Lecture Notes in Computer Science 7747

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

Sven Jörges

Construction and Evolution of Code Generators

A Model-Driven and Service-Oriented Approach

Springer

Author

Sven Jörges
Technische Universität Dortmund
Department of Computer Science
Otto-Hahn-Straße 14
44227 Dortmund, Germany
E-mail: sven.joerges@tu-dortmund.de

This monograph constitutes a revised version of the author's doctoral dissertation, which was submitted to TU Dortmund, Department of Computer Science, Chair of Programming Systems, Otto-Hahn-Straße 14, 44227 Dortmund, Germany, under the original title "Genesys: A Model-Driven and Service-Oriented Approach to the Construction and Evolution of Code Generators," and which was accepted in December 2011.

Lots of figures in this book use icons from the excellent Tango Desktop Project (http://tango.freedesktop.org/).

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-36126-5 e-ISBN 978-3-642-36127-2
DOI 10.1007/978-3-642-36127-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012955656

CR Subject Classification (1998): D.3.4, D.2.11, D.2.1-4, D.2.13, D.3.2-3, J.1

LNCS Sublibrary: SL 2 – Programming and Software Engineering

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

Domain-specific languages are key for canalizing the steadily growing demand for tailored applications. They help strengthen the role of application experts during the development process, sometimes to the extent that they can adapt or even develop special applications themselves. At the same time, tailored user-oriented hardware allows software applications to slowly conquer everyone's lives. Think of the iPhone, for example, whose immense portfolio of applications (train schedules including delay information, route planning, cinema programs, calendar, compass, camera, Internet browsing, and thousands more) makes clear that the phone itself is no longer the dominant part. Code generators are required to bridge the gap between these two independently growing dimensions. Although code generation is a well-developed discipline, the support to efficiently keep these two dimensions in synchrony is small. In essence, even though there are code generator workbenches that help the programming expert to design dedicated code generators, more high-level activities like process support, variability management, or product lining are not foreseen.

Sven Jörges's thesis is unique in addressing this important niche:

- It develops a code generation framework (Genesys) on the basis of the jABC framework with its underlying eXtreme Model-Driven Development (XMDD) style, which combines ideas from extreme programming, model-driven design, and service orientation. This allows one to manage product lines at the model level, and to control their features via constraint solving, e.g., via model checking.
- It illustrates its impact not only by sketching numerous case studies, but also by showing how arbitrary domain-specific languages can be captured as long as they have an Ecore-based meta-model. This part of the thesis is particularly elegant, as it exploits the reflexivity of Ecore for radical bootstrapping of code generator functionality, and the ease of Genesys to be integrated, e.g., into the Eclipse framework.

- It shows how to overcome typical problems of round-trip engineering by employing full code generation. In the corresponding case study this is achieved by integrating AndroMDA, a code generation framework aiming at the translation of static aspects of a programming language.

Each of these three parts has been realized with great care and a mature look at adequate design, practicality, and quality assurance. Particularly outstanding, however, is the way all this is composed: things fit together nicely, and there is a very careful discussion of related work that puts the contributions of the thesis into perspective. It addresses topics as diverse as model-driven architectures, meta-modeling, domain-specific languages, computer-aided software engineering, round-trip engineering, code generation, generative programming, extreme programming, aspect orientation, product line management, and quality assurance. This way the reader obtains a competent goal-oriented entry into all of these areas, which is not just conceptual, but profits from experience Sven Jörges gained while realizing Genesys as a robust and flexible framework. Indeed, Genesys has been used and extended by numerous students, and it has been applied in various industrial projects.

Characteristic of the thesis is the sovereign handling of models at different (meta) levels for varying purposes. Models are used for modeling applications, but, continuously following the underlying XMDD approach, also the code generators, temporal formulae, and the test cases are modeled graphically, all in a similar fashion. In addition there is the hierarchy of meta-models specified in Ecore. This elegant, homogeneous structuring is an entry hurdle for "newcomers" to orient themselves, but Sven Jörges takes great care to clarify the context in the various scenarios. On the other hand, the continuous use of models opens the door for bootstrapping, as impressively demonstrated in various settings: e.g., a just-modeled code generator can be applied to itself, now seen as the application to be translated, in order to obtain a code generator in native code. This technique, which is later also applied to generate the required functionality for the code generation for Ecore, simply exploits the homogeneous notion of models and the fact that in jABC, models are executable via interpretation.

