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).

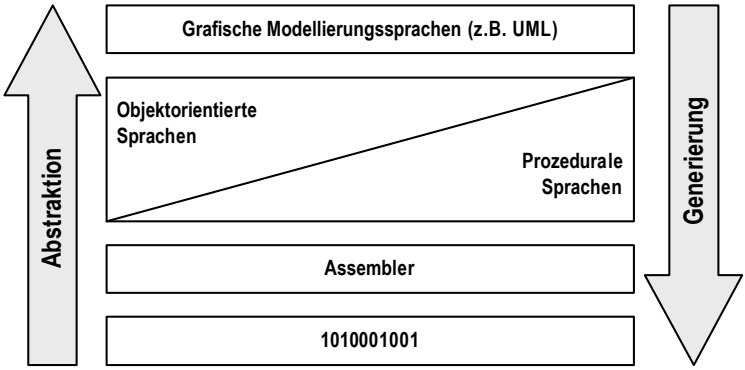


Abbildung 1.1 *Steigende Abstraktion der Programmiersprachen*

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

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.

OMG
⇒ S. 201

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.

UML und SysML

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.

Profil SYSMOD
Stereotyp
⇒ S. 299

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.

Seitenrand

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

1.1.5 Wohin geht der Weg?

Ausbreitung der UML

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?

Lingua franca?

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.

Kompatible Sprachfamilie

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.

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

1.2 Systems Engineering

Unterschiedliche Disziplinen, unterschiedliche Methoden

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 Charaktere

Die vielen Witze über die Eigenarten von Physikern, Ingenieuren, Informatikern, Mathematikern & Co. machen deutlich, dass bei interdisziplinären Projekten unterschiedliche Welten aufeinanderstoßen[4]. Missverständnisse und Konflikte sind vorprogrammiert.

Gute Einzellösung, schlechte Gesamtlösung

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.

1+1=3

1+1 ergibt also 3. Die Entwicklung disziplinenspezifischer 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

[4] Ein 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.«

bewährte 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 [56].

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 [7]. 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 disziplinenspezifischer 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 Einzelelementen nicht erreicht werden kann. Ein Baustein kann aus Software, Hardware, Personen oder beliebigen anderen Einheiten bestehen (nach [73]).

Große Systeme

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, bestehend aus Flughäfen und Luftfahrtverkehrsstraßen.

Modellgröße vs. Systemgröße

Bei der Größenbetrachtung des Systems müssen Sie deutlich zwischen der Größe des realen Systems und der Größe des Systemmodells unterscheiden. Das 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 Luftfahrtverkehrssystems verwenden wir aber als kleinste Einheiten beispielsweise Flugzeuge und Flughäfen. Dieses Beispiel ist übrigens ein typisches System of Systems (SoS).

SoS
⇒ S. 192

Definition Engineering

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 Systems Engineering

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

Definition

Das **Systems Engineering** ist ein interdisziplinärer Ansatz und konzentriert sich auf die Definition und Dokumentation der Systemanforderungen in der frühen Entwicklungsphase, die Erarbeitung einer Systemarchitektur und die Überprüfung des Systems auf Einhaltung der gestellten Anforderungen unter Berücksichtigung des Gesamtproblems: Betrieb, Zeit, Test, Erstellung, Kosten & Planung, Training & Wartung und Entsorgung (nach [73]).

Metadisziplin

Das Systems Engineering integriert alle Disziplinen und beschreibt einen strukturierten Entwicklungsprozess vom Konzept über die Produktions- bis zur Betriebsphase und abschließend wieder die Außerbetriebnahme des Systems. Es werden sowohl die technischen als auch die wirtschaftlichen Aspekte betrachtet, um ein System zu entwickeln, das den Benutzerbedürfnissen entspricht. Diese ganzheitliche Denkweise schließt beispielsweise auch Lösungen zu Problemen ein, die erst durch die Einführung eines neuen Systems entstehen.

