

**Tim Weilkiens • Alexander Huwaldt • Jürgen Mottok •
Stephan Roth • Andreas Willert**

Modellbasierte Softwareentwicklung für eingebettete Systeme verstehen und anwenden



dpunkt.verlag

Inhaltsverzeichnis

1	Einleitung	1
1.1	Warum gerade jetzt dieses Buch?	1
1.2	Wie sollte man dieses Buch lesen?	3
1.3	Was ist an eingebetteten Systemen so besonders?	4
1.4	Wie sieht das typische Zielsystem aus?	5
1.5	Fallbeispiele	7
1.5.1	Die Anlagensteuerung	7
1.5.2	Die Baumaschine	8
1.5.3	Das Energie-Monitoring-System	9
1.5.4	Das Beispielsystem SimLine	9
1.6	Das Manifest	10
2	Basiswissen	13
2.1	Was sind eingebettete Systeme?	13
2.2	Software Engineering für eingebettete Systeme	16
2.3	Der Qualitätsbegriff	19
2.3.1	Externe Qualität	19
2.3.2	Interne Qualität	19
2.3.3	Qualitätskriterien nach ISO/IEC 25010	20
2.4	Einführung in Model-Driven Software Engineering	23
2.5	Komplexität	24
2.6	Unser Gehirn als Engpass	26
2.7	Vorgehensweisen und Techniken, um einer steigenden Komplexität zu begegnen	27
2.8	Komplexen Systemen lässt sich nicht mit simplen Methoden begegnen	28

2.9	Verstehbarkeit und Redundanz	29
2.10	Was ist ein Modell?	33
2.11	Modelle helfen, die Zukunft vorwegzunehmen?	36
2.12	Modelle helfen zu abstrahieren?	38
2.13	Resümee: Nutzen von MDSE?	40
3	Modellbasierte Softwareprozesse und Toollandschaften	43
3.1	Klassifikation von Modellierungswerkzeugen	43
3.1.1	Modellierungswerkzeuge der Kategorie A (universelle Malwerkzeuge)	47
3.1.2	Modellierungswerkzeuge der Kategorie B (spezialisierte Malwerkzeuge)	47
3.1.3	Modellierungswerkzeuge der Kategorie C (einfache Modellierungswerkzeuge mit Modelldatenbank) ..	47
3.1.4	Modellierungswerkzeuge der Kategorie D (vollständiger Zyklus einer Embedded-Software- Engineering-Werkzeugkette)	48
3.2	Vorgehensmodelle	48
3.2.1	SYSMOD Zickzack trifft auf IBM Rational Harmony	49
3.2.2	Spezifikation oder Lösung?	50
3.2.3	Wiederverwendung	52
3.3	Best Practice für kleine Teams	53
4	Modellbasiertes Requirements Engineering	61
4.1	Requirements Engineering	61
4.2	Anforderungen in der Modellierung	62
4.3	Anforderungen und Architektur im Zickzack	64
4.4	Szenario: Modellbasierte Spezifikation mit UML erstellen	66
4.4.1	Überblick über Methode	66
4.4.2	Problemanalyse	66
4.4.3	Basisarchitektur	68
4.4.4	Systemidee und Systemziele	69
4.4.5	Stakeholder und Anforderungen	70
4.4.6	Systemkontext	71
4.4.7	Anwendungsfälle und Aktivitäten	72
4.4.8	Fachwissen	75
4.4.9	Szenarien	76
4.5	Mehr Modellierung: Ausführbare Spezifikation	77
4.6	Weniger Modellierung: Diagramme für Anforderungsdokumente ...	80

