A UNIQUE FORMALISM FOR THE MULTILINGUAL SWDPS

DR. MARIA RAFFAI PH.D.
Professor at Szechenyi Istvan University, Hungary
☎ *Tel/Fax: (+36) 96-613-525* 💻 *E-Mail: raffai@sze.hu*
HomePage: http://rs1.sze.hu/~raffai

As the changing circumstances force the organizations to use the most up-to-date Information Technologies (IT) and to work in multinational environment is an urgent need appears to deal with the key factors of the effective software development projects. Analyzing the results of the Software Development Projects (SwDP) of the last several years, we have to state, that most of them are failed, to be more precise: 76% of the SwDPs could not reach the user's goals. The main reason is, that the developers do not co-operate enough with the users, and they do not perform the adequate change and configuration management, the quality assurance, the testing processes or the other well-known development and control practices. From the experiences in project management and from the publications of the scientific results in this context, we learned that the SwDP's success depends also on the right scope, the strategy, the appropriate methodology and most of all the effective co-operation/communication among users, developers and project members.

The communication problem presents itself even in the case of international projects since the members of these projects are coming from different countries, speaking different languages and having different culture in the background, and they know and work on different computer platforms. In my presentation I intend to point to the important role of the standard visualization techniques that gives great help in communication and understanding.

1. BEYOND THE SOFTWARE DEVELOPMENT PROJECTS

The software development professionals are responsible for providing the functionality and usability of the applications and also *for delivering them on time, on budget* and often *with zero defects.* But at the beginning the user's requirements are usually unknown or estimated and can be satisfied only with a disciplined and pragmatic development process. In particular, an effective process must support the continuous integration, must use standard solutions and tools in order to realize a successful communication among the project members [4].

This *evolution process* forces the enterprises to use effective solutions and *to fit* themselves into the global e-economy: the so-called *new economy*. Both from theoretical and practical aspects, *the organizations need to define the problem-space, to model the business domain and the Corporate IS* they must *apply reusable components* and *computer-aided tools for planning and implementing* applications*.* As it makes a high demand to find an appropriate philosophy for the real world's abstraction as the user-analyst co-operation needs tools for understanding and communication, as the developers are coming from different cultural and linguistic environment we need to develop and apply new technologies and techniques.

From my more then two decades of experience in the field of software development processes, I can support the above-mentioned statement. In the last years I have definitely learned, that the achievements and the efficiency of the SwD projects depend to a great extend mainly on the following factors: (1) the expert-knowledge of the project members, (2) the talent and the communication skills of the project manager, the analysts and the use case designers, (3) the methods and tools applied in the project-management and the software development process and (4) last but not least the communication and understanding of all the actors taking part from all sides of the projects. My presentation focuses mainly on solutions, which lead to understand and specify the user's expectations and then on the interests of IT project managers and developers who want to create highly effective teams for the SwDPs. I intend to point to the most important aspects of the concepts and tools for managing software development projects, and to show *a unique solution* what *helps* the participants of the project *in communicating* and *in better understanding* each other [6].

1.1. The Model Driven Concept

Developing a model for an industrial-strength software system prior to its construction or renovation is a well-considered *abstraction process.* The good models are essential for communication among project team members and for assuring architectural soundness. As the complexity of systems increases, so does the importance of the effective modeling techniques. The basic target of building models is to know and understand the structure and the behavior of the system, its components and their relations and to give a precise description of it.

A *model* is a simplification of the reality, a blueprint of a system. It is the result of an abstraction process, which reflects the general, essential and permanent features from the modeling target's view. It is a formal specification to describe the functionality, the structure, and/or the behavior of the system. Modeling is a proven and well-accepted engineering technique. A good model includes those elements that have broad effect and omits those minor elements that are irrelevant to the given level of abstraction. But the reality may be described from different aspects, and these model views are therefore semantically closed abstractions of a system. A model may be *structural,* emphasizing the organization of a system, or it may be *behavioral,* emphasizing the dynamics of the system. Through modeling the specialists intend to achieve four main goals:

- to specify the structure and the behavior of the system,
- to describe and visualize the existing and the designed system,
- to make possible to document the process, the results and the decisions and
- to give a template for constructing and implementing systems.

The modeling tools help to create technology-neutral designs that are then transformed into the platform independent models.