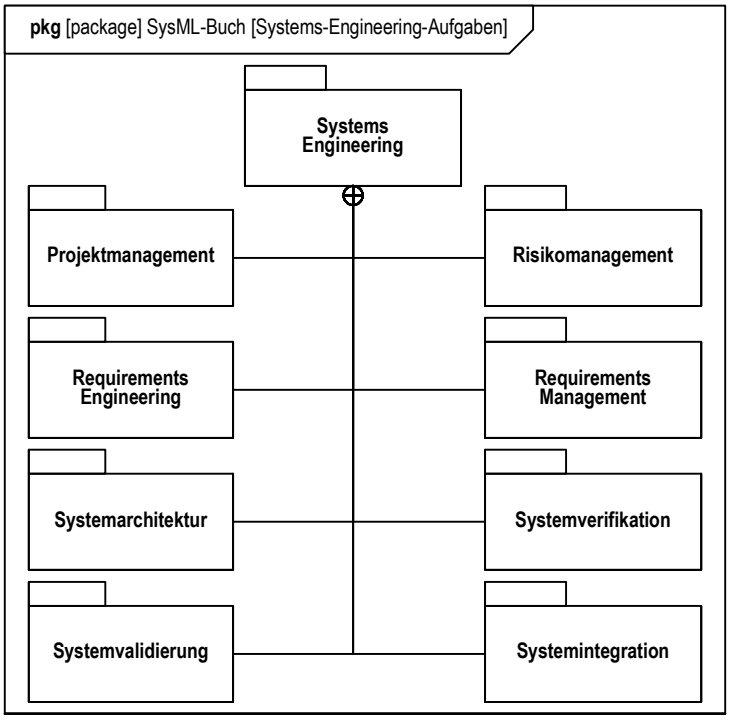


Abbildung 1.2
Disziplinen des Systems Engineering

Paket
⇒ S. 296

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.

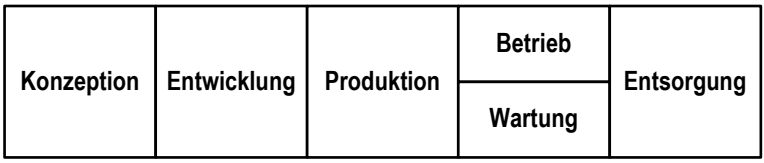


Abbildung 1.3 *Lebensphasen eines Systems*

Systemlebensphasen

Systems Engineering betrachtet den gesamten Lebenszyklus eines Systems. Abbildung 1.3 zeigt die im Standard ISO/IEC 15288 definierten Phasen eines Systems [73]. Das Lebensphasenmodell beschreibt die zeitlichen Abschnitte Entwicklung, Realisierung, Nutzung und Entsorgung. Eine Herausforderung 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?

Beispiel

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 viele weitere Fragen sind zu beantworten.

Problemlösungszyklus

Der Problemlösungszyklus beschreibt den Weg vom Auftreten des Problems bis zur Lösung [41]. Die grobe Struktur besteht aus drei Schritten, die nicht nur linear, sondern auch mit Rückkopplungen durchlaufen werden:

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 ausge-

lassen oder in Punkt 2 nur eine Lösung betrachtet. Das Vorgehen in diesem Buch beginnt mit der Formulierung des Ziels der Systementwicklung (Abschnitt 2.3). Die Analyse und die Architektur des Systems in einem Modell unterstützen die Betrachtung unterschiedlicher Varianten, sodass die optimale Lösung ausgewählt werden kann (siehe auch Abschnitt 2.10.2).

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 folgende Aufgaben:

- ***S**tate the problem*
- ***I**nvestigate alternatives*
- ***M**odel system*
- ***I**ntegrate*
- ***L**aunch the system*
- ***A**ssess performance*
- ***R**e-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 Systems Engineer beschäftigt sich mit Anforderungen auf Systemebene.

SYSMOD-Zickzackmuster
⇒ S. 182

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

Alternative Lösungen prüfen (Investigate alternatives)

Eine wichtige Aufgabe des Systems Engineers 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 Systems Engineer entwirft – basierend auf dem Anforderungsmodell – also nicht nur eine Systemarchitektur, sondern üblicherweise auch alternative Architekturen. 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 Parameter abgeleitet, um die notwendigen Aspekte unmittelbar sichtbar, vergleichbar und am Modell messbar zu machen. SysML bietet hierzu beispielsweise das Zusicherungsdiagramm an (Abschnitt 4.6).

Systemmodell erstellen (Model system)

Modelle werden bereits beim Abwägen der alternativen Lösungsmöglichkeiten erstellt. Jetzt wird das Modell der ausgewählten Lösung detailliert.

Die Modellierung mit SysML ermöglicht eine gute Verfolgbarkeit (*Traceability*) von Aspekten, beispielsweise die Umsetzung einer Anforderung (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 Systemschnittstellen.

System implementieren (Launch the system)

Das System wird gemäß dem spezifizierten Architekturmodell 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 Zusam-

menhang 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 Systems Engineer

Bindeglied

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

Organigramm

Organisatorisch ist die Systems-Engineering-Disziplin häufig ähnlich einer Stabsstelle eingeordnet. Sie berichtet direkt an die Gesamtprojektleitung, die wiederum direkt mit den anderen Entwicklungsabteilungen kommuniziert. Der Systems Engineer ist nicht der Vermittler zwischen Projektleitung und den Entwicklungsabteilungen. 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 [73].

