Identifikation von Varianten durch Berechnung der semantischen Differenz von Modellen

Sylvia Melzer[1], Ralf God[1], Thorsten Kiehl[1], Ralf Möller[2], Michael Wessel[2]

[1]Technische Universität Hamburg-Harburg, Institut für Flugzeug-Kabinensysteme, Nesspriel 5, 21129 Hamburg, sylvia.melzer@tuhh.de, ralf.god@tuhh.de,
[2]Technische Universität Hamburg-Harburg, Institut für Softwaresysteme, Schwarzenbergstraße 95, 21079 Hamburg, moeller@tuhh.de

Zusammenfassung: Flugzeuge und deren Systeme, die als typische Vertreter komplexer Systeme gelten, werden heutzutage mit Methoden und Werkzeugen des Systems-Engineering entwickelt. Auch bei der Dokumentation und Optimierung von existierenden Systemen, wie z.B. einer sicheren Luftfracht-Transportkette, hilft das Systems-Engineering, die Komplexität zu beherrschen. Sowohl bei der Entwicklung als auch bei der Optimierung von komplexen Systemen ist ein modellbasiertes Vorgehen methodisch und werkzeugtechnisch zukunftsweisend. In beiden Fällen ist es hilfreich, einen aktuellen Entwicklungs- oder Systemzustand von künftig variierenden oder detaillierten Funktionen in einem Modell abgrenzen zu können. Dazu müssen Änderungen oder Varianten in einem Modell identifiziert werden. In diesem Beitrag wird diese Aufgabe durch Nutzung des sogenannten Abox-Differenz-Operators gelöst, welcher durch die Berechnung der semantischen Differenz ein Basis-Modell aufzeigen kann.

1 Einleitung

Im Lebenszyklus komplexer Systeme führt eine über die Zeit zunehmende Funktionsvielfalt oder eine bei den Stakeholdern unterschiedlich erfolgende Interpretation von Regeln und daher differierende Implementierung von Prozessen zu einer stetig steigenden Systemdiversifizierung. Das Kabinenmanagementsystem moderner Verkehrsflugzeuge, welches beim Betrieb der Flugzeugkabine für die Steuerung aller Kabinensysteme zuständig ist, stellt ein hier betrachtetes komplexes System dar, dessen Funktionsvielfalt aufgrund von Kundenwünschen und einer Verbesserung von Betriebs- und Geschäftsprozessen kontinuierlich zunimmt [GH2012, HG2012]. Für die weltweite Luftfrachtsicherheit, als zweites Beispiel, wurden Anfang dieses Jahres in Europa neue Verordnungen veröffentlicht [ABL2010, ABL2008], gemäß derer aktuell von den EU-Staaten im komplexen Luftfrachtsystem neue Prozesse Luftfrachtsystems umgesetzt werden. Bei beiden genannten Beispielen werden Methoden und Werkzeuge zur Systementwicklung und -optimierung untersucht [GH2012, HG2012, S2013], welche modellbasiert eine Beherrschung der Komplexität erleichtern und verbessern sollen. Sowohl die Weiterentwicklung als auch die Optimierung komplexer Systeme kann im Lebenszyklus durch die Abgrenzung von Basis-Systemmodellen und deren Funktionen, die sich auf die grundlegenden oder ursprünglichen Anforderungen beziehen, verdeutlicht werden. Aus den grundlegenden oder ursprünglichen Anforderungen ergibt sich für die Systementwicklung bzw. für den

Systembetrieb die konstitutive oder maßgebliche Systemspezifikation. Da es im Lebenszyklus eines Systems oft gilt, dieses mit der Zeit an Kundenwünsche oder auf Bedürfnisse von Akteuren anzupassen oder zu optimieren, kommt es zur Herausbildung von Varianten des grundlegenden bzw. ursprünglichen Systems.

In der Praxis kommt der Fall vor, dass für bereits entwickelte oder existierende komplexe Systeme ein Ausgangszustand in Form eines Modells abgebildet vorliegt. Bildet man die über die Zeit durch kundenspezifische Ausdifferenzierung des Systems oder durch akteursspezifische Implementierung und Integration von Prozessvorgaben entstandenen Systemvarianten als Modelle ab, so sind diese Varianten durch Differenzen zu einem grundlegenden oder Basis-Modell charakterisierbar. D.h., Basis-Modell + Differenz = Variante, wobei das Basis-Modell den Systemkern und die Differenz den Unterschied von Systemkomponenten repräsentiert.

