
        - Patterns
        - Psuedocode
        - Commentary

Single Request

Below are are four short descriptions of problems in specific contexts. Two solutions are required for each problem - the second an elaboration on the first. Each solution is to be structured as follows:

Briefly outline of the relevant design pattern(s), with a focus on outlining the responsibilities associated with specific roles in the selected pattern.

5 Marks

A solution to Version 1 expressed as a class diagram and with a high level sketch of the classes involved. Methods in the class can be expressed in any suitable pseudocode.

10 Marks

A Solution to Version 2, expressed as modifications to Version 1 with further classes and pseudocode if necessary.

10 Marks

A short summary of the benefits of adopting the selected patterns.

8 Marks

Select *any three* of the problems and propose outline solutions. All problems are awarded equal marks.

Example Problem 1

As part of an XML messaging processing system, a class ProcessMessage has been put in place to handle incoming messages. It has a method filter(...) which will be passed each message as it arrives. Among other tasks (logging, security checking etc...), the filter() method is also to identify any element in the messages with the <extension> tag. When these are encountered this part of the message it to be passed to another class for processing - called the ExtensionProcessor. There may be multiple <extension> elements in a document.

- Version 1 of the design should enable just one ExtensionProcessor to be installed. All <extension> elements are to be passed to this class.

- Version 2 should support multiple ExtensionProcessors. The <extension> should be offered to each processor in some well define sequence. If the <extension> can be processed by one of the processors, then it is considered “consumed” and the next <extension> should be processed.

Example Problem 2

An email client is to be designed. It is to have a graphical user interface and provide standard email functionality. The client user interface is to support account settings, inbox browsing, email reading and email composition. In the proposed design interface to the email server is encapsulated in a well structured model and is to be developed independent of the user interface.

- Version 1:. Propose a simple user interface independent model to encapsulate the email system. Use Separated Presentation & Flow Synchronization as the core patterns for the application design.

- Version 2: Enhance the design to use Model View Presenter + Observer Synchronization patterns.

Example Problem 3

A bug tracking application is to be developed to support the automated management of bugs in a software project. Among the features to be implemented is a core set of bug management commands: a bug can be entered, deleted, allocated to a particular subsystem, its status updated (pending, fixed, not reproduced) and its priority level (1,2 or 3) adjusted. Each of these commands can be entered on a console or driven through a graphical user interface.

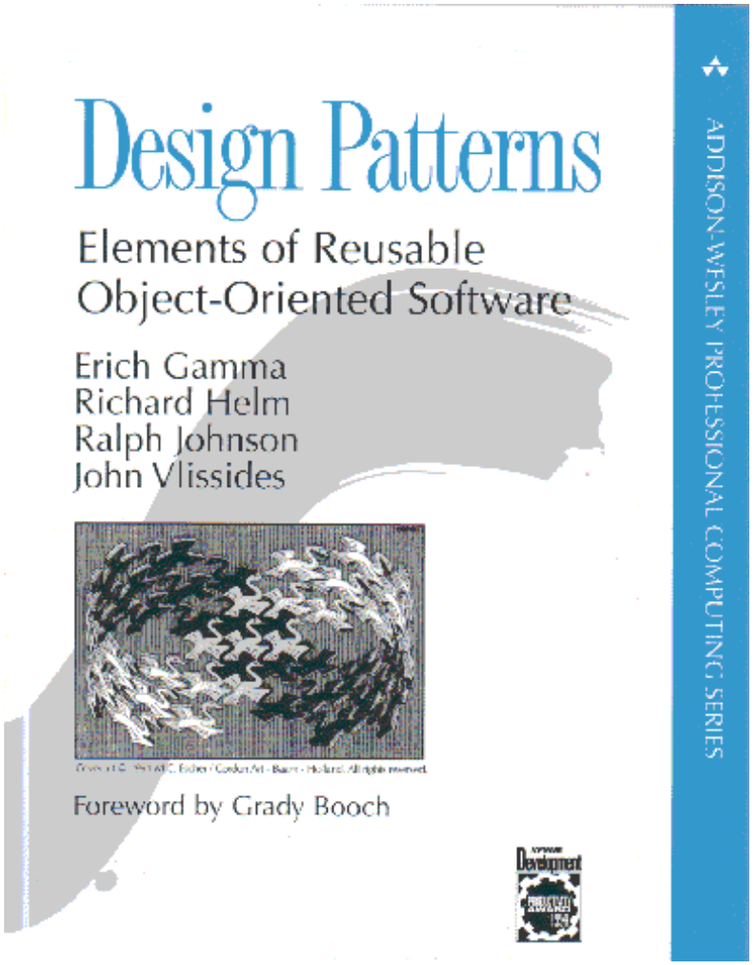
- Version 1 should support the commands as specified and ensure appropriate encapsulation of the command execution and an elementary command logging facility.

