Using the UML to Describe Design Patterns

Diane Strode
School of Computing
Whitireia Community Polytechnic
Porirua, NZ
d.strode@whitireia.ac.nz

Proceedings of the 16th Annual NACCQ, Palmerston North New Zealand July, 2003 (eds) Mann, S. and Williamson, A. www.naccq.ac.nz

ABSTRACT

This report investigates the Unified Modeling Language (UML), its status as an international standard in computing practice and its strengths and weaknesses. How the language is used to describe patterns and frameworks and the weaknesses of UML in this regard are explained. The temporal relationship of object-oriented languages, UML, and design patterns is described.

Keywords

UML, design patterns, OMG meta-model, UML weaknesses, application frameworks.

1. INTRODUCTION

Currently the Unified Modeling Language (UML) is taught in higher education institutes across New Zealand and internationally and it is supported by all major CASE tools. It is considered to be a universal modelling language (Engels, Heckel and Sauer, 2000). This article investigates how the UML meets the demands of object-oriented software design and some of its limitations, especially when used to describe design patterns. Section 2 covers the history of UML and in section 3 the impact of the OMG meta-model is described. Section 4 covers the major strengths and weaknesses of UML. Section 5 introduces design patterns and their relationship to frameworks and section 6 describes the relationship between UML and design patterns. A timeline describing the major events in both areas is provided to show their evolution.

2. HISTORY OF UML

UML is an amalgamation of the individual work of three methodologists; Rumbaugh, Jacobson and Booch (1999). They developed modelling languages appropriate for the object-oriented languages that were becoming increasingly important to the software development community in the early 1990s. In 1997 the Object Management Group (OMG) adopted the Unified Modeling Language (UML) as a standard modelling language (Kobryn, 1999). OMG is made up of about 800 international organisations and individuals involved in software development (OMG, 2001). In the same year the International Standards Organisation (ISO) formally recognised UML version 1.1 as an international standard for information technology and an international team of vendors and system integrators have made controlled refinements from that time (Kobryn, 1999). The OMG has recently accepted version 2.0 of the UML (not published at the time of submission of this article).

3. UML AND THE META-MODEL

The present OMG standard is based on a common UML meta-model. The meta-model is an abstract class diagram and a set of semantic and syntactic rules that defines the core elements and relationships used in UML. All other terms are derived from this core set. The existence of a meta-model for UML means that it is more systematically and internally consistent than any earlier software modelling technology. It also means that tool vendors and modellers know exactly how to, for example, transform an element in a class diagram into its equivalent element in a sequence diagram or a statechart.

The meta-model allows for additions to the UML kernel language with a concept called extensions. Extensions are graphical additions to the existing symbol set. The extensions are stereotypes, tags or constraints. An extension mechanism is used to describe some particular domain or incorporate a specialist notation system. These new notations can then be incorporated into UML diagrams and used by UML tools. Extensions (especially stereotypes) allow

for modifications to the standard UML from slight changes to a complete redefinition of the base language (Berner, Glinz and Joos, 1999) and enable UML to describe any problem domain (Engels, Heckel and Sauer, 2000).

OCL (Object Constraint Language) is also part of the UML and provides a formal rule-based notation that can be used in 'design by contract' object system development.

OMG has also provided for the automatic generation of Interface Definition Language (IDL) and Extensible Mark-up Language (XML) from UML diagrams. This is possible because UML has a meta-model consistent with that of other OMG standards. This makes UML considerably more useful to organisations that want to model their business using UML, then generate IDL interfaces from UML, and also generate XML for data interchange across distributed systems. Use of UML at this level is named Architecture Driven development and is especially important to e-business development.

UML is presently undergoing a refinement so that it will meet the MOF (OMG Meta Object Facility) meta-model standards. Then it can be used in conjunction with the international CORBA standard, enhancing both standards and enabling interoperability across distributed platforms. This refinement will also allow for the extension of UML into the domain of business modelling and provide a framework for corporate applications (Harmon, 2001).

The goal of this type of standardisation is to allow systems developed on different platforms using different software and database systems to interact transparently with one another (Harmon, 2001).

