Object-Oriented Analysis and Design using JAVA

B.Tech (CSE/IT) 5th SEM
2020-2021

Lecture-14 Introduction to UML

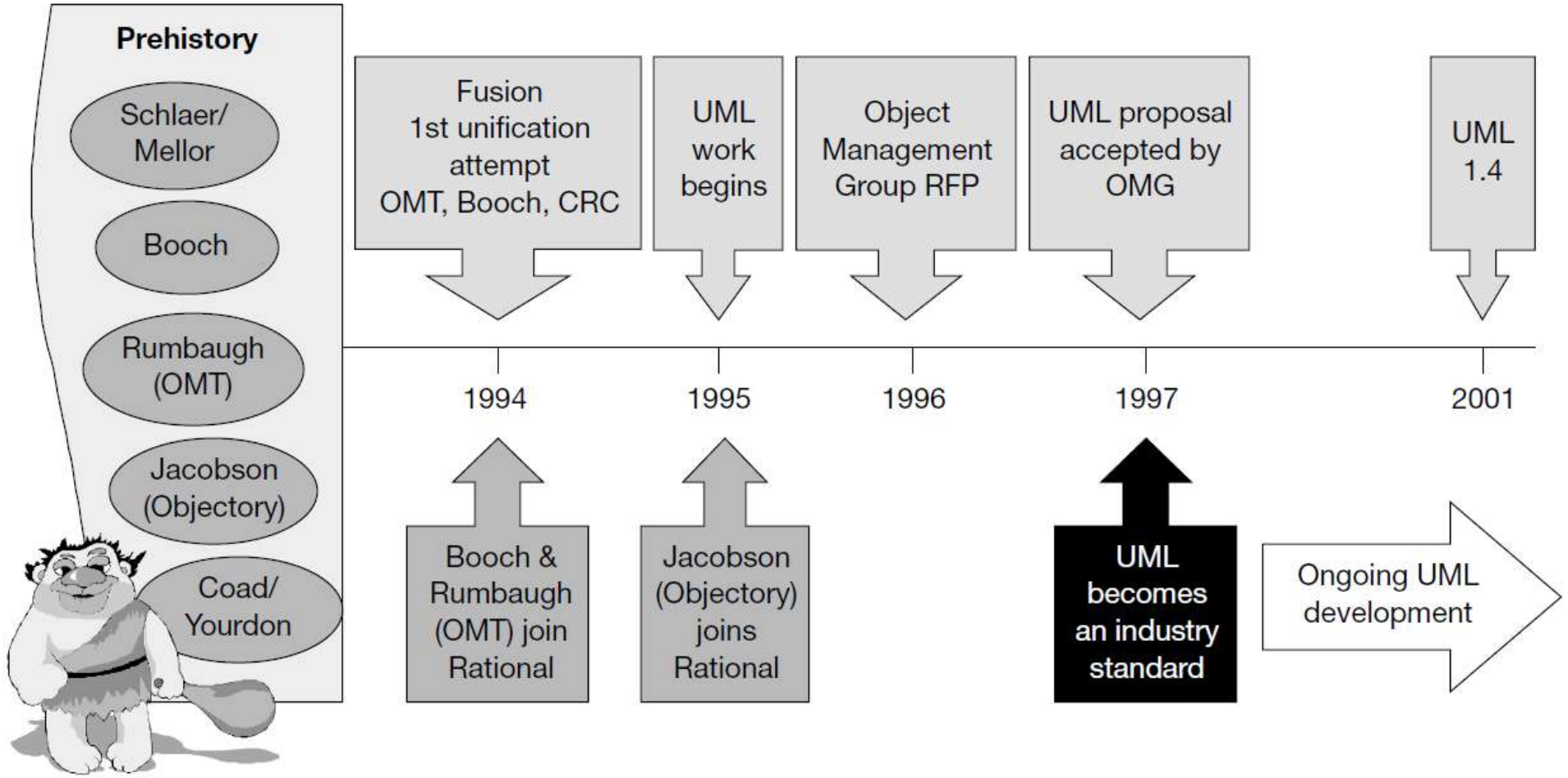
What is UML?