In vielen Projekten fehlt die in Abbildung 1.4[5] gezeigte Organisationseinheit *Systems Engineering*. Ihre Aufgaben werden von der Gesamtprojektleitung und den einzelnen Entwicklungsabteilungen wahrgenommen.

Implizites Systems Engineering

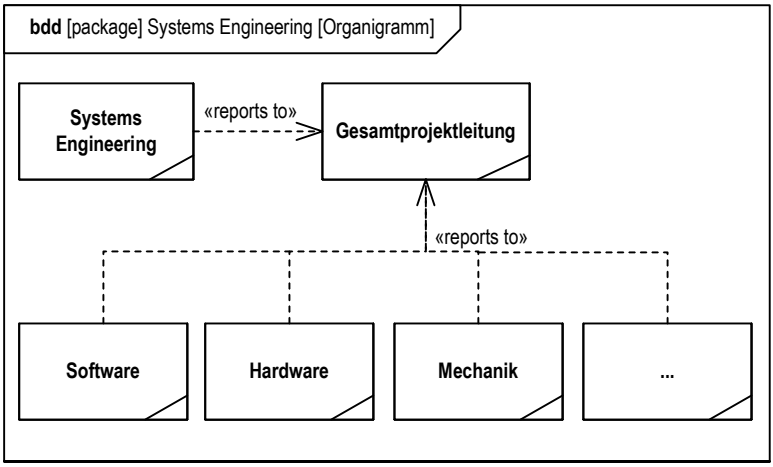


Abbildung 1.4 *Organisatorische Einordnung des Systems Engineering*

Auch wenn die Gesamtprojektleitung inhaltlich in der Lage wäre, die Aufgaben des Systems Engineering zu übernehmen, fehlt neben den Leitungsaufgaben die notwendige Zeit. Die Entwicklungsabteilungen kommunizieren explizit nur mit ihren unmittelbaren Nachbardisziplinen, zu denen Schnittstellen existieren. Dadurch ist zwar ein gewisser Weitblick vorhanden, 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 Systems Engineers geprägt von der jeweiligen Disziplin. Wechselwirkungen zwischen den Lösungsvarianten der

[5] Die Abbildung zeigt ein Organigramm in UML-Notation. Näheres zur Geschäftsprozessmodellierung mit UML finden Sie in [61].

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.

Tendenziell ist die Disziplin Systems Engineering auf dem Vormarsch und wird zunehmend als eigenständige Organisationseinheit und in Prozessen realisiert.

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

50er-Jahre

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.

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 [42].

ISO/IEC 15288

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 [49] 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.

Aufgrund der Historie dominiert das klassische Systems Engineering nach wie vor in den USA und in den Domänen Luft- und Raumfahrt sowie in militärischen Projekten. Insbesondere die SysML hat dem Thema modellbasiertes Systems Engineering (MBSE) großen Vorschub geleistet. Ein Überblick über MBSE finden Sie in Abschnitt 1.3.

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 9500 Mitglieder kommen vorwiegend aus den USA (ca. 5600 Mitglieder), Großbritannien (ca. 820 Mitglieder) und Frankreich (ca. 510 Mitglieder). Aus Deutschland kommen ca. 420 Mitglieder (Abb. 1.5). Zu den amerikanischen Mitgliedern gehören beispielsweise Mitarbeiter der Firmen NASA, Boeing und General Motors. Zu den deutschen Vertretern zählen Mitarbeiter der Firmen Siemens, BMW und oose.

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

Weitere Informationen zu INCOSE und der GfSE finden Sie auf den Webseiten *www.incose.org* und *www.gfse.de*.

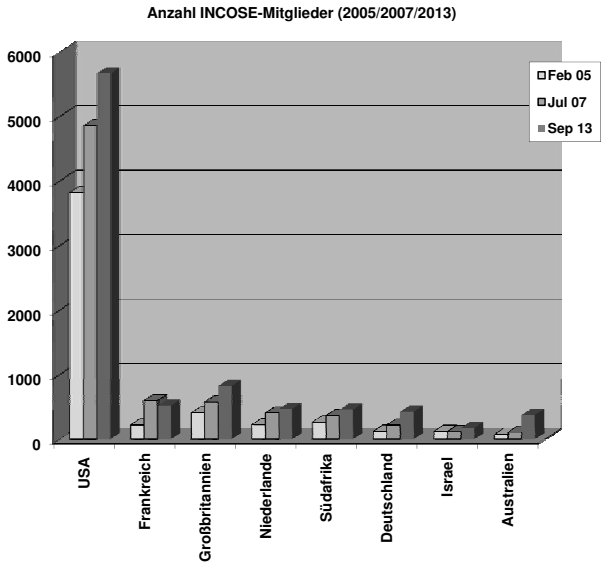


Abbildung 1.5 *INCOSE-Mitgliederverteilung (2005/2007/2013)*

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 zwar 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 Systems Engineer entsprechend der oben genannten Definition. Glücklicherweise steigen aber die Zahlen und Systems Engineering ist in Deutschland deutlich auf dem Vormarsch.

