Software engineering
Diego Axel Valenzuela Rios | 200904

Unified modeling language (UML)

Models

- Simplification of reality
- Abstract

Uses

- Can be represented in different views
- Visualize new or existing systems
- Helps to understand any system
- Communicate decisions to project's stakeholders
- Document decisions made in each OODD workflow
- Help to specify aspect of system
- Are specified from the requirements model
- Structure Diagram
- Behavior Diagram
- Template for implementing solution model
- Solution model

Software Models

- Architecture Model
- Requirements Model
- Design Model
- Solution Model
- Code
- Stakeholders Needs Model

Figure 2-9 Software Development as a Series of Model Transformations

UML

- A graphical language for visualizing, specifying, constructing, and documenting the artifacts of a system
- Elements
- Relationships
- Diagrams
- Views

Figure 2-10 Elements of UML Diagrams

Elements

- Things
  - Structural Things
  - Behavioral Things
  - Grouping Things
  - Annotational Things
- Relationships
  - Dependencies
  - Associations
  - Generalizations
  - Realizations

Kind of Diagrams

- Class Diagram
- Object Diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diagram
- Package Diagram
- Profile Diagram
- Use Case Diagram
- Activity Diagram
- Communication Diagram
- Interaction Overview Diagram
- Sequence Diagram
- State Machine Diagram
- Timing Diagram

Figure 2-11 Set of UML Diagrams

Views

- Structural
- Behavioral

Figure 2-12 Views Created From UML Diagrams

Mechanisms

- Group together any UML elements and diagrams
- Show the dynamic behavior of objects in a system
- Group elements into UML element classes or another
- Elements can connect to a type of relationship
- Elements can connect to a type of diagram

Figure 3-10 Example of Using Packages

Figure 2-14 Example of Using a UML Note

Figure 2-15 Example of UML Dependencies and Stereotypes

Elements and connectors

- Adds notes to any aspect of diagrams
- Annotations (classes, methods, components, etc)
- Balance = balance + amt
- Balance = balance - amt

Dependency

- Dependency
- Stereotypes

Table 3-4 Standard UML Stereotypes

Stereotype	Description
<<actor>>	An actor defines a coherent set of roles that users of the system can play when interacting with the use cases. Actors are instances of classes, and they interact with the system through one or more use cases.
<<interface>>	Used to a conceptual stage identifying major classes and their interfaces, or to represent a header, IDL (CORBA), or Java (J2EE).

Table 3-5 Standard UML Stereotypes (Continued)

Stereotype	Description
<<extend>>	Indicates that the base use case may, at certain points, be extended by the use case that is the target of the extend relationship.
<<signal>>	Signal is a message or process stimulating between two active objects, that is, between two processes. Signal carries a message to be processed. Signal is similar to a message in object-oriented terms.
<<include>>	An include relationship from one use case to another specifies that the base use case will include the behavior specified in the target use case.
<<stereotype>>	An stereotype is an profile that allows you to specify variations of UML elements, such as classes and attributes, in a platform-specific and domain-specific manner. (UML v1.x page 119)
<<table>>	Indicates that a class represents a database table.

Table 3-7 Non-Standard UML Stereotypes

Stereotype	Definition
<<entity>>	This stereotype delineates the set of methods that retrieve the definition of a class.
<<boundary>>	This stereotype delineates the set of operations that interface with the outside world.
<<control>>	This stereotype delineates the set of business rules and other logic that is not tied to a specific object.
<<process>>	This stereotype delineates the set of methods that coordinate the flow of control.
<<thread>>	A class that defines a graphical user interface (GUI) or a window.

Table 3-8 Non-Standard UML Stereotypes (Continued)

Stereotype	Definition
<<persistent>>	This stereotype delineates the set of methods that retrieve the definition of a class.
<<transient>>	This stereotype delineates the set of operations that do not need to be persistent.
<<utility>>	This stereotype delineates the set of methods that are not associated with any class.
<<exception>>	A class that defines a graphical user interface (GUI) or a window.

Non-Standard stereotypes

Table 7-1 Example Associations With Relationship and Role Labels

- Represent relationships between classes
- Aggregation
- Composition

Figure 7-1 Example Associations With Relationship and Role Labels

Figure 7-2 Example Associations With Multiplicity Labels

Table 7-2 Different Types of Multiplicity

Multiplicity Indicator	Meaning
1	One and only one object.
0..n	Zero or more, but no more than n. This is an optional relationship.
1..n	Zero or one object. The multiplicity value is used to indicate that there is a maximum of one object.
n,n	There are either exactly n objects or exactly n objects. This syntax can be used in combination with the other syntax. For example, 1..3,4,9..10 is valid syntax.
0..* or *	There may be zero or more objects. There is at least one object but there might be more.

Figure 7-3 Example Association With Navigation Arrows

- Determines the direction an association can be traversed of run time

Associations

- Relationship
- Role
- Multiplicity
- Navigation

Class diagrams

- Class Name
- Class attributes
- Class methods
- Generalization
- Attribute compartment
- Operation compartment

Figure 7-13 Step 5 – Record the Association Navigation

Modeling tools

- Forward engineering
- Reverse engineering

Principles

- Is used to create executable models
- Is a programming language
- Is a development process (methodology)
- Can be used to build code skeletons
- Serve as maps to most OO languages
- Can be used as a tool during the activities of a development process (methodology)
- Visual
- Object-oriented
- Can be used as a tool during the activities of a development process (methodology)
- Visual
- Object-oriented

Diagrams

Figure 7-4 Example of a UML Class Diagram

Summary Symbols (Class)

- Class
- Association
- Aggregation
- Composition
- Generalization
- Dependency (<<include>>/<<extend>> for Use cases)
- Realization

Chapter 1 | Software Engineering

Software disaster

- Ariane Project
- Radiation Overdose

Reasons behind software failure

- Doesn't meet client expectations
- Surpasses established budget
- Runs behind schedule
- Poor project management

Software systems

- Meet client expectations
- Are reliable
- Are efficient
- Are maintainable

Activities

- Process
- Product
- Project
- Process
- Product
- Project
- Process
- Product
- Project