Design Patterns 2016

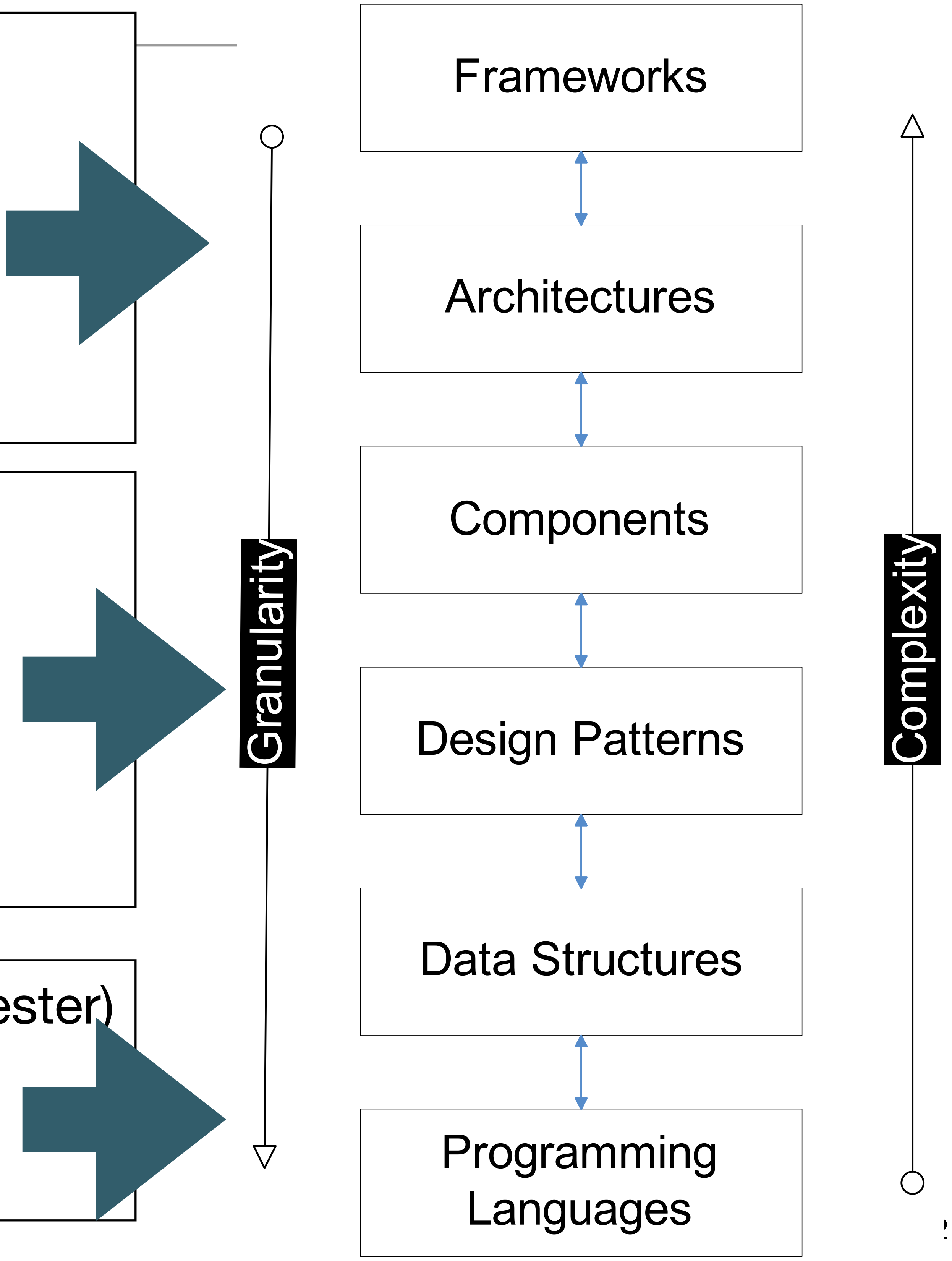
Assessment Structure

Design Patterns Course Structure

- Part 2: Software Architectures
  - Weeks 6-12
  - Pierre Peclier

- Part 1: Classic Design Patterns
  - Weeks 1-6
  - Eamonn de Leastar

- Agile Software Development (last semester)
  - Dr. Siobhan Drohan



Assessment Breakdown

- 50% Final Examination
- 50% Continuous Assessment

Examination

- 2 Hour Duration, 2 Parts
- Part 1 : Problem Oriented
  - Given a problem specification - develop narrative for outline solution(s).
  - Narrative to include
    - Structure / design
    - Patterns, Psuedocode, Commentary
- Part 2: Discursive
  - Exploration of Architectures, styles, notations and case studies
  - Focus on forces, trends and large scale enterprise architectural decision making and tradeoffs.