Das Problem besteht darin, Differenzen automatisch bestimmen zu können, da zurzeit in den Software-Werkzeugen Variantendifferenzierung nur strukturorientiert unterstützt wird (vgl. SiDiff [F2007] oder Cameo Systems Modeler [N2013]). Hintergrundwissen – außer vielleicht Strukturwissen ausgedrückt durch Grammatiken [BCMNP2003] – wird hierbei nicht verwendet, sodass häufig zu viele, d.h. auch falsch-positive Differenzen gefunden werden. In diesem Beitrag wird der sogenannte Abox-Differenz-Operator eingesetzt, um dieses Problem zu lösen. Dieser Operator identifiziert die Differenzen bei den Systemkomponenten automatisch. Die Systemkomponenten fungieren bei der Berechnung jeweils als eigenes Modell. Deshalb wird hier auch von der Differenz von Modellen gesprochen. Das Besondere an diesem Beitrag ist die Berechnung der Differenz unter Einbeziehung einer Wissensbasis zur Darstellung von Hintergrundwissen über die Anwendung und beschreibt damit die Schaffung eines neuartigen, semantischen Ansatzes zur Berechnung der Differenz von Modellen.

2 Grundlagen für die Repräsentation von Modellen

In diesem Beitrag werden für das Hintergrundwissen in dem hier gewählten semantischen Ansatz zur Berechnung der Differenz von Modellen Beschreibungslogiken [BCMNP2003] verwendet. Die hier als Beispiel genutzten Modelle sind in der Modellierungssprache SysML (Systems Modeling Language) formuliert, und sie stammen aus dem bekannten Software-Engineering-Projekt (siehe [MBSE2012]). Es wird vorgestellt, wie man aus einem SysML-Modell, in beschreibungslogische Ausdrücke übersetzt und Varianten unter Nutzung des beschreibungslogischen Abox-Differenz-Operators identifiziert.

2.1 Beschreibungslogik

Beschreibungslogiken sind eine Familie von Sprachen zur Wissensrepräsentation. In diesem Beitrag wird die Beschreibungslogik *ALC(Attributive (Concept) Language with Complement)* ohne Existenz- und Werterestriktion verwendet. In *ALC* können intuitiv

Beschreibungen für komplexe Konzepte C oder D mittels Konzeptkonstruktoren wie folgt vorgenommen werden:

Syntax	Konstruktor
A	Atomares Konzept
R	Atomare Rolle
$C \sqcap D$	Konjunktion
$C \sqcup D$	Disjunktion
$\neg C$	Negation

wobei A eine Konzept- und R eine Rollenbeschreibung ist. Um Begriffe (atomare Konzepte) zu beschreiben, werden sogenannte Konzeptdefinitionen $A \equiv C$ verwendet. Ein Beispiel für eine Konzeptdefinition ist:
PhasingWavefrontSensor $\equiv$ *hasPart.ZEUS* $\sqcap$ *hasPart.SHAPS*

Ein Phasing Wavefront Sensor besteht aus einem Subsystem ZEUS (Zernike sensing) und einem Subsystem SHAPS (ShackHartmann sensing).

Eine Menge von Konzeptdefinitionen nennen wir Tbox.

Die Semantik der Konzept- und Rollenbeschreibungen wird über *Interpretationen* I definiert. Eine Interpretation besteht aus einer nicht leeren Menge Δ^I, der sogenannten Domäne, und einer Interpretationsfunktion ^{I}. Jeder atomaren Konzeptbeschreibung A ist durch I eine Menge $A^I \subseteq \Delta^I$ zugeordnet. Jeder Rollenbeschreibung wird eine Tupelmenge $R \subseteq \Delta^I \times \Delta^I$ zugeordnet. Die Interpretationsfunktion lässt sich wie folgt auf Konzeptbeschreibungen erweitern:

$$(C \sqcap D)^I = C^I \cap D^I$$
$$(C \sqcup D)^I = C^I \cup D^I$$
$$(\neg C)^I = \Delta^I \setminus C^I$$

Eine Interpretation $(I = (\Delta^I, \ ^I)$ *erfüllt* eine Konzeptbeschreibung C, wenn gilt $C^I \neq \emptyset$. In diesem Fall wird I eine *erfüllende Interpretation* für C genannt. (Der Begriff erfüllende Interpretation wird in der Beschreibungslogik Modell genannt. Der Begriff ist jedoch nicht synonym zum Begriff Modell, welches in diesem Artikel im Kontext des Systems-Engineering gebraucht wird. Daher wird hier zur Unterscheidung und zur Vermeidung von Missverständnissen der Begriff erfüllende Interpretation verwendet).

