UML Unified Modeling Language

1. The Importance of Modeling.
2. Four Principals of Modeling.
3. The essential blueprints of a software system.
4. Object-Oriented Modeling.

Why we model.

A successful software organization is one that consistently deploys quality software that meets the needs of its users.

The primary product of a development team … is good software that satisfies the evolving needs of it users and the business.

The Importance of Modeling.

Unfortunately, many software organizations confuse "secondary" with "irrelevant."

To deploy software that satisfies its intended purpose, you have to meet and engage users in a disciplined fashion, to expose the real requirements of your system.

To develop software of lasting quality, you have to craft a solid architectural foundation that is resilient to change.

We build models to manage risk.

Modeling is a central part of all the activities that lead up to the deployment of good software. We build models to communicate the desired structure and behavior of our system. We build models to visualize and control the system's architecture. We build models to better understand the system we are building, often exposing opportunities for simplification and reuse. And we build models to manage risk.

Given the increasing demand to develop software quickly, development teams often fall back on the only thing they really know how to do well – pound out lines of code. It often seems that working harder is the proper reaction to any crisis in development. However, these are not necessarily the right lines of code, and some projects are of such a magnitude that even adding more hours to the workday is not enough to get the job done.

Unsuccessful Software Projects fail in their own unique ways, but all successful software projects are alike in many ways. There are many elements that contribute to a successful software organization: one common thread is the use of modeling.

Modeling is a proven and well – accepted engineering technique.

What is a model? A model is a simplification (abstraction) of reality.

A model provides a blueprint of a system. Models may encompass detailed plans, as well

as more general plans.

A good model includes those elements that have broad effects and omits those minor elements that are not relevant to the given level of abstraction.

Why do we Model?

There is one fundamental reason. We build models so that we can better understand the system we are developing.

Through modeling we achieve four aims.

1. Models help us to visualize a system as it is or as we want it to be.
2. Models permit us to specify the structure or behavior of a system.
3. Models give us a template that guides us in constructing a system.
4. Models document the decisions we have made.

Modeling is not just for big systems. However, it's definitely true that the larger and more complex the system, the more important modeling becomes for one reason:

We build models of complex systems because we cannot comprehend such a system in its entirety.

Through modeling, we narrow the problem we are studying by focusing on only one aspect at a time.

Divide a hard problem into a series of smaller problems.

Principles of Modeling

The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped. The right model will brilliantly illuminate the most wicked development problems, offering insight that you simply could not gain otherwise.

In software, the models you choose can greatly affect your would view. If you build a system through the eyes of a database developer, you will likely focus on entity-relationship models that push behavior into triggers and stored procedures.

Every model may be expressed at different levels of precision. Sometimes a quick and simple executable model of the user interface is exactly what you need; at other times you have to get down and dirty with the bits, such as when you are specifying cross-system interface.

The best models are connected to reality. All models simplify reality; the trick is to be sure that your simplification don't mask any important details.

No single model or view is sufficient. Every nontrivial system is best approached through a small set of nearly independent models with multiple viewpoints.

OBJECT-ORIENTED MODELING

Two ways to approach a model.

In software, there are several ways to approach a model. The two most common ways are from an algorithmic perspective and from an object-oriented perspective. The traditional view of software development takes an algorithmic perspective. In this approach, the main building block of all software is the procedure or function. This view leads developers to focus on issues of control and the decomposition of large algorithms into smaller ones. There's nothing inherently evil about such a point of view except that it tends to yield brittle systems.

An object-oriented approach the main building block of all software systems is the object or class. An object is a thing generally drawn from the vocabulary of the problem space or the solution space. A class is a description of a set of objects that are similar enough to share a specification.

Every object has identity, state and behavior.

Every object has identity – you can name it – state – generally some data associated with it – and behavior – you can do things to the object and it can do things to other objects.