

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 Systems Engineering?
- Welcher Zusammenhang besteht zwischen den Sprachen OMG SysML™ und UML™?
- Wie verhält sich der Inhalt dieses Buchs zu verwandten Themen wie beispielsweise CMMI™, RIF, AUTOSAR oder V-Modell® XT?

1.1 Vorweg

»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...«

*»Früher war alles
besser!«*

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

Wechselstimmung

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

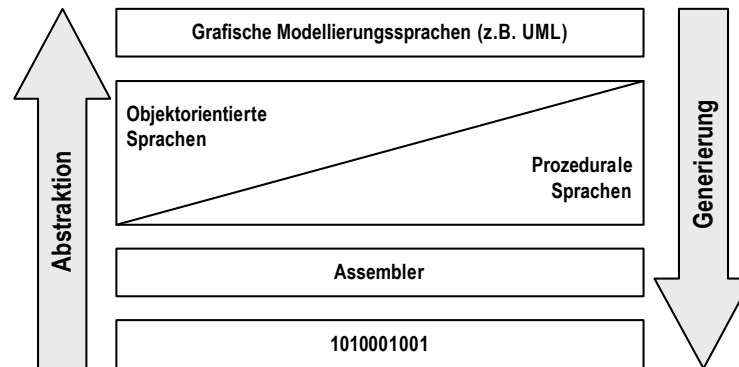
»Fortschritt ist unaufhaltsam«

Wir können nicht erwarten, immer fortschrittlichere, größere, bessere usw. Systeme zu entwickeln und dabei immer noch dieselben Werkzeuge einzusetzen. Die Vorgehensweisen, Modellierungssprachen, Entwicklungsumgebungen und Rollen (z. B. der Systemingenieur) 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 (0/1 bis Klassen/Objekte) und erleichtern so die Beschreibung komplexer Systeme. Der Weg zur nächsten Generation ist bereits eingeschlagen: Die grafische *Unified Modeling Language* (UML™) wird zunehmend verwendet, um Softwaresysteme zu entwickeln, und mit ihr 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

Einzelne Bausteine des Gesamtsystems können effizient mit bewährten Methoden entwickelt werden. Das Gesamtsystem ist aber mehr als die Summe seiner Bausteine. Die Wechselwirkungen zwischen den Elementen können komplex und schwer kontrollierbar sein.

Eingebettete Systeme

In der Softwareentwicklung ist die Notwendigkeit nach einer ganzheitlichen Denkweise besonders zu spüren. Die Einbettung der Software in ein System ist ein wesentlicher Komplexitätsfaktor. 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.

Das Systems Engineering (Systementwicklung¹) setzt sich schon länger mit dieser Problematik auseinander. Die Entwicklung großer Systeme, bei der viele unterschiedliche Disziplinen beteiligt sind, erfordert ganzheitliche Sichtweisen. Also losgelöst von spezifischen 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.

Systems Engineering

Im Systems Engineering fordert der Fortschritt einen Paradigmenwechsel: vom dokumentenzentrierten zum modellzentrierten Systems Engineering. 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 *Systems Modeling Language* (OMG SysMLTM) und UML spielen in diesem Szenario natürlich eine wichtige Rolle.

*Modellzentriertes
Systems Engineering*

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 zu einem weltweiten Standard etabliert. Dem Systems Engineering fehlte bisher eine standardisierte Modellierungssprache. Das wird sich durch die neue SysML² ändern. Sie basiert auf der UML und wird von führenden Unternehmen aus der Systems-Engineering-Branche einschließlich dem *International Council on Systems Engineering* (INCOSE) unterstützt.

Voneinander lernen

UML
⇒ S. 199

SysML
⇒ S. 299
INCOSE
⇒ S. 19

1.1.1 Passt das Buch zu mir?

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

- die neue Modellierungssprache SysML kennenlernen wollen.
- Systeme beschreiben, spezifizieren und entwickeln.

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

²Korrekt wäre die Bezeichnung *OMG SysMLTM*. Ich werde aber in diesem Buch häufig die Kurzform *SysML* verwenden.

- ❑ mit komplexen Systemen arbeiten.
- ❑ Systems Engineering mit SysML oder UML durchführen wollen.
- ❑ Systeme mit gemischten Disziplinen – beispielsweise Software und Hardware – entwickeln.
- ❑ ein ganzheitliches Modell erstellen wollen, in dem Elemente nachvollziehbar zusammenhängen (*Traceability*).
- ❑ einen durchgängigen Weg von der Systemidee zum Systemdesign kennenlernen wollen.
- ❑ in der Softwareentwicklung tätig sind und eine ganzheitliche Sichtweise haben.

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: twe@system-modellierung.de.

Werkzeugkasten
SYSMOD

Sie lernen in diesem Buch einen Werkzeugkasten kennen, mit dessen Hilfe Sie einen Weg von der Idee eines Systems zum Design finden. Die Ergebnisse halten wir in detaillierten und aussagekräftigen Modellen fest. Die Modellierungssprache ist SysML oder UML. Sie lernen beide Sprachen kennen sowie ihre Gemeinsamkeiten und Unterschiede. Damit das Kind auch einen Namen hat, nenne ich den Werkzeugkasten SYSMOD – abgeleitet von Systemmodellierungsprozess.

1.1.2 Was bietet mir das Buch?

In diesem Buch finden Sie:

- ❑ einen Werkzeugkasten zur Systementwicklung von der Systemidee zum Design (SYSMOD)
- ❑ Steckbriefe zu den Werkzeugen, so dass Sie sie leicht individuell zu einem Gesamtvorgehen zusammensetzen können. Das Buch gibt Ihnen schon einen Standardweg vor.
- ❑ eine Beschreibung der SysML und UML
- ❑ eine Einführung in das Systems Engineering
- ❑ Randnotizen beispielsweise über Variantenmanagement, Simulation, Testen und Modellierungsmuster.
- ❑ Best Practices zu SysML und SYSMOD

1.1.3 Wie ist das Buch entstanden? und danke!

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.

UML ohne Software

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.

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.

*Systems Engineering
und SysML*

*OMG
⇒ S. 199*

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.

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

*Danke, Kunden und
Partner!*

und Robert Karban, Andreas Peukert sowie Dr. Rudolf Hauber aus dem *INCOSE MBSE Challenge Team SE²*³.

Danke, SysML!

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

Danke, Pavel Hruby!

Ich möchte auch Pavel Hruby für die hervorragende UML-Visio-Schablone danken (<http://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: twe@systemmodellierung.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 zum Nachschlagen geeignet.

³Im Rahmen der *Model-based Systems Engineering* (MBSE) Initiative des *International Council on Systems Engineering* (INCOSE) haben sich weltweit *Challenge Teams* zur Weiterentwicklung des MBSE gebildet. Das Team SE² untersucht primär die Anwendungsmöglichkeiten von SysML und besteht im Kern aus Robert Karban, Andreas Peukert, Dr. Rudolf Hauber und meiner eigenen Person.

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.

Einleitung

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

Vorgehen

Am Ende von Kapitel 2 streife ich weitere Themen, die für das Vorgehen wichtig sind, wie Variantenmanagement, *System of Systems* (SoS) und weitere.

Randnotizen

In Kapitel 3 und Kapitel 4 erläutere ich die Modellierungssprachen SysML und UML. Da SysML auf der UML aufbaut, stehen im SysML-Kapitel nur die erweiternden Elemente. Im UML-Kapitel werden entsprechend nur die Sprachelemente erläutert, 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.

SysML und UML

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

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 [42] 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 – sowie mit SysML auch das *International Council on Systems Engineering* (INCOSE).

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 das keine Sprache erreichen wird. Eine gewisse Vielfalt ist not-

Kompatible Sprachfamilie

wendig. Die UML legt aber die Saat für künftige Modellierungssprachen, was dazu führt, dass sie sich im Kern ähneln und in gewissen Grenzen kompatibel sind.

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:**
<http://www.system-modellierung.de>
<http://www.system-modeling.com> (englische Version)
- ❑ **SYSMOD als EPF^a-Modell:**
<http://www.sysmod.de>
- ❑ **Beispielmodell Online:**
<http://www.system-modellierung.de/sysml.beispiel>
- ❑ **Direkter Kontakt:**
twe@system-modellierung.de
- ❑ **Beratung, Seminare, Projektarbeit: oose.syng**
<http://www.syng.de>
- ❑ **OMGTM**
<http://www.omg.org> – Hauptseite
<http://www.omgsysml.org> – OMG SysMLTM
<http://www.uml.org> – UMLTM
- ❑ **INCOSE**
<http://www.incose.org>
- ❑ **Gesellschaft für Systems Engineering e.V.**
<http://www.gfse.de>

^aEclipse Process Framework

1.2 Systems Engineering

Jede Disziplin und jede Branche hat ihre eigenen Methoden. Sei es die Softwareentwicklung, die Hardwareentwicklung, die Mechanik, das Bauwesen, die Psychologie und so weiter. Diese Abgrenzungen funktionieren gut, solange man in einem Projekt innerhalb einer Disziplin bleibt. Bei disziplinübergreifenden Projekten wird es schwieriger. Zwischen den Methoden der jeweiligen Disziplinen müssen Schnittstellen geschaffen und aufeinander abgestimmt werden. Das Projektmanagement steht dann der Herausforderung gegenüber, die Besonderheiten aller Disziplinen zu berücksichtigen.

