→ 3., überarbeitete und aktualisierte Auflage

Tim Weilkiens

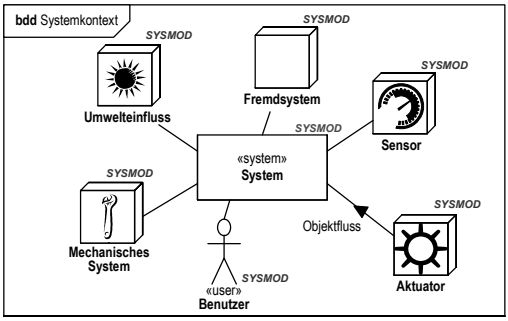
Systems Engineering mit SysML/UML

Anforderungen, Analyse, Architektur

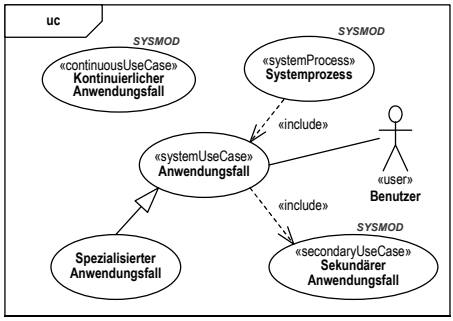
→ Mit einem Geleitwort von Richard Mark Soley

dpunkt.verlag

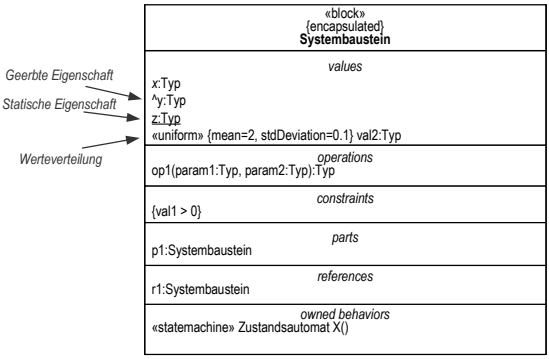
Systemkontextdiagramm

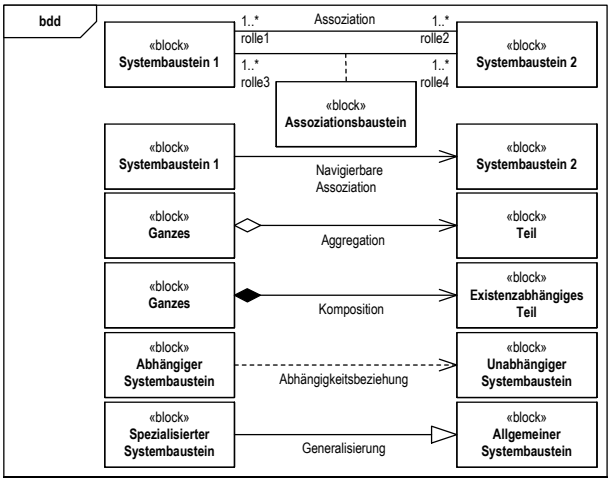


Anwendungsfalldiagramm

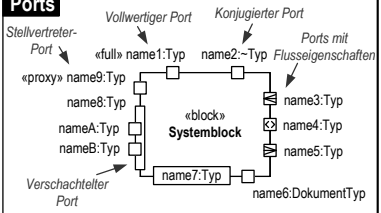


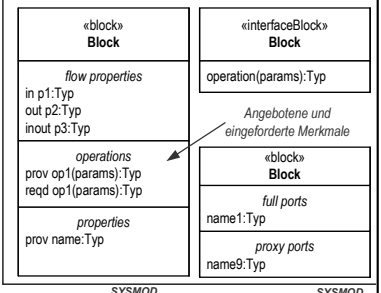
Blockdefinitionsdiagramm

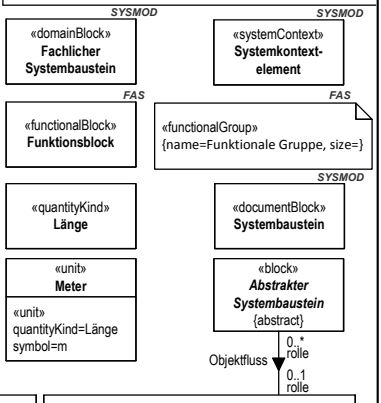


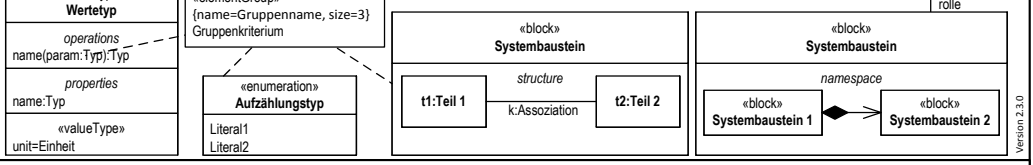


Ports





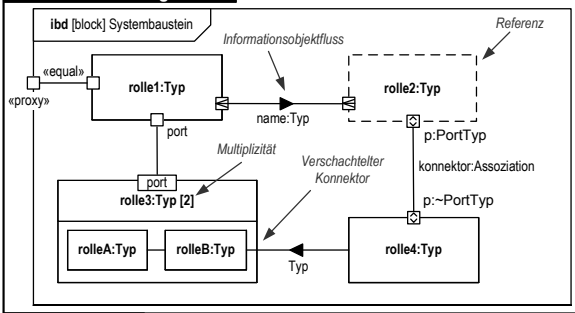


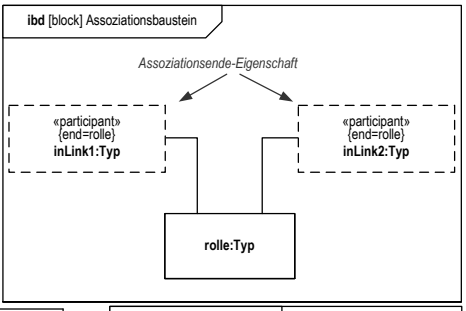


OMG Systems Modeling Language (OMG SysML™) 1.4 *(http://www.omgsysml.org)*
inkl. Stereotypen aus SYSMOD *(http://www.sysmod.de)* **und FAS** *(http://www.fas-method.org)*
MBSE Blog: *http://www.model-based-systems-engineering.com*

