What Practitioners Are Saying About the Unified Modeling Language

Brian Dobing, University of Lethbridge, 4401 University Drive W., Lethbridge, AB, T1K 3M4, Canada; E-mail: brian.dobing@uleth.ca
Jeffrey Parsons, Memorial University of Newfoundland, St. John's, NF, A1B 3X5, Canada; E-mail: jeffreyp@mun.ca

INTRODUCTION

The Unified Modeling Language (UML) was formally introduced in the late 1990s, with much of it based on earlier object-oriented systems analysis and design (OOAD) methods. The UML quickly became the language of choice for modeling object-oriented systems and there are now numerous books, many written by practitioners, which describe the UML and suggest approaches for using it. The UML has continued to evolve, notably with the release of UML 2.0, driven largely by a "best practices" approach. This paper offers qualitative evidence of the broad scope of practitioner views on the UML in the form of comments from by UML practitioners in response to a survey on UML usage patterns. The range of comments suggests some interesting directions for future research.

RESEARCH METHODOLOGY

A web survey containing 38 questions, many with multiple parts (e.g., a list of possible reasons for not using a particular UML diagram), was developed and posted on the web in March 2003. The Object Management Group (OMG) supported the project by sending an email to their members to inform them of the survey, and by posting a link on their main web page. The survey was "intended for current systems analysts/architects who have used or considered using the UML in systems development projects" and some initial items asked about that experience. A few respondents did not belong to our target population, but no reported comments or data come from them. There were no rewards offered for participating, except for a copy of the results, so there was no incentive for non-UML practitioners to complete the survey with fictitious responses.

Most of the questions used standard Likert scale items and check boxes. Some of this quantitative data was presented and analyzed in Dobing and Parsons (2006). Respondents were also given substantial opportunities to add comments and further explanations. This paper focuses on these written responses, selected from 52 surveys. While these comments may not always reflect the majority views of the UML practitioner community, they are useful in identifying issues and concerns among a group of people who are largely committed to the UML and to its continued improvement.

The quotes provided in this paper are generally as submitted with any minor modifications (generally to preserve grammatical correctness) inserted in square brackets. Typographical errors have been corrected without any indication. Any use of upper case for emphasis has been left as in the original.

RESULTS

This section addresses some important UML issues which received higher levels of written responses. As might be expected with a relatively new language, it is being used in quite different ways by different people. As one respondent put it:

Used, vs. used appropriately, is probably a telling difference. Many places are using the components but in a relatively brain-dead manner. I don't think this is a UML issue. It is a brain-dead issue.

Of course, one developer's "best practice" can be another's "brain-dead manner." Our goal is not to offer solutions but simply to identify some of the issues about which UML practitioners feel strongly and are trying to address. These, in turn, should be useful in guiding further research in this area.

Use Cases

The UML is a language and not a development methodology, but all the major books on the subject adopt the "Use Case-driven" approach espoused by Jacobson (1992). There are some critics (e.g., Anderson, 1999; Simons, 1999), but most warn about incorrect application (e.g., using too many, going into too much detail, etc.) and do not reject Use Cases completely. However, our survey found that only 63% of respondents employed Use Cases in at least one-third of their projects. One respondent was "curious [about our] bias … towards Use Case narratives" saying "This is not a mandatory way to elicit initial requirements." Others described Use Cases as "close to useless," "just unformatted text notes," and "too imprecise to invest much effort in." "The ambiguity of Use Cases in particular is problematic." Another argued for a more limited role, saying "They are critical in defining the boundary behavior of the system but only that." Thus, there is a sharp discrepancy between practice and conventional wisdom in the literature.

One respondent questioned whether the Use Case Narrative is a model, asking:

Use cases allow technologists to build something that resembles a business solution. They are useless in modeling the actual business processes. ... Yes, you can put some information into a narrative, but is that a model?

Perhaps the key underlying issue concerns the role of Use Cases in requirements gathering, which is their intended purpose, and in design where the other UML diagrams are more relevant. One respondent stated:

Use Cases MUST be combined with functional requirements and constraints. Use Cases only show the behavior of the system.

