
Inhaltsverzeichnis

1 Anwendungen der Modellierung in der Programmierung:	
Modeling4Programming	1
1.1 Überblick	1
1.2 Papyrus-Framework zur Modellierung mit SysML/UML	4
1.2.1 UML-Aktivitätsdiagramme mit Eclipse-Papyrus	5
1.2.2 Zustandsdiagramme mit Eclipse-Papyrus	6
1.2.3 Das Erstellen von Sequenzdiagrammen mit Eclipse-Papyrus	7
1.2.4 Das Erstellen von Anwendungsfalldiagrammen mit Eclipse-Papyrus	7
1.3 Obeo-UML-Designer	8
1.3.1 Das Visualisieren der Diagramme auf Basis von UML 2.5	9
1.3.2 Überblicke über UML-Diagramme mit Eclipse-UML-Designer	13
1.3.2.1 Strukturelle Diagramme	15
1.3.2.2 Verhaltensbasierte Diagramme	16
1.3.3 Beispiele von UML-Diagrammen	16
1.3.3.1 Klassendiagramme für Stromversorgungstester	17
1.3.3.2 Komponentendiagramme für die Websystemqualität	17
1.3.3.3 Zustandsdiagramme für die Websystemqualität	19
1.3.3.4 Profildiagramme für parallele Prozesse mit dem Asynchronmotor	19
1.3.3.5 Verteilungsdiagramme oder Deployment-Diagramm	21
1.4 UML-SysML-Struktur	22
1.4.1 Blockdefinitionsdiagramme	22
1.4.2 Interne Blockdiagramme	24
1.4.3 Anforderungsdiagramme	25
1.4.4 Zusicherungsdiagramme	25
1.5 IT-Lösungen mit Modeling4Programming (M4P)	28
1.5.1 Modellierung von IT-Lösungen mit C++	28
1.5.1.1 Anwendungen der Funktionalitäten der Elektronik in der Modellierung	29

1.5.1.2	Modellierungsaspekte mithilfe der Programmierung	36
1.5.1.2.1	Aktivitätsdiagramm	37
1.5.1.2.2	Sequenzdiagramm	41
1.5.1.2.3	Kommunikationsdiagramm	42
1.5.1.2.4	Zustandsdiagramm	43
1.5.1.2.5	Inneres Klassendiagramm	44
1.5.2	Modellierungen von Java-Anwendungen mithilfe von UML	53
1.5.2.1	Modellierung von Anwendungen für den Einsatz von Resonanzelementen Kondensator und Spule mit Eclipse UML Designer	53
1.5.2.2	Anwendung der Programmierperformance in der Berechnung der Kenngröße Wirkungsgrad des Motors. . .	58
1.5.2.3	Modellierung der Vermeidung der Kollision zwischen gleichnamigen Methoden aus zwei unterschiedlichen Interfaces	61
1.5.3	Umsetzung der funktionellen Modellierung in der Programmierung.	67
1.5.3.1	Java für die funktionelle Modellierung	67
1.5.3.1.1	Lambda-Ausdrücke zur Modellierung der Berechnungen	68
1.5.3.1.2	Modellierung der funktionalen Berechnung mithilfe von konstanten Eingaben.	72
1.5.3.2	C++ für die funktionelle Modellierung	74
1.5.3.2.1	Modellierung der Entfernung eines gezielten Elements im Vektor.	75
1.5.3.2.2	C++-Standardbibliothek mit Funktionstemplate zum Modellieren der Funktionalität des Wirkungsgrads	76
1.5.3.2.3	„C++-Standardbibliothek mit Funktionstemplate“, Iteratoren und Überladen der <i>Operatoren</i> zum Modellieren der Orthogonalität zwischen zwei Vektoren.	78
1.5.3.2.4	Lambda-Funktion zum Darstellen der Derivation einer Funktion	81
1.5.3.2.5	Anwendungen der C++-Standardbibliothek in der Modellierung der Parallelisierung . . .	83
1.5.3.2.6	Anwendungen der C++-Standardbibliothek in der Implementierung der Funktion <i>std::tuple()</i>	87