*Unterschiedliche
Disziplinen,
unterschiedliche
Methoden*

Die vielen Witze über die Eigenarten von Physikern, Ingenieuren, Informatikern, Mathematikern & Co. machen deutlich, dass bei interdisziplinären Projekten unterschiedliche Welten aufeinanderstoßen⁴. Missverständnisse und Konflikte sind vorprogrammiert.

*Unterschiedliche
Charaktere*

Die Softwareentwickler überlegen sich beispielsweise eine neue leistungsfähige Architektur. Diese benötigt allerdings mehr Speicherressourcen. Das fällt erst sehr viel später bei der Integration mit der Zielhardware auf. Die Hardwareentwickler sehen eine Möglichkeit, die Hardware ausreichend mit Speicher zu erweitern. Das Problem scheint gelöst. Aber noch viel später stellt man fest, dass die erweiterte Hardware keinen Platz mehr im Gehäuse hat, da die Gehäusedesigner von der Änderung nicht informiert worden sind. Die erhöhte Wärmeentwicklung der Hardware wird die geforderten Grenzwerte überschreiten. Jede Disziplin hat in ihrem Bereich die bestmögliche Lösung eingearbeitet. Die Summe der besten Einzellösungen ist aber nicht immer die beste Lösung für das Gesamtsystem.

*Gute Einzellösung,
schlechte
Gesamtlösung*

1+1 ergibt also 3. Die Entwicklung einzelner Subsysteme wird gut beherrscht. Aber das »+« erhöht die Komplexität und führt zu vielfältigen Problemen. Allzu häufig fehlt in Projekten die verantwortliche Rolle für die Summenbildung, also die ganzheitliche, systemweite Sicht. Dabei gibt es hierfür bereits bewährte

1+1=3

⁴Ein Maschinenbauer, ein Chemiker und ein Informatiker fahren in einem Auto durch die Wüste. Plötzlich bleibt das Auto stehen, und die drei beginnen über die Ausfallursache zu streiten. Der Chemiker: »Sicher ein unvermuteter Entropiezuwachs im Motorraum!« Der Maschinenbauer: »Blödsinn, es ist einfach der Keilriemen gerissen oder der Zündverteiler hat sich verabschiedet!« ... usw ... usw ... Bis es dem Informatiker zu dumm wird: »Ist doch egal, wir steigen einfach aus und wieder ein, dann wird's schon wieder laufen.«

Methoden und Strategien. Die Disziplin des Systems Engineering beschäftigt sich seit über 50 Jahren mit diesem Thema.

Was bedeutet
komplex?

»Das ist ganz schön komplex!« Das haben Sie bestimmt schon einmal gesagt. Was meint man eigentlich damit? Was bedeutet komplex? Es gibt noch ein ähnliches Wort: kompliziert. Ist es dasselbe? Nein! Etwas Komplexes muss nicht kompliziert sein. Die folgende Definition geht auf Georg Klaus zurück [34].

Definition

Die **Komplexität** bezieht sich auf die Anzahl und Art der Beziehungen zwischen Elementen in einem System.

Die **Kompliziertheit** bezieht sich auf die Anzahl der unterschiedlichen Elemente.

Gemäß dieser Definition sind viele Systeme komplex und hybride Systeme zusätzlich kompliziert. Das ist die Kernherausforderung des Systems Engineering.

Essenzielle
Komplexität

Der Umgang mit der Komplexität zielt primär nicht darauf, sie zu reduzieren, sondern sie einfach erscheinen zu lassen. Systeme besitzen eine essenzielle Komplexität, die in der Natur des Systems liegt [8]. Sie kann nicht reduziert werden, ohne den Systemzweck zu verändern. Schlechte Lösungen können eine unnötige technische Komplexität einführen, die im Rahmen des Systems Engineering oder disziplinspezifischer Betrachtungen reduziert werden kann.

1.2.1 Was ist Systems Engineering?

Allgemeinbegriff

Sicherlich haben Sie schon einmal von der Disziplin Systems Engineering gehört. Und wenn nicht, kommt der Begriff Ihnen vermutlich nicht fremd vor. Sowohl *System* als auch *Engineering* sind bekannte Allgemeinbegriffe. In Kombination hat auch jeder eine gewisse Vorstellung, was diese Disziplin ausmacht. Diese Vorstellungen sind nur oft sehr verschieden.

Definition System

Was verstehen Sie unter einem System? Ein Flugzeug? Ein Auto? Eine Lagerverwaltung? Ein Navigationssystem? Eine Textverarbeitungssoftware? Ihr Laptop? Ein Unternehmen? Alle Beispiele sind korrekt.

Definition

Ein **System** ist ein von Menschen erstelltes Artefakt bestehend aus Systembausteinen, die gemeinsam ein Ziel verfolgen, das von den Einzelementen nicht erreicht werden kann. Ein Baustein kann aus Software, Hardware, Personen oder beliebigen anderen Einheiten bestehen [52].

Diese Definition des Systembegriffs ist bewusst sehr allgemein gehalten. Sie können im Sinne des Systems Engineering auch an sehr große Systeme denken. Beispielsweise an das Luftfahrtverkehrssystem in Deutschland bestehend aus Flughäfen und Luftfahrtverkehrsstraßen.

Große Systeme

Bei der Größenbetrachtung des Systems müssen Sie deutlich zwischen der Größe des realen Systems und der Größe des Systemmodells unterscheiden. Das deutsche Luftfahrtverkehrssystem ist real deutlich größer als ein Auto. Wenn wir die zugehörigen Systemmodelle betrachten, kann es umgekehrt sein. Es ist abhängig von der Detaillierungstiefe der Modelle. Das Auto können wir beispielsweise bis zur letzten Schraube modellieren. Im Systemmodell des Luftverkehrssystems verwenden wir aber als kleinste Einheiten beispielsweise Flugzeuge und Flughäfen.

*Modellgröße vs.
Systemgröße*

Jetzt kennen Sie den ersten Teil des Begriffs *Systems Engineering*. Der zweite Teil – der englische Begriff *Engineering* – steht allgemein für eine Disziplin, die Methoden und Werkzeuge strukturiert einsetzt, um ein Produkt zu entwickeln.

Definition Engineering

Zusammengesetzt beschreibt der Begriff *Systems Engineering* Methoden, um erfolgreich Systeme zu entwickeln.

*Definition
Systems Engineering*

Definition

Das **Systems Engineering** konzentriert sich auf die Definition und Dokumentation der Systemanforderungen in der frühen Entwicklungsphase, die Erarbeitung eines Systemdesigns und die Überprüfung des Systems auf Einhaltung der gestellten Anforderungen unter Berücksichtigung des Gesamtproblems: Betrieb, Zeit, Test, Erstellung, Kosten & Planung, Training & Support und Entsorgung [52].

Das Systems Engineering integriert alle Disziplinen und beschreibt einen strukturierten Entwicklungsprozess vom Konzept über die Produktion- bis zur Betriebsphase und abschließend wie-

Metadisziplin

der die Außerbetriebnahme des Systems. Es werden sowohl die technischen als auch die wirtschaftlichen Aspekte betrachtet, um ein System zu entwickeln, das den Benutzerbedürfnissen entspricht.

Es steht damit über spezifischen Disziplinen wie beispielsweise der Softwareentwicklung. Als sogenannte Metadisziplin beschäftigt sich das Systems Engineering mit der Vorgehensweise von der Idee, ein System zu erstellen (Auslöser), über die Entwicklung, Realisierung und Nutzung bis hin zur Entsorgung des Systems. Diese ganzheitliche Denkweise schließt beispielsweise auch Lösungen zu Problemen ein, die erst durch die Einführung eines neuen Systems entstehen.

Paket
⇒ S. 288

Abbildung 1.2 zeigt Aufgabenbereiche des Systems Engineering in Form eines SysML-Paketdiagramms. Die einzelnen Bereiche werden jeweils nur auf Systemebene betrachtet und tauchen nicht in die Details einer Disziplin ein.

