Understanding the Unified Modeling Language (UML)

By Sinan Si Alhir

Sinan Si Alhir (salhir@earthlink.net, http://home.earthlink.net/~salhir) is a consultant and author of "UML in a Nutshell" (O'Reilly and Associates, Inc., 1998).

Published in "Methods & Tools" (April 1999) − An international software engineering digital newsletter published by Martinig & Associates.

Introduction

Organizations compete in a global market that is characterized by opportunities and risks where ongoing business and technological change fuel ever−increasing competition. Organizations must not only manage change and the complexity that results from adapting to change, but capitalize on the lessons learned, best practices, and knowledge gained through this evolutionary process. The importance and criticality of knowledge has given way to the Knowledge Revolution. This revolution can be characterized by the radical and fundamental paradigm shift that is occurring within the business and technology industries where an organization's application of knowledge defines its competitive advantage. Organizations can no longer only rely on physical muscle and size, but must harness intellectual capital and creativity to be successful. Knowledge takes on many forms, however, its value is demonstrated through its application and the realization of solutions to problems. If knowledge is captured and reapplied, an organization is enabled to become even more competitive and proactive, rather than reactive, to change and complexity; thus, increasing an organization's probability of success. This raises the fundamental question of how does an organization best capture, communicate, and leverage knowledge in order to gain a competitive advantage?

Historically, organizations have attempted various methods for procuring intellectual capital. Within the information system and technology industry, we have encountered the use of structured techniques to minimize the impacts of change and complexity, the use of Computer Assisted Software Engineering (CASE) tools to automate the development process, the use of business reengineering techniques to optimize organizational processes, the use of object−oriented techniques to facilitate reuse, the use of patterns to capture solutions to recurring problems, and the use of components to actualize reusable parts. Inherent to these techniques is the encapsulation of knowledge.

With the emergence of the Unified Modeling Language (UML) from Rational Software Corporation and the Object Management Group (OMG), it is very conceivable that such a language that unifies the many threads and incarnations of the Knowledge Revolution is the most viable means for organizations to best realize a competitive advantage via capturing, communicating, and leveraging knowledge. Rational Software Corporation and three of the most prominent methodologists in the information systems and technology industry, Grady Booch, James Rumbaugh, and Ivar Jacobson (the Three Amigos) originally conceived the UML. The UML emerged from the unification that occurred in the 1990s following the "method wars" of the 1970s and 1980s to gain significant industry support from various organizations via the UML Partners Consortium and be submitted to and adopted by the OMG as a standard (November 17, 1997).

The UML is a modeling language for specifying, visualizing, constructing, and documenting the artifacts of a system−intensive process.

- Within a system−intensive process, a method is applied as a process to derive or evolve a system.
- As a language, it is used for communication. That is, a means to capture knowledge (semantics) about a

subject and express knowledge (syntax) regarding the subject for the purpose of communication. The subject is the system under discussion.
- As a modeling language, it focuses on understanding a subject via the formulation of a model of the subject (and its related context). The model embodies knowledge regarding the subject, and the appropriate application of this knowledge constitutes intelligence.
- Regarding unification, it unifies the information systems and technology industry's best engineering practices across types of systems (software and non−software), domains (business versus software), and life−cycle processes.
- As it applies to specifying systems, it can be used to communicate "what" is required of a system, and "how" a system may be realized or implemented.
- As it applies to visualizing systems, it can be used to visually depict a system before it is realized.
- As it applies to constructing systems, it can be used to guide the realization of a system similar to a "blueprint".
- As it applies to documenting systems, it can be used for capturing knowledge about a system throughout its life cycle.

The UML is not:

- A visual programming language, but a visual modeling language.
- A tool or repository specification, but a modeling language specification.
- A process, but enables processes.

The UML is an evolutionary general−purpose, broadly applicable, tool−supported, and industry−standardized modeling language.

- As a general−purpose modeling language, it focuses on a set of concepts for acquiring, sharing, and utilizing knowledge coupled with extensibility mechanisms.
- As a broadly applicable modeling language, it may be applied to different types of systems (software and non−software), domains (business versus software), and methods or processes.
- As a tool−supported modeling language, tools are readily available to support the application of the language to specify, visualize, construct, and document systems.
- As an industry−standardized modeling language, it is not a proprietary and closed language but an open and fully extensible industry−recognized language.

