

Modellgetriebene generative Entwicklung von Web- Informationssystemen

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

vorgelegt von

Diplom-Wirtschaftsinformatiker

Dirk Reiß

aus Braunschweig

Berichter: Universitätsprofessor Dr. rer. nat. Bernhard Rumpe
Universitätsprofessor Dr. rer. nat. Albert Zündorf

Tag der mündlichen Prüfung: 7. Dezember 2015

Aachener Informatik-Berichte, Software Engineering

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

Band 22

Dirk Reiß

Modellgetriebene generative Entwicklung von Web-Informationssystemen

Shaker Verlag
Aachen 2016

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, 2015)

Copyright Shaker Verlag 2016

Alle Rechte, auch das des auszugsweisen Nachdruckes, der auszugsweisen oder vollständigen Wiedergabe, der Speicherung in Datenverarbeitungsanlagen und der Übersetzung, vorbehalten.

Printed in Germany.

ISBN 978-3-8440-4446-1

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

Die Entwicklung von Web-Informationssystemen, wie sie heutzutage an vielen Stellen im Netz zu finden sind, erfordert auch mit dem Einsatz aktueller Technologien immer noch einen hohen Anteil an repetitiven Tätigkeiten, die nicht nur aufwendig, sondern in der Regel auch fehleranfällig sind. Durch die Identifikation abstrahierbarer Aspekte der Anwendung und deren Abbildung auf Modelle können durch Verwendung des Konzepts der generativen Softwareentwicklung große Teile der Applikation automatisch erzeugt werden. Um möglichst flexibel hinsichtlich modellierbarer Domänen und Anwendungsgebiete zu sein, müssen neben der reinen Abstraktion auf Modelle jedoch auch Mechanismen vorhanden sein, um komplexe Geschäftslogik in die Anwendung integrieren zu können. Die identifizierten modellierbaren Kernaspekte des Web-Informationssystems betreffen die zugrunde liegende Datenstruktur, die Darstellung der verarbeiteten Daten inklusive einfacher aber flexibler Gestaltungsmöglichkeiten, die Berechtigungsstruktur innerhalb der Anwendung und die damit einhergehenden unterschiedlichen Sichten auf die Applikation, die Abfolge von Seitenaufrufen sowie einzel- und mehrbenutzerfähige Abläufe zur Abbildung der Geschäftslogik.

Im Rahmen dieser Arbeit wurden die beschriebenen Aspekte auf eine Menge textueller Modelle abgebildet, die zur Beschreibung und Generierung eines voll funktionsfähigen Web-Informationssystems verwendet werden. Diese sind zum Teil dem UML-Profil UML/P entnommen, zum Teil eine Erweiterung dieses Profils um ein Profil der UML-Aktivitätsdiagramme und zum Teil vollkommen eigenständige, domänenspezifische Sprachen. Die notwendige Flexibilität wird durch die Verwendung von Java als Aktionssprache innerhalb der Aktionen der Aktivitätsdiagramme gewährleistet. Das in dieser Arbeit entwickelte System setzt auf dem MontiCore-Framework auf und nutzt seine Möglichkeiten zur modularen Sprachdefinition für die Prüfung von Abhängigkeiten zwischen den Modellen.

Die Kombination dieser Sprachen erlaubt es sehr agil, auf Basis minimaler Modelle und durch ausgeprägtes, hinzugeneriertes Standardverhalten Prototypen zu erstellen, bietet aber auch genügend Freiheiten zur Modellierung komplexer Anwendungen verschiedenster Anwendungsdomänen.

Abstract

Even though the development of web information systems is supported by a number of actual web frameworks, it still requires a lot of redundant steps that are tedious and error prone in most cases. By identifying common aspects of a web information system and their mapping to models, we can apply mechanisms of generative software development to create major portions of the application automatically. In order to use the approach in various application domains, we further need means to integrate complex business logic into the description of the system. The identified core aspects of a web information system comprise the underlying data structure, the presentation of data including simple but flexible page design, the user rights management within the application alongside with the resulting different views for different user groups, as well as the page- and workflow for the specification of business logic.

The described aspects are mapped to a set of textual modeling languages which are used to describe and generate a complete web information system. Some of these languages are part of the UML profile UML/P, the used UML activity diagrams extend this profile and some of the developed languages are completely standalone and domain specific. In order to achieve the necessary flexibility, Java is used as an action language within the activity diagrams. The languages and generators are implemented based on the MontiCore framework, whose functionalities for modular language definitions are used to verify interdependency of the different models.

