to be agile
lessen the learning curve

Object-Oriented Analysis and Design Patterns

This in-depth sixteen-hour training course presents the same core set of design patterns as our one-day Design Pattern Developer Essentials along with advanced techniques for object-oriented analysis and behavior driven design, as well as an expanded group design exercise.

You'll learn effective ways to analyze software problems, define behaviors that produce observable results, break down tasks by acceptance criteria, and use acceptance tests to discover requirements and flesh out edge cases.

You'll see how understanding patterns reveals the essence of object-oriented thinking and creates a useful context for solving a variety of software problems. You'll come to understand patterns as more than just "reusable solutions to common problems within a given context" and recognize them as a collection of forces that help you penetrate deeper into problems, discovering elegantly simple solutions that make your software more robust and easier to maintain.

The expanded view of patterns you'll gain from this course will help you encapsulate and abstract virtually any problem with maximum flexibility and without over-complicating the solution. You'll learn to make better coding choices and will master a shared vocabulary for talking about design that dramatically improves inter-team communication.

By the end of this training, you'll be armed with several new, effective tools for solving design problems that will empower you to produce immediate improvements in the quality of the software you design and build.

Course Benefits

Completing this course will give you a deeper understanding of the object-oriented development paradigm, and enable you to:

- Explain what patterns are and know when to use them
- Read and write the 3 most-important UML diagrams
- Employ design principles to create higher-quality code
- Define acceptance criteria through specifying behavior
- Use a common vocabulary for communicating designs
- Adopt simple methods to find patterns in problems
- Understand patterns by what they encapsulate
- Apply patterns just-in-time to avoid over-design
- Appreciate the value of shared coding standards
- Contribute to design reviews and evaluate others' designs
- Gain several techniques for doing effective analysis and design
- Master techniques for emerging designs in iterative development

to be agile
lessen the learning curve

Agenda

In Two Full-Day (8-Hour) In-Person Sessions	In Four Half-Day (4-Hour) Online Sessions
Session 1 — Analysis **Introduction**: Purpose and objectives **OOAD Revised**: OO Analysis, BDD **Paradigms and Wisdom: Advice from** GoF **Principles and Perspectives**: Agile design **Software Patterns**: Misconceptions; forces **Encapsulating Varying Behavior I**: Strategy and Template Method Patterns **Encapsulating Varying Behavior II**: State and Bridge Patterns **Encapsulating Foreignness**: Adapter, Façade Session 2 — Design **Review: Integrate and deepen what we covered** **Encapsulating Construction**: Factory-Method, Abstract Factory, and Singleton **Design Exercise**: Set up and group design exercise to be solved using patterns **Design Exercise Continued**: Emerging designs based on additional requirements **Exercise Review: Review of Designs** **Exercise Debrief: Discussion of design tradeoffs** **Encapsulating Sequence, Cardinality and Selection**: Proxy, Decorator, CoR **Case Study Revisited and Wrap-Up**: Solve the initial design problem by refactoring to patterns	Session 1 — Analysis **Introduction**: Purpose and objectives **OOAD Revised**: OO Analysis, BDD **Paradigms and Wisdom: Advice from** GoF **Principles and Perspectives**: Agile design Session 1A — Analysis **Software Patterns**: Misconceptions; forces **Encapsulating Varying Behavior I**: Strategy and Template Method Patterns **Encapsulating Varying Behavior II**: State and Bridge Patterns **Encapsulating Foreignness**: Adapter, Façade Session 2 — Design **Review: Integrate and deepen what we covered** **Encapsulating Construction**: Factory-Method, Abstract Factory, and Singleton **Design Exercise**: Set up and group design exercise to be solved using patterns **Design Exercise Continued**: Emerging designs based on additional requirements Session 2A — Design **Exercise Review: Review of Designs** **Exercise Debrief: Discussion of design tradeoffs** **Encapsulating Sequence, Cardinality and Selection**: Proxy, Decorator, CoR **Case Study Revisited and Wrap-Up**: Solve the initial design problem by refactoring to patterns

Who Should Take This Course

This training will benefit all team members including architects; business analysts; DBAs; designers and developers; development managers; directors; documentation specialists; operations and support staffers; product and project managers; software engineers/programmers; testers and QA engineers; and technical writers, analysts and leads.

Your Instructor

My continuing passion for software design and construction has led me to train more than 10,000 professional software developers for clients that have included Fortune 500 firms such as Microsoft, IBM, Yahoo, Boeing, AT&T, Sprint, Medtronic, SunGard, State Farm, Vanguard, and Weyerhaeuser. As a longtime IBM consultant, I trained software engineers around the globe, giving them the skills to write the next generation of applications and operating system software while earning one of the highest satisfaction ratings in the history of IBM education. Since 2006, I've devoted my consulting practice to providing organizations with training and coaching for software developers and teams transitioning to Agile, Scrum, and Extreme Programming practices.

Praise for this Training

> "This class is essential to any technical professional in a development environment. The approaches covered will help with every phase of the development cycle on any size team or project." – Tyler Ashbridge, Director of Systems Development
>
> "This course helped me to understand how to do quality software in a sustainable manner and support our software development organization to great results."
> —Mikko Ala-Fossi, Dev. Mgr.
>
> "If you are interested in learning techniques that guarantee high quality code—take this class—the payback will start immediately and be significant."
> —Jonathan Lister, SOA Architect/Platform Product Owner
>
> "I thought I was an okay developer. Taking this class has made me realize that there is a better path to follow, which will not only improve the way I work but improve the software I create… I would say take the class and let it change your life."
> —Stephen Jones, GIS Architect