Lecture Notes in Computer Science 3466

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
University of Dortmund, Germany

Madhu Sudan
Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos
New York University, NY, USA

Doug Tygar
University of California, Berkeley, CA, USA

Moshe Y. Vardi
Rice University, Houston, TX, USA

Gerhard Weikum
Max-Planck Institute of Computer Science, Saarbruecken, Germany

Stefan Leue Tarja Johanna Systä (Eds.)

Scenarios: Models, Transformations and Tools

International Workshop
Dagstuhl Castle, Germany, September 7-12, 2003
Revised Selected Papers

Springer

Volume Editors

Stefan Leue
University of Konstanz
Department of Computer and Information Science
78457 Konstanz, Germany
E-mail: Stefan.Leue@uni-konstanz.de

Tarja Johanna Systä
Tampere University of Technology
Institute of Software Systems
33101 Tampere, Finland
E-mail: cstasy@cs.uta.fi

Library of Congress Control Number: 2005928335

CR Subject Classification (1998): F.3.1-2, C.2.4, D.2.1, D.2.4-5, D.3.1, K.6.5

ISSN 0302-9743
ISBN-10 3-540-26189-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-26189-6 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2005
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11495628 06/3142 5 4 3 2 1 0

Preface

Visual notations and languages continue to play a pivotal rôle in the design of complex software systems. In many cases visual notations are used to describe usage or interaction scenarios of software systems or their components. While representing scenarios using a visual notation is not the only possibility, a vast majority of scenario description languages is visual. Scenarios are used in telecommunications as Message Sequence Charts, in object-oriented system design as Sequence Diagrams, in reverse engineering as execution traces, and in requirements engineering as, for example, Use Case Maps or Life Sequence Charts. These techniques are used to capture requirements, to capture use cases in system documentation, to specify test cases, or to visualize runs of existing systems. They are often employed to represent concurrent systems that interact via message passing or method invocation. In telecommunications, for more than 15 years the International Telecommunication Union has standardized the Message Sequence Charts (MSCs) notation in its recommendation Z.120. More recently, with the emergence of UML as a predominant software design methodology, there has been special interest in the development of the sequence diagram notation. As a result, the most recent version, 2.0, of UML encompasses the Message Sequence Chart notation, including its hierarchical modeling features. Other scenario-flavored diagrams in UML 2.0 include activity diagrams and timing diagrams.

To a large extent the attractiveness of visual scenario notations stems from the ease with which these diagrams can be recognized and understood. On the other hand, the ease of use of these diagrams brings with it the danger that system specifications and designs understate the inherent system complexity and lead to incomplete system models. A research focus is therefore directed at making scenario notations amenable to formal treatment – this includes models for their formal representations, transformations between different notations and abstraction levels, and tools that support editing, analysis and synthesis for scenario notations.

The seminar on which this proceedings volume reports was entitled *Scenarios: Models, Transformations and Tools* and was held as Seminar Number 03371 during September 7–12, 2003, at Schloss Dagstuhl, Germany. It was organized as a continuation of a series of workshops that have been co-located with larger conferences such as the International Conference on Software Engineering (ICSE) and the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOSPLA) since 2000. This volume is a post-event proceedings volume and contains selected papers based on presentations given during the seminar. All included papers were thoroughly peer-reviewed in two rounds of reviewing.

The paper by Haugen, Husa, Runde and Stølen opens the first section of papers that deal with the semantics and analysis of scenario notations. The authors of this paper argue for the need to use a three-event semantics which distinguishes the sending event, the receiving event and the consumption event in timed sequence diagrams. An interactive scenario design process by which the system synthesizes a design model by learning from sets of positive and negative scenarios, represented as sequence diagrams, is described in the paper by Harel, Kugler and Weiss. An analysis tool stands at the end of their tool chain. When analyzing Scenario specifications it is important to recognize the limits of decidability. The paper by Muscholl and Peled reviews important decidability results regarding Sequence Diagrams and Message Sequence Charts, another popular visual scenario notation. It is frequently observed that the application of modeling formalisms in specific application domains requires dedicated semantics. Cremers and Mauw propose in their paper an operational semantics for Messages Sequence Charts applied in the domain of security protocols.