Eine Konzeptdefinition $A \equiv C$ nennen wir durch eine Interpretation I erfüllt, wenn $A^I \subseteq C^I$ gilt. Eine Tbox wird durch I erfüllt, wenn alle enthaltenen Konzeptdefinitionen erfüllt sind. Wir sagen, ein Konzept C wird durch ein Konzept D subsumiert (bzgl. einer Tbox), wenn für alle die Tbox erfüllenden Interpretationen I gilt $C^I \subseteq D$. Über Subsumption zwischen den atomaren Konzepten, die in einer Tbox erwähnt sind, induzieren die Konzeptdefinitionen einer Tbox also eine Hierarchie von Konzeptnamen (atomaren Konzepten), wenn man nur direkte Subsumptionsbeziehungen betrachtet. Diese Hierarchie heißt auch Subsumptionshierarchie (der atomaren Konzepte einer Tbox).

Das generellste Konzept in einer gegebenen Hierarchie nennt man *top Konzept* ($\top$), welches für Δ^I steht und alle Konzeptnamen subsumiert. Analog nennt man das niedrigste Konzept in einer gegebenen Hierarchie *bottom Konzept* ($\bot$). Dieses Konzept ist mit der leeren Menge gleichzusetzen.

Entscheidungsprobleme bzgl. Erfüllbarkeit und Subsumption ergeben sich sofort aus den hier gegebenen Definitionen der Begriffe.

Neben dem Wissen über Begriffe (Konzepte und Rollen) möchten wir auch Wissen über einzelne Objekte der Domäne Δ^I formalisieren. In einer sogenannten *Abox* repräsentieren wir Objektwissen durch eine Menge von Assertionen der Form $i: C$ oder $(i, j): R$, wobei C eine Konzeptbeschreibung, R eine Rollenbeschreibung und i, j sogenannte Individuen (Namen für Objekte aus Δ^I) sind. Die Funktion *inds(A)* liefere die in einer Abox *A* erwähnten Individuen.

Eine Konzeptassertion $i: C$ wird durch eine Interpretation I erfüllt, wenn $i^I \in C^I$ gilt. Analog gilt dies für die Rollenassertion $(i, j) \in R$; sie wird durch I erfüllt, wenn $(i^I, j^I) \in R^I$ gilt. Eine Interpretation I, die alle Assertionen in einer Abox *A* erfüllt, nennt man erfüllende Interpretation für *A*. Das Konsistenzproblem für eine Abox besteht darin zu prüfen, ob eine solche erfüllende Interpretation existiert.

Die Formalisierung des Hintergrundwissens, auch Ontologie $\Sigma = (T,A)$ genannt, setzt sich aus einer Tbox *T* und Abox *A* zusammen. Das Folgerbarkeitsproblem $\Sigma \models \alpha$ besteht darin, zu prüfen, ob für alle erfüllenden Interpretationen I von Σ gilt: I erfüllt α, wobei α eine Abox oder eine einzelne Assertion sein kann.

Die semantische Differenz $\Delta_{A,B}$ zweier Aboxen $A = \{a_1, \ldots, a_n\}$ und $B = \{b_1, \ldots, b_m\}$ ist definiert als eine Menge von Assertionen von *A*, mit $n, m \in \mathbb{N}$, sodass gilt:

1. Es existiert eine (nicht notwendigerweise totale) Abbildung φ:*inds(B)*$\rightarrow$*inds(A)*, sodass $(T, \varphi(B) \cup \Delta_{A,B}) \models A$ gilt, wobei φ wie folgt erweitert wird
   $\varphi(A) \stackrel{\text{def}}{=} \{\varphi(a_1), \ldots, \varphi(a_n)\}$,
   $\varphi(i:C) \stackrel{\text{def}}{=} \phi(i):C$,
   $\varphi((i, j):R) \stackrel{\text{def}}{=} (\phi(i), \phi(j)):R$.
2. $B \cup \Delta_{A,B}$ ist konsistent.
3. $\Delta_{A,B}$ ist minimal bezüglich $\subseteq$.

Die semantische Differenz (kurz: Abox-Differenz) ist eine erweiterte Form des Abduktionsproblem (Berechnung von $\Delta_{A,B}$) [EKM2009, EKMMW2007]. Der Abox-Differenz-Operator ist kein kommutativer Operator. Beispiele für seine Anwendung zeigen wir in Kapitel 3 auf, nachdem wir ein Modell in der Sprache SysML (Systems Modeling Language) aufgestellt und als beschreibungslogische Aboxen repräsentiert haben. Die Ergebnisse der semantischen Differenz liefern uns Hinweise, ob beispielweise Systemkomponenten eines Modells Varianten sind.

Eine Variante $\Lambda_{A,B}$ zweier Aboxen *A* und *B* ist definiert als eine Assertionsmenge, die sich aus der Summe der semantischen Differenz $\Delta_{A,B}$ und des Basis-Modells *BM* ergibt.