4. STRENGTHS AND WEAKNESSES OF UML

UML provides the modeller with the tools to design a system. UML is used to "visualise, specify, construct and document the artefacts of a software system" (Booch et al., 1999, p. 3). UML models provide: a method of communication between development-teams, project documentation and a contract between developers and customers. Standardised UML provides a common interpretation of the language (Engels, Hausmann, Heckel and Sauer, 2000) and controlled changes to the standard UML specification make it possible for vendors to update their products (e.g. CASE tools) to meet the specifications of particular versions.

Reasons for the success of UML include (Kobryn, 1999):

- Timing and positioning - in the 1990s a standardised methodology for designing object-oriented software systems was needed.
- Proven concepts - were available from the software development field to base a modelling language upon.
- Rigorous UML development process - a team of committed and knowledgeable people were available to develop the process in a controlled and systematic way.
- Robust architecture - from its inception the UML was designed to be scalable and flexible.

Kobryn also believes that UML is able to precisely specify platform infrastructures and business component frameworks something that was difficult to do in the pre-UML modelling era.

However standard UML is known to have significant problems in its extension mechanisms as they are not fully described or consistently named and organised. Kobryn (1999) notes that examples and guidelines for using extension mechanisms are not available for advanced applications of the language. Bennett, McRobb and Farmer (1999) note that there are incomplete semantics and notation for activity graph and both Kobryn (1999) and Harmon (2001) report that because UML is a general modelling language, its size and expressive power overwhelm some developers. Kobryn called this problem semantic bloat (too many different symbols and symbols with loosely defined meanings).

Another weakness is the statechart which describes the behaviour of an object over time. Each state that an object can enter (dynamic behaviour) is defined in a statechart, however the dynamic interaction of the modelled object with its associated objects is not part of a statechart and it is not shown in any of the other UML diagrams. Researchers have tackled this problem in various ways using techniques such as graph transformation of collaboration diagrams (Engels and Heckel, 2000) and Fontoura and Lucena (2001) use extension mechanisms to depict object interactions and recommend the use of roles to clarify object interactions. Engels, Groenewegen and Kappel (2000) discuss the problems of modelling the coordinated collaboration of objects. They argue that to model realistic situations, "much more detailed and fine-grained modelling expressivity is needed" (p. 309). These situations occur where a mixture of both synchronous and asynchronous message passing is occurring and needs to be described with an appropriate model.

