A Beginners Guide to UML Part I

Dan Brown, Dunstan Thomas Consulting
http://consulting.dthomas.co.uk

Summary

If you're an analyst, developer or architect, the chances are that you have heard of the UML (unified modelling language). If you're not already familiar of using OOA/D design methods such as the UML then there is a fair chance the pressure is on to utilize this standard as part of your analysis and design process.

To those not familiar with OOAD, either the laymen or seasoned developer alike, the UML can be seen overly comprehensive and daunting to learn. With the perceived value gained from this technique being far outweighed by the learning curve and upfront analysis involved.

This article will attempt to provide a fast track introduction for those that need to learn the UML basics and to begin to start understanding UML so that it can be incorporated into your development project. I shall begin by clarifying exactly what UML is and is not. Then question why we should use the UML at all. Then I will conclude part 1 of this article with a high-level tour of the UML modelling toolset. In part 2 of this article I shall continue by applying the models and notation already discussed to a real life business problem with a working example.

What is the UML?

In 1997 the OMG (Object Management Group) developed the UML as a common architectural framework for modelling object orientated systems and applications. The UML is derived primarily from the strengths of three notations; Booch OOD (Object-Oriented Design), Rumbaugh OMT (Object Modelling Technique), and Jacobson OOSE (Object-Oriented Software Engineering).
The OMG described UML is a language representing unified best engineering practices for specifying, visualizing, constructing, and documenting the elements of business modelling, software and even non-software systems.

- **Specification:** The UML can be used for specifying "what" is required of a system, and "how" a system may be implemented. It captures the all-important requirements, analysis, design, and implementation decisions that need to be established during a system development lifecycle.
- **Visualization:** The graphical nature of UML allows the visualization of systems before they are implemented. Using shapes representing well defined semantics, to communicate to a wider audience more succinctly than a descriptive narrative and more comprehensively than what often can be represented by a programming language.
- **Construction:** The UML can be used to guide and craft the implementation of a complicated system. Furthermore, with the aid various case tools on the market, its possible to generate object orientated source code from UML models, while it's also possible to reverse engineer source code into UML models.
- **Documenting:** The UML offers a means of capturing knowledge and documenting deliverables, such as requirements documents, functional specifications, and test plans. These are all critical in controlling, measuring, and communicating a system throughout its life cycle.

These four modelling applications of the UML should not be confused with a process. There are many processes available which use the UML; furthermore there are many tools available on the market that aid the construction of the UML and in some cases also facilitates the following of a particular process. Therefore the UML is not:

- **A Process:** It is a modelling toolkit with its own notation and syntax. A process goes further by describing the steps you take when developing software, which diagrams are produced and in which order, who does what and so on. The premise behind the UML is that it is process-independent, but enables and facilitates further processes.
- **Visual Programming Language:** It is a visual modelling language from which programs can be derived. The notation behind UML modelling is comprised of a set of specialized shapes used for the construction of different kinds of software diagrams, while the UML syntax specifies how these shapes can be combined.

Therefore further to learning the basics of UML it is recommend that:

- A process or methodology is adopted
- A UML development tool is utilised

UML may be used to support a number of methodologies, such as the Rational Unified Process. Some methodologies are more suited to larger enterprise applications with a large team of architects and developers. While others are more appropriate for a single person or small teams working on small embedded systems. In this article, I'm not going to assume nor advocate any fixed method of development.

Similarly there are many UML development tools available, such as Rational Rose (Rational Rose Corporation), Enterprise Architect (Sparx Systems), Describe (Embarcadero Technologies) and even Microsoft Visio.

Why Use UML?

With many of the rapid application development (RAD) tools available such as Delphi or Visual Basic, developing an application is fairly easy. Most people familiar with a drawing package can design and create forms and most people with a basic programming knowledge can double click on a control and enter some code. But does this method result in a professional quality application?

Deborh Kurata (1998) states that if an application is to be of a professional quality, it must:

- meet the needs of the users
- be robust
- be maintainable
- be documented

Many developers using RAD tools will believe it makes sense to develop an application rapidly. Write a prototype, and then keep adding more code until the application is complete. There is however, a fundamental problem to this approach. The resulting application will lack a well-defined architecture because it would not have been thought out properly. This will more often than not compromise fundamental object orientated principles and result in undocumented, inefficient and difficult to maintain code.

With the use of UML, an appropriate UML development tool, and an applicable process or methodology, the design and refining of the application is shifted from the development phase to an analysis and design phase. Therefore reducing risks and providing a vehicle for testing the architecture of a system before it is coding begins. The analysis and design overhead will eventually pay dividends as the system has been user driven, documented and when it's time to start developing, many UML tools will generate skeleton code that will be efficient, object orientated and promote re-use.

Sinan Si Alhir (1998) describes the UML as enabling:

"… the capturing, communicating, and leveraging of strategic, tactical, and operational knowledge to facilitate increasing value by increasing quality, reducing costs, and reducing time-to-market while managing risks and being proactive in regard to ever-increasing change and complexity."