Das Basis-Modell *BM* zweier Aboxen *A* und *B* ist definiert als eine Menge von Assertionen $\varphi(B^*)$, sodass gilt:
$(T, \varphi(B^*) \cup \Delta_{\varphi(A),\varphi(B)}) \models \Delta_{A,B}$ und $(T, \varphi(A^*) \cup \Delta_{\varphi(B),\varphi(A)}) \models \Delta_{B,A}$,
wobei die Variablen von $\varphi(B^*)$ umbenannt werden.

2.2 Systems Modeling Language (SysML)

SysML wurde von der Object Management Group (OMG) [OMG2014] definiert, um die modellbasierte Entwicklung komplexer Systeme in ihren drei wichtigsten Bereichen dem Systementwurf, die Implementierung und die Systemintegration zu unterstützen. SysML enthält eine Teilmenge der standardisierten Sprache UML 2, weist aber gegenüber dieser auch einige Erweiterungen auf. Für eine vertiefende Erläuterung der Sprache SysML wird auf die Quellen [W2009] und [A2012] verwiesen. Jedes SysML-Modell berücksichtigt zwei wesentliche Systemaspekte: die Struktur und das Verhalten eines Systems. Die Struktur zeigt Entitäten (Individuen) in einem System und repräsentiert die Beziehungen zwischen ihnen. Das Verhalten zeigt, wie sich ein System über die Zeit verhält. Es werden die Anforderungen und Bedingungen an ein System oder auch Interaktionen zwischen Systemen berücksichtigt.
In diesem Beitrag liegt der Fokus auf der Struktur eines Modells, um Varianten zu identifizieren. In SysML repräsentieren die so genannten Blöcke die Subsysteme oder Systemkomponenten.

3 Identifikation von Varianten in einem Beispielmodell

Im Projekt Active Phasing Experiment (APE), einem Projekt der ESO (European Southern Observatory), wurde ein Teleskop-System-Modell, genannt APE-Modell, erstellt [MBSE2012]. Das Modell wurde entwickelt, um zu zeigen, dass man SysML gewinnbringend beim Systems-Engineering-Prozess für ein reales komplexes System einsetzen kann. Seitdem dient das APE-Modell bei der Systemmodellierung häufig als literaturbekannte Referenz. Auch wir nutzen daher in diesem Beitrag das APE-Modell als Beispiel für die Berechnung der semantischen Differenz von Modellen.

3.1 Abbildung von SysML auf ALC

Die Abbildung von Elementen der Sprache SysML auf die Elemente der Beschreibungslogik *ALC* wird in diesem Abschnitt anhand eines repräsentativen Ausschnitts des APE-Modells vorgestellt.
In [MBSE2012] sind die Systemkomponenten des APE-Systems modelliert. Das APE-System besteht u.a. aus den Komponenten ZEUS, SHAPS und DIPSI (Diffraction sensing), die spezielle Sensoren des Phasing Wavefront Sensor sind. Für diesen Beitrag wählen wir aus Platzgründen einen repräsentativen Ausschnitt des APE-Modells, welcher in Bild 1 dargestellt ist.

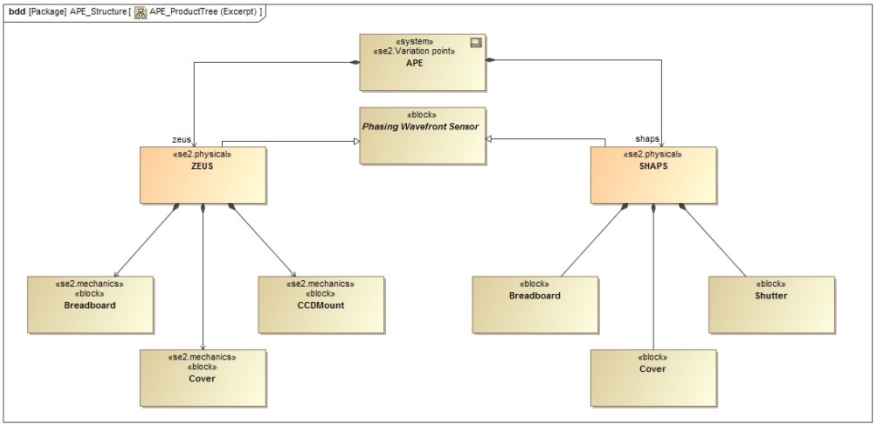

Bild 1. Ausschnitt des APE-Modells aus [MBSE2012] in der Sprache SysML.