Die Konzepte des Systems Engineering findet man in den spezifischen Disziplinen wieder. Die Art und Weise, Probleme zu formulieren und Lösungswege zu entwickeln, weist hier viele Parallelen auf, die im Systems Engineering allgemein beschrieben sind. Inhalte sind beispielsweise die Beschreibung von Lebensphasenmodellen und Problemlösungszyklen.

Lebensphasen

Das Lebensphasenmodell beschreibt die zeitlichen Abschnitte Entwicklung, Realisierung, Nutzung und Entsorgung (siehe auch ISO/IEC 15288 in Abschnitt 1.4.4). Die Probleme in der Entwicklung von Systemen liegen darin, dass in der Entwicklungsphase Entscheidungen getroffen werden müssen, deren Folgen erst in späteren Phasen wirksam werden. In den späteren Lebensphasen besteht aber nur noch wenig Einfluss auf die Systemeigenschaften. Daraus leiten sich wichtige Fragen ab, die wir uns in der Entwicklungsphase stellen müssen, z. B.:

- Welche Probleme löst das System?
- Welche Probleme erzeugt das System?
- In welchem Umfeld wird das System eingesetzt?
- Wie lange soll das System eingesetzt werden?
- Wie wird das System durch einen Nachfolger ersetzt?

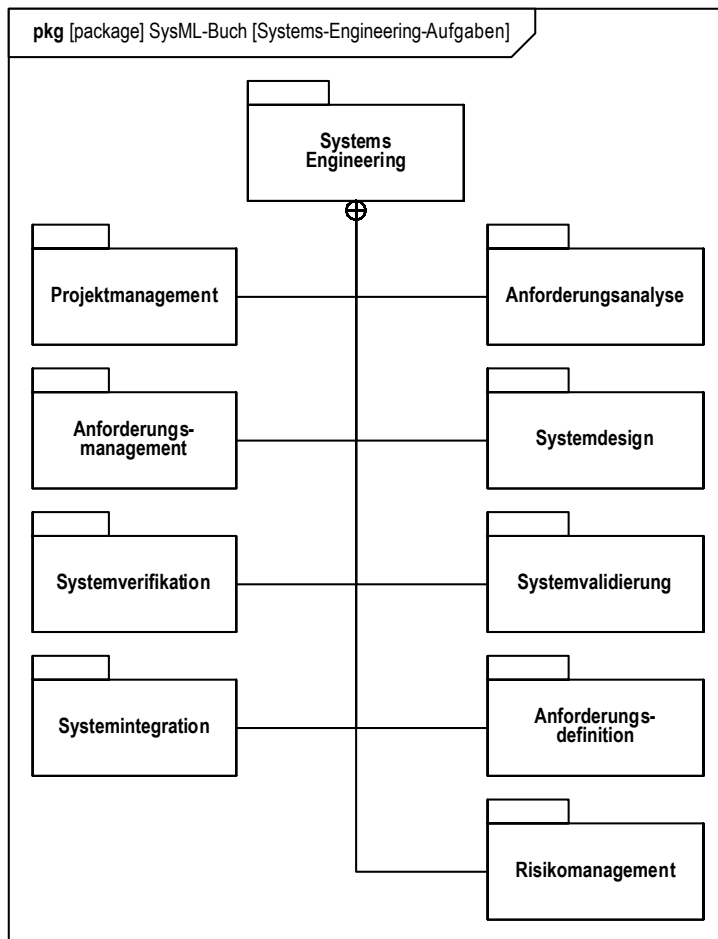


Abbildung 1.2
Aufgabenbereiche des
Systems Engineering

Angenommen Sie entwickeln einen Flugzeugantrieb auf Basis einer neu entdeckten Substanz als Treibstoff. Neben der reinen Antriebsentwicklung müssen Sie auch klären, wie die Tanksysteme auf den Flughäfen aussehen müssen. Kann der Antrieb nach herkömmlichen Methoden entsorgt werden, oder sind eigene Entsorgungssysteme notwendig? Welche Auswirkungen haben die Abgase auf Mensch und Natur? Und weitere Fragen.

Beispiel

Der Problemlösungszyklus beschreibt den Weg vom Auftreten des Problems bis zur Lösung. Die grobe Struktur besteht aus drei Schritten, die nicht nur linear, sondern auch mit Rückkopplungen durchlaufen werden.

Problemlösungszyklus

1. Aktuelle Situation beschreiben und das zu erreichende Ziel formulieren
2. Lösungsmöglichkeiten erarbeiten
3. Beste Lösung auswählen

Was so einfach und selbstverständlich klingt, wird in der Praxis nicht immer so gelebt. Allzu oft wird beispielsweise Punkt 1 ausgelassen oder in Punkt 2 nur eine Lösung betrachtet. Das Vorgehen in diesem Buch beginnt mit der Formulierung des Ziels der Systementwicklung (Abschnitt 2.1.1). Die Analyse und das Design des Systems in einem Modell unterstützen die Betrachtung unterschiedlicher Varianten, so dass die optimale Lösung ausgewählt werden kann (siehe auch Abschnitt 2.8.1).

1.2.2 Systems-Engineering-Prozesse

Einen guten Überblick über einen Systems-Engineering-Prozess liefert das SIMILAR-Prozessmodell [1]. Es ist eine Abkürzung und steht für

- *State the problem*
- *Investigate alternatives*
- *Model system*
- *Integrate*
- *Launch the system*
- *Assess performance*
- *Re-evaluate*

Im Einzelnen bedeuten die Aufgabenbereiche:

Anforderungsmodell erstellen (State the problem)

Am Anfang der Systementwicklung steht die Beschreibung der Aufgabe. Nur zu einer gut gestellten Aufgabe kann man auch eine gute Lösung finden. Fehler in dieser Phase können sehr kostspielig werden, sowohl finanziell als auch für das Ansehen. Die Aufgabenstellung definiert, was das System leisten bzw. welche Anforderungen das System erfüllen soll. Der Systemingenieur beschäftigt sich mit Anforderungen auf Systemebene.

Das Anforderungsmodell beschreibt auf oberster Ebene nicht die Lösung. Dies würde die Möglichkeit verbauen, alternative Lösungen zu evaluieren.

Alternative Lösungen prüfen (Investigate alternatives)

Eine wichtige Aufgabe des Systemingenieurs ist es, alternative Konzepte zu prüfen und gegeneinander abzuwägen. Leider wird diese Aufgabe oft vernachlässigt. Der – menschliche – Drang, sich auf nur eine Lösung zum Problem zu konzentrieren, verdeckt den Blick darauf, dass eine alternative Lösung gegebenenfalls besser geeignet ist.

Der Systemingenieur entwirft – basierend auf dem Anforderungsmodell – also nicht nur ein Systemdesign, sondern üblicherweise auch alternative Designs. Hiermit können verschiedene Lösungswege gegeneinander abgewogen werden. Selten wird es eine eindeutige Lösung geben, die alle Vorteile in sich vereint. Es müssen also verschiedene Kriterien und Prioritäten betrachtet werden. Typische Aspekte sind beispielsweise Kosten, Größe, Gewicht, Entwicklungszeit, Time-to-Market und Risiken. Aus den Systemmodellen werden simulierbare Parameter abgeleitet, um die notwendigen Aspekte unmittelbar sichtbar, vergleichbar und am Modell messbar zu machen. SysML bietet hierzu beispielsweise das Zusicherungsdiagramm an (Kap. 4.6).

Systemdesignmodell erstellen (Model system)

Designmodelle werden bereits beim Abwägen der alternativen Lösungsmöglichkeiten erstellt. Jetzt wird das Modell der ausgewählten Lösung detailliert. Aus Sicht des Systemingenieurs dient das Modell neben der Spezifikation des Systemdesigns auch zum Verwalten des gesamten Lebenszyklus des Systems.

Die Modellierung mit SysML ermöglicht eine gute Verfolgbarkeit (*Traceability*) von Aspekten, beispielsweise die Umsetzung einer Anforderung ins Design (siehe Erfüllungsbeziehung in Abschnitt 4.3.4).

System einbetten (Integrate)

Das System wird nicht alleine dastehen und einem Selbstzweck dienen. Es wird in eine Umgebung eingebettet und mit dieser Umgebung interagieren. Die Einbettung wird in dieser Aufgabe betrachtet. Sie beinhaltet beispielsweise die Definition der System-schnittstellen.

System implementieren (Launch the system)

Das System wird gemäß des spezifizierten Designmodells erstellt und in Betrieb genommen. Die Modelle aus den vorherigen Schritten geben Implementierungsanforderungen für die Software, Hardware, Mechanik usw. vor.

Technische Messwerte prüfen (Assess performance)

Das lauffähige System wird getestet und vermessen. Die Werte müssen den gestellten Anforderungen entsprechen. Der Zusammenhang zwischen Test- und Anforderungsmodell wird in SysML mit der Prüfbeziehung hergestellt (Abschnitt 4.3.6).