1.5.3.2.7	Anwendung der <i>std::map</i> -Objekte in der Analyse der Elemente einer Sammlung	90
1.5.3.2.8	Modellierung der Ausnahmebehandlung mithilfe der Ein- und Ausgabemöglichkeit.	93
1.5.3.2.9	Modellierung des Überladens der Operatoren < und << mithilfe des <i>set-Containers</i>	95
1.5.3.2.10	Modellierung der Anwendung der Funktion <i>evaluate()</i> in der Analyse der Zeiger auf Funktionen	97
1.5.3.2.11	Modellierung der Funktionalität von <i>std::for_each()</i> zum Darstellen der Lambda-Ausdrücke-Rolle	98
1.5.3.2.12	Anwendungen der Funktionen <i>std::copy()</i> und <i>std::transform()</i> zum Darstellen der Parallelisierung.	101
1.6	Softwarearchitektur mit Eclipse-Papyrus und UML-Designer	103
1.6.1	Modellierung eines Klassendiagramms mithilfe der Open Source Eclipse-Papyrus zum Analysieren eines Testprogramms mit Junit	103
1.6.1.1	Modellierung eines Testsystems für die Energietools	104
1.6.1.2	Modellierung eines Testsystems für WLAN-Systeme. . . .	107
1.6.2	Modellierung des Klassendiagramms zum Beschreiben der parametrisierten Systeme mit Eclipse Ecore Framework.	110
1.6.3	Modellierungen der parallelen Implementierungen von Interfaces	119
1.6.4	Modellierung der Funktionalitäten der Pattern-Methoden mithilfe des Klassendiagramms von Eclipse UML Designer	131
1.6.5	Modellierung der Anwendungen des Interface <i>Collection</i> mit dem Klassendiagramm von Eclipse UML Designer	135
1.7	C++ mit Open Source Eclipse	138
1.7.1	Überblick über den Compiler Cygwin zum Programmieren mit C/C++.	138
1.7.2	Das Programmieren mit Eclipse CDT	143
1.7.2.1	Erstellung von Projekten mit C++.	144
1.7.2.2	Code-Einblicke mit C++	144
1.7.2.2.1	Code-Einblicke für eine Klasse	145
1.7.2.2.2	Code-Einblicke für <i>Abstraktion</i>	152

1.8	Neues von Java und Jakarta EE	159
1.8.1	Anwendungen mit Record-Klasse	160
1.8.2	Jakarta EE.	169
1.8.2.1	Entwicklung von UI-Anwendungen mithilfe von Jakarta EE Server und Jakarta Faces	169
1.8.2.2	Entwicklung von Enterprise-Anwendungen mithilfe von Jakarta EE Server und Enterprise Java Beans.	174
1.9	Zusammenfassung	185
	Literatur.	187
2	UML-Modellierung mit der Eclipse-Umgebung	189
2.1	Modellierung des Klassendiagramms mit Obeo UML Designer.	189
2.1.1	Vererbung	195
2.1.2	Eigenschaften der Klassen	198
2.1.3	Modellierung des Klassendiagramms mithilfe der Operationen . . .	200
2.1.4	Praxisbeispiel: Anwendung der Klassendiagramme in der Modellierung des Durchlassverhaltens des Transistors	202
2.2	Kompositionsstrukturdiagramm von Obeo Designer UML.	214
2.3	Zustandsdiagramm von Obeo UML Designer.	223
2.3.1	Überblick über Erstellungstools des Zustandsdiagramms	225
2.3.2	Notationselemente	225
2.3.3	Anwendung des Zustandsdiagramms in der Energietechnik	226
2.4	Komponentendiagramm.	227
2.4.1	Komponentenmodell von Jakarta EE	228
2.4.2	Komponenten für Jakarta EE	229
2.4.3	Komponenten für <i>Jakarta Faces</i> (oder <i>Jakarta Server Faces</i>), <i>Jakarta Persistence</i> (früher <i>JPA</i>) und <i>Contexts and Dependency Injection (CDI)</i>	230
2.4.3.1	<i>Jakarta Faces</i> (früher JSF)	231
2.4.3.2	<i>Jakarta Persistence</i> 2.3 (früher <i>JPA</i>)	231
2.4.3.3	<i>Jakarta Contexts and Dependency Injection (CDI)</i>	232
2.5	Verteilungsdiagramm (Deployment-Diagramm)	232
2.5.1	Device für <i>Application-Server Jakarta EE</i>	232
2.5.2	<i>Device-Client</i>	234
2.5.3	Device für MySQL-Datenbankserver.	234
2.6	Zusammenfassung	234
	Literatur.	236
3	Eclipse-Papyrus-Framework	239
3.1	Erstellung eines UML-Klassendiagramms	239
3.1.1	Struktur des UML-Klassendiagramms.	244
3.1.2	Beispiel: Inneres Klassendiagramm.	248
3.1.2.1	Überblick über Assoziationen	250