1.2. The Object-Oriented Paradigm

The object-oriented (OO) paradigm has become a dominant force for today in the computing world. According to a recent survey conducted by International Data Corporation (IDC 2000) more than 80% of the development organizations are expected to use object technology as the basis for their distributed development strategy. The object technology is a powerful modeling paradigm with important mechanisms for handling complexity, any process design and redesign. The OO concept makes possible to design applications in the context of the objects which are faithful modeled components of the realty, and which can be later reused partially or wholly. Thus a new object must never be designed and created from scratch again, since there exists probably already an object from which it can be *derived* [9]. The advantages of the Object Technology (OT) include improved software quality, economic gains profiting from the reusable objects and components, shorter project development life-cycle, and the creation of truly distributed software across the different platforms. The object-orientation recommends several approaches to deal with heterogeneous nature of reality. Thus, a diversity of options is required in order to capture the diversity of reality [3]. Let us see the main features of the object technology (OT) from the aspects of its key ideas:

- The basic elements of the OT are *objects, messages* and *classes*.
- The mechanism is behind the *encapsulation, polymorphism* and *inheritance*.
- The advantages of OT lie in the *increased productivity, quality* and *adaptivity*.
- The innovations and extensions added to OT in the last few years are the *handling of interfaces*, the paradigm of *delegation* and *distribution*.
- The factors for succeeding object technology are defined in *motivation, education* and in *determination*.

1.3. The Role of Visualization

In the middle of the eighties the scientists and the world-leading software companies (with my considerable participation) established a Consortium in order to work out a language being suitable for visualization. The language that is called Unified Modeling Language (UML) helps the innovation projects to focus on fitting the changing environments and with its visualization capability is suitable

- to describe every aspects of the business domain,
- to study management's efforts and
- to analyze the existing system and comprehend its behavior.

The UML was first accepted as a modeling standard in 1997 by the OMG (Object Management Group) in order to give an effective analyzing and designing tool for the developers and to put a communication technique at the project members' disposal.

2. THE UNIFIED MODELING LANGUAGE

The UML is *a human-readable graphical/textual notation* which, by its visualizing capability, is primarily for analyzing, understanding and specifying the technology-independent business domain, for mapping the business processes, for designing, constructing and documenting artifacts of the developed software-system. The UML as an industry-wide breakthrough for visual modeling satisfies the newly emerged requirements, based on experiences of the users and the specialists of the SwDPs. The most important features are: usability, component-based development, executability and configurability. The UML as a well-defined and widely accepted modeling language includes (1) *model elements:* fundamental modeling concepts and semantics, (2) *notations:* visual rendering of model elements and (3) *guidelines:* idioms of usage within the trade.

2.1. The Language Features

The UML is a language that defines rules for combining words and provides a vocabulary for the purpose of the communication. For software intensive systems a language is required for addressing the different views of the system's architecture and behavior through the software development life cycle. Though a modeling language is not capable to perform the abstraction process aimed to reveal the business situation, the vocabulary and rules are still great help in describing the main features, and in creating, reading and manage well formed models [6]. Henceforth let us see what is the UML for, and how can it satisfy the user's requirements!

A language for visualization

As the developers are working on complex systems and in different teams, they have to face several problems that should be solved. (1) The main problem is that during the communication of the different model views to others there can be misunderstanding among the members unless everyone involved speaks the same language. Although some projects develop their own language, it is difficult to imagine what is going on if you are an outsider or new to the group. (2) There are some circumstances about a software system that cannot be comprehensible unless the built models are expressed in textual programming languages (see for example the class hierarchy). (3) Last but not least if the developers make program codes without designing models then the domain information will loss partially or fully forever because the implementation in most cases is not recreatable. There are things that are best modeled textually, others are best modeled graphically. The UML settles the abovementioned problems with its notations and the well-defined semantics. This means that somebody develops a model in UML and the others in present or a later project can understand and implement it unambiguously.

A language for specifying and constructing

By specifying models they became precise, unambiguous and complete. The UML addresses specification for describing the analysis, design and implementation model views, and for deploying the software intensive systems. As UML is a visual language it is suitable to map from a model to a source code written in some programming languages. The code generation from a model to a programming language is a forward engineering technology. The reverse is also possible; there are already software tools, which encode the programs to UML models. The process is called reverse engineering.