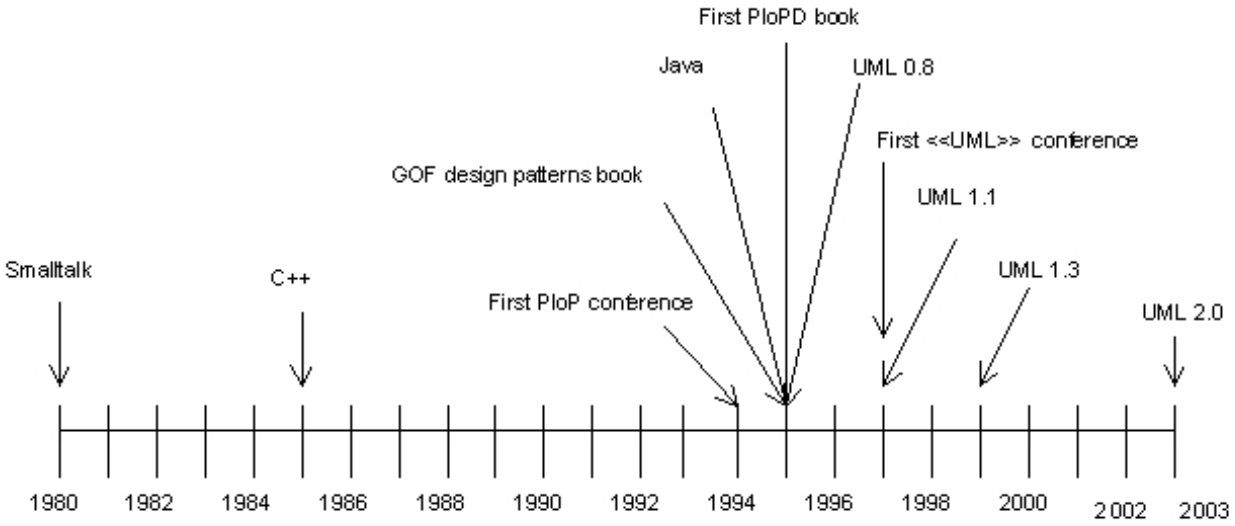

Figure 1: Timeline showing common O-O languages, UML and patterns movement development.

The implementation diagrams (component and deployment diagrams) are two UML models reported by Engels *et al.* (2000) as needing enhancement. Yacoub and Ammar (2000) developed the 'pattern diagram' to compensate for the problems they encountered in modelling this high-level aspect of their system. Glinz (2000) specifies nine deficiencies of UML version 1.3 especially in the area of requirements modelling and reiterates others findings regarding the weakness of the language when used to describe system and subsystem behaviour.

The UML provides many positive benefits but problems remain with the inability of the language to model all situations.

5. DESIGN PATTERNS

Engineers in the physical sciences reuse existing proven solutions to solve the problems of construction each time they build. Software engineers believe that this same technique will improve the success and quality of software systems and reduce the time and cost of system development. The pattern movement describes problems and proven solutions for use in the development of systems written using the object-oriented programming languages. Reuse of experience and design is the goal.

A design pattern is a clearly documented description of a problem that recurs in software development, and it's solution. Developers can reuse a pattern at different times and the final product, or instantiation of the pattern, is different each time it is used. The 'Gang of Four' (GOF) described the idea of patterns to the wider software development community (Gamma, Helm, Johnson and Vlissides, 1995). After the publishing of this seminal work, PloP (Pattern Languages of Design) conferences were held to add to and revise the work on patterns. Following the conferences books describing these patterns (the PloPD books) were published.

As shown in figure 1 the development of object-oriented languages was followed by the beginning of the patterns movement, which was followed by the standardisation of UML.

Patterns are used to produce software products. A major application for design patterns is their use in application frameworks. Yacoub and Ammar (2000) describe a framework made up of design patterns. The design patterns are 'glued' together to form a framework that is then instantiated for use. Fayad (2000) describes a framework as a re-usable application skeleton that can be specialised to build a new application in a specific domain. A white-box framework is a framework that requires the developer to understand the structure of the framework and the hot spots where application-specific functions are added, whereas a black-box framework provides only executable code and code extensions are simply added to create the application. A general approach to framework construction and instantiation is described by Braga and Masiero (2001) and a simplified version is shown in figure 2.