© 2014 by Tim Weilkiens

Internes Blockdiagramm





Anforderungen

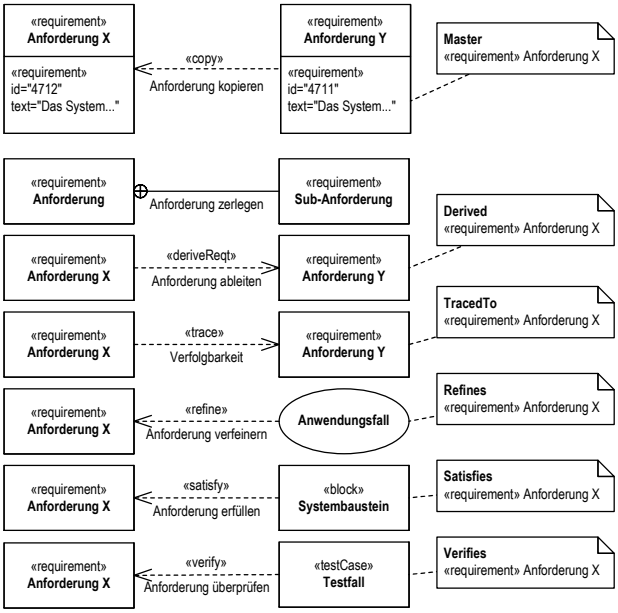
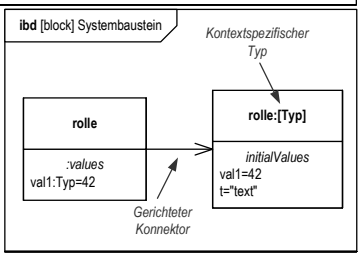
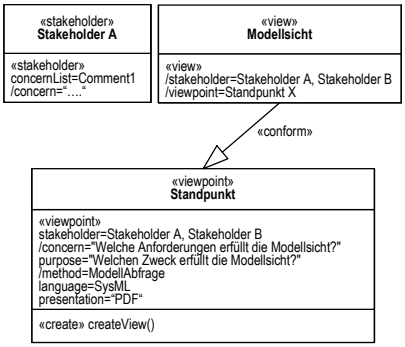


table Anforderungen in Tabellennotation

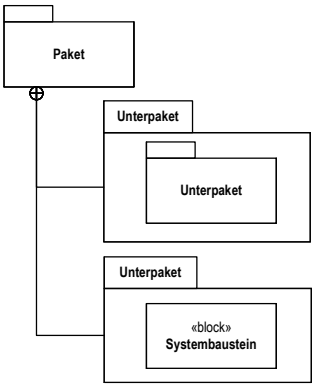
ID	Name	Text
4711	Anforderung	Das System....
...	...	...



Modellsicht



Pakete



Version 2.3.0

© 2014 by Tim Weilkiens

Tim Weilkiens ist Geschäftsführer der Beratungsfirma oose Innovative Informatik GmbH. Seine thematischen Schwerpunkte sind die Modellierung und Entwicklungsprozesse für Systeme. Er ist für oose GmbH Repräsentant bei der OMG, ist u. a. Koautor der SysML-Spezifikation und Koentwickler des Zertifizierungsprogramms OMG Certified Systems Modeling Professional (OCSMP).

Sie erreichen ihn unter tim@larus.de.

Papier plus+ PDF.

Zu diesem Buch – sowie zu vielen weiteren dpunkt.büchern – können Sie auch das entsprechende E-Book im PDF-Format herunterladen. Werden Sie dazu einfach Mitglied bei dpunkt.plus+:

www.dpunkt.de/plus

Tim Weilkiens

Systems Engineering mit SysML/UML

Anforderungen, Analyse, Architektur

3., überarbeitete und aktualisierte Auflage

dpunkt.verlag

Tim Weilkiens
tim@larus.de

Lektorat: Christa Preisendanz
Copy Editing: Ursula Zimpfer, Herrenberg
Satz: Tim Weilkiens
Herstellung: Frank Heidt
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

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.

ISBN
Buch 978-3-86490-091-4
PDF 978-3-86491-543-7
ePub 978-3-86491-544-4

3. Auflage 2014
Copyright © 2014 dpunkt.verlag GmbH
Wieblinger Weg 17
69123 Heidelberg

Copyright-Hinweis für den Verlag
Der Inhalt, die Notationsübersichten für die Umschlaginnenseiten und das Glossar ist weitestgehend durch Mitarbeiter der oose GmbH während dessen bezahlter Arbeitszeit entstanden oder basiert auf anderen Materialien der oose GmbH und sind somit Eigentum der oose GmbH. Ich erteile jedoch als Geschäftsführer der oose GmbH dem dpunkt-Verlag hiermit das unbegrenzte und nicht-exklusive Nutzungsrecht für die Verwendung im nachfolgenden Buch.
Tim Weilkiens

Copyright-Hinweis für das Buch
BPMN, ReqIF, OMG SysML, OMG Systems Modeling Language, the OMG SysML logo, OCSMP, UML, Unified Modeling Language and the UML logo are trademarks or registered trademarks of the Object Management Group, Inc. in the United States, the European Union and other countries.

Die vorliegende Publikation ist urheberrechtlich geschützt. Alle Rechte vorbehalten. Die Verwendung der Texte und Abbildungen, auch auszugsweise, ist ohne die schriftliche Zustimmung des Verlags urheberrechtswidrig und daher strafbar. Dies gilt insbesondere für die Vervielfältigung, Übersetzung oder die Verwendung in elektronischen Systemen.
Es wird darauf hingewiesen, dass die im Buch verwendeten Soft- und Hardware-Bezeichnungen sowie Markennamen und Produktbezeichnungen der jeweiligen Firmen im Allgemeinen warenzeichen-, marken- oder patentrechtlichem Schutz unterliegen.
Alle Angaben und Programme in diesem Buch wurden mit größter Sorgfalt kontrolliert. Weder Autor noch Verlag können jedoch für Schäden haftbar gemacht werden, die in Zusammenhang mit der Verwendung dieses Buches stehen.
5 4 3 2 1 0

Vorwort zur 3. Auflage

Seit der 2. Auflage sind nun 6 Jahre vergangen. In dieser Zeit hat sich die Sprache SysML, das Vorgehen SYSMOD und die MBSE-Disziplin (*Model Based Systems Engineering*) weiterentwickelt. Das Thema unterliegt einer hohen Dynamik, der das Medium Buch nur schwer folgen kann. Das war für mich der Grund, vor ein paar Jahren den MBSE-Blog unter `www.model-based-systems-engineering.com` zu starten. Neue Erkenntnisse und Weiterentwicklungen kann ich dort zeitnah publizieren. Jetzt ist aber auch ein guter Zeitpunkt, den aktuellen Stand kompakt und zusammenhängend in diesem Buch festzuhalten. Die wesentlichen Änderungen der 3. Auflage sind:

- Das Buch basiert auf der aktuellen SysML-Version 1.4, die einige Neuerung mitbringt. Ebenso enthält es auch die Elemente der Vorgängerversion 1.3, die es zum Zeitpunkt der 2. Auflage noch nicht gegeben hat.
- SYSMOD adressiert jetzt explizit die Architekturtypen: Basisarchitektur, logische Architektur, physische Produktarchitektur und funktionale Architektur.
- Diverse kleinere Anpassungen von SYSMOD, die auf Erfahrungen und Feedbacks aus der Praxis basieren, sind enthalten.
- Ein neues Kapitel zur Vorbereitung auf die OCSMP-Zertifizierung der OMG ist hinzugekommen.
- Feedbacks, die ich zur 2. Auflage erhalten habe, sind eingearbeitet. Vielen Dank an alle diejenigen, die mir eine Rückmeldung gegeben haben.

Ich wünsche Ihnen viel Freude und Erkenntnisse beim Lesen und freue mich auf Ihre Rückmeldungen.

In einem fliegenden Verband namens Fairchild Dornier 328-110
irgendwo über Deutschland.
April 2014, Tim Weilkiens

Vorwort zur 2. Auflage

Im April 2006 ist die *OMG Systems Modeling Language* (OMG SysML™) veröffentlicht und im September 2007 offiziell von der *Object Management Group* (OMG) verabschiedet worden.

