

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, 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, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology Madras, Chennai, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany


More information about this series at <http://www.springer.com/series/7407>


Leen Lambers · Jens Weber (Eds.)

Graph Transformation

11th International Conference, ICGT 2018
Held as Part of STAF 2018
Toulouse, France, June 25–26, 2018
Proceedings

Editors

Leen Lambers 
University of Potsdam
Potsdam
Germany

Jens Weber 
University of Victoria
Victoria, BC
Canada

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-92990-3 ISBN 978-3-319-92991-0 (eBook)
<https://doi.org/10.1007/978-3-319-92991-0>

Library of Congress Control Number: 2018944418

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG
part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword

Software Technologies: Applications and Foundations (STAF) is a federation of leading conferences on software technologies. It provides a loose umbrella organization with a Steering Committee that ensures continuity. The STAF federated event takes place annually. The participating conferences and workshops may vary from year to year, but they all focus on foundational and practical advances in software technology. The conferences address all aspects of software technology, from object-oriented design, testing, mathematical approaches to modeling and verification, transformation, model-driven engineering, aspect-oriented techniques, and tools. STAF was created in 2013 as a follow-up to the TOOLS conference series that played a key role in the deployment of object-oriented technologies. TOOLS was created in 1988 by Jean Bézivin and Bertrand Meyer and STAF 2018 can be considered as its 30th birthday.

STAF 2018 took place in Toulouse, France, during June 25–29, 2018, and hosted: five conferences, ECMFA 2018, ICGT 2018, ICMT 2018, SEFM 2018, TAP 2018, and the Transformation Tool Contest TTC 2018; eight workshops and associated events. STAF 2018 featured seven internationally renowned keynote speakers, welcomed participants from all around the world, and had the pleasure to host a talk by the founders of the TOOLS conference Jean Bézivin and Bertrand Meyer.

The STAF 2018 Organizing Committee would like to thank (a) all participants for submitting to and attending the event, (b) the Program Committees and Steering Committees of all the individual conferences and satellite events for their hard work, (c) the keynote speakers for their thoughtful, insightful, and inspiring talks, and (d) the École Nationale Supérieure d'Électrotechnique, d'Électronique, Hydraulique et des Télécommunications (ENSEEIH), the Institut National Polytechnique de Toulouse (Toulouse INP), the Institut de Recherche en Informatique de Toulouse (IRIT), the région Occitanie, and all sponsors for their support. A special thanks goes to all the members of the Software and System Reliability Department of the IRIT laboratory and the members of the INP-Act SAIC, coping with all the foreseen and unforeseen work to prepare a memorable event.

June 2018

Marc Pantel
Jean-Michel Bruel

Preface

This volume contains the proceedings of ICGT 2018, the 11th International Conference on Graph Transformation held during June 25–26, 2018 in Toulouse, France. ICGT 2018 was affiliated with STAF (Software Technologies: Applications and Foundations), a federation of leading conferences on software technologies. ICGT 2018 took place under the auspices of the European Association of Theoretical Computer Science (EATCS), the European Association of Software Science and Technology (EASST), and the IFIP Working Group 1.3, Foundations of Systems Specification.

The aim of the ICGT series is to bring together researchers from different areas interested in all aspects of graph transformation. Graph structures are used almost everywhere when representing or modeling data and systems, not only in computer science, but also in the natural sciences and in engineering. Graph transformation and graph grammars are the fundamental modeling paradigms for describing, formalizing, and analyzing graphs that change over time when modeling, e.g., dynamic data structures, systems, or models. The conference series promotes the cross-fertilizing exchange of novel ideas, new results, and experiences in this context among researchers and students from different communities.

ICGT 2018 continued the series of conferences previously held in Barcelona (Spain) in 2002, Rome (Italy) in 2004, Natal (Brazil) in 2006, Leicester (UK) in 2008, Enschede (The Netherlands) in 2010, Bremen (Germany) in 2012, York (UK) in 2014, L'Aquila (Italy) in 2015, Vienna (Austria) in 2016, and Marburg (Germany) in 2017, following a series of six International Workshops on Graph Grammars and Their Application to Computer Science from 1978 to 1998 in Europe and in the USA.

