UML: the language of blueprints for software?

Panel Session

Moderator: Derek Coleman
Panelists: John Artim, Viktor Ohnjec, Erick Rivas, Jim Rumbaugh, Rebecca Wirfs-Brock

Abstract

The Unified Method was launched by Grady Booth and Jim Rumbaugh at an OOPSLA'95 Conference Fringe meeting organised by Rational Software Corporation. In 1996 the Unified Method was re-scoped to a notation, and renamed the Unified Modeling Language (UML). Earlier this year, UML was submitted to the Object Management Group for standardisation and has been endorsed by Microsoft, IBM, HP, Platinum Technologies, ObjectTime and many other corporations. No wonder UML is the leading contender as the *de facto* standard notation for object-oriented analysis and design.

The panel will take a sanity check, and will go beyond the hype and newsgroup flames and attempt to form an objective view of UML and its prospects.

Introduction

The members of the panel have been working closely with UML in many different roles, including that of UML language designer, end-user, consultant; CASE [Computer-aided software engineering] tool expert, and object-oriented methodologist. The discussion will focus on how UML matches up in practice against one of its original raisons d'etre as "the language of blueprints for software".

Specific issues to be addressed include:

- l What is the advantage of UML over existing OOA/D [object oriented analysis/design] notations?
- Can UML be used on real projects today?
- Is the language sufficiently simple, and well-enough defined, to become the de facto standard?
- Will UML lead to improved OOA/D methods and CASE tools?
- What is the importance of the meta-model in UML?

Derek Coleman is currently Professor and Head of Department of Computer Science at King's College, University of London, UK. Prior to this Derek was a manager at HP Laboratories in Palo Alto and Bristol (England) where he led the development of the Fusion OOA/D method. He has extensive experience as a consultant transitioning projects to using object-oriented methods, both in Hewlett-Packard and in financial services and telecommunications companies. He has

Originally published in the OOPSLA 1997 Conference Proceedings

authored many research papers on software engineering and object technology and is currently co-authoring a book on a new version of Fusion using UML.

John Artim

I would like to evaluate the state of the Unified Modeling Language and its prospects through the following questions.

1) Does UML embody sufficient notational richness and meta- model flexibility to support a wide range of in-house and commercial development?" UML is the first method I know of to publish a meta-model in its own notation - in itself a significant sign of technical fitness. UML includes notation to express most of the requirements, analysis, or design information I have come across. Though diverse, UML does have its limitations.

With Responsibility Driven Design and Object Behavior Analysis, among others, preferred by a significant number of practitioners, larger projects will, based on methodological preference, continue to experience communication issues and outright schisms among colleagues. Until the strengths and semantics of these complementary notations and methods have been incorporated into UML, it will remain difficult to unite a diverse group of practitioners behind UML. It is especially important that UML's creators look further into the role of cognitive differences in choice of preferred notation and method. The human factors issues behind these preferences must be better understood.

2) Does UML include or accommodate organizing constructs such as framework and design patterns?" Though Booch [1] discusses the concept of framework and how it relates to class, I was unable to find a formalized definition of a framework within UML's meta-model. From a standpoint of engineering pragmatics, I am troubled that UML appears not to aid in coordinating the representation of frameworks. This reflects on UML's completeness but also on the viability, or at least the timeliness, of OMG coming forward as a means of integrating vertical and horizontal frameworks. This deficit may, however, be offset by the advantage of a published meta-model. I would hope to see future discussions of 00 concepts expressed in terms of this meta-model.

3) "Is there an on-going dialog between UML's creators and the broad ranks of day-to-day practitioners?" In publishing a detailed meta-model and description, UML. has significantly raised the bar for technical communication, of OOAD issues. Unfortunately, the dialog seems to have become one-sided. A quick scan of OOPSLA workshop offerings over the past couple of years shows an absence of UML-specific discussion. OOPSLA workshops I have attended often seem to end up discussing the use and relative merits of various methods, including UML. I do not know of any systematic way this information is getting back to UML's organizers. Nor does there seem to be, at least at this point, much effort to solicit detailed feedback through other than the OMG review process. Participation in this approval process is resource intensive and therefore tends to exclude many practitioners. The same discussion mechanisms that laid the groundwork for the predecessor methods of UML seem to no longer function to ensure its continuing evolution.

