Hans-Gerhard Gross

Component-Based Software Testing with UML

Hans-Gerhard Gross

Component-Based Software Testing with UML

With 121 Figures and 24 Tables

Springer

Hans-Gerhard Gross
Fraunhofer Institute
for Experimental Software Engineering
Sauerwiesen 6
67661 Kaiserslautern
e-mail: hans-gerhard.gross@iese.fraunhofer.de

Library of Congress Control Number: 2004111059

ACM Computing Classification (1998): D.2.4, D.2.5, D.2.13

ISBN 3-540-20864-X Springer Berlin Heidelberg New York

Cover design: KünkelLopka, Heidelberg
Production: LE-TeX Jelonek, Schmidt & Vöckler GbR, Leipzig
Typesetting: by the author
Printed on acid-free paper 45/3142/YL - 5 4 3 2 1 0

To Marc, Heiko, Isabell,
and Oliver

Foreword

The promise of object-oriented development to enable the flexible composition and reuse of software artifacts finds an extended realization in component technologies such as CORBA components, EJB, and .Net. Although off-the-shelf components and component composition, configuration and deployment methods are nowadays available, an intrinsic problem of component-based software construction is not yet well addressed. How can software engineers ensure that a component used in a certain component assembly fits the requirements and the context? How can software engineers select a component from a component repository? How can software engineers check its correctness and performance in that component assembly? Only if the overall efforts of developing reusable components, their composition into new component assemblies and correctness checks thereof are made more efficient than developing everything from scratch will component-based software development become a new software development paradigm in industry.

Traditional testing methods where the test system is independent from the tested system do not offer good support for the evaluation of software components in new contexts when procuring off-the-shelf components or reusing in-house components. Although testing is the means to obtain objective quality metrics about components in their target component assembly, the separation of functional component and test component in traditional testing leaves the problem up to the component user to design and develop tests that assess the correctness and performance of a component in a new component assembly. However, the argument of reuse should not only apply to a functional component but also to its test component. The idea is as simple as it is compelling: why not bring test components up to the same level of reuse as the components they are supposed to evaluate?

Built-in contract-based tests as developed and put forward in this book advocate a tight coupling of components with their test components. Built-in tests provide basic test functionality to flexibly assess the behavior of a software component. In addition, they enable checks by the component it-

self of its environment, as inadequate contexts will typically make a software component fail and should be detected by the component itself.

Although built-in tests follow well-known principles of hardware development, where readily built-in validation infrastructure belongs to any high-quality system, software component testing imposes new challenges when designing and developing built-in component tests as well as when applying and using them. First of all, as components may bear complex internal data and behavior that might show nondeterministic reactions when driven via component interfaces only, components must be made observable and controllable by opening the access to component internals in order to gain unambiguous, well-focused and repeatable results. Built-in tests provide solutions here. Secondly, as components will be typically used in new, possibly unforeseen contexts, the component tests must be made flexibly adaptable and extendable so as to analyze new usage scenarios and the reactions of a component in new assemblies. While the client components can freely compose new tests by invoking the base test functionality offered via the component's testing interface, the component itself can analyze its requirements that are preconditions for its correct behavior in a given context. Thirdly, testing a component assembly is an ongoing activity when constructing component assemblies, but when putting this assembly into production no checks or only sanity checks should be kept in order to avoid unnecessary code. Component technologies such as built-in testing help here as well.

However, the solution of built-in contract-based testing would only be half of a solution if not accompanied by approaches for how to design and construct them. This book puts the ideas of built-in contract-based tests into a UML-based development process and discusses the various aspects and views of how to develop them. The powerful KobrA development method is used as a basis, both for the system and the test system development, which brings test development not only onto the same level as system development but enables also the exchange and reuse of system artifacts in test development, and vice versa. In addition, it enables the exchange of information and the support of interaction between system architects, designers and developers on the one hand, and test architects, designers and developers on the other hand.

Hans-Gerhard Gross and I were members of the MDTS project which focused on "Model-Based Development of Telecommunication Systems" in which one of the major subjects was the integrated development of a system and its test system. While the first approaches were developed in this project, this book nicely completes the ideas and discusses the various parts in great detail. The fundamental concept of built-in contract-based tests, although not new in itself, is thoroughly discussed in the context of software components. Although some questions such as the quality of tests remain open, the book advances the state of the art in component testing. Of particular interest to me is the use of the UML 2.0 Testing Profile in this book and its application and extension to the special case of built-in tests. This demonstrates not only the capabilities of UML-based test specification in a standardized way but

also the flexibility of the testing profile to address real needs from testing software components.

This book is an exhaustive compendium for component-based software testing based on UML, and it provides good examples for applying the developed approach. As a reader, I enjoyed the detailed discussion arguing about business-, technical- and process-oriented aspects, and I would like to see follow-ups.

Berlin,
July 2004

Ina Schieferdecker

Preface

This book summarizes and consolidates my primary research interests over the course of the last four to six years, and its main subjects represent the outcome of three major research projects, Component+, MDTS, and Empress, in which I was involved during my work at the Fraunhofer Institute for Experimental Software Engineering (IESE), Kaiserslautern, Germany. The three projects were carried out almost in a sequence that represents the "natural evolutionary steps" in developing the technologies introduced in this book.

