CARL VON OSSIETZKY universität OLDENBURG

Dilshodbek Kuryazov

Model Difference Representation

Computer Science in Oldenburg: Selected Works

Editors: Prof. Dr.-Ing. Oliver Theel
Prof. Dr. Andreas Winter

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Model Difference Representation

Von der Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften der Carl von Ossietzky Universität Oldenburg zur Erlangung des Grades und Titels eines

Doktors der Ingenieurswissenschaften (Dr.-Ing.)

angenommene Dissertation

von Herrn Dilshodbek Kuryazov

geboren am 24.09.1985 in Usbekistan

Oldenburg, February 2019

Gutachter:

Prof. Dr. Andreas Winter

Prof. Dr. Ralf Reussner

Prof. Dr. Khakimjon Zayniddinov

Tag der Disputation:

11.02.2019

Computer Science in Oldenburg: Selected Works

Band 1

Dilshodbek Kuryazov

Model Difference Representation

Shaker Verlag
Düren 2019

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Zugl.: Oldenburg, Univ., Diss., 2019

Printed in Germany.

ISBN 978-3-8440-6883-2
ISSN 2629-8910

Shaker Verlag GmbH • Am Langen Graben 15a • 52353 Düren
Phone: 0049/2421/99011-0 • Telefax: 0049/2421/99011-9
Internet: www.shaker.de • e-mail: info@shaker.de

Abstract.

As a software engineering paradigm, Model-Driven Software Engineering (MDSE) is the modern day style of software development which supports well-suited abstraction concepts to software development activities. It intends to improve the productivity of the software development, maintenance activities, and communication among various team members and stakeholders. In MDSE, software models which also comprise source code are the central artifacts. MDSE brings several main advantages such as a productivity boost, models become a single point of truth, and they are reusable and automatically kept up-to-date with the code they represent.

Software models (e.g., in UML) are the key artifacts in MDSE activities. Software models are well-suited for designing, developing and producing large-scale software projects. In order to cope with constantly growing amounts of software artifacts and their complexity, software systems to be developed and maintained are initially shifted to abstract forms using modeling concepts. Software models are the documentation and implementation of software systems being developed and evolved.

Like the source code of software systems, software models are constantly changed during their development and evolutionary life-cycle. They are constantly evolved and maintained undergoing diverse changes such as extensions, corrections, optimization, adaptations and other improvements. All development and maintenance activities contribute to the evolution of software models resulting in several subsequent revisions. During the evolution process, models become large and complex raising a need for concurrent collaboration of several developers, designers and stakeholders (i.e., collaborators) on shared models, i.e., *Collaborative Modeling*.

Constantly modifying software models results in several subsequent *revisions* of the same modeling artifact. The differences between subsequent model revisions are identified and represented in model repositories as *modeling deltas*. Modeling deltas serve as information resources in further manipulations and analysis of software models. Modeling deltas representing changes between the subsequent revisions of models are the first-class entities in storing the histories of model changes in model repositories.

As models become large and complex over time, maintaining and developing such models require the concurrent collaboration of several developers in real-time. Concurrent collaboration is usually dedicated to instantly creating, modifying and maintaining huge, shared and centralized models in real-time by a team of collaborators. Thus, the changes made by collaborators have to be continually detected and synchronized among the several concurrent copies of that model. As long as synchronization has to occur in real-time, performance of interaction matters. Thus, model changes have to be represented and synchronized using very compact notations. Concurrent model copies can be differentiated by changes represented in small *modeling deltas*. The required performance of synchronization in real-time can be achieved by exchanging modeling deltas.

In collaborative modeling, *modeling deltas* are the first-class entities and play an essential role in storing, exchanging and synchronizing the changes between the subsequent and parallel revisions of evolving models. Thus, the efficient representation of changes in modeling deltas is crucial. An *efficient change representation notation* is needed for collaborative modeling which serves as the common underlying base-technology to represent modeling deltas.

This thesis introduces a *Difference Language (DL)* to the problem of model difference representation in collaborative modeling. The proposed DL is meta-model generic, operation-based, modeling tool generic, reusable, applicable, and extensible. DL is conceptually a family of domain-specific languages. Specific DLs for specific modeling languages can be generated from the meta-models of these modeling languages. Then, changes in instance models are described in terms of DL in modeling deltas. The DL-based modeling deltas consist of the executable descriptions of model changes.