Projektergebnisse überprüfen (Re-evaluate)

Diese Aufgabe steht übergreifend über allen anderen Aktivitäten. Ergebnisse aus dem Prozess werden kritisch geprüft und bewertet. Als Konsequenz werden die gewonnenen Erkenntnisse als Rückkopplung wieder in den Prozess eingegeben.

Risikomanagement

Ein weiteres Themenfeld des Systems Engineering, das nicht Teil der Abkürzung SIMILAR ist, ist das Risikomanagement bzw. – positiver formuliert – das Vorsorgemanagement.

Um in einem Projekt ein gutes Risikomanagement betreiben zu können, ist ein allgemeines Grundverständnis aller Beteiligten notwendig: Die Projektmitarbeiter sind in ihrer Arbeit nicht absolut perfekt, und die Projektleiter können nicht zaubern. Dinge gehen einfach schief. Das ist der Normalzustand.

Das Risikomanagement sorgt dafür, dass potenzielle Risiken identifiziert und Maßnahmen definiert werden, um das Risiko zu minimieren bzw. das Problem bei Eintritt des Risikofalls zu lösen. Es ist Teil der Entscheidungsprozesse im Projekt und bedarf regelmäßiger Beobachtung, um den Eintritt eines prognostizierten Risikofalls auch rechtzeitig zu entdecken und entsprechend zu reagieren.

1.2.3 Der Systemingenieur

Bindeglied

Der Systemingenieur ist das Bindeglied zwischen den einzelnen – oftmals sehr unterschiedlichen – Disziplinen eines Projekts. Er denkt systemweit unabhängig von Software, Hardware oder anderen spezifischen Gesichtspunkten.

Organisatorisch ist die Systems-Engineering-Disziplin ähnlich einer Stabsstelle eingeordnet. Sie berichtet direkt an die Gesamtprojektleitung, die wiederum direkt mit den anderen Entwicklungsabteilungen kommuniziert. Der Systemingenieur ist nicht der Vermittler zwischen Projektleitung und den Entwicklungsabteilungen. Seine Rolle ist vergleichbar mit der des Softwarearchitekten⁵ auf Systemebene. Er legt die Architektur des Systems fest und muss in der Lage sein, sich mit den unterschiedlichen Entwicklungsabteilungen fachlich auseinanderzusetzen und nicht nur die Rolle des Beobachters und Nachrichtenüberbringers spielen. Insbesondere ist er auch mit Entscheidungsbefugnissen ausgestattet. Laut INCOSE sollten 20 – 30% des gesamten Projektbudgets in das Systems Engineering fließen [52].

In vielen Projekten fehlt die in Abbildung 1.3⁶ gezeigte Organisationseinheit *Systems Engineering*. Ihre Aufgaben werden von der Gesamtprojektleitung und den einzelnen Entwicklungsabteilungen wahrgenommen. Auch wenn die Gesamtprojektleitung in-

Organigramm

*Implizites
Systems Engineering*

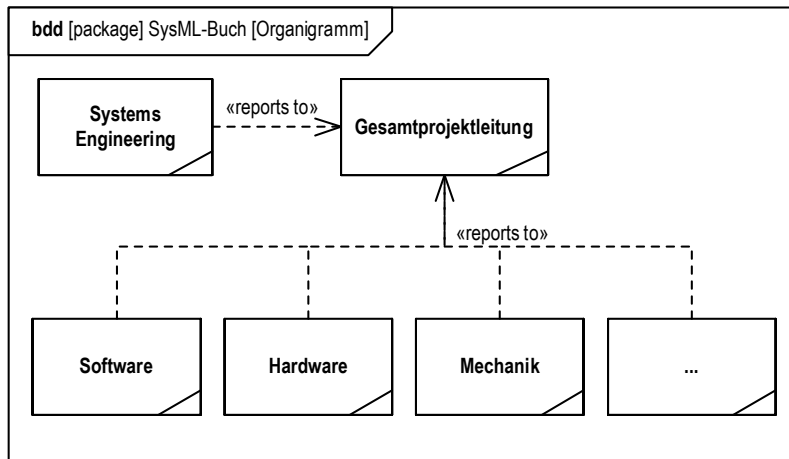


Abbildung 1.3
*Organisatorische
Einordnung
Systems Engineering*

haltlich in der Lage wäre, die Aufgaben des Systems Engineering zu übernehmen, fehlt neben den Leitungsaufgaben die notwendige Zeit. Die Entwicklungsabteilungen kommunizieren nur mit ihren unmittelbaren Nachbardisziplinen, zu denen Schnittstellen existieren. Dadurch ist zwar ein gewisser Weitblick vorhanden,

⁵Der Begriff *Architekt* ist allerdings schon sehr überstrapaziert.

⁶Die Abbildung zeigt ein Organigramm in UML-Notation. Näheres zur Geschäftsprozessmodellierung mit UML finden Sie in [42]. Das Diagramm ist ein Blockdefinitionsdiagramm (bdd). Sie werden in Kapitel 4.5 beschrieben.

aber keine ganzheitliche Systemsichtweise. Das führt zu den in der Praxis beobachtbaren Problemen, dass leicht der Überblick über komplexe Systeme verloren geht.

*Historische
Strukturen*

Ein häufiges Szenario ist, dass die Disziplin, die (meist) historisch bedingt im Unternehmen dominiert, die Aufgaben des Systems Engineering übernimmt. Spätestens sobald andere Disziplinen an Bedeutung zunehmen, entstehen in dieser Konstellation Konflikt- und Missverständnispotenziale. Beispielsweise ein Maschinenbauer mit einer zunehmend dominierenden Softwaredisziplin.

*Gesamtlösung
fokussieren*

Die Entwicklungsabteilungen des Projekts lösen Teilprobleme des Gesamtproblems. Jede Teillösung ist Auswahl aus einer Menge mehrerer Lösungsvarianten. Die Auswahlkriterien sind ohne die Rolle des Systemingenieurs geprägt von der jeweiligen Disziplin. Wechselwirkungen zwischen den Lösungsvarianten der einzelnen Entwicklungsabteilungen bleiben unbemerkt. Das angewendete Teile-und-Herrsche-Prinzip funktioniert nur, wenn jemand das Gesamtproblem im Fokus hat und dafür sorgt, dass die Summe der besten Teillösungen die beste der möglichen Gesamtlösungen ergibt. Also erst Herrschen, dann Teilen. Das ist eine der Aufgaben des Systems Engineering.

Prozesse einführen

Die Einbettung des Systems Engineering in die Projektstruktur zeigt, dass es nicht ohne Unterstützung aller Abteilungen – insbesondere des Managements – eingeführt werden kann. Neben der organisatorischen Veränderung ist es auch eine Änderung der Projektkultur. Klare Regeln und für alle Stakeholder transparente Aufgabenbeschreibungen sind wichtige Werkzeuge bei der Einführung eines Entwicklungsprozesses. Es kann nur funktionieren, wenn es sowohl von oben (Management) als auch von unten (Entwicklung) getrieben wird und man sich in der Mitte trifft bzw. sehr nahe kommt.

1.2.4 Historie des Systems Engineering

*Damals vor 5000
Jahren...*

Die Menschheit hat schon vor über 5000 Jahren Systeme entwickelt. Damals haben die Ägypter ihre imposanten Pyramiden gebaut. Von Systems Engineering hat zu der Zeit noch niemand gesprochen. Auch wenn dies zum Teil bereits als Beginn der Disziplin angesehen wird, machen wir einen großen Zeitsprung nach vorne zum Anfang des 20. Jahrhunderts und setzen dort die Geburtsstunde des Systems Engineering. Die industrielle Revolution hat viele Systeme hervorgebracht, die eher mit dem Begriff des Systems Engineering in Verbindung gebracht werden: Autos,

Flugzeuge, Maschinen. Den Ingenieuren war das Systems Engineering aber nach wie vor fremd. Ein Chefindenieur war in der Lage, das gesamte System zu überblicken, und die Teildisziplinen konnten relativ autonom entwickelt werden, da die Schnittstellen einfach waren.

Das heutige Systems Engineering hat sich Ende der fünfziger Jahre entwickelt. In dieser Zeit hat die Menschheit Systeme in Angriff genommen, deren Komplexität und Projektdurchführung alles bisher Dagewesene übertrafen. Das Wettrennen der Weltmächte im Weltraum oder das militärische Wettrüsten hat Projekte ins Leben gerufen, die auf der einen Seite sehr komplexe Systeme entwickeln mussten und auf der anderen Seite gezwungen waren, schnell und erfolgreich zu sein. Dieser immense Druck hat dazu geführt, dass Methoden des Systems Engineering entwickelt wurden.