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. Ihre Anzahl hat sich von 2005 bis 2013 verdreifacht.

Aus- und Weiterbildung

Mit zunehmender Bedeutung des Systems Engineering steigt auch die Anzahl der Aus- und Weiterbildungsangebote. Neben Beratungs- und Trainingsanbietern wie die Firma oose listet die Webseite der GfSE[6] über 30 Hochschulangebote[7] auf.

Systems Engineering Leadership

Ich arbeite aktuell gemeinsam mit der FH Campus02 aus Graz am berufsbegleitenden Masterstudiengang *Systems Engineering Leadership*[8]. Er setzt auf den positiven Erfahrungen des Masterstudiengangs *Software Engineering Leadership*[9] auf. Ziel ist es, die Studierenden für Führungsrollen im Systems Engineering zu befähigen.

1.3 Modellbasiertes Systems Engineering (MBSE)

Schlüsseltechnologie

In many respects, the future of systems engineering can be said to be model-based.

Das ist die Aussage von INCOSE in ihrem Papier zur Vision 2020 [69]. Modellierung ist eine Schlüsseltechnologie zur Entwicklung komplexer Systeme. Die Modelle halten die Informationen konsistent zusammen und ermöglichen verschiedene Sichten mit unterschiedlichen Abstraktionsebenen. Das ist notwendig, da eine einzelne Person die Komplexität mit all ihren interdisziplinären Abhängigkeiten nicht mehr bewältigen kann. INCOSE definiert MBSE wie folgt:

Definition

Modellbasiertes Systems Engineering (MBSE) ist die formalisierte Anwendung der Modellierung, um die Aktivitäten zu Systemanforderungen, Architektur, Analyse, Verifikation und Validierung von Beginn der konzeptuellen Architekturphase über die Entwicklung bis zu den späteren Phasen des Systemlebenszyklus zu unterstützen (nach [69]).

[6] *www.gfse.de*
[7] Stand Februar 2012.
[8] *www.systems-engineering-leadership.de*
[9] *www.software-engineering-leadership.de*

MBSE=SysML?

Die Definition klingt auf den ersten Blick eingängig. MBSE ist vereinfacht formuliert, Systems Engineering mit einem Systemmodell. Auf den zweiten Blick stellt sich die Frage, was ein Modell ist. Handelt es sich um ein SysML-Modell? SysML wird in der Definition von INCOSE zu MBSE aber nicht erwähnt. MBSE ist weit mehr als SysML. SysML ist nur eine Sprache, die im MBSE eingesetzt wird.

INCOSE definiert nicht den Begriff des Modells. In SYSMOD definiere ich den Begriff des Systemmodells wie folgt:

Definition

Das Systemmodell im Kontext des MBSE ist das Abbild eines realen oder noch zu entwickelnden Systems, wobei mittels Abstraktion nur die für einen definierten Zweck relevanten Attribute berücksichtigt werden. Das Systemmodell ist gekennzeichnet durch die folgenden Eigenschaften:

- Das Systemmodell darf sich aus mehreren Repositories zusammensetzen, muss aber in sich konsistent sein und sich nach außen wie ein einzelnes Modell verhalten.
- Das Systemmodell erlaubt unterschiedliche Sichten auf die Informationen.
- Das Systemmodell ist maschinell auswertbar und liegt in einer abstrakten Syntax vor, die explizit MBSE-Konzepte wie Anforderungen oder Systemarchitekturen unterstützt.

Nach dieser Definition ist ein einzelnes SysML-Modell ein Systemmodell. Das Gleiche gilt für ein SysML-Modell, das mit einem Anforderungsmodell, einem UML-Modell und einem CAD-Modell verbunden ist. Ein Textverarbeitungsdokument ist kein Systemmodell, sondern nur eine mögliche Sicht auf ein Modell.

In den folgenden Abschnitten gebe ich Ihnen einen kurzen Überblick über eine Auswahl an Modellierungssprachen im Umfeld von MBSE.

1.3.1 Die Sprachen UML und OMG SysML

UML

⇒ S. 201

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).

OMG
⇒ S. 201

Trotz zahlreicher Initiativen, die Prozesse des Systems Engineering zu standardisieren (z. B. 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.

Standard gesucht

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 in der UML integrierten Erweiterungsmechanismus (Stereotypen) können neue Modellierungsvokabeln definiert und somit die UML an bestimmte Domänen und Disziplinen angepasst werden. Mit der Version UML 2.0[10] ist die Bandbreite der Einsatzmöglichkeiten gegenüber der UML1 noch größer geworden (siehe z. B. [61]).

