MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Dipl.-Inform. Holger Krahn
aus Braunschweig

Berichter: Universitätsprofessor Dr. rer. nat. Bernhard Rumpe
Universitätsprofessor Dr. rer. nat. Andy Schürr

Tag der mündlichen Prüfung: 18.12.2009

Aachener Informatik-Berichte, Software Engineering

herausgegeben von
Prof. Dr. rer. nat. Bernhard Rumpe
Software Engineering
RWTH Aachen University

Band 1

Holger Krahn

MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering

Shaker Verlag
Aachen 2010

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Zugl.: D 82 (Diss. RWTH Aachen University, 2009)

Printed in Germany.

ISBN 978-3-8322-8948-5
ISSN 1869-9170

Shaker Verlag GmbH • Postfach 101818 • 52018 Aachen
Telefon: 02407 / 95 96 - 0 • Telefax: 02407 / 95 96 - 9
Internet: www.shaker.de • E-Mail: info@shaker.de

Kurzfassung

Domänenspezifische Sprachen (engl. domain specific language - DSL) sind Sprachen der Informatik, mit denen kompakte Problemlösungen aus eng umrissenen fachlichen oder technischen Anwendungsgebieten formuliert werden können. Durch die Nutzung einer fachspezifischen Notation gelingt die Integration von Experten einfacher als bei einer herkömmlichen Softwareentwicklung, weil die Modelle von ihnen besser verstanden werden. Die automatische Erzeugung von Produktivcode aus domänenspezifischen Modellen ist eine effektive Form der modellgetriebenen Entwicklung.

Die derzeitige DSL-Entwicklung erschwert aufgrund der fehlenden zentralen Sprachreferenz, die die abstrakte und konkrete Syntax umfasst, und der unzureichenden Modularisierung eine agile und effiziente Vorgehensweise. Es mangelt an Methoden und Referenzarchitekturen, um komplexe modellbasierte Werkzeuge strukturiert entwerfen und in der Softwareentwicklung einsetzen zu können.

In dieser Arbeit wird daher die DSL-Entwicklung mit dem MontiCore-Framework beschrieben, das die modulare Entwicklung von textuellen DSLs und darauf basierten Werkzeugen erlaubt. Die wichtigsten wissenschaftlichen Beiträge lassen sich wie folgt zusammenfassen:

- Die Definition von textuellen domänenspezifischen Sprachen wird durch ein kompaktes grammatikbasiertes Format ermöglicht, das sowohl die abstrakte als auch die konkrete Syntax einer Sprache definiert und so als zentrale Dokumentation für die Entwickler und Nutzer einer DSL fungiert. Die entstehende abstrakte Syntax entspricht etablierten Metamodellierungs-Technologien und erweitert übliche grammatikbasierte Ansätze.

- Die Wiederverwendung von Teilsprachen innerhalb der modellgetriebenen Entwicklung wird durch zwei Modularitätsmechanismen vereinfacht, da so existierende Artefakte auf eine strukturierte Art und Weise in neuen Sprachdefinitionen eingesetzt werden können. Grammatikvererbung erlaubt die Spezialisierung und Erweiterung von Sprachen. Einbettung ermöglicht die flexible Kombination mehrerer Sprachen, die sich auch in ihrer lexikalischen Struktur grundlegend unterscheiden können.

- Das MontiCore-Grammatikformat ist modular erweiterbar, so dass zusätzliche Informationen als so genannte Konzepte spezifiziert werden können und darauf aufbauend weitere Infrastruktur aus der Sprachdefinition generiert werden kann. Die Erweiterungsfähigkeit wird durch Konzepte zur deklarativen Spezifikation von Links in der abstrakten Syntax und durch ein Attributgrammatiksystem demonstriert.

- Die Entwicklung von Codegeneratoren und Analysewerkzeugen für DSLs wird durch die Bereitstellung einer Referenzarchitektur entscheidend vereinfacht, so dass bewährte Lösungen ohne zusätzlichen Entwicklungsaufwand genutzt werden können. Die Qualität der entstehenden Werkzeuge wird damit im Vergleich zu existierenden Ansätzen gehoben.

- Es wird demonstriert, wie compilierbare Templates ein integriertes Refactoring der Templates und einer Laufzeitumgebung ermöglichen. Darauf aufbauend wird eine Methodik definiert, die aus exemplarischem Quellcode schrittweise einen Generator entwickelt, dessen Datenmodell automatisiert abgeleitet werden kann.

Die beschriebenen Sprachen und Methoden sind innerhalb des Frameworks MontiCore implementiert. Ihre Anwendbarkeit wird durch die Entwicklung des MontiCore-Frameworks im Bootstrapping-Verfahren und durch zwei weitere Fallstudien demonstriert.

Abstract

Domain specific languages (DSLs) are languages in computer science with permit specifying compact solutions in clear-cut functional or technical application areas. Using a domain specific notation simplifies the integration of experts in comparison to conventional software development because the models are easier understood by them. The automatic creation of production code from domain specific models is an effective form of model-driven development.

An agile and efficient development process is hard to establish using existing DSL development methods because of the missing central language reference which includes abstract and concrete syntax and inadequate modularization techniques. Methods and reference architectures are lacking for designing and using complex and model-based tools in structured way for software development.

Thus, in this thesis the modular development of textual DSLs and tools with the MontiCore-framework is described. The most important contributions to research can be summarized as follows:

- Textual domain specific languages can be defined by a compact grammar-based format that defines abstract syntax as well as concrete syntax of a language and can therefore be used as a central documentation for developers and users of a DSL. The emerging abstract syntax is equivalent to well-established metamodeling-techniques and extends common grammar-based approaches.

