CISC 322
Software Architecture

UML - The Unified Modelling Language
Thursday, September 16, 2010

Nicolas Bettenburg
DEFINITION

The Unified Modelling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components.
UML combines previous blueprinting approaches.

- Data Modelling
  - ER-Diagrams
- Business Modelling
  - Workflow Diagrams
- Object Modelling
  - Class Diagrams
- Component Modelling
  - Composition Diagrams
Why UML?

“Developing a model for an industrial-strength software system prior to its construction or renovation is as essential as having a blueprint for large building.”

ISO/IEC 19501

- Good models essential for team communication
- Comprehension of systems through visualization
- Reduce the risk of failure
- Industry standard - training, tools, meaning costs
“UML provides notation and semantics that addresses all scales of architectural complexity and across all domains.”

ISO/IEC 19501

- Ensure architectural soundness
- Solve recurring architectural problems (Patterns)
- Split up complex architectures in smaller parts
History

Jim Rumbaugh
IBM Rational

Grady Booch
IBM Rational

Ivar Jacobson
Objectory
1994  Booch and Rumbaugh unify BMT and OMT

1995  Jacobson joins Rational merging in OOSE

1996  UML v0.9  first specification

1997  UML v1.0  public, non-proprietary open

2005  UML v1.4 widely adopted in industry
      becomes international ISO standard.

Today  UML v2.3 released in March 2010
Types of UML Diagrams

http://www.omg.org/spec/UML/2.2/
Structure Diagrams

• Emphasize things that must be present in the system (Existence).
• Extensively used in documenting the architecture of a system.
• A Component Diagram describes how a software system is split up into components and shows the dependencies among them.
• A Package Diagram describes how a system is split up into logical groupings by showing the dependencies among these groupings.
In this course: simplified notation

Component A \rightarrow dependency \rightarrow Component B
In this course: simplified notation

Component A

Component B
In this course: simplified notation

Component A

- Sub-Component A1
- Sub-Component A2
- Sub-Component C3
- Sub-Component D1

- expected
- unexpected not logical
- unexpected but logical
Interactive Demo: Structure of a Web Browser
Behaviour Diagrams

- Emphasize things that must happen in the system (Functionality).
- Extensively used in documenting the functional behaviour of a system.
Interaction Diagrams

• Emphasize control and data flow in the system (Static and Dynamic Behaviour).

• A *Sequence Diagram* shows how objects communicate with each other. Also indicates lifetime and activity time of objects.
In this course: simplified notation

Component A
In this course: simplified notation

Component A
In this course: simplified notation

Component A

Component B
In this course: simplified notation
Interactive Demo: Behaviour of a Web Browser
You will be using the same notation in your project deliverables.