The UML enables 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.

The Big Picture

To successfully leverage the UML, we must first understand the overall context in which the UML applies.

Problems, Solutions, and Problem Solving

Organizations produce and deliver products and services that address customer needs and requirements. Requirements may be characterized as problems (often referred to as as−is situations). Products and services that address requirements are characterized as solutions (often referred to as to−be situations). To deliver valued solutions, organizations must apply knowledge in problem−solving efforts; therefore, knowledge and the ability to apply it is the determining factor of success.

- Projects are problem−solving efforts that involve stakeholders and deliverables or work products in order to formalize the "work hard and hope for the best" approach to problem solving.
- Programs are collections of problem−solving efforts.
- Methods specify how to conduct problem−solving efforts.
- Processes are realizations of methods.
- Methodologies are taxonomies, or well−organized collections, of related methods.

The role of the UML is to enable and facilitate the following:

- Specifying, visualizing, understanding, and documenting problems.
- Capturing, communicating, and levering knowledge in problem solving.
- Specifying, visualizing, constructing, and documenting solutions.

However, the UML does not prescribe any particular problem−solving approach, but is very flexible and customizable to fit any approach. It enables and promotes (but does not require nor mandate) a use−case−driven, architecture−centric, iterative, and incremental process that is object oriented and component based.

- Use cases are used to manage and provide focus for a problem−solving effort.
- Architecture is used to manage complexity and maintain integrity and focus as a solution to a problem evolves.
- Iterations and increments are used to repeatedly apply a process to evolve a solution to a problem.

Fundamentally, the UML provides a means for addressing issues and risks concerning problems, solutions, and problem solving.

Problems and Solutions

Problems and solutions occur within a context (domain or space). The problem (system) must be understood in order to be solved. The solution (system) to a problem must be understood in order to be constructed and used. The solution must be organized (architecture) in order to facilitate its realization and adhere to the various constraints of the context in which it will be realized. To solve the problem, appropriate knowledge about the problem and solution must be captured (models), organized around decisions regarding the problem and solution (architectural views), and depicted (diagrams) using some language that enables it to be communicated and leveraged in the problem−solving process.

Therefore, the following concepts are critical to problems and solutions:

- Domains or spaces are organized collections of related elements in a self−contained situation or area of interest.
- Systems are organized collections of interacting and connected elements cooperating to accomplish a purpose.
- Architectures are schemes involving the structural and behavioral organization of systems within contexts (domains or spaces).
- Models are complete abstractions of systems or contexts. Abstraction involves focusing on those things that are relevant (essential) while avoiding those things that are irrelevant (incidental) to understanding something.
- Architectural views are abstractions of models.
- Diagrams are graphical projections of sets of model elements.

Problem Solving

Problem−solving approaches are organized (life cycles) to offer a management perspective and a development perspective. The two perspectives enable the effort to be managed and performed.

Problem solving requires being able to view the problem (paradigm) for the purpose of understanding it, and being able to view the solution (paradigm) for the purpose of realizing it. The problem−solving process involves leveraging knowledge to derive the solution (artifacts) to the problem through a series of (possibly concurrent) steps (activities) in which knowledge and rules of thumb (heuristics) gained from other problem−solving efforts may be used.

Therefore, the following concepts are critical to problems solving:

- Life cycles are collections of phases that divide efforts into several more manageable and controllable subordinate efforts.
- Paradigms are organized, self−contained collections of related components that form the basis for models.
- Artifacts are work products or deliverables resulting from efforts.
- Activities are efforts or collections of tasks directed at producing or developing artifacts. Workflows are collections of activities that are performed by specific roles.
- Heuristics are empirical or experience−based guidelines or rules of thumb.

System development may be characterized as problem solving, including understanding or conceptualize a problem, solving the problem, and implementing or realizing the solution. Conceptualizing a problem involves representing the problem using representational constructs (mental notions or ideas). Solving the problem involves manipulating representational constructs from the problem domain and the solution domain to derive a representation of the desired solution. Realizing a solution involves mapping those representational constructs of the solution unto the solution world, that is, constructing the solution. The use of representational constructs is a very natural process that often occurs subtly and sometimes unconsciously in problem solving. Underlying this scheme is the use of a paradigm in determining the possible types of representations utilized in problem−solving efforts.