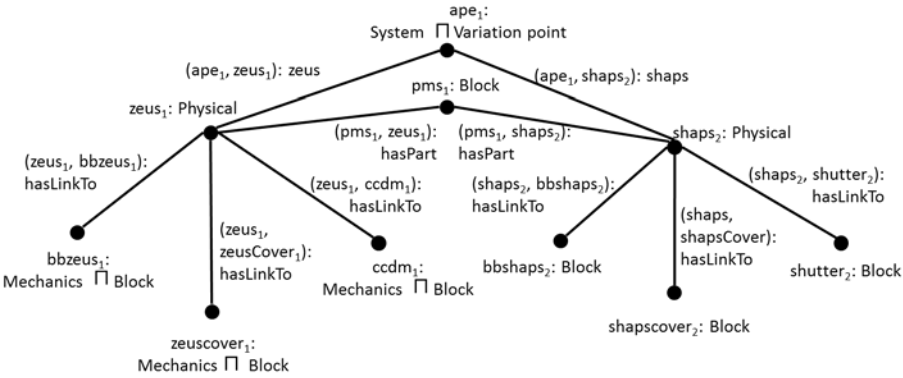

Bild 2. Abbildung des APE-Modells aus Bild 1 auf die Sprache *ALC*.

Das System *APE*, welches als Block mit dem Stereotypen <<system>> und <<Variation point>> spezifiziert wurde, hat jeweils eine nicht beschriftete Kompositionsbeziehung zum Subsystem *ZEUS* und zum Subsystem *SHAPS*. Zwischen dem System *APE* und den Subsystemen *ZEUS* und *SHAPS* gibt es jeweils eine Rolle mit den Namen *zeus* bzw. *shaps*.
Die Subsysteme *ZEUS* und *SHAPS* sind Blöcke mit dem Stereotypen <<physical>> und sind jeweils eine Spezialisierung des abstrakten Systems *Phasing Wavefront Sensor*. *ZEUS* hat jeweils eine nicht beschriftete Kompositionsbeziehung zu den Systemkomponenten *Breadboard*, *Cover* und *CCDMount*. Die drei Systemkomponenten sind Blöcke mit den Stereotypen <<mechanics>> und <<block>>. *SHAPS* hat jeweils eine nicht beschriftete Kompositionsbeziehung zu den Systemkomponenten *Breadboard*, *Cover* und *Shutter*. Diese drei Systemkomponenten sind Blöcke mit dem Stereotypen <<block>>.

Bild 2 repräsentiert die Abbildung des Modells aus Bild 1 auf die Sprache ALC. Die Stereotypen <<block>>, <<physical>> usw. werden auf atomare Konzepte (Block, Physical usw.) abgebildet. Die Namen der Subsysteme werden als technische Konzepte verwendet und sind daher nicht im Bild dargestellt. (Für die Berechnung semantischer Differenzen sind solche technische Konzepte notwendig; die Notwendigkeit wird in den Beispielen des nächsten Abschnittes deutlich.) Der Name eines Blocks (*ZEUS*, *SHAPS* usw.) wird auf ein Individuum ($zeus_1$, $shaps_2$), nicht beschriftete Generalisierungsbeziehungen (*Phasing Wavefront Sensor* ist Generalisierung von *ZEUS*) werden auf *hasPart*-Relationen ($(pws_1,zeus_1)$:*hasPart*) und nicht beschrifte Kompositionsbeziehungen werden auf *hasLinkTo*-Relationen abgebildet. Die Rollen *zeus* und *shaps* werden auch auf *hasLinkTo*-Relationen abgebildet, wobei der Rollenname für die Relation verwendet wird.

Wie nun zu erkennen ist, handelt es sich in Bild 2 um eine Abox. Die Tbox ist zunächst leer. Der Grund für die Abbildung des SysML Modells auf eine Abox liegt darin, dass auf konzeptioneller Ebene (Blockdiagramm (engl. Block definition diagram, kurz: bdd)) Objekte definiert sind. Ein solches Modell bezeichnen wir als ein Prototypen-Modell. Das Problem, was durch eine Prototypen-Modellierung entsteht, ist das wachsende Variantenspektrum. Dieses Problem kann mittels der Berechnung semantischer Differenzen gelöst werden, weil dadurch Varianten identifiziert werden können.

3.2 Berechnung der semantischen Differenz und die Identifikation von Varianten

Wenn sich ein Ingenieur sich in das APE-Modell einarbeitet, interessiert er sich eventuell für die Unterschiede vom ZEUS-Sensor zu dem SHAPS-Sensor. Die Aboxen für die Sensoren ZEUS (Abox *A*) und SHAPS (Abox *B*) sind, wie oben beschrieben, wie folgt definiert:

