Column 2. Views on Architectural Modelling

Object-Oriented Modelling with Situation Theory

Author: Eugene Doma, University of Sydney, Australia, David Levy, University of Sydney, Australia and Bran, Selic Malina Software Corp, Canada

Despite early skepticism about its practical viability, object-oriented programming (OOP) as well as object-oriented analysis and design are now widely used in industrial practice. The emergence of this paradigm was accompanied by a proliferation of object-oriented analysis and design notations, which have since coalesced into the Unified Modeling Language (UML) [11], which is now widely supported by numerous commercial and open-source tools. This has, in turn, given rise to model-based software engineering (MBSE) (also frequently referred to as model-driven development (MDD)), an innovative approach to software development characterized by use of higher levels of abstraction when compared to traditional programming methods.

Although many software projects owe their success to MBSE processes and tools [10, 17], the level of adoption for MBSE is far lower than might be expected. The need for more core research and a systematic theoretical foundation have been identified as one of the key issues [13] posing obstacles to wider adoption. At the University of Sydney we are exploring the combination of situation theory [2, 7] and the object paradigm as a candidate theoretical framework upon which to base MBSE.

Within this framework, situation theory provides the lowest level of detailed modeling in our system. For specific domains, such as business systems, we compose larger objects that express domain specific knowledge and associated patterns and which facilitate construction of larger systems in the relevant domain. These larger objects are composed in accordance with the axioms of the underlying situation theory and therefore preserve the rigor and precision of situation theory, while freeing the user to work at a higher level of abstraction. This process, of building meta-models, is repeatable, where useful, to levels of abstraction as high as appropriate for the domain.

A simplified view of OOP considers objects to encapsulate both state and behaviour. To eliminate undue coupling, processing is accomplished by sending messages, which are also objects, to collaborator objects and receiving messages in response [9]. Further developments with actor languages [1] have integrated objects and processes, resulting in so-called active objects, such as defined in the UML v2 standard.

Situation theory, on the other hand, \views the world as a collection of objects, relationships between objects and properties of those objects"[16], and, thus, is compatible with an object-oriented modeling approach. In this view, in addition to the traditional passive and active objects, objects may be used to represent processes, data ows and even imperative constructs.

The fundamental unit of information in situation theory is the infon [8], a term deliberately intended "to draw a parallel with `elementary' particles of physics"[6]. In standard linear notation an infon is represented as :

$$<< r, a_1, a_2; \ldots a_n ; p>>$$

where `r' is a relation; `$a_1 \ldots a_n$' are arguments, and the `p' is the polarity of the infon. A polarity value of false is used to indicate contra-factuals. Each infon defines a single relationship between the specified arguments. A logical conjunction of infons is used to represent a situation. Situations may in turn be arguments to other infons and, thus, other situations.

We consider infons and situations to be fundamental units from which we may synthesise objects as recognised in OO analysis and design terms. Infon algebra [6] de_nes a set of core relations and operations on infons. This mathematically rigorous foundation may be used to provide formal proofs, simulations and traceability.

A more detailed treatment of situation theory may be found in a foundation study [15] and the "working person's guide"[5].

For the purposes of sharing our vision of a situation theoretic foundation for MBSE, we note that situation theory has been extensively used by linguists to analyse the structure and meaning of utterances and the situations that they represent; rather like the process by which requirements are gathered and analysed.

The statements about requirements elicited from stakeholders are encoded in declarative domain-specific modeling languages. The declarative nature stems from the need to capture what is required without prematurely making implementation decisions. During the requirements gathering stage, several different, yet inter-related, models are built; the models have the same relation to the desired outcomes as models have in other areas of engineering, for example, clay models of automobiles, architectural scale models of proposed buildings, computer models of bridges, aeroplanes. These models portray some aspect of what is being engineered, facilitating analysis and simulation. With computer-based models, engineers in other engineering disciplines undertake many explorations of various facets and design choices before deciding on a single one.

Unlike other areas of engineering, the lack of physical constraints in software design is sometimes a disadvantage. Models, especially during early stages, have many critical details missing. Furthermore, the network of mutually related model elements quickly becomes complex [14]. Situation theory together with infon algebra promises to provide us with the means by which to effectively address these issues.

