



Agile Incremental Modeling with UML 2.1

5-7 Days

1. Introduction

Object Oriented Programming (OOP) environments offer state of the art and powerful tools for building flexible and extensible software. Agile technologies such as SCRUM, XP, and TDD describe processes and technologies and work habits that increase productivity, control and lead to a shorter time to value (market). Using OOP along with Agile practices is often not sufficient. Maximum benefits and risk reduction are gained only if the SW is appropriately designed and its architecture can effectively be communicated among stakeholders and team members.

Agile Incremental Modeling (AIM) with UML 2.1 combines Agile development practices with formal model driven development, empowering teams to incrementally capture requirements, analyze, design and talk about flexible system architectures. This saves a lot of time and money not only in later stages of a project but right from the beginning.

2. Course Description

In this course we will discuss how to define detailed system requirements, propose a detailed OO implementation independent solution to those requirements and craft a layered, component based architecture that maximizes maintainability, re-use and extensibility of the resulting code. You will see how using UML, teams can talk about the requirements and design decisions in a common language. Special focus is set on using the UML to uncover wholes and inconsistencies in the analysis and design as well as on the process that seamlessly translates requirements into a flexible layered, component based architecture.

The course does not only simply present knowledge. This whole class is all about skills building. After each main topic, we provide an interactive straight-to-the-point comprehensive exercise making you think and ensure that you "own" the learned topic. Two major examples are followed through the course; one discussed as each UML diagram is introduced, and one discussed as the process of Agile requirements capture, analysis and design is discussed. For UML exercises yet another example is used.

3. Goals

The primary goals of this course are:

- To acquire enough competence in the UML for talking about requirements, solution approaches and SW architecture.
- To understand the main design and architecture principles of good OO design.
- To gain enough competence in OO requirements capture, analysis and design to tackle on your own a complete OO project.



4. Participants

The course is meant for software engineers, experienced programmers and system analysts.

5. Pre-requisites

Some experience in software engineering or system analysis is a must. The course doesn't require (but highly recommends) knowledge of an OO language and doesn't concentrate on a specific one.

Course Outline

<p>Day 1:</p> <p>During the first day, the fundamental concepts of the object paradigm and requirements capture are presented.</p> <p>Introduction</p> <ul style="list-style-type: none"> The object paradigm – Overview on the UML – Overview on the agile incremental process. <p>Requirements Capture</p> <ul style="list-style-type: none"> Types of requirements – Vision – Domain Model – Use case model – supplementary requirements – How detailed (Agile) to describe requirements. <p>Day 2:</p> <p>During the second day the UML diagrams needed for analysis are presented.</p> <ul style="list-style-type: none"> Class – Package – Object Diagram. Activity – State Machine – Sequence and Communication – Interaction Overview Diagram. <p>Day 3:</p> <p>During the third day the process of transforming functional requirements into an implementation free conceptual solution is presented. The required detail (Agility) of the analysis model is discussed.</p>	<p>Architectural Analysis</p> <ul style="list-style-type: none"> Identifying classes – Describing classes – Characteristics of good associations and generalizations. <p>Use Case Analysis</p> <ul style="list-style-type: none"> Modeling behaviors though use case realizations - Modeling state dependent behavior. <p>Day 4</p> <p>During the fourth day the UML diagrams and elements needed for representing system architecture are presented.</p> <ul style="list-style-type: none"> Active Classes – Composite Structure – Component – Deployment diagram. <p>Day 5</p> <p>On the fifth day the process of designing flexible system architecture is presented.</p> <p>Logical Architecture</p> <ul style="list-style-type: none"> Design Principles – Architecture Principles – Package View – Modeling System Interfaces – Design Patterns. <p>Physical Architecture</p> <ul style="list-style-type: none"> Process View – Deployment View – Internal Component Design. <p>Focus is given to the required detail (Agility) of the design in order to effectively transition to implementation.</p>
---	--

We can demonstrate a variety of CASE tools (Rhapsody, Enterprise Architect and more). However we do not recommend using them for the exercises as the details of driving them distracts from the main focus and they do not promote team working in the class.

We strongly recommend extending the course by 2 days in order to apply the theory in practice by tackling a real life project of your choice or through a provided sample project.