4.7	Neuentwicklung versus Weiterentwicklung	80
4.7.1	Basisarchitektur	81
4.7.2	Anwendungsfälle	81
4.7.3	Szenarien	81
5	Modellbasierte Architekturbeschreibung	83
5.1	Architektur – Was ist das eigentlich?	83
5.2	Die technische Softwarearchitektur	85
5.3	Architekturmuster und deren Bedeutung	88
5.4	Das Laufzeitsystem als Basismuster in eingebetteten Applikationen	90
5.5	Referenzmuster für eine Laufzeitarchitektur	94
5.5.1	Sensorik, Einlesen, Filtern, Bewerten	96
5.5.2	Transformation der Sensorik in Aktivitäten (Verarbeitung)	97
5.5.3	Ausgabe der Daten und Ansteuerung der Aktoren	97
5.6	Fachliche Architektur	97
5.7	Architektur-Assessment	98
6	Modellbasiertes Softwaredesign	103
6.1	Gesichtspunkte der fachlichen Architektur und des Designs	103
6.2	Hierarchische Dekomposition	105
6.3	Diskretisierungs- und Laufzeiteffekte im Design	115
6.4	Softwaredesignprinzipien	118
6.4.1	Was ist ein Prinzip?	118
6.4.2	Grundlegende Designprinzipien	119
6.4.3	Designprinzipien in der Modellierung	123
6.5	Hardwareabstraktion	123
6.5.1	Ausgangssituation	124
6.5.2	Evolution der Mikrocontrollerprogrammierung in C	124
6.5.3	Die klassische Vorgehensweise mit der UML	127
6.5.4	Die graue Theorie	130
6.5.5	Lösungsansätze für die Modellierung	133
6.5.6	Lösungsansätze für die Codegenerierung	136
7	Modellbasiertes Testen	145
7.1	Warum eigentlich testen?	145
7.2	Nicht nur sicherstellen, dass es funktioniert	146
7.2.1	Ein angstfreies Refactoring ermöglichen	146
7.2.2	Besseres Softwaredesign	146
7.2.3	Ausführbare Dokumentation	147
7.2.4	Tests helfen, Entwicklungskosten zu sparen	147

7.3	Die Testpyramide	148
7.4	Test-Driven Development (TDD)	151
7.4.1	Viel älter als vermutet: Test First!	151
7.4.2	TDD: Red – Green – Refactor	153
7.5	Model-Based Testing (MBT)	154
7.6	UML Testing Profile (UTP)	155
7.7	Ein praktisches Beispiel	156
7.8	Werkzeuge, die dieses Vorgehen unterstützen	163
8	Integration von Werkzeugen	171
8.1	Anforderungen an Schnittstellen zu Werkzeugen unterschiedlicher Disziplinen	172
8.1.1	Digital Twin	172
8.1.2	Traceability aus Safety-Gesichtspunkten	173
8.1.3	Projekt- und Workload-Management	173
8.2	Synchronisation der Daten zwischen Repositories	175
8.3	Zentrales Repository	176
8.4	Ein Werkzeug für alles	178
8.5	OSLC – Open Services for Lifecycle Collaboration	179
8.6	XMI – Austausch von Daten innerhalb der Fachdomäne MDSE ...	181
8.7	ReqIF zum Austausch von Anforderungen	181
8.8	FMI (Functional Mock-up Interface)	182
8.9	SysML Extension for Physical Interaction and Signal Flow Simulation (SysPhS)	183
8.10	AUTOSAR	184
8.11	Stand der Praxis	184
9	Modellbasierte funktionale Sicherheit	187
9.1	Funktionale Sicherheit	187
9.2	Entwurfsmuster der funktionalen Sicherheit	190
9.2.1	Zufällige und systematische Fehler	191
9.2.2	Symmetrische und diversitäre Redundanz	193
9.2.3	Architekturmuster	195
9.2.4	Monitor-Actuator Pattern (MAP)	198
9.2.5	Homogenous Redundancy Pattern (HRP)	200
9.2.6	Triple Modular Redundancy Pattern (TMR)	202
9.2.7	SIL-Empfehlung für die Safety and Reliability Design Patterns	206

9.3	Vom Modell zum sicheren Quellcode: Coding-Standards für die sichere Programmierung	207
9.3.1	Normativer Hintergrund	207
9.3.2	MISRA-C und MISRA-C++	208
9.3.3	Prüfwerkzeuge	209
9.3.4	Codegenerierung	209
9.4	Das Safety and Reliability Profile der UML	210
9.5	Praktische Anwendung der Modellierung im Kontext sicherheitskritischer Systeme	211
9.5.1	Beweis der Korrektheit der Transformation	212
9.5.2	Verifikation der Qualität des Werkzeugs	213
9.5.3	Zertifiziertes Framework	213
9.6	Vorteile der modellgetriebenen Entwicklung im Safety-Kontext	214
9.6.1	Traceability	214
9.6.2	Semiformale Spezifikation	215
9.6.3	Normen empfehlen den Einsatz von formalen und/oder semiformalen Methoden und Notationen	216
9.6.4	Automatisierte Transformationen	216
9.6.5	Modellierungsrichtlinien (Guidelines)	217
9.6.6	Dokumentation	218
9.7	Modellgetriebene Entwicklung mit der UML	218
10	Metamodellierung	221
10.1	Modell und Metamodell	221
10.2	UML-Metamodelle	223
10.3	EAST-ADL	225
10.4	AADL	230
10.5	Vergleich EAST-ADL und AADL	232
11	Einführung eines modellbasierten Ansatzes in einer Organisation	235
11.1	Ausgangslage	236
11.2	Vorgehen	237
11.2.1	Problem analysieren	237
11.2.2	Idee und Ziele des Vorgehens	238
11.2.3	Stakeholder und Anforderungen	239
11.2.4	Methodikkontext	240
11.2.5	Anwendungsfälle	240
11.2.6	Fachwissenmodell	242
11.2.7	Verifikation und Validierung	242
11.3	Auswahl der Modellierungssprachen	243