50er Jahre

Auch im kommerziellen Bereich wurden die Systeme komplexer und der Bedarf nach ganzheitlichen Methoden größer. Aus dem Telekommunikationsbereich stammt beispielsweise eine frühe Publikation von Arthur Hall »*Methodology for Systems Engineering*« von 1962 [25].

Langsam, aber stetig ist die Bedeutung des Systems Engineering angewachsen. Es sind nicht mehr nur die Riesenprojekte aus der Luft- und Raumfahrt, sondern auch »gewöhnliche« Projekte, die die Techniken dieser Disziplin einsetzen. Im Jahre 2002 ist der Prozessrahmen ISO/IEC 15288 [30] eingeführt worden, der die Prozesse entlang des Lebenszyklus eines Systems beschreibt. Damit ist das Systems Engineering auch formal eine anerkannte Disziplin in der Entwicklung von Systemen.

ISO/IEC 15288

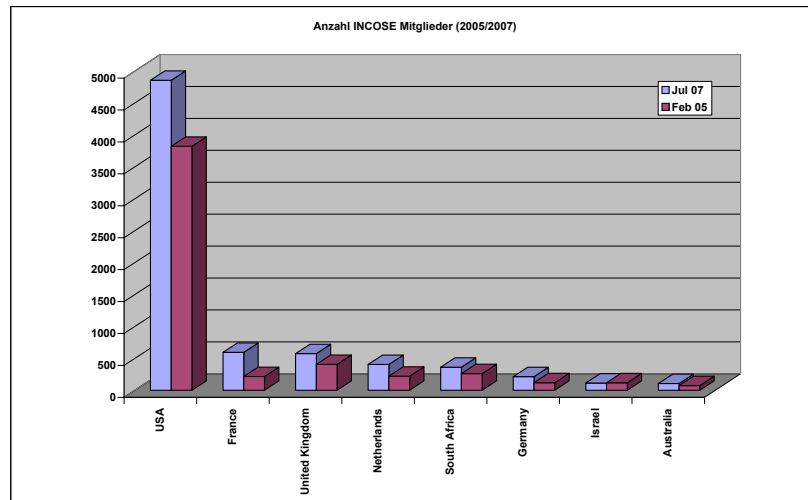
Aufgrund der Historie dominiert das Systems Engineering nach wie vor in den USA und in den Domänen Luft- und Raumfahrt sowie in militärischen Projekten.

1.2.5 International Council on Systems Engineering

1990 wurde in den USA die Organisation *National Council on Systems Engineering* (NCOSE) gegründet. Die Ziele waren die Entwicklung und Förderung des Systems Engineering in den USA. Nur fünf Jahre später sind diese Ziele auf internationale Ebene ausgedehnt worden, und aus NCOSE wurde das *International Council on Systems Engineering* (INCOSE).

Die über 5000 Mitglieder kommen vorwiegend aus den USA (> 4800 Mitglieder), Frankreich und Großbritannien (jeweils > 500 Mitglieder). Aus Deutschland kommen ca. 200 Mitglieder (Abb. 1.4). Zu den amerikanischen Mitgliedern gehören beispielsweise Mitarbeiter der Firmen NASA, Boeing und General Motors. Zu den deutschen Vertretern zählen z. B. Siemens, BMW und oose Innovative Informatik GmbH.

Abbildung 1.4
INCOSE-
Mitgliederverteilung
(2005/2007)



INCOSE ist in sogenannte Chapters unterteilt. In Deutschland ist das INCOSE-Chapter die *Gesellschaft für Systems Engineering e. V. (GfSE)*.

1.2.6 Systems Engineering in Deutschland

*Implizites
Systems Engineering*

Deutschland hat hinsichtlich erfolgreicher Produktentwicklung weltweit eine bedeutende Stellung. Die Vermutung liegt nahe, dass das Systems Engineering konsequent betrieben wird. Näher betrachtet ist aber festzustellen, dass die methodischen Ansätze des Systems Engineering angewendet werden, die Disziplin bisher aber keine hervorgehobene Rolle einnimmt. Nur selten findet man explizit einen etablierten Systems-Engineering-Prozess, eine Systems-Engineering-Abteilung oder einen Systemingenieur entsprechend der oben genannten Definition.

Tendenz steigend

Mein persönlicher Eindruck aus einer Vielzahl von Projekten, die ich im Rahmen meiner Berater- und Trainertätigkeit kennengelernt habe, ist, dass es ein zunehmendes Bewusstsein für die Notwendigkeit des Systems Engineering gibt. Eine Kennzahl, die

meine Wahrnehmung untermauert, ist die deutlich steigende Zahl an Mitgliedern in der GfSE, dem German Chapter von INCOSE (Abschnitt 1.2.5). Ihre Anzahl hat sich von 2005 auf 2007 verdoppelt (Abb. 1.4).

In anderen Ländern – allen voran die USA – ist die Rolle des Systemingenieurs häufiger explizit benannt. Die USA sind auch der Treiber dieser Disziplin. Große, ehrgeizige Projekte seit den 50er Jahren wie die Mondlandung und das militärische Wettrüsten haben dort die Notwendigkeit einer neuen Vorgehensweise erfordert. In Großprojekten fehlte bis dahin die systemische Denkweise, d. h. das strukturierte Vorgehen verschiedener Disziplinen, um das gemeinsame Ziel zu erreichen. Im Rahmen dieser Entwicklung hat sich auch in den Unternehmen die konkrete Rolle des Systemingenieurs gebildet.

USA

1.2.7 Systems Engineering vs. Software Engineering

»Was hat Systems Engineering mit Software Engineering zu tun?« Diese Frage höre ich häufiger, vorwiegend von Softwareentwicklern. Die Antwort steht eigentlich schon in den vorherigen Abschnitten. Ich möchte es hier dennoch einmal explizit erwähnen.

Software Engineering, Hardware Engineering, Verfahrenstechnik, Automatisierungstechnik usw. sind Disziplinen, die bestimmte Bausteine des Systems entwickeln. Das Systems Engineering betrachtet das System ganzheitlich: die Gesamtarchitektur, das korrekte Zusammenspiel der Systembausteine, die Anforderungen und deren Umsetzung und Überprüfung sowie neben der Entwicklung auch die anderen Lebensphasen eines Systems wie beispielsweise Betrieb und Entsorgung.

Querschnittsdisziplin

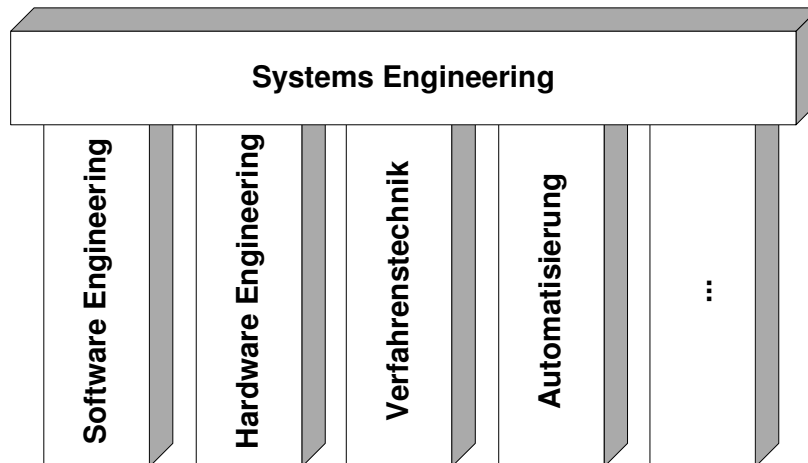
Abbildung 1.5 verdeutlicht die Querschnittsfunktionalität des Systems Engineering. Sie zeigt insbesondere, dass das Systems Engineering das Software Engineering nicht ersetzt. Es wird sich in die Belange der Software nur insofern einmischen, dass Anforderungen vorgegeben werden, um die bestmögliche Integration in das Gesamtsystem zu erwirken.

1.2.8 Randnotizen

Die allgemeine Systemtheorie stammt von dem Biologen Ludwig von Bertalanffy. Er hat Gemeinsamkeiten aus unterschiedlichen Wissensgebieten herausgearbeitet und in seinem Buch [64] beschrieben. Im Sinne der allgemeinen Systemtheorie sind technische Systeme nur ein mögliches Anwendungsfeld von vielen. Wei-

Systemtheorie

Abbildung 1.5
Systems-Engineering-
Disziplinen



tere wären beispielsweise Wirtschaftswissenschaften, Soziologie und Psychologie. Die zwischenmenschlichen Beziehungen können genauso als ein System betrachtet werden wie die Weltwirtschaft oder ein Automobil. In diesem Buch betrachten wir nur Systeme, die von Menschenhand gemacht werden können.

