Embedded Software Engineering Kongress

# ESE Kongress 2013

## Vortragsskript:

# Systems-Engineering mit der SysML*)

### Wichtige Diagramme, Notationen und Anwendungen im Überblick

Thomas Batt, MicroConsult GmbH, t.batt@microconsult.de

**Dieser Beitrag richtet sich in erster Linie an Systemarchitekten, die einfache aber auch sehr komplexe Systeme mit einer standardisierten Notation spezifizieren und / oder dokumentieren. Ganz speziell für diese Aufgaben hat die OMG (Object Management Group) [2] die SysML (Systems Modeling Language) [3] standardisiert. Die SysML spezifiziert insgesamt neun Diagramme und deren Notationen. Davon stellt dieser Beitrag die wichtigsten vier und deren Anwendung in dem Beispielprojekt der Motorsteuerung vor. Werkzeuge unterstützen den Systemarchitekten bei der Anwendung der SysML. Erfahren Sie, wie ein SysML Modell aufgebaut sein kann.**

**Was ist SysML?**

In der dokumentenzentrierten Entwicklung sind die technischen Projektinformationen in vielen Dokumenten verteilt und häufig nur sehr mühsam aufzufinden. In der modellzentrierten Entwicklung wird nun versucht, all diese Informationen zentral in einem Modell zu halten. Die entsprechenden zuvor erwähnten Dokumente, z.B. die Systemarchitektur-Spezifikation, können dann toolgestützt per Knopfdruck aus dem Modell heraus erzeugt werden. Damit diese Modelle von allen Projektbeteiligen verstehbar sind, müssen die verwendeten Notationen bekannt sein und am besten einem Standard folgen. In der Softwareentwicklung ist hierzu der Einsatz der UML*) (Unified Modeling Language) [4] bereits weit verbreitet. Sie eignet sich aber nur bedingt zur Systemmodellierung.

Aus diesem Grunde existiert der artverwandte SysML (Systems Modeling Language) Standard speziell für die Systemmodellierung.

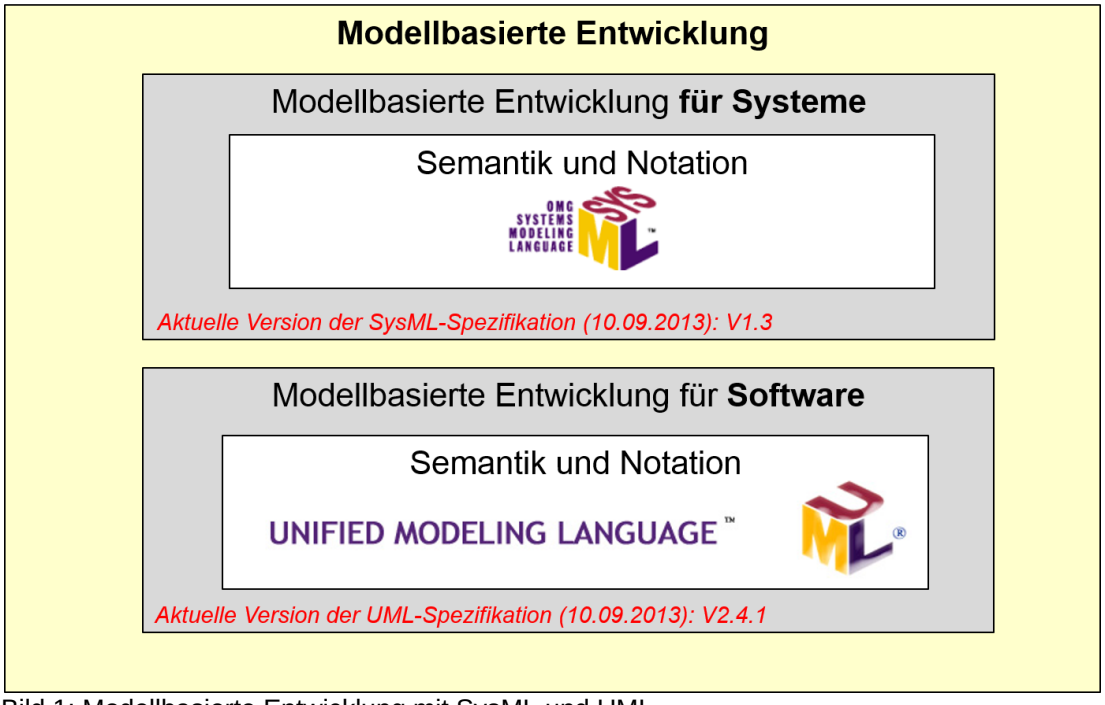


Bild 1: Modellbasierte Entwicklung mit SysML und UML

UML und SysML stellen Semantik (Bedeutung) und Notation (grafische Repräsentation der Bedeutung) in Form von Diagrammen bereit.

Historisch gesehen existierte die UML vor der SysML. Im Bild 2 ist zu erkennen, dass beide Standards teilweise die gleichen Diagramme spezifizieren.

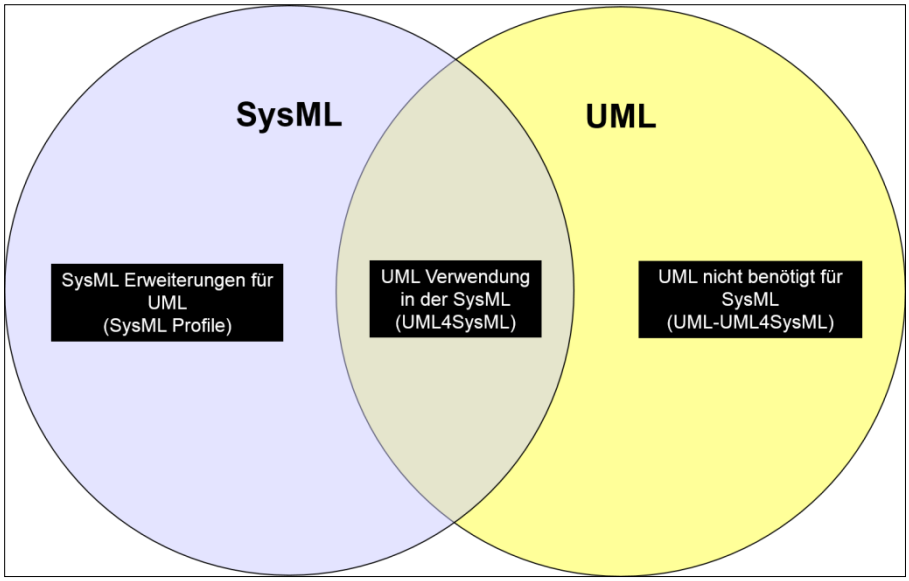


Bild 2: Zusammenhang zwischen SysML und UML

Eine genaue Übersicht der in der SysML standardisierten Diagramme und den Diagrammzusammenhang zur UML zeigt Bild 3.

