Unified Modeling Language

Lecture 13

Sources: Learning UML 2.0 (by R. Miles and K. Hamilton)

What is UML?

- The standard modeling language for software and system development

- What to model? Why?
  - Abstraction - Manage the complexity of software or system
  - Share your design between stakeholders
  - Keep track of records

- What do you need for modeling?
  - A common language
    - Pseudo-code, actual code, pictures, diagrams, or long description, ...

Why UML?

- Formal
- Concise
- Comprehensive
- Scalable
- Built on lessons learned
- It is the standard
  - Maintained by open standards group with active contributions from vendors and academics
  - Transformable and interoperable

Bigger picture is missing entirely!
Models and Diagrams

- **Model = A set of diagrams?**
  - UML modeling is not just drawing diagrams
  - Rather, modeling is to capture a system as a model

- **Degrees of using UML**
  - Sketch – convey key points only. Throwaway use
  - Blueprint
    - Detailed specifications in UML
    - Use of CASE tools
    - Together with forward and backward engineering
  - Programming language
    - Directly from UML to executable code
    - Very detailed. Automatic transformation tools are needed.

UML 2.0 Diagram Landscape

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

UML 2.0

4+1 View Model (P. Krucheten)

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

Class, object, state machine, interaction

Logical View

Physical View

Process View

Development View

Deployment

Package and component
A First UML Example

- **Notes (comments)**
  - A Note

- **Stereotypes**
  - Special use or intent applied to an UML elements
  - Often, associated with special icons

Stereotypes (cont.)

- **Stereotype applied to classes**
  - utility
    - Represent a class that provides utility services through static methods (e.g., Java's Math class)

- **Stereotypes applied to components**
  - service
    - A stateless, functional component that computes a value (e.g., web service)
  - subsystem
    - A large component that is a subordinate of a larger system

- **Stereotypes applied to artifacts**
  - executable, file, library, source

Stereotypes and Tagged Values

- **Tagged values**
  - extra information related to the element they are applied.
  - Connected to stereotypes

Name of stereotypes in the note notation

Connected with dotted line with a Circle at the element end