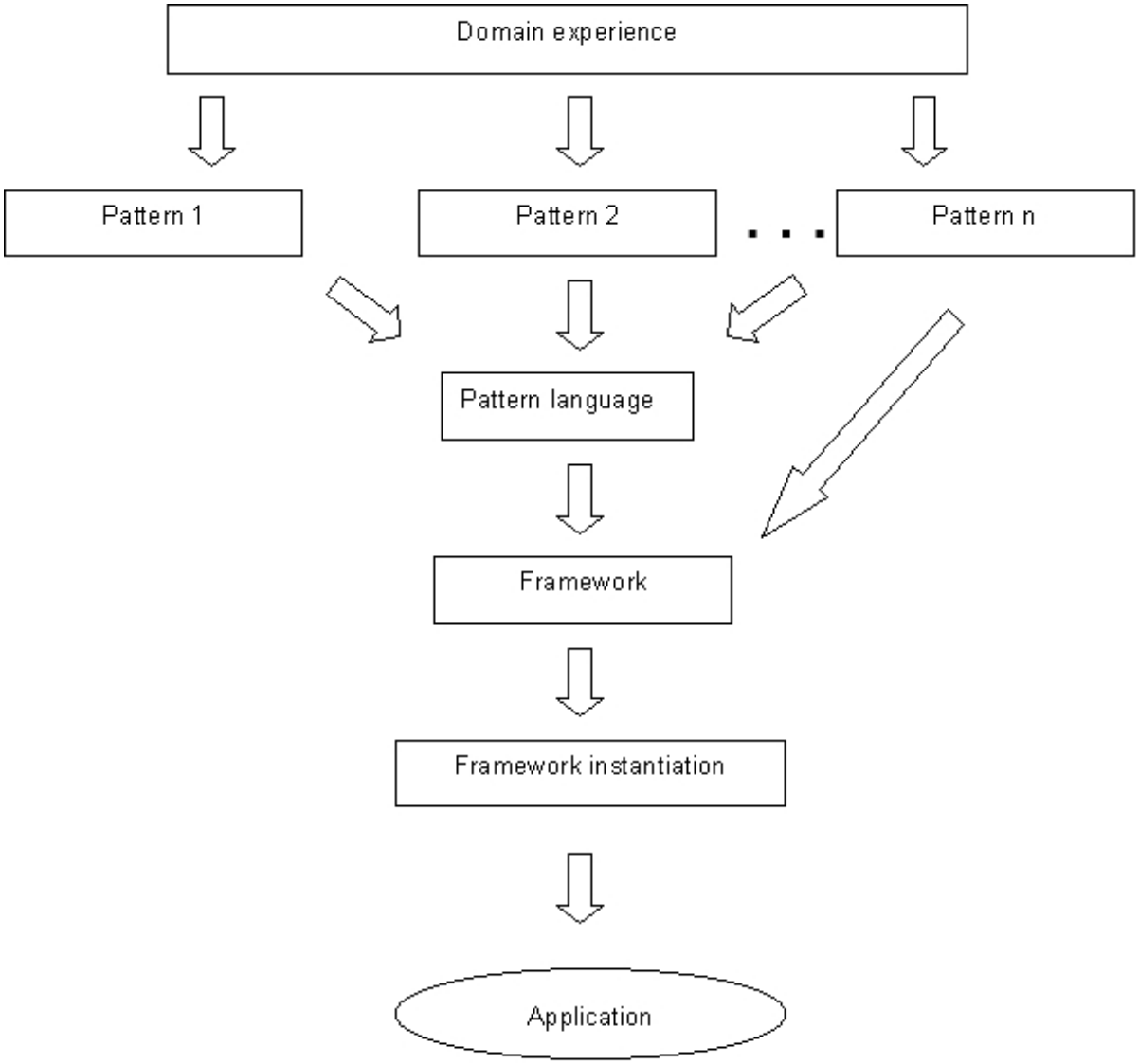

Figure 2: The relationship between patterns, pattern languages, frameworks and applications.

6. UML FOR PATTERN DESCRIPTION

A pattern must be described in a rigorous way so that it can be classified, identified and compared with other patterns. Most design pattern solutions are described using a mixture of narrative and UML. There are different methods for specifying patterns such as those given by Gamma *et al.* (1995), and The Hillside Group (2001). Gamma *et al.* (1995) recommended that the pattern solution is described using the Object Modeling Technique (OMT) developed by Rumbaugh, Blaha, Premerlani, Eddy and Lorensen (1991) and interaction diagrams developed by Jacobson, Christerson, Jonsson and Overgaard (1992) and Booch (1994). However there is no explicit requirement to describe design patterns in anything more specific than narrative, and dynamic behaviour is typically described with narrative, if it is addressed at all. This could be because of the recognised weaknesses of the statechart and activity diagram notation, and the lack of a diagram that completely specifies dynamic object interactions.

When version 2.0 of the UML enhances the implementation diagrams this will enable higher-level architecture of systems to be modelled more clearly and will have an impact on the pattern language process and the development of domain-specific frameworks from these patterns. As each new version of the UML

is published by the OMG, patterns can be reworked using the newer improved notation. The patterns could then be improved to better describe the dynamic behaviour of a system, which is an integral part of the dynamic interplay between the objects of the system. Strengthening UML in these areas may also provide a means to describe a new subset of design patterns - object collaboration-interaction patterns.