4) Can UML embrace co-evolving standards of practice in disciplines allied with object-oriented technology such as user interface or documentation design?' The only place in the UML submissions I could find a modeled notion of software engineering process and the flow of artifacts was in the business process modeling extensions needed by Objectory. Expanding these extensions into a development process meta-model, would, in principle, enable UML to integrate development artifacts across the software engineering lifecycle. There was a discussion of this topic at a CHI 97 workshop on the use of object-oriented methods in user interface design [5].

The workshop's participants are relying on one participant's company status as an OMG reviewer to relay feedback on UML's meta-model design. A mechanism for submitting technical review commentary by non-OMG participants would guarantee a more reliable means of providing feedback but at the cost of an increase in volume of review comments. Note that this is not an issue of approval but rather it is an issue of communication regarding UML and its place in a larger and evolving technical community.

5) "Is there a comprehensive (with respect to UML), complete (with respect to an example domain and system) and publicly accessible sample model(s) documenting intended usage of UML notations and techniques?" I do not think such a model is within the scope of an OMG proposal. I would turn, instead, to the published works of UML's three creators ([1], [2], and [4]). The model snippets used throughout each of their initial texts fall short of satisfying these criteria. Rumbaugh's *OMT Insights* [3] clears up much ambiguity with its comprehensive set of examples but these are examples in isolation. The utility of a more elaborate reference model would be in illustrating some of the many engineering decisions that must be made when modeling a real-world system. Probably the two most important kinds of decisions to illustrate are appropriate levels-of-detail within a model and the trade-off between problem complexity, model complexity and reader comprehension. This is especially needed given that UML is a standard meta-model and notation but not a standard methodology. Illustrations of each of the three principal architect's methodologies would help clarify the range of intended use of UML.

What do I believe the state of UML to be? I recommended UML as the basis for our development center's standard process and I see no reason to change that recommendation. The UML OMG proposal has considerably clarified UML's content and notation. Serious gaps remain in the areas of development process meta-modeling and framework description support.

These missing pieces are not immediately necessary but a viable standard in this area must demonstrate that it can grow to fill these gaps. Finally, the on-going evolution of UML seems much less open than much of the discussion that led up to its creation. Ultimately, this may increase UML's risk of premature obsolescence.

For the past 3 years John Artim has worked at Orient Overseas Container Lines, a global container shipping company, in the Information Services Development Center in San Jose,

California. He is the user interface architect for an enterprise-wide system of applications supporting customer service and the shipment life cycle. In addition to working on the current phase of application delivery, John is interested in putting into shop practice a user interface style guide based on user task descriptions corresponding to the content of an application's use cases. Prior to 1995 John spent 6 years working at IBM, primarily at the Santa Teresa Laboratory in San Jose, California. He worked in development, user interface architecture and human factors. The principal products he worked on included various CASE and development tools supporting structured and object-oriented programming. He holds an M.A. in Experimental Psychology and a B.A, in Psychobiology from the University of California at Santa Cruz.

Viktor Ohnjec

Genesis will specifically offer comments on the following issues:

- Can UML be used on real projects today?
- How well suited is UML to addressing the issues of distributed object systems and the Internet?
- Is the language sufficiently simple, and well enough defined, to become the de facto standard?
- Will UML lead to improved OOAD methods and CASE tools?

As an organization, Genesis refers to itself as "object transition specialists". As such, we target the transfer of technology and knowledge for our clients in a variety of object technology related areas. These areas include analysis and design methodology usage. We see the OMG OOAD standardization effort as a very positive move for the industry that develops both the methodologies and tools to support software development and see the evolution of UML as a very positive example of what submissions into the standardization effort should be. There are issues that we must contend with and that are not, in our opinion, sufficiently covered with the current version of UML. We hope to discuss those issues during the panel discussion.

