A Framework Development Process for Product-Line Architectures



Detlef Schmitt
Sema GmbH, Kurt-Schumacher-Strasse 241
D-26389 Wilhelmshaven, Germany
and
Open University
Milton Keynes, MK7 6AA, United Kingdom
Phone: +49-4421-802-224, Fax: +49-4421-802-555
Email: Detlef.Schmitt@so.sema.de



Contents

- Introduction
- Background and Objectives of the Project
  - Background and Settings
  - Goals and Objectives
- A Process for the Development of an Object-Oriented Framework
  - Overview of the Process
  - Domain Analysis of Discrete Event Simulation
  - Design of the Simulation Framework
  - Framework Implementation
  - Instantiation of the Framework
- Lessons Learned
- Summary

Introduction

- Software product-lines are becoming more and more common
- OO frameworks can serve as a kernel for software product-lines
- Current analysis and design processes are only concerned with the creation of new applications
- Need for a specific development process for frameworks

Background and Goals

Background and Settings

- Development of simulators in the past years
  - Individual, hand-crafted simulators
  - No pre-planned, large-scale reuse
- Need for
  - cost reduction
  - improved quality
  - simplified maintenance

Background and Goals

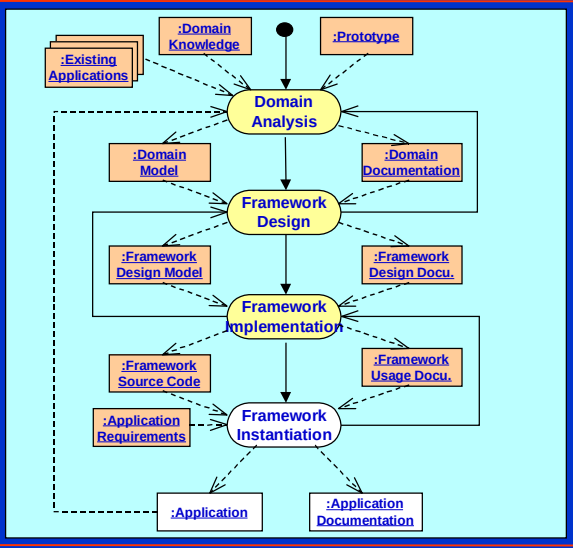
Goals and Objectives

- Enhance flexibility
- Adaptable to different requirements
- Improve software quality by reuse of exisiting and tested artifacts
- Cost reduction through reuse
- Build software components as simple, abstract, and configurable as possible

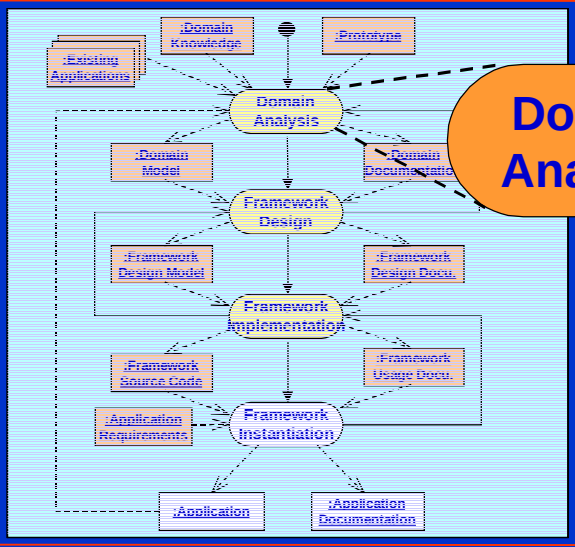
Overview of the Process

- Iterative, incremental, and adaptive approach
- Work on different levels of abstraction in parallel
- Frequent feedback throughout the process

