

# Inhaltsverzeichnis

<b>1. Einleitung</b>	<b>1</b>
1.1. Kontext und Einordnung der Arbeit . . . . .	1
1.2. Wissenschaftliche Fragestellung dieser Arbeit . . . . .	3
1.3. Wesentliche Ergebnisse . . . . .	4
1.4. Aufbau der Arbeit . . . . .	5
<b>2. Anwendung und Entwicklung von Modelltransformationen</b>	<b>9</b>
2.1. Modelle und domänenspezifische Sprachen im Software-Engineering . . . . .	9
2.1.1. Einsatz domänenspezifischer Sprachen . . . . .	9
2.1.2. Standardisierungsansätze zur Modellierung . . . . .	10
2.1.3. Grafische und textuelle Sprachen . . . . .	11
2.1.4. Ansätze zur Sprachdefinition . . . . .	12
2.2. Methodik der Nutzung von Modelltransformationen . . . . .	14
2.2.1. Nutzungsszenarien . . . . .	15
2.2.2. Nutzbarkeit von Transformationssprachen für Domänenexperten . . . . .	18
2.3. Kategorisierung von Transformationssprachen und -werkzeugen . . . . .	19
2.3.1. Unterscheidungskriterien . . . . .	19
2.3.2. Einordnung dieser Arbeit . . . . .	21
2.4. Wissenschaftliche Herausforderungen für Modelltransformationen . . . . .	22
2.5. Motivation der vorliegenden Arbeit . . . . .	24
<b>3. Durchgängiges Beispiel: Vereinfachung hierarchischer Statecharts</b>	<b>25</b>
3.1. Motivation und Anwendbarkeit . . . . .	25
3.2. Einführung in die Statechartsprache . . . . .	26
3.2.1. Zustände und Transitionen . . . . .	27
3.2.2. Stimuli, Aktionen und Guards . . . . .	28
3.2.3. Invarianten, Vor- und Nachbedingungen . . . . .	29
3.2.4. Entry- und Exit-Aktionen . . . . .	29
3.2.5. Do-Aktivitäten und interne Transitionen . . . . .	30
3.2.6. Die textuelle Syntax der Statechartsprache . . . . .	30
3.3. Semantikerhaltende Transformationsregeln . . . . .	33
3.3.1. Do-Aktivitäten eliminieren . . . . .	34
3.3.2. Transitionen an initiale Subzustände umleiten . . . . .	35
3.3.3. Transitionen aus finalen Subzuständen starten lassen . . . . .	36
3.3.4. Exit-Aktionen an ausgehende Transitionen verschieben . . . . .	36
3.3.5. Entry-Aktionen an eingehende Transitionen verschieben . . . . .	37

3.3.6. Interne Transitionen eliminieren . . . . .	37
3.3.7. Zustandsinvarianten an Subzustände propagieren . . . . .	38
3.3.8. Hierarchisch zergliederte Zustände entfernen . . . . .	38
3.4. Verknüpfung der Regeln . . . . .	39
3.5. Anforderungen an die Transformationssprache . . . . .	41
<b>4. Transformationsregeln in Objektdiagramm-Notation</b>	<b>43</b>
4.1. Grundlagen . . . . .	43
4.1.1. Konkrete Syntax, abstrakte Syntax und attributierte Graphen . . . . .	43
4.1.2. Graphersetzungsrregeln . . . . .	45
4.2. Aufbau der Regeln . . . . .	48
4.3. Effizientes Pattern-Matching . . . . .	51
4.3.1. Reduktion des Suchraums . . . . .	52
4.3.2. Kostenmodell und Optimierung des Suchplans . . . . .	53
4.3.3. Implementierung des Pattern-Matching-Algorithmus . . . . .	55
4.4. Negative Objekte und Listenobjekte . . . . .	57
4.5. Ersetzung mit dem Entwurfsmuster Command . . . . .	59
4.6. Verwandte Arbeiten . . . . .	60
4.6.1. Verwendung von Constraint-Solvern . . . . .	61
4.6.2. Abbildung auf das relationale Datenmodell . . . . .	61
4.6.3. Suchplangesteuerte Ansätze . . . . .	62
<b>5. Eine Regelsprache mit domänenspezifischer Syntax</b>	<b>65</b>
5.1. Syntax und Semantik der Regeln . . . . .	65
5.1.1. Pattern Matching . . . . .	66
5.1.2. Modifikationen des Modells . . . . .	69
5.2. Ableitung der kontextfreien Syntax aus der Basis-DSL . . . . .	72
5.2.1. Terme in Transformationsregeln . . . . .	72
5.2.2. Umgang mit Attributen . . . . .	73
5.3. Kontextbedingungen und Empfehlungen für Anwender . . . . .	77
5.3.1. Typen der Schemavariablen . . . . .	77
5.3.2. Disjunktheit der Matches von Listenknoten . . . . .	78
5.3.3. Formatierungs- und Programmierrichtlinien . . . . .	79
5.4. Übersetzung in Objektdiagramm-Notation . . . . .	81
5.4.1. Generierung der linken und rechten Regelseite . . . . .	81
5.4.2. Listen- und negative Knoten, Constraints und nichtisomorphe Matches . . . . .	82
5.4.3. Implementierung des Transformators . . . . .	85
5.5. Umgang mit Bezeichnern . . . . .	86
5.5.1. Bezeichner bei der Mustersuche . . . . .	87
5.5.2. Bezeichnerersetzung . . . . .	87
5.5.3. Bezeichner bei der Generierung von Objektdiagrammregeln . . . . .	88
5.6. Verwandte Arbeiten . . . . .	89