Continuous Assessment

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

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.

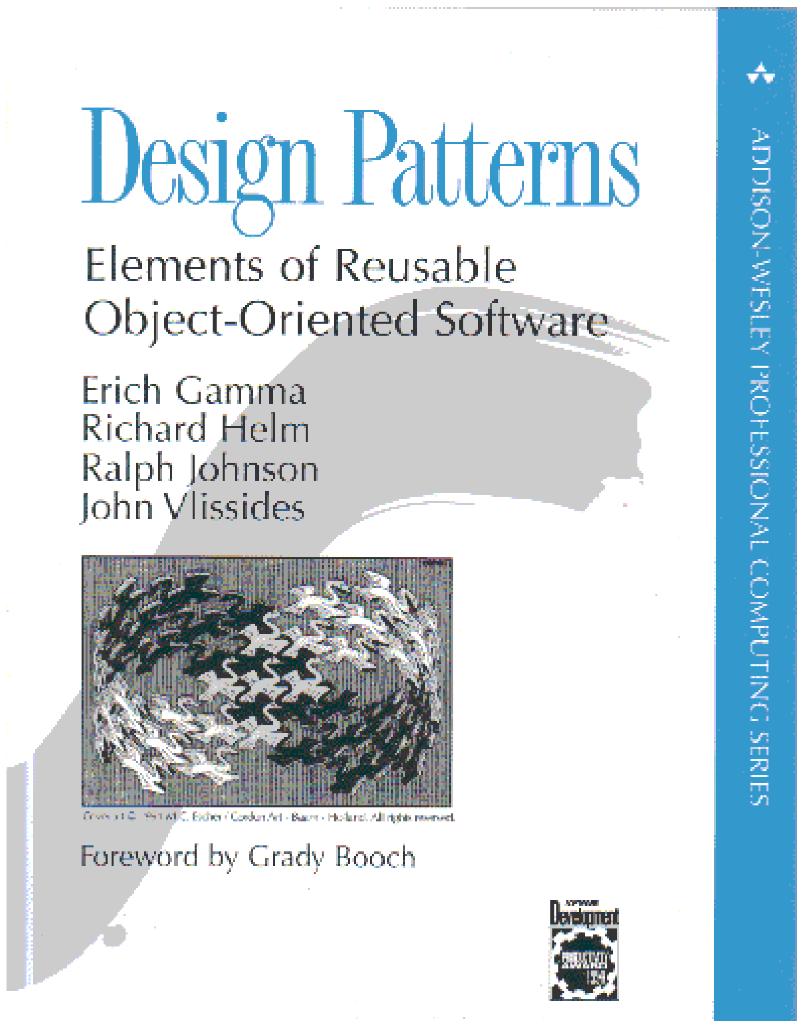
Example Problem 3

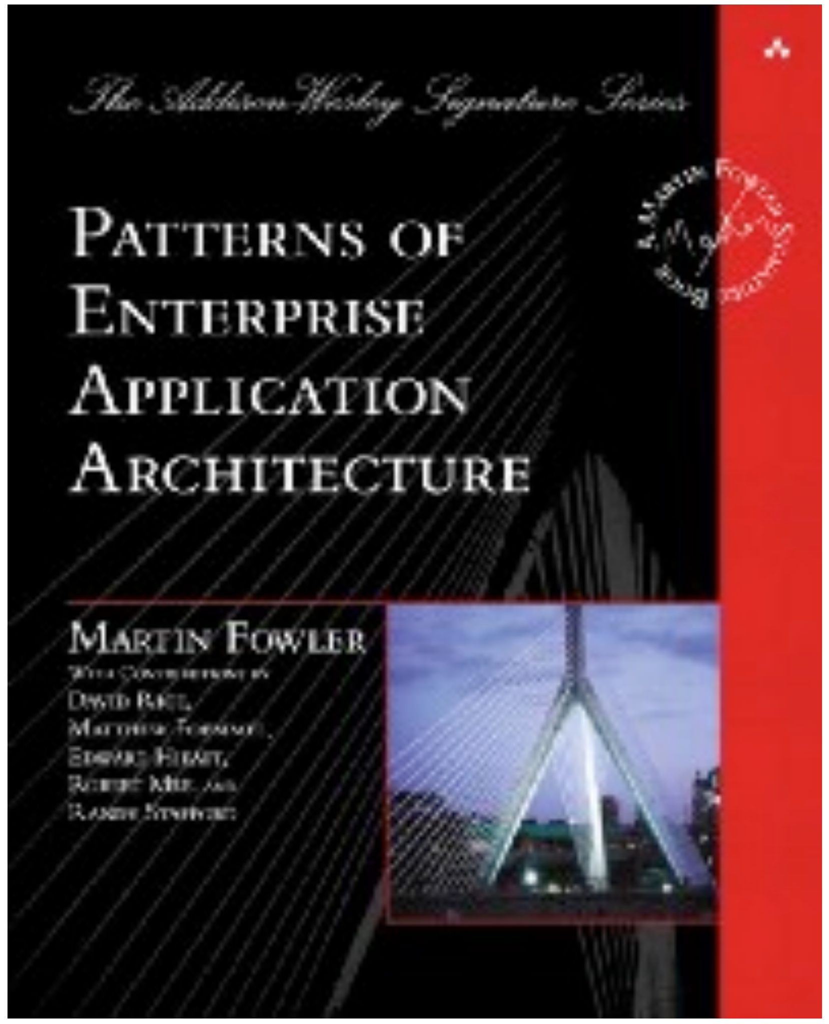
A bug tracking application is to be developed to support the automated management of bugs in a software project. Among the features to be implemented is a core set of bug management commands: a bug can be entered, deleted, allocated to a particular subsystem, its status updated (pending, fixed, not reproduced) 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.