3.1.2.2	Überblick über Generalisierung.	250
3.1.2.3	Vererbungskaskade	251
3.2	Paketdiagramm.	253
3.2.1	Paketdiagramm mit dem Design-Pattern <i>Model View Controller(MVC)</i>	260
3.2.2	Überblick über Java-Code in dem Modell	261
3.3	Class Tree Table.	262
3.3.1	Struktur der Tabelle	262
3.3.2	Vertikale Position	266
3.3.3	Horizontale Position.	268
3.4	Sequenzdiagramme mit Eclipse-Papyrus.	270
3.5	Kommunikationsdiagramm mit Eclipse-Papyrus	279
3.6	Objektdiagramme mit Eclipse Papyrus	285
3.6.1	Das Erstellen eines Klassendiagrammes mit Eclipse-Papyrus.	286
3.6.2	Das Erstellen eines Objektdiagramms mit Eclipse-Papyrus	291
3.6.2.1	Elemente des Objektdiagramms	311
3.6.2.2	Grafische Darstellung des Objektdiagramms	312
3.7	Kompositionsstrukturdiagramm.	312
3.7.1	Komposition.	313
3.7.2	Klassifikator	325
3.8	Komponentendiagramm mit Eclipse Papyrus	326
3.8.1	Praxisbeispiel: Abhängigkeit zwischen Komponenten und Interface.	326
3.8.2	Kapselung von Zustand und Verhalten.	335
3.9	Zusammenfassung	336
	Literatur.	341
4	SysML-Modellierung mit Eclipse Papyrus.	343
4.1	Blockdefinitionsdiagramm.	345
4.1.1	Aufbau von Blockdefinitionsdiagrammen	345
4.1.2	Erstellung von Blockdefinitionsdiagrammen (BDD) mit Eclipse Papyrus	345
4.1.2.1	Praxisbeispiel: Modellierung der Schaltung vom Schwingkreiswechselrichter	347
4.1.2.2	Überblicke über Merkmale der Blöcke zur SysML-Modellierung	361
4.2	Internes Blockdiagramm (IBD)	363
4.2.1	Modellierung der Schaltung eines Blindleistungsstromrichters mit IBD.	364
4.2.1.1	Modellierung der Funktionalität des Blindleistungsstromrichters mithilfe von SysML-Informationsobjektflüssen	364

4.2.1.2	Modellierung der Schaltung des Blindleistungsstromrichters mithilfe der SysML-Objektflussesports	371
4.2.2	Modellierung der Leistungssteuerung durch die Spannungsverstellung bei Schwingkreiswechselrichtern mit IBD.	372
4.3	Anforderungsdiagramm	378
4.3.1	Anforderungsdiagramm des Schwingkreiswechselrichters mit Papyrus.	379
4.3.2	Anforderungstabelle des Solar-Schwingkreiswechselrichters mit Papyrus.	385
4.4	Zusicherungsdiagramm (Parametrisierdiagramm).	393
4.4.1	Modellierung von Verlusten für „Insulate Gate Bipolar Transistor“ (IGBT) mithilfe der Sicherungsdiagramme auf Basis von Eclipse Papyrus SysML.	394
4.4.2	Modellierung von Blindleistungen mit Zusicherungsdiagrammen	396
4.5	Zusammenfassung	398
	Literatur.	401
5	Parallele Modellierung mit Obeo UML Designer	403
5.1	Modellierung mit Zustandsdiagrammen	403
5.1.1	Horizontale Modellierung	404
5.1.2	Vertikale Modellierung.	405
5.2	Modellierung mit Aktivitätsdiagrammen.	407
5.2.1	Vertikale Integration	414
5.2.2	Horizontale Integration.	416
5.3	Modellierung mit Klassendiagrammen	418
5.3.1	Vererbungshierarchie	418
5.3.2	Das Modellieren der Strukturen der Klassen	419
5.3.3	Parallelisierung der objektorientierten Modellierung.	421
5.4	Modellierung mit Sequenzdiagrammen.	422
5.4.1	Darstellung der parallelen Prozesse mit Objekten oder Lebenslinien	423
5.4.2	Darstellung der parallelen Prozesse mit Interaktionen.	425
5.5	Zusammenfassung	426
	Literatur.	429
6	Vom Modellieren zum Programmieren	431
6.1	Anwendung von Java Swing in der Entwicklung der grafischen Oberfläche	431
6.2	Design Pattern Interface.	442
6.2.1	Implementierung der Berechnungen mithilfe eines Interface	442

6.2.2	Implementierung der Kapselung der Daten mithilfe der Interfaces und Record-Klassen.	446
6.2.3	Implementierung der Parallelisierung einer Anwendung mithilfe der Interfaces und Record-Klassen.	450
6.2.4	Implementierung von Session-Beans mithilfe eines Remote-Interface	458
6.2.5	Implementierung von einem <i>Sealed</i> -Interface durch <i>Permits</i> -Klassen	463
6.2.6	Implementierung von Interfaces mithilfe des Pattern-Matchings in Switch-Ausdrücken	468
6.3	Codes aus Java 19	475
6.3.1	Codes für die neue Features mit Java 19	475
6.3.1.1	Das Überladen von Methoden mithilfe des <i>Pattern-Matchings</i> für <i>Switch</i> -Ausdrücke und <i>InstanceOf</i>	475
6.3.1.2	Das Überladen von Methoden mithilfe des <i>Structured-Concurrency-Patterns</i>	488
6.3.2	Codegenerierung aus Ecore-Modellen	494
6.4	Zusammenfassung	533
	Literatur	535
	Stichwortverzeichnis	537