Sven Jörges proposes a change from viewing the construction of code generators merely as a technical low-level activity to a truly building-block-oriented construction paradigm exploiting model-driven and service-oriented approaches at higher abstraction levels. This paves the way for "mass construction/customization" of code generators in a growing landscape of domain-specific languages running on increasingly diverse hardware. I am convinced that the proposed model-based construction, validation, maintenance, evolution, and migration significantly reduces cost, increases reliability, and helps master the management of immensely growing sets of code generators.

Bernhard Steffen

Acknowledgments

I am grateful to many people for supporting, inspiring, and encouraging me during the last six years.

First, I would like to express gratitude to my PhD supervisor Prof. Dr. Bernhard Steffen for enabling me to write this book as a member of his research group. His advice and enthusiastic support have always been a great source of inspiration and motivation for me. In particular, I am very thankful that he had confidence in me when he allowed me to gather valuable experience in numerous interesting projects.

Furthermore, thanks are due to Prof. Dr. Jens Knoop and Prof. Dr. Rainer Hähnle for reviewing the monograph. I am also much obliged to the other members of my committee, Prof. Dr. Jakob Rehof and Dr. Oliver Rüthing.

Special thanks go to my colleagues Markus Doedt, Maik Merten, and Johannes Neubauer, who sacrificed a significant amount of their time to proofread the text. I am happy about each and every mistake you spotted! Likewise, "Diolch!" to Julia Rehder for lending me her English vocabulary and grammar intuition where mine reached their limits.

I would like to thank Christian Wagner for the constant exchange of ideas, literature findings, for the confusing discussions that heavily featured the "meta" prefix, and for introducing me to the delightful world of Bodum-brewed coffee. Furthermore, I am grateful to Prof. Tiziana Margaria for allowing me to benefit from her manifold contacts and for showing me how to be "context-sensitive," Dunja Rauh for supporting me with her advice and her inexhaustible collection of wise sayings, Ralf Nagel for being my interceder and for keeping me close to pragmatism, Christian Kubczak for the humor and the music, as well as Marco Bakera and Clemens Renner for patiently answering my questions on model checking. Thanks to all the colleagues from the LS5 for the great working atmosphere – I will surely miss the bad Friday puns.

Moreover, I would like to thank all the students who contributed to the Genesys project, in particular my former student assistants Benjamin Bentmann and Akif Köse, and the authors of all related diploma, bachelor, and master's theses.

Last but not least, I am deeply grateful to my family for their unlimited support that always keeps me going.

August 2011 Sven Jörges

Abstract

Automatic code generation is an essential cornerstone of model-driven approaches to software development. It closes the gap that emerges when models are used to abstract from a concrete software system, and thus is to models what compilers are to high-level languages. Consequently, the simple and fast development of code generators is a key requirement of today's approaches to model-driven development, which are increasingly characterized by a strong focus on agility and domain-specificity. Currently, many techniques are available that support the specification and implementation of code generators, such as engines based on templates or rule-based transformations. All these techniques have in common that code generators are either directly programmed or described by means of textual specifications.

This monograph presents *Genesys*, a general approach, which advocates the *graphical* development of code generators for arbitrary source and target languages, on the basis of *models* and *services*. In particular, it is designed to support incremental language development on arbitrary metalevels. The use of models allows building of code generators in a truly platform-independent and domain-specific way. Furthermore, models are amenable to formal verification methods such as model checking, which increase the reliability and robustness of the code generators. Services enable the reuse and integration of existing code generation frameworks and tools regardless of their complexity, and at the same time manifest as easy-to-use building blocks that facilitate agile development through quick interchangeability. Both models and services are reusable and thus form a growing repository for the fast creation and evolution of code generators.