A language for documenting

In an effective software project the experts from all fields of information science and technology produces different artifacts such as requirements, project plan, prototypes, architecture, design, source code, releases, and some products that are critical only from the viewpoint of controlling, measuring and communicating. Depending on the development culture, these artifacts are treated more or less formally than others, but it cannot be denied that through the UML visualized documentation the users are able to better understand the running software and the developers are capable to make the necessary changes in the applications.

In addition, the UML is sufficiently expressive and unambiguous to permit the direct execution of models, the simulation of systems and the instrumentation of the running systems.

2.2. The New UML Standard

As the first standard version of the UML had not satisfy entirely the user's need it had to be taken under reviewing and reorganizing [1],[7]. The U2 Partners Consortium who is responsible for the revised version of UML consists of both tool vendors and power users dedicated to make UML a language that is easier to use, to implement and to customize. This means that several major changes both in semantics, notations and in extensibility mechanism was made related to the earlier versions [3] such as:

- The *architectural modeling concept* suits the MOF meta-meta-model standard[1], and it makes possible to interconnect the different model-views, namely the use cases, the behavior model (sequences, cooperation, activity and state transition), the components and the nodes.
- The activity graph semantic is separated from that of the state machine. There are new rules to express *alternative paths* and *concurrency* by the sequence diagram and to specify state machine generalization.
- The process of the component design is *interface-based*, that is to say it is possible to assign the required (input) and the proposed (output) interfaces to the components, but there are also new solutions to embed components.
- The *extensibility mechanism* is consistent with the 4-layer meta-model architecture.
- The *kernel* of the UML specification is the definition of the syntax and semantics [2], it includes related definitions for model interchange (UML CORBA facility, XMI DTD), language extensions (UML Standard Profiles) and constraint (OCL: Object Constraint Language).

The revision process is being done in four parts: the Infrastructure, the Superstructure, the OCL (Object Constraint Language) and the Diagram Interchange [1]**.** Several additional specifications help to tailor the UML to model the functionality and dynamism of the system more precisely: the new Action Semantics specification will enhance the language representation of behavior; a human-readable Textual Notation will enable a new class of UML editor programs and enhance the way, how the UML models can be manipulated. Notation elements will map the one-to-one to the more verbose XMI, but the syntax will differ. Analyzing the UML v 2.0, we do not forget to mention the standard Software Process Engineering Metamodel (SPEM) which defines a framework for describing methodologies in a standard way. It will not standardize any particular methodology, but will enhance interoperability from one methodology to another.

This new version of UML that was completed as a standard at the end of 2003 has several additional specifications, such as action-semantic specification, human readability by textual notations in the form of a class, and a standard development methodology framework as a metamodel. The UML Version 2.0 fulfilling the newly emerged requirements based on experiences from users and tool vendors is expected to be the basis for many tools, including those for visual modeling, simulation and development environments, and it is qualified for a base of many advanced techniques. From the Figure 1 we can see the connection between the different model views and the transformation process.

Summarizing all the aspects of today's software engineering process or better to say software technology we have to state that *it is necessary to change the developer's mind and style*. As the user's requirements and the information technology has been going through considerable changes, as serious value of information assets has accumulated in the last decades so the SDPs are required to focus on different IT targets that aim to save the existing IT and to give solutions for integrating the legacy and the new applications. The new standard has been unified in the MDA framework [5] which emphasizes the importance of integration and assure to meet all demands such as:

- technology-independent representation of the business on high level of abstraction and on an interface-based implementation,
- smooth and rapid integration of yesterday's and the tomorrow's architectures on intra- and inter-business boundaries (across deployment technologies),
- reduced time-period and costs throughout the application life-cycle taking advantages of reusability of models, codes, training and people,
- two level of modeling (PIM, PSM) and more views in each level,
- protecting the earlier investments and increasing return on new technology investments,
- scalability, robustness and security via generated code by stable model-based approach,
- improving application quality.

[1] The *Meta Object Facility* represents the integration of work currently underway in the areas of object repositories, object modeling tools, and meta-data management in distributed object environments. The MOF specification uses UML notation. The purpose of this key building block is to provide a set of CORBA interfaces that can be used to define and manipulate a set of interoperable metamodels.