The combination of the developed languages is used during the agile development of web information systems, where a minimal set of models and extensive default behavior leads to full-fledged prototypes while iterative extensions allow the generation of complex applications from different problem domains.

Danksagung

Der lange Weg zu meiner Promotion war keiner, den ich allein beschritten habe. Vielmehr wurde ich dabei von mehreren Menschen begleitet, bei denen ich mich an dieser Stelle ganz herzlich bedanken möchte.

Zuallererst gilt mein ganz besonderer Dank Herrn Prof. Dr. Bernhard Rumpel für die Betreuung dieser Dissertation, die er auch nach seinem Wechsel von der TU Braunschweig an die RWTH Aachen weitergeführt hat. Durch das Ermöglichen der Teilnahme an den regelmäßigen Promotionsworkshops und die effektiven Diskussionen sowohl mit ihm als auch den Kollegen ergaben sich wertvolle Hinweise und Sichtweisen, die in diese Arbeit eingeflossen sind.

Ich danke Herrn Prof. Dr. Albert Zündorf für die Übernahme des Zweitgutachtens meiner Arbeit und den Herren Prof. Dr. Martin Grohe und Prof. Dr. Klaus Wehrle für die Bereitschaft, als Mitglieder der Promotionskommission meine Prüfung durchzuführen.

Meinen Kollegen an der RWTH Aachen und der TU Braunschweig verdanke ich eine wunderbare und spannende gemeinsame Zeit. Bezogen auf diese Arbeit sind jedoch einige von ihnen besonders hervorzuheben. Herrn Dr. Holger Krahn, Herrn Dr. Steven Völkel und Herrn Dr. Martin Schindler danke ich für die Grundlagen, die sie im Rahmen ihrer Promotionen gelegt haben und auf denen die vorliegende Arbeit in Teilen aufsetzt. Dank gebührt ebenfalls den Herren Mark Stein und Michael Dukaczewski für die fruchtbaren Diskussionen, die zu Beginn in Vorarbeiten zu dieser Arbeit in Kooperation stattfanden, sich dann aber in unterschiedliche Richtungen entwickelten. Letzterem danke ich herzlich für seine überaus hilfreichen Anmerkungen und die Durchsicht dieser Arbeit. Den Herren Thomas Kurpick, Ludger Steens, Robert Eikermann sowie Dominik Merkelbach gilt mein Dank für die Nutzung von MontiWIS in ihren eigenen Arbeiten und den Herren Florian Leppers, Jerome Pfeiffer und Philipp Schütze für die Betreuung des mit MontiWIS entwickelten Bibliothekssystems. Sie alle lieferten gewinnbringende Anmerkungen, die zu sinnvollen Anpassungen in dieser Arbeit geführt haben.

Abschließend danke ich meinen Eltern, meinem Bruder und meiner Schwägerin sowie all meinen Freunden für jegliche Unterstützung und den Rückhalt, die diese Arbeit überhaupt erst möglich gemacht haben.

Inhaltsverzeichnis

1	Einführung	1
1.1	Motivation	2
1.2	Ziele und Anforderungen	3
1.3	Aufbau der Arbeit	5
2	Web-Informationssysteme	7
2.1	Einordnung und Abgrenzung	7
2.2	Anforderungen an Web-Informationssysteme	8
2.2.1	Funktionale Anforderungen	8
2.2.2	Qualitative (nicht-funktionale) Anforderungen	11
2.3	Web Engineering allgemein und Ansätze der Implementierung	11
2.3.1	Manuelle Implementierung	12
2.3.2	Erhöhte Abstraktion durch den Einsatz von Frameworks	14
2.3.3	Generative Ansätze	16
2.4	Durchgängiges Beispiel	16
3	Modellgetriebene generative Softwareentwicklung	19
3.1	Begriffsbildung und Abgrenzungen	19
3.2	Werkzeuge für die modellgetriebene Entwicklung	21
3.2.1	Grafische Ansätze	22
3.2.2	Textuelle Ansätze	22
3.3	MontiCore	23
3.3.1	Grundlagen der Entwicklung mit MontiCore	23
3.3.2	Symboltabelleninfrastruktur	24
3.4	Anforderungen an die Modelle und die Infrastruktur zur Modellverarbeitung	26
3.4.1	Zu modellierende Aspekte	26
3.4.2	Anforderungen an die technische Umsetzung der Sprachen	30
3.5	Verwandte Arbeiten im Bereich des modellbasierten Web Engineerings	30
4	Ablauf der Generierung	39
4.1	Gesamtüberblick über den Generierungsprozess	39
4.1.1	Überblick über beteiligte Komponenten	39
4.1.2	Zusammenspiel der Komponenten	41
4.2	Sprachverarbeitung mit MontiCore	43