Behavior is, of course, exactly what a Use Case Narrative is designed to show, 'what' not 'how' (Jacobson et al., 1994, p. 146). Whether the 'how' includes the user interface remains a concern. Constantine and Lockwood (1999, p.102) argue in favor of 'essential' Use Cases, which exclude interface details, claiming that "conventional use cases typically contain too many built-in assumptions … about the form of the user interface." But respondents who commented on this issue generally took the opposite view. One stated that "end-users have difficulties [when] validating a Use Case Narrative without any draft of the UI." Another goes further, saying, "It is easier for clients to understand the functionality of software through user interface sketches." While it is important to "promote better Use Case writing styles," there was no consensus among our respondents on what the style should be.

Use Cases were acknowledged to be useful in "defining test cases" and "effort estimation." The associated Use Case Diagrams, which generally were viewed less favorably, provide an "overall view" and help with "scheduling [and] risk assessment." While consistent with the literature, these are not generally considered the key reasons justifying Use Case development.

Other respondents identified alternatives. One "eliminated Use Case Narratives in favor of Activity Diagrams" while another said that "a state model hierarchy is sufficient." 'State model hierarchy' is not a UML term, or even one commonly associated with system development, but is used in connection with knowledge bases for expert systems. Some indicated they use "text" instead but without specifying how their documentation differs from Use Cases.

Another issue with Use Cases is their organization and maintenance. This is particularly true for those who favor more of them with more detail. One respon-

dent who did not use them stated, "My guess is it would be too hard to store and recover the Use Case narratives to use them in later [maintenance]." Of the 39 respondents who reported the number of Use Cases on "typical" projects, the top 15 ranged from 100 to 800. We did not ask about average length. One proponent said, "our typical Use Cases consume one - two pages each." but an infrequent user said that "Use Cases tend to become remarkably complex and highly error prone - have seen Use Cases as large as 250 pages." One reason for this variation may be the lack of any official UML specification. The OMG (2005. p.574) simply states that "Use Cases are typically specified in various idiosyncratic formats such as natural language, tables, trees, etc." There are now several books devoted to Use Cases (Armour and Miller, 2001; Cockburn, 2001; Adolph and Bramble, 2003; Bittner and Spence, 2003; Denny, 2005; Övergaard and Palmkvist, 2005) along with a few web sites, notably Cockburn's (http://www.usecases.org). But that isn't sufficient for one respondent who argues, "too much written but still too little [clarity on] what goes where [and] tools - little benefit for the time and effort invested in Use Cases."

There is insufficient understanding of why so many practitioners are avoiding Use Cases and what they are using as alternatives. Also, if a project has no Use Cases it certainly isn't Use Case driven. What, if anything, is driving their projects?

Activity Diagrams

We would expect differences of opinions on all UML diagrams but, after Use Cases, they were expressed most sharply for the Activity Diagram. One proponent stated, "We are currently using Activity Diagrams for the detailed specification of all Use Cases" while another said they were "useful to understand business activities [and] useful to understand the flow of activities in a Use Case." Another called for client involvement, "These being the only container we have that can hold business process information, it is critical that business people are engaged as much as possible." Their role was explained more fully:

Activity diagrams are very important when the client begins to describe process-like behavior and logic. Good to help understand and convey a business process, or an underlying algorithm. In these cases, I think an Activity Diagram is essential – compared to pure text.