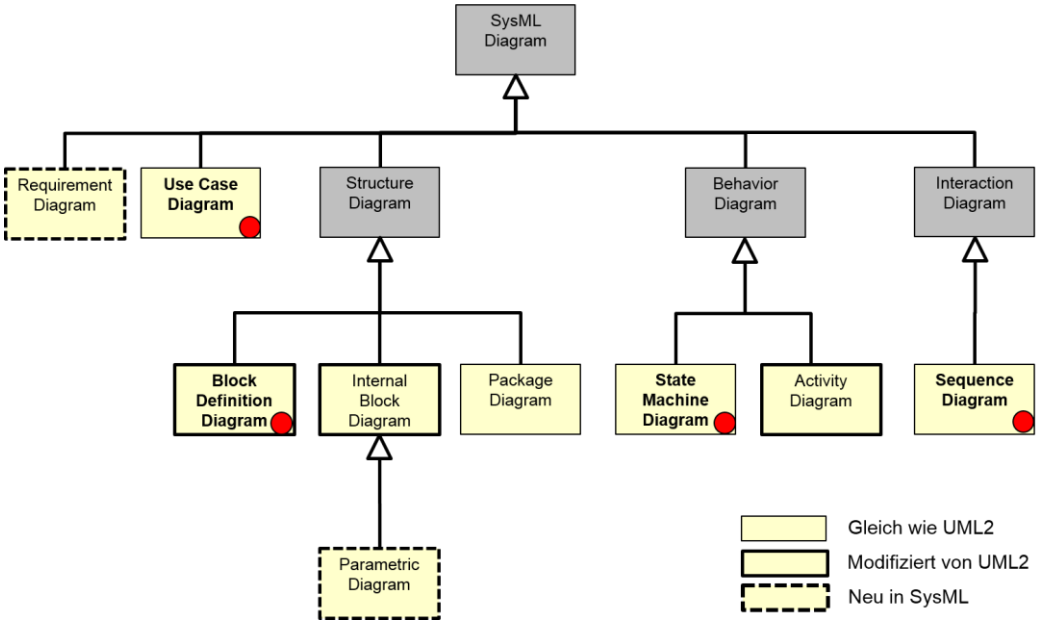


Bild 3: SysML-Diagrammübersicht

Nur die in Bild 3 mit einem Punkt gekennzeichneten Diagramme sind folgend als erster Einstieg in das Thema Systemmodellierung mit der SysML berücksichtigt:
Use-Case-Diagramm, Sequenzdiagramm, Block-Definition-Diagramm und Zustandsfolge-Diagramm.

**Vorstellung Systembeispiel Motorsteuerung**

Zur praktischen Anwendung der SysML-Diagramme soll das Beispielsystem der Motorsteuerung aus Bild 4 dienen. Die Funktionalität des Systems Motorsteuerung lässt sich wie folgt zusammenfassen:

Das System besteht aus einer Motoreinheit, einer Motorregel-Einheit und einer Motoransteuer-Einheit. Die Motoreinheit besteht aus einem Encoder, einem bürstenlosen Gleichstrom-Motor und einem Getriebe. Die Motorregel-Einheit übernimmt die direkte Ansteuerung und Drehzahlregelung des BLCD-Motors. Die Motorsteuer-Einheit verantwortet die Benutzerinteraktion über eine LCD-Touch-Kombination und führt die folgende Applikation aus.

Die Applikation bietet dem Benutzter die Möglichkeit, die Drehgeschwindigkeit des BLDC-Motors mit virtuellen, angezeigten Werten von -7.0 bis +7.0 mit einer Schrittweit von +/-0.5 einzustellen. Das Vorzeichen bestimmt die Drehrichtung und 0 bedeutet Stillstand.

Die Applikation stellt dem Benutzer die zwei übergeordneten Betriebsarten STOP und RUN bereit. Zwischen den beiden Betriebsarten kann der Benutzer mit der STOP/RUN Taste wählen. Im STOP Mode ist der Antrieb deaktiviert. Im RUN Mode ist der Antrieb aktiviert und stellt dem Benutzer die zwei über die MODETaste auswählbaren Betriebsarten DIRECT und PRESELECT bereit.

