Diagrams in SRS

Various Unified Modelling Language Diagrams used for writing SRS

Arnav Dixit

***Abstract*—There are various types of diagrams required for representation, Unified Modelling Language(UML) allows us to represent these diagrams as needed and present the requirements and specifications. This paper discusses the various types of diagrams to write a SRS Report.**

Contents

I. Unified Modelling Language(UML)

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system[1].

UML 2 has many types of diagrams, which are divided into two categories. Some types represent structural information, and the rest represent general types of behavior, including a few that represent different aspects of interactions[2]. These diagrams can be categorized hierarchically as shown in fig. 1

These diagrams csn be categorised into two types[3] Behaviour Diagram and Structure Diagram.

II. Behaviour Diagrams:

They are used to represent the dynamic aspect of the system. It shows what must happen in the system being modeled. They are used widely to describe the functionality

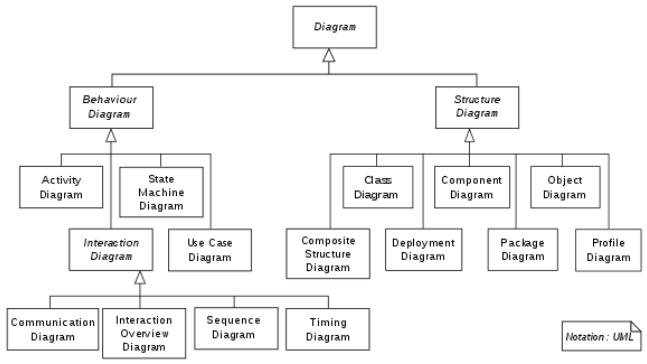

Figure 1: UML-Diagrams

of software systems. Behaviour Diagrams are further divided as:
1. Activity Diagram
2. Communication Diagram
3. Interaction Overview Diagram
4. Sequence Diagram
5. State Diagram
6. Timing Diagram
7. Use case Diagram

A. Activity Diagram:

Action diagrams are graphical portrayals of work processes of stepwise exercises and actions with help for decision, cycle and simultaneousness[4]. In UML, action outlines are planned to demonstrate both computational and hierarchical cycles, just as the information streams converging with the related activities[5][6].

B. Communication Diagram:

A Communication diagram models the interactions between objects or parts in terms of sequenced messages[7]. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.

C. Interaction Overview Diagram

The interaction overview diagram is similar to the activity diagram, in that both visualize a sequence of activities. The difference is that, for an interaction overview, each individual activity is pictured as a frame which can contain a nested interaction diagram[8]. This makes the interaction overview diagram useful to "deconstruct a complex scenario that would otherwise require multiple if-then-else paths to be illustrated as a single sequence diagram"[7].

D. Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario[5].

E. State Diagram

In UML, states are represented as rounded rectangles labeled with state names. The transitions, represented as arrows, are labeled with the triggering events followed optionally by the list of executed actions[5].

F. Timing Diagram

Timing diagrams are used to explore the behaviors of objects throughout a given period of time. A timing diagram is a special form of a sequence diagram[5]. The differences between timing diagram and sequence diagram are the axes are reversed so that the time increases from left to right and the lifelines are shown in separate compartments arranged vertically.

G. Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different[3] types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

III. Structure Diagrams:

Structure diagrams represents the static aspects of the system. It emphasize the things that must be present in the system being modeled. They are used extensively in documenting the software architecture of software systems.

Structural Diagram types:
1. Class Diagram
2. Component Diagram
3. Composite structure Diagram
4. Deployment Diagram
5. Object Diagram
6. Package Diagram
7. Profile Diagram

A. Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.

B. Component Diagram

A component diagram depicts how components are wired together to form larger components or software systems. They are used to illustrate the structure of arbitrarily complex systems[9].

C. Composite Structure Diagram

It shows the internal structure of a class and the collaborations that this structure makes possible. A composite structure is a set of interconnected elements that collaborate at runtime to achieve some purpose. Each element has some defined[3] role in the collaboration.

D. Deployment Diagram

A deployment diagram models the physical deployment of artifacts on nodes[2]. A deployment diagram would show what hardware components exist, what software components run on each node, and how the different pieces are connected together.

E. Object Diagram

An object diagram shows a complete or partial view of the structure of a modeled system at a specific time. The use of object diagrams is fairly limited, namely to show examples of data structure.

F. Package Diagram

A package diagram depicts the dependencies between the packages that make up a model. There are two special types of dependencies defined between packages, package import and package merge. A package import is a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace[2]. A package merge is a directed relationship between two packages, that indicates that the contents of the two packages are to be combined[2].

G. Profile Diagram

A profile diagram operates at the metamodel level to show stereotypes as classes and profiles as packages. The extension relation indicates what metamodel element a given stereotype is extending[10].

References

[1] G. Booch, J. Rumbaugh, and I. Jacobson, "Unified modeling language user guide, t he." Addison-Wesley Object Technology Series, 2009.

[2] O. A. Specification, "Omg unified modeling language (omg uml), superstructure, v2. 1.2," *Object Management Group*, vol. 70, 2007.

[3] "Unified Modeling Language," *Wikipedia*. Jan-2021.

[4] "Glossary of Key Terms," *MH Education*. Jan-2021.

[5] U. R. T. Force, "Omg unified modeling language specification, version 1.4 (final draft)." February, 2001.

[6] G. Booch, J. Rumbaugh, and I. Jacobson, "The unified modeling language reference manual," 1999.

[7] M. Fowler, *UML distilled: A brief guide to the standard object modeling language*. Addison-Wesley Professional, 2004.

[8] "UML Interaction Overview Diagrams," *Altova*. Jan-2021.

[9] D. Bell, "Uml basics: The component diagram," *IBM Global Services*, 2004.

[10] M. Fontoura, W. Pree, and B. Rumpe, *The UML profile for framework architectures*. Addison-Wesley Professional, 2002.