However, others took strongly critical positions. One simply said, "'Activity' actually - sorry – sucks." According to different respondents, "they are very time-consuming to produce," have "unclear semantics and an unclear connection to the rest of the UML Diagrams," do not represent "the concept of a 'business process,'" are redundant "if Use Cases are well written and well modeled," lack a "wealth of information" about them in the literature and have "very poor tool support and integration (via tools) with the rest of the notation." Some suggested modifying them, "[without] the Erikkson Penker extensions … UML Activity Diagrams would be pretty useless for business modeling" and "we used Activity Diagrams, but with our own semantics." Others said "they need to be more like DFDs if they're to be useful" and "neither the concept of a 'business process' is well represented, nor is the value of good old DFD diagrams available." The latter respondent also claimed to "have developed an interesting and useful notation for processes/tasks/workflow." Alternatively, perhaps they are not sufficiently connected to the Class Diagram:

UML does not handle business process modeling. Activity Diagrams are the nearest, but: (a) I want to show classes, not instances. (b) I want to show resources (as classes!) input and output…

Thus, practitioners seem to have very different expectations for how Activity Diagrams should be used, and thus on how they should be improved.

Project Communication

Communication within the project team has long been considered critical for successful outcomes. Questions on this topic generated a large number of responses. As noted earlier, the UML literature generally stresses the importance of Use Cases for client-analyst communication and the remaining diagrams for analyst-programmer communication. (However, it should be noted that not all projects have distinct clients, analysts and programmers.) In our survey, respondents reported that clients were more involved with the more technical UML diagrams than the literature would suggest (Dobing and Parsons 2006). However, this may reflect the types of clients involved in these projects. In some cases, the clients were likely engineers. And in any organization, when introducing new technology the initial clients are often chosen based on their enthusiasm for trying new approaches.

Some respondents supported the view in the UML literature. For example, one respondent said, "[The Class Diagram] is too far from the business customers and users to be effective [or] useful," while another described it as "the most important diagram for technical members." The separation of Use Cases and Class Diagrams was also mentioned:

Class diagrams are nearly all at a design level, so have little to do with requirements (except that they result from requirements).Class diagrams are essential to deriving/documenting a long-lived, robust, software architecture.

These problems may be exacerbated when analysts take the view of the respondent who said "the only benefit of Use Case diagrams is to please managers."

Simply put, "The only problem is communicating with people not familiar with the UML." But how should that familiarity be achieved? One view is that clients, in general, will not be interested in or able to understand the UML. Those holding this view argue that the UML is "too geeky and techie for non-technical people" and "[our clients] eyes glaze over when we try to present UML/UP artifacts." If analysts proceed anyway, "explaining the notation sometimes dominates the discussions." As a result, there is "little involvement of key business people." These respondents conclude: "give up on this goal" and "don't try – UML is technical, clients are not."

Others see training as the solution. They emphasized the "need to educate clients … in UML," and called for "more awareness programs conducted in the industry." "The most important one is to get the client understanding OO and UML formalism, once this is achieved, you're in heaven." In some earlier informal discussions with early adopters of the UML, we found that sending both developers and clients to joint training sessions was common. Perhaps more efforts of this type are needed. Another said, "Other parties usually don't speak UML. You guys should do something about it." It's not clear if this refers to the OMG or academics, but as members of the latter group we need to think about whether students outside Information Systems majors should be exposed to the UML.

The complexity of the UML is a concern when training anyone in its use, and particularly clients. Simplification is one suggestion:

I think the only way is to teach clients to understand UML diagrams at the necessary conceptual level. This can be achieved by publishing a set of simple articles that use the minimal UML notation and describe [to] customers how it should be read. ... Maybe there is a need to define [a] basic subset used for Analysis. Maybe there should be subspecs describing these basic aspects.

More specifically, "Stereotypes and fancy arrowheads on class and sequence diagrams are details that are lost on the client."

Training is also needed within the technical staff, along with changes in attitude. There were references to "overnight experts [who] refuse to listen to anyone who's done the work before," "people not being able to model … modeling is HARD" and "misunderstanding of the object oriented concept (for analyst and programmers)." There were concerns about both "Luddite programmers," who are presumably experienced but not in object-oriented methods or the UML in particular, and "new staff and novice software developers."

Some of the solutions suggested include consistent terminology across products ("Terminology is the key to success … Visio calls this that and IBM calls this that and J2EE calls this that …"),, "introduce a Use Case Flow Diagram" (to show possible flows), "low-fidelity pieces – sort of a 'UML for Dummies'," agile modeling (and some related practices), and executable UML (which received quite strong support).