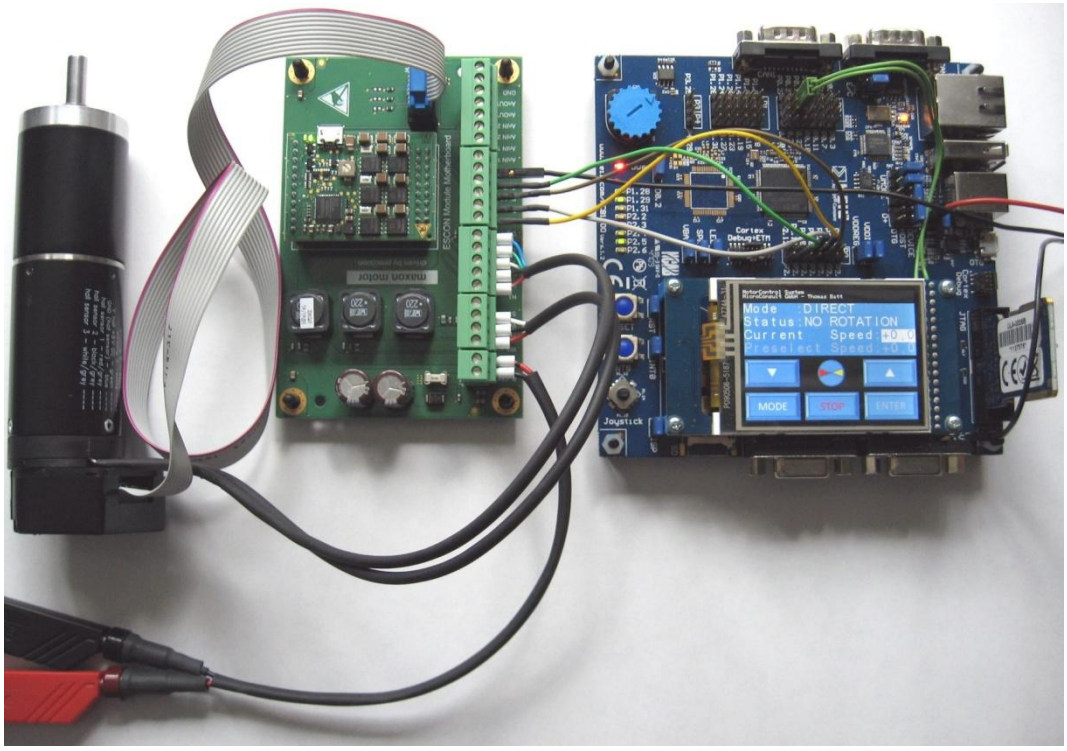


Bild 4: Systembeispiel Motorsteuerung

Im DIRECT Mode wirkt sich das Betätigen der Tasten INCREASE ▲ und DECREASE ▼ unmittelbar auf die aktuelle Drehgeschwindigkeit (Current Speed) des Motors aus. Die ENTER-Taste hat im DIRECT Mode keine Funktionalität. Im PRESELECT Mode kann der Benutzer mit den Tasten INCREASE ▲ und DECREASE ▼ die Vorgabe-Drehgeschwindigkeit (Preselect Speed) bestimmen. Dabei bleibt die Current Speed des Motors zunächst unverändert. Erst mit dem Betätigen der ENTER-Taste wird die Preselect Speed zur Current Speed und damit für den Motor gültig. Mit der Drehgeschwindigkeits-übernahme setzt die Applikation die Preselect Speed automatisch wieder zu Null zurück.

**Use-Case-Diagramm der SysML – Notation**

Im Kontext des System-Entwicklungsprozesses unterstützt das Use-Case-Diagramm die System-Anforderungsanalyse (vgl. Tabelle 1) durch die grafische Visualisierung funktionaler Anforderungen und deren Zusammenhänge.

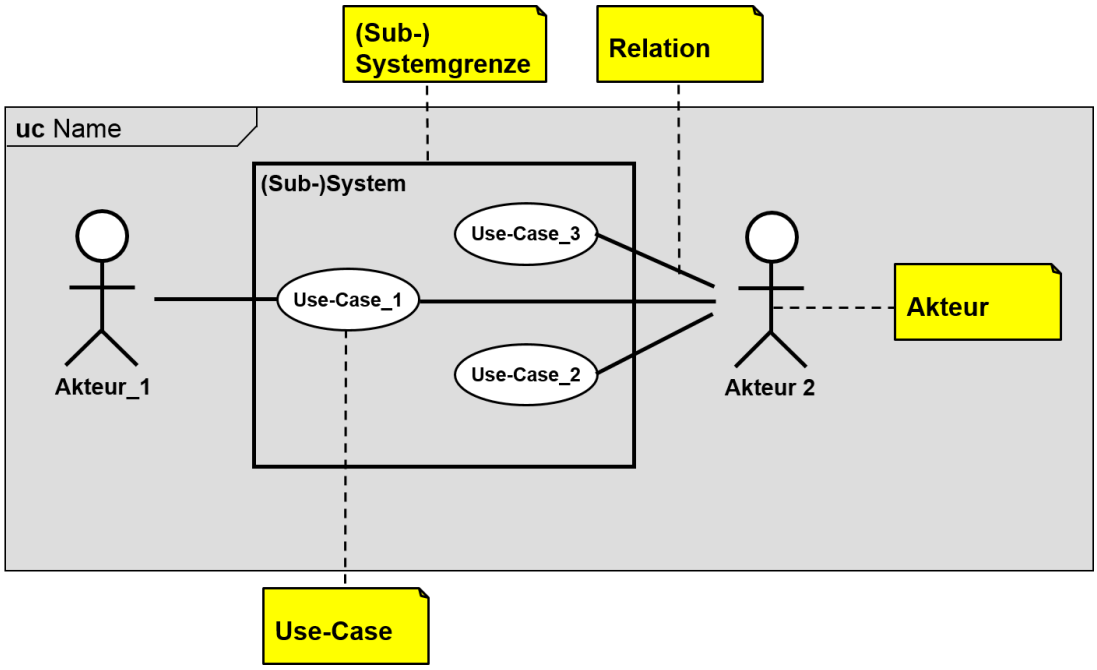


Bild 5: Use-Case-Diagramm Notation

Die (Sub-) Systemgrenze legt exakt das zu betrachtende System fest. Alles, was Teil des Systems ist, befindet sich innerhalb der Grenze. Akteure bilden alle Elemente ab, die mit dem System eine Wechselwirkung haben. Das können z.B. Personen, aber auch andere externe Systeme sein. Ein Use-Case (Anwendungsfall) repräsentiert eine Summe funktionaler Anforderungen. Funktionale Anforderungen spezifizieren, was das System nach außen hin sichtbar leistet. D

amit der Use-Case-Name möglichst aussagekräftig ist, sollte er aus einer Substantiv-Verb-Kombination bestehen. Die Assoziation als Relation zwischen Akteur und System, genauer zwischen Akteur und Use-Case, notiert die Wechselwirkung dazwischen. Hierbei ist eine Richtungsangabe durch Pfeile möglich (unidirektional, bidirektional). In der obigen Darstellung ist die Richtung nicht spezifiziert. Dies ist durch die einfache Verbindungslinie ohne Pfeile dargestellt.

Das Use-Case-Diagramm bietet einige weitere Notationen, um auch komplexere Sachverhalte darzustellen.

**Use-Case Diagramm der SysML - Praxisbeispiel Motorsteuerung**

Die Kontextsicht in Bild 6 zeigt das Umfeld, in das sich das System integriert. Kennzeichen der Kontextsicht ist der eine allumfassende Use-Case, der die komplette zentrale Systemfunktionalität(en) repräsentiert.