Key Sources

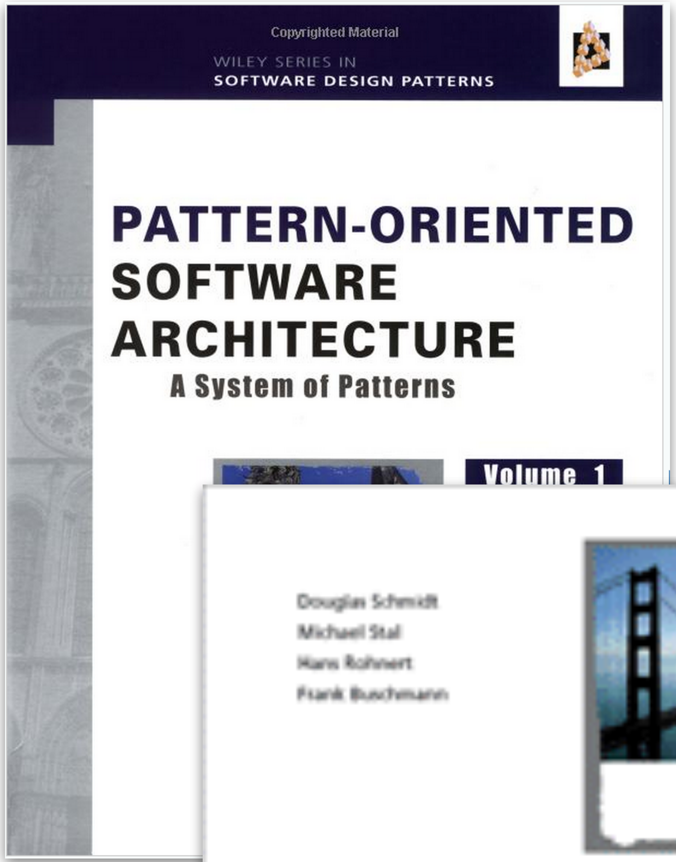




- GOF
- Most typically illustrated within GUI applications (Android, IOS, Native)

- PEAA
- Most typically illustrated within Web Applications (

Other sources - POSA - 5 Volumes!



Copyrighted Material

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED
SOFTWARE
ARCHITECTURE
A System of Patterns

Volume 1

Douglas Schmidt
Michael Stal
Hans Rohnert
Frank Buschmann

PATTERN-ORIENTED
SOFTWARE
ARCHITECTURE
Volume 2 Patterns for Concurrent and Networked Objects