As system projects, development teams, and the number of stakeholders affected become continually larger, communication becomes even more important. A common language, such as the UML, can be very helpful. However, the wide variation in how the UML is used, including how it is augmented and handled by different tools, limits that commonality.

Augmenting the UML with Additional Modeling Approaches

We asked respondents which other modeling constructs they have used to augment the UML. The suggestions were numerous and varied. Entity relationship diagrams were the most popular. One reason was that "data is persisted in an RDBMS," e.g., "an Oracle database." E-R Diagrams were not being used instead of the Class Diagram. Of the 17 respondents who reported using E-R Diagrams specifically or just data modeling in general, 15 said they used Class Diagrams on every project. Other older approaches used include Data Flow or Process Flow Diagrams and, more generally, Structured System Analysis and Design. Respondents did not address whether they were using older approaches to phase in conversion to UML or if they planned to retain them in the longer term, although one said that flowcharts were an "appeasement for crusty [business analysts]" and another noted that "[customers] have a tradition of descriptions in a certain format and are not used to a UML view of the system." Some respondents also mentioned using alternative object-oriented approaches, including OMT (Booch), Shlaer-Mellor, Class Responsibility Collaborator (CRC) cards, and Coad/Yourdon.

There were six respondents who used supplementary approaches for user interfaces. As one respondent noted, "UML does not really cover this as such." Respondents were later asked about possible difficulties that had occurred that "could be attributed to the UML." Of the five categories listed, user interface was checked most frequently (36%). Thus, this would appear to be area for further development within the UML. One respondent suggested a possible product:

Lucid [from elegance technologies (www.elegancetech.com)] is a framework for user interface development. It is used very frequently in my company because there are no tools in UML for user interface modeling.

There are other products in this domain, although none were recommended in this survey.

Respondents also used additional tools for enterprise architecture, including Popkin Software and ICONIX. Others did not specify the approach used, e.g., "P2P, 2-tier, n-tier, centralized and distributed architectures."

There is a long list of other products being used to augment the UML, including IDEF (Knowledge Based Systems), Business Process Modeling (OMG), TurboCase (Hatley-Pirbhai method real-time systems) code prototypes ("see it, believe it!") and "ad hoc bubble diagrams." Six respondents reported using their own approaches and another five used various extensions (e.g., "a proprietary spatio/temporal extension" and "extensions for web applications") to the UML.

In answering this question, a number of criticisms of the UML were also put forward. They said the UML is "just a language, so [it] is just adequate for 'drawing' thoughts" and "the UML just a notation." The latter is correct, but many of the alternatives suggested are essentially notations as well. Another said:

[UML] methods ... are too constraining for the average customer. They want to do UML but balk at all the discipline and structure that is required to make it work.

In summary, the UML is often being used with other approaches. Some pre-date the UML and are used for continuity; others supplement the UML to provide additional capabilities. The extent and variety of these practices is a challenge to the claim of a "Unified" language.

Tools

Considerable dissatisfaction was expressed with both the quality ("nobody really implements the standard") and cost of UML tools available at the time this survey was completed. Rational products were used most frequently, with TogetherSoft a distant second. There were over 40 products mentioned in total, but this includes Microsoft Office products (e.g., using Word for Use Cases) and other system development tools not specifically designed for the UML market. Visio came third, but again cannot be considered a full UML tool although it does support the notation. The highly fragmented nature of this market is clearly an issue in the development of good UML tools, and no doubt contributes to problems getting different tools to work together. One, perhaps too cynical, observation was that developers see providing greater interoperability as a competitive disadvantage:

this is not a priority for CASE tool developers as this would allow analysts to change the tools they use in a too easy way.

One lengthy response sums up the general feeling:

One point that might be of more use is asking why there are no (zero) tools that completely support the full UML spec? As well as why tools are of such minimal use on large projects. Who is directing these tool development projects? I guess they start with Use Case diagrams of the problem domain. That would contribute to the use - but I think the problem is elsewhere. You should find a small group of enterprise architects who have also developed large real-world projects and fund them to develop usable tools. The reason I participated in this survey is in the hope we might someday see a tool that provides 15% of what is needed.