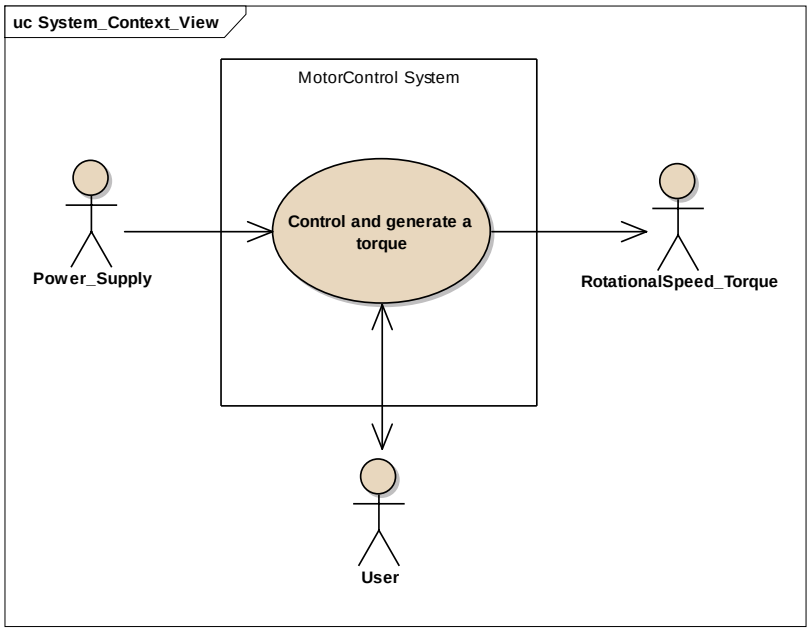


Bild 6: Kontextsicht mit dem Use-Case-Diagramm

In diesem Beispiel wirkt die `Power_Supply` auf das System, der `User` interagiert bidirektional, und das System erzeugt eine `RotationalSpeed_Torque`.

Die funktionale Anforderungssicht verfeinert den zentralen Use-Case der Kontextsicht. Für komplexere Systeme bietet sich eine hierarchische Organisation der funktionalen Anforderungssichten an.

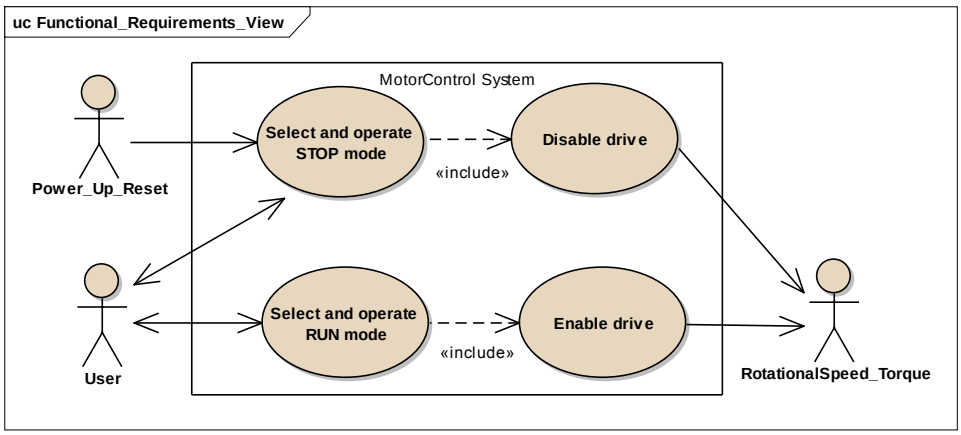


Bild 7: Funktionale Anforderungssicht – oberste Ebene

In der obersten Ebene stehen die Betriebsarten STOP und RUN im Mittelpunkt. Der Use-Case `Select and operate RUN mode` führt ebenfalls den Use-Case `Enable drive` aus, was über die <<include>> Beziehung modelliert ist. Dies gilt in Analogie auch für die beiden anderen Use-Cases.

Bild 8 zeigt das Use-Case-Diagramm, welches den Use-Case `Select and operate RUN mode` detaillierter beschreibt.

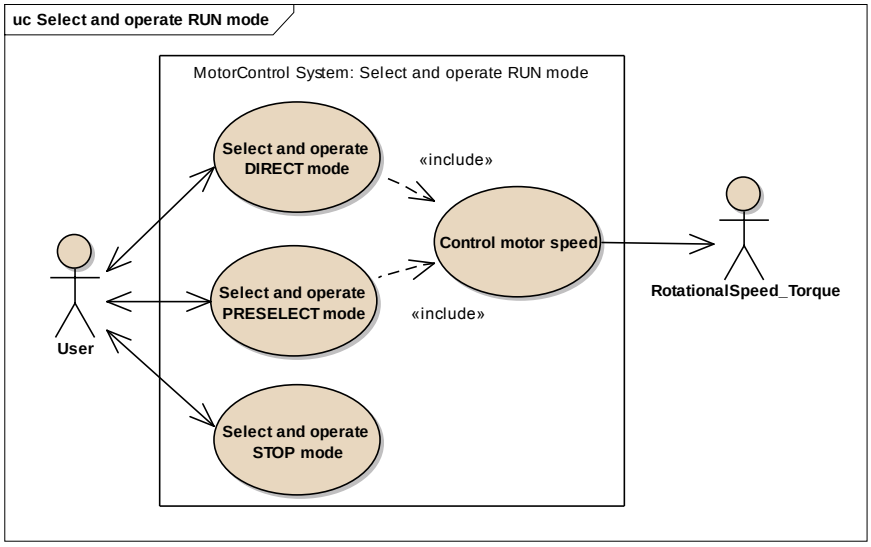


Bild 8: Funktionale Anforderungssicht für den Use-Case `Select and operate RUN mode`

Die Betriebsarten DIRECT und PRESELECT können in weiteren Use-Case-Diagrammen verfeinert werden.

