

## What is UML?

---



This section provides an introduction to UML and covers principles and goals of UML as well as UML standards.

### Do I need UML?

A guy sits at a bar, chuckling to himself while he watches two women at a table try to explain the rules of baseball to a man using toothpicks, salt and pepper shakers, and sugar packets. "What a goofball," he thinks. "It's such a simple game, and he doesn't get it," and he turns back to the bar.

The man at the table asks a question about a subtle rule, and the guy at the bar realizes he doesn't know the answer, either. This game is more complicated than he realizes because he grew up just playing the game; he's never tried to explain it from scratch. He peeks over his shoulder to watch a sugar packet advance to second base as the pepper scores a run. "Oh, I see. Huh, that's interesting. I thought it would have been the salt at second."

Situationally, it is understood that the condiment menagerie represents assorted players. The mental picture formed from these objects clarified the problem and brought everyone to a common understanding.

This same principle holds true in the world of software development. Complex problems require explanations, and many times a graphic representation expresses a solution more elegantly and succinctly than text alone. In response to this idea, graphic notations were developed for several major development methods. Collaboration became difficult as methodologies were combined and translations of the software models became necessary.

Enter the 'Three Amigos': Grady Booch, Ivar Jacobson, and James Rumbaugh. All three had developed their own methods, but collaborated to combine them into the Unified Method, which evolved into the Unified Modelling Language (UML). With these initial UML releases, dozens of competing notations were replaced by the language- and method-independent UML.

The benefits of UML are undeniable, and adoption rates are sky-high. A study conducted in July of 2004 by BZ Research reported that over two-thirds of development managers say that UML is used within their organization.



## Introduction to UML

Systems being developed now are more complex than ever, and old software development methods simply do not efficiently scale up to the size of current systems. New paradigms are needed to keep up.

Engineers in other disciplines have long used blueprints and models to design and construct complex systems. They are concise, precise and allow the viewer to understand at a glance what is going on. They also contain an enormous amount of information. The standards used for blueprinting buildings are the same, a door or window is always rendered the same way. In the past, this was not the case with software blueprints. Notational languages were language and method specific, so that a class in one language could look completely different in a different notation.

Not so anymore. The Unified Modelling Language (UML) is a standard widely-adopted graphical language that describes the artifacts of software systems with a focus on conceptual and physical representations. It provides a good bird's eye view as well as the minute details of the structural and behavioural aspects of a single system through the various views offered by UML. It is proprietary and language independent so that it may be used in any number of development environments.

The Object Management Group (OMG) is the body responsible for creating and maintaining the language specifications. They define UML as, “a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software intensive system”. It is based on the UML Metamodel, which is a UML class diagram that specifies the syntactic and semantic characteristics of elements and relationships.

Current modelling trends involve models that can be translated into compilable and runnable code. This is known as Model Driven Architecture (MDA) and is also being regulated by the OMG. As UML is the most widely used modelling language, it is very closely linked with MDA.

The relationship between the code and the model is not one-way. Forward engineering takes the model and generates source code from it. Reverse engineering takes the source code and creates a model. Code can be coupled with models so that modifying one automatically updates the other in a process known as roundtrip engineering, effectively keeping the model and source code synchronized. Tight integration of the code and the model results in the best of both worlds – direct access to the code with all of the benefits of visual representations of that code.



Some of the biggest uses of UML include real-time embedded systems that deal with very complex problems and behaviours, business rules, game scenarios and grid computing, and patterns.

## A Brief History of UML

In response to the general agreement that a language to describe object-oriented notation was in order, many were developed. They happened to be tied very closely to particular development methods. While this may have solved some of the communication and planning problems within an organization, there was no guarantee that this information would be understood outside of that sphere.

Enter the “Three Amigos”: Grady Booch, Ivar Jacobson, and James Rumbaugh. All three had developed their own methods, but collaborated to combine them into the Unified Method. Version 0.8 was released in October 1995.