This year, the conference solicited research papers describing new unpublished contributions in the theory and applications of graph transformation as well as tool presentation papers that demonstrate main new features and functionalities of graph-based tools. All papers were reviewed thoroughly by at least three Program Committee members and additional reviewers. We received 16 submissions, and the Program Committee selected nine research papers and two tool presentation papers for publication in these proceedings, after careful reviewing and extensive discussions. The topics of the accepted papers range over a wide spectrum, including advanced concepts and tooling for graph language definition, new graph transformation formalisms fitting various application fields, theory on conflicts and parallel independence for different graph transformation formalisms, as well as practical approaches to graph transformation and verification. In addition to these paper presentations, the conference program included an invited talk, given by Olivier Rey (GraphApps, France).

We would like to thank all who contributed to the success of ICGT 2018, the invited speaker Olivier Rey, the authors of all submitted papers, as well as the members of the Program Committee and the additional reviewers for their valuable contributions to the selection process. We are grateful to Reiko Heckel, the chair of the Steering Committee of ICGT for his valuable suggestions; to Marc Pantel and Jean-Michel Bruel,

the organization co-chairs of STAF; and to the STAF federation of conferences for hosting ICGT 2018. We would also like to thank EasyChair for providing support for the review process.

June 2018

Leen Lambers
Jens Weber

Organization

Steering Committee

Paolo Bottoni	Sapienza University of Rome, Italy
Andrea Corradini	University of Pisa, Italy
Gregor Engels	University of Paderborn, Germany
Holger Giese	Hasso Plattner Institute at the University of Potsdam, Germany
Reiko Heckel (Chair)	University of Leicester, UK
Dirk Janssens	University of Antwerp, Belgium
Barbara König	University of Duisburg-Essen, Germany
Hans-Jörg Kreowski	University of Bremen, Germany
Ugo Montanari	University of Pisa, Italy
Mohamed Mosbah	LaBRI, University of Bordeaux, France
Manfred Nagl	RWTH Aachen, Germany
Fernando Orejas	Technical University of Catalonia, Spain
Francesco Parisi-Presicce	Sapienza University of Rome, Italy
John Pfaltz	University of Virginia, Charlottesville, USA
Detlef Plump	University of York, UK
Arend Rensink	University of Twente, The Netherlands
Leila Ribeiro	University Federal do Rio Grande do Sul, Brazil
Grzegorz Rozenberg	University of Leiden, The Netherlands
Andy Schürr	Technical University of Darmstadt, Germany
Gabriele Taentzer	University of Marburg, Germany
Bernhard Westfechtel	University of Bayreuth, Germany

Program Committee

Anthony Anjorin	University of Paderborn, Germany
Paolo Baldan	University of Padua, Italy
Gábor Bergmann	Budapest University of Technology and Economics, Hungary
Paolo Bottoni	Sapienza University of Rome, Italy
Andrea Corradini	University of Pisa, Italy
Juan De Lara	Autonomous University of Madrid, Spain
Juergen Dingel	Queen's University, Canada
Rachid Echahed	CNRS and University of Grenoble, France
Holger Giese	Hasso Plattner Institute at the University of Potsdam, Germany
Annegret Habel	University of Oldenburg, Germany
Reiko Heckel	University of Leicester, UK

Berthold Hoffmann	University of Bremen, Germany
Dirk Janssens	University of Antwerp, Belgium
Barbara König	University of Duisburg-Essen, Germany
Leen Lambers (Co-chair)	Hasso Plattner Institute at the University of Potsdam, Germany
Yngve Lamo	Bergen University College, Norway
Mark Minas	Bundeswehr University Munich, Germany
Mohamed Mosbah	LaBRI, University of Bordeaux, France
Fernando Orejas	Technical University of Catalonia, Spain
Francesco Parisi-Presicce	Sapienza University of Rome, Italy
Detlef Plump	University of York, UK
Arend Rensink	University of Twente, The Netherlands
Leila Ribeiro	University Federal do Rio Grande do Sul, Brazil
Andy Schürr	Technical University of Darmstadt, Germany
Gabriele Taentzer	Philipps University of Marburg, Germany
Jens Weber (Co-chair)	University of Victoria, Canada
Bernhard Westfechtel	University of Bayreuth, Germany
Albert Zündorf	University of Kassel, Germany