7. CONCLUSION

UML is not perfect because it is a universal modelling language rather than a domain-specific language. The enhancement of the UML standard to meet the MOF specification will improve the ability of designers to design and develop system-wide architectures incorporating CORBA, XML and other standards. Improvements to UML specification in the area of dynamic behaviour will mean the descriptions of solutions of design patterns can be improved, pattern languages can be improved, and frameworks will be easier to design. Some problems with UML will be resolved in version 2.0, and this may lead to new design patterns emerging that effectively model object-collaboration.

REFERENCES

Bennett, S., McRobb, S. & Farmer, R. (1999). Object-oriented systems analysis and design using UML. McGraw-Hill: England.

Berner, S., Glinz, M. & Joos, S. (1999). A classification of stereotypes for object-oriented modeling languages. Proceedings of the Second International Conference on the Unified Modeling Language, p. 249-264. Retrieved 24 October 2001. http://www.ifi.unizh.ch/groups/req/staff/glinz/activities.html

Booch, G. (1994). Object-Oriented Analysis and Design with Applications (2nd Ed.). Benjamin/Cummings: Redwood City, CA.

Braga, R. T. V. & Masiero, P. C. (2001). Position Paper. 1st ASERC Workshop on Software Architecture, 24-25. http://www.icmsc.sc.usp.br/~rtvb/ingles.html

Engels, G., Hausmann, J. H., Heckel, R. & Sauer, S. (2000). Dynamic meta modeling: a graphical approach to the operational semantics of behavioural diagrams in UML. In Proc. UML 2000,.LNCS 1939, p. 323-337. Evans, A., Kent, S., & Selic, B. (Eds.). Berlin: Springer-Verlag. www.upb.de/cs/ag-engels/Papers/2000/EngelsUMLOO.pdf

Engels, G., Groenewegen, L. & Kappal, G. (2000). Coordinated collaboration of objects. Advances in Object-oriented Data Modeling. Papazoglou, M., Schmidt, J. W. & Mylopoulos, J. (Eds.). MIT Press: Massachusetts.

Engels, G. & Heckel, R. (2000). Graph transformation as a conceptual and formal framework for system modeling and model evolution. Retrieved 24 October 2001, www.uni-paderborn.de/cs/ag-engels/Papers/2000/EngelsIcalpOO.pdf

Engels, G., Heckel, R. & Sauer, S. (2000). UML - a universal modeling language? ICATPN 2000, LNCS 1825, pp. 24-38. Springer-Verlag.

Fayad, M. E. (2000). Introduction to the computing surveys' electronic symposium on object-oriented application frameworks. ACM Computing Surveys, Vol 32, No1 p. 1-9.

Fontoura, M. & Lucena, C. J. P. (2001, March). Extending UML to improve the representation of design patterns. Journal of Object-Oriented Programming, 12 - 19.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). Design Patterns: elements of reusable object-oriented software. Addison-Wesley: Massachusetts.

Glinz, M. (2000). Problems and deficiencies of UML as a requirements specification language. Proceedings of the 10th International Workshop on Software Specification and Design, p.11 - 22.

Harmon, P. (2001, April - May). UML models e-business. Software Magazine. 3-15.

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

Kobryn, C. (1999). UML 2001: a standardization odyssey. Communications of the ACM, vol. 42, No. 10.

Object Management Group, retrieved October 2001. http://www.omg.org

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorenson, W. (1991). Object-Oriented Modeling and Design. Englewood Cliffs, Prentice Hall: NJ.

Rumbaugh, J., Jacobson, I. & Booch, G. (1999). The unified modeling language reference manual. Addison-Wesley Longman: Massachusetts.

The Hillside Group. (2001). Patterns and software: essential concepts and terminology. Retrieved October 24, 2001, from http://

www.enteract.com/~bradapp/docs/patterns-intro.html

Yacoub, S. M & Ammar, H. H. (January 2000). Toward pattern-oriented frameworks. Journal of Object-oriented Programming, p. 25 - 35.