Introduction to UML

- **A Model** is created for the purpose of understanding something before building it
- **A modeling language** is nothing more than a convention for how we'll *draw our model on paper*

- Source code: Too-detailed language for modeling
- Natural Language: Too-verbose and ambiguous language for modeling
- To effectively model a system—avoiding confusion, ambiguity, and unnecessary details—we need a *formal modeling language*
- *UML*

How We Got to the UML

- OO modeling languages appeared in late 80's.
- As the usefulness of OO programming became undeniable, more OO modeling languages began to appear
- In 1994 the UML effort officially began as a collaborative effort between Booch and Rumbaugh.   Jacobson was soon after included in the effort
- The goal of UML is to be a comprehensive modeling language (all things to all people) that will facilitate communication between all members of the development effort

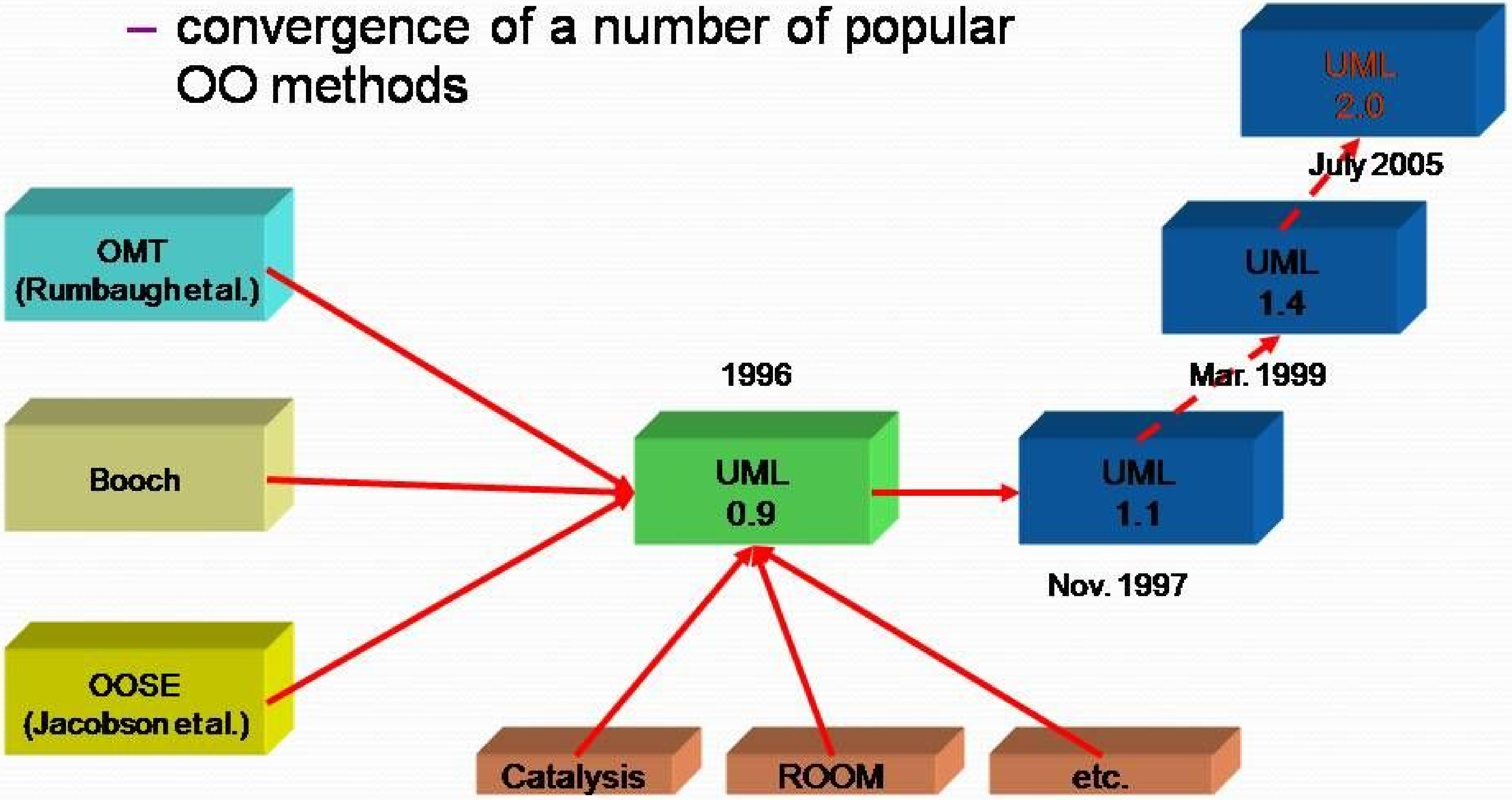
What is the UML?