Q: Can UML be used on real projects today?
A: Genesis believes the answer is yes, because we have to date already used UML in a variety of client specific situations. UML is very reasonable as a notation and although there are some inconsistencies in the semantics and notation, they are minor in comparison to the benefit that using a modeling language brings to organizations. We look at methodologies and at UML as a mechanism to assist communication of ideas between team members on a project. As such, we look at whether UML can help team members to describe the requirements, analysis, and design ideas that they feel are important. We have seen that UML can do this and specifically have focused on Use Cases, class diagrams, state diagrams and interaction diagrams (collaboration and sequence diagrams to be specific) as the most important elements of our modeling efforts.

Where we feel that UML is lacking is in the area of process. In many ways, the UML is a fine notation, with reasonable syntax and semantics; clever use of stereotypes to reduce complexity, and as a general rule, the language is well received by both the vendors, the end-users and the standardization effort. What it is not, and admittedly, it does not claim to be, is process rich. We

have found that clients are eagerly awaiting some guidance from the “three amigos” on a suggested approach for how to use UML. We find that although many approaches are reasonable, having a single approach that is suggested would make the convergence effort even more rapid.

We at Genesis have also noted that methodologies like Team Fusion or some of the areas of OPEN may in fact add the most value add to end-users through the suggestion of process. Team Fusion is already incorporating UML as the notation, and essentially offering the updated Fusion process around UML. As a company that assists end-users through pragmatic use of OOAD, we consider such efforts very important.

Q: How well suited is UML to addressing the issues of distributed object systems and the Internet?
A: Few methodologies can truly be used throughout the entire lifecycle in creating solutions based on distributed object technology-UML is no different. The challenge faced by all methodologies is to break out of the notion of application development to move more towards infrastructure and component development. These new directions are not trivial, however, and must be undertaken with extreme care.

First, a standard mechanism must be identified and adopted for the definitions necessary in creating any solution with object technology. UML is already well on the way to providing this as we have already discussed here. But we as an industry need to continue to encourage that it is in fact sufficient as the *de facto* standard today. Again, having additional process-related information available is about the only real area that should be improved upon. Note however that we do expect further refinements to be shown (most likely through stereotypes of some form) especially in the way that UML will support business process engineering activities and distributed object computing activities. At present, we see the current version of UML being reasonable in its support of these areas (so long as the approach that an organization chooses to take in supporting BPE [business process engineering] or DOC [distributed object computing] remains self-consistent within all groups that are working together in the organization).

Q: Will UML lead to improved OOAD methods and CASE tools?
A: Regarding improved OOA/D methods, Genesis sees convergence on a single method (or smaller set of methods) as positive and UML is already helping to reduce the number of methods available. So, yes, we think this will help, but we believe that the main help is to reduce the “religious wars” rather than suddenly make developers so much more intelligent in their use of OOAD methods. It will, however, allow the infrastructure, object architecture, object modeling, issues to deal with to be raised from “how can I learn the method?” to “how do I use the method effectively?” more quickly. This is good.

As a consulting organization that is unbiased by what specific tool or vendor an organization chooses, we consider the question “will UML lead to improved CASE tools” from a service perspective. We expect that the CASE tool organization that will become most dominant, will be the one that has good services available to support not only their tool (through internal

resources), but most importantly, the effective use of the method and the tool to bring projects to fruition. As such, we look for the organization that will partner with other groups that have pragmatic experience, and that can offer experience through mentoring to be the vendor that will see the most dominance. This is especially true since the area of process, that we feel so strongly about, is one that the service side of the CASE tool usage must support.

We also see the need for greater levels of integration between the "so called" CASE tools and products that support the remaining aspects of object or component based development, namely requirements capture tools, documentation tools, configuration management and version control tools and testing tools. We feel that end-users will become more and more sophisticated in their development and component assembly approaches and thus UML will need to ensure that it and the tools that support its use, will be easily integrated through-out the software lifecycle.

Incidentally, we wonder out loud if any of the CASE tool vendors plan to formalize mappings from UML to areas beyond. In such cases, and although work has definitely been published in these dress already, we expect to see things like formal Use Case to Test Case mapping, formal simulation of distributed components and debuggers, and automatic component "producer and consumer" models where it is expected that business objects will be assembled from base objects and that "infrastructural objects" will in essence be present to support the "foundation" that all business objects will require for an application to run. In such a model, we see an evolution in how people would present models that they built using UML rather than a necessary evolution in UML.