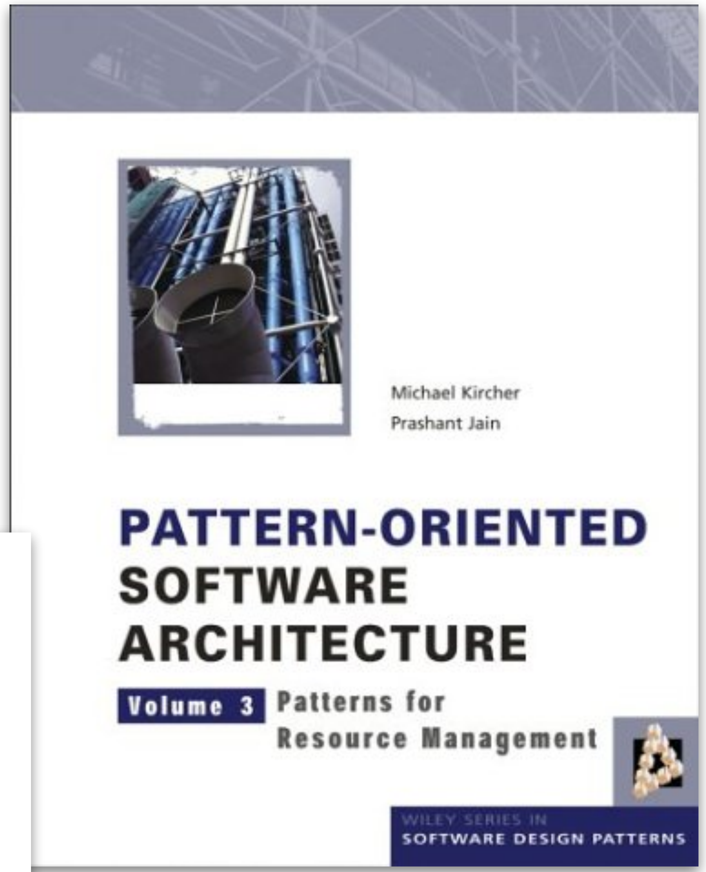
WILEY

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED
SOFTWARE
ARCHITECTURE
A Pattern Language for
Distributed Computing

Volume 4



Michael Kircher
Prashant Jain

PATTERN-ORIENTED
SOFTWARE
ARCHITECTURE
Volume 3 Patterns for Resource Management

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

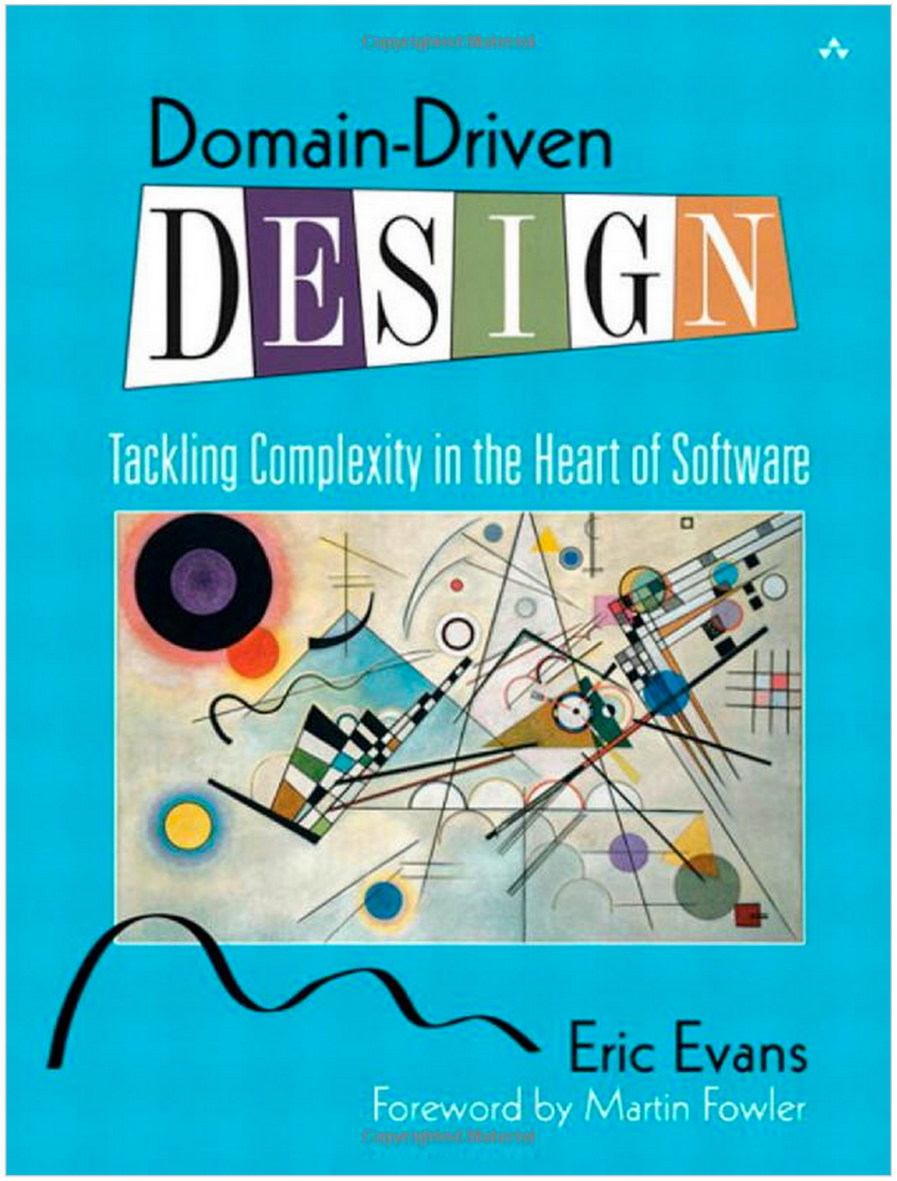
WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