Die Erstauflage dieses Buchs ist kurz nach der Veröffentlichung der SysML erschienen. Anfang 2008 habe ich eine aktualisierte Version des Buchs auf Englisch beim Morgan Kaufmann Verlag in der OMGPress publiziert. Bis heute ist es weltweit das einzige Buch über SysML. So sehr mir das Monopol natürlich zugute kommt, so sehr wünsche ich den Anwendern aber auch die Vielfalt durch weitere SysML-Bücher. Nur so kann sich die Sprache erfolgreich entfalten.

Es gibt bereits viele Projekte, die SysML verwenden bzw. den Einsatz der Sprache planen und vorbereiten. Erfahrungen und Best Practices, die ich als Berater in diesen Systems-Engineering-Projekten gesammelt habe, sind Teil der Neuerungen in dieser 2. Auflage des Buchs. Während der Arbeit an der Neuauflage arbeitete ich auch in der *SysML Revision Task Force* der OMG an der nächsten SysML Version 1.1. Sie wird Anfang 2009 erscheinen. Aus dieser Arbeit stammen ebenfalls ganz aktuelle Informationen, die Sie in diesem Buch finden.

Konkrete Themen aus Projekten die SysML betreffen und die ich in der neuen Auflage des Buchs aufgenommen bzw. vertieft habe, sind Heuristiken zur Anforderungsmodellierung (Abschnitt 2.5.2), Variantenmodellierung (Abschnitt 2.10.2), Modellmanagement (Abschnitt 2.10.1) und die Vorstellung eines Intensitätsmodells (Abschnitt 2.10.7).

Es gibt aus dieser Arbeit auch Themen, die ich hier nicht oder nur am Rande erwähne, da sie den Rahmen des Buchs sprengen würden oder da sie noch zu unreif für eine Publikation dieser Art sind. Dazu zählt beispielsweise die Integration von Modellen. Ebenso wie das Systems Engineering eine Querschnittsfunktionalität über alle Disziplinen des Projekts bildet, hat auch das SysML-Modell diese Rolle inne. Es beschreibt die ganzheitliche Systemarchitektur und hat zur Aufgabe, die Modelle der einzelnen

Disziplinen wie Software und Mechanik zu integrieren. Teil dieser Thematik sind Datenstandardformate wie XMI und ISO AP-233.

Weitere Themen sind die Anwendung des Zusicherungsdiagramms und modellbasierte Simulationen.

Bei dem Aufsehen, für das SysML bisher gesorgt hat, darf nicht vergessen werden, dass SysML nur eine Sprache ist. Sie ist nur ein Baustein in der Systementwicklung und nutzlos, wenn ein entsprechendes Vorgehen fehlt. Insbesondere der Mensch selbst ist ein wesentlicher Faktor für den Projekterfolg (Stichwort: Soft Skills).

Ich habe SysML oft als Türöffner für Systems-Engineering-Prozesse erlebt. Projekte, die SysML als direkten Problemlöser gesehen haben, sind über die ersten Gehversuche mit der Sprache auf mangelhafte oder fehlende disziplinenübergreifende Prozesse und Projektstrukturen aufmerksam geworden. Da lagen die eigentlichen Probleme. Die Folge bzw. ist die Einführung des Systems Engineering als explizite Disziplin im Projekt oder Unternehmen. Die ersten positiven Synergieeffekte traten dann schnell auf.

Der Erfolg der SysML zeigt, dass hier der richtige Weg eingeschlagen worden ist. Wir stehen erst am Anfang des Weges: *»This is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.«(Winston Churchill)*

In einem fliegenden Verband namens A300
irgendwo über Deutschland.
Juni 2008, Tim Weilkiens

Geleitwort von Richard Mark Soley

Laut Flugzeugbauer Boeing hat das Flugzeug vom Typ 747-400 ein maximales Bruttostartgewicht von fast 400.000 kg (einschließlich einer durchschnittlichen Anzahl von 416 Passagieren, einer Frachtladung von 171 Kubikmetern und über 200.000 kg Sprit). Vier Riesenmotoren treiben den Vogel mit bis zu 88% Schallgeschwindigkeit über unglaubliche Entfernungen – bis zu 13.500 km – an, ohne nachtanken zu müssen. Allein die Länge des Flugzeugs (45 m) ist länger als der gesamte Erstflug der Gebrüder Wright.

Doch diese erstaunlichen Zahlen, die nach 30 Jahren durch noch größere Passagierflugzeuge in den Schatten gestellt werden, sind nichts im Vergleich zur Komplexität des Systems, bestehend wiederum aus Systemen (*System of Systems*), aus denen sich die Boeing 747-400 zusammensetzt. Das Elektrosystem des Flugzeugs umfasst etwa 274 km Kabel. Die hochfesten Aluminium- und Titanteile arbeiten sowohl im Stillstand als auch unter Hitzeeinwirkung durch das enorm schnelle Durchströmen der Luft. Backup-Systeme halten die Navigation und lebenswichtige Systeme in Gang für den Fall, dass das Hauptsystem ausfällt. Sogar das Unterhaltungselektroniknetz in der Kabine gehört zu den komplexeren Systemen auf der Welt. Die Boeing 747-400 umfasst etwa sechs Millionen Teile, wovon fast die Hälfte einfache Befestigungselemente wie Schrauben und Muttern sind. Kein Wunder, dass ein Boeing-Sprecher einmal witzelte: »Wir betrachten eine 777 als Sammlung von Teilen, die im geschlossenen Verband fliegen.« Als Vielflieger hoffe ich inständig, dass dieser »geschlossene Verband« eingehalten wird!

Alle diese Fakten spiegeln eine Tatsache wider, die jeder Ingenieur kennt: Komplexität ist schwierig zu handhaben, und die meisten interessanten Systeme sind komplex. Noch schlimmer: Die Zusammensetzung von Systemen zu Systemen, die wiederum aus Systemen bestehen, z. B. die Elektro-, Hydraulik-, Antriebs- (Hub-, Schub- und Dauerbetrieb), Navigations- und weiteren Systeme des Flugzeugs in unserer Analogie, tendieren dazu, neue

Komplexitäten in Form unerwarteter Teileüberlappung und unerwarteten Verhaltens von zuvor sich wohlverhaltenden Systemen einzuführen.

Als Folge der steigenden Komplexität von Systemen gibt es einen dringenden Bedarf, Komponentensysteme so beschreiben zu können, dass sich die Designs leicht unter Designerteams austauschen lassen. Eine gemeinsame Sprache ist eine entscheidende Komponente der Design-Methodologie eines jeden Systems oder Prozesses. Dabei spielt es keine Rolle, ob es sich um ein elektrisches, mechanisches oder chemisches System oder um Software handelt. Und wenn diese gemeinsame Sprache auf einer bereits existierenden basiert, wird das Team, das die Designs gemeinsam nutzt, rascher in der Lage sein, die Designaufgaben – und letztlich die noch anspruchsvolleren, langfristigen Wartungs- und Integrationsaufgaben – besser, schneller und billiger zu bewältigen.