4.3	Verwendung von Symboltabelle und Kontextbedingungen	44
4.4	Verwendung der Zwischenstruktur	45
4.5	Rekonstruktion von Zwischenstrukturelementen	46
4.6	Generierung des Web-Informationssystems	46
5	Generiertes System	49
5.1	Überblick über das generierte Web-Informationssystem	49
5.2	Datenstruktur und Persistenz	51
5.2.1	Datenklassen	51
5.2.2	Data Access Object Klassen	51
5.3	Nutzerinteraktion und clientseitige Technologien	52
5.3.1	Generelle Funktionsprinzipien	52
5.3.2	JSP Seiten	53
5.3.3	PageEntities	54
5.3.4	Ressourcenklassen	54
5.4	Ablaufsteuerung	55
5.4.1	Generierte Knotenstruktur	56
5.4.2	Kontextklassen zur Speicherung des Ablaufzustands	56
5.4.3	Executorklassen	56
5.4.4	Ressourcenklassen für Abläufe	57
5.5	Applikationssteuerung	57
5.5.1	Menüs	57
5.5.2	Nutzerverwaltung	57
6	Datenstruktur	59
6.1	Grundlagen und Anforderungen	59
6.1.1	Grundlagen der Datenmodellierung	59
6.1.2	Anforderungen an die Datenmodellierungssprache	59
6.2	Sprache	60
6.2.1	Aufbau der Klassendiagrammsprache	61
6.2.2	Verwendung der Klassendiagrammsprache	63
6.2.3	Vordefinierte Datentypen	69
6.2.4	Profil für MontiWIS und Modellierungsrichtlinien	72
6.3	Generierungsprozess	77
6.3.1	Kontextbedingungen	78
6.3.2	Aufbau der Zwischenstruktur	78
6.3.3	Modellverarbeitung durch einen Visitor	83
6.3.4	Rekonstruktion der Klassendiagramminformationen	86
6.3.5	Erzeugung von Standardseiten für jede Klasse	87
6.3.6	Generator	90
6.4	Implementierung im Zielsystem	94
6.4.1	Datenstruktur und Persistenzmechanismen	94
6.4.2	Generierung der Standardseiten	101
6.5	Andere Ansätze	106

6.5.1	Andere Ansätze der Implementierung	106
6.5.2	Andere Ansätze der Datenmodellierung	107
7	Seitenbeschreibung	109
7.1	Grundlagen und Anforderungen	109
7.1.1	Grundlagen	109
7.1.2	Anforderungen an die Seitenbeschreibungssprache	110
7.2	Sprache	112
7.2.1	Grammatik der Seitenbeschreibungssprache	113
7.2.2	Verwendung der Seitenbeschreibungssprache	116
7.2.3	Darstellung im generierten System	128
7.3	Generierungsprozess	137
7.3.1	Kontextbedingungen	137
7.3.2	Aufbau der Zwischenstruktur	139
7.3.3	Modellverarbeitung durch einen Visitor	144
7.3.4	Rekonstruktion des Seitenaufbaus	144
7.3.5	Generator	146
7.4	Implementierung im Zielsystem	152
7.4.1	Generelle Seitenarchitektur	152
7.4.2	Umsetzung der verschiedenen Datentypen	157
7.4.3	PageEntity- und PageEntityInitializer-Klassen	163
7.4.4	Ressourcenklassen für Seiten	167
7.4.5	Sicherheitsüberlegungen	168
7.5	Andere Ansätze	170
7.5.1	Imperativer Ansatz	170
7.5.2	Deklarativer Ansatz	172
8	Ablaufsteuerung	173
8.1	Grundlagen und Anforderungen	173
8.1.1	Grundlagen	173
8.1.2	Anforderungen an die Ablaufsteuerung	173
8.2	Sprache	176
8.2.1	Aufbau der Aktivitätsdiagrammsprache	176
8.2.2	Verwendung der Aktivitätsdiagrammsprache	181
8.2.3	Beschreibung der einzelnen Sprachelemente	184
8.3	Inhalt von Aktionen	195
8.3.1	Parameter	196
8.3.2	Variablen	197
8.3.3	Aufruf einer Seite	197
8.3.4	Java	198
8.3.5	Kommandos	200
8.3.6	Aufruf anderer Aktivitäten	208
8.4	Generierungsprozess	209
8.4.1	Kontextbedingungen	210