- The Unified Modeling Language (UML) is a general purpose visual modeling language for systems. Although UML is most often associated with modeling OO software systems, it has a much wider application than this due to its in-built extensibility mechanisms.

- It is important to realize that UML does *not* give us any kind of modeling methodology. Naturally, some aspects of methodology are implied by the elements that comprise a UML model, but UML itself just provides a visual syntax that we can use to construct models.

- UML is *not* tied to any specific methodology or lifecycle, and indeed it is capable of being used with all existing methodologies. Unified Process (UP) uses UML as its underlying visual modeling syntax and you can therefore think of UP as being the *preferred* method for UML, as it is the best adapted to it, but UML itself can (and does) provide the visual modeling support for other methods.

The birth of UML



Reference: **UML and the Unified Process** Practical object-oriented analysis and design-By-Jim Arlow , Neustadt

Why “unified”?

UML unification is not just historical in scope, UML attempts (and largely succeeds) in being unified across several different domains.

- Development lifecycle – UML provides visual syntax for modeling right through the software development lifecycle from requirements engineering to implementation.

- Application domains – UML has been used to model everything from hard real-time embedded systems to management decision support systems.

- Implementation languages and platforms – UML is language and platform neutral. Naturally, it has excellent support for pure OO languages (Smalltalk, Java, C#, etc.) but it is also effective for hybrid OO languages such as C++ and object-based languages such as Visual Basic. It has even been used to model for non-OO languages such as C.

- Development processes – although UP and its variants are probably the preferred development processes for OO systems, UML can (and does) support many other software engineering processes. Its own internal concepts – UML valiantly tries to be consistent and uniform in its application of a small set of internal concepts. It doesn’t (as yet) always succeed, but it is still a big improvement on prior attempts.

Objects and UML

The basic premise of UML is that we can model software and other systems as *collections of collaborating objects*. This is clearly a great fit with OO software systems and languages, but it also works very well for business processes and other applications. There are two aspects to a UML model.

- Static structure – this describes what types of objects are important for modeling the system and how they are related.

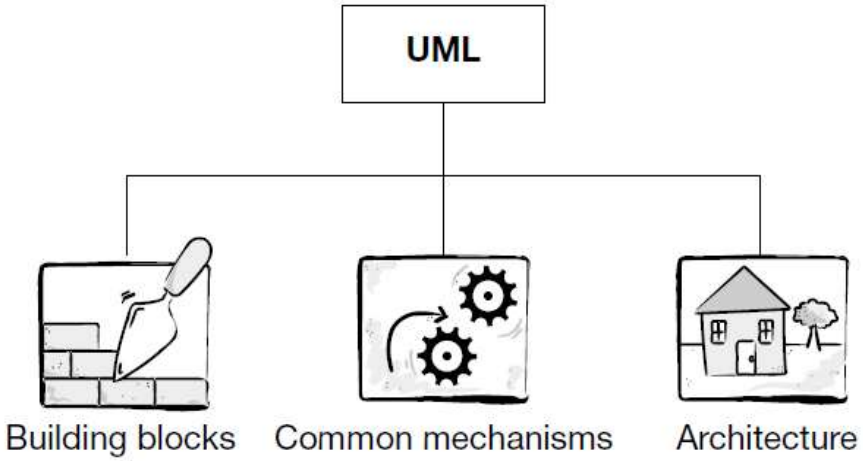
- Dynamic behavior – this describes the lifecycles of these objects and how they collaborate together to deliver the required system functionality.

These two aspects of the UML model go hand-in-glove, and one is not truly complete without the other.