In closing, Genesis believes that standardization is a positive activity and since UML has shown the ability to gather momentum and support, it has in essence helped to accelerate the standardization effort. Is it the absolute best that it can be today? Probably not, but that isn't as important as the enthusiasm that it has generated and therefore the interest that people once again have in performing proper analysis and design prior to attempting to code away at a project!

Viktor Ohnjec is Vice President, Professional Services, at Genesis Development Corporation. Viktor is responsible for ensuring customer satisfaction in training and consulting engagements through the management of technical resources and courseware development for Genesis. With 12 years of Object Technology experience, Viktor has been directly involved in planning, mentoring and leading teams through large systems development in military, aerospace, commercial, manufacturing, financial and petrochemical areas.

Viktor has introduced Object Technology to executives, middle managers and developers and influenced organization, infrastructure, object architecture, object modeling, development and integration in both distributed and non- distributed computing projects worldwide.

Viktor has presented at numerous conferences on a variety of technology and management topics. As a participating member of OMG, he chairs the Test Special Interest Group and is a member of the Architecture Group and Analysis and Design Task Force.

As an author, Viktor has written articles on emerging technology trends including the Application Development Trends February, 1997 cover story "Converging on OOAD Agreement" and June, 1997 titled "The Brave New World of Distributed Object Computing". A book by Viktor is expected by late-1997.

Erick Rivas

Erick Rivas works for Platinum Technologies.

Jim Rumbaugh

UML is intended to consolidate the experience of the OO modeling community by providing a set of semantic modeling concepts and a corresponding notation that can provide a standard general-purpose modeling language for expressing most kinds of models. Its main advantage over other OOAD notations is the opportunity to end the petty wars over minor differences in semantics and notation by adopting a broad consensus developed by a number of methodologists and vendors. UML contains some newer features, but its core is based on years of experience with several of the leading OO methods. Yes, UML can be used on real projects today.

UML is not a tiny language, but neither are C++, Smalltalk, Java, or Eiffel. It needs to accommodate analysis and design, large and small projects, be compatible with many programming languages but dependent on none. In particular, it needs to embrace the systems of today (they are no longer in the future) that are inherently concurrent, distributed, and multilingual. We have made UML as simple as possible subject to these needs, but we don't expect someone to learn it in a day. We have structured the core concepts of UML to be straightforward; however; users of most popular methods SHOULD learn enough in a day to continue working productively in UML.

We feel that UML is better defined than any other comparable modeling language. It has a self-referential meta-model (any general purpose modeling language should be able to model itself) as well as a built-in constraint language for defining non-syntactic restrictions. We have tried to strike a balance between formality and pragmatism. People will find flaws, of course; nothing of this size is ever perfect, but that is no different from most software products. We feel that the language is robust and can be repaired or extended easily, if the need arises.

By providing a standard, comprehensive definition of a modeling language covering all major areas of concern we feel that UML can lead to a great increase in quality and quantity of modeling tools. Vendors can develop more effective tools if they can count on a widely accepted standard, rather than having to choose among dozens of potential candidates, and users benefit

from being able to choose the best implementation rather than worrying about which notation is supported.

Jim Rumbaugh works for Rational Software Corporation in Santa Clara, California. He received his Ph.D. from MIT in 1975. He has been involved in software modeling for almost 30 years, as well as software development including an operating system, a parallel machine architecture, a compiler, a transaction management system, an X-ray tomography system, an object-oriented programming language, and an OOAD CASE tool. He is the lead developer of the OMT method and the book "Object-Oriented Modeling and Design" with colleagues from GE and Calma. Since 1994 he has been working with Grady Booth and later Ivar Jacobson at Rational Software Corp. to unify their methods into a single approach. The UML is the first fruit of that collaboration, but they continue to work on process unification also.

