

# Inhaltsverzeichnis

<b>1. Einführung</b>	<b>1</b>
1.1. Motivation . . . . .	2
1.2. Zielsetzung . . . . .	3
1.3. Aufbau der Arbeit . . . . .	3
<b>2. Grundlagen</b>	<b>4</b>
2.1. Modellgetriebene Softwareentwicklung (MDSD) . . . . .	5
2.1.1. Domäne . . . . .	5
2.1.1.1. Definition . . . . .	5
2.1.1.2. Domänenanalyse . . . . .	5
2.1.1.3. Feature Modelling . . . . .	5
2.1.2. Metamodell . . . . .	5
2.1.2.1. Definition . . . . .	5
2.1.2.2. Abstraktes Modell . . . . .	5
2.1.2.3. Konkretes Modell . . . . .	5
2.1.3. Domänenspezifische Sprache (DSL) . . . . .	5
2.1.3.1. Definition . . . . .	5
2.1.3.2. General Purpose Language (GPL) . . . . .	5
2.1.3.3. Interne DSLs . . . . .	5
2.1.3.4. Externe DSLs . . . . .	5
2.1.3.5. Parser . . . . .	5
2.1.4. Code Generator . . . . .	5
2.1.4.1. Definition . . . . .	5
2.1.4.2. Techniken zur Generierung von Code . . . . .	5
2.1.4.3. Zusammenhang zu Transformatoren . . . . .	5
2.1.4.4. Abgrenzung zum Compilerbau . . . . .	5
2.1.5. MDSD Entwicklungsprozess . . . . .	5
2.2. Software Engineering . . . . .	5
2.2.1. Architektur . . . . .	5
2.2.1.1. Prinzipien in der Softwaretechnik . . . . .	5
2.2.1.2. Trennung durch Modelle . . . . .	5
2.2.1.3. Modell-Transformations-Pipeline . . . . .	5
2.2.2. Design Pattern objektorientierter Programmierung . . . . .	5
2.2.2.1. Visitor Pattern . . . . .	5
2.2.2.2. Builder Pattern und Fluent Interfaces . . . . .	5

2.2.2.3. Factory Pattern . . . . .	5
<b>3. Problemstellung</b>	<b>6</b>
3.1. Wahl der zu unterstützenden Features . . . . .	6
3.2. Definition der Variabilitäten auf Basis einer Referenzimplementation . . .	6
3.3. Verwendung von Zwischenmodellen . . . . .	6
3.4. Entscheidung zwischen Meta-Generator und Transformator . . . . .	6
3.5. Abstrakte Modellierung von Java-Code . . . . .	6
<b>4. Lösung: Spectrum (Proof of Concept)</b>	<b>7</b>
4.1. Verwendete Bibliotheken . . . . .	8
4.1.1. JavaParser mit JavaSymbolSolver . . . . .	8
4.1.2. JavaPoet . . . . .	8
4.2. Architekturübersicht . . . . .	8
4.2.1. Amber . . . . .	8
4.2.1.1. Annotationen . . . . .	8
4.2.1.2. Parser . . . . .	8
4.2.1.3. Modell . . . . .	8
4.2.2. Cherry . . . . .	8
4.2.2.1. Generator . . . . .	8
4.2.2.2. Modell . . . . .	8
4.2.2.3. Generierte Builder . . . . .	8
4.2.3. Jade . . . . .	8
4.2.3.1. Transformator . . . . .	8
4.2.4. Scarlet . . . . .	8
4.2.4.1. Generator . . . . .	8
4.2.4.2. Modell . . . . .	8
<b>5. Evaluierung</b>	<b>10</b>
5.1. Softwarequalität . . . . .	10
5.1.1. Functionality . . . . .	10
5.1.2. Maintainability . . . . .	10
5.1.3. Performance . . . . .	10
5.1.4. Usability . . . . .	10
5.2. Grenzen des Lösungsansatzes . . . . .	10
<b>6. Zusammenfassung und Ausblick</b>	<b>12</b>
6.1. Zusammenfassung . . . . .	12
6.2. Ausblick . . . . .	12
<b>A. Dokumentation</b>	<b>13</b>
A.1. Verwendung der Annotationen . . . . .	13
A.2. Verwendung der generierten CodeUnit-Builder . . . . .	13

## *Inhaltsverzeichnis*

A.3. Klassendokumentation . . . . .	13
A.3.1. Amber . . . . .	13
A.3.2. Cherry . . . . .	13
A.3.3. Jade . . . . .	13
A.3.4. Scarlet . . . . .	13
<b>Verzeichnisse</b>	<b>14</b>
<b>Literatur</b>	<b>16</b>
<b>Eidesstattliche Erklärung</b>	<b>17</b>
<b>Zustimmung zur Plagiatsüberprüfung</b>	<b>18</b>