UML Structure

UML structure consists of:

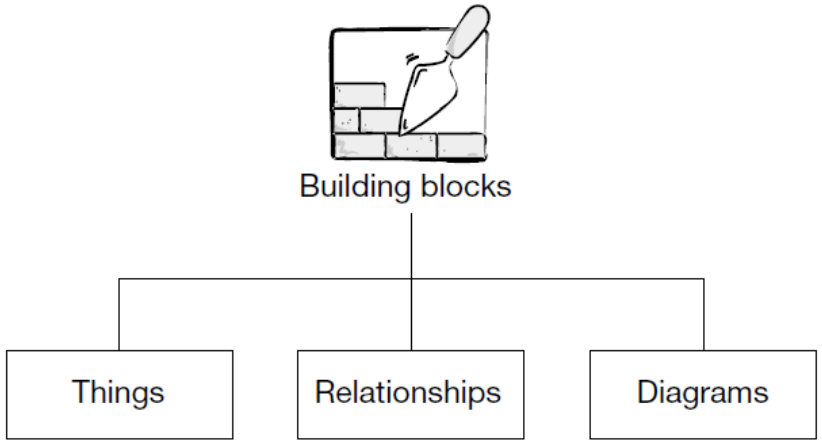
- Building blocks – these are the basic UML modeling elements, relationships and diagrams;
- Common mechanisms – common UML ways of achieving specific goals;
- Architecture – the UML view of system architecture.



UML building blocks

According to *The Unified Modeling Language User Guide*, UML is composed of just three building blocks:

- Things – these are the modeling elements themselves;
- Relationships – these tie things together – relationships specify how two or more things are semantically related;
- Diagrams – these are *views* into UML models – they show collections of things that "tell a story" about the software system and are our way of visualizing *what* the system will do (analysis-level diagrams) or *how* it will do it (design-level diagrams).



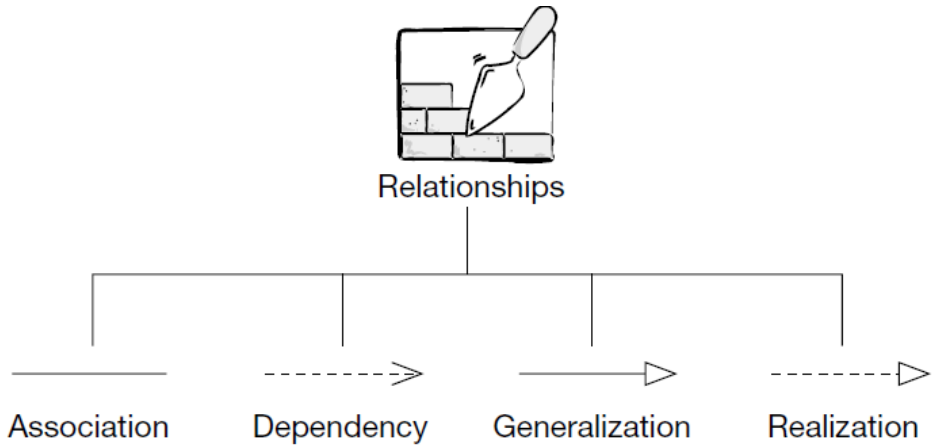
Things

UML things may be partitioned into:

- Structural things – the nouns of a UML model such as class, interface, collaboration, use case, active class, component, node;
- Behavioral things – the verbs of a UML model such as interactions, state machines;
- Grouping things – the package, which is used to group semantically related modeling elements into cohesive units;
- Annotational things – the note, which may be appended to the model to capture ad hoc information, very much like a yellow sticky note.