**Sequenzdiagramm der SysML – Notation**

Im Kontext des System-Entwicklungsprozesses unterstützt das Sequenzdiagramm die System-Anforderungsanalyse und System-Architekturanalyse (vgl. Tabelle 1) durch die grafische Darstellung interaktiver und nicht generischer Interaktionen zwischen verschiedenen Elementen. Diese Verhaltensauszüge werden auch als Szenarien bezeichnet. Sie sind z.B. als Testszenarien in den verschiedenen Ebenen der Entwicklung einsetzbar

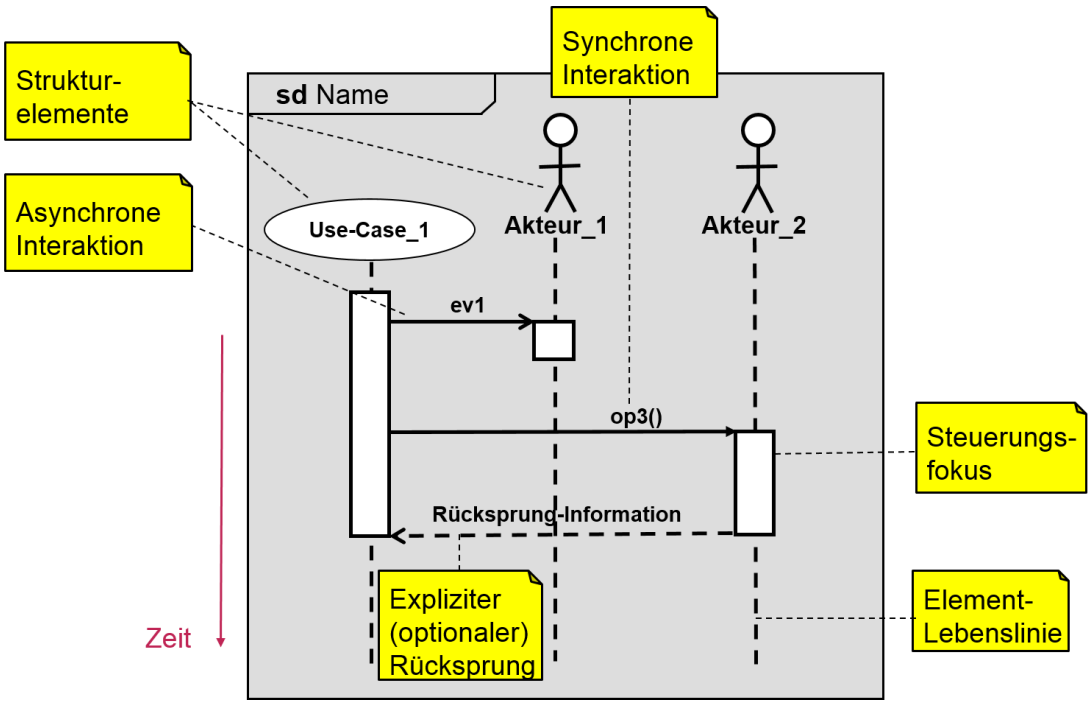


Bild 9: Sequenzdiagramm Notation

Die verwendeten Strukturelemente, die über der Zeit kommunizieren, sind abhängig vom aktuellen Schritt im Entwicklungsprozess (vgl. Tabelle 1). Jedes Strukturelement enthält eine Lebenslinie, die die Lebenszeit des Elementes kennzeichnet. In der System-Anforderungsanalyse sind Use-Cases und Akteure, über die systemexterne Kommunikation darstellbar. In der System-Architekturanalyse und im System-Architekturdesign sind es die nachfolgend vorgestellten Blöcke mit Ihren Ports. Das Sequenzdiagramm unterscheidet zwischen synchroner (ausgefüllter Pfeil) und asynchroner (offener Pfeil) Interaktion. Bei der synchronen Kommunikation wartet der Aufrufer, bis die Interaktion abgeschlossen ist, und arbeitet dann weiter. Der Steuerungsfokus kennzeichnet die Aufrufdauer. Mit dem Abschluss der synchronen Interaktion erfolgt ein Rücksprung, der Informationen an den Aufrufer liefern kann. Bei der asynchronen Kommunikation triggert der Aufrufer (Auslöser) z.B. über Signale oder Ereignisse die Interaktion und arbeitet dann unmittelbar ohne zu warten weiter.

Das Sequenzdiagramm bietet viele weiter mächtige Notationen, um auch komplexeste, nicht generische Abläufe darzustellen.

**Sequenzdiagramm der SysML - Praxisbeispiel Motorsteuerung**

Das Szenario in Bild 10 beschreibt einen konkreten Ablauf für den Use-Case `Select and operate PRESELECT mode`. Das Szenario beschreibt exakt den Ablauf, um von den spezifizierten Vorbedingungen zu den Nachbedingungen zu gelangen – nicht mehr!

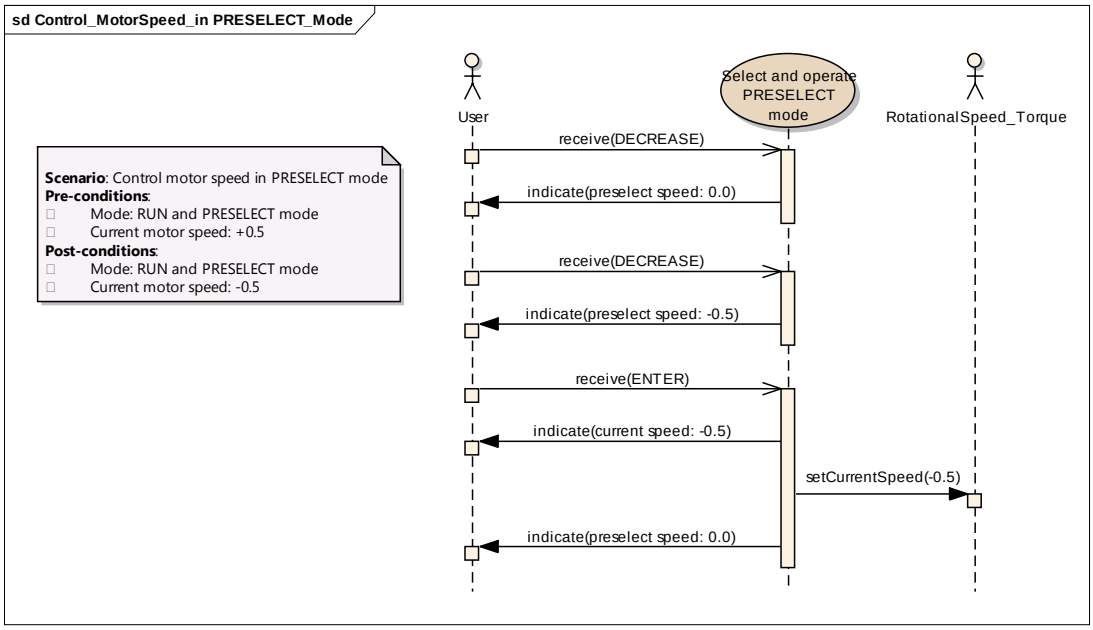


Bild 10: Szenario für den Use-Case `Select and operate PRESELECT mode`

**Block-Definition-Diagramm der SysML – Notation**

Im Kontext des System-Entwicklungsprozesses unterstützt das Block-Definition-Diagramm die System-Architekturanalyse und das System-Architekturdesign (vgl. Tabelle 1) durch die grafische Darstellung von funktionalen und physikalischen Strukturelementen und deren Kommunikationsverbindungen.

