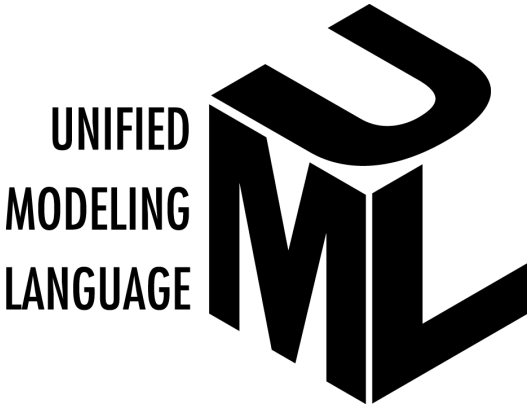

UML Extension for Objectory Process for Software Engineering

version 1.1
1 September 1997

Rational Software ■ Microsoft ■ Hewlett-Packard ■ Oracle
Sterling Software ■ MCI Systemhouse ■ Unisys ■ ICON Computing
IntelliCorp ■ i-Logix ■ IBM ■ ObjecTime ■ Platinum Technology ■ Ptech
Taskon ■ Reich Technologies ■ Softeam

ad97-08-06

Contents

1. INTRODUCTION

This document defines the *UML Extension for Objectory Process for Software Engineering*, defined in terms of the UML's extension mechanisms, namely Stereotypes, TaggedValues, and Constraints.

See the *UML Semantics* document for a full description of the UML extension mechanisms.

This document is not meant to be a complete definition of the Objectory process and how to apply it, but it serves the purpose of registering this extension, including its icons.

2. SUMMARY OF EXTENSION

2.1 STEREOTYPES

Metamodel Class	**Stereotype Name**
Model	use case model
Model	analysis model
Model	design model
Model	implementation model
Package	use case system
Subsystem	analysis system
Subsystem	design system
Package	implementation system
Subsystem	analysis subsystem
Subsystem	design subsystem
Package	implementation subsystem
Package	use case package
Subsystem	analysis service package
Subsystem	design service package
Class	boundary
Class	entity
Class	control
Association	communicates
Association	subscribes
Collaboration	use case realization

2.2 TAGGEDVALUES

This extension does not currently introduce any new TaggedValues.

2.3 CONSTRAINTS

This extension does not currently introduce any new Constraints, other than those associated with the well-formedness semantics of the stereotypes introduced.

2.4 PREREQUISITE EXTENSIONS

This extension requires no other extensions to the UML for its definition.

3. STEREOTYPES AND NOTATION

3.1 MODEL, PACKAGE, AND SUBSYSTEM STEREOTYPES

An Objectory system comprises several different but related models. Objectory models are characterized by the lifecycle stage that they represent. The different models are stereotypes of model:

- Use Case
- Analysis
- Design
- Implementation

3.1.1 Use Case

A *Use Case Model* is a model in which the top-level package is a use case system.

A *Use Case System* is a top-level package. A use case system contains use case packages and/or use cases and/or actors and relationships.

A *Use Case Package* is a package containing use cases and actors with relationships. A use case is not partitioned over several use case packages.

3.1.2 Analysis

An *Analysis Model* is a model whose top-level package is an analysis system.

An *Analysis System* is a top-level subsystem. An analysis system contains analysis subsystems and/or analysis service packages and/or analysis classes (i.e. entity, boundary, and control) and relationships.

An *Analysis Subsystem* is a subsystem containing other analysis subsystems, analysis service packages, analysis classes (i.e. entity, boundary, and control), and relationships.

An *Analysis Service Package* is a subsystem containing analysis classes (i.e. entity, boundary, and control) and relationships.

3.1.3 Design

A *Design Model* is a model whose top-level package is a design system.

A *Design System* is a top-level subsystem. A design system contains design subsystems and/or design service packages and/or design classes and relationships.

A *Design Subsystem* is a subsystem containing other design subsystems, design service packages, design classes, and relationships.

A *Design Service Package* is a subsystem containing design classes and relationships.

3.1.4 Implementation

An *Implementation Model* is a model whose top-level package is an implementation system.

An *Implementation System* is a top-level package. An implementation system contains implementation subsystems and/or components and relationships.

An *Implementation Subsystem* is a package containing implementation subsystems and/or components and relationships.

3.1.5 Notation

Package stereotypes are indicated with stereotype keywords in guillemets («stereotype name»). There are no stereotyped icons for packages.