Dieser Denkprozess führte die Object Management Group (OMG) mit ihren Hunderten von Mitgliedsfirmen aus aller Welt zur Entwicklung und Pflege der Systems Modeling Language (SysML). Der offene Standard weist eine Reihe positiver Merkmale auf:

- Die Sprache basiert auf einer Erweiterung der OMG-eigenen Unified Modeling Language (UML). Softwareentwickler, die mit UML vertraut sind, können daher leicht auf SysML umsteigen, und Werkzeugentwickler, die mit UML-Werkzeugen arbeiten, können SysML problemlos unterstützen.
- Die Sprache ist grafisch. Mich verblüfft es immer wieder, dass Softwareentwickler ihr Design mithilfe von Grafikelementen austauschen, dann aber z. B. »C++-Code« schreiben, um dieses Design auf den Computer zu übertragen. Ähnlich arbeiten Fertigungstechniker die Einzelheiten einer Produktfamilie mithilfe von Graphen, Kästen und Linien aus, um sie dann letztlich in Textform zu umreißen, z. B. mit PDES/STEP. Grafische Sprachen eignen sich naturgemäß für Designs und werden seit Tausenden von Jahren benutzt, wie etwa alte Gebäude- und Schiffsblaupausen, elektrische Schaltbilder usw. belegen.
- Die Sprache hat viele Implementierungen. Internationale Standards sind einfach nicht das Papier wert, auf dem sie gedruckt werden (oder den Festplattenplatz, den sie belegen), falls man sie nicht implementiert. Noch bevor SysML Anfang 2006 in der endgültigen Spezifikation verfügbar war,

kündigten bereits mehrere Firmen die Implementierung des Standards an.

Wichtig ist dabei, dass es sich bei SysML nicht einfach nur um eine weitere Modellierungssprache für die Softwareentwicklung handelt. Software ist heute in nahezu allen komplexen Systemen zwar eine wichtige Komponente, aber fast nie die einzige. Flugzeuge, Züge und Autos beinhalten Software ebenso wie Gebäuderegelungseinrichtungen oder Chemiewerke. Sie alle umfassen aber auch u. a. Haustechnik-, Hydraulik- und Elektrosysteme, die es aufeinander abzustimmen und übergreifend zu entwerfen gilt. Ebenso müssen die komplexen Systeme, die von Menschen und automatisierten Prozessen genutzt werden sollen, auch entsprechend geplant werden. Nur so lassen sie sich sachgerecht und sinnvoll nutzen (sowie warten und im Hinblick auf neue Anforderungen in künftige Systeme integrieren). Bei all diesen designübergreifenden Fragen muss berücksichtigt werden, dass alle komplexen Systeme in mehreren Konfigurationen verfügbar sind, je nach Anforderung des Einsatzumfelds, der Mitarbeiterqualifikation und zahlreicher weiterer Variablen.

Das ist der Fokus von SysML, einer Sprache zur Modellierung all dieser unterschiedlichen Faktoren im Verlauf der Entwicklung komplexer Systeme und aus Systemen bestehender Systeme. Der Zweck ist dabei nicht das Verbergen der Komplexität jener Systeme, sondern vielmehr ihre Offenlegung und Kontrolle angesichts sich ständig ändernder Geschäftsanforderungen und Infrastrukturen.

SysML wird wahrscheinlich nicht die letzte Sprache für die Entwicklung komplexer Systeme sein. Durch den schnellen Wandel in den meisten Entwicklungs- und Konstruktionsbereichen (und der damit einhergehenden Einführung völlig neuer Bereiche, wie z. B. der Biotechnik) dürfte sich auch die Spezifikationssprache selbst ändern. Für einen Ingenieur sollte das keine große Überraschung sein, denn viele langlebige Spezifikationssprachen haben sich im Laufe der Zeit gewandelt. Ein einfaches Beispiel: Gebäudebaupläne gibt es schon seit mehreren Hundert Jahren, diejenigen aus der Zeit bis etwa 1880 hatten aber kein internationales Symbol für elektrische Anschlüsse, da dieses Merkmal damals noch nicht existierte. Sprachen ändern sich, um sich an geänderte Konstruktionsanforderungen anzupassen. Vorteilhaft bei SysML ist, dass die Sprache auf einer Metamodell-Infrastruktur basiert – der Meta Object Facility (MOF), die ebenfalls von der OMG standardisiert wurde – und folglich für künftige

Änderungen ausgelegt wurde. Die Technologie kann auch benutzt werden, um bestehende Designs, die in älteren Modellierungssprachen, wie etwa IDEF (eine häufig in der Luftfahrt und vielen anderen Bereichen verwendete Sprache), geschrieben wurden, umzuwandeln und zu integrieren.

Womit sich der Kreis schließt: Wir sind wieder angelangt bei der Sammlung von sechs Millionen Teilen, die im geschlossenen Verband fliegen. Die Aussicht, eine derart große Teilesammlung mit dem entsprechenden Bedarf an Menschen und Einrichtungen, Hardware und Software, Hydraulik, Elektrik und Navigation ohne eine gemeinsame Designsprache entwickeln zu können, die den konstanten Wandel von Komponentensystemen berücksichtigt, ist gleich null. SysML wurde eigens für diesen Bedarf ausgelegt. Die Sprache bewährt sich bereits in der Praxis für winzige, eingebettete Controller für Robotersysteme bis hin zu riesigen Industriekomponenten.

Lernen Sie also die Sprache und haben Sie Spaß dabei! Sie ist zwar nicht ganz einfach – komplexe Probleme bedürfen eben komplexer Lösungen –, aber logisch und beruht auf guten Designpraktiken. Und sorgen Sie letztlich dafür, dass jene Teile im geschlossenen Verband fliegen![1]

Richard Mark Soley, Ph.D.
Chairman and Chief Executive Officer
Object Management Group, Inc.
30. Mai 2006

Zusatz zur 3. Auflage

Jede Metrik darüber, wie gut ein Standard angenommen wird, muss nicht nur berücksichtigen, wie viele Anwender mit dem Standard arbeiten (die Anwender-Community), sondern auch wie enthusiastisch die Community ist, die den Standard entwickelt. SysML ist erfolgreich in beiden Metriken. Es gibt regelmäßige Aktualisierungen des Standards, die die Bedürfnisse und Anregungen von Tausenden von Anwendern aus aller Welt berücksichtigen. Der Wälzer, den Sie gerade in Ihren Händen halten, deckt die Änderungen der SysML bis zur Version 1.4 ab sowie Änderungen an der SYSMOD-Methodik des Autors zur Entwicklung realer, komplexer Systeme (und denken Sie daran, trotz

[1] Das englische Original ist auf der englischen Webseite des Buchs *www.model-based-systems-engineering.com/mbse-book* zu finden.

unserer Vorliebe für Ockhams Rasiermesser, komplexe Probleme erfordern oft komplexe Lösungen).

SysML ersetzt ältere Systems-Engineering-Notationen mit einer Notation, die mit Sprachen und Methoden aller Disziplinen von der Softwareentwicklung bis zur Anlagenplanung integriert werden kann und von der echte Verkehrsflugzeuge (oder wie ich damals schrieb, »eine Ansammlung von Teilen, die in einem geschlossenen Verband fliegen«) abhängen. Zusammen mit der weltweiten Systems-Engineering-Zertifizierung (inklusive SysML) ist jetzt ein guter Zeitpunkt, um geradewegs zu starten, die SysML zu lernen und anzuwenden. Dieses Buch ist eine fantastische Unterstützung für dieses Vorhaben.

