The Role of Formal Methods in Component-Based Software Engineering

P.S.C. Alencar D.D. Cowan

Computer Science Department

University of Waterloo

Waterloo, Ontario N2L 3G1

{palencar,dcowan}@csg.uwaterloo.ca

Abstract.

In this chapter we intend to present a survey recent progress in the development and application of formal techniques for component-based software systems. This chapter may be included in part 4 or part 3 of the proposed book outline. As central issues, we discuss formal methods and their applications to component descriptions (such as COM) and architecture descriptions (such as UNICOM). As a starting point, we discuss formal models for the specification and verification of Module Interconnection Languages (MILs). Topics that we plan to discuss in this chapter include the precise specifications of component-based and architectural models and their properties, automated analysis of the models, and use of design constraints to provide guarantees about, for example, and the system structure and behavior. Finally, we discuss some general promising research directions in this area.

The main idea of this chapter is to describe and emphasize the current and future (potential) benefits of the interaction between formal methods and component-based software engineering. The content of the chapter, although ``formal'', should be presented in a ``user friendly'' way. The chapter may be adapted to fit the negotiated final content of the book; for example, we can restrict ourselves to component descriptions and omit section 5 on the formalization of ADLs.

1. Introduction

Component-based software engineering has been widely accepted to be an engineering discipline. Typically, all engineering disciplines today are based on some theoretical foundations, and these foundations are a basis for understanding the involved engineering tasks, rules, procedures, and processes. Also, component-based software engineering needs its own theoretical foundations like any other engineering discipline.

The goal of the following proposed sections is to describe the current state of the application of formal techniques to component-based software engineering and give some indication about promising related research directions in the area.

2. Formal Methods

In the context of software, the term ``formal methods'' refers to the use of techniques from formal logic and discrete mathematics in the specification, design, and construction of computer software [7,8,9,10]. By using formal or rigorous specifications, much of the ambiguity that is found inevitably in informal

specifications can be eliminated. The use of formal specifications allows the software systems to be analyzed in various useful ways: many ways. Formal proofs eliminate the subjectivity and ambiguity of requirements by providing a logical and precise argument of the behavior of the requirements. The use of formal specifications and formal proofs also provide a systematic approach to analysis. Formal specifications and formal analysis can be applied at any life cycle phase, can be supported by computer-based tools, and can complement (formal) testing approaches [11,12].

In this section, we plan to briefly describe some representative formal techniques relevant to this chapter will be presented, such as model-based, state-based, and event-based.

3. Module Interconnection Languages (MILs)

Proposed as a significant step towards programming in a higher level of abstraction, Module Interconnection Languages (MILs) were originally proposed in [1]. Extensions, such as object-oriented MILs, were also later proposed [2].

MILs were formalized in [3]. A formal specification language, Z [15], was used to describe the MIL models. The model allowed the basic MIL components to be instantiated and connected to each other. The general MIL model was instantiated into two well-known MIL variants. Illustrative parts of the MIL models will be shown. See [14] for an alternative formalization. The benefits of these formalizations are discussed.

We also discuss how extended MIL models can be formally analyzed. We describe some work on analyzing the MIL models with respect to their structural properties and their configurations. We describe how reasoning about changes can be performed in this context [4,5].

4. Component Models

In this section we describe applications of formal methods related to representative component models. We plan to describe current efforts to specify general component models, as well formal aspects related to COM (CORBA, and Java BEANS) and formally analyzing these models. Illustrative COM formal specifications and verifications based on the formal models can be found in [20,31,32].

We also briefly describe some of the efforts towards component-based software composition [34-40,6] and other related instances where formal methods were applied [33].

5. Architectural Description Languages

In this section we describe applications of formal methods related to ADLs [13,16-19,21-24,29,30,42,43]. We plan to concentrate not so much on the description of the formal basis for some representative ADLs, but on applications taking advantage of the formal descriptions such as formal analysis and testing. Formal analysis includes verification techniques (theorem proving and model checking [26,27]).

6. Research Directions

Promising research directions related to the application of formal methods in CBSE are presented. We will discuss some issues about the specification, formal design, refinement, formal analysis and testing (including specification-based testing), maintenance (including adaptation of components), and formal

retrieval of components.

References

[1] DeRemer, F., Kron, H., Programming-in-the-Large versus Programming-in-the-Small, IEEE Transactions on Software Engineering, pp. 321-327, June 1976.