UML für Systems Engineering

Die Anpassung der UML für das Systems Engineering hat den Namen *OMG Systems Modeling Language* (OMG SysML™) und basiert in der aktuellen Version 1.4 auf der UML 2.5. Die wichtigsten Erweiterungen der SysML gegenüber der UML sind:

OMG SysML™
⇒ S. 309

- 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.
- Objektflüsse im internen Blockdiagramm
- Unterstützung der *Enhanced Functional Flow Block Diagrams* (EFFBD) und kontinuierlicher Funktionen durch Aktions- und Objektknoten im Aktivitätsdiagramm.
- Zwei neue Diagrammarten: Anforderungsdiagramm, Zusicherungsdiagramm
- Unterstützung des Datenformats ISO AP-233, um Datenaustausch zwischen verschiedenen Werkzeugen zu ermöglichen.

[10] Ich 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. Wenn ich nur UML schreibe, meine ich die aktuellste Version.

- Explizites Herausstreichen von UML-Elementen, die im Systems Engineering nicht benötigt werden, z. B. die softwarelastigen Komponenten

Warum noch eine »ML«?

Berechtigt ist die Frage, ob diese Erweiterungen überhaupt notwendig gewesen sind, oder anders gefragt: »Warum noch eine Modellierungssprache? Hätte man nicht einfach die UML verwenden können?« Die UML ist zwar sehr mächtig, weist aber hinsichtlich des Systems Engineering entscheidende Defizite wie die fehlende Anforderungsmodellierung auf. Ein weiterer Grund, 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 die Sprache. Hierzu gehören ARTiSAN Software Tools, IBM, NoMagic und Sparx Systems. TOPCASED[11] ist ein Beispiel für ein Open-Source-Modellierungswerkzeug. Eine aktuelle Übersicht über SysML-Modellierungswerkzeuge finden Sie auf der Homepage des Buchs unter `www.model-based-systems-engineering.com`.

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 1.5

Seitdem sind etliche Verbesserungen an der Sprache vorgenommen worden. Aktuell wird an der Version 1.5 gearbeitet. Die Erhöhung der zweiten Versionsnummer zeigt, dass nur Fehler behoben und kleinere Änderungen vorgenommen werden. In der Version 1.3 ist das Portkonzept der SysML auf Basis von Erfahrungen aus dem praktischen Einsatz der SysML überarbeitet worden. Die Version 1.4 führt unter anderem ein überarbeitetes Konzept der Sicht ein, ermöglicht die Gruppierung von beliebigen Elementen und bietet weitere Möglichkeiten für die Modellierung von Einheiten. Umfangreiche Änderungen der Sprache können erst mit der Version 2.0 eingeführt werden, die derzeit noch nicht geplant ist.

SysML/UML im Buch

Das in Kapitel 2 beschriebene Vorgehen wendet die SysML an. Die Kapitel 3 und 4 beschreiben die Sprachen UML und SysML

[11] `www.topcased.org`

getrennt voneinander. Somit erfahren Sie, welche Elemente aus der UML stammen und welche Erweiterungen SysML einführt. In Summe beschreiben die Kapitel 3 und 4 den vollständigen Sprachumfang der SysML. Von der UML stelle ich nur die SysML-relevanten Teile vor. Das Kapitel 5 führt weitere Elemente in Form eines Profils (SYSMOD) ein, die im Rahmen des Vorgehens in diesem Buch verwendet werden. Das Kapitel 6 stellt das Zertifizierungsprogramm *OMG Certified Systems Modeling Professional* (OCSMP) vor und unterstützt bei der Vorbereitung zur Zertifizierungsprüfung.

Profil
⇒ S. 299

1.3.2 BPMN

Geschäftliche Abläufe

Neben den Abläufen innerhalb eines (technischen) Systems sind auch die Prozesse im Umfeld des Systems wichtig. Welche Arbeitsabläufe sind in der Entwicklung, in der Produktion, im Betrieb und in der Entsorgung von dem System notwendig? In welche Abläufe bettet sich das System ein?

Geschäftsprozessmodellierung

Die Modellierung dieser Abläufe ist das Feld der Geschäftsprozessmodellierung, einer Unterdisziplin des Geschäftsprozessmanagements (GPM) [78][12]. Statt technischer Systeme werden hier geschäftliche Systeme – also Organisationen – 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 [61].

Die Modellierungssprache der Geschäftsprozessmodellierung ist die *Business Process Model and Notation* (BPMN). Sie stellt einen standardisierten Satz grafischer Elemente bereit, um geschäftliche Abläufe zu modellieren. Das ermöglicht einen besseren Austausch zwischen Menschen. Gleichzeitig ist BPMN nicht nur eine Notation, sondern auch ein formales Modell, sodass die Prozesse auch ausgeführt werden können.