Additional Reviewers

Atkinson, Timothy	Nolte, Dennis
Azzi, Guilherme	Peuser, Christoph
Dyck, Johannes	Sakizloglou, Lucas
Farkas, Rebeka	Semeráth, Oszkár
Kluge, Roland	

Introduction to Graph-Oriented Programming (Keynote)

Olivier Rey 

GraphApps, France
rey.olivier@gmail.com
orey.github.io/papers

Abstract. Graph-oriented programming is a new programming paradigm that defines a graph-oriented way to build enterprise software, using directed attributed graph databases as backend. Graph-oriented programming is inspired by object-oriented programming, functional programming, design by contract, rule-based programming and the semantic web. It integrates all those programming paradigms consistently. Graph-oriented programming enables software developers to build enterprise software that does not generate technical debt. Its use is particularly adapted to enterprise software managing very complex data structures, evolving regulations and/or high numbers of business rules.

Couplings in Enterprise Software

The way the software industry currently builds enterprise software generates a lot of “structural and temporal couplings”. Structural coupling occurs when software and, in particular, data structures, are implemented such that artificial dependencies are generated. A dependency is artificial if it occurs in the implementation but not in the underlying semantic concepts. Temporal couplings are artificial dependencies generated by holding several versions of business rules in the same program, those rules being applicable to data that are stored in the last version of the data structures.

Those couplings are at the very core of what is commonly called “technical debt”. This debt generates over-costs each time a software evolves. Generally, the requirements change, the software is partially redesigned to accommodate the modification, the data structures evolve, the existing data must be migrated, and all programs must be non-regressed. In order to implement a small modification in an enterprise software, a change in regulation for instance, overcoming the technical debt may represent up to 90–95% of the total workload [5, 6].

The software industry has, for a long time, identified the costs associated to technical debts, and in particular those costs seem to grow exponentially with time [5]. That means that the productivity of any maintenance team of fixed size will constantly decrease throughout the evolution process. In order to address this core issue of enterprise software, a lot of engineering-oriented work-arounds can be found: design patterns that are supposed to enhance software extensibility [1], software architecture practices that define modules and layers inside an enterprise software [2, 4], or best

practices for software refactoring to reduce the costs of the refactoring phase itself [3]. However, every software vendor knows that the core problem of the technical debt has not been solved.

Graph-Oriented Programming

Graph-oriented programming is meant as an alternative programming paradigm not collecting technical debts. This paradigm is based on three concepts: (1) Using directed attributed graph databases to store the business entities without storing their relationships in the entities themselves, i.e. there are no foreign keys; (2) Designing programs so that the knowledge about relationships between entities (business nodes) is captured in functional code located “outside” of the nodes, encapsulated in graph transformation rules; (3) Using best practices in graph transformation design to guarantee a minimal or even no generation of technical debt. This programming paradigm can be applied using an object-oriented or functional programming language.

The expected advantages of using graph-oriented programming are multiple: reusability of software is increased due to less software dependencies; multiple views of the same data can be implemented in the same application; multiple versions of data structures and business rules can cohabit, meaning that the software and the data can be timelined; software maintenance can be done by adding new software rather than by modifying existing software.

At last, graph-oriented programming enables to build a different kind of enterprise software that proposes, through the use of a graph-oriented navigation, a new user experience, closer to our mental way of representing things.