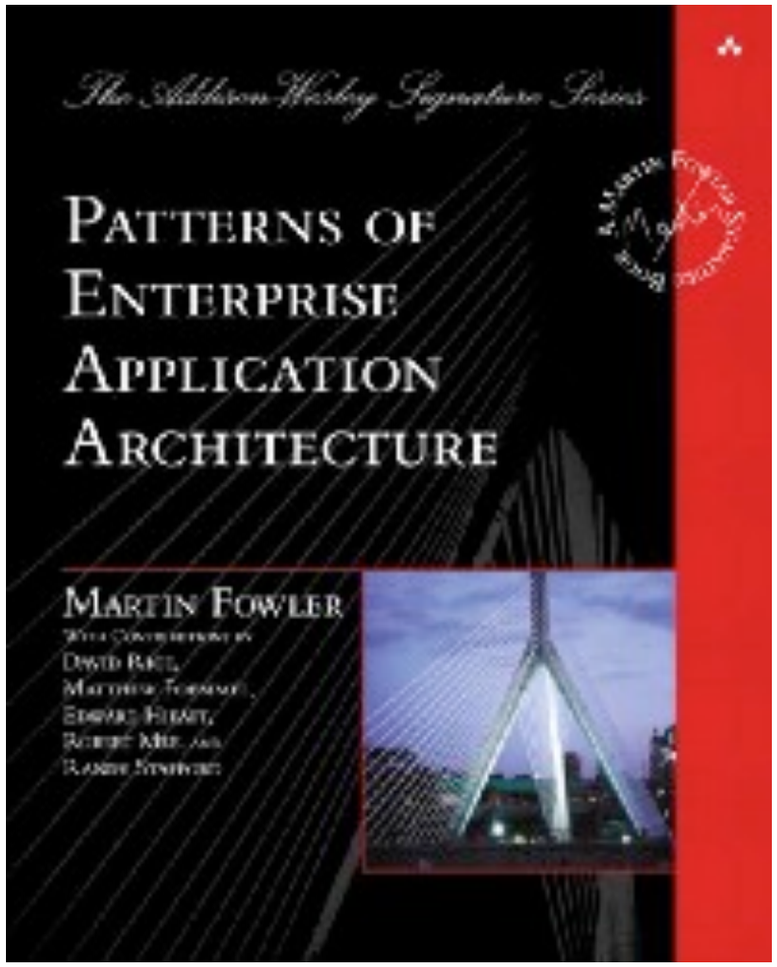
- Version 2 should support an generic undo/redo capability for selected commands.

Continuous Assessment

- 2 Elements
  - A Programming Assignment
  - A “Patterns Companion” document for the project
- Submit final version at end of semester