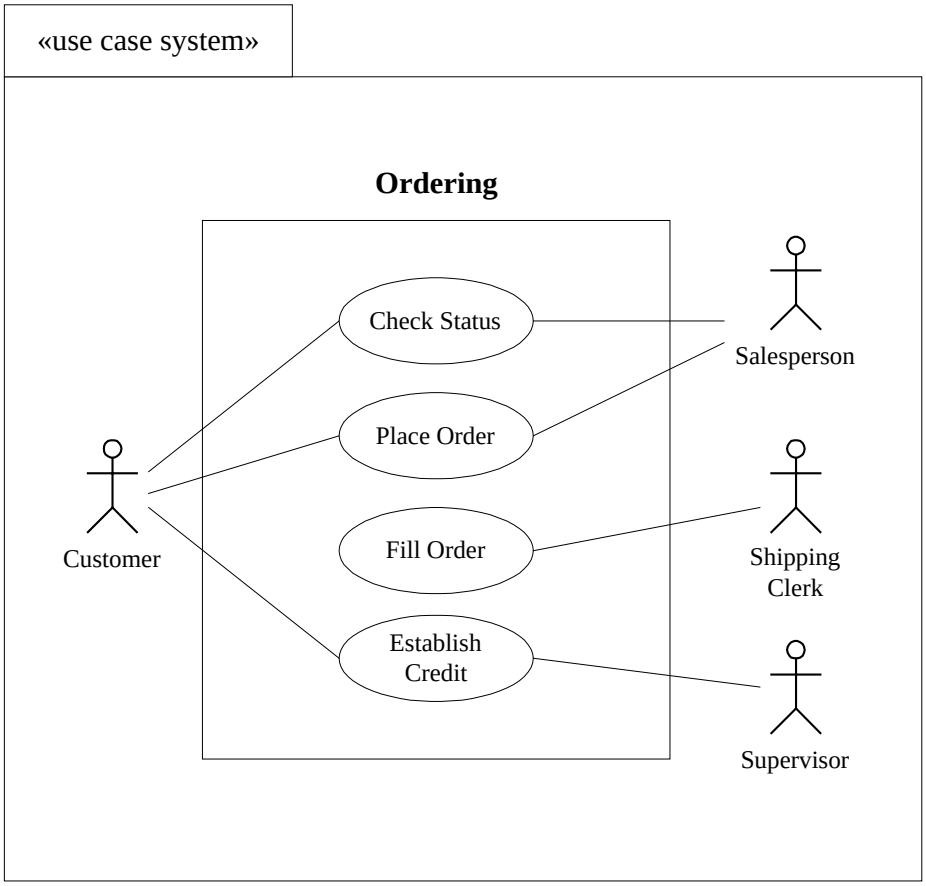

Figure 1. Objectory packages

3.2 CLASS STEREOTYPES

Analysis classes come in the following three kinds (design classes are not stereotyped in the Objectory process):

3.2.1 Entity

Entity is a class that is passive; that is, it does not initiate interactions on its own. An entity object may participate in many different use case realizations and usually outlives any single interaction.

3.2.2 Control

Control is a class, an object of which denotes an entity that controls interactions between a collection of objects. A control class usually has behavior specific for one use case and a control object usually does not outlive the use case realizations in which it participates.

3.2.3 Boundary

A *Boundary* is a class that lies on the periphery of a system but within it. It interacts with actors outside the system as well as objects of all three kinds of analysis classes within the system.

3.2.4 Notation

Class stereotypes can be shown with keywords in guillemets. They can also be shown with the following special icons:

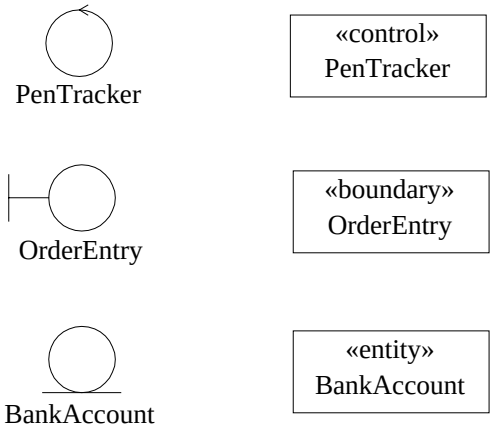

Figure 2. Class stereotypes

3.3 ASSOCIATION STEREOTYPES

The following are special Objectory associations between classes:

3.3.1 Communicates

Communicates is an association between actors and use cases denoting that the actor sends messages to the use case and/or the use case sends messages to the actor. This may be one-way or two-way navigation. The direction of communication is indicated by the navigability of the association.

3.3.2 Subscribes

Subscribes is a association whose source is a class (called the subscriber) and whose target is a class (called the publisher). The subscriber specifies a set of events. The subscriber is notified when one of those events occurs in the target.

3.3.3 Notation

Association stereotypes are indicated by keywords in guillemets. There are no special stereotype icons. The stereotype «communicates» on Actor-Use Case associations may be omitted, since it is the only kind of relationships between actors and use cases.

4. WELL-FORMEDNESS RULES

Stereotyped model elements are subject to certain constraints in addition to the constraints imposed on all elements of their kind.

4.1 GENERALIZATION

All the modeling elements in a generalization must be of the same stereotype.

4.2 ASSOCIATION

Apart from standard UML combinations the following combinations are allowed for each stereotype:

Table 1. Valid association stereotype combinations

To: **From:**	**actor**	**boundary**	**entity**	**control**
actor		communicates		
boundary	communicates	communicates	communicates subscribes	communicates
entity			communicates subscribes	
control		communicates	communicates subscribes	communicates