[2] Hall, P., Weedon, R., Object-Oriented Module Interconnection Languages, Advances in Software Reuse, Selected papers from the Second International Workshop on Software Reusability, pp. 29-38, March 24-26, Lucca, Italy, 1993.

[3] Rice, M., Seidman, S., A Formal Model for Module Interconnection Languages, IEEE Transactions on Software Engineering, 20, pp. 80-101, January 1994.

[4] Alencar, P. S. C., Lucena, C. J. P., A Logical Framework for Evolving Software Systems, Formal Aspects of Computing, vol. 8, pp. 3-46, 1996.

[5] Lucena, C.J.P., Alencar, P. S. C., A Formal Description of Evolving Software Systems Architectures, Science of Programming vol. 24, pp. 41-61, 1995.

[6] Alencar, P.S.C., Cowan, D.D., CASCON'98 Workshop on Component-based Software Composition, Dec. 3, 1998, Toronto, Ontario, IBM Report, 1999. Also, CASCON'9 Workshop on Patterns and Frameworks, 1997, Toronto, Ontario, IBM Report, 1998.

[7] Craigen, D., Gerhart, S., Ralston, T., An International Survey of Industrial Applications of Formal Methods, Volume I: Purpose, Approach, Analysis and Conclusions, National Institute of Standards and Technology (NIST), Gaithersburg, MD, March 1993.

[8] Craigen, D., Gerhart, S., Ralston, T., An International Survey of Industrial Applications of Formal Methods, Volume II: Case Studies, National Institute of Standards and Technology (NIST), Gaithersburg, MD, March 1993.

[9] Clarke, E.M., Wing, Jeannette, Formal Methods: State of the Art and Future Directions, ACM Computing Surveys 28, 4, pp. 626-643, December 1996.

[10] Rushby, J., Formal Methods and the Certification of Critical Systems, SRI-CSL-93-07, Menlo Park, California, SRI International, November 1993.

[11] National Aeronautics and Space Administration, Formal Methods for Specification and Verification Guidebook for Software and Computer Systems, Volume I: Planning and Technology Insertion, Office of Safety and Mission Assurance, Washington, D.C., July 1995.

[12]] National Aeronautics and Space Administration, Formal Methods for Specification and Verification Guidebook for Software and Computer Systems, Volume I: Planning and Technology Insertion, Office of Safety and Mission Assurance, Washington, D.C., July 1995.

[13] Shaw, M., and Garlan, D., Software Architecture: An Emerging Discipline. Prentice-Hall, Englewood Cliffs, NJ, 1996.

[14] Dean, T.R. and Lamb, D.A., A Theory Model Core for Module Interconnection Languages, Proceedings of CASCON'94 – Integrated Solutions, Toronto, Ontario, Oct. 31-Nov. 3, pp. 1-8, IBM Centre for Advanced Studies, 1994.

[15] Spivey, J.M., The Z Notation: A Reference Manual, ISICS. Prentice-Hall, second edition, 1992.

[16] Clements, P. and Northrop, L., Software Architecture: An Executive Overview, Tech. Rep. CMU/SEI-96-TR-003, Software Engineering Institute, February 1996.

[17] Shaw, M., DeLine, R., Klein, D.V., Ross, T.L., Young, D.M. and Zelesnik, G., Abstractions for Software Architecture and Tools to Support Them, IEEE Transactions on Software Engineering, vol. 21, no. 9, pp. 717-734, Sept. 1995.

[18] Luckham, D.C. and Vera, J., An Event-Based Architecture Definition Language, IEEE Transactions on Software Engineering, vol. 21, no. 9, pp. 717-734, September 1995.

[19] Garlan, D. and Perry, D.R., Introduction to the Special Issue on Software Architecture}, pp. 269-274, 1995.

[20] Sullivan, K.J., Socha, J., Marchukov, M., Using Formal Methods to Reason about Architectural Standards, Proceedings of the 19th International Conference on Software Engineering, (ICSE'97), pp. 503-513, 1997.

[21]Allen, R.J., Garlan, D., Ivers, J., Formal Modeling and Analysis of the HLA Component Integration Standard, Proceedings of the 6th International Symposium on the Foundations of Software Engineering (FSE-6), November 3-5, 1998 (to appear).

[22] Moriconi, M., Qian, X., Riemenschneider, R., Correct Architecture Refinement, IEEE Transactions on Software Engineering, vol. 21, no. 4, pp. 356-372, April 1995.