Richard Mark Soley, Ph.D.
Chairman and Chief Executive Officer
Object Management Group, Inc.
Circa 32.000 Fuß über Nebraska, U.S.A.
14. Januar 2014

Inhaltsverzeichnis

1 Einleitung 1
1.1 Vorweg 1
1.1.1 Passt das Buch zu mir? 3
1.1.2 Was bietet mir das Buch? 4
1.1.3 Wie ist das Buch entstanden? Und danke! 4
1.1.4 Wie lese ich das Buch? 6
1.1.5 Wohin geht der Weg? 7
1.2 Systems Engineering 9
1.2.1 Was ist Systems Engineering? 10
1.2.2 Systems-Engineering-Prozesse 14
1.2.3 Der Systems Engineer 16
1.2.4 Historie des Systems Engineering 18
1.2.5 International Council on Systems Engineering 19
1.2.6 Systems Engineering in Deutschland 20
1.3 Modellbasiertes Systems Engineering (MBSE) 21
1.3.1 Die Sprachen UML und OMG SysML 22
1.3.2 BPMN 25
1.3.3 MATLAB/Simulink 26
1.3.4 Modelica 26
1.3.5 Specification and Description Language (SDL) 27
1.4 Randnotizen 27
1.4.1 AUTOSAR 27
1.4.2 Capability Maturity Model Integration (CMMI) 28
1.4.3 Industrie 4.0 29
1.4.4 ISO/IEC 15288 30
1.4.5 Product Lifecycle Management (PLM) 31
1.4.6 Requirement Interchange Format (ReqIF) 32
1.4.7 STEP 33
1.4.8 V-Modell® XT 34

2 Pragmatischer Modellierungsprozess SYSMOD 37
2.1 Fallbeispiel 41
2.2 Die Systemidee 43

2.3 Systemidee und Systemziele beschreiben 47
2.4 Basisarchitektur festlegen 51
2.5 Anforderungen ermitteln 53
2.5.1 Stakeholder identifizieren 54
2.5.2 Anforderungen aufnehmen 59
2.6 Systemkontext modellieren 69
2.6.1 Systemakteure identifizieren 71
2.6.2 System/Akteur-Objektfluss modellieren 79
2.6.3 Systeminteraktionspunkte identifizieren 86
2.7 Anwendungsfälle modellieren 90
2.7.1 Anwendungsfälle identifizieren 92
2.7.2 Anwendungsfälle essenziell beschreiben 104
2.7.3 Systemprozesse beschreiben 110
2.7.4 Anwendungsfälle redundanzfrei modellieren 115
2.7.5 Anwendungsfallabläufe modellieren 121
2.7.6 Objektfluss modellieren 131
2.8 Fachwissen modellieren 140
2.9 Logische Architektur modellieren 146
2.9.1 System/Akteur-Interaktion modellieren 149
2.9.2 Systemschnittstellen ableiten 152
2.9.3 Systemstrukturen modellieren 158
2.9.4 Zustandsmodell erstellen 168
2.9.5 Physische Produktarchitektur modellieren 172
2.10 Randnotizen 175
2.10.1 Modellmanagement 175
2.10.2 Variantenmanagement 177
2.10.3 SYSMOD-Zickzackmuster 182
2.10.4 Funktionale Architektur 183
2.10.5 Agiles Systems Engineering 185
2.10.6 Datenaustauschformate 186
2.10.7 SYSMOD-Intensitätsmodell 187
2.10.8 Modellsimulation 190
2.10.9 Testen 190
2.10.10 *System of Systems* (SoS) 192
2.10.11 Modellierungsmuster 194
2.10.12 Tod des Akteurs! Lang lebe der Akteur! 200

3 UML – Unified Modeling Language 201
3.1 Historie 202
3.2 Aufbau und Konzepte 205
3.3 Das Klassendiagramm 208
3.3.1 Classifier 209
3.3.2 Klasse 210

3.3.3 Eigenschaft 212
3.3.4 Operation 215
3.3.5 Assoziation 216
3.3.6 Aggregation und Komposition 218
3.3.7 Objektspezifikation 220
3.3.8 Abhängigkeitsbeziehung 221
3.3.9 Abstraktionsbeziehung 222
3.3.10 Generalisierung 223
3.3.11 Signal 226
3.3.12 Datentypen 226
3.3.13 Assoziationsklasse 228
3.4 Das Kompositionsstrukturdiagramm 229
3.4.1 Eigenschaft 231
3.4.2 Konnektor 232
3.4.3 Port 232
3.5 Das Anwendungsfalldiagramm 234
3.5.1 Anwendungsfall 235
3.5.2 Akteur 239
3.5.3 Enthältbeziehung 241
3.5.4 Erweiterungsbeziehung 243
3.6 Das Aktivitätsdiagramm 244
3.6.1 Aktivität 244
3.6.2 Aktion und Pin 248
3.6.3 Aktivitätskante 254
3.6.4 Aktivitätspartition 256
3.6.5 Parametermenge 257
3.6.6 Entscheidung und Zusammenführung 258
3.6.7 Mengenverarbeitung 260
3.6.8 Splitting und Synchronisation 261
3.6.9 Start- und Endknoten 264
3.6.10 Unterbrechbarer Aktivitätsbereich 266
3.6.11 Zentralpuffer und Datenspeicher 267
3.7 Das Zustandsdiagramm 268
3.7.1 Zustandsautomat 269
3.7.2 Zustand 270
3.7.3 Transition 274
3.7.4 Auslöser und Ereignis 276
3.7.5 Start- und Endzustand 278
3.7.6 Pseudozustand 279
3.8 Die Interaktionsdiagramme 283
3.8.1 Interaktion 284
3.8.2 Lebenslinie 285
3.8.3 Nachricht 287

3.8.4 Kombiniertes Fragment 289
3.8.5 Interaktionsreferenz 292
3.8.6 Zustandsinvariante 293
3.8.7 Zeitliche Zusicherungen 295
3.9 Das Paketdiagramm 296
3.9.1 Paket 296
3.9.2 Importbeziehung 297
3.10 Sonstige Modellelemente 298
3.10.1 Diagrammrahmen 298
3.10.2 Erweiterungsmechanismus Stereotyp 299
3.10.3 Informationsfluss 304
3.10.4 Kommentar 304
3.10.5 Modell 305
3.10.6 Zusicherung 306

4 SysML – Systems Modeling Language 309
4.1 Historie 311
4.2 Aufbau und Konzepte 312
4.3 Das Anforderungsdiagramm 314
4.3.1 Anforderung 315
4.3.2 Ableitungsbeziehung 317
4.3.3 Enthältbeziehung 318
4.3.4 Erfüllungsbeziehung 319
4.3.5 Kopiebeziehung 321
4.3.6 Prüfbeziehung 324
4.3.7 Verfeinerungsbeziehung 325
4.3.8 Verfolgungsbeziehung 327
4.3.9 Testfall 328
4.3.10 Tabellennotation 329
4.4 Die Zuteilung 330
4.4.1 Zuteilungsbeziehung 331
4.4.2 Zuteilungspartition 334
4.4.3 Tabellennotation 335
4.5 Die Blockdiagramme 336
4.5.1 Systembaustein 337
4.5.2 Port 343
4.5.3 Assoziationsbaustein 346
4.5.4 Bindungskonnektor 347
4.5.5 Einheit und Basisgröße 348
4.5.6 Einschränkende Referenz und Pfadendemultiplizität 349
4.5.7 Objektfluss 352
4.5.8 Schnittstellenbaustein 353
4.5.9 Werteverteilung 354