- The reuse of language fragments within model-driven development is supported by two modularity mechanisms that combine existing artifacts in a structured way to form new languages: Grammar inheritance supports the specialization and extension of languages. Embedding permits the flexible combination of multiple languages, which can also differ fundamentally in their lexical structure.

- The used grammar format is extensible in a modular way such that additional information can be specified as so-called concepts. Based on them, further infrastructure can be generated from the language definition. The extensibility by concepts is demonstrated by a declarative way to specify links in the abstract syntax and by an attribute grammar system.

- The development of code generators and tools for the analysis of DSLs is simplified considerably by making a reference architecture available. Approved solutions can be used without further development effort. Thus, the quality of the emerging tools is increased in comparison to existing approaches.

- It is demonstrated how compilable templates can be used for an integrated refactoring of templates and a runtime environment. Based on that, a method is defined to develop a generator in a stepwise manner from existing source code. The data model of the generator can be derived automatically.

The abovementioned languages and methods are developed within the framework MontiCore. Their applicability is demonstrated by the development of the framework itself in a bootstrapping-process and by two further case studies.

Danksagung

An dieser Stelle möchte ich mich bei Prof. Dr. Bernhard Rumpe für die Betreuung dieser Dissertation bedanken. Durch zahlreiche konstruktive Diskussionen und wertvolle Ratschläge hat er entscheidend zum Gelingen der Promotion beigetragen. Insbesondere möchte ich ihm für die Möglichkeit, neben der akademischen Arbeit auch immer wieder Einblicke in die Praxis erlangen zu können, und für ein abwechselungsreiches und spannendes Arbeitsumfeld danken.

Weiterer Dank gebührt Prof. Dr. Andy Schürr für die Bereitschaft, die Dissertation ebenfalls zu betreuen, durch konstruktive Anmerkungen mich zum Nachdenken anzuregen und neue Perspektiven aufzuzeigen.

Herrn Prof. Dr.-Ing. Stefan Kowalewski möchte ich für die Leitung der Promotionskommision danken und Herrn Prof. Dr. Wolfgang Thomas für die Bereitschaft, mich zu prüfen.

Die Entwicklung des MontiCore-Frameworks ist keine Einzelleistung, sondern in Zusammenarbeit mit Kollegen und Studierenden am Lehrstuhl entstanden. Für die konstante Diskussion und die Rückmeldungen, für den nötigen sportlichen Ausgleich, für die hoffentlich weiterhin erfolgreiche und spannende Arbeit am Quellcode und an Veröffentlichungen und das gemeinsame Feiern möchte ich mich besonders bei den folgenden Personen bedanken: Christian Berger, Marita Breuer, Michael Dukaczewski, Christoph Ficek, Diego Gaviola, Tim Gülke, Arne Haber, Christoph Herrmann, Conrad Hoffmann, Christian Jordan, Björn Mahler, Ingo Maier, Fabian May, Miguel Paulos Nunes, Claas Pinkernell, Dirk Reiß, Jan-Oliver Ringert, Henning Sahlbach, Martin Schindler, Mark Stein, Jens Stephani, Jens Theeß, Christoph Torens, Galina Volkova, Ingo Weisemöller und Gerald Winter.

Besonders hervorzuheben ist Steven Völkel für das Führen von fachlichen Diskussionen und die Zusammenarbeit am Quellcode und an Papieren über MontiCore, die stetige Motivation und seine pragmatische Herangehensweise, die mir oft eine Hilfe bei der Lösung von Problemen war, sowie seine kritische Auseinandersetzung mit dieser Ausarbeitung.

Ebenfalls besonders hilfreich war die Unterstützung von Hans Grönniger. Vielen Dank für die Zusammenarbeit an den gemeinsamen Projekten und Papieren, die intensive Beschäftigung mit der vorliegenden Arbeit und die interessanten Einblicke in die formalen Seiten des Software Engineerings.

Zusätzlich möchte ich Holger Rendel für Kommentare zu Teilen dieser Arbeit danken und für seinen Enthusiasmus bei der Entwicklung von MontiCore.

Danken möchte ich auch Arne, Cecile, Daniela, Edgar, Gerrit, Jens, Jörg, Karolina, Marcus, Melanie, Jens, Ralf, Sabine und Sabrina für die spannenden und entspannenden Momente neben der Arbeit und die Abwechslung, die einem immer wieder neue Perspektiven aufzeigt.

Meinen Eltern möchte ich für die Unterstützung auf meinem Lebensweg und die Hilfe in schwierigen Phasen danken. Meinem Bruder Oliver gebührt Dank für die aufmunternden Worte und Taten sowie sein stetes Vorbild. Ihm wünsche ich alles Gute, damit bald die nächste Doktorfeier folgen kann.

Der größte Dank geht an meine Frau Angelika und meine Tochter Johanna. Angelika möchte ich für ihren aufopfernden Einsatz für unsere Familie danken, der mir den notwendigen Freiraum für diese Arbeit verschafft hat. Sie war stets eine konstruktive und kritische Leserin der Arbeit und fleißige Lektorin, abwechselnd eine fordernde Antreiberin und eine verständnisvolle Zuhörerin. Sie hat so einen bedeutenden Teil zur Fertigstellung dieser Dissertation beigetragen. Johanna danke ich für ihre freundliche Art, die mich mit einem einzigen Lächeln auch aus dem tiefsten Tal befreien kann.

Inhaltsverzeichnis