[23] Allen, R. and Garlan, D., {\it A Formal Basis for Architectural Connection, ACM Transactions on Software Engineering and Methodology, July 1997.

[24] Magee, J., Dulay, N., Eisenbach, S., and Kramer, J., Specifying Distributed Software Architectures, Proceedings of ESEC'95, September 1995.

[25] Zhang, X., A Rigorous Approach to Comparison of Representational Properties of Object-Oriented Analysis and Design Methods, PhD Thesis, Queen's University, August 1997.

[26] Gordon, M.J.C. and Melham, T.F., Introduction to HOL: A Theorem Proving Environment for Higher Order

Logic}, Cambridge University Press, New York, 1993.

[27] Owre, S., Rushby, J., Shankar, N., PVS: A Prototype Verification Sytem, Proceedings of the 11th International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science, vol. 607, pp. 748-752, Springer-Verlag, June 1992.

[28] Hoare, C.A.R., Communicating Sequential Processes, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[29] Lichtner, K., Alencar, P.S.C., Cowan, D.D., Using View-Based Models to Formalize Architectural Description, Proceedings of the International Software Architecture Workshop (ISAW'98), FSE'98, ACM SIGSOFT, pp. 97-100, November 1998.

[30] Lichtner, K., Alencar, P.S.C., Cowan, D.D., Formalizing Architecturre Description Languages, Technical Report CS-98-07, Computer Science Department, University of Waterloo, 1998.

[31] Outhred, G., Potter, J., Extending COM's Aggregation Model, Component-Oriented Software Engineering Workshop (COSE'98), in conjunction with Australian Software Engineering Conference (ASWEC'98), to appear, November 1998.

[32] Ibrahim R., Component-Based Systems: A Formal Approach, , Component-Oriented Software Engineering Workshop (COSE'98), in conjunction with Australian Software Engineering Conference (ASWEC'98), to appear, November 1998.

[33] Bumbulis, P., Alencar, P.S.C., Cowan, D.D., Lucena, C.J.P., Validating Properties of Component-Based Graphical User Interfaces Proceedings of the 3rd Eurographics Workshop on Design, Specification, Verification of Interactive Systems, F. Bodart and J. Vanderdonckt (eds.), pp. 347-365, Namur, Belgium, June 5-7, DSV-IS'96, FNRS, Springer-Verlag, 1996.

[34] ATP Focused Program: Component-Based Software, National Institute of Standards and Technology Gaithersburg Maryland, http://www.atp.nist.gov/atp/focus/cbs.htm, http://www.atp.nist.gov/www/press/9706cbs.htm

[35] Nierstrasz O., Tsichritzis D., Object-oriented Software Composition Prentice Hall 1995.

[36] International Workshop on *Large-Scale Software Composition* held in conjunction with *DEXA'98 (Vienna, Austria, Aug. 24-28), 1998, http://www.iro.montreal.ca/~keller/Workshops/DEXA98/index.html.*

[37] CAiSE*98 Ws. on Component-based Inf. Systems Eng. (CBISE'98), Pisa, Italy, 8-9 June 1998, http://www.cs.waikato.ac.nz/~jgrundy/caise98_workshop/.

[38] Intl. Ws. on Component-based Software Engineering, Kyoto, Japan, April 25-26, 1998, http://www.sei.cmu.edu/cbs/icsewkshp.html.

[39] Workshop on Compositional Software Architectures, Monterey, CA, USA, 6-8 January, 1998, http://www.objs.com/workshops/ws9801/.

[40] A. Brown K. Wallnau, "Engineering of Component-Based Systems" in Component-Based Software Engineering, IEEE Computer Society Press, Los Alamitos, CA., 1996.

[41] Medvidovic, N. Rosenblum, D.S., Taylor, R.N., A Type Theory for Software Architectures, Department of Information and Computer Science, University of California, Irvine, Technical Report UCI-CS-98-04, February 1998.

[42] Rosenblum, D.S., Challenges in Exploiting Architectural Models for Software Testing, Proceedings of the NSF/CNR Workshop on the Role of Software Architecture Testing and Analysis, pp. 49-53, July

1998.

[43] Penix, J. and P. Alexander, ``Formal Specifications for Component Retrieval and Reuse," *Hawai'i International Conference on Systems Sciences (HICSS 98)*, June 5, 1997.

[44] Penix, J. and P. Alexander, ``Declarative Specification of Software Architectures," *Automated Software Engineering Conference (ASE 97)*, November 2-5, 1997.