One objective of the Dagstuhl seminar was to entice practical work that assesses the suitability of different scenario design approaches to a common case study. Two half-days during the seminar were devoted to modeling the case study known as the *Autonomous Shuttle System* using different design approaches and tools. The paper by Giese and Klein describe this case study. Some of the subsequent papers in this volume refer to it.

We mentioned above that many but not all scenario formalisms are visual. In his paper, Dromey introduces a textual scenario description language called Design Behavior Trees and exemplifies this design notation by application to the Early Warning System case study proposed by Harel and Politi.

The paper by Diethelm, Geiger and Zündorf offers a thorough treatment of the Autonomous Shuttle System case study using the Story Driven Modeling design approach. The CASE tool Fujaba, which underlies this study, enables editing, analysis and synthesis based on a collection of scenarios. The Use Case Maps notation has recently evolved as a new visual requirements notation that focusses on expressing the causalities of events happening along use cases. In their paper, Petriu, Amyot, Woodside and Jiang illustrate the use of the Use Case Maps notation by applying it to capturing requirements for the Autonomous Shuttle System case study.

It has long been recognized that Message Sequence Charts and related scenario notations can prove helpful in software testing. The paper by Beyer and Dulz suggests the use of collections of scenarios in the synthesis of a stochastic usage model, called Markov Chain Usage Models. These models are later used as the basis for testing stochastic properties of real-time systems.

Both the formal analysis of variants of Message Sequence Chart models and the synthesis of correct executable code from these models are at the heart of the paper by Bontemps, Heymans and Schobbens. Since both problems are either computationally expensive or intractable, the authors propose sound and complete "lightweight" approximations of the original problems. The synthesis problem is also the subject of the paper by Giese, Klein and Burmester. The

authors suggest the derivation of behavior patterns from scenario specifications. The patterns will later be used for compositional system verification.

The modeling of mobile systems is addressed in the paper by Kosiuczenko. The author suggests a graphical scenario notation to represent object mobility as an extension of UML Sequence Diagrams and suggests a semi-formal interpretation for this notation.

Message Sequence Charts are frequently used at the early stages of the software design process, and it is desirable to derive executable design models from them. The MSC2SDL tool that Khendek and Zhang describe synthesizes SDL models from collections of MSC specifications. The authors illustrate their approach by using the Autonomous Shuttle System case study as a reference.

Object-oriented systems tend to be described by the services that the object instances can provide, and often assume that an object may provide different services as it plays different rôles. The paper by Krüger and Mathews illustrates the use of Scenario Diagrams in describing the different services that object instances may provide. They also show how a complete system view can be derived from this model. The authors exemplify the use of their notation by applying it to the Center TRACON Automation System (CTAS) case study, another benchmark case study for scenario-based system design.

The collection of papers included in this volume covers a major portion of the discussions that took place during the seminar. More information, including the program, transparencies of the presentations, and a summary of the outcome of the seminar, is available online under the URL `http://www.dagstuhl.de/03371/`

Acknowledgements. We thank Francis Bordeleau for co-organizing this seminar with us and for helping us in the initial phases of the editing of this volume. We are truly grateful to Schloss Dagstuhl and its staff for providing us with the very pleasant atmosphere that made a very productive seminar come about. The permission to use the Springer LNCS online reviewing system helped us a lot in the compilation of this volume, and we wish to thank Tiziana Margaria and Martin Karusseit for their support.

March 2005

Tarja Systä (Tampere)
Stefan Leue (Konstanz)

Organization

Seminar Organizers

F. Bordeleau
S. Leue
T. Systä

Referees

D. Amyot
Y. Bontemps
F. Bordeleau
J.P. Corriveau
H. Giese
M. Glinz
S. Graf
R. Grosu
Ø. Haugen
K. Heljanko
F. Khendek
A. Knapp
H. Kugler
S. Leue
C. Lohr
E. Mäkinen
S. Mauw
D. Peled
I. Schieferdecker
S. Somé
T. Systä
S. Uchitel
G. Weiss
M. Woodside
A. Zündorf

Table of Contents

Scenarios: Models, Transformations and Tools