Abstraktion

Damit befindet sich die allgemeine Systemtheorie auf einer sehr abstrakten Ebene. Jeweils auf ein konkretes Wissensgebiet übertragen, können viele wichtige konkrete Werkzeuge und Methoden abgeleitet werden. Das systemische Denken erlaubt, mit Systemen zu agieren, ohne Detailwissen über die einzelnen Systembausteine zu haben.

1.3 Die Sprachen OMG SysML und UML

Das zentrale Thema dieses Buches ist die Modellierung. Und für Modelle sind Modellierungssprachen notwendig.

UML
⇒ S. 199
OMG
⇒ S. 199

Im Bereich der Softwareentwicklung hat sich die *Unified Modeling Language* (UML) als Modellierungssprache etabliert. Es ist ein weltweiter Standard, der von der *Object Management Group* (OMG) spezifiziert wird. Die UML ist auch als ISO-Standard anerkannt (ISO/IEC 19501).

Standard gesucht

Trotz zahlreicher Initiativen, die Prozesse des Systems Engineering zu standardisieren (z. B. Prozess ISO/IEC 15288, EIA-632), hat sich bisher keine einheitliche Modellierungssprache ergeben. Das führt zu erheblichen Reibungsverlusten in interdisziplinären Projekten. Informationen in Form von Modellen können

nur schlecht kommuniziert werden, Missverständnisse treten auf, verschiedene Werkzeuge sind notwendig und so fort.

Im Jahr 2001 hat INCOSE sich zum Ziel gesetzt, die UML zu einer Standardsprache des Systems Engineering zu machen. Die UML erfüllt im Wesentlichen die Anforderungen, ist verbreitet, an spezifische Bedürfnisse anpassbar, und es gibt eine Vielzahl an Modellierungswerkzeugen, Literatur sowie Beratungs- und Seminaranbietern. Aufgrund des integrierten Erweiterungsmechanismus (Stereotypen) können neue Modellierungsvokabeln definiert und somit die UML an bestimmte Domänen und Disziplinen angepasst werden. Mit der Version UML 2.0⁷ ist die Bandbreite der Einsatzmöglichkeiten gegenüber der UML1 noch größer geworden (siehe z. B. [42]).

Die Anpassung der UML für das Systems Engineering hat den Namen *Systems Modeling Language* (OMG SysML™) und basiert in der Version 1.0 auf der UML 2.1.1. Die wichtigsten Erweiterungen sind:

- ❑ Die Klassen der UML werden in SysML Systembausteine genannt, das Klassendiagramm *Blockdefinitionsdiagramm*. Das UML-Kompositionsstrukturdiagramm heißt in SysML *Internes Blockdiagramm*. Für beide Diagrammformen sind etliche Erweiterungen definiert.
- ❑ Informationsobjektflüsse im internen Blockdiagramm
- ❑ Unterstützung der *Enhanced Functional Flow Block Diagrams* (EFFBD) und kontinuierlicher Funktionen durch Aktions- und Objektknoten im Aktivitätsdiagramm
- ❑ Zwei neue Diagrammarten: Anforderungsdiagramm, Zusage-/Zusicherungsdiagramm
- ❑ Unterstützung des Datenformats ISO AP-233, um Datenaustausch zwischen verschiedenen Werkzeugen zu ermöglichen
- ❑ Explizites Herausstreichen von UML-Elementen, die im Systems Engineering nicht benötigt werden, z. B. die softwarelastigen Komponenten

Berechtigt ist die Frage, ob diese Erweiterungen überhaupt notwendig gewesen sind, oder anders gefragt: »Warum noch eine Modellierungssprache?« Die UML ist zwar sehr mächtig, weist aber hinsichtlich des Systems Engineering entscheidende Defizite wie die fehlende Anforderungsmodellierung auf. Ein weiterer Grund,

UML für Systems Engineering

OMG SysML™
⇒ S. 299

Warum noch eine »ML«?

⁷Ich verwende im Buch die Bezeichnung UML1 bzw. UML2 für die Versionen 1.x bzw. 2.x der UML. Wenn ich eine spezielle Version anspreche, gebe ich sie vollständig an, beispielsweise UML 2.0.

der die Notwendigkeit von SysML begründet, ist die Tatsache, dass UML zum Teil softwarelastig und stark von der Objektorientierung geprägt ist, im Systems Engineering aber disziplinenneutral modelliert wird. Die Verwendung der UML kann leicht zu Akzeptanzproblemen und Missverständnissen in der Projektkommunikation führen.

OMG SysML™ 1.0

Die SysML ist im April 2006 in der Version 1.0 von der OMG als Standard angenommen worden. Etliche Hersteller unterstützen bereits die Sprache. Hierzu gehören ARTiSAN Software Tools, EmbeddedPlus Engineering, Telelogic, NoMagic und Sparx Systems. Eine aktuelle Übersicht über SysML-Modellierungswerkzeuge finden Sie auf der Homepage des Buchs unter <http://www.system-modellierung.de>.

Ein Jahr lang hat die *Finalization Task Force* (FTF) am Feinschliff der Version 1.0 gearbeitet. Im April 2007 ist OMG SysML™ 1.0 als offizieller Standard der OMG verabschiedet und im September 2007 veröffentlicht worden.

OMG SysML 1.1

Seitdem wird an der Version 1.1 gearbeitet. Die Erhöhung der zweiten Versionsnummer zeigt, dass nur Fehler behoben und geringfügige Änderungen vorgenommen werden. Umfangreiche Änderungen der Sprache können erst mit der Version 2.0 eingeführt werden, die derzeit noch nicht geplant ist. Die SysML 1.1 wird voraussichtlich Anfang 2009 veröffentlicht.

SysML/UML im Buch

Das in Kapitel 2 beschriebene Vorgehen verwendet die SysML. Die Kapitel 3 und 4 beschreiben die Sprachen SysML und UML getrennt voneinander. Somit erfahren Sie, welche Elemente aus der UML stammen und welche Erweiterungen SysML einführt. Mit diesem Wissen können Sie auch Systems Engineering nur mit UML betreiben und das in diesem Buch beschriebene Vorgehen ohne die SysML-Erweiterung anwenden. Das Kapitel 5 führt weitere Elemente in Form eines Profils (SYSMOD) ein, die im Rahmen des Vorgehens in diesem Buch verwendet werden.

Profil
⇒ S. 290

1.4 Buchkontext

Ich möchte Ihnen das unmittelbare Umfeld dieses Buches kurz vorstellen, damit Sie es besser einordnen und gegenüber ähnlichen Themen abgrenzen können. Abbildung 1.6 zeigt in Anlehnung an das in Kapitel 2.3 beschriebene Systemkontextdiagramm ein Buchkontextdiagramm⁸.

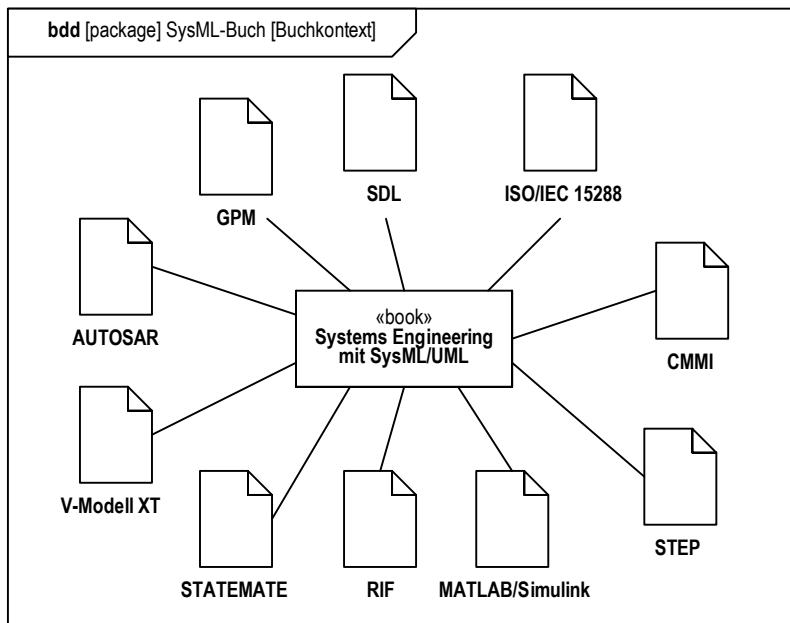


Abbildung 1.6
Buchkontext

1.4.1 AUTOSAR

AUTOSAR steht für *Automotive Open System Architecture*. Dahinter steht eine internationale Organisation, deren Ziel ein offener Standard für Elektronik-Architekturen im Automobil ist. Mitglieder sind im Wesentlichen Automobilhersteller und -zulieferer. Eine Grundlage von *AUTOSAR* ist u. a. das Projekt *EAST-EEA*.