4.5.10 Wertetyp 355
4.6 Das Zusicherungsdiagramm 356
4.6.1 Zusicherungsbaustein 357
4.7 Das Anwendungsfalldiagramm 359
4.8 Das Aktivitätsdiagramm 360
4.8.1 Aktivitätsbaum 361
4.8.2 Kontrolloperator 363
4.8.3 Rate 364
4.8.4 Spezielle Objektknoteneigenschaften 365
4.8.5 Wahrscheinlichkeit 367
4.8.6 Zeitliche Zusicherungen 368
4.9 Das Zustandsdiagramm 369
4.10 Die Interaktionsdiagramme 370
4.11 Allgemeine Modellierungselemente 371
4.11.1 Begründung 371
4.11.2 Diagrammrahmen 372
4.11.3 Gruppe 374
4.11.4 Modellsicht und Standpunkt 375
4.11.5 Problem 377
4.11.6 Stakeholder 378

5 Systems-Engineering-Profil SYSMOD 381
5.1 Akteurskategorien 382
5.2 Aktivitäten 384
5.3 Erweiterte Anforderung 385
5.4 Spezielle Anwendungsfälle 389
5.5 Benutzerschnittstelle 390
5.6 Disziplinenspezifische Elemente 391
5.7 Gewichtete Beziehungen 392
5.8 Erweiterter Stakeholder 394
5.9 System und Subsystem 396
5.10 Spezielle Systembausteine 397
5.11 System- und Zusicherungskontextelement 398
5.12 Systemprozess 399
5.13 Varianten 401
5.14 Ziel 403

6 OMG Certified Systems Modeling Professional (OCSMP) 405
6.1 Sinn und Unsinn von Zertifizierungen 405
6.2 Der Zertifizierungsprozess 407
6.3 Das OCSMP-Zertifizierungsprogramm 410
6.4 Andere Zertifizierungsprogramme 412

6.5 OCSMP Model User 412
6.5.1 OCSMP-Basiselemente der SysML 413
6.5.2 Coverage-Map OCSMP Model User 414
6.5.3 Referenzen 416
6.5.4 Beispielfragen 418
6.6 OCSMP Model Builder Fundamental 422
6.6.1 Coverage-Map OCSMP Model Builder Fundamental 422
6.6.2 Referenzen 423
6.7 OCSMP Model Builder Intermediate 423
6.7.1 Coverage-Map OCSMP Model Builder Intermediate 424
6.7.2 Referenzen 426
6.8 OCSMP Model Builder Advanced 426
6.8.1 Coverage-Map OCSMP Model Builder Advanced 426
6.8.2 Referenzen 428

A Anhang 429
A.1 Glossar 429
A.2 SysML auf Deutsch 457
A.3 Veraltete Konzepte der SysML 464
A.3.1 Standardport und Objektflussport 464
A.3.2 Modellsicht und Standpunkt 467
A.3.3 Einheit 468
A.4 Lösungen 469

Literaturverzeichnis 471

Index 477

1 Einleitung

Die Technik entwickelt sich vom Primitiven über das Komplizierte zum Einfachen.
(Antoine de Saint-Exupéry)

Nach der Lektüre dieses Kapitels können Sie die folgenden Fragen beantworten:

- Warum gibt es dieses Buch?
- Was ist modellbasiertes Systems Engineering (Vogelperspektive)?
- Welcher Zusammenhang besteht zwischen den Sprachen OMG SysML™ und UML™?
- Wie verhält sich der Inhalt dieses Buchs zu verwandten Themen wie beispielsweise CMMI™, ReqIF, Modelica oder dem V-Modell?

1.1 Vorweg

»Früher war alles besser!«

»Früher war alles viel einfacher. Da konnte man noch mit einer Handvoll Leute etwas auf die Beine stellen. Heute sind es schnell hundert Leute, die man braucht, um ein ordentliches System zu entwickeln. Und die kriegen dann meist nichts auf die Reihe ... Denk doch nur mal an das Projekt *P08/15*. Dabei sind dort Experten aus allen Bereichen vertreten ...«

Wechselstimmung

Solche oder ähnliche Feierabendmonologe höre ich immer häufiger. Als Geschäftsführer eines Beratungsunternehmens treffe ich viele Personen aus den unterschiedlichsten Branchen. Der Tenor ist aber gleich. Woran liegt das? Ganz einfach: am Fortschritt.

Wir sind an einem Punkt angekommen, an dem komplexe und verteilte Systeme gefordert werden, die von komplexen und verteilten Organisationen entwickelt werden. Die herkömmlichen Entwicklungsmethoden sind aber noch nicht bereit, diese ausreichend schnell und in einem akzeptablen Kostenrahmen zur Verfügung zu stellen.

»Fortschritt ist unaufhaltsam.«

Wir können nicht erwarten, immer fortschrittlichere, größere, bessere Systeme zu entwickeln und dabei weiterhin dieselben Werkzeuge einzusetzen. Die Vorgehensweisen, Modellierungssprachen, Entwicklungsumgebungen und Rollen wie der Systems Engineer müssen an dem Fortschritt teilhaben und sich ebenfalls weiterentwickeln.

Von Bits zu Objekten

In der Softwareentwicklung ist dieser Weg an einem Beispiel deutlich zu sehen. Angefangen bei Lochkarten über Assembler, prozedurale Programmiersprachen bis hin zu den objektorientierten Sprachen kennen die Entwicklungswerkzeuge immer größere Bausteine (von Bits bis Klassen/Objekte). Sie erleichtern so die Beschreibung komplexer Systeme. Der Weg zur nächsten Generation ist bereits eingeschlagen: Die grafische *Unified Modeling Language* (UML™) oder domänenspezifische Sprachen (DSL) werden zunehmend verwendet, um Softwaresysteme zu entwickeln, und mit diesen Sprachen werden immer mehr Aufgaben gelöst, die vorher mit herkömmlichen Programmiersprachen erledigt wurden (Abb. 1.1).

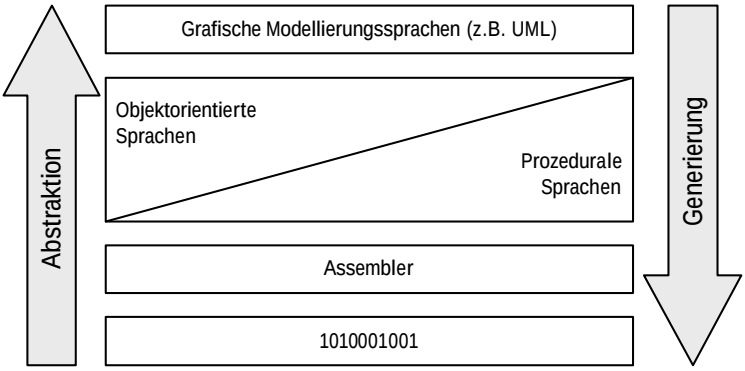


Abbildung 1.1 *Steigende Abstraktion der Programmiersprachen*

1+1=3

Das Gesamtsystem ist mehr als die Summe seiner Bausteine. Die Wechselwirkungen zwischen den Elementen können komplex und schwer kontrollierbar sein. Das führt dazu, dass Vorgehensmodelle und Notationen benötigt werden, um effektiv diese Systeme entwickeln zu können. Ansonsten werden die Kosten der Entwicklung in naher Zukunft in keinem Verhältnis zum akzeptablen Preis der Produkte stehen.