This is a fairly convincing statement in itself, Sinan Si Alhir states that the UML will increase quality and reduce development time while being flexible enough to respond to changing requirements.

Furthermore, the use of UML will help;

- The communication of the desired structure and behaviour of a system between analysts,
- Architects, developers, stakeholders and users.
- The visualisation and control of a systems architecture
- Promote a deeper understanding of the system, exposing opportunities for simplification and reuse.
- Manage Risk.

So what are these models?

Hopefully now I have clearly defined UML and made a case for incorporating UML in your next project. So what models are available, what use are they and how do they link together?

Firstly we need to consider the primary modelling purposes of UML. These are:

- Business process modelling with use cases
- Class and object modelling
- Behaviour modelling
- Component modelling
- Distribution and deployment modelling

Each model is designed to let analysts, developers and customers view a system from different perspectives and with varying levels of abstraction. Each diagram will fit somewhere into these five architectural views representing a distinct problem solution space. These can be described as the; user model view, structural model view, behavioural model view, implementation model view and the environment model view.

The User Model View

The user model view encompasses the models which define a solution to a problem as understood by the client or stakeholders. This view is often also referred to as the use case or scenario view. The main model encompassed by this view is the:

- **Use case Diagram:** These models depict the functionality required by the system and the interaction of users and other elements (known as actors) with respect to the specific solution.

The Structural model View

The structural view encompasses the models which provide the static, structural dimensions and properties of the modelled system. This view is often also referred to as the static or logical view. Models applicable to this view include:

- **Class Diagrams:** These models describe the static structure and contents of a system using elements such as classes, packages and objects to display relationships such as containment, inheritance and associations.
- **Object Diagrams:** Depict a class or the static structure of a system at a particular point in time

The Behavioural Model View

These models describe the behavioural and dynamic features and methods of the modelled system. This view is often also referred to as the dynamic, process, concurrent, or collaborative view. Models applicable to this view include:

- **Sequence diagrams**: Describe timing sequence of the objects over a vertical time dimension. With interactions between objects depicted on a horizontal dimension.
- **Collaboration diagrams**: Describe the interactions and relationships between objects and sequences of a system organized in time and space. Numbers are used to show the sequence of messages.
- **State diagrams**: Describe the sequence, status conditions and appropriate responses or actions to conditions during the life of the objects within the system.
- **Activity diagrams**: Describe the methods, activities and resulting transitions after completion of the elements as flows of processing within a system.

The Implementation Model View

The implementation view combines the structural and behavioural dimensions of the solutions realisation or implementation. This view is often also referred to as the component or development view. Models applicable to this view include:

- **Component diagrams**: These depict the high level organisation and dependencies of source code components, binary components and executable components and whether these components exist at compile, link or run time.

The Environment Model View

These models describe both the structural and behavioural dimensions of the domain or environment in which the solution in implemented. This view is often also referred to as the deployment or physical view. Models applicable to this view include:

- **Deployment diagrams:** Theses models depict and describe the environmental elements and configuration of runtime processing components, libraries, and objects that will reside on them.

How do the models fit together?

After a high-level tour of the architectural views and diagrams available, it is important to remember once again that UML is a not a process, therefore there is no right or wrong order in which these models should be constructed. In practise, the only real pre-requisite to a model is a business process model, use case or a use case diagram. From then on in, a method of refinement on each model will often be used as many elements of the system will not become obvious until it is modelled from a different perspective. Therefore the activities of analysis (what are the objects?) and design (the allocation of behaviour) will be iterative and be a mutually complementary process.

Conclusion

We have taken a look at the origins and definition of the UML to provide a simplistic understanding of what it is, and what the UML can offer us. We have also examined how we can benefit from its use on our next development project and briefly explored the architectural views and models available and how these can link together.

In the concluding part of this article I shall apply the principles and models discussed and explored in this article to a real life business problem and development solution using example UML models where applicable.

References

Alhir, Sinan Si. The True Value of the Unified Modeling Language (UML)". Distributed Computing Magazine. DC Corp. July 1998.
Alhir, Sinan Si. UML in a Nutshell. O'Reilly and Associates, Inc. 1998
Alhir, Sinan Si. Understanding the Unified Modelling Language (UML). Methods & Tools. Martinig & Associates. April 1998.
Kurata, Deborah. Develop a Professional Application. Visual Basic Programmer's Journal. pp 83-86. March 1998.

Further Reading

UML Tutorial, Sparx Systems:
http://www.sparxsystems.com.au/UML_Tutorial.htm
What is UML, Embarcadero Technologies:
http://www.embarcadero.com/support/what_is_uml.asp
Introduction to OMG's Unified Modeling Language™. OMG:
http://www.omg.org/gettingstarted/what_is_uml.htm
Understanding the Unified Modelling Language. Sinan Si Alhir:
http://home.earthlink.net/~salhir/UnderstandingTheUML.PDF
Nine Tips to Incorporating UML into Your Project. Doug Rosenberg. SD magazine:
http://www.sdmagazine.com/documents/s=815/s=815/sdm0003z/0003z2.htm