8.4.2	Aufbau der Zwischenstruktur	213
8.4.3	Modellverarbeitung von Aktivitätsdiagrammen durch einen Visitor	216
8.4.4	Rekonstruktion der Aktivitätsdiagrammelemente	220
8.4.5	Generator	221
8.5	Umsetzung im Zielsystem	225
8.5.1	Übersicht über die Laufzeitumgebung	225
8.5.2	Ausführung von Abläufen	234
8.5.3	Weitere Laufzeitkomponenten	241
8.6	Andere Ansätze	241
8.6.1	Workflowbeschreibungssprachen	241
8.6.2	Web-Modellierungsansätze	243
9	Applikationssprache	245
9.1	Grundlagen und Anforderungen	245
9.1.1	Grundlagen	245
9.1.2	Anforderungen an die Applikationssprache	245
9.2	Sprache	247
9.2.1	Aufbau der Applikationssprache	247
9.2.2	Verwendung der Applikationssprache	249
9.3	Generierungsprozess	253
9.3.1	Kontextbedingungen	254
9.3.2	Aufbau der Zwischenstruktur	255
9.3.3	Modellverarbeitung durch einen Visitor	257
9.3.4	Rekonstruktion	258
9.3.5	Generator	258
9.4	Umsetzung im generierten System	262
9.4.1	Umsetzung des Menüs	262
9.4.2	Umsetzung der Rechte	264
9.5	Andere Ansätze	267
9.5.1	Möglichkeiten der Zugriffssteuerung	267
9.5.2	Methoden zur Spezifikation von Menüs in Webanwendungen	268
10	Anwendungsmethodik	271
10.1	Überblick über MontiWIS-Anwendungsprojekte	271
10.1.1	Anlegen eines neuen Projekts	271
10.1.2	Aufbau eines MontiWIS-Anwendungsprojektes	272
10.2	Durchführung der Generierung	276
10.2.1	Lokale Vorbedingungen	277
10.2.2	Lokale Generierung	277
10.2.3	Generierung über OSTP	278
10.2.4	Schritte nach der Generierung	278
10.3	Iterative Entwicklung des Web-Informationssystems	279
10.3.1	Festlegen von Anwendungsfällen und Akteuren	280
10.3.2	Erweiterung der Datenstruktur	280

10.3.3 Erstellen von Aktivitäten	280
10.3.4 Spezifikation von Seiten	281
10.4 Anpassung / Erweiterung der Implementierung	281
10.4.1 Umbenennen von Buttons und Standardbezeichnungen	281
10.4.2 CSS-Anpassungen	283
11 Fallstudien und Bewertung	285
11.1 Bibliotheksverwaltung	285
11.1.1 Datenmodell	285
11.1.2 Aktivitäten	285
11.1.3 Applikationsmodell	287
11.2 Modulhandbuch	288
11.2.1 Datenmodell	288
11.2.2 Applikationsmodell	288
11.2.3 Aktivitäten	289
11.3 Robotersteuerung	291
11.3.1 Datenmodell	292
11.3.2 Aktivitäten	292
11.3.3 Applikationsmodell	292
11.4 Weitere Verwendung	292
11.4.1 Feedback bezüglich statischen Komponenten von MontiWIS	292
11.4.2 Feedback bezüglich dynamischer Komponenten von MontiWIS	293
11.4.3 Online-Demonstrator	293
11.5 Bewertung der Ergebnisse	293
11.5.1 Quantitative Bewertung	294
11.5.2 Qualitative Bewertung	295
11.5.3 Modell- und Infrastrukturanforderungen	297
11.5.4 Bewertung der Designentscheidungen	298
12 Zusammenfassung und Ausblick	301
12.1 Zusammenfassung	301
12.2 Ausblick	303
Literaturverzeichnis	305
Abbildungsverzeichnis	325
Tabellenverzeichnis	329
A Glossar	331
B Abkürzungen	335
C Grammatiken	339
C.1 MontiCore Literals	339

C.2	MontiCore Types	347
C.3	UML/P Common	353
C.4	Java	357
C.5	Klassendiagramme	381
C.6	Seitenbeschreibung	387
C.7	Aktivitätsdiagramme	393
C.8	Applikationsprache	403
D	Lebenslauf	407