Object Oriented Model of Objectory Process

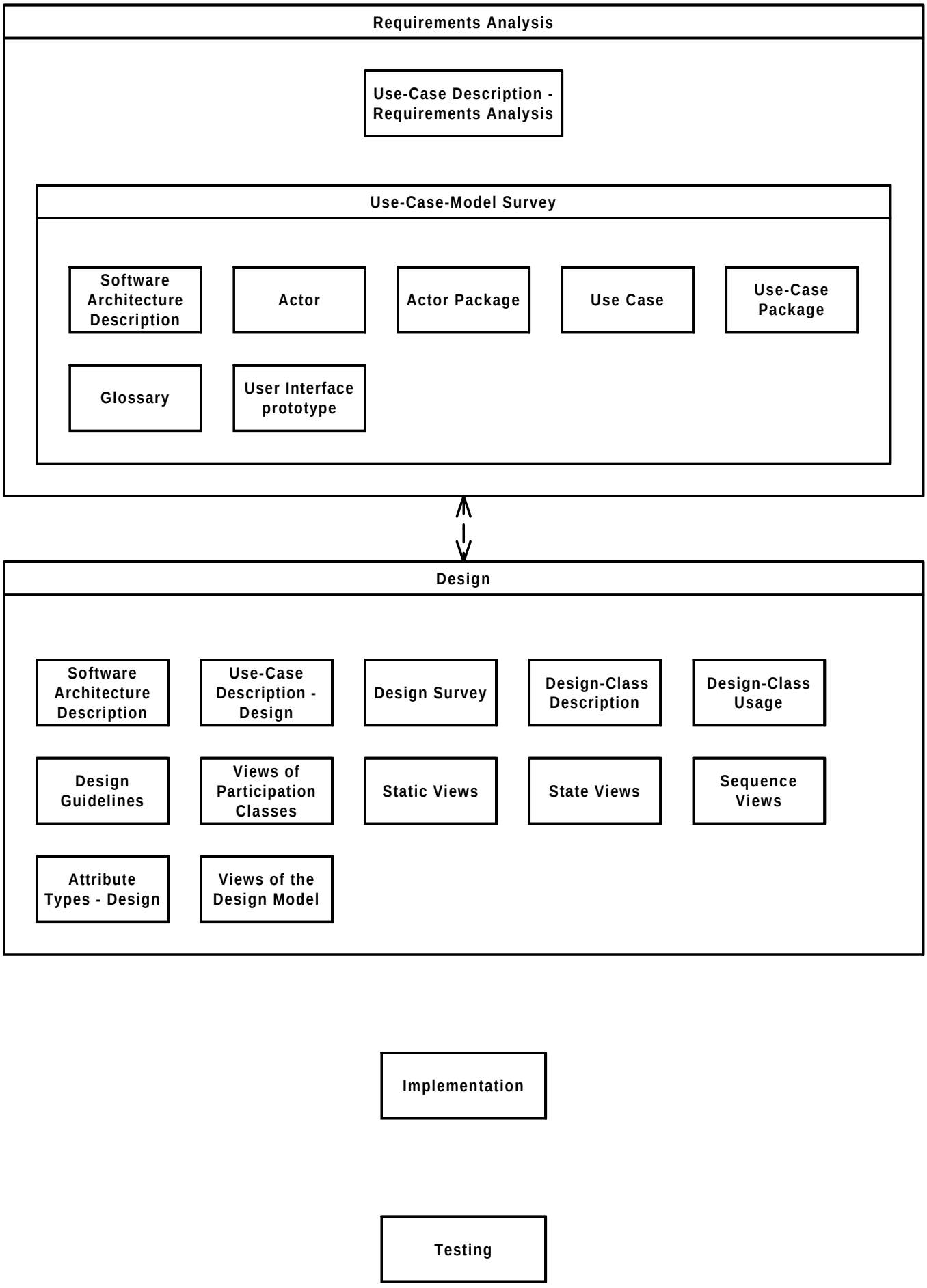
Characteristics of Original Process

The original Objectory Process version 4.0 (demo version, Rational, 1997) is complex, but it is made more manageable by viewing the process in different ways such as by notation, workflow, documentation, artifacts and workers. It is also made more manageable by necessary configuration (adaptation) to meet specific needs. The representation of the process in Objectory Online helps significantly in managing the complexity of the process.

Benefits of Object-Oriented Representation

- An object-oriented model is simpler than the original one: the same information is described in 11 documents in the object-oriented model and in 26 documents in Objectory Online.
- An object-oriented model provides a consistent framework: it makes small terminology inconsistencies in the original process noticeable and draws attention to missing information.