- The object−oriented paradigm focuses on constructing reusable units and encompasses the conceptualization and specification principles of abstraction, encapsulation, inheritance, and polymorphism.
- The component−oriented paradigm focuses on the assembly of reusable units and encompasses the specification and realization principles of components, interfaces, and infrastructure.

The Unified Modeling Language

To successfully apply the UML, we must understand how the UML is holistically and cohesively organized to facilitate problem solving.

A language consists of a collection of concepts (semantics) with a notation (syntax) and rules (guidelines) governing the concepts and notation. Underlying a language and the methods that utilize a language is a foundation consisting of fundamental principles (axioms). These fundamental principles involve essential and "universally" accepted elements or constituents that define the constructs upon which a language is established (means) and facilitate some goals and scope to which the language applies (ends). The ends of the UML encompass models, architectural views, and diagrams to address why the UML exists. The means of the UML encompass the object−oriented paradigm and component−based development to address how the UML facilitates satisfying its ends.

The Architecture of the UML

To understand the architecture of the UML, consider how computer programs and programming languages are related. There are many different programming languages (C, C++, Java, Smalltalk, etc.), and each particular program is developed using a specific programming language. All of these languages support various declarative constructs for declaring data, and procedural constructs for defining the logic that manipulates data. Because a model is an abstraction, each of these concepts may be captured in set of related models. Programming language concepts are defined in a model called a metamodel. Each particular programming language is defined in a model that utilizes and specializes the concepts within the metamodel. Each program implemented in a programming language may be defined in a model called a user model that utilizes and instanteates the concepts within the model of the appropriate language. This scheme of a metamodel representing computer programming constructs, models representing computer programming languages, and user models representing computer programs exemplifies the architecture of the UML.

The UML is defined within a conceptual framework for modeling that consists of the following four distinct layers or levels of abstraction:

- The meta−metamodel layer consists of the most basic elements on which the UML is based −− the concept of a "Thing", representing anything that may be defined. This level of abstraction is used to formalize the notion of a concept and define a language for specifying metamodels.
- The metamodel layer consists of those elements that constitute the UML, including concepts from the object−oriented and component−oriented paradigms. Each concept within this level is an instance (via stereotyping) of the meta−metamodel concept "Thing". This level of abstraction is used to formalize paradigm concepts and define a language for specifying models.
- The model layer consists of UML models. This is the level at which modeling of problems, solutions, or systems occur. Each concept within this level is an instance (via stereotyping) of a concept within the metamodel layer. This level of abstraction is used to formalize concepts and define a language for communicating expressions regarding a give subject. Models in this layer are often called class or type models.
- The user model layer consists of those elements that exemplify UML models. Each concept within this level is an instance (via classifying) of a concept within the model layer and an instance (via stereotyping) of a concept within the metamodel layer. This level of abstraction is used to formalize specific expressions regarding a give subject. Models in this layer are often called object or instance models.

Within the fundamental UML notation, concepts are depicted as symbols and relationships among concepts are depicted as paths (lines) connecting symbols.

Models

Models capture the structural, or static, features of systems and the behavioral, or dynamic, features of systems. Models may be viewed via a small set of holistic but nearly independent and non−overlapping dimensions (aspects) that emphasize particular qualities of a model. The structural model dimension emphasizes the static features of the modeled system, and the behavioral model dimension emphasizes the dynamic features of the modeled system. Fundamentally, models capture knowledge (semantics).

Architectural Views

Architectural views organize models and knowledge around specific sets of concerns (architectural focus). The UML provides the following architectural views regarding models of problems and solutions:

- The user model view encompasses a problem and solution as understood by those individuals whose problem the solution addresses. This view is also known as the use case or scenario view.
- The structural model view encompasses the structural dimension of a problem and solution. This view is also known as the static or logical view.
- The behavioral model view encompasses the behavioral dimension of a problem and solution. This view is also known as the dynamic, process, concurrent, or collaborative view.
- The implementation model view encompasses the structural and behavioral dimensions of the solution's realization. This view is also known as the component or development view.
- The environment model view encompasses the structural and behavioral dimensions of the domain in which the solution is realized. This view is also known as the deployment or physical view.
- Other model views may be defined and used as necessary. An architectural focus is defined by a set of concerns (particular to stakeholders). An architectural view is defined by the set of elements from a model that address an architectural focus. For example, security issues may define an architectural focus. A security architectural view includes the set of elements from a model that address security issues.