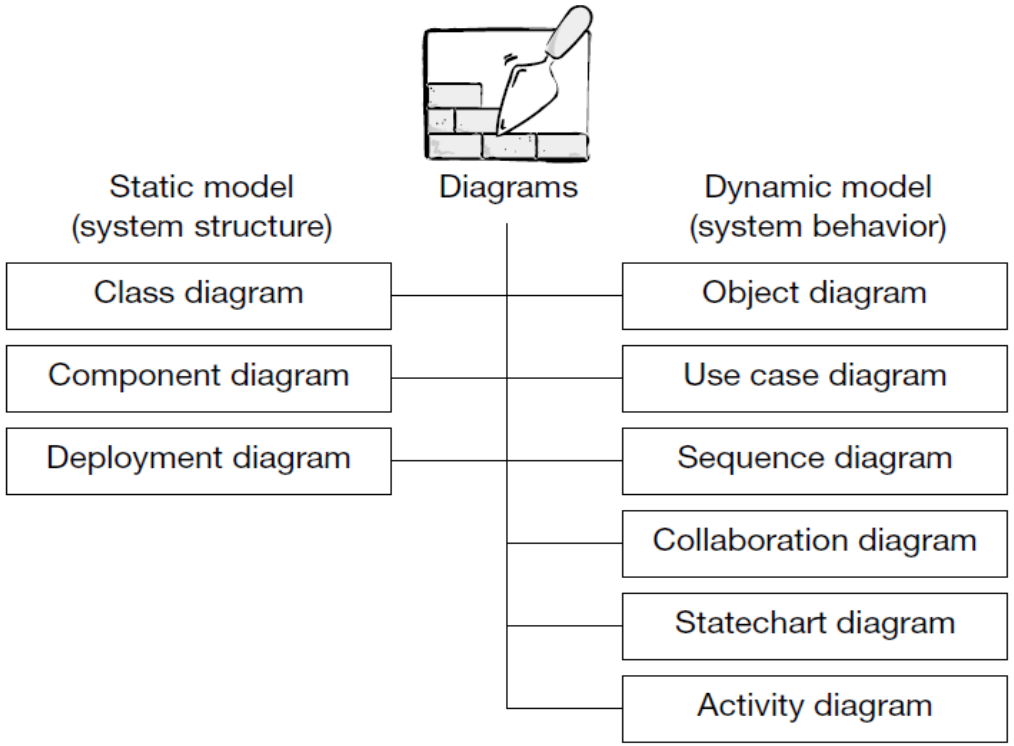
Relationships

Relationships allow you to show on a model how two or more things relate to each other. Thinking of families, and the relationships between all of the people in a family, gives you a pretty good idea of the role relationships play in UML models – they allow you to capture meaningful (semantic) connections between things. Relationships apply to the structural and grouping things in a model



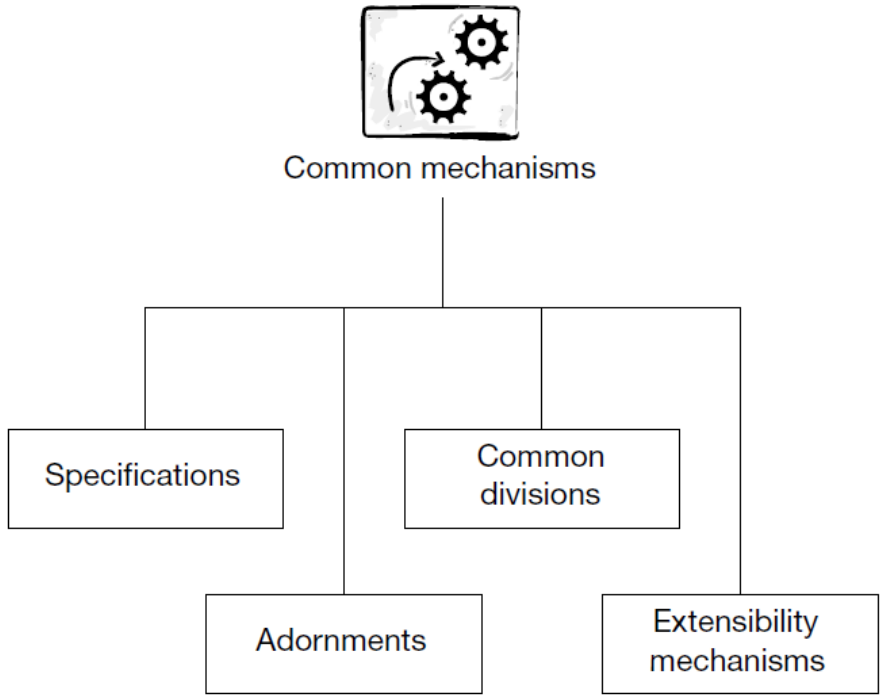
Diagrams

In all UML CASE tools, when you create a new thing or new relationship, it is added to the model. The model is the repository of all of the things and relationships that you have created to help describe the required behavior of the software system you are trying to design



UML common mechanism

UML has four common mechanisms that apply consistently throughout the language. They describe four strategies for approaching object modeling which are applied again and again in different contexts throughout UML. Once again, we see that UML has a simple and elegant structure