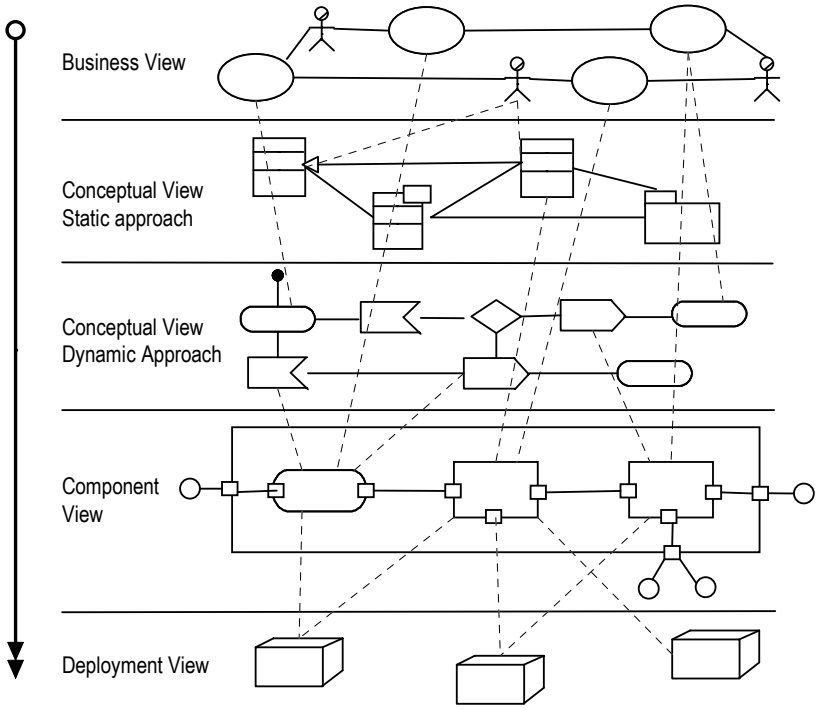

Figure 1. From business view to the deployment view

3. CONCLUSIONS

As one of the primary goals of convergent engineering is the business continuity, the managers need to go back to control both in terms of their bargaining power and management of product and service negotiations as well as in terms of their internal IT planning. By this concept the managers can see and understand what is happening, how resources are being utilized to produce business relevant results. The UML is the key enabling technology for the Model Driven Architecture, but it can be effective only if the project managers and the developers change their mind, culture and way of working, and if the applications are based on the normative, platform-independent, UML specified models. By emphasizing standard's universality, a unified framework allows the developers working in international teams to create applications that satisfy the long term user's need and that are portable and interoperable naturally across a broad spectrum of systems from embedded to desktop, to server, to mainframe and across the Internet.

References

[1] *Introducing to OMG's Unified Modeling Language* – http://www.omg.org/gettingstarted/ June 2002

[2] JACOBSON, I. – BOOCH, G. – RUMBAUGH, J.: *The Unified Software Development Process* – Addison-Wesley Longman Inc., 1999.

[3] KNAPMAN, J.: *Business-Oriented Constraint Language* – 3rd International Conference on the Unified Modeling Language, University of York, UK, October, 2000.

[4] KORBYN, C.: *A Standardization Odyssey* – Communications of the ACM, 1999. Vol. 42. No. 10.

[5] *MDA Specifications* – http://www.omg.org/mda/specs.htm June 2002.

[6] RAFFAI, M.: *Object Technology* – 1st volume*: Objects in Business Modeling – The Paradigm and the Methods of the Object-oriented Technology* 2nd volume: *Unified Software Development Solutions – UML Modeling Language, RUP Methodology* – Publisher Novadat, 2001.

[7] RAFFAI, M.: *The History on Computing* – Publisher Springer, 1997.

[8] RAFFAI, M.: *Managing Software Development Projects* – SENET Project Management Rev, Vol.3/No.1. 2003.

[9] RAIM, M.: *Implementation Infrastructure: enablers for rapid Enterprise Integration* – OMG Information Day, 2002.

[10] TAYLOR, DAVID A. – ZAMIR, SABA: *The Object-Oriented Paradigm - The Keys to Object Technology*– Handbook of Object Technology, editor in chief: ZAMIR, SABA, CRC Press, 1998.