SysML-Modelle maschinell verstehen und verknüpfen

Philipp Mochine[1], Atakan Sünnetcioglu[2], Oskar von Dungern[3], Rainer Stark[1]

[1]Technische Universität Berlin, Pascalstraße 8-9, 10587 Berlin, rainer.stark@tu-berlin.de
[2]Fraunhofer IPK, Pascalstr. 8-9, 10587 Berlin, atakan.suennetcioglu@ipk.fraunhofer.de
[3]adesso AG, Cross-Industries, Rotherstrasse 19, 10245 Berlin, oskar.dungern@adesso.de

Zusammenfassung: Bisher setzte die Systemmodellierung ausgewachsene Autorensysteme voraus. Sie implementieren etwa das SysML Metamodell mit herstellerspezifischen Interpretationen, so dass der Austausch von Modellen in der Praxis nur zwischen Werkzeugen gleichen Typs möglich ist. Die Integration mit Informationen aus anderen Quellen wird nur punktuell unterstützt.

Am Beispiel der TdSE Kaffeemaschine wird gezeigt, wie SysML Modelldiagramme aus Office-Werkzeugen maschinell interpretiert und die gesamte Modellinformation mit Anforderungslisten zu einem integrierten Systemmodell zusammen geführt werden kann.

Darüber hinaus wird diskutiert, welchen Nutzen es bringt Modellelemente und ihre Beziehungen automatisch aus Diagrammen zu identifizieren und für die maschinelle Verarbeitung aufzubereiten. Insbesondere das weitgehend automatische Ermitteln der sog. „Tracelinks" unterstützt die Navigation im Modell und das Nachvollziehen von Abhängigkeiten im Systemmodell. Weiterhin wird der Austausch von Modellinformation zwischen Werkzeugen verschiedenen Typs ermöglicht, und zwar auch auf der Bedeutungsebene.

1 Einleitung

Systemspezifikationen spielen eine entscheidende Rolle als Ausgangspunkt der Produktentwicklung. Sowohl bei agilen wie bei klassischen Vorgehensweisen sorgen sie für eine effektive Kommunikation zwischen allen Beteiligten und dienen als Referenz für Entscheidungen oder Entwicklungs- und Testaufgaben. Dafür müssen sie verständlich sein und eine hohe Qualität aufweisen.

Aus *methodischer Perspektive* sind anspruchsvolle Systeme mit Texten allein nicht hinreichend genau zu beschreiben. In der Praxis werden daher strukturierte Anforderungslisten (z.B. auch User-Stories) und Modelldiagramme verschiedener Art verwendet.

Aus *inhaltlicher Perspektive* fließen Informationen oft aus verschiedenen Organisationen ein und beleuchten etwa Funktionen aus Nutzersicht oder Anforderungen aus Produktstrategie, Gesetzen, Verbraucherschutz und nicht zuletzt Betrieb. Um Informationen und Modelle verschiedener Quellen zusammenführen zu können, ist ein gemeinsamer Kontext zu schaffen. Dabei werden folgende Ziele verfolgt:

- In Informationen verschiedener Quellen übergreifend suchen und navigieren,
- Identische Elemente in verschiedenen Modellen finden und konsolidieren,

- Abhängigkeiten und logische Beziehungen zwischen Modellelementen hinterlegen, also verschiedene Modelle semantisch verknüpfen,
- Lücken, Inkonsistenzen oder Verletzung von Entwurfsregeln erkennen,
- Die interdisziplinäre Systemkonzeption unterstützen.

Ziel ist ein integriertes Systemmodell, das die Entwicklung und ggf. die Produktionsplanung eines mechatronischen Produkts begleitet. Für unterschiedliche Verwendungen lassen sich konsistente Sichten erzeugen: Von der Dokumentation in verschiedenen Systemen/Formaten über die Simulation bis zur Übernahme in ALM/PLM Systeme.

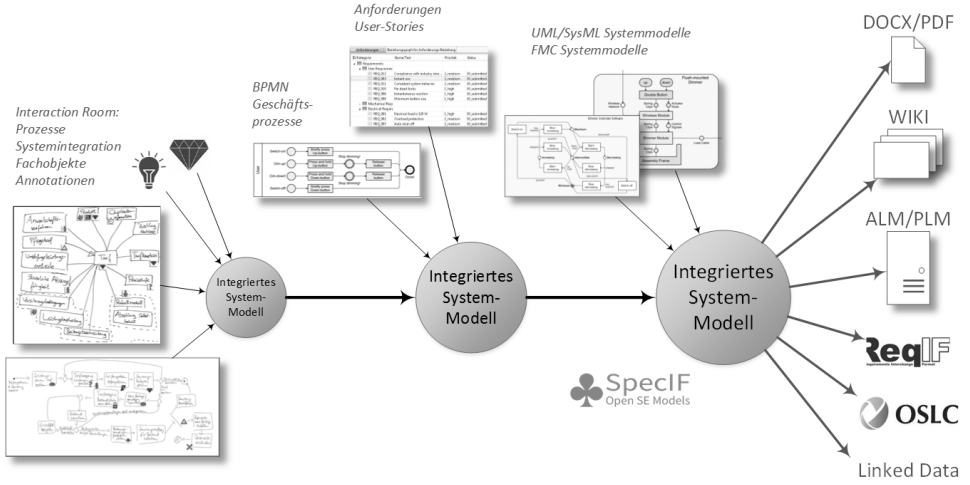

Bild 1: Modellkern mit Sichten und Ausleitung in verschiedenen Formaten

Bild 1 zeigt, wie aus Diagrammen und Listen unterschiedlicher Arten und Quellen schrittweise ein gemeinsamer Modellkern gebildet wird. Für jedes identifizierte graphische Element wird automatisch ein Modellelement als Datenobjekt angelegt, wobei identische Modellelemente konsolidiert werden: Darstellungen auf verschiedenen Diagrammen weisen nun jeweils auf das gleiche Modellelement. Namensänderungen werden sofort und überall sichtbar; Element-Beschreibungen gibt es nur einmal.

