Which UML models are most applicable at different stages of the Software Development Life Cycle?

The software development lifecycle (SDLC) can be split into 6 stages:

1. Planning and Requirement Analysis
2. Defining Requirements
3. Design
4. Development
5. Testing
6. Deployment and Maintenance

The UMLmodels are primarily concerned with stages 1 to 3 and may be helpful for software developer to communicate high level design and functionality with clients. Booch, Jacobson and Rumbach (2005) categorise the UML diagrams into 4 groups:

1. Structural classification
2. Dynamic behaviour
3. Physical layout
4. Model management

It is recognised that there is a loose formality between the categorisation and use of the diagrams and they may be mixed.

1. Structural classification

Structural diagrams such as a class diagram, describe the objects in the system. These show the attributes and methods associated with the objects. These UML models are helpful to formulise the planning of a system and ensure that the correct objects have been captures and described. This would include the relationship between objects which may by components of a composite. The structural classification must also consider stage 2 of the SDLC and will introduce the actor to the system. The actor exists outside of the system itself but interacts with it. An example of an actor would be a customer buying an item from an online shopping system.

2. Dynamic behaviour

Dynamic behaviour is captured in UML diagrams such as the activity diagram or the sequence diagram. The purpose of these models is to communicate the actions that take place within the system and the change of the state for the objects over time. These can be viewed as a series of snapshots over time and are analogous to traditional flow diagrams. Thes models are particularly useful at the design stage of the model – prior to coding.

3. Physical layout

The physical layout and deployment UML models show how the system will be physical structured. For example, a system concerning a driverless would need software installed in the car as and communication to other hardware such as GPS systems, online map servers and smart road systems. Similarly, in commerce, it may be necessary for the system to include a series of sales terminals, within a series of shops which link to a warehouse stock control server. The physical layout UML models are vital for system design but are also a necessary part of the testing process, particularly when there are several interfaces.

4. Model management

The model management diagram is a model of the complete system. This can be shown from a variety of viewpoints. This is a high-level view which shows the model broken into packages alongside their dependencies. Each package contains a set of objects and data. Packages can include other packages. For example, a school management information system may contain a student package and this package may in turn contain other packages such as academic records, contact details or special educational needs support. The model management view is most helpful at the beginning of the SDLC.

The set of UML models have their greatest use in the planning and design phases, However, the class diagrams and dynamic are particularly helpful for coding and development ensuring that the desired system functionality is captured. State diagrams can aid successful testing prior to deployment.

The complete set of well designed UML models and diagrams will inevitably aid maintenance, particularly on large projects or those extending over prolonged time, where different software engineers will be maintain a complex system.

References

Rumbaugh, J., Jacobson, I. & Booch, G. (2005) *The Unified Moddeling Language Reference Manual. Second Edition.* Boston: Addison-Wesley

Glinz, M. (2000) 'Problems and deficiencies of UML as a requirements specification language', *Tenth International Workshop on Software Specification and Design.* San Diego, 07 November 2000. San Diego: IEEE. 11-22.