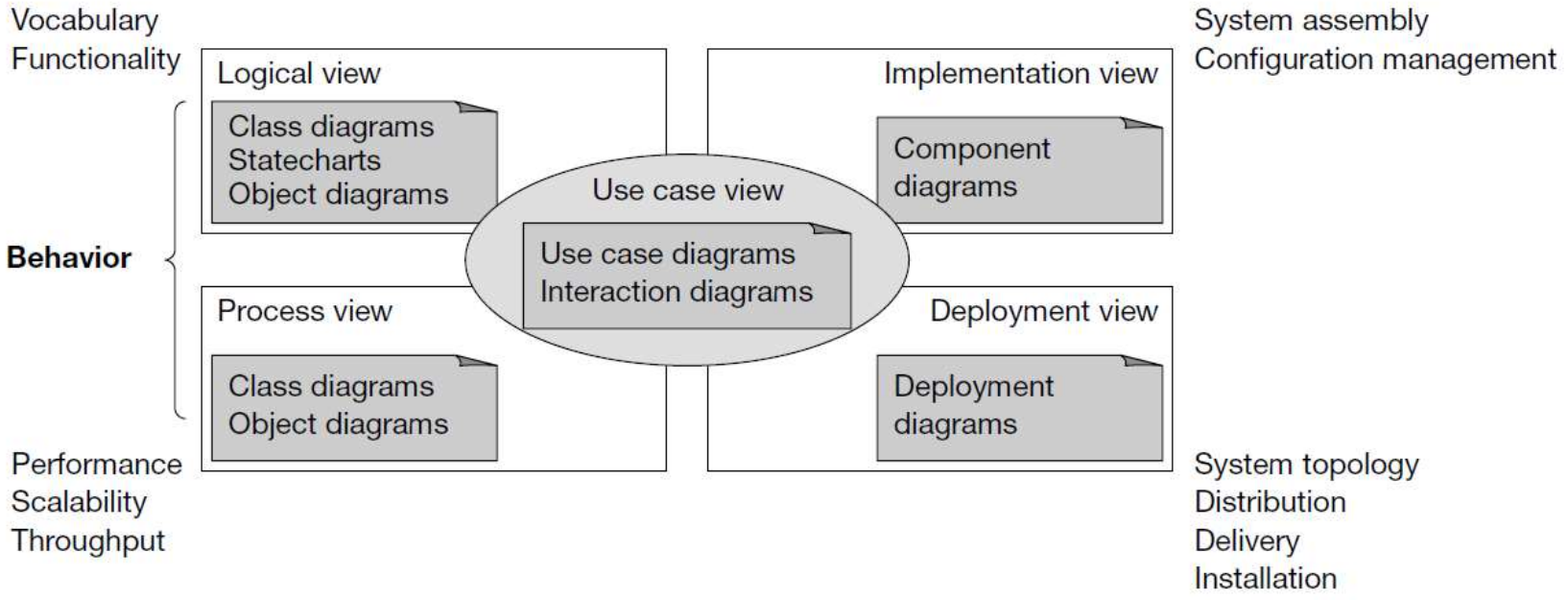
Reference: UML and the Unified Process Practical object-oriented analysis and design-By-Jim Arlow , Neustadt

Extensibility mechanisms: The designers of UML realized that it was simply not possible to design a completely universal modeling language that would satisfy everyone's needs present and future, so UML incorporates three simple extensibility mechanisms that we summarize in Table

UML extensibility mechanisms	
Constraints	These extend the semantics of an element by allowing us to add new rules
Stereotypes	A stereotype allows us to define a new UML modeling element based on an existing one – we define the semantics of the stereotype ourselves
	Stereotypes add new elements to the UML metamodel
Tagged values	These provide a way of extending an element's specification by allowing us to add new, ad hoc information to it

Software architecture with UML

The UML Reference Manual [Rumbaugh 1] defines system architecture as, “The organizational structure of a system, including its decomposition into parts, their connectivity, interaction, mechanisms and the guiding principles that inform the design of a system



Key references

UML and the Unified Process Practical object-oriented analysis and design-By-Jim Arlow , Neustadt

Thank You