<b>6. Die Kontrollflusssprache für Modelltransformationen</b>	<b>93</b>
6.1. Aufbau der Transformationsmodule . . . . .	94
6.1.1. Module und Methoden . . . . .	94
6.1.2. Sequenzielle Ausführung von Anweisungen . . . . .	95
6.1.3. Bedingte Anweisungen und Schleifen . . . . .	96
6.1.4. Eingebettete Java-Ausdrücke . . . . .	97
6.1.5. Typsystem der Kontrollflusssprache . . . . .	97
6.2. Einbettung der Transformationsregeln und Methodenaufrufe . . . . .	98
6.3. Nichtdeterminismus . . . . .	100
6.4. Codegenerierung . . . . .	102
6.4.1. Struktur des generierten Codes . . . . .	102
6.4.2. Ablauf der Codegenerierung . . . . .	103
6.4.3. Abbildung des Nichtdeterminismus . . . . .	106
6.5. Verwandte Arbeiten . . . . .	106
<b>7. Generierung von Transformationssprachen aus DSL-Grammatiken</b>	<b>111</b>
7.1. Anwendungsszenario . . . . .	111
7.2. Generierung der Regelgrammatik . . . . .	112
7.2.1. Inhalt der Regelgrammatiken . . . . .	113
7.2.2. Technische Umsetzung . . . . .	117
7.3. Generierung des Transformators . . . . .	117
7.3.1. Semantik der DSLs und Semantik der generierten Regelsprachen . . . . .	118
7.3.2. Arbeitsweise und Aufbau des Transformators . . . . .	119
7.3.3. Technische Umsetzung . . . . .	119
7.4. Einbettung in die Kontrollflusssprache und generierte Werkzeuge . . . . .	125
<b>8. Nutzung von Transformationen und Transformationssprachen</b>	<b>129</b>
8.1. Generierung der Transformationssprache durch den Sprachentwickler . . . . .	129
8.2. Aufruf der generierten Werkzeuge und Workflows . . . . .	130
8.2.1. DSLTools für Transformationen . . . . .	131
8.2.2. Workflows . . . . .	131
8.3. Integration von Transformationen in den Codegenerierungsprozess . . . . .	132
8.3.1. Sprachen, Transformationen und Modelle . . . . .	133
8.3.2. Transformationen in der Codegenerierung für Statecharts . . . . .	133
8.4. Fallbeispiele aus der Vereinfachung hierarchischer Statecharts . . . . .	135
8.4.1. Verschieben von Elementen . . . . .	136
8.4.2. Kopieren von Elementen . . . . .	136
8.4.3. Veränderung gebundener Bezeichner . . . . .	137
8.4.4. Verwendung von Kontrollstrukturen . . . . .	138
8.4.5. Erstellung eines ausführbaren Programms . . . . .	139

<b>9. Die Entwicklung der Modelltransformationsengine</b>	<b>141</b>
9.1. Projektstruktur und agile Entwicklung mit MontiCore . . . . .	141
9.1.1. Innere Struktur der Transformationsprojekte . . . . .	141
9.1.2. Testgetriebene Entwicklung der MontiCore-Transformationsengine . .	143
9.2. Prototypenbasierte Entwicklung der Generatoren . . . . .	144
9.2.1. Prototypen- und Generatorprojekte mit Modultests . . . . .	145
9.2.2. Systemtest . . . . .	146
9.3. Testgetriebene Weiterentwicklung und Wartung . . . . .	147
9.3.1. Testen einfacher Generatoren . . . . .	148
9.3.2. Testen des Generators für generierte Regelsprachen . . . . .	149
9.4. Vor- und Nachteile der Ansätze . . . . .	150
<b>10. Epilog</b>	<b>153</b>
10.1. Zusammenfassung . . . . .	153
10.2. Ausblick . . . . .	155
10.2.1. Anwendungen der Ergebnisse dieser Arbeit . . . . .	155
10.2.2. Konzeptuell verwandte Problemstellungen . . . . .	156
10.2.3. Technische Erweiterungen . . . . .	157
10.3. Abschließende Bemerkungen . . . . .	159
<b>Literaturverzeichnis</b>	<b>161</b>
<b>A. Glossar und Abkürzungsverzeichnis</b>	<b>175</b>
<b>B. Notationen</b>	<b>181</b>
<b>C. Grammatiken</b>	<b>183</b>
C.1. Grammatik der Statechartsprache . . . . .	183
C.2. Grammatik der Transformationsregeln auf Statecharts . . . . .	185
<b>D. Transformation zur Vereinfachung hierarchischer Statecharts</b>	<b>211</b>
<b>E. Lebenslauf</b>	<b>219</b>