A= *{(ape_1: (APE)), (ape_1: (System ⊓ Variation Point)), ($zeus_1$:Physical), ($zeus_1$: PWS), ((ape_1,$zeus_1$):zeus), (pws_1:PWS), ((pws_1,$zeus_1$):hasPart), ($breadboard_1$: Breadboard), ($breadboard_1$: (Mechanics ⊓ Block)), (($zeus_1$, $breadboard_1$): hasLinkTo), ($cover_1$:Cover), ($cover_1$: (Mechanics ⊓ Block)), (($zeus_1$, $cover_1$):hasLinkTo), ($ccdmount_1$: CcdMount), ($ccdmount_1$: (Mechanics ⊓ Block)), (($zeus_1$, $ccdmount_1$):hasLinkTo)}*

B= *{(ape_2: (APE)), (ape_2: (System ⊓ Variation Point)), ($shaps_2$:Physical), ($shaps_2$: PWS), ((ape_2,$shaps_2$):shaps), (pws_1:PWS), ((pws_1,$shaps_2$):hasPart), ($breadboard_2$: Breadboard), ($breadboard_2$: Block), (($shaps_2$, $breadboard_2$): hasLinkTo), ($cover_2$:Cover), ($cover_2$: Block), (($shaps_2$, $cover_2$):hasLinkTo), ($shutter_2$: Shutter), ($shutter_2$: Block), (($shaps_2$, $shutter_2$):hasLinkTo)}*

Die kursiv gedruckten Konzepte sind technische Konzepte, die der Ontologie nicht hinzugefügt, aber zur Berechnung semantischer Differenzen benötigt werden. Für die Berechnungen semantischer Differenzen haben wir den DL-Reasoner RacerPro [HMW2007] verwendet, in dem der Abox-Differenz-Operator integriert ist. Die semantische Differenz lautet für die Sensoren ZEUS und SHAPS:

$\Delta_{A,B}$ = *{((ape_1,$zeus_1$):zeus), ($breadboard_1$: (Breadboard ⊓ Mechanics ⊓ Block)), ($cover_1$: Cover ⊓ Mechanics ⊓ Block)), ($ccdmount_1$: (CcdMount ⊓ Mechanics ⊓ Block))}*

$\Delta_{B,A}$ = *{(ape_2,$shaps_2$):shaps, ($shutter_2$: Shutter ⊓ Block)}*

Das Beispiel zeigt, dass es eine semantische Differenz zwischen ZEUS und SHAPS gibt. Die semantischen Differenzen $\Delta_{A,B}$ und $\Delta_{B,A}$ unterscheiden sich, weil die Blöcke in SHAPS nur den Stereotypen <<block>> besitzen. Wenn der Stereotyp <<Mechanics>> bei der Modellierung der SHAPS-Komponentenblöcke vergessen wurde, kann die Ergebnismenge der semantischen Differenz hilfreich sein, Hinweise auf eine vollständige Modellierung zu liefern.
Ergänzt man die Abox *B* um den Stereotypen <<Mechanics>>, ändert sich die Abox *B* wie folgt:

$B^* = \{(ape_2: (APE)), (ape_2: (System \sqcap Variation\ Point)), (shaps_2:Physical), (shaps_2: PWS), ((ape_2,shaps_2):shaps), (pws_1:PWS), ((pws_1,shaps_2):hasPart), (breadboard_2: Breadboard), (breadboard_2: Mechanics \sqcap Block), ((shaps_2, breadboard_2): hasLinkTo), (cover_2:Cover), (cover_2: Mechanics \sqcap Block), ((shaps_2, cover_2):hasLinkTo), (shutter_2: Shutter), (shutter_2: Mechanics \sqcap Block), ((shaps_2, shutter_2):hasLinkTo)\}$

Die semantischen Differenzen lauten:

$\Delta_{A, B^*} = \{((ape_1,zeus_1):zeus), (ccdmount_1: (CcdMount \sqcap Mechanics \sqcap Block))\}$

$\Delta_{B^*, A} = \{(ape_2,shaps_2):shaps, (shutter_2: Shutter \sqcap Mechanics \sqcap Block)\}$

Die Assertionen, die sich aufeinander abbilden lassen, werden einem Basis-Modell *BM* hinzugefügt und gleichzeitig aus der Abox entfernt. Damit erreicht man, dass sich nur der variierende Anteil in der Abox befindet. Die Vorteile für die Definition eines Basis-Modells präsentieren wir an einem Beispiel, nachdem wir die Varianten der Sensoren bestimmt haben. Die Varianten sind $\Lambda_{A,B} = BM_{AB} \cup \Delta_{A, B}$ bzw. $\Lambda_{B,A} = BM_{AB} \cup \Delta_{B, A}$ mit $BM_{AB} = \varphi(B^*)$ und

$\varphi(B^*) = \{(ape: (APE)), (ape: (System \sqcap Variation\ Point)), (sensor:Physical), (sensor: PWS), (pws:PWS), ((pws,sensor):hasPart), ((sensor, breadboard): hasLinkTo), ((sensor, cover):hasLinkTo)\}$