- The Unified Modeling Language is a **family of graphical notations** that help in describing and designing software systems, particularly software systems built using the object-oriented style
- UML first appeared in 1997
- UML is standardized.
- Its content is controlled by the Object Management Group (OMG), a consortium of companies.

What is the UML?

- Unified
  - UML combined the best from object-oriented software modeling methodologies that were in existence during the early 1990's.
  - Grady Booch, James Rumbaugh, and Ivor Jacobson are the primary contributors to UML
- UML is not a development method by itself but it can be used with any leading development methods

- General-purpose OO *modeling* language
  - convergence of a number of popular OO methods



What is the UML?

- UML is a graphical language that follows a precise syntax.
- UML can be used for modeling systems ranging from enterprise information systems to distributed web based applications and even to hard real time embedded systems
- UML can be used to model non-software systems
- UML is a language for documenting design
    - Provides a record of what has been built
- Useful for bringing new programmers up to speed.

We will mainly follow UML 2.0

- UML is a language for
  - **Visualizing**
    - An explicit model facilitates communication
  - **Specifying**
    - Building models that are precise, unambiguous and complete
  - **Constructing**
    - UML models can be directly connected to a variety of programming language
    - Both forward and reverse engineering is possible
  - **Documenting**
    - Program versus Software

Program

- A program is an executable file residing in a disk file.

- The terms computer program, software program, applications program, or just **program** are used to refer to a collection of source code and libraries which have been compiled into an executable or otherwise interpreted to "run" in (active) computer memory ...

  - Wikipedia

- An organized list of instructions that, when executed, causes the computer to behave in a predetermined manner.

Software

"Software is

(1) Instructions (computer programs) that when executed provide desired features, function and performance

(2) data structures that enable the program to adequately manipulate information and

(3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs"

Roger S. Pressman

Three ways people apply UML …

- **UML as sketch**
  - Informal and incomplete diagrams (often hand sketched)
  - Created to explore difficult parts of the problem
  - Agile modeling emphasizes UML *as sketch*.
- **UML as blueprint**
  - relatively detailed design diagrams used either for reverse engineering to visualize and better understand existing code,
  - or for forward engineering to guide for code generation, either manually or automatically
- **UML as a programming language**
  - Complete executable specification of a sw system

- The UML may be used to:
  - Represent the Elements of a system or a domain and their Relationships in a Static Structure

    => *class and object diagrams*

  - Model the Behavior of objects

    => *State transition diagrams*

- The UML may be used to:
  - Reveal the Physical Implementation Architecture with *component & deployment diagrams*
  - Display the Boundary of a System & its major Functions using *use cases and actors*
  - Illustrate Use Case Realizations with *interaction diagrams*

A Conceptual Model of the UML

- Three types of building blocks:
  - Things
  - Relationships and
  - Diagrams.
- **Things** are the basic object-oriented building blocks of the UML
- **relationships** tie these things together
- **diagrams** group interesting collections of things

A Conceptual Model of the UML

- **Structural things**

  These are the nouns of UML models. These are mostly static parts of a model, representing elements that are either conceptual or physical.

  1. Classes
  2. Interfaces
  3. Collaborations
  4. Use cases
  5. Components
  6. Active classes and
  7. Nodes

A Conceptual Model of the UML

- **Behavioral things**

  These are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space.

  - Interactions (set of messages)
  - State Machine (sequence of states)
  - Activity (sequence of steps)
- **Grouping things**
  - Packages
- **Annotational things**
  - Notes

A Conceptual Model of the UML

- Relationships:
  - Dependency
  - Association
  - Generalization
  - Realization.
- Diagrams: class, object, use case, sequence, communication, state machine, activity, component and deployment.