Lecture Notes in Computer Science 2262
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Peter Müller

Modular Specification and Verification of Object-Oriented Programs

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Author

Peter Müller
Staufenstraße 34
60323 Frankfurt, Germany
E-mail: p.mueller@web.de

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Müller, Peter:
Modular specification and verification of object oriented programs / Peter Müller. - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Tokyo : Springer, 2002
(Lecture notes in computer science ; Vol. 2262)
ISBN 3-540-43167-5

CR Subject Classification (1998):D.2, D.3, D.1.5, F.3

ISSN 0302-9743
ISBN 3-540-43167-5 Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna
Printed on acid-free paper SPIN 10846107 06/3142 5 4 3 2 1 0

Foreword

Software systems play an increasingly important role in modern societies. Smart cards for personal identification, e-banking, software-controlled medical tools, airbags in cars, and autopilots for aircraft control are only some examples that illustrate how everyday life depends on the good behavior of software. Consequently, techniques and methods for the development of high-quality, dependable software systems are a central research topic in computer science.

A fundamental approach to this area is to use formal specification and verification. Specification languages allow one to describe the crucial properties of software systems in an abstract, mathematically precise, and implementation-independent way. By formal verification, one can then prove that an implementation really has the desired, specified properties. Although this formal methods approach has been a research topic for more than 30 years, its practical success is still restricted to domains in which development costs are of minor importance. Two aspects are crucial to widen the application area of formal methods:

- Formal specification techniques have to be smoothly integrated into the software and program development process.
- The techniques have to be applicable to reusable software components. This way, the quality gain can be exploited for more than one system, thereby justifying the higher development costs.

Starting from these considerations, Peter Müller has developed new techniques for the formal specification and verification of object-oriented software. The specification techniques are declarative and implementation-independent. They can be used for object-oriented design and programming. To illustrate the techniques and to make them directly applicable, his book develops a detailed framework for the specification and verification of classes and interfaces written in a Java subset.

The main contributions of his work concern the modularity problem. In this context, modularity means that software modules can be specified and verified independently and that their specifications and proofs remain valid under composition. In addition, the specification techniques have to be sufficiently complete to verify a module based on the specifications of the imported modules. Modularity is of critical importance for reuse and the emerging

paradigm of component-based programming, and yet something that is by far underdeveloped in the literature.

One of the difficult aspects of modular verification is the so-called frame problem, that is, the treatment of the modification behavior of methods and procedures. The frame problem is particularly interesting when software modules have an encapsulated state the implementation details of which should not be exposed to the public. Thus, the frame problem is not restricted to object-oriented programming, but is relevant for classical procedural modules as well. To handle encapsulation and modularity, state changes have to be specified in an abstract way, that is, without referring to the hidden variables. The main technique employed is to use so-called abstract variables. However, this approach leads to dependencies between abstract and concrete variables, which cause a major problem for modular verification.

The developed solution to the frame problem builds on and clearly goes beyond existing approaches. It supports very general and expressive dependencies which are important to increase the flexibility of frame property specification. In addition, the higher expressiveness allows one to apply the approach to class invariants by regarding them as abstract boolean variables. That simplifies the formal framework, because a special treatment of invariants becomes dispensable. The key to the developed solutions is a new, type-based technique for alias control. The so-called universe type system enforces a hierarchical structure in the object store and is a valuable contribution in its own right.

All presented concepts and techniques are well motivated and precisely described. In particular, the different aspects of modular programming are nicely explained so that their implications can be followed into the detailed formal framework. I can only wish that many readers take the time to dive into the deep waters of the following chapters. They will be rewarded by interesting and valuable insights and by the beauty of a coherent formal framework solving the sketched problems. I wish even more that the work encourages other researchers to further develop the theory, techniques, and tools for modular specification and verification.

October 2001 *Arnd Poetzsch-Heffter*

Preface

The paradigm shift from procedural to object-oriented programming has promoted modular software development. The reuse of prefabricated software modules has especially increased the demand for precise specifications and quality certification, and thus for *modular* specification and verification techniques. Such techniques must be capable of handling object-oriented language features such as subtyping, inheritance, and dynamic method binding, and have to support modular development of specifications and proofs. In particular, they should enable specifications and proofs to be reused along with implementations.