Key Sources

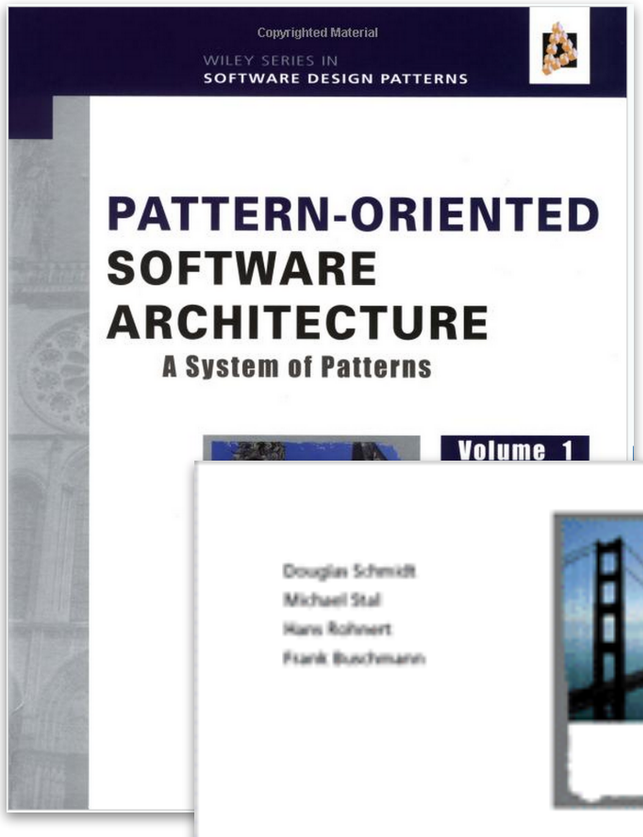


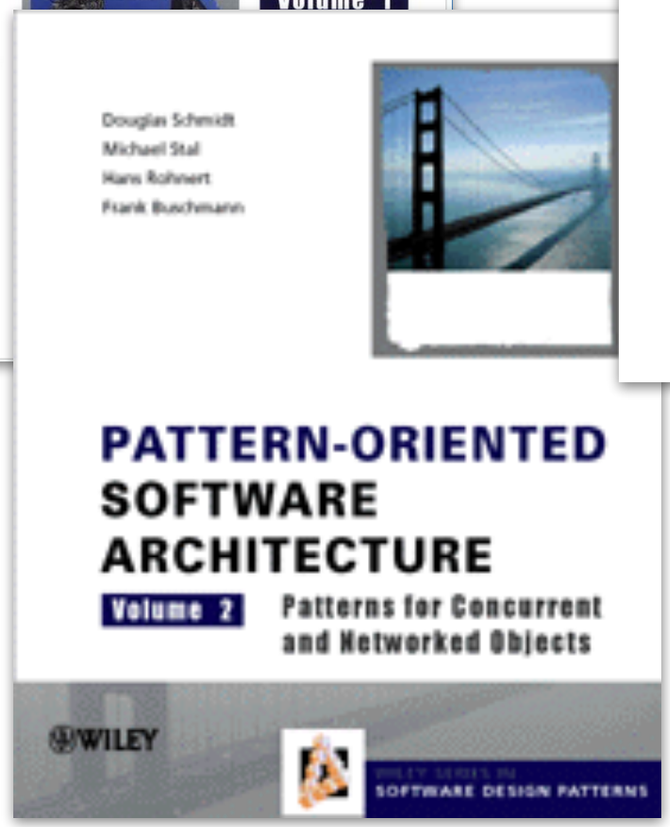


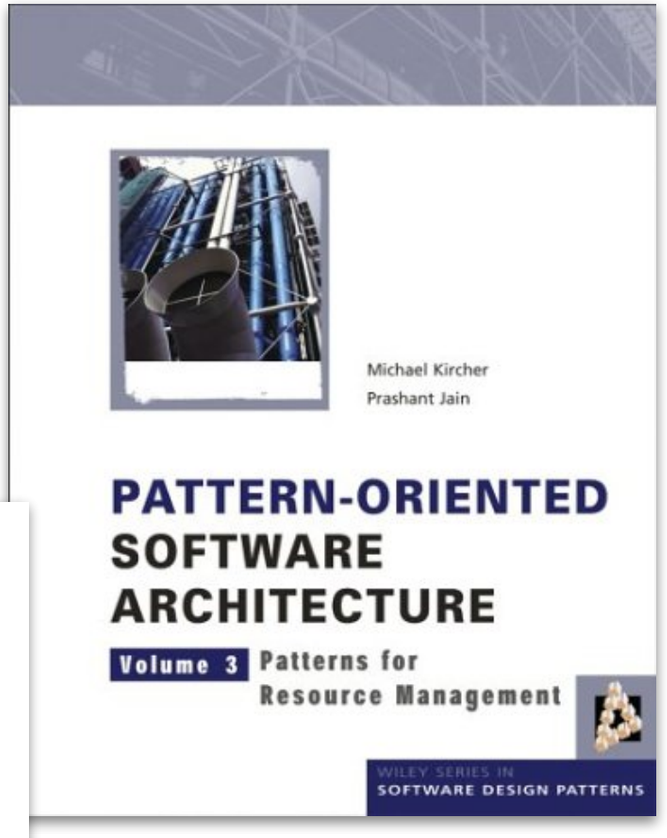
- GOF
- Most typically illustrated within GUI applications (Android, IOS, Native)

- PEAA
- Most typically illustrated within Web Applications (

Other sources - POSA - 5 Volumes!







Other Sources - Domain Driven Design