SysML

Die BPMN ist wie die UML und SysML eine Modellierungssprache der OMG. Etliche Modellierungswerkzeuge zu SysML bieten auch die BPMN-Modellierung an. Dadurch können Informationen aus einem SysML- und einem BPMN-Modell leicht miteinander verknüpft werden.

[12] engl. *Business Process Management.* (BPM) [79]

1.3.3 MATLAB/Simulink

Entwicklungsumgebung 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, die die Modellierung und Simulation endlicher Zustandsautomaten ermöglicht.

SysML

MATLAB/Simulink ist ein weitverbreitetes Werkzeug. Trotz seiner Fähigkeiten hat es den 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, ergänzen sich beide Umgebungen. 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.

1.3.4 Modelica

Physikalische Modelle

Modelica ist eine Sprache zur Beschreibung physikalischer Modelle. Sie kann in verschiedenen Bereichen wie beispielsweise Mechanik, Regelungstechnik und Elektrotechnik eingesetzt werden. Es gibt grafische Entwicklungsumgebungen zu Modelica, um Simulationsmodelle zu erstellen. Ähnlich wie es in MATLAB/Simulink möglich ist. Die Sprache Modelica ist frei verfügbar. Verantwortlich für ihre Entwicklung ist die Modelica Association. Das *Open Source Modelica Consortium* (OSMC) bietet unter `www.openmodelica.org` eine freie Umgebung für Modelica an.

SysML

SysML und Modelica können sich gut ergänzen. Die Stärken der SysML liegen in der Beschreibung komplexer Systeme. Die Stärken von Modelica liegen in der Analyse diskreter und kontinuierlicher zeitlicher Aspekte in komplexen Systemen.

Die *SysML-Modelica Transformation Specification* [37] standardisiert die Abbildung zwischen SysML- und Modelica-Modellen. Damit ermöglicht sie die kollaborative Anwendung beider Modelle in einer Umgebung. Eine Werkzeugkette zwischen SysML und Modelica ist in [55] beschrieben.

1.3.5 Specification and Description Language (SDL)

Telekommunikation

Im Telekommunikationsbereich ist die *Specification and Description Language* (SDL) entwickelt worden [52]. 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.

SysML

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 [53]. Die vielen Gemeinsamkeiten erlauben eine Abbildung von SDL-Modellen in SysML/UML-Modelle [70].

1.4 Randnotizen

Ich möchte Ihnen einige Themen aus dem Umfeld dieses Buchs kurz vorstellen, damit Sie sie besser einordnen und abgrenzen können.

1.4.1 AUTOSAR

Internationales Projekt

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*.

Ziele

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

ITEA-Projekt

Die Abkürzung EAST-EEA steht für *Electronics Architecture and Software Technologies – Embedded Electronic Architecture* [14]. 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 UML 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 konkurrieren, sondern sich gegenseitig ergänzen 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)

Qualität

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

Das CMM definiert fünf Stufen, die die Qualität einer Organisation und ihrer Prozesse auszeichnen. Betrachtet werden 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

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. 327

1.4.3 Industrie 4.0

Das Internet hat die Welt klein und die Märkte groß gemacht. Angefangen mit weltweiter Kommunikation, die es ermöglichte, dass jeder lokale Anbieter seine Produkte nun weltweit vertreiben kann, hat Web 2.0 zu globalen Kooperationen und neuen Möglichkeiten geführt. Jetzt verbinden wir nicht nur Menschen, sondern auch Dinge miteinander. Das *Internet of Things* bringt Geräte und Maschinen ins Internet: vom heimischen Kühlschrank bis zum Maschinenpark einer Fabrik. Die Dinge können cyberphysische Systeme sein. Das sind Systeme mit Netzwerkfähigkeit und physischen Ein- und Ausgabewerten.

Vernetzung

Der Begriff *Industrie 4.0* bezeichnet ein Projekt der deutschen Bundesregierung, das die deutsche Industrie auf die Herausforderungen der neuen globalen vernetzten Welt vorbereiten soll. Die Vision sind intelligente Fabriken, in denen die Werkstücke mit den Maschinen kommunizieren und ihnen erzählen, wie sie bearbeitet werden möchten. Der Mensch wird zum Dirigenten des Maschinenorchesters.

Industrie 4.0

Der Lenkungskreis der Plattform Industrie 4.0 hat den Begriff wie folgt beschrieben:

»Der Begriff Industrie 4.0 steht für die vierte industrielle Revolution, einer neuen Stufe der Organisation und Steuerung der gesamten Wertschöpfungskette über den Lebenszyklus von Produkten. Dieser Zyklus orientiert sich an den zunehmend individualisierten Kundenwünschen und erstreckt sich von der Idee, dem Auftrag über die Entwicklung und Fertigung, die Auslieferung eines Produkts an den Endkunden bis hin zum Recycling, einschließlich der damit verbundenen Dienstleistungen.