Fundamentally, architectural views organize knowledge in accordance with guidelines expressing idioms of usage.

Diagrams

Diagrams depict knowledge in a communicable form. The UML provides the following diagrams, organized around architectural views, regarding models of problems and solutions:

- The User Model View
    - Use case diagrams depict the functionality of a system.
- The Structural Model View
    - Class diagrams depicts the static structure of a system.
    - Object diagrams depict the static structure of a system at a particular time.
- The Behavioral Model View
    - Sequence diagrams depict an interaction among elements of a system organized in time sequence.

    - Collaboration diagrams depict an interaction among elements of a system and their relationships organized in time and space.
    - State diagrams depict the status conditions and responses of elements of a system.
    - Activity diagrams depict the activities of elements of a system.
- The Implementation Model View
    - Component diagrams depict the organization of elements realizing a system.
- The Environment Model View
    - Deployment diagrams depict the configuration of environment elements and the mapping of elements realizing a system onto them.
- Other diagrams may be defined and used as necessary.

Fundamentally, diagrams depict knowledge (syntax).

Modeling Mechanisms

Mechanisms are practices for approaching modeling and diagramming that enable the creation of more precise and communicable models.

- Perspectives define a particular point of view from which to draw or read a diagram. They enable clear viewpoints to be associated with diagrams, and are used to heighten the effectiveness of the communication.
- Dichotomies define how something may be viewed from two different perspectives. They enable something to be viewed from multiple perspectives, and are used to discover inconsistencies within models.
- Layers or levels of abstraction define a particular level of abstraction and establish a level of detail at which attention and concentration are focused regarding a subject (problem or solution). They enable focused communication, and are used for organizing all of the diagrams pertaining to a single model into a coherent body of knowledge.
- Extension mechanisms define the means for customizing and extending the UML. They enable the UML to be flexible and adaptive, and used to ensure that the UML will evolve rather than be redefined to meet changing needs and requirements.

Within a problem−solving process, knowledge regarding a problem and solution is captured, organized around decisions, and depicted using the UML so that it can be communicated and leveraged. When deciding what diagram to use for communicating, consider the question or questions the communication is addressing, and what diagram or set of diagrams most effectively communicate the response. This decision centers on what dimensions of a model are to be emphasized in the response. Fundamentally, each diagram type emphasizes different dimensions of a model.

And within a problem−solving process, it is the underlying method that suggests how knowledge is utilized to realize a solution to a problem. This includes suggesting which diagrams to use and the perspective and the level of abstraction used to render and interpret these diagrams. Methods should be considered as suggestions and recommendations that organize and facilitate the problem−solving process rather than being considered rigid and inflexible rules that restrict the art of problem solving.

Conclusion

Conclusively, the Unified Modeling Language is an evolutionary general−purpose, broadly applicable, tool−supported, and industry standardized language for specifying, visualizing, constructing, and documenting the artifacts of a system−intensive process. It is a fundamental communication mechanism that empowers organizations to capture, communicate, and leverage strategic, tactical, and operational business and technological knowledge on an enterprise−wide scale. Such knowledge can be applied to improve value by increasing quality, reducing costs, and reducing time−to−market while managing risks and being proactive to ever−increasing change and complexity.

By understanding the overall context in which the UML applies and how the UML is holistically and cohesively organized to facilitate solving problems, we have reviewed the required foundation for strategically determining how to successfully apply the language to maximize its benefits. However, caution should be emphasized. Simply because the UML evolved primarily from various second−generation object−oriented methods, the UML is not simply a third generation object−oriented modeling language. Its scope extends its usability far beyond its predecessors. And it is experience, experimentation, and gradual adoption of the standard that will reveal its true potential and enable organizations to realize its benefits.