Associated technical support further focuses on providing *a catalog of supplementary services* which allow for reusing and exploiting modeling deltas represented by DL. The DL-based modeling deltas are calculated, applied and reused by these DL services. These supplementary services extend the application areas of DL.

As the proof of the concept, the DL approach is applied to several applications areas in this thesis. *Concurrent collaborative modeling*, *sequential collaborative modeling* and *model history analysis* applications are taken into account in this thesis as the application areas of the DL-based difference representation approach. These applications are built by the specific orchestrations of the DL services following a certain data- and control-flow.

Besides this research work provides the open source, prototypical implementations of the DL services, applications and present empirical case studies and experiments for evaluating their usability and applicability.

Kurzfassung.
Als Paradigma der Softwareentwicklung ist Model-Driven Software Engineering (MDSE) eine moderne Form der Softwareentwicklung, die Softwareentwicklungsaktivitäten durch geeignete Abstraktionskonzepte unterstützt. Es zielt darauf ab, die Produktivität der Softwareentwicklung sowie Wartungsaktivitäten und Kommunikation zwischen verschiedenen Teammitgliedern und Stakeholdern zu verbessern. Im Kontext von MDSE sind Modelle, die auch Quellcode umfassen können, die zentralen Artefakte. MDSE bietet mehrere wichtige Vorteile, u.a. einen Produktivitätsschub. Modelle werden zum zentralen Entwicklungsmittel. Sie sind wiederverwendbar und werden automatisch mit dem Code aktualisiert, den sie repräsentieren.

Softwaremodelle (z. B. in der UML) sind die wichtigsten Artefakte in MDSE-Aktivitäten. Softwaremodelle eignen sich hervorragend zum Entwerfen und Entwickeln von großen Softwareprojekten. Um die ständig wachsenden Mengen von Softwareartefakten und deren Komplexität zu bewältigen, werden zu entwickelnde und zu wartende Softwaresysteme als Modelle abstrahiert. Softwaremodelle können zur Dokumentation und Implementierung von Softwaresystemen, die entwickelt und weiterentwickelt werden, erstellt werden.

Wie Quellcode von Softwaresystemen werden Softwaremodelle während ihrer Entwicklung und ihres evolutionären Lebenszyklus ständig verändert. Sie werden ständig weiterentwickelt und gewartet und durchlaufen diverse Änderungen wie Erweiterungen, Korrekturen, Optimierungen, Anpassungen und andere Verbesserungen. Alle Entwicklungs- und Wartungsaktivitäten tragen zur Entwicklung von Softwaremodellen bei, was zu mehreren sequentiellen Revisionen führt. Während des Entwicklungsprozesses werden Modelle zu großen und komplexen Artefakten, was die gleichzeitige (d.h. kollaborative) Zusammenarbeit von mehreren Entwicklern, Designern und Interessensvertretern an gemeinsamen Modellen, d.h. *Kollaborative Modellierung*, erfordert.

Das wiederholte Ändern von Softwaremodellen führt zu mehreren nachfolgenden *Revisionen* desselben Modellierungsartefakts. Die Differenzen zwischen sequentiellen Modellrevisionen werden identifiziert und in Repositories als *Modellierungsdeltas* gespeichert. *Modellierungsdeltas* dienen als Informationsressourcen bei weiteren Manipulationen und Analysen von Softwaremodellen. Modellierungsdeltas, die Veränderungen zwischen sequentiellen Revisionen von Modellen darstellen, sind zentralen Entitäten zum Speichern der Historien von Modelländerungen in Repositories.

Da Modelle im Laufe der Zeit groß und komplex werden, erfordert die Wartung und Entwicklung solcher Modelle die gleichzeitige Zusammenarbeit mehrerer Entwickler in Echtzeit. Daher müssen die von Entwicklern vorgenommenen Änderungen kontinuierlich zwischen mehreren Kopien dieses Modells erkannt und synchronisiert werden. Immer wenn die Synchronisation in Echtzeit stattfinden muss, spielt die Interaktion zwischen Entwicklern eine wichtige Rolle. Modelländerungen müssen daher mit sehr kompakten Notationen dargestellt und synchronisiert werden. Gleichzeitige Modellinstanzen können durch Änderungen in kleinen

Modelldeltas unterschieden werden. Die erforderliche Synchronisationsleistung in Echtzeit kann durch den Austausch kleiner Modelldeltas erreicht werden.