This book shows these and further advantages arising from the Genesys approach by means of a full-fledged reference implementation, which has been field-tested in a large number of case studies.

List of Abbreviations

ABS	Abstract Behavioral Specification
AO	aspect orientation
ASL	Action Specification Language
ASSL	Autonomic System Specification Language
API	Application Programming Interface
ASP.NET	Active Server Pages .NET
ATL	Atlas Transformation Language
BiBiServ	Bielefeld University Bioinformatics Server
BNF	Backus-Naur Form
BPEL	Business Process Execution Language
BPM	Business Process Modeling
BPMN	Business Model & Notation
CASE	Computer-Aided Software Engineering
CDR	Common Data Representation
CIM	Computation-Independent Model
CLDC	Connected Limited Device Configuration
CORBA	Common Object Request Broker Architecture
CTL	Computation Tree Logic
CWM	Common Warehouse Metamodel
DBC	design by contract
DDBJ	DNA Data Bank of Japan
DOM	Document Object Model
DSL	domain-specific language
DSM	Domain-Specific Modeling
EBI	European Bioinformatics Institute
EJB	Enterprise JavaBean
EL	expression language
EMBL	European Molecular Biology Laboratory
EMF	Eclipse Modeling Framework
EMP	Eclipse Modeling Project
EMOF	Essential Meta-Object Facility

ENF	Engeler Normal Form
ERP	enterprise resource planning
FBB	Formula Building Block
GME	Generic Modeling Environment
GMF	Graphical Modeling Framework
GP	Generative Programming
GPS	Global Positioning System
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HUTN	Human-Usable Textual Notation
IDE	Integrated Development Environment
ITE	Integrated Test Environment
jABC EE	jABC Execution Engine
JAXB	Java Architecture for XML Binding
JCE	Java Class Extruder
JCG1	Java Class Generator 1
JCG2-SC	Java Class Generator 2, structured code variant
JCG2-LI	Java Class Generator 2, lightweight interpreter variant, reflective service calls
JCG2-LI-GS	Java Class Generator 2, lightweight interpreter variant, generated service calls
JDK	Java Development Kit
JEE	Java Enterprise Edition
jETI	Java Electronic Tool Integration
JME	Java Micro Edition
JMI	Java Metadata Interface
JML	Java Modeling Language
JSE	Java Standard Edition
JSP	JavaServer Pages
JSTL	JavaServer Pages Standard Tag Library
JVM	Java Virtual Machine
KTS	Kripke Transition System
MIL	model-in-the-loop
MPS	Meta Programming System
M2M	Model 2 Model
MD*	Umbrella term for any model-driven approaches to software development
MDA	Model Driven Architecture
MDD	Model-Driven Development
MDE	Model-Driven Engineering
MDSD	Model-Driven Software Development
MDTD	Model-Driven Test Development
MDT	Model Development Tools
MOF	Meta-Object Facility
MOFM2T	MOF Model to Text Transformation Language

MSA	Multiple Sequence Alignment
NXC	Not eXactly C
OASIS	Organization for the Advancement of Structured Information Standards
OCS	Online Conference Service
oCIOBS	Dortmunder Online Bibliographieservice
OMG	Object Management Group
OTA	One-Thing-Approach
PIL	processor-in-the-loop
PIM	Platform-Independent Model
POJO	Plain Old Java Object
POM	Project Object Model
PSM	Platform-Specific Model
QVT	Query/View/Transformation
RCX	Robotic Command Explorer
RMI	Remote Method Invocation
RTE	round-trip engineering
SIB	Service Independent Building Block
SIL	software-in-the-loop
SLG	Service Logic Graph
SUT	system under test
SWT	Standard Widget Toolkit
UID	unique identifier
UML	Unified Modeling Language
URL	Uniform Resource Locator
V&V	Verification & Validation
VTL	Velocity Template Language
WSDL	Web Services Description Language
XMDD	Extreme Model-Driven Development
XMI	XML Metadata Interchange
XML	Extensible Markup Language

List of Figures

List of Tables

Contents