Systems Engineering

Das Systems Engineering (Systementwicklung[1]) setzt sich schon länger mit dieser Problematik auseinander. Die Entwick-

[1] Eigentlich bevorzuge ich deutsche Begriffe. Mit *Systems Engineering* mache ich eine Ausnahme, da dieser Begriff auch im deutschsprachigen Raum geläufiger ist als die Übersetzung.

lung großer Systeme, bei der viele unterschiedliche Disziplinen beteiligt sind, erfordert ganzheitliche Sichtweisen. Also losgelöst vom spezifischem Detailwissen werden die Anforderungen und Strukturen des Systems betrachtet, der gesamte Lebenszyklus von der Idee bis zur Entsorgung geplant, um insgesamt ein System zu entwickeln, das den Wünschen aller Beteiligten entspricht.

Modellbasiertes Systems Engineering (MBSE)

Im Systems Engineering fordert der Fortschritt einen Paradigmenwechsel: vom dokumentenzentrierten zum modellbasierten Systems Engineering (MBSE). Das Modell wird zur Quelle aller relevanten Informationen. Um gleich einem gängigen Missverständnis vorzugreifen: Die Dokumente sollen nicht verschwinden. Nur sind sie nicht mehr die Quelle der Informationen, sondern eine Sicht auf das Modell, beispielsweise automatisch erzeugt von einem Modellierungswerkzeug. Die Modellierungssprachen *OMG Systems Modeling Language* (OMG SysML™) und UML spielen in diesem Szenario natürlich eine wichtige Rolle.

Voneinander lernen

Die Softwareentwicklung kann viel vom Systems Engineering lernen. Zum Nehmen gehört aber auch ein Geben. Umgekehrt kann das Systems Engineering auch von der Softwareentwicklung lernen. Werkzeuge und Methoden zur Modellierung sind hier schon sehr weit entwickelt. Die UML ist eine Modellierungssprache aus diesem Umfeld. Sie hat sich als weltweiter Standard etabliert. Dem Systems Engineering fehlte bisher eine standardisierte Modellierungssprache. Das wird sich durch die neue SysML[2] ändern. Sie basiert auf der UML und wird von führenden Unternehmen aus der Systems-Engineering-Branche einschließlich des *International Council on Systems Engineering* (INCOSE) unterstützt.

UML
⇒ S. 201

SysML
⇒ S. 309

INCOSE
⇒ S. 19

1.1.1 Passt das Buch zu mir?

Dieses Buch wird Sie interessieren und für Sie sehr hilfreich sein, wenn Sie ...

- die Modellierungssprache SysML lernen wollen,
- Systeme beschreiben, spezifizieren und entwickeln,
- mit komplexen Systemen arbeiten,
- Systems Engineering mit SysML durchführen wollen,
- Systeme mit gemischten Disziplinen – beispielsweise Software und Hardware – entwickeln,

[2] Korrekter wäre die Bezeichnung *OMG SysML™*. Ich werde aber in diesem Buch häufig die Kurzform *SysML* verwenden.

- ein ganzheitliches Modell erstellen wollen, in dem Elemente nachvollziehbar zusammenhängen (*Traceability*),
- einen durchgängigen Weg von der Systemidee zum Systemdesign kennenlernen wollen.

Wenn Sie sich nicht in den obigen Punkten wiederfinden, fragen Sie mich, ob das Buch für Sie wertvoll sein könnte. Ich freue mich über eine E-Mail von Ihnen: *tim@larus.de*.

Werkzeugkasten SYSMOD

Sie lernen in diesem Buch einen Werkzeugkasten kennen, mit dessen Hilfe Sie einen Weg von der Idee eines Systems zur Architektur finden. Die Ergebnisse halten wir in detaillierten und aussagekräftigen Modellen fest. Die Modellierungssprache ist SysML. Der Werkzeugkasten heißt SYSMOD – abgeleitet von Systemmodellierungsprozess bzw. *Systems Modeling Process*.

1.1.2 Was bietet mir das Buch?

In diesem Buch finden Sie:

- den Werkzeugkasten SYSMOD zur Systementwicklung von der Systemidee zur Systemarchitektur,
- Steckbriefe zu den Werkzeugen, die Sie individuell zu einem Gesamtvorgehen zusammensetzen können – das Buch gibt Ihnen einen Standardweg vor,
- eine Beschreibung der SysML und UML,
- eine Einführung in das Systems Engineering,
- Randnotizen beispielsweise über Variantenmanagement, funktionale Architekturen, Simulation, Testen und Modellierungsmuster,
- Best Practices zu SysML und SYSMOD.

1.1.3 Wie ist das Buch entstanden? Und danke!

UML ohne Software

Die Idee zu diesem Buch hatte ich im Jahr 2003 bekommen. Zu dieser Zeit hatte ich viele Seminare und Beratungseinsätze zum Thema Analyse und Design mit der UML. Der Teilnehmerkreis konzentrierte sich auf Softwareentwickler – vom Programmierer bis zum Projektleiter. In einem Seminar hatte sich allerdings ein außergewöhnlicher Teilnehmerkreis zusammengefunden: Ingenieure, z. B. Nachrichtentechniker, aber kein einziger Softwareentwickler. Sie planten ein großes Projekt, das Software, aber auch bauliche Maßnahmen, Hardware und andere Disziplinen beinhaltete. In der Schulung habe ich den Softwareaspekt reduziert und

die Analyse- und Designtechniken allgemeiner erläutert. Für die Teilnehmer hat sich hieraus eine hervorragende Vorgehensweise für ihr Projekt ergeben.

Diese Teilnehmerkonstellation war ab dann kein Einzelfall mehr. Es folgten weitere Seminare, Workshops und Beratungsaufträge, in denen keine Softwareentwickler, sondern Ingenieure aus anderen Disziplinen teilnahmen, die Analyse und Design mit der UML für ihre Arbeit kennenlernen wollten. In mir keimten zunehmend Fragestellungen, Ideen und weiterführende Überlegungen auf: Wie viel Software enthält eigentlich die UML? Wie beschreibe ich Anforderungen mit der UML? Wie gehe ich mit hybriden Systemen um? Mir wurde langsam bewusst, dass die Sprache UML und die Vorgehensweise in vielen Bereichen unabhängig von Software eingesetzt werden kann.

Systems Engineering und SysML

OMG
⇒ S. 201

Der Blick über den Tellerrand hinaus hat mich zur Disziplin Systems Engineering geführt. Hier konnte ich mich mit meinen Gedanken gut einordnen. Der Zufall wollte es, dass ich in der Zeit durch meine Arbeit in der OMG an der UML2 angesprochen wurde, ob ich die Entwicklung der SysML unterstützen könnte. Das passte hervorragend in meine aktuellen Überlegungen, und ich habe sofort zugesagt. Seitdem bin ich aktiv an der Entwicklung von SysML beteiligt und Koautor der SysML-Spezifikation.

Die allgemeine Vorgehensweise in Analyse und Design, das Systems Engineering und die Modellierungssprache SysML – das Mosaik wurde vollständig. Das Bild, das sich daraus ergeben hat, möchte ich Ihnen in diesem Buch darstellen.

Danke, Kunden und Partner!