Overview of the Process



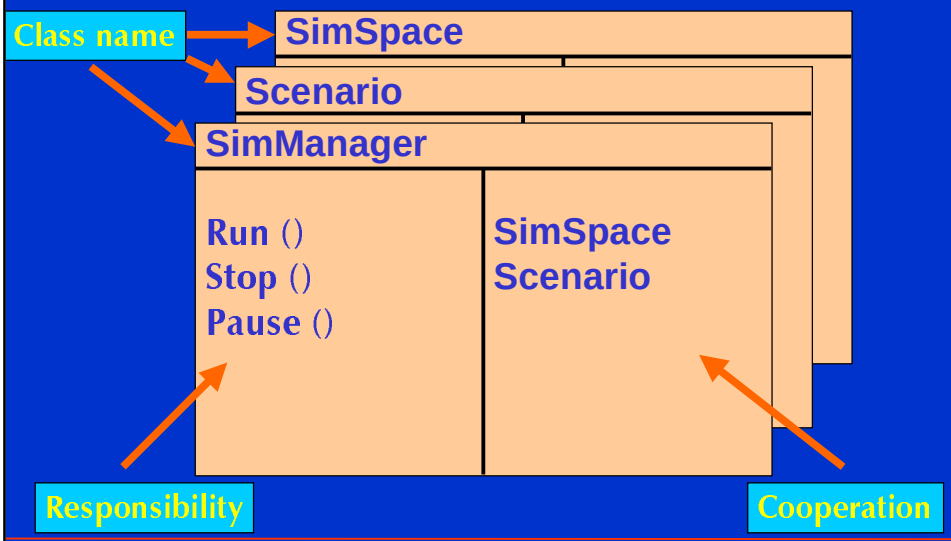
Domain Analysis



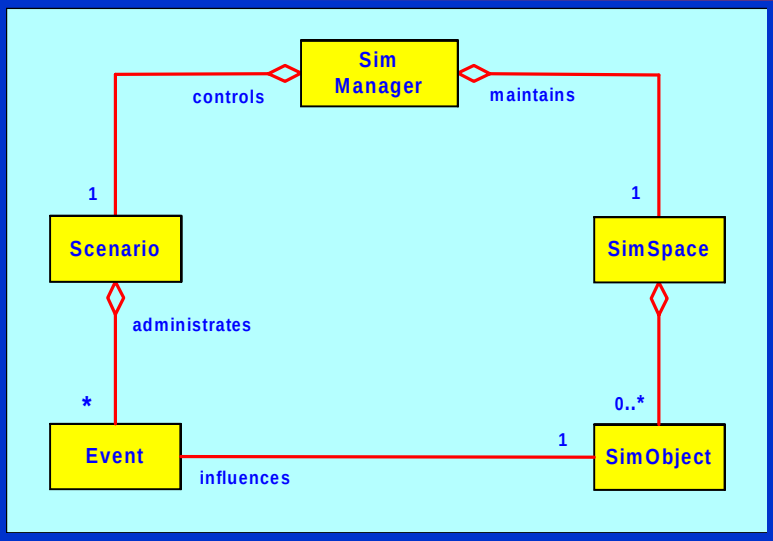
Domain Analysis

- Purpose: Capture and Structure Information, Knowledge, and Expertise in a Given Area of Application (problem domain)
- Goal: Model the Problem as Part of Reality
- Sources: Domain Experts, Available Applications, Prototypes, Scientific & Technical Literature, etc.
- Means: Analysis, Modeling, CRC Cards, Use Cases, Class Diagrams, Sequence Diagrams, etc.

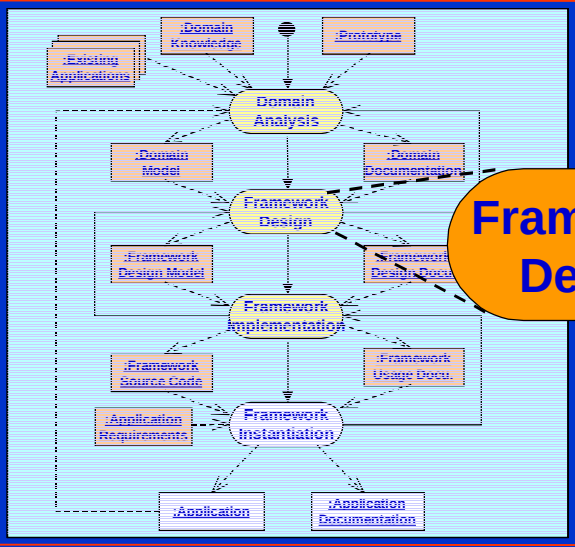
Domain Analysis: CRC Cards



Domain Analysis: Class Diagram



Framework Design



Framework Design

Design Considerations

- Small Is Beautiful
- As Simple As Possible
- No GUI As Part of the Framework
- Only Basic Functionality Is Required

Framework Design

- Refinement of Domain Model (problem -> solution)
- Detailing of Class Responsibilities
- Hot-Spot Analysis to Identify Frozen and Hot-Spots
- Refinement of Classes
  - Relationships
  - Cardinality
- Definition of the Main Methods

Framework Design

Frozen Spots:

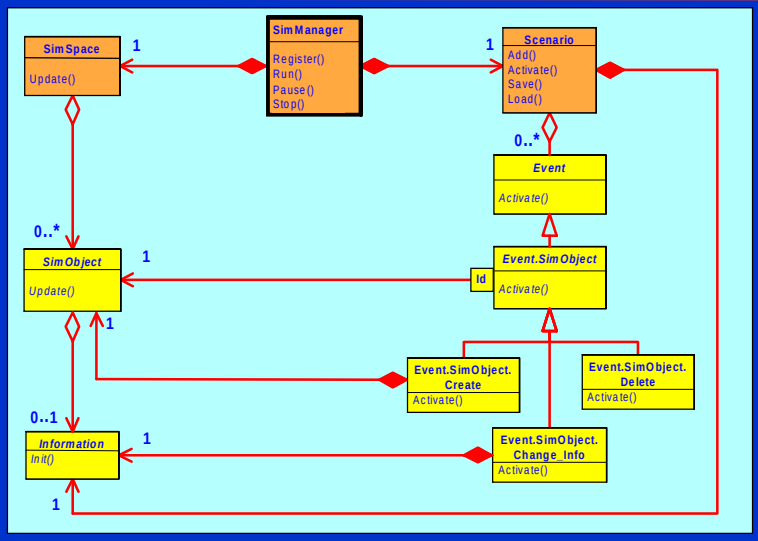
- Simulation Manager (Active and Concrete Class SimManager)
- Scenario (Concrete Class Scenario)
- Simulation Space (Concrete Class SimSpace)