Basis ist die Verfügbarkeit aller relevanten Informationen in Echtzeit durch Vernetzung aller an der Wertschöpfung beteiligten Instanzen sowie die Fähigkeit aus den Daten den zu jedem Zeitpunkt optimalen Wertschöpfungsfluss abzuleiten. Durch die Verbindung von Menschen, Objekten und Systemen entstehen dynamische, echtzeitoptimierte und selbst organisierende, unternehmensübergreifende Wertschöpfungsnetzwerke, die sich nach unterschiedlichen Kriterien wie bspw. Kosten, Verfügbarkeit und Ressourcenverbrauch optimieren lassen.«[45]

Revolution

Der Projektname Industrie 4.0 weist auf die 4. industrielle Revolution hin, zu der die zunehmende Vernetzung führt. Die erste industrielle Revolution begann Mitte des 18. Jahrhunderts mit der Einführung mechanischer Produktionsanlagen. Als zweite industrielle Revolution wurde zu Beginn des 20. Jahrhunderts die Einführung der Massenproduktion bezeichnet. Die zunehmende Automatisierung durch IT und Elektronik hat etwa zur Mitte der siebziger Jahre die dritte Revolution eingeleitet. Die Revolution ist kein singuläres Ereignis. Die 4. industrielle Revolution steht uns nicht bevor, wir sind schon mittendrin.

SysML

Die Systeme der Industrie 4.0 sind sehr komplex. Sie haben vielfältige interdisziplinäre Abhängigkeiten. Die Modellierung ist eine geeignete Technik, um diese Komplexität beherrschbar zu machen. MBSE und die Verknüpfung von Modellen verschiedener Disziplinen werden im Kontext von Industrie 4.0 eine wichtige Rolle spielen. Und so hat auch SysML als die Sprache des MBSE eine hohe Bedeutung für Entwicklung der Systeme der Industrie 4.0.

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 12207[13], 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

[13] SPICE ist ein Bewertungsverfahren zur Norm ISO/IEC 12207.

Norm und erfordern, dass sie für ein konkretes Projekt adaptiert werden.

V-Modell XT

Das V-Modell® XT [74] 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 A*nforderungen 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 Product Lifecycle Management (PLM)

Die Definition von PLM ist ähnlich der Definition der Systems-Engineering-Disziplin (Abschnitt 1.2.1):

Definition

Product Lifecycle Management (PLM) ist der Prozess zur Verwaltung des gesamten Lebenszyklus eines Produkts von der Konzeption über die Architektur und Produktion bis zur Wartung und Entsorgung (nach [80]).

Verwaltung vs. Engineering

Die Definitionen sind ähnlich, aber nicht gleich. Der entscheidende Unterschied ist, dass PLM sich um die Verwaltung und Systems Engineering sich um das Engineering von Dingen kümmert. Insofern kann PLM das Systems Engineering hervorragend ergänzen. Die Disziplinen sind unabhängig voneinander entstanden. Seit einigen Jahren finden aber zunehmend mehr Berührungspunkte statt und es wird methodisch und technisch am Zusammenspiel beider Disziplinen gearbeitet (siehe z. B. [15]).

SysML Ein PLM-System kann das physische Repository eines SysML-Modells verwalten, ebenso Verbindungen von Modellelementen in andere Modelle wie ein Anforderungsmodell, ein Modelica-Modell oder ein CAD-Modell. Das PLM-System stellt sicher, dass es konsistente Zusammensetzungen der Modelle in unterschiedlichen Versionen gibt.

1.4.6 Requirement Interchange Format (ReqIF)

Anforderungen austauschen Das *Requirement Interchange Format* (ReqIF) ist auf Initiative der Automobilindustrie entstanden. Ziel war die Förderung des Austauschs von Anforderungen zwischen Automobilhersteller und Zulieferer. Trotz dieses Hintergrunds ist das Datenaustauschformat unabhängig von der Automobil-Domäne und kann auch in anderen Bereichen eingesetzt werden.

Von RIF zu ReqIF Urheber ist die *Herstellerinitiative Software* (HIS), in der sich Audi, BMW, Daimler AG, Porsche und Volkswagen zusammengefunden haben. Als weiterer Automobilhersteller ist auch die Adam Opel AG an der Spezifikation beteiligt. Zu der Zeit wurde das Format noch RIF genannt. 2010 ist RIF zur Standardisierung an die OMG übergeben worden. Da es bereits einen Standard mit dem Namen RIF gab (*W3C Rule Interchange Format*), wurde RIF in ReqIF umbenannt. Die aktuelle Version ist 1.0.1 [34].