Die passenden Mosaiksteine konnte ich nur finden, da meine Ideen in vielen Diskussionen mit meinen Kunden und Partnern reifen konnten. Dafür ein großes Dankeschön! Besonders erwähnen möchte ich Reiner Dittel, Marc Enzmann, Ulrich Lenk und Robert Karban, Andreas Peukert sowie Dr. Rudolf Hauber und weitere Mitglieder aus dem *INCOSE MBSE Challenge Team SE^2*[3]. Vielen Dank auch an Jesko Lamm von der Firma Bernafon für die zahlreichen Diskussionen und das gemeinsame Entwickeln der Methodik zu funktionalen Architekturen für Systeme (FAS) [58].

Danke, SysML!

Die kreativen Auseinandersetzungen in der SysML-Arbeitsgruppe der OMG mit sehr kompetenten Modellierern und Systemingenieuren haben mich große Schritte vorangebracht. Mein be-

[3] Im Rahmen der MBSE-Initiative von INCOSE haben sich weltweit *Challenge Teams* zur Weiterentwicklung des MBSE gebildet. Die Ergebnisse des SE^2-Teams finden Sie auf der Webseite `mbse.gfse.de`.

sonderer Dank gilt Conrad Bock, Sanford Friedenthal, Bran Selic, Alan Moore und Roger Burkhart.

Danke, oose! Sehr wichtig für meine fachliche und persönliche Weiterentwicklung sind die Umgebung und die Freiräume, die mir meine Firma oose Innovative Informatik GmbH gibt. Vielen Dank an meine Kollegen und ganz besonders an Bernd Oestereich für die Unterstützung. Einige Abbildungen in diesem Buch wurden freundlicherweise von der oose Innovative Informatik GmbH zur Verfügung gestellt.

Danke, Verlag! Ein großes Lob und Dankeschön an den dpunkt.verlag. Vor allem natürlich an Christa Preisendanz!

Danke, Richard! Es ist immer eine Freude, mit Richard M. Soley zu kommunizieren. Vielen Dank für das fantastische Geleitwort.

Danke, Reviewer! Ohne ein fachliches Review kann ein Buch keine ausreichende Qualität erreichen. Der Blick von außen ist wichtig. Vielen Dank besonders an Wolfgang Krenzer von IBM Automotive Industry und Andreas Korff von ARTiSAN Software Tools GmbH für die Reviews der 1. Auflage dieses Buchs.

Danke, Pavel Hruby! Ich möchte auch Pavel Hruby für die hervorragende UML-Visio-Schablone danken (`www.phruby.com`). Sie wird übrigens auch für die offizielle UML- und SysML-Spezifikation verwendet.

Danke, Leser! Vielen Dank natürlich auch an Sie, dass Sie dieses Buch gekauft haben und sich für das Thema interessieren. An Ihrem Feedback bin ich sehr interessiert. Schreiben Sie mir: *tim@larus.de*.

1.1.4 Wie lese ich das Buch?

Lesen und Nachschlagen Das Buch besteht aus zwei Teilen: Kapitel 1 und Kapitel 2 sind zum durchgängigen Lesen, Kapitel 3 und folgende sind primär zum Nachschlagen geeignet.

Einleitung Sie befinden sich gerade in Kapitel 1. Hier erfahren Sie etwas über das Buch selbst – beispielsweise diese Anleitung zum Lesen – sowie einführende Grundlagen zum Thema Systems Engineering, SysML und UML. Sie können dieses Kapitel auch überspringen und direkt in Kapitel 2 einsteigen.

SYSMOD In Kapitel 2 beschreibe ich das Vorgehen SYSMOD, um die Anforderungen an ein System zu erfassen, zu modellieren und eine Architektur abzuleiten, die den gestellten Anforderungen genügt. Die Ergebnisse des Vorgehens werden mit SysML modelliert.

Randnotizen Am Ende von Kapitel 2 streife ich weitere Themen, die für das Vorgehen wichtig sind, wie beispielsweise funktionale Architekturen, Variantenmanagement und Modellmanagement.

In Kapitel 4 und Kapitel 3 erläutere ich die Modellierungssprachen UML und SysML. Da SysML auf der UML aufbaut, stehen im SysML-Kapitel nur die erweiternden Elemente. Im UML-Kapitel erläutere ich entsprechend nur die Sprachelemente, die auch in SysML verwendet werden. Beide Kapitel stellen also insgesamt den SysML-Sprachumfang dar. Die Abgrenzung zeigt Ihnen deutlich, was die UML leistet und was die SysML ergänzt.

UML und SysML

In Kapitel 5 beschreibe ich die Spracherweiterungen (Stereotypen), die das in Kapitel 2 beschriebene Vorgehen SYSMOD benötigt. Sie gehören nicht zu den Sprachstandards SysML und UML.

Profil SYSMOD
Stereotyp
⇒ S. 299

Am Seitenrand stehen Zwischenüberschriften zur besseren Orientierung und Querverweise mit Seitenangaben zu Zusatzinformationen, die für das Verständnis des Absatzes hilfreich sind.

Seitenrand

1.1.5 Wohin geht der Weg?

Mit SysML hat die UML ihr Ausbreitungsgebiet weiter vergrößert. Ausgehend von einer reinen Modellierungssprache zur Softwareentwicklung über die Geschäftsprozessmodellierung [61] nun zur Modellierungssprache des Systems Engineering. Wohin führt dieser Weg?

Ausbreitung der UML

Es scheint, dass die UML auf dem besten Weg ist, zur *Lingua franca* der Modellierung zu werden. Das Potenzial hat sie. Die UML ist erweiterbar und somit an spezifische Bedürfnisse anpassbar. Sie ist international verbreitet, erprobt und anerkannt. Und hinter ihr steht die OMG – ein international agierendes Konsortium, dem führende IT-Firmen angehören.

Lingua franca?

Auch wenn ich ein starker Verfechter der UML bin und sie derzeit in ihrer Rolle konkurrenzlos sehe, glaube ich nicht, dass sie die *Lingua franca* der Modellierung wird. Ich denke (und hoffe), dass dies keine Sprache erreichen wird. Eine gewisse Vielfalt ist notwendig. Die UML legt aber die Saat für künftige Modellierungssprachen, was dazu führt, dass sie sich im Kern ähneln und in gewissen Grenzen kompatibel sind.

Kompatible Sprachfamilie

Die Zukunft wird einen großen Bedarf an Modellierungssprachen bringen. Unsere Systeme, die wir entwickeln, werden immer komplexer. Die Modellierungssprachen erlauben es mir, mich auf verschiedenen Abstraktionsebenen zu bewegen. Je abstrakter ich werde, desto einfacher erscheint mir das System. Ein guter Modellierer beherrscht die Kunst, auf abstrakter Ebene konkret zu werden.

Weiterführende Referenzen ✔

- **Die Homepage zum Buch und MBSE-Blog:**
  `www.model-based-systems-engineering.com`

- **SYSMOD als Eclipse-Process-Framework-Modell:**
  `www.sysmod.de`

- **Beispielmodelle online:**
  Kfz-Zugangssystem:
  `example.model-based-systems-engineering.com`
  Teleskopprojekt: `mbse.gfse.de`

- **Direkter Kontakt:**
  tim@larus.de

- **Beratung, Trainings, Projektarbeit:**
  Systems-Engineering@oose
  `www.oose.de/themen/systems-engineering`

- **OMG™**
  `www.omg.org` – Hauptseite
  `www.omgsysml.org` – OMG SysML™

- **INCOSE, Gesellschaft für Systems Engineering e.V.**
  `www.incose.org`
  `www.gfse.de`