Framework Design

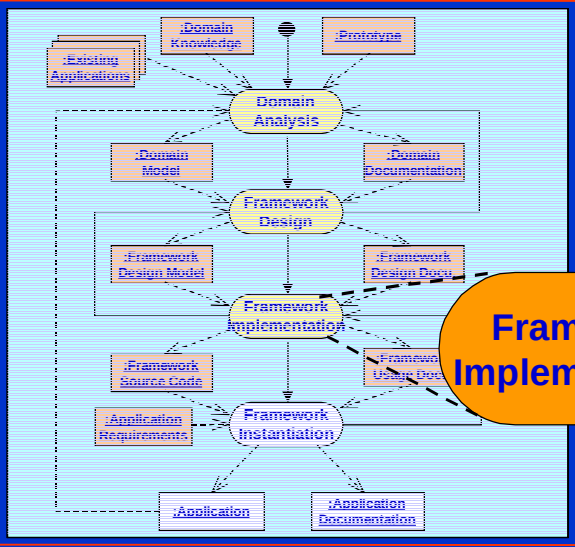
Hot-Spots (Variation Points):

- Simulation Object (Abstract Class SimObject)
- Information (Abstract Class Information)
- Event (Abstract Class Event)
- Event.SimObject (Abstract Class)
- Event Service Classes (Event.SimObject.Create, Event.SimObject. Delete, Event.SimObject.Change_Info)

Framework Design



Framework Implementation



Mapping: Requirements

Requirements For the Mapping Process:

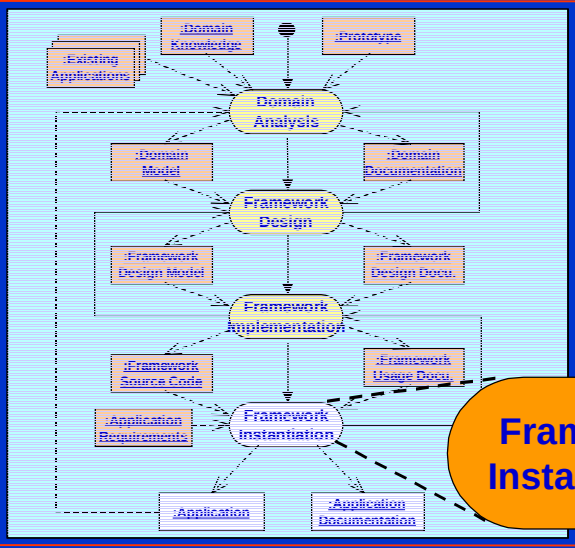
- Translation from Design to Language Idioms
  - Uniformity
  - Orthogonality
- Simplicity of the Overall Approach
- Coverage of All Relevant Details in UML Constructs
- Comprehensibility of the Mapping

Mapping: Concepts

Object-Oriented Concepts to be Mapped:

- Classes With Their Attributes and Methods
- Inheritance
- Aggregations
- Compositions

Framework Instantiation



Framework Instantiation

- Write a Wrapper Program (e.g. GUI)
- Instantiate Objects From Concrete Classes
  SimManager, SimSpace, and Scenario
- Derive Concrete Classes From SimObject
- Add Behavior By Implementing Methods
- Instantiate Respective Objects

Framework Instantiation

- Derive Concrete Classes from Event.SimObject For Instantiation of Events
- Add Behavior to Event.SimObject By Implementing Appropriate Methods
- Instantiate and Link Information Objects to Events

Lessons Learned

- Problem Domain Entities and Their Relationships Can Be Implemented As a Framework -> efficient reuse
- Concentration on essential properties
- Framework Design was Restructured and Revised Several Times (Iterative Process)

Lessons Learned

- Increased Flexibility in Different Applications
- Components Derived from Framework Can Be Used With Other Product-Line Components
- Documentation Is Crucial

Lessons Learned

- UML Was Successfully Used for Process Modeling, Domain Analysis, and Design
- Heterogenious Environment With Different Component Technologies and Languages is possible

Summary

- ✓ Introduction
- ✓ Background and Objectives of the Project
  - ✓ Background and Settings
  - ✓ Goals and Objectives
- ✓ A Process for the Development of an Object-Oriented Framework
  - ✓ Overview of the Process
  - ✓ Domain Analysis of Discrete Event Simulation
  - ✓ Design of the Simulation Framework
  - ✓ Framework Implementation
  - ✓ Instantiation of the Framework
- ✓ Lessons Learned

Questions

?