Understanding the EAMF Approach

Part I:
- Requirements Engineering
- Requirements Model Engineering
- Business Architecture Modeling

Learning Objectives

- Understand requirements engineering “ala” EAMF
- Command of requirements model engineering “ala” EAMF
- Ability to use EAMF to analyze and design business architectures

Topical Outline

In this presentation:
- Traditional/Object-Oriented Architecture Analysis & Design Approach
- EAMF Architecture Analysis & Design Approach

Roadmap

Object Oriented Architecture Analysis & Design Approach
- From Requirements to Analysis
- Analysis Class
- Use Case Realization
- Example: Pay Invoice Use Case
- Pay Invoice Use Case: Flow of Events
- Analysis Class Diagram Realizing a Pay Invoice Use Case
- Collaboration Diagram Realizing a Pay Invoice Use Case
- Analysis and Design Workflow
- Defining the Architecture
- Analysis and Design Roles, Activities, and Artifacts
From Requirements to Analysis

- Requirements specify the behavior of the largest-grained component: the system
  - Specifies the behaviors the system provides for and with the actors
  - “The system shall…”
- Object-oriented analysis is reduction of a system to a coarse set of discovered objects with responsibilities, so that the set of objects is able to fulfill the behavioral requirements of the user
- Analysis limits the effort to objects found in the domain – otherwise you are doing design

Analysis Class

- Analysis classes represent an early conceptual model for ‘things in the system which have responsibilities and behavior’
  - Class, stereotyped as «boundary», «entity» or «control»
  - Entity classes
  - Long-lived, real-life object or event in the application domain
  - Data and behavior
  - Usually persistent (saved in a file or database)
  - Boundary classes
  - Interaction between the system and its actors
  - At the system boundary
  - Control classes
  - Coordination and sequencing of system behavior
  - Transactions

Use-Case Realization

- A use-case realization is a description of how a particular use case is realized within the design model, in terms of collaborating objects

Example: Pay Invoice Use Case

- The use case Pay Invoice is used by a Buyer to schedule invoice payments. The Pay Invoice use case then completes the payment on the due date
- Pre-condition: The Buyer has received the goods or services ordered and at least one invoice from the system. The Buyer now plans to schedule the invoice(s)
- Post-condition: The use-case ends when the invoice has been paid or the invoice payment was canceled and no money was transferred
Pay Invoice Use Case: Flow of Events

- Primary Scenario (i.e., basic path)
  1. The buyer invokes the use case by beginning to browse the invoices received by the system. The system checks that the content of each invoice is consistent with the order confirmations received earlier (as part of the Confirm Order use case) and somehow indicates this to the buyer. The order confirmation describes which items will be delivered, when, where, and at what price.
  2. The buyer decides to schedule an invoice for payment by the bank, and the system generates a payment request to transfer money to the seller’s account. Note that a buyer may not schedule the same invoice for payment twice.
  3. Later, if there is enough money in the buyer’s account, a payment transaction is made on the scheduled date. During the transaction, money is transferred from the buyer’s account to the seller’s account, as described by the abstract use case Perform Transactions (which is used by Pay Invoice). The buyer and the seller are notified of the result of the transaction. The bank collects a fee for the transaction, which is withdrawn from the buyer’s account by the system.
  4. The use-case instance terminates

Analysis Class Diagram Realizing a Pay Invoice Use Case

Collaboration Diagram Realizing a Pay Invoice Use Case

Analysis and Design Workflow
Summary: Architecture Engineering Steps

- Create an initial sketch of the architecture of the system
- Define an initial set of architecturally significant elements to be used as the basis for analysis
  - Issues that are typically architecturally significant include performance, scaling, process and thread synchronization, and distribution
- Define an initial set of analysis mechanisms
- Define the initial layering and organization of the system
- Define the use-case realizations to be addressed in the current iteration
- Identify analysis classes from the architecturally significant use cases
- Update the use-case realizations with analysis class interactions

Summary: Architecture Engineering Steps (cont.)

- How to Staff
  - Small team with cross-functional team members
  - The team should include members with domain experience who can identify key abstractions
  - The team should also have experience with model organization and layering
- Work Guidelines
  - The work is best done in several sessions, perhaps performed over a few days (or weeks and months for very large systems)
  - Iterate between Architectural Analysis and Use-Case Analysis

Summary: Architecture Engineering Artifacts
Roadmap

EAMF Architecture Analysis & Design Approach
- Building Pattern Cluster Networks Using EAMF
- EAMF Methodology
- EAMF-Augmented Incremental/Iterative SDLC Used for ABS-B2
- ABS-B2’s Project Requirements Types and Enterprise Requirements Categories
- Requirements Traceability Graph
- Use of IBM Rational ReqPro for ABS-B2 Requirements Engineering Phase
- Building Pattern Cluster Networks Using EAMF
- EAMF Requirements Model Engineering
- etc.

EAMF Methodology

EAMF-Augmented Iterative/Incremental SDLC Used for ABS-B2
ABS-B2's Project Requirements Types and Enterprise Requirements Categories

Requirements Traceability Graph

Use of IBM Rational ReqPro for ABS-B2's Requirements Engineering Phase

Building Pattern Cluster Networks Using EAMF
Architectural Analysis Using EAMF

EAMF Requirements Model Engineering

Requirements and Definitions Traceability Graph

Reasoning About Business Entities and Their Dependencies and Goals

Pattern Language Structure for Agent Patterns Selection
Use of Sparx Systems EA to Engineer the ABS-B2 EAMF Requirements Model

Use of IBM Rational ReqPro for ABS-B2's Requirements Model Engineering Phase

Architectural Analysis Using EAMF 30 September 15 2006

Architectural Analysis Using EAMF 31 September 15 2006
Architectural Analysis Using EAMF 33 September 15 2006

jUCMNav GRL Modeling Constructs

- Actor
- Resource
- Goal
- Task
- Softgoal
- Data
- Satisfied
- Unsatisfied
- Weakly Satisfied
- Weakly Unsatisfied
- Unconfirmed
- Confirmed
- Derived
- Denied
- AND
- OR

(a) GRL Elements (b) GRL Satisfaction Levels (c) Link Composition

(d) GRL Links

(e) GRL Contribution Types

Architectural Analysis Using EAMF 34 September 15 2006

jUCMNav GRL Modeling Constructs (continued)

Actor-Dependency Diagram for the ESLM Domain (Early Requirements Discipline)
UCM Notations Summarized (continued)

Inter-Scenario Relationships Design Patterns Used in EAMF’s BPM approach

Integration of Goal Oriented and Scenario-Based Modeling

Use of Sparx Systems EA to Analyze the ABS-B2 Business Architecture
Actor/Dependency Diagrams (Late Requirements Discipline)

Composite UCM Root Map for ESLM

Plug-In Map for HandleAllEvents in Root Map

Generic Plug-In Map for HandlePolicyServiceEvents and HandleInvoiceServiceEvents
Generic Plug-In Map for ProcessAllRequests Stub in Root Map