Rebecca Wirfs-Brock

I have spent a dozen years exploring informal techniques and ways of thinking about object system development. I see notation as an aid to conceptualizing, specifying and communicating various aspects of object models. Unified Modeling Language is becoming the lingua-franca for object modeling in the 1990s. One of the more subtle, but important characteristics of UML is its formal underpinning - there are semantics behind each modeling construct. UML was also designed to be formally extended. These good intentions remain to be proven.

I am still getting used to speaking this object modeling Esperanto. It takes a bit of effort for me to distinguish between an instance and a class (underlined names are a pretty subtle distinction). The notion of drawing objects as squares also seems alien. I have always equated circles with objects! I am getting over my ingrained notion of "circle" as "object", because I too wish to communicate and be understood by others, without translation. This is important. In the process of adopting a modeling language standard, we give up colloquialisms to gain common understanding.

UML isn't pretty or elegant; however, I think it is utilitarian. It won't meet everyone's aesthetics, but in the end this does not matter. I intend to make serious attempts to be UML compliant when it expresses the constructs I need. But when I must color outside the lines to communicate an idea, I will do so. I still conceptualize object responsibilities on CRC cards, then transform them to individual operations, attributes and associations to fit into UML. I augment use case models with quite a few embellishments and believe that UML can be woven into a process that incorporates a rich set of informal techniques and modeling constructs.

However, I want to caution you - don't become complacent with these new standards! The notation war may be over, but UML falls far short of expressing all we need to understand about object-oriented software systems! UML does not address many intrinsically, non object-oriented details that are important to describe. And UML doesn't speak to extended use case models nor does it particularly well describe dynamic relationships between objects, patterns, algorithmic hot-spots, event models, object subassemblies or subsystems, Bertrand Meyer's notion of

contracts, invariants, business rules, or many of the artifacts of our own Responsibility-Driven Design process.

I have difficulty distinguishing between levels of detail, precision and abstraction when looking at models expressed in UML. The basic constructs of object, interface, class, sequence diagram, etc. are used to express analysis, design and implementation models. Some may argue that this is a virtue, it's objects from the top to the bottom. But you really haven't communicated with me if I can't discriminate between various levels of abstraction. I leave it to object development processes and forward-thinking modelers to provide us guidance in this area.

Is the state of modeling with UML so bad? On my worst days, I fear that people will stop thinking about their objects once they have sketched a few UML diagrams, and that future object modeling innovations will be ignored by those biased towards a rigid standard. However, I ask you to put UML into its proper perspective. UML shouldn't be burdened with having to be all things to all people for all times. I hope UML continues to remain a living language, adapted by people to create new modeling idioms appropriate to the task at hand.

Rebecca has been active in the object community for over a dozen years. Her object experience began in 1984, when she managed the Tektronix software team that developed the first commercial Smalltalk. Rebecca was the lead author of the popular Designing Object-Oriented Software, Prentice-Hall, 1990 and has written columns for the Smalltalk Report and the Report on Object Analysis and Design. Rebecca left Tektronix in 1991 to pursue her object analysis and design passion full-time and to direct the consulting and training practice at Instantiations. Through two high tech mergers, she came to be the Director of Frameworks and Methods at ParcPlace-Digitalk. She is an internationally recognized author, teacher and speaker on object analysis and design and co-inventor of the Responsibility-Driven development method.

References

1. Booch, G. (1994). *Object-Oriented Analysis and Design.* Redwood City, California: The Benjamin/Cummings Publishing Company, Inc.
2. Jacobson, I., M. Christerson, P. Jonsson, and G. Overgard (1995). *Object-Oriented Software Engineering: A Use Case Driven Approach.* Menlo Park, California: Addison-Wesley Publishing Company.
3. Rumbaugh, J. (1996). *OMT Insights.* New York, New York: SIGS Books.
4. Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorenson (1991). *Object-Oriented Modeling and Design.* Englewood Cliffs, New Jersey: Prentice Hall.
5. van Harmelen, M., J. Artim, K. Butler, A. Henderson, D. Roberts, M. B. Rosson, J. Tarby, and S. Wilson (1997). Object Models in User Interface Design: CHI '97 Workshop Summary. In preparation for SIGCHI Bulletin, October 1997.