Wir betrachten nun den Sensor DIPSI aus [MBSE2012]. DIPSI hat die gleichen Komponenten und Komponentenbeziehungen wie SHAPS. Die entsprechende Abox *C* ist daher wie folgt definiert:

$C = \{(ape_3: (APE)), (ape_3: (System \sqcap Variation\ Point)), (dipsi_3:Physical), (dipsi_3: PWS), ((ape_3, dipsi_3): dipsi), (pws_1:PWS), ((pws_1, dipsi_3):hasPart), (breadboard_3: Breadboard), (breadboard_3: Block), ((dipsi_3, breadboard_3): hasLinkTo), (cover_3:Cover), (cover_3: Block), ((dipsi_3, cover_3):hasLinkTo), (shutter_3: Shutter), (shutter_3: Block), ((dipsi_3, shutter_3):hasLinkTo)\}$

Die semantischen Differenzen lauten: $\Delta_{B,C}=\{(ape_2, shaps_2): shaps\}$ und $\Delta_{C,B}=\{(ape_3, dipsi_3): dipsi\}$.
Die Ergebnisse zeigen, dass es keine Differenz von Systemkomponenten zwischen SHAPS und DIPSI gibt. Daraus folgt, dass diese Sensoren keine Varianten des Phasing Wavefront Sensor sind. Auch hier liefert der Operator Hinweise auf eine (un-) vollständige Modellierung.
Die Varianten sind:

$\Lambda_{B,C} = BM_{BC} \cup \Delta_{B,C}$ bzw. $\Lambda_{C,B} = BM_{BC} \cup \Delta_{C, B}$ mit

$BM_{BC} = \{(ape: (APE)), (ape: (System \sqcap Variation\ Point)), (sensor:Physical), (sensor: PWS), (pws:PWS), ((pws, sensor):hasPart), (breadboard: Breadboard), (breadboard: Block), ((sensor, breadboard): hasLinkTo), (cover:Cover), (cover: Block), ((sensor, cover):hasLinkTo), (shutter: Shutter), (shutter: Block), ((sensor, shutter):hasLinkTo)\}$.

Die Definition eines Basis-Modells kann für einen Ingenieur hilfreich sein, wenn ein bei der Erweiterung eines vorhandenen Modells, Hinweise für eine korrekte Modellierung eines Systems geliefert werden.
Zum Beispiel nehmen wir an, dass ein Ingenieur den Physical Wavefront Sensor um einen weiteren Sensor *D* erweitern möchte.
Er fügt in seinem Modell für die Darstellung des Sensors *D* einen Block mit dem Stereotypen <<physical>> und dem Namen *D* hinzu. Die entsprechende Abox lautet:
$D = \{((ape_4: (APE)), (ape_4: (System \sqcap Variation\ Point)), (sensor_4:Physical), (sensor_4: PWS), ((ape_4, sensor_4): sensor), (pws_1:PWS), ((pws_1, sensor_4):hasPart)\}$

Der Vergleich des Sensors *D* zum Basis-Modell BM_{BC} liefert dem Ingenieur den Hinweis, dass der Sensor *D* um die Subkomponenten Breadboard, Cover und Shutter erweitern werden muss.

$BM_{BC} = \{((ape,sensor):sensor), (breadboard: Breadboard), (breadboard: Block), ((sensor, breadboard): hasLinkTo), (cover:Cover), (cover: Block), ((sensor, cover):hasLinkTo), (shutter: Shutter), (shutter: Block), ((sensor, shutter):hasLinkTo)\}$

Bisher haben wir noch nicht gezeigt, dass die Tbox für die Berechnung semantischer Differenzen einen signifikanten Einfluss hat. Im nächsten Beispiel zeigen wir hingegen, den positiven Effekt einer Tbox auf die Berechnung von Differenzen.
Angenommen das APE-Modell soll nicht nur erweitert, sondern Komponenten durch qualitativ höherwertige Komponenten ausgetauscht werden. Beim SHAPS-Sensor (Abox *B**) wird *Cover* durch einen speziellen *ShapsCover* ersetzt. Die Information, dass es sich beim Shaps-Cover um einen Cover handelt, wird in der Tbox hinterlegt. Die Tbox lautet hierfür: $T=\{shapsCover \sqsubseteq Cover\}$. Die geänderte SHAPS-Abox *B*** lautet:

