SAVCBS 2001 Proceedings

Specification and Verification of Component-Based Systems
Workshop at OOPSLA 2001

Dimitra Giannakopoulou, Gary T. Leavens, and Murali Sitaraman (editors)

TR #01-09a
October 14, 2001, revised November 6, 2001

Keywords: Specification, verification, component-based systems.

2000 CR Categories: D.1.m [*Programming Techniques*] Miscellaneous — component-based programming, reflection; D.2.1 [*Software Engineering*] Requirements/Specifications — languages, methodology, theory, tools; D.2.4 [*Software Engineering*] Software/Program Verification — assertion checkers, class invariants, correctness proofs, formal methods, model checking, programming by contract, reliability, validation; D.2.5 [*Software Engineering*] Testing and Debugging — testing tools; D.2.11 [*Software Engineering*] Software Architecture — languages; D.2.m [*Software Engineering*] Miscellaneous — component-based systems, reusable software; D.3.1 [*Programming Languages*] Formal Definitions and Theory — semantics; D.3.3 [*Programming Languages*] Language Constructs and Features — data types and structures; F.3.1 [*Logics and Meaning of Programs*] Specifying and verifying and reasoning about programs — assertions, invariants, logics of programs, pre- and post-conditions, specification techniques; F.3.m [*Logics and Meaning of Programs*] Miscellaneous — reasoning about performance.

Department of Computer Science
226 Atanasoff Hall
Iowa State University
Ames, Iowa 50011-1040, USA

Table of Contents

Session I: Specification-Based Testing and Run-Time Analysis

Session II: Architecture and Composition

Session III: Keynote

Session IV: Compositional Verification

Session V: Discussion

Other Accepted Papers

Preface

The goal of this workshop was to explore how formal (i.e., mathematical) techniques can be or should be used to establish a suitable foundation for specification and verification of component-based systems. Component-based systems are a growing concern for the object-oriented community. Specification and reasoning techniques are urgently needed to permit composition of systems from components, for which source code is unavailable.

We wanted to bring together researchers and practitioners in the areas of component-based software and formal methods, to address the specification and verification problems. Several representatives from Microsoft research attended the workshop, and presented their approach to specification and verification in the context of Microsoft products. However, it was generally agreed that a lot remains to be done to address the needs of industry. On the other hand, papers on testing, run-time checking of assertions, and the use of message sequence charts addressed more practical concerns. Another goal was to focus more of the effort in formal methods on component-based systems; time will tell if we have contributed to realizing this goal.

The main expected result of the meeting would be an outline of collaborative research topics and a list of areas for further exploration. Some of these ideas were presented in our OOPSLA poster.

The papers at the workshop and those included in the proceedings were selected from papers submitted by researchers worldwide. Due to time limitations at the workshop, only a few papers could be presented

The discussion at the workshop itself was quite interesting. All agreed that compositional, modular reasoning is a necessary goal in this area. We discussed several strategies for making reasoning more tractable, including proving less, checking parts of a proof at run-time (as in run-time assertion checking), decomposing proofs by using stronger specifications, and writing components in ways that make proofs easier (e.g., by limiting the use of pointers). We also discussed ways to add value to specifications, including providing support for testing and run-time assertion checking. Barnett and Schulte pointed out that in one case at Microsoft, a specification was “orders of magnitude” smaller than the code it specified. We discussed ways to extend type systems, to incorporate architectural constraints and message sequence information. Several of the techniques discussed focused on component interaction at interface boundaries, which is helpful in reasoning about compositions.

We also identified several areas that seem ripe for future work. One is putting together trace-based concurrency reasoning with reasoning about data values. Another is how to reason about performance (i.e., time and space behavior); one paper at the workshop discussed this, but there is more to be done, and this kind of reasoning is important for embedded systems. Another area is how to make reasoning easier. One direction for making reasoning easier is finding limits on programs that have a big impact on ease of reasoning. There was a lot of discussion of the idea of Weide and Heym to encapsulate references (pointers) in components, so that all variables in a program are not general references. We also talked about finding the right abstractions for reasoning about compositions. And we discussed extending type systems to incorporate more specification information, while still allowing them to be decidable and efficiently checkable.

The workshop was organized by Dimitra Giannakopoulou (NASA Ames/RIACS), Gary T. Leavens (Iowa State University), and Murali Sitaraman (Clemson University). The program committee that selected papers consisted of the organizers and Betty H. C. Cheng (Michigan State University), Steve Edwards (Virginia Tech), K. Rustan M. Leino (Compaq Systems Research Center), and Markus Lumpe (Iowa State University). We thank the organizers of OOPSLA 2001 for hosting the workshop.