diagrams represent the structure, they are used extensively in documenting the software architecture of software systems.

- Class diagram: describes the structure of a system by showing the system's classes, their attributes, and the relationships among the classes.
- Component diagram: describes how a software system is split up into components and shows the dependencies among these components.
- Composite structure diagram: describes the internal structure of a class and the collaborations that this structure makes possible.
- Deployment diagram: describes the hardware used in system implementations and the execution environments and artifacts deployed on the hardware.
- Object diagram: shows a complete or partial view of the structure of a modeled system at a specific time.
- Package diagram: describes how a system is split up into logical groupings by showing the dependencies among these groupings.
- Profile diagram: operates at the metamodel level to show stereotypes as classes with the <<stereotype>> stereotype, and profiles as packages with the <<profile>> stereotype. The extension relation (solid line with closed, filled arrowhead) indicates what metamodel element a given stereotype is extending.

Behaviour Diagrams

Behavior diagrams emphasize what must happen in the system being modeled. Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems.

- Use case diagram: describes the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies among those use cases.
- Activity diagram: describes the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
- state machine diagram: describes the states and state transitions of the system.

Interaction Diagrams

Interaction diagrams, a subset of behaviour diagrams, emphasize the flow of control and data among the things in the system being modeled:

- Sequence diagram: shows how objects communicate with each other in terms of a sequence of messages. Also indicates the lifespans of objects relative to those messages.
- Communication diagram: shows the interactions between objects or parts in terms of sequenced messages. They represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.
- Interaction overview diagram: provides an overview in which the nodes represent communication diagrams.
- Timing diagrams: a specific type of interaction diagram where the focus is on timing constraints.

UML Modelling Tools

To draw UML diagrams, all you need is a pencil and a piece of paper. However, for a software engineer that seems a little outdated, hence most of us will use tools. The simplest tools are simply drawing programs, like Visio or Dia. The diagrams generated this way look nice, but are not really that useful, since they do not include the code generation feature.

Hence, when deciding on a UML modelling tool (sometimes also called CASE tool)[9] you should make sure, that it allows for code generation and even better, it should also allow for reverse engineering. Combined, these two are also referred to as round-trip engineering. Any serious tool should be able to do that. Finally, UML models can be exchanged among UML tools by using the XMI interchange format, hence you should check that your tool of choice supports this.

Since the Rational Software Corporation so to say 'invented' UML, the most well-known UML modelling tool is IBM Rational Rose. Other tools include Rational Rhapsody, MagicDraw UML, StarUML, ArgoUML, Umbrello, BOUML, PowerDesigner, Visio and Dia. Some of popular development environments also offer UML modelling tools, i. e. Eclipse, NetBeans, and Visual Studio. [10]

References

1. http://www.omg.org/ Object Management Group
2. http://en.wikipedia.org/w/index.php?title=Unified_Modeling_Language&oldid=413683022 Unified Modeling Language