The Foundation of the Unified Modeling Language (UML)

By Sinan Si Alhir (August 7, 1998)
Updated August 9, 1998

Abstract

The Unified Modeling Language (UML) is a modeling language for specifying, visualizing, constructing, and documenting the artifacts of a system−intensive process. It was originally conceived by 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). The language has gained significant industry support from various organizations via the UML Partners Consortium and has been submitted to and approved by the Object Management Group (OMG) as a standard (November 17, 1997).

This paper elaborates on the *foundation* of the UML.

Contents

Introduction

The seemingly simple expression `1+2` requires awareness of the following constructs in order to be understood:

- The symbols `1`, `2`, and `+`.
- The concepts `one`, `two`, and `addition` that the symbols depict.
- The rules that govern the concepts and symbols.

The body of knowledge encompassing these constructs is the calculus commonly known as *arithmetic*. A *calculus* is a collection of rule−governed symbols that facilitate reasoning about something. Arithmetic defines the Arabic numerals with the addition, subtraction, multiplication, and division symbols, and associated rules to facilitate reasoning about problems and solutions involving numeric values. A *formal language* is an uninterpreted calculus where the mechanical application of rules to a symbolic representation of a problem yields the appropriate solution. An uninterpreted calculus involves the application of rules without knowing why the solution is derived or grasping the semantics or rational of the symbols themselves. An *artificial language* is an interpreted calculus where the

symbols are interpreted via the concepts they represent. An interpreted calculus involves the application of rules by knowing why the solution is derived and grasping the semantics or rational of the symbols themselves. A *natural language* is an ordinary human language such as English, Arabic, Russian, etc.

Fundamentally, 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*). That is, a language is created for a reason or purpose.

Other foundations may be similarly viewed and understood. Consider the following example:

- Money is the *foundation* underlying our economic reality.
- Money facilitates the exchange and attainment of goods and services (ends).
- Money defines physical articles that are a medium of exchange and measure of value (means).
- Conclusively, money is the *means* for pursing various *ends* within an economic reality.

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* to address *how* the UML facilitates satisfying its ends. A brief comparison between natural languages and the UML exemplifies the foundation elements of the UML:

Natural Language	***UML***	***Definition and Significance***
Alphabet	Symbol Fragment or String Character	The primitive parts of a language (letters, characters, signs, marks).
Word	Symbol (Node or Path) or String	The fundamental unit of meaning.
Sentence	Diagram Fragment	Grammatical unit of meaning containing a subject and a predicate (proposition).
Paragraph	Diagram	Unit of thought containing grammatical units of meaning about a subject (or related subjects).
Section	Architectural View	Organized units of thought.
Document	Model	Organized knowledge.

It is imperative that the foundation of the Unified Modeling Language (UML) be understood such that the UML can be successfully applied.

Return to Contents

Object Orientation

Object orientation is a concept−centric paradigm encompassing the following pillars (*first principles*): *abstraction*, *encapsulation*, *inheritance*, and *polymorphism*. Because the object−oriented paradigm is derived from the convergence of other fundamental paradigms, it is reducible to the other paradigms as required by its application via a language or method.

Therefore, because the UML is based on the object−oriented paradigm, it is flexible enough to facilitate and express artifacts within approaches based on other paradigms (functional, data, responsibility, etc.).

Return to Contents

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).

Return to Contents

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.

Return to Contents

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* depict 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 the *specification* of behavior.
  - *Collaboration diagrams* depict the *realization* of behavior.
  - *State diagrams* depict the *status conditions* and *responses* of participants involved in behavior.
  - *Activity diagrams* depict the *activities* of participants involved in behavior.
- The Implementation Model View
  - *Component diagrams* depict the *organization* of solution components.
- The Environment Model View
  - *Deployment diagrams* depict the *configuration* of environment elements and the mapping of solution components onto them.
- Other *diagrams* may be defined and used as necessary.

Fundamentally, diagrams *depict* knowledge (syntax).

Return to Contents

Conclusion

Something is *necessary* if it is required, and *sufficient* if it, by itself, is enough to satisfy a given purpose. Because the foundation of the UML constitutes the *necessary* and *sufficient* engineering practices for problem solving, processes that utilize the UML are assured of resting upon a foundation that provides the *potential* for success.

The success of the UML will be measured by its appropriate use on successful projects. The UML does not guarantee success, but establishes a suitable foundation for success.

Return to Contents

References

[Alhir]

Sinan Si Alhir. "UML in a Nutshell : A Desktop Quick Reference". O’Reilly Associates, Inc., 1998.

Return to Contents

Email salhir@earthlink.net
Web Site home.earthlink.net/~salhir