This book presents modular specification and verification techniques for the functional behavior, frame properties, and type invariants of OO-programs. The key idea underlying this work is the formal integration of state-of-the-art specification and verification techniques with a type system for alias control.

We present the universe type system that can be used to control aliasing statically. It combines strong type constraints for readwrite references with the flexibility of readonly references. This combination guarantees an invariant that enables modular verification while retaining enough flexibility to handle most common implementation patterns, especially patterns such as binary methods and iterators that are not supported by related approaches.

The declarative interface specification technique presented in this book provides pre-post-specifications, abstract fields with explicit dependencies, modifies-clauses, and type invariants. Functional method behavior can be covered by pre-post-specifications. Abstract fields are used to map object structures to values of an abstract domain. The dependencies of an abstract field on the concrete fields that represent it are explicitly declared. Together with modifies-clauses, these declarations are used to express frame properties. Frame properties are particularly difficult to verify in a modular way since they require one to prove that certain abstractions are not modified by a method even if these abstractions are declared in other modules. To cope with this problem, we exploit the invariant guaranteed by the universe type system to define a novel semantics for modifies-clauses and to restrict the permissible dependencies of abstract fields in a way that makes modular verification of frame properties possible. Regarding type invariants as special

abstract fields allows us to apply the specification and verification technique for frame properties to invariants.

For verification, we use a Hoare-style programming logic that is capable of handling OO-features and modularity. In particular, it ensures that only those properties of a module can be proved that hold in all well-formed contexts in which the module might be reused. That is, the logic guarantees the modular soundness of our verification technique.

Our techniques are presented for a programming language similar to sequential Java, but can be adapted to procedural and other object-oriented languages as well.

This book is based on my dissertation which was accepted by FernUniversität Hagen, Germany, in April 2001. The underlying research was carried out at FernUniversität Hagen, Germany, Iowa State University, USA, and Technische Universität München, Germany, during the previous five years.

During that time, I was advised by Prof. Dr. Arnd Poetzsch-Heffter. His thoroughness and his focus on the essential semantics of the artifacts under consideration guided me during my work on my thesis. I'm especially grateful for his ample support, his confidence, numerous inspiring discussions, and for always taking my ideas and concerns seriously. Far beyond his guidance in professional matters, Arnd has substantially influenced my perspective on life, making the past five years such a valuable experience.

I would like to express my gratitude to Prof. Gary T. Leavens, Ph.D., for serving on my thesis committee. He raised my interest in the frame problem and pointed me to promising approaches to its solution. His valuable suggestions and several fruitful discussions had an important impact on my work. In particular, I would like to thank Gary and his wife Janet for being such great hosts during my stay at Iowa State University.

For the encouraging working atmosphere in our group as well as for continuous helpfulness, I am indebted to my team members Monika Lücke and Jörg Meyer. Jörg played an important role in my work on this book. I am thankful for the close collaboration, countless discussions, and Jörg's great sense of humor. Moreover, I would like to thank him and his wife Ilka for their friendship during the past years.

The contents of this book benefited from numerous discussions and joint work with my advisor and my fellow team members. To express my appreciation for this team work, I use the first person plural in the following, although I am the only author of this book.

Prof. Dr. Jürgen Eickel gave me the opportunity to work in his group at the Technical University of Munich before I moved to Hagen. I am especially grateful to him for making my frequent visits to Munich possible by hosting me as a guest at his chair.

This book benefited from the valuable comments of several proof readers. I highly appreciate the efforts of Marco Avitabile, Marcel Labeth, Volker

Markl, Jörg Meyer, David von Oheimb, Günther Rackl, and Christian Schiller. Furthermore, I thank those who have directly or indirectly contributed to my work.

Last, but not least, my special thanks go to my girlfriend Annette Boseck for her understanding and encouragement as well as to my sister Sabine Müller and my parents Josephine and Claus Müller for their ongoing support.

September 2001 *Peter Müller*

Contents