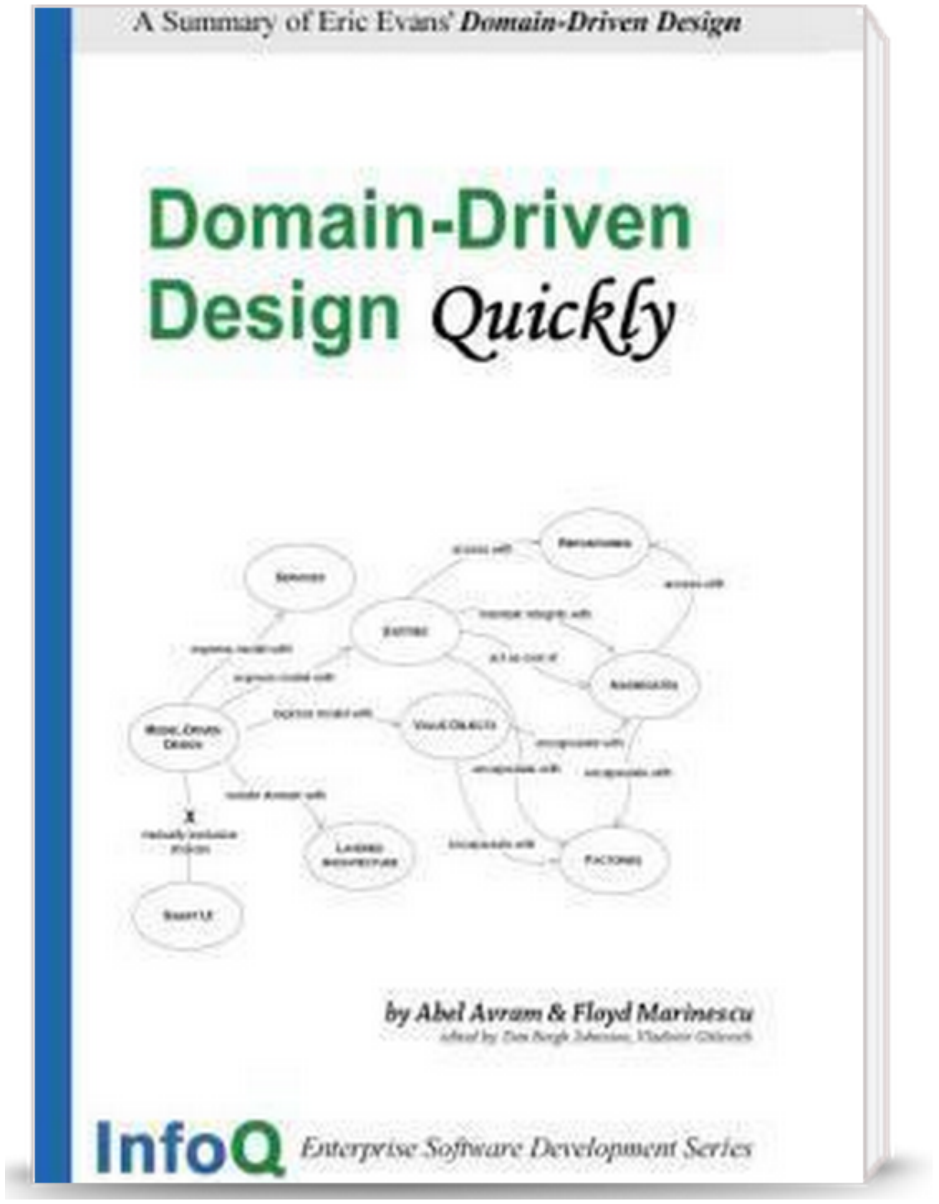
PATTERN-ORIENTED
SOFTWARE
ARCHITECTURE
On Patterns and Pattern Languages

Volume 5

Frank Buschmann
Kevlin Henney
Douglas C. Schmidt

Other Sources - Domain Driven Design


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler


A Summary of Eric Evans' Domain-Driven Design
Domain-Driven
Design Quickly
by Abel Avram & Floyd Marinescu
InfoQ Enterprise Software Development Series