$B^{**} = \{(ape_2: (APE)), (ape_2: (System \sqcap Variation\ Point)), (shaps_2:Physical), (shaps_2: PWS), ((ape_2,shaps_2):shaps), (pws_1:PWS), ((pws_1,shaps_2):hasPart), (breadboard_2: Breadboard), (breadboard_2: Block), ((shaps_2, breadboard_2): hasLinkTo), (cover_2:ShapsCover), (cover_2: Block), ((shaps_2, cover_2):hasLinkTo), (shutter_2: Shutter), (shutter_2: Block), ((shaps_2, shutter_2):hasLinkTo)\}$

Nun sollen die Sensoren ZEUS und SHAPS im erweiterten APE-Modell verglichen werden. Dazu bilden wir wieder die semantische Differenz:

$\Delta_{A, B^{**}} = \{((ape_1,zeus_1):zeus),(ccdmount_1: (CcdMount \sqcap Mechanics \sqcap Block))\}$

$\Delta_{B^{**}, A} = \{(ape_2,shaps_2):shaps, (shutter_2: Shutter \sqcap Mechanics \sqcap Block), (cover2: ShapsCover \sqcap Mechanics \sqcap Block\}$

Das Ergebnis zeigt, dass sich SHAPS im Vergleich zu ZEUS um die Subkomponente CCDMount unterscheidet. Dass im Ergebnis kein Cover enthalten ist, kommt aufgrund der Tbox zustande. Vergleicht man ZEUS mit SHAPS erhält man als Differenz die Subkomponenten Shutter und ShapsCover.

In diesem Beitrag haben wir gezeigt, dass durch die Berechnung semantischer Differenzen das Problem des Variantenspektrums für eine Prototypmodellierung gelöst sowie eine Optimierung von Systemen vorgenommen werden kann und die Dokumentation von Modellen erleichtert wird. Anhand von Beispielen haben wir veranschaulicht, wie auf Basis semantischer Differenzen Varianten identifiziert und das Basis-Modell definiert werden können.

Literaturverzeichnis

[A2012] Alt, O.: Modellbasierte System-Entwicklung mit SysML. Hanser Verlag München, 2012.

[BCMNP2003] Baader, F.; Calvanese, D.; Nardi, D.; Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Application. Cambridge University Press, 2003.

[ABL2008] Europäisches Parlament; Rat: Verordnung (EG) Nr. 300/2008 des Europäischen Parlaments und des Rates vom 11. März 2008 über gemeinsame Vorschriften für die Sicherheit in der Zivilluftfahrt und zur Aufhebung der Verordnung (EG) Nr. 2320/2002. Amtsblatt der Europäischen Union L 97/72, 9.4.2008.

[ABL2010] Kommission: Verordnung (EU) Nr. 185/2010 der Kommission vom 4. März 2010 zur Festlegung von detaillierten Maßnahmen für die Durchführung der gemeinsamen Grundstandards in der Luftsicherheit. Amtsblatt der Europäischen Union L 55/1, 5.3.2010.

[EKMMW2007] Espinosa, S.; Kaya, A.; Melzer, S.; Möller, R.; Wessel, M.: Multimedia Interpretation as Abduction. In Proc. of the 2007 International Workshop on Description Logics DL-2007, 2007.

[EKM2009] Espinosa, S.; Kaya, A.; Möller, R.: Formalizing multimedia interpretation based on abduction over description logic aboxes. In Proc. of the 2009 International Workshop on Description Logics DL-2009, Oxford, United Kingdom, 2009.

[F2007] Falk, J.: Entwicklung von Differenzmetriken und deren Visualisierung. Diplomarbeit, Universität Siegen, 2007.

[GH2012] God, R.; Hintze, H.: Komplexität beherrschen: Methodologie für die modellbasierte Entwicklung von Kabinensystemen. Ingenieurspiegel Vol. 1, 2012, S.34-36.

[HG2012] Hintze, H.; God, R.: Ansatz für einen Systems Security Engineering Prozess zur Entwicklung eines Kabinenmanagementsystems der nächsten Generation. Deutscher Luft- und Raumfahrtkongress, Berlin, 10.-12. September 2012.

[HMW2007] Haarslev, V.; Möller, R.; Wessel, M.: RacerPro User's Guide and Reference Manual. Version 1.9.1, 2007.

[MBSE2012] http://mbse.gfse.de/extdocs/ape.html, 2012.

[N2013] No Magic: Cameo Systems Modeler. http://www.nomagic.com/products/cameo-systems-modeler.html, 2014.

[OMG2014] OMG: OMG Systems Modeling Language. http://www.omgsysml.org, 2014.

[S2013] Sichere Luftfracht-Transportkette: Konzepte, Strategien und Technologien für sichere und effiziente Luftfracht-Transportketten. http://www.silufra.de

[W2012] Weilkiens, T.: Systems Engineering mit SysML/UML: Modellierung, Analyse, Design. Dpunkt. Verlag, 2009.