Static Structure Diagram of Objectory Process



Examples of Process Class Specifications

Requirements Analysis
«constructor» 1. Capture a Common Vocabulary 2. Find Use Cases and Actors 3. Describe the Use-Case Model 4. Define the Use-Case View 5. Describe a Use Case 6. Structure the Use-Case Model «quality assurance» It is formally verified that the results of requirements analysis conform to the customer's view of the system. The customer should approve it for you to continue system development.
Purpose (responsibility): To come to an agreement with the customer and the users on what the system should do. To give system developers a better understanding of the requirements on the system. To delimit the system. To provide a basis for planning the technical contents of iterations. Workers: Use-case-model architect, Use-case specifier, requirements reviewer, architect Used by: customer, users, system developers

Use-Case-Model Survey

Introduction
Survey Description (information about the system that the use-case model does not cover, but that the reader must know to understand the system's functions.)
Actors
Use Cases
Objectory Glossary
Views of the Use-Case Model (relation between actors and use cases)
State Views (behavior of actor instances)

«constructor»
Not provided in demo version

«quality assurance»
Verify that the use-case model correctly describes the functional requirements on the system; serves as a good basis for design, implementation, and test; and is consistent with respect to the general use-case-modeling guidelines.

Purpose (responsibility):
There should be one Use-Case-Model Survey for the system. This document provides comprehensive information about the system. It describes the use-case model, briefly defining all the actors and use cases. It can serve as the customer contract when considered with other documentation. This document is extremely important.

Workers: Use-Case-Model architect (responsible for the document), requirements reviewer

Used by: Everyone interested in the system should see it: the customer, potential users, use-case designers, the manager, people outside the project but within the organization, such as executives and steering committees, the people who review the use-case model, the people who work with design, the people who test, the people who develop the next version of the system, the people who write documentation.

Actor

Name
Brief description
Outgoing associations, such as inheritance-associations to other actors and communication-associations to use cases.

«constructor»
given by examples or various actors

«quality assurance»
Correctness: Have you found all the actors; that is, have you accounted for and modeled all the roles in the system's environment? Although you should check this, you cannot be sure of the answer until you have found and described all the use cases.

Consistency: Is each actor involved with at least one use case? Remove any actors not mentioned in the use-case descriptions or any actors without communication- associations to or from a use case. On the other hand, an actor mentioned in a use-case description is likely to have a communication-association to or from that particular use case.

Structure: Can you name at least two people who would be able to perform as a particular actor? If not, check if the role the actor models is only part of another role. If so, you should merge the actor with another actor. Do any actors play similar roles in relation to the system? If so, you should merge them into a single actor. The communication-associations and use-case descriptions show how the actors and the system interrelate. Do two actors play the same role in relation to a use case? If so, you should use inheritance-associations to model their shared behavior. Will a particular actor use the system in several (completely different) ways or does he have several (completely different) purposes with using the use case? If so, you should probably have more than one actor.

Readability: Do the actors have intuitive and descriptive names? Can both users and customers understand the names? It is important that actor names correspond to their roles. If not, change them.

Purpose (responsibility): To fully understand the system's purpose you must know who the system is for, that is, who will use it.

Workers: Use-Case-Model Architect, Use-Case Specifier

Used by: not specified

Use-Case Description - Requirements Analysis

«constructor»
1. Describe a Use Case
2. Structure the Use-Case Description
3. Describe Interfaces
4. Describe Communication Protocols

«quality assurance»
Evaluate your results. More information not provided in demo version.

Purpose (responsibility): To describe the use case's flow of events in detail so that the customer and the users can understand it.

Workers: Use-Case Specifier

Used by: not specified

Design

«constructor»
1. Define the Use-Case View
2. Create a Preliminary Logical View
3. Find Design Classes Based on a Use Case
4. Homogenize the Design Model and the Logical View
5. Adjust Design Classes
6. Distribute Behavior
7. Define Operations
8. Define the Organization of Subsystems
9. Organize the Classes in Service Packages
10. Define the Implementation View
11. Define the Process View
12. Define the Deployment View
13. Maintain Package

«quality assurance»
1. Review the Architecture
2. Review the Design Model

Purpose (responsibility): not provided in demo version

Workers: Architect, Use-Case Designer, Designer, Package Owner, Design Reviewer

Design Survey

Information about methods not provided in demo version

Purpose (responsibility): This document describes the design model comprehensively, in terms of how the system is structured into categories and what design classes are in the system.

Workers: not provided in demo version

Used by: use-case designer, design-class designer, people who design the next revision, people who review the design model, unit testers, design manager

Sequence Views

Information about methods not provided in demo version

Purpose (responsibility): A Sequence View describes how objects interact to perform the behavior of the use cases by showing in detail for a particular use case how control is passed between objects.
Workers: not provided in demo version
Used by: not specified

Design Class Description

Information about methods not provided in demo version

Purpose (responsibility): The purpose of the document is to completely describe a design class. A design class represents an abstraction of classes in the system's implementation. A design class defines a set of design objects with attributes, associations, and operations.

Workers: Not provided in demo version

Used by: design-class designers, people who design the next revision of the system, people who review.

Use-Case Description - Design

Information about methods not provided in demo version

Purpose (responsibility): A Use-Case Description – Design describes how a particular use case's flow of events is performed by means of communicating design objects. There is one document for each use case.

Workers: not provided in demo version

Used by: not specified

View of Participating Classes

Information about methods not provided in demo version

Purpose (responsibility): A view of participating classes illustrates the classes whose objects participate in the use case's flow of events, and how the objects participate in terms of associations between the classes. You can make one or more views of this kind for each use case.

Workers: not provided in demo version

Used by: not specified