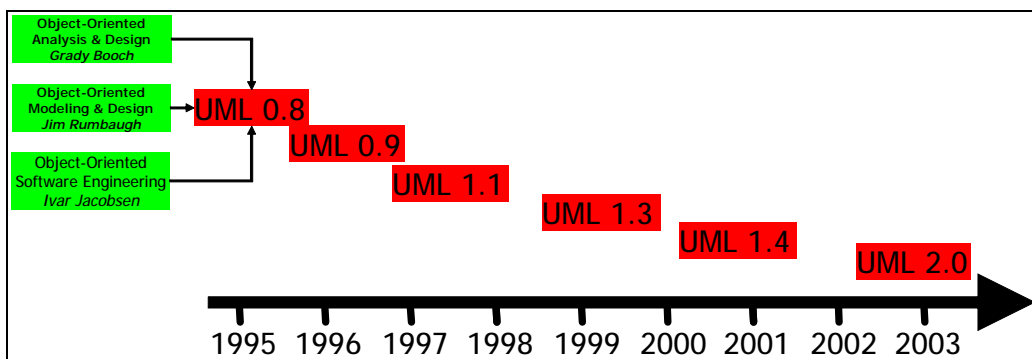


Figure 1: History of UML

In 1996, the OMG announced plans for a standard OO notation, and in June of 1996 UML version 0.9 was released. UML version 1.1 was adopted by the OMG in November of 1997. With these initial UML releases, dozens of competing methodologies were replaced by the language- and method-independent UML.

Several factors contributed greatly to the widespread adoption of UML.

1. UML is language independent.
2. It does not advocate nor require a particular method.
3. It is readily accessible as UML specifications are free for download and any company may join the OMG.

## Who uses UML?

Complex problems particularly require sufficient planning in order to avoid problems in development and more importantly in maintenance and update stages where changes are more arduous and costly. UML is one single language that allows people from different disciplines to work together to identify and solve these problems before they occur.

**Developers** and **business analysts** can map out requirements together in one language. There is no need to consider the underlying technology at this stage, which allows non-programmers to become more involved in the development process. Diagrams enhance the narrative and are stored electronically for simple archival.

UML is used in complex systems to not only capture object-oriented software information, but business rules information as well. From these business rules, a description of interfaces between systems and components can be generated. Often, this will require passing information between dissimilar languages and data formats, which makes language- and platform-independent UML a natural choice. The business models created form the framework for subsequent systems.

Lots of developers still use a 'code only' approach. Any modelling done is informal and/or code-based (such as using packages) and is often captured on whiteboards and left undocumented. This can cause severe problems when the system is expanded or the original developers are no longer working on the project.

With other people involved in the design process and a more efficient method of planning, developers are freed up for further architecting, analysis, design, and testing.

## Advantages of UML

Fundamental to communication is a common language, and if a common language does not exist we are left to create our own. Think of stories of travellers in a foreign country who, while attempting to ask the price of a loaf of bread with a combination of hastily learned travel phrases and some impromptu sign language, inadvertently say something very amusing (or perhaps offensive) to the shopkeeper. While this may provide a good anecdote for the merchant to share later on, it certainly does not help the customer to buy lunch or the merchant to sell it.

The inherent disadvantages to the lack of a common spoken language can also be seen in the increasing number of international software development teams throughout the world. These teams may sit in the same room or may teleconference from opposite sides of the globe, but the problem remains the same. And doesn't it sometimes seem like the people in the Business Department (or Software Development Department) down the hall speak another language? UML is the language that can be used to clarify and refine ideas that spoken language may not be adequate to cover. The graphical nature of the language makes it much more intuitive to learn and use.

Imagine the same bread-buying traveller driving in a foreign country – the signs may not be familiar, but there is enough information given from the context that it is often not hard to make the connection between the graphics and their meanings. Even those who do not drive and do not know the rules of the road can have a general impression of what is going on and effectively participate in conversations on this topic.

People who are new to UML may initially have some reservations about reading models, but there is a very short learning curve to becoming familiar with UML. The graphical nature of the language means that even those with very little knowledge of programming can participate in the development process, for example gathering requirements. Say Lester from the sales department has come up with an innovative new plan for the order-taking process for his company. Now he has the problem of trying to describe this to the software department. Lester has never taken a course in computing, and has no concrete idea of how to make this work. He does, however, have a very good idea of how he wants to interface with the new application. With just a few easily learned elements from the UML use case diagram, he can tell the software department exactly what he wants the program to do.