Software development tools have generally not received much attention from academic researchers. However, they can have considerable influence over how their underlying methodology is used.

CONCLUSIONS

A survey of UML use revealed considerable differences in the level of use and perceived value of UML diagrams (Dobing and Parsons 2006). This paper provides clues about possible reasons for these differences and points to opportunities for further research.

One concern is that the informality and lack of standards for writing Use Case Narratives limit their usefulness, both for documenting business processes and for supporting the development of other UML artifacts, especially in view of the amount of effort needed to create them. Given that UML advocates often prescribe a "Use Case driven" approach, research is needed to better understand why practitioners disagree on the value of Use Cases, as well as to develop and evaluate potential guidelines for using them consistently and effectively.

A second issue arising from the comments is the lack of agreement on the role and value of Activity Diagrams for describing business processes, as well as their relationship to Use Case Narratives and other UML diagrams. Some respondents found them to be very useful, while others held the opposite view. This suggests the need for research to better understand the role Activity Diagrams should play in UML models.

Another concern raised in the comments from respondents is the (un)suitability of UML diagrams for communication with clients. Despite quantitative evidence that clients are more involved in the development, review, and approval of UML artifacts than the existing practitioner literature suggests (Dobing and Parsons 2006), comments suggest that these diagrams are too technical for clients to understand and use. Clearly, there is a need for research to examine the use and usefulness of UML diagrams in communicating with clients.

Finally, despite the scope and complexity of the UML, some organizations are augmenting it with other modeling approaches. Further research is needed to understand whether this is done to accommodate transitions from older methods to the UML, or because of perceived limitations of the UML.

In conclusion, despite the standardization UML has brought to object-oriented analysis and design, there is a wide range of opinions about how to use the language and how useful the various constituent diagrams are. This offers a significant opportunity for researchers to contribute to a better understanding of effective UML adoption.

REFERENCES

Adolph, S., & Bramble, P. (2003). *Patterns for Effective Use Cases.* Boston: Addison-Wesley.

Anderson, D. (1999). Use Cases Still Considered Dangerous. *UIDesign.Net*, Oct. 1999. Available: http://www.uidesign.net/1999/imho/oct_imho.html

Armour, F., & Miller, G. (2001). *Advanced Use Case Modeling.* Boston: Addison-Wesley.

Bittner, K., & Spence, I. (2003). *Use Case Modeling.* Boston: Addison-Wesley.

Cockburn, A. (2001). *Writing Effective Use Cases*. Boston: Addison-Wesley.

Constantine, L. & Lockwood, L. (1999) *Software for Use.* Reading, MA: ACM Press.

Denny, R. (2005). *Succeeding with Use Cases: Working Smart to Deliver Quality*. Upper Saddle River, NJ: Addison-Wesley.

Dobing, B. And Parsons, J. (2000). Understanding the Role of Use Cases in UML: A Review and Research Agenda. *Journal of Database Management 11*, 4, 28-36.

Dobing. B and Parsons, J. (2006). How the UML is Used." *Communications of the ACM* 49, 5, 109-113.

Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G. (1992) *Object-Oriented Software Engineering: A Use Case Driven Approach.* Addison-Wesley.

Jacobson, I., Ericsson, M., And Jacobson, A. (1994) *The Object Advantage: Business Process Reengineering with Object Technology.* Addison-Wesley.

OMG. (2005). Unified Modeling Language: Superstructure, Version 2.0, formal/05-07-04. Available: http://www.omg.org/technology/documents/formal/uml.htm.

Övergaard, G., & Palmkvist, K. (2005). *Use Cases: Patterns and Blueprints.* Indianapolis, IN: Addison-Wesley.

Simons, A. (1999). Use cases considered harmful. In *Proceedings of the 29th Conference on Technology for Object-Oriented Programming Languages and Systems (TOOLS)*, 194-203.