The primary testing technology described here, built-in testing, with its two main incarnations, built-in contract testing and built-in quality-of-service testing, were developed within the Component+ project (www.component-plus.org) which was funded under the 5th IST European Framework Program. These technologies are described mainly in Chaps. 4, 5, and 7. The project consortium comprised a balanced number of academic research institutes and industry partners who applied the built-in testing approaches in a number of real-world case studies. I did not use these case studies as examples in this book because none of them addresses all the issues that are discussed in the book. I found it easier to explain the issues using two examples only, a thought-out vending machine specification and a real system specification from the Fraunhofer IGD's Resource Information Network. The descriptions of the industrial case studies and their results can be obtained from the Component+ project Web site.

The second project, MDTS, which stands for Model-Driven Development of Telecom Systems (www.fokus.gmd.de/mdts), funded by the German National Department of Education and Research (BMBF), devised the model-based testing and test modeling approaches that are introduced in the book. The validation of these concepts was based on the Resource Information Network (RIN), an application that was developed by Fraunhofer IGD, Darmstadt, Germany. The technologies of this project are described mainly in Chaps. 3 and 5.

The third research project, the ITEA project Empress (www.empress-itea.org), also funded by the German government, dealt with software evo-

lution for embedded real-time systems. Here, my primary focus was on the adoption of search-based software engineering technologies for dynamic timing analysis and validation of real-time software according to modern object-oriented and component-based development principles. This work represents a combination of the main subject from Component+ with the work that I carried out at the University of Glamorgan, Wales, UK, prior to my position at IESE. This more visionary subject is treated in Chap. 7.

Another major subject of the book, and I am going to start with this, is the overall unifying framework for all introduced technologies: the KobrA method (www.iese.fhg.de/Projects/Kobra_Method). This is a component-based and model-driven software development method that was devised with significant involvement by IESE in another project funded by the German government, which resulted in a book by Colin Atkinson, and others, with the title "Component-Based Product Line Engineering with UML;" its title is similar to that of this book, and this reflects the similarity of the context. The work described in this book can be seen as an extension of the KobrA method in the area of testing. All the principles introduced are fully in line with the KobrA method, and the projects that I have mentioned above were applying, at least partially, principles of the KobrA method. I give an overview on the most fundamental and important topics of the KobrA method, so that the book can be read more cohesively. A much more comprehensive treatment of the KobrA method can be found in the other book. I introduce the KobrA principles in Chap. 2 of this book.

This book is by no means an exhaustive treatment of the combination of the subjects component-based development, modeling, and testing. It only deals with a few facets of these subjects, and, as such, only reflects my personal opinions and experiences in these subjects. Each of them are fields of extensive research and development in their own right. Some readers will disagree with my opinions, some others will be disappointed with the choice of subjects that I have treated in the book. I have tried to include pointers, to my best knowledge, to other similiar, related subjects of interest throughout the text.

Acknowledgements

Books are not simply cast into form. Besides the author, there are usually many more people who contribute in terms of ideas and discussions, and in the form of technical as well as emotional support. I cannot list all who helped to sustain me over the course of writing this book, and I apologize for any possible omissions.

First of all, I would like to thank all the people in the Component+ project in which most of the results that eventually led to the production of the book were produced. These are Håkan Edler and Jonas Hörnstein (IVF, Mölndal, Sweden), Franck Barbier, Jean Michel Bruel, and Nicolas Belloir (University of Pau, France), John Kinghorn and Peter Lay (Philips, Southampton, UK), Graham King (Southampton Institute, UK), Jonathan Vincent (Bournemouth

University, UK), Stefano di Panfilis and Matteo Melideo (Engineering Informatica, Rome, Italy), Aracéli Gomez, José Maria Lázaro, and Joseba Iñaki Angulo Redondo (Labein, Bilbao, Spain), Atanas Manafov (ISOFT, Sofia, Bulgaria), Mariusz Momotko, Bartosz Nowicki, and Witold Staniszkis (RODAN, Warsaw, Poland), Enn Öunapuu (Technical University of Tallinn, Estonia), Ivan Stanev (University of Rousse, Bulgaria), Istvan Forgacs and Eva Takacs (4D Soft, Budapest, Hungary).

In addition, I would like to thank the people from the MDTS project for the outstanding cooperation over the course of nearly three years of working together: Marc Born, especially for the crash course on component platforms, Julia Reznik (Fraunhofer Fokus, Berlin), Pedro Sanchez (Fraunhofer IGD, Darmstadt), for the provision of the RIN case study including ongoing support, and, in particular, Ina Schieferdecker (Humboldt University, Berlin), for advice on model-based testing and the TTCN-3, and for writing the Foreword.

For the fruitful cooperation in the Empress project I would like to thank, in particular, the people with whom I have cooperated most intensively: Stefan van Baelen (KU Leuven, Belgium), Linde Loomans (Jabil, Hasselt, Belgium), and Ruurd Kuiper (Technical University of Eindhoven, Netherlands).

Very special thanks go to Colin Atkinson, a former colleague of mine at the Fraunhofer Institute, now affiliated with the University of Mannheim. I see him as my mentor and teacher in the area of component-based development and modeling. I would also like to thank my current colleagues at Fraunhofer IESE: Christian Bunse for his advice and Christian Peper for reviewing the UML diagrams. My former colleagues Nikolas Mayer and Javier Mauricio Paredes deserve special recognition for their commitment to implementing and assessing optimization-based timing analysis.

In addition, I would like to thank the Springer team, in particular Ralf Gerstner, for ongoing support in technical issues, and the reviewers and the copy editor who provided me with valuable feedback and helped to improve the book considerably.

Finally, I would also like to thank my friends and family, for their support and love throughout the time of writing, and Isabell Beutke, especially, for tolerating my occasional phases of social disability over a considerable period of time.

Neunkirchen, *Hans-Gerhard Gross*
July 2004

Contents