Die Auftraggeber/Auftragnehmer-Konstellation ist ein typisches Szenario, in dem Anforderungen ausgetauscht werden müssen. Zwischen beiden Rollen liegt häufig 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 ändern[14].

ReqIF 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.

SysML Das ReqIF-Modell ist in UML beschrieben, die Implementierung beruht auf XML. Der Import/Export in bzw. aus einem SysML-Modell ist somit möglich. In der SysML-Arbeitsgruppe der

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

OMG wird bereits darüber diskutiert, SysML und ReqIF kompatibel zueinander zu gestalten.

1.4.7 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 folgender Daten:

- Anforderungen
- Funktionale und strukturelle Daten
- Physikalische Strukturen
- Konfigurationsdaten
- Projekt- und Datenmanagementdaten

SysML

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, sodass 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.

1.4.8 V-Modell® XT

Vorgehensmodell

Das V-Modell XT 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.

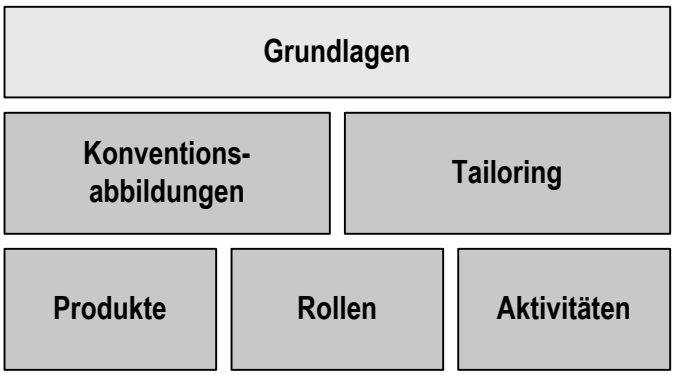
XT

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 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. 1.6). 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 1.6
Überblick V-Modell XT



Allgemeines V-Modell

Im deutschsprachigen Raum ist mit dem Begriff V-Modell entweder das V-Modell XT oder das allgemeine V-Modell gemeint. Im internationalen Umfeld spielt das V-Modell XT keine besondere Rolle. Hier wird in der Regel unter V-Modell nur das allgemeine V-Modell verstanden. Mit dem allgemeinen V-Modell ist die V-artige Struktur des Entwicklungs- und Testastes gemeint (Abb. 1.7).

Vorgeschlagen wurde dieses Vorgehen, das auf dem Wasserfallmodell basiert, zuerst von dem US-amerikanischen Softwareingenieur Barry Boehm im Jahre 1979 [4].

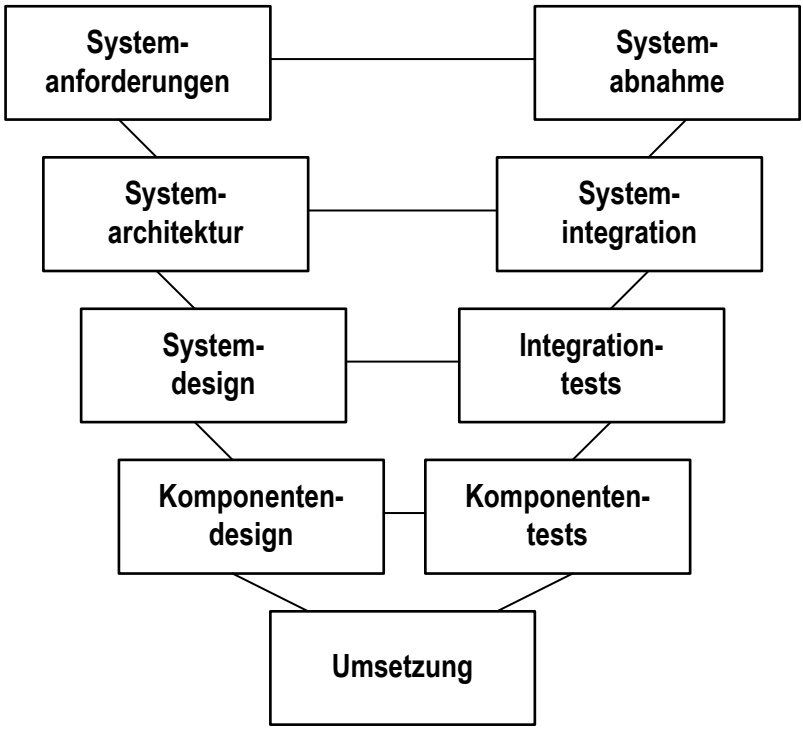


Abbildung 1.7
Allgemeines V-Modell

SysML

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.