In der kollaborativen Modellierung sind *Modellierungsdeltas* Entitäten erster Klasse und spielen eine wesentliche Rolle beim Speichern, Austauschen und Synchronisieren der Änderungen zwischen den sequentiellen und parallelen Revisionen von sich entwickelnden Modellen. Daher ist die effiziente und kompakte Repräsentation von Modellierungsdeltas entscheidend. Eine *effiziente Repräsentationsnotation für Änderungen* wird für die kollaborative Modellierung benötigt, die als die gemeinsame zugrunde liegende Basistechnologie zur Repräsentation von Modellierungsdeltas dient.

Diese Dissertation führt eine *Difference Language (DL)* ein, um Modelldifferenzen zu repräsentatieren. Die vorgeschlagene DL ist generisch, operatorbasiert, metamodellgenerisch, wiederverwendbar, anwendbar und erweiterbar. DL ist konzeptionell eine Familie von domänenspezifischen Sprachen. Spezifische DLs für spezifische Modellierungssprachen können aus den Metamodellen dieser Modellierungssprachen generiert werden. Die DL-basierten Modellierungsdeltas bestehen aus ausführbaren Beschreibungen von Modelländerungen.

Die zugehörige technische Unterstützung konzentriert sich auch auf die Bereitstellung *eines Katalogs ergänzender DL Dienste (Services)*, die die Wiederverwendung und Nutzung von Modellierungsdeltas ermöglichen. Es werden Services zum Berechnen, Ausführen und zur Wiederverwenden von Modellierungsdeltas bereitgestellt. Diese Services werden wiederum für verschiedene Anwendungsfälle wiederverwendet.

Modellversionskontrolle, *gleichzeitige Modellierung in Echtzeit* und *Analyse der Modellhistorie* werden in dieser Arbeit als Anwendungsfälle des DL-basierten Differenzrepräsentation Ansatzes berücksichtigt. Diese Anwendungen werden durch spezifische Orchestrierungen der DL-Services sowohl für Daten als auch für den Kontrollfluss umgesetzt.

Darüber hinaus umfasst die Arbeit eine prototypische Implementierung der DL-Services als Open Source, Anwendungen und aktuelle empirische Fallstudien und Experimente zur Bewertung ihrer Relevanz, Nützlichkeit und Anwendbarkeit.

Acknowledgements

This thesis would not have been possible without the valuable contributions of many people. I owe my gratitude to all those, who have supported me during the time when I worked on this thesis and who made this time to be a precious experience for me.

My deepest gratitude is to my supervisor Prof. Dr. Andreas Winter, who provided me with every kind of support, assistance and freedom to explore. He constantly encouraged me to aim high and to keep pushing forward, while also giving me the guidance to get back on track when I struggled. I am indebted to Prof. Dr. Andreas Winter for the support with valuable feedback and always kindly encouraged me to succeed with my thesis and scientific career. I am very grateful to Prof. Dr. Ralf Reussner (my second supervisor) and Prof. Dr.-Ing. habil. Jorge Marx Gómez (Erasmus Mundus Target II coordinator at the University of Oldenburg) who had been both great mentors throughout my time working on my thesis.

I am deeply thankful to my colleagues Jan Jelschen, Johannes Meier, Christian Schönberg and Ruthbetha Kateule, who have always been there to share both my successes and my failings unconditionally. They always provided me with their help, advice, and their helpful comments during presentations. Furthermore, I am very thankful to students in the project group `Kotelett` for their contributions in prototypical development of an application for this thesis.

I would like to thank the full PhD scholarship program of the Erasmus Mundus Target II project and DAAD STIBET contact scholarship for granting me the scholarship and to come to Germany to write my PhD thesis.

Last but not least, I am very thankful to my beloved ones; my parents, my wife and my daughter for supporting me and always being there for me.

Contents