Most projects are likely to require several different kinds of models, depending on the viewpoint of different stakeholders, and thus may require multiple viewpoint-specific (or, more commonly, domain-specific modeling languages. Situation theory may be used here as a meta-meta-modeling language [3], by means of which to define such languages. Regardless of which domain-specific modeling language is used, the model elements are all encoded as infons. In turn, model elements are able to be referenced across different models and levels of abstraction. Model-to-model transformations preserve the underlying relationships and enable full traceability for all model elements.

The deterministic and formal nature of infons makes it possible to implement executable models which may be subjected to a variety of simulations. The results from the simulations provide further

details to assist with design and implementation decisions.

Since requirements models typically remain decoupled from the implementation, the generator may produce deployments based upon functions, shared libraries, threads, processes, services, clusters, clouds or hybrid solutions including digital signal processors and FPGAs.

Conceptually situation theory and object-oriented analysis, design and programming practices have much in common. Domain-specific models may be synthesised using infons and form a framework for MBSE processes. At the University of Sydney we continue to explore situation theory with a view to developing a robust modeling framework and toolset based on well established mathematical foundations, emulating our colleagues in other branches of science and engineering. We have a prototype situation-theoretic toolset, InfonLab, which we are using to model business processes, such as those defined in the Business Analysis Body of Knowledge [4]. InfonLab gives us the means to explore a wide range of executable models, providing a sound foundation to advance the field of MBSE into different domains and using various design paradigms.

REFERENCES

1. G. Agha. An overview of actor languages. In Proceedings of the 1986 SIGPLAN workshop on Object-oriented programming, pages 58-67. ACM, 1986.

2. J. Barwise. The Situation in Logic. CLSI, Standford, CA, 1989.

3. J. B_ezivin. On the uni_cation power of models. Software and Systems Modeling, 4(2):171-188, 2005.

4. K. Brennan. A Guide to the Business Analysis Body of Knowledge (BABOK Guide). International Institute of Business Analysis, second edition, 2009.

5. R. Cooper. A working person's guide to situation theory. In Ste_en Leo Hansen and Finn Soerensen, editors, Topics in Semantic Interpretation. Samfundslitteratur, Frederiksberg, Denmark, 1991.

6. K.J. Devlin. Infons as mathematical objects. Minds and Machines, 2(2):185-201, 1992.

7. K.J. Devlin. Situation theory and situation semantics. In D.M. Gabbay and J. Woods, editors, Logic and the Modalities in the Twentieth Century, volume 7 of Handbook of the History of Logic, pages 601-664. Elsevier, Amsterdam, 2006.

8. T. Fernando. On the logic of situation theory. In R. Cooper, K. Mukai, and J. Perry, editors, Situation theory and its applications, volume 1, pages 97-116.CLSI, Standford, CA, 1990.

9. A. Goldberg. Why smalltalk? Commun. ACM, 38(10):105-107, 1995.

10. N.J. Nunes et al. Industry track papers. In UML Modeling Languages and Applications, volume LNCS 3297, pages 94-233. hhUMLii 2004 Satellite Activities, Lisbon, Portugal, October 2004, Springer-Verlang, 2004.

11. Object Management Group (OMG). Uni_ed Modelling Language (UML) Superstructure Speci_cations, v2.3, 2009.

12. B.V. Selic. On the semantic foundations of standard uml 2.0. Formal Methods for the Design of Real-Time Systems, LNCS 3185:181-199, 2004.

13. B.V. Selic. MDA Manifestations. The European Journal for the Informatics Professional, IX(2), 2008.

14. B.V. Selic. Personal reections on automation, programming culture, and model based software engineering. Automated Software Engineering, 15(3):379-391, 2008.

15. J. Seligman and L.S. Moss. Situation theory. In Johan vanBenthem and Alice ter Meulen, editors, Handbook of Logic and Language, pages 239-309. Elsevier, 1997.

16. E. Tin and V. Akman. Situations and computation: An overview of recent research. In Proceedings of Topics in Constraint Grammar Formalism for Computational Linguistics: Papers Presented at the Workshop on Grammar Formalisms for Natural Language Processing held at ESSLLI-94, Copenhagen, Denmark, pages 77-106, 1995.

17. T. Weigert and F. Weil. Practical experiences in using model-driven engineering to develop trustworthy computing systems. In IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, volume 1, pages 208-217, 2006.