11.4	Auswahl der Modellierungswerkzeuge	244
11.5	Typische Fehler	245
11.5.1	Schnellstart	245
11.5.2	Übergewicht	245
11.5.3	Einsame Insel	246
11.5.4	Elfenbeinturm	246
11.5.5	Aus der Lernkurve fliegen	246
12	Lebenslanges Lernen	249
12.1	Lernen – die Sicht des Konstruktivismus	249
12.2	Kompetenzen – der Blick auf die modellbasierte Softwareentwicklung	251
12.3	Agilität – Lernen mit Methoden und Praktiken	254
12.4	Psychologische Grundlagen von Fehlern	261
12.4.1	Denkfallen als Fehlerursache	261
12.5	Software Craftsmanship – ein Beispiel	264
12.6	Modellierungskultur – ein Kodex	266
12.6.1	Manifest – Modeling of Embedded Systems	266
12.6.2	Big Picture – der Blick auf den Kodex einer Modellierungskultur	266
12.6.3	Moderation – die konstruktive Kommunikation	267
12.6.4	Reflexion – Lernen durch Reflektieren	268
12.6.5	Selbstverpflichtung – selbstgesteuertes Lernen als Entwickler und als Team	269
12.6.6	Teamradar – ein Feedbackbarometer	270
12.6.7	Experten als Teamcoach – Agenten der Veränderung	272
12.6.8	Funktionale Sicherheit – ein Beitrag der normativen Sicherheitskultur	272
13	Fazit	277
	Anhang	
A	Ausblick: Skizze eines Reifegradmodells für MDSE	281
A.1	Hintergrund und Motivation	281
A.2	Die Skizze als ein Start – Diskussionsforum	283
A.3	Ausgangslage Manifest – Ziele kompakt	284
A.4	Die Reifegrade – der Weg in die Modellierungskultur	285
A.5	Evaluation und Fragenkatalog	292

B	Kurzreferenz UML und SysML	295
B.1	Eine kurze Geschichte der UML	296
B.2	Aufbau und Architektur der UML	297
B.3	Anwendungsfalldiagramm	299
B.3.1	Akteur	299
B.3.2	Anwendungsfall	300
B.3.3	Anwendungsfallbeziehungen	300
B.4	Aktivitätsdiagramm	300
B.4.1	Aktivität und Aktivitätsparameter	301
B.4.2	Aktion und Pin	302
B.4.3	Kontroll- und Objektfluss	303
B.4.4	Start- und Endknoten	304
B.4.5	Entscheidung und Zusammenführung	304
B.4.6	Splitting und Synchronisation	304
B.5	Klassendiagramm	305
B.5.1	Klasse und Objekt	305
B.5.2	Attribut	306
B.5.3	Operation	306
B.5.4	Assoziation	307
B.5.5	Komposition	308
B.5.6	Generalisierung	308
B.5.7	Signal	308
B.5.8	Datentyp	309
B.5.9	Templates	309
B.6	Kompositionsstrukturdiagramm	310
B.6.1	Konnektor	311
B.6.2	Port	312
B.7	Sequenzdiagramm	314
B.7.1	Interaktion	314
B.7.2	Lebenslinie	315
B.7.3	Nachricht	316
B.7.4	Kombiniertes Fragment	316
B.7.5	Zeitliche Zusicherungen	317
B.8	Zustandsdiagramm	317
B.8.1	Zustandsautomat	318
B.8.2	Zustand	319
B.8.3	Transition	319
B.8.4	Start- und Endzustand	320
B.8.5	Pseudozustand	321

B.9	Paketdiagramm	322
B.9.1	Paket und Modell	322
B.9.2	Pakete importieren	323
B.9.3	Modellbibliothek	323
B.10	Querschnittselemente	323
B.10.1	Kommentar	323
B.10.2	Zusicherung	324
B.10.3	Trace-Beziehung	324
B.11	Profil	324
B.11.1	SysML	326
B.11.2	MARTE	329
B.11.3	UML Testing Profile (UTP)	331
B.11.4	MDESE-Profil (basierend auf SYSMOD-Profil)	332
C	Glossar	335
D	Literaturverzeichnis	355
	Stichwortverzeichnis	363