Das Ziel von *AUTOSAR* ist die bessere Austauschbarkeit von Komponenten in der Automobilelektronik zwischen Zulieferer und Hersteller sowie zwischen verschiedenen Produktlinien. Die dafür notwendige Vereinheitlichung findet unter Beibehaltung des Wettbewerbs statt. Berücksichtigt werden die Bereiche Karosserie-

Internationales
Projekt

Ziele

⁸Das Stereotyp «book» ist natürlich nicht Teil des SYSMOD-Profiles (Kap. 5), und das Buchkontextdiagramm gehört nicht zum Vorgehen.

Elektronik, Antrieb, Fahrwerk, Sicherheit, Multimedia-Systeme, Telematik sowie die Mensch-Maschine-Schnittstelle.

ITEA-Projekt

Die Abkürzung EAST-EEA steht für *Electronics Architecture and Software Technologies – Embedded Electronic Architecture* [15]. Es ist ein Projekt im Rahmen des europäischen *ITEA*-Programms (*Information Technology for European Advancement*). Beteiligt an EAST-EEA sind Automobilhersteller und -zulieferer. Die Ergebnisse des Projekts bilden die Basis von AUTOSAR.

Modellierungssprache

Im Rahmen der Architektur ist die *EAST-ADL (Architecture Description Language)* entstanden. Sie ist ein Profil der UML2 zur Modellierung elektronischer Systeme im Automotive-Bereich. Schwerpunkte sind Anforderungsmodellierung, Durchgängigkeit über mehrere Abstraktionsebenen sowie Validierung und Verifikation.

EAST-ADL ist in sechs Bereiche aufgeteilt:

- Struktur
- Verhalten
- Anforderungen
- Validierung und Verifikation
- Support
- Varianten

Für jeden dieser Bereiche werden Sprachkonstrukte zur Verfügung gestellt.

SysML und EAST-ADL

Damit ergeben sich viele Überschneidungen mit den Fähigkeiten und Zielen der Sprache SysML. Erste Ansätze, die Sprachen näher zusammenzubringen, liegen vor. So ist die Anforderungsmodellierung von EAST-ADL eine Erweiterung des SysML-Ansatzes, allerdings basierend auf der SysML-Version 0.3. Da SysML allgemeiner ist – unabhängig von der Automobilbranche –, wird die Sprache auch einen höheren Verbreitungsgrad erreichen. Ich erwarte bzw. hoffe, dass beide Sprachen in Zukunft nicht konkurrierend, sondern gegenseitig ergänzend sind und eingesetzt werden.

SysML und AUTOSAR

SysML und AUTOSAR können nicht direkt verglichen werden. AUTOSAR ist eine Architektur mit standardisierten Schnittstellenbeschreibungen, Komponenten usw. SysML ist »nur« eine Modellierungssprache. Sie kann verwendet werden, um ein System gemäß der AUTOSAR-Architektur zu beschreiben.

1.4.2 Capability Maturity Model Integration (CMMI)

Dass Softwareentwicklungsprojekte scheitern, galt (gilt?) lange als normal. Der erfolgreiche Abschluss war eine Ausnahme, d. h. Funktionalität wie gefordert, Entwicklungsdauer und -kosten wie geplant. Um dem entgegenzuwirken, hat das amerikanische Verteidigungsministerium die Entwicklung des Qualitätsmanagementmodells *Capability Maturity Model (CMM)* angestoßen, damit sie ihre Auftragnehmer besser bewerten können. Entstanden ist CMM Mitte der achtziger Jahre am *Software Engineering Institute (SEI)* der Carnegie Mellon University in Pittsburgh.

Qualität

Das CMM definiert fünf Stufen, die die Qualität einer Organisation und ihrer Prozesse auszeichnen. Betrachtet wird beispielsweise die Projektplanung, das Risiko- und das Anforderungsmanagement.

CMM

Im Jahr 2000 ist der Nachfolger des CMM, das *Capability Maturity Model Integration (CMMI)*, veröffentlicht worden. Hier sind die Erfahrungen aus der Anwendung des CMM mit eingeflossen. Neben der Softwareentwicklung beleuchtet CMMI nun auch das Systems Engineering.

CMMI

Die Sprache SysML kann nicht direkt mit CMMI verglichen werden, da es zwei unterschiedliche Dinge sind. Gute SysML-Modelle und die Prozesse, sie zu erstellen, helfen, die Qualitätskriterien des CMMI zu erfüllen. Beispielsweise wird von CMMI die Verfolgbarkeit von Anforderungen gefordert, was sich sehr gut in SysML abbilden lässt.

SysML

Verfolgungsbeziehung
⇒ S. 317

1.4.3 GPM

Das Systems Engineering setzt sich nicht nur mit Abläufen innerhalb eines (technischen) Systems auseinander. Wichtig sind auch die Prozesse im Umfeld des Systems. Welche Arbeitsabläufe sind in der Entwicklung, in der Produktion, im Betrieb und in der Entsorgung von dem System betroffen?

Geschäftliche Abläufe

Die Modellierung dieser Abläufe ist das Feld der Geschäftsprozessmodellierung (GPM)⁹. Statt technischer Systeme werden hier geschäftliche Systeme – also Unternehmen – modelliert, entwickelt und optimiert. Zwischen den beiden Disziplinen gibt es nicht nur Berührungspunkte, sondern auch viele Parallelen. Die Werkzeuge und Vorgehensweisen der Modellierung sind sich auf abstrakter Ebene in vielen Bereichen ähnlich [42].

GPM

⁹engl. *Business Process Modeling (BPM)*

1.4.4 ISO/IEC 15288

*Hybrides
Prozessrahmenwerk*

Die Norm ISO/IEC 15288 ist entwickelt worden, um ein Rahmenwerk für Prozesse zur Entwicklung technischer Systeme anzubieten, das Software und Hardware jeweils denselben Stellenwert einräumt. Als Anfang der neunziger Jahre die Arbeiten an der Norm angestoßen wurden, gab es ein derartiges hybrides Prozessrahmenwerk noch nicht. Basis ist die Norm ISO/IEC 12207¹⁰, die sich nur auf Software bezieht.

Das Rahmenwerk ist für kleine und für große Unternehmen gleichermaßen geeignet. Ebenso ist es unabhängig von einer spezifischen Domäne. Entsprechend allgemein sind die Vorgaben der Norm und erfordern, dass sie für ein konkretes Projekt adaptiert werden.

V-Modell XT

Das V-Modell[®] XT [61] bietet eine Konventionsabbildung für die Norm ISO/IEC 15288. Damit werden Begriffe und Konzepte der beiden Standards in Beziehung gesetzt, z. B. wird der ISO/IEC-15288-Prozess *Stakeholder Requirements Definition* auf die Aktivitätsgruppe *Anforderungen und Analysen* aus dem V-Modell XT abgebildet.

Die Norm beschreibt fünf Prozessbereiche:

1. Akquisitionsprozesse
2. Unternehmensprozesse, z. B. Qualitätssicherung, Ressourcenmanagement
3. Projektprozesse, z. B. Projektplanung, Risikomanagement, Controlling
4. Technische Prozesse, z. B. Anforderungsanalyse, Architektur, Implementierung, Betrieb, Entsorgung
5. Sonstige, z. B. Tailoring

SysML

SysML ist eine mögliche Sprache, um Ergebnisse der Aktivitäten in den Prozessen zu beschreiben.

1.4.5 MATLAB/Simulink

*Entwicklungs-
umgebung und
Sprache*

MATLAB[™] (*Matrix Laboratory*) ist eine proprietäre Entwicklungsumgebung und Programmiersprache zur Visualisierung, Berechnung und Programmierung mathematischer Ausdrücke. Die Anwendung wird von der Firma *The MathWorks* entwickelt.

Simulation

Simulink[™] ist eine Erweiterung von *MATLAB* zur Modellierung, Simulation und Analyse dynamischer Systeme. Hierfür werden Blockschaltbilder verwendet. *Stateflow* ist eine Erweiterung,

¹⁰SPICE ist ein Bewertungsverfahren zur Norm ISO/IEC 12207.

die die Modellierung und Simulation endlicher Zustandsautomaten ermöglicht.

MATLAB/Simulink ist ein weit verbreitetes Werkzeug. Trotz seiner Fähigkeiten ist es ein großer Nachteil, dass es ein proprietäres System ist und kein Standard wie beispielsweise SysML. Ein SysML-Modellierungswerkzeug steht nicht in direkter Konkurrenz zu MATLAB. Auch wenn es Überschneidungen gibt, beispielsweise im Bereich der Zustandsmodellierung, können sich beide Umgebungen auch ergänzen. SysML ist mächtiger im Bereich der Anforderungsmodellierung und der Systemarchitektur, während MATLAB/Simulink im Simulationsbereich seine Stärken hat. Wenn Sie beide Umgebungen einsetzen, benötigen Sie eine Werkzeugkette, um die Durchgängigkeit Ihrer Modelle nicht zu verlieren. Eine Werkzeugkette zwischen SysML und Modelica ist in [32] beschrieben.