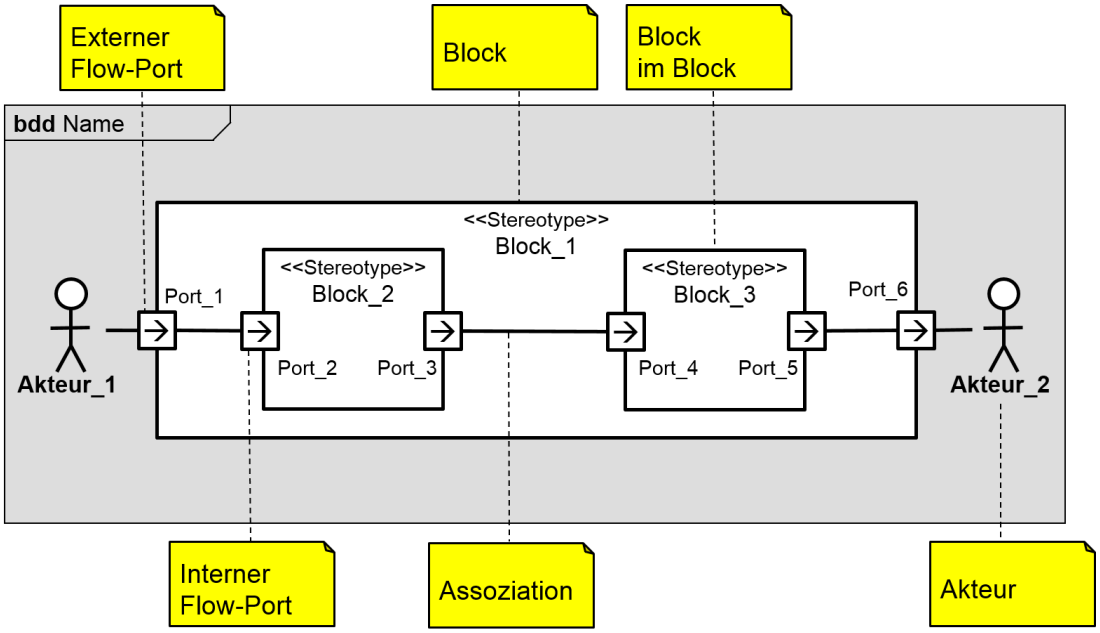


Bild 11: Block-Definition Diagramm Notation

Ein Block repräsentiert ein physikalisches oder funktionales Architekturelement. Die genaue Kennzeichnung ob physikalisch oder funktional könnte beispielsweise über Stereotypen erfolgen. Blöcke können verschachtelt sein, so kann beispielsweise ein physikalischer Block mehrere funktionale Blöcke enthalten. Assoziationen zwischen den Blöcken bestimmen die Kommunikationswege. Zur genaueren Spezifikation der Kommunikationspunkte an den Blöcken dienen die Flow-Ports. Aus dem Use-Case-Diagramm lassen sich die dortigen Akteure als systemexterne Elemente mit Blöcken, wahlweise über Ports, mittels der Assoziation verbinden.

Ist ein Block gleichen Typs mehrfach in einen System enthalten, so lassen sich Blöcke und auch Ports instanziieren. Für physikalische Blöcke bedeutet dies, dass der Block einmal entwickelt, aber mehrfach produziert und mehrfach im System verbaut wird. Auch identische und / oder parametrisierbare funktionale Blöcke können mehrfach zum Einsatz kommen. Das Block-Definition-Diagramm bietet viele weitere mächtige Notationen, um auch komplexeste Systemarchitekturen darzustellen.

## Block-Definition-Diagramm der SysML - Praxisbeispiel Motorsteuerung

Bild 12 zeigt die Systemarchitektur aus der physikalischen Sicht mit den internen Kommunikationswegen, aber auch die Kommunikation zu externen Elementen (Akteuren) ist modelliert.