The Approach Taken at GraphApps

At GraphApps, we developed a graph-oriented designer in Eclipse whose purpose is to model node and relationship types, as they occur in business applications, and to group them in semantic domains. Code generators, coupled to the designer, generate parameterized web pages proposing a default view of defined types of business entities. For each semantic domain, an independent `jar` file is generated. In addition, we developed a graph-oriented web framework, which loads the `jar` files and enables us to integrate them in the graphical web framework. All domains can be integrated without introducing any new code dependency. Each domain may include custom code, in order to implement graph transformations, web page modifications, or new pages. Moreover, the framework proposes reusable components that offer generic reusable mechanisms such as business node classification (every business node can be referenced in a tree of shared folders), business node timelines, navigation history, personalized links between business nodes, or alternate navigation.

Those tools support a quick prototyping of large and complex applications, the implementation of time-based business rules, and the cooperative work of several teams collaborating to the same core model. The way the code is organized enables us to modify the behavior of the core system, without having to modify existing code,

migrating data, or performing non-regressing testing. We have used this set of tools for many business prototypes and we are using it currently to build a complete innovative aerospace maintenance information system (composed by many semantic domains) from scratch.

Conclusion

The paradigm of graph-oriented programming enables us to build a new generation of enterprise software that will be much easier to maintain and that can address the high complexity of business entity structures and their life cycles, as well as time-sensitive business rules. This paradigm may be used to rewrite a huge number of enterprise software in the coming decades in order to decrease drastically the maintenance costs, to enhance the capability of personalization of the software and to create new user experiences by proposing more intuitive ways to navigate within the software.

References

1. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns, Elements of Reusable Object Oriented Software. Addison-Wesley (1994)
2. Buschmann, F.: *POSA* Volume 1 - A System of Patterns. Wiley (1996)
3. Fowler, M.: Refactoring. Addison-Wesley (1999)
4. Alur, D.: Core J2EE Patterns, 2nd edn. Prentice-Hall (2003)
5. Nugroho, A., Joost, V., Tobias, K.: An empirical model of technical debt and interest. In: Proceedings of the 2nd Workshop on Managing Technical Debt. ACM (2011)
6. Li, Z., Avgeriou, P., Liang, P.: A systematic mapping study on technical debt and its management. *J. Syst. Softw.* **101**, 193–220 (2015)

Contents

Graph Languages

Splicing/Fusion Grammars and Their Relation to Hypergraph Grammars	3
<i>Hans-Jörg Kreowski, Sabine Kuske, and Aaron Lye</i>	
Synchronous Hyperedge Replacement Graph Grammars.	20
<i>Corey Pennycuff, Satyaki Sikdar, Catalina Vajiac, David Chiang, and Tim Weninger</i>	
CoReS: A Tool for Computing Core Graphs via SAT/SMT Solvers	37
<i>Barbara König, Maxime Nederkorn, and Dennis Nolte</i>	

Graph Transformation Formalisms

Graph Surfing by Reaction Systems	45
<i>Hans-Jörg Kreowski and Grzegorz Rozenberg</i>	
Probabilistic Graph Programs for Randomised and Evolutionary Algorithms	63
<i>Timothy Atkinson, Detlef Plump, and Susan Stepney</i>	
Graph-Rewriting Petri Nets	79
<i>Géza Kulcsár, Malte Lochau, and Andy Schürr</i>	

Parallel Independence and Conflicts

On the Essence and Initiality of Conflicts.	99
<i>Guilherme Grochau Azzì, Andrea Corradini, and Leila Ribeiro</i>	
Characterisation of Parallel Independence in AGREE-Rewriting	118
<i>Michael Löwe</i>	
Equivalence and Independence in Controlled Graph-Rewriting Processes	134
<i>Géza Kulcsár, Andrea Corradini, and Malte Lochau</i>	

Graph Conditions and Verification

Verifying Graph Transformation Systems with Description Logics	155
<i>Jon Haël Brenas, Rachid Echahed, and Martin Strecker</i>	

OCL2AC: Automatic Translation of OCL Constraints to Graph Constraints
and Application Conditions for Transformation Rules 171
Nebras Nassar, Jens Kosiol, Thorsten Arendt, and Gabriele Taentzer

Author Index 179