SysML

1.4.6 Requirement Interchange Format (RIF)

Das *Requirement Interchange Format* (RIF) ist auf Initiative der Automobilindustrie¹¹ entstanden. Ziel ist der Austausch von Anforderungen zwischen Automobilhersteller und Zulieferer. Trotz dieses Hintergrunds ist das Datenaustauschformat unabhängig von der Automotive-Domäne und kann auch in anderen Bereichen eingesetzt werden.

*Anforderungen
austauschen*

Die Auftraggeber/Auftragnehmer-Konstellation ist ein typisches Szenario, in dem Anforderungen ausgetauscht werden müssen. Zwischen beiden Rollen liegt üblicherweise die Unternehmensgrenze, und der Zugriff auf eine gemeinsame Anforderungsdatenbank ist nur selten möglich. Die Reibungsverluste und somit verursachte Fehler, Kosten, Zeitverzug und Missstimmung überschreiten schnell die Schmerzgrenze. Und wenn etwas weh tut, ist es an der Zeit, es zu ändern¹².

RIF schließt die Lücke, um Anforderungen über Werkzeug- und Unternehmensgrenzen hinweg austauschen zu können. Es beschreibt ein generisches Format, um Anforderungen abzulegen. Neben der Anforderung selbst sind beispielsweise Gruppen, Hierarchien, Beziehungen und Zugriffsrechte beschreibbar.

¹¹Urheber ist die *Herstellerinitiative Software* (HIS), in der sich Audi, BMW, Daimler Chrysler, Porsche und Volkswagen zusammengefunden haben. Als weiterer Automobilhersteller ist auch die Adam Opel AG an der RIF-Spezifikation beteiligt.

¹²Kent Beck hätte es wohl Geruchsgrenze genannt (*If it stinks, change it*) [2].

SysML Das RIF-Modell ist in UML beschrieben, die Implementierung ist in XML. Der Import/Export in bzw. aus einem SysML-Modell ist somit möglich. In der SysML-Arbeitsgruppe der OMG wird bereits darüber diskutiert, SysML und RIF offiziell kompatibel zueinander zu gestalten.

1.4.7 STATEMATE

Zustandsautomaten *STATEMATE* ist ein grafisches Modellierungswerkzeug der Firma I-Logix zur Entwicklung eingebetteter Systeme. Es ist verbreitet in der Automotive- und Luftfahrtbranche. Das zentrale Modell in *STATEMATE* sind Zustandsautomaten. Es basiert auf einer Arbeit von David Harel, der Mitbegründer der Firma I-Logix ist [26].

SysML *STATEMATE* ist vor der UML entwickelt worden. Mit der UML hat I-Logix das Modellierungswerkzeug *Rhapsody* veröffentlicht, das viele der Funktionen von *STATEMATE* enthält. *Rhapsody* ist auch ein SysML-Modellierungswerkzeug. Im Jahr 2006 ist I-Logix von der Firma Telelogic übernommen worden, die derzeit wiederum von IBM zwecks Übernahme beäugt wird.

1.4.8 STEP

ISO-Baukasten *STEP* beschreibt eine Serie von ISO-10303-Standards und steht für *Standard for the Exchange of Product model data*. Es ist eine Art Baukasten bestehend aus mehreren Dokumenten. Hierzu gehören:

- ❑ die Sprache *EXPRESS* zur Beschreibung von objektorientierten Datenmodellen
- ❑ Implementierungsmethoden zur Umsetzung der Datenmodelle, z. B. ein Textformat (ISO 10303-21), XML-Format (ISO 10303-28) oder eine API (ISO 10303-22)
- ❑ Basismodelle für Datenklassen, z. B. Produktidentifikation und -konfiguration (ISO 10303-41), visuelle Darstellung (ISO 10303-46) oder mathematische Beschreibungen (ISO 10303-51)
- ❑ Anwendungsmodelle als Erweiterung der Basismodelle, z. B. für finite Elemente Methoden (ISO 10303-104) oder Kinematik (ISO 10303-105)
- ❑ Anwendungsprotokolle zur Beschreibung von Produktdaten unter einem spezifischen Aspekt, z. B. ISO AP-214 zur Beschreibung von Produktdaten im Automotive-Bereich (ISO 10303-214)

Im Rahmen von STEP wird auch das Anwendungsprotokoll ISO AP-233 für Systems-Engineering-Daten entwickelt. Es enthält Elemente zur Beschreibung von

- ❑ Anforderungen
- ❑ Funktionalen und strukturellen Daten
- ❑ Physikalischen Strukturen
- ❑ Konfigurationsdaten
- ❑ Projekt- und Datenmanagementdaten

Das ISO AP-233-Team hat gemeinsam mit der OMG und INCOSE die Anforderungen für SysML aufgestellt und sich an der Entwicklung von SysML beteiligt. SysML ist abgestimmt mit ISO AP-233, so dass SysML-Modelle über ISO AP-233 in andere Systems-Engineering-Werkzeuge übertragen werden können und umgekehrt. Der Modellaustausch kann gemäß der STEP-Implementierungsmethoden über XMI (*XML Metadata Interchange*) oder über eine API erfolgen.

SysML

1.4.9 Specification and Description Language (SDL)

Im Telekommunikationsbereich ist die *Specification and Description Language* (SDL) entwickelt worden [45]. Sie wird von der *International Telecommunication Union* (ITU) als Standard herausgegeben. Inzwischen ist SDL auch außerhalb der Telekommunikationsbranche im Einsatz, beispielsweise zur Entwicklung medizinischer Systeme oder in der Luft- und Raumfahrt.

Telekommunikation

Die Sprache SDL hat viele Gemeinsamkeiten mit der UML und somit auch SysML. Beispielsweise stammen die Sequenzdiagramme von den *Message Sequence Charts* (MSC) der SDL ab [46]. Die vielen Gemeinsamkeiten erlauben eine Abbildung von SDL-Modellen in SysML/UML-Modelle [50].

SysML

1.4.10 V-Modell[®] XT

Das V-Modell ist ein Vorgehensmodell entwickelt im Auftrag der Bundesrepublik Deutschland zum Planen und Durchführen von Systementwicklungsprojekten. Ganz im Sinne des Systems Engineering wird dabei der gesamte Lebenszyklus des Systems berücksichtigt.

Vorgehensmodell

Das aktuelle V-Modell XT aus dem Jahr 2004 basiert auf dem Vorgänger V-Modell 97. Die Überarbeitung ist angestoßen worden, da das alte V-Modell nach 7 Jahren nicht mehr dem aktuellen

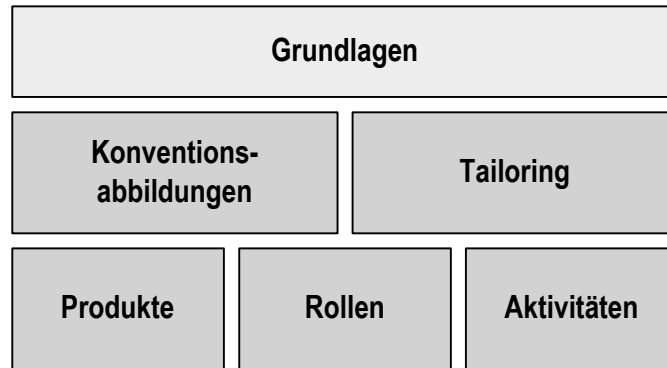
XT

Stand in Projekten entsprach. Neueste Techniken und Methoden wurden nicht ausreichend unterstützt.

Werkzeugkasten

Das V-Modell XT ist ein Werkzeugkasten bestehend aus definierten Rollen, Produkten und Aktivitäten (Abb. 1.7). Damit kann das Vorgehen spezifisch an ein Projekt angepasst werden. Das »XT« im Namen steht für *Extreme Tailoring*. Regeln sorgen dafür, dass das zusammengestellte Vorgehen schlüssig und konsistent ist.

Abbildung 1.7
Überblick V-Modell
XT



Das V-Modell direkt mit SysML zu vergleichen ist wie der berühmte Vergleich zwischen Äpfeln und Birnen. SysML ist eine Sprache und enthält keine Anleitungen, wie sie in Projekten eingesetzt werden kann. Das V-Modell enthält die Anleitungen. Für die Ergebnisse, die dabei erstellt werden, kann in etlichen Bereichen SysML verwendet werden. Das in diesem Buch beschriebene Vorgehen SYSMOD deckt Teile des V-Modells ab.