UML also makes it easy to focus discussions on the problem at hand while removing distractions. Details can be abstracted away, leaving only the essentials. For example, there is no need to worry about the operating system, language, or hardware used to implement the code. All of this is dealt with in a later stage, allowing architects to work at a higher conceptual level. The Platform Independent Model (PIM) is lasting because it can be applied to further technologies as they arrive. The PIMs developed may be used to create Platform Dependent Models, although in many cases the PIM is sufficient.

Because the PIM is not tied to anything implementation-specific, it is generic enough to lend itself to reuse. Sections of the model can be reused by completely different types of models. Graphic modules and components are easily understood, and can therefore be re-implemented with a minimum of study. All of this reinforces good design habits.

Good design is important in order to have maintainable and expandable code. A well known company using a UML approach ended up with a change in their project requirements at the last minute. The change took about one hour, but they estimate that the change using their traditional methods would have required 1–2 days to complete.

UML can also speed up hardware development. An elevator control manufacturer used UML sequence diagrams and state diagrams to debug logic without the need for the actual hardware to be present. Before using UML for this process, they were tied to testing on the hardware, meaning testing took place at a much later stage and required much more effort.

Although the initial planning state may take longer than with manual coding, development time overall is decreased. The more intensive initial planning stage decreases errors and maintenance in the future. These factors combined reduce costs.

A case study from an aerospace company of a client-server application found amazing results. In this project, 22,000 lines of code were manually generated before introducing UML. The model was used for initial code generation only. The pattern produced was 1,406 lines of code and 32,000 lines of code were automatically generated. The conclusion reached was that 1,000 person-effort days were saved in code-cutting, with a projected estimate of 22 times fewer errors.

A major electronics manufacturer enumerated the three main benefits they experienced with their UML adoption.

1. They found that common terminology increases communication.
2. Documentation is more complete and consistent across the development teams.
3. More design evaluation is conducted before the project is complete, leading to smoother integration.

## Integrating UML

While there is a small bit of initial effort needed to begin using UML, it is surprisingly easy to get started. Learning just a few bits of notation can provide the means to create a wealth of information within one or two diagram types.

It may be hard at first for developers to get used to doing lots of work up front without generating any code. But as has been demonstrated already, the upfront effort more than pays off in the end.

Start slowly to give yourself time to get used to the technology and to give sceptics time to see the merits. Use case diagrams are a great place to begin, as the symbols and relationships are quite intuitive. Once the basics are sketched out, you can go back and add more and more details. Even the simplest diagrams are very effective communication tools.

There is no need to tackle the entire UML specification at once. UML has been designed to allow you to use only the sections you need, and add the rest incrementally later. Should you not need a particular diagram or element, simply ignore it.

## UML Goals

The UML has seven goals:

1. To provide users with a ready-to-use, expressive visual modelling language for the development and exchange of meaningful models
2. To provide mechanisms for extensibility and specialisation in order to extend the central concepts
3. To be independent from specific programming languages and development processes
4. To provide a formal foundation for understanding the modelling language
5. To encourage further development in the OO tools market
6. To support higher-level development concepts including collaborations, frameworks, patterns, and components
7. To integrate best practices

## The Future of UML

Finalizations for the new UML 2.0 specifications are currently under way. They make the language more concise and descriptive. Several new diagram types have been added, while the existing ones have undergone some revisions.



But not to worry. Just as you can incrementally learn UML, so can you implement new features of UML 2.0. The old and new versions are not mutually exclusive, meaning that you are able to add elements of UML 2.0 to your existing models without having to completely reconstruct them.

With the promise of MDA, enhanced UML 2.0, the evidence to prove its efficacy, and the growing number of free UML tools available, there is no reason to wait to get started with UML. Your bottom line will thank you.

Some modelling tool vendors – including Gentleware – already support UML 2.0.



We have been looking at why UML is needed and described some principles and goals of UML as well as UML standards.