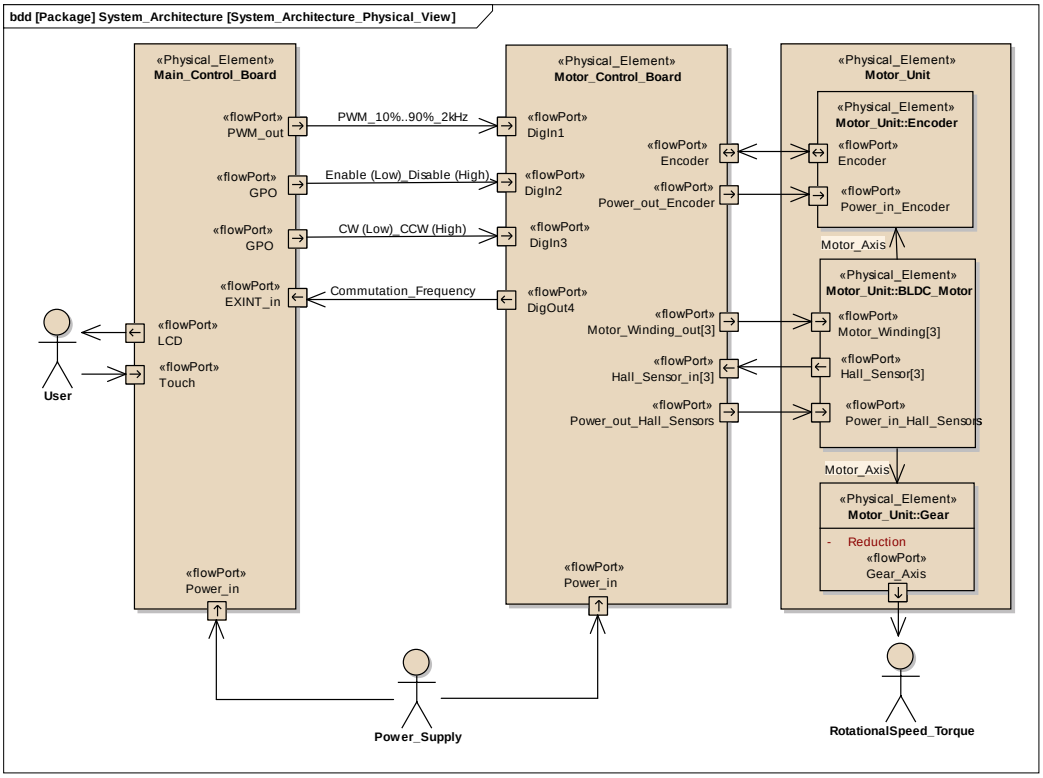


Bild 12: Physikalische Systemarchitektur-Sicht

In diesem Beispiel spezifiziert das Attribut `Reduction` das Übersetzungsverhältnis des Motorgetriebes.

## Zustandsfolge-Diagramm der SysML – Notation

Im Kontext des System-Entwicklungsprozesses unterstützt das Zustandsfolge-Diagramm die **System-Anforderungsanalyse** und das **System-Architekturdesign** (vgl. Tabelle 1) durch die grafische Darstellung generischen Verhaltens einzelner Elemente, z.B. eines Use-Cases oder eines Blocks.

Es ist möglich, die Blöcke mit weiteren Elementen auszustatten, z.B. kann der Systemarchitekt über Attribute die entsprechenden Blockeigenschaften spezifizieren.

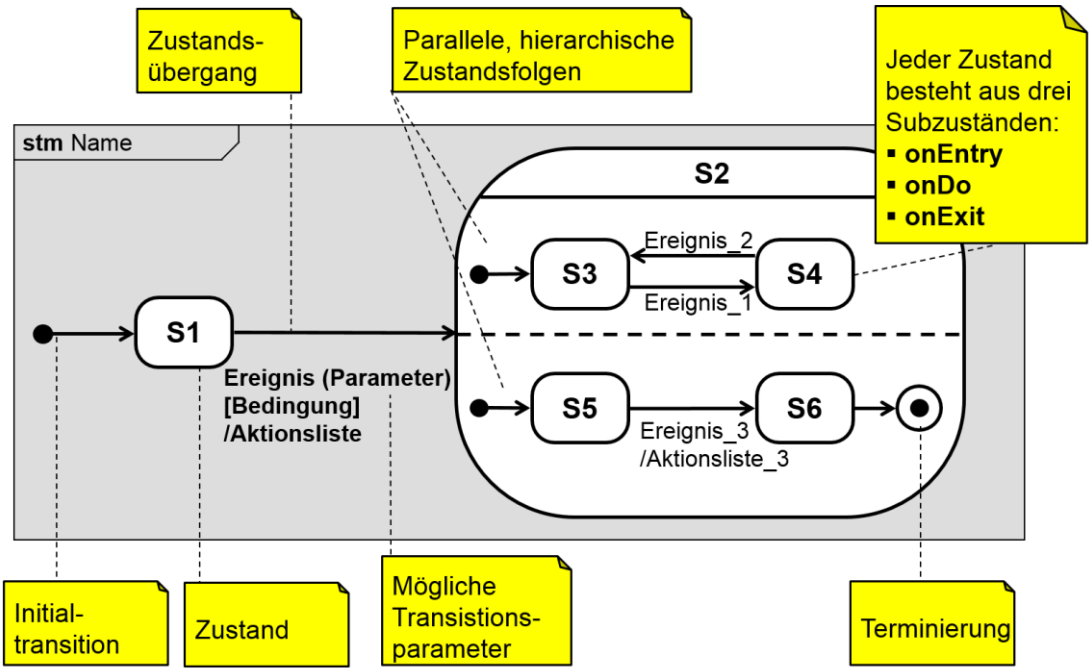


Bild 13: Zustandsfolge-Diagramm Notation

Zustandsfolgeautomaten kommen in vielen Entwicklungsdisziplinen zum Einsatz, nicht nur in der Systemmodellierung, sondern auch in der Software- und Logikentwicklung. Ein Zustandsautomat besteht aus Zuständen, die das beschriebene Element annehmen kann, und den möglichen Zustandsübergängen (Transitionen) dazwischen. Die Initial-Transition legt den Startpunkt des Zustandsfolge-Automaten fest. Jede Transition kann beliebig kombinierbar mit einem parametrierbaren Ereignis, einer Bedingung und einer Aktionsliste verknüpft sein. Sind Ereignis und Bedingung formuliert, müssen zur Ausführung der Transition beide vorhanden sein.

Jeder Zustand selbst ist in drei Subzustände unterteilt, die automatisch durchlaufen werden. Beim Eintritt in einen Zustand wird der onEntry-Subzustand durchlaufen und die damit optional verknüpfte Aktionsliste ausgeführt. Danach befindet sich der Automat im onDo-Subzustand, der optional ebenfalls mit einer Aktionsliste verknüpfbar ist. Vor der Ausführung einer Austrittstransition wird der onExit-Subzustand durchlaufen und die optional damit verknüpfte Aktionsliste ausgeführt. Die Subzustände mit ihren Aktionslisten sind auch für die Selbsttransition gültig. Zustandsfolgeautomaten lassen sich hierarchisch (verschachtelt) aufbauen und können parallele Automaten enthalten.

Das Zustandsfolge-Diagramm bietet viele weitere Notationen, um auch komplexeste generische Abläufe im System darzustellen.

**Zustandsfolge-Diagramm der SysML - Praxisbeispiel Motorsteuerung**

Bild 14 zeigt das Zustandsverhalten des `Main_Control_Boards`. Es ist damit noch keine Aussagen darüber getroffen, ob es später in Hardware oder Software realisiert wird.

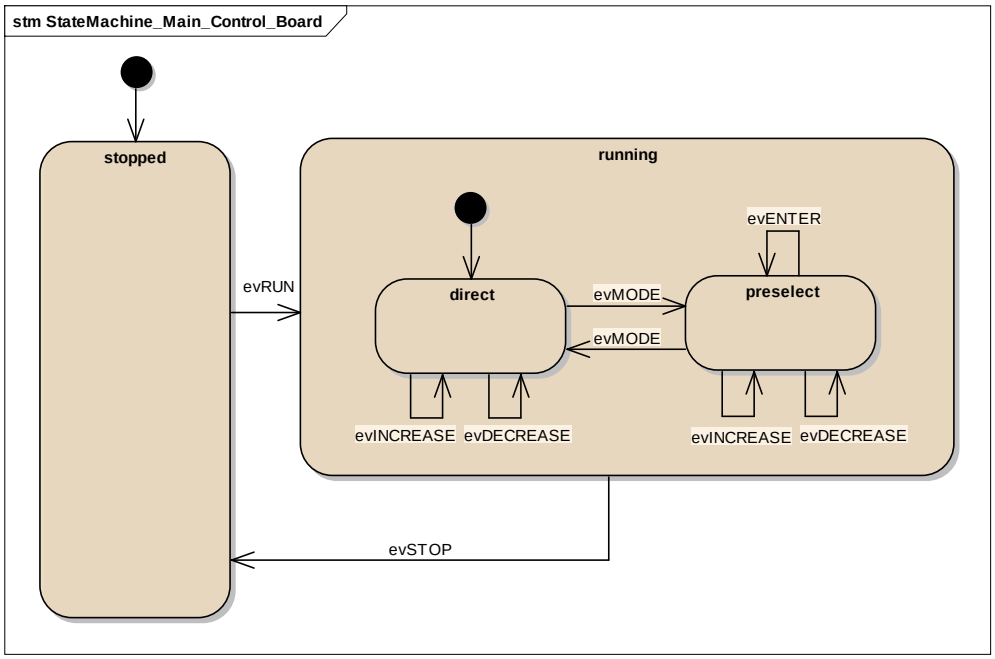


Bild 14: Zustandsfolge-Automat des `Main_Control_Boards`

**SysML-Modellaufbau**
Sicherlich gibt es viele Ansätze, ein SysML-Modell aufzubauen und zu organisieren. Unabhängig für welche Modellstruktur Sie sich letztendlich entscheiden, es wichtig, dass jeder Projektmitarbeiter diese Struktur kennt und auch selbst lebt. Dazu sollte es im Unternehmen ein Templatemodell geben. Alle Modellierungsvereinbarungen sollten in einem unternehmensspezifischen SysML-Style-Guide vereinbart sein.

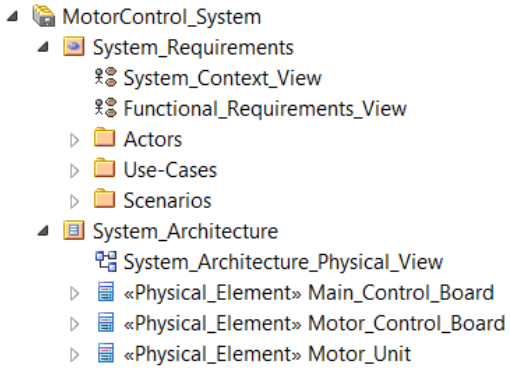


Bild 15: Modellorganisation

Bild 15 zeigt eine beispielhafte Modellorganisation. Das SysML-Modell der Motorsteuerung ist auf der obersten Ebene in `System_Requirements` und `System_Architektur` unterteilt, also in das WAS (Requirements) von dem WIE (Architektur) getrennt.

**Zusammenfassung**

Der SysML-Standard spezifiziert Diagramme und deren Notationen. Er sagt aber nichts direkt über deren Einsatz im System-Entwicklungsprozess aus. Dies ist die Herausforderung eines jeden Unternehmens, dieses für sich zu definieren. Einen ersten Anhaltspunkt dazu zeigt die zusammenfassende Tabelle 1.

| **Aktivität in der Systementwicklung** | **Visualisierter Sachverhalt** | **SysML-Diagramm** |
|---|---|---|
| **System-Anforderungsanalyse** | System-Kontextsicht | Use-Case-Diagramm |
| | Funktionale Anforderungssicht | Use-Case-Diagramm |
| | Szenarien externer Kommunikation (Use-Case- und Akteur-basierend) | Sequenzdiagramm |
| **System-Architekturanalyse** | Funktionale und / oder physikalische Systemarchitektur | Block-Definition-Diagramm |
| | Kommunikationsarchitektur | Block-Definition-Diagramm |
| | Szenarien interner Kommunikation (Block- und Port-basierend) | Sequenzdiagramm |
| **System-Architekturdesign** | Verfeinerte funktionale und / oder physikalische Systemarchitektur | Block-Definition-Diagramm |
| | Generisches Verhalten für funktionale und / oder physikalische Blöcke | Zustandsfolge-Diagramm<br>Aktivitätsdiagramm<br>(hier nicht vorgestellt) |

Tabelle 1: SysML-Einsatz im Kontext des System-Entwicklungsprozesses

**Referenzierte und weiterführende Links:**

[1] MicroConsult-Download für Präsentation und SysML-Beispielmodell
http://download.microconsult.net/ese2013/sysml.zip
[2] OMG (Object Management Group)
www.omg.org
[3] SysML (Systems Modeling Language) bei der OMG
www.omgsysml.org
[4] UML (Unified Modeling Language) bei der OMG
www.uml.org
[5] INCOSE (International Council on Systems Engineering):
www.incose.org
[6] AP233 (Application Protocol 233):
www.ap233.org
[7] STEP (Standard for the Exchange of Product Model Data), ISO 10303:
www.iso.org
*) SysML und UML sind eingetragene & geschützte Marken der Object Management Group (OMG) [2]

**Autor**

Dipl.-Ing. (FH) Thomas Batt ist gebürtiger Freiburger. Nach seiner Ausbildung als Radio- und Fernsehtechniker studierte er Nachrichtentechnik an der Fachhochschule in Offenburg. Thomas Batt war in der Entwicklung in den Bereichen Leistungselektronik, Medizinelektronik und CompactPCI/VME-Bus CPU- und Peripherie-Boards tätig. Heute verantwortet er bei MicroConsult GmbH als Trainer und Coach die Themenbereiche Software Engineering für Embedded-/Realtime-Systeme sowie Entwicklungsprozess-Beratung.