Weiterhin werden Beziehungen zwischen den Modellelementen hergestellt. Beispielsweise wird automatisch ermittelt, wenn ein Plan eine Komponente ‚B' als Teil von ‚A' zeigt oder wenn ein Prozess-Schritt ‚X' ein Dokument ‚D' erzeugt. Die Zuordnung von Anforderungen zu Systemkomponenten erfolgt manuell; diese Zuordnungen helfen die Vollständigkeit und Realisierbarkeit der Anforderungen zu beurteilen.

Das Konsolidieren gleicher Modellelemente von verschiedenen Plänen und das Ableiten expliziter logischer Beziehungen werden hier als semantische Vernetzung bezeichnet. Damit sind viele praktische Vorteile verbunden:

- Das semantische Netz kann für die Navigation und die Prüfung heran gezogen werden. Zum Beispiel sind die Anforderungen, die von einer bestimmten Systemkomponente zu erfüllen sind, sofort ersichtlich.
- Ein Modellkern hilft redundante Informationen zu vermeiden. Zugleich wird die konsequente Ausprägung von Gleichteilen erleichtert.
- Inkonsistenzen lassen sich aufdecken, teilweise sogar automatisch.

Der im weiteren Verlauf beschriebene Ansatz ist bewährt: Die Integration von Modellsichten der Fundamental Modelling Concepts (FMC) hat ihren Nutzen in etlichen Industrieprojekten gezeigt [Dun14]. Hier wird vorgestellt, wie Modelle aus unterschiedlichen Werkzeugen und Notationen integriert werden können: UML/SysML Diagramme aus einem Zeichenwerkzeug (MS-Visio) und Anforderungen aus einer Tabelle (MS-Excel) werden im offenen Datenformat SpecIF [Spec17] zusammen geführt.

Gelegentlich wird argumentiert, dass verfügbare Modellierungs-Werkzeuge die genannten Ziele bereits erfüllen. Die Praxis zeigt jedoch, dass verschiedene Werkzeuge unterschiedliche Interpretationen des UML-Metamodells verkörpern und dass es daher auch mittels standardisiertem XMI nicht gelingt Modelle auszutauschen oder gar zusammen zu führen. Das hier vorgestellte Vorgehen verwendet hingegen verbreitete Zeichenwerkzeuge und Notationen sowie ein offenes Metamodell. Ein weiterer Unterschied ergibt sich durch die Abstraktion auf wenige fundamentale Modellelement-Typen, die eine automatische Integration überhaupt erst ermöglicht [Dun16].

2 Dokumente und Modelle in ein Integrationsmodell überführen

Der bereits vorangeschrittene Wandel von einer dokumentenbasierten zu einer modellbasierten Arbeitsweise wird mit der Specification Integration Facility (SpecIF) unterstützt. SpecIF kann als Sprache (mit Syntax und Semantik) für Systembeschreibungen aufgefasst werden; textuelle und graphische Inhalte werden in einen gemeinsamen formalen Kontext gestellt. SpecIF definiert also eine Ebene der Verständigung auf Basis bekannter technischer Formate und Protokolle [Spec17]. So kann ein Systemmodell auch in ein ALM/PLM-System überführt werden (Bild 1) und als Ausgangspunkt für die weitere Entwicklung in den Ingenieursdisziplinen Mechanik, Elektronik und Software dienen.

Um ein integriertes Systemmodell zu erhalten, müssen Teilmodelle verstanden und miteinander verknüpft werden. Hierzu werden fünf Prinzipien der semantischen Integration heran gezogen, von denen vier in einem früheren Beitrag vorgestellt wurden [Dun16]:

1. Trennung von Sichten und Modell,
2. Abstraktion der Modellelemente,
3. Konsolidierung,
4. Vernetzung,
5. Verwendung eines vereinbarten Vokabulars.

Trennung von Sichten und Modell: Hierbei wird zwischen Sicht und Modell unterschieden. Eine Sicht, etwa ein Diagramm oder eine nach speziellen Kriterien gefilterte Liste, ist eine bestimmte partielle Visualisierung des gemeinsamen logischen Modellkerns.

Abstraktion der Modellelemente: Allein SysML bietet für ein Modell 163 graphische Knotentypen, 22 Datentypen, 211 Metaklassen und 25 Stereotypen an [OMG15]. Damit ist es praktisch nicht möglich, alle Elemente verschiedener Modelle sinnvoll zuzuordnen. Die Fundamental Modelling Concepts (FMC) nach Prof. Dr. Siegfried Wendt bieten eine Abstraktion von Modellelement-Typen an [Knö05], mit denen sich Modelle aller gängigen Methoden aus drei fundamentalen Modellelement-Typen aufbauen lassen [Spec17]:

- ■ Akteur: Verkörpert ein aktives Element, wie eine Aktivität, Funktion, Prozess-Schritt, Systemkomponente oder eine Rolle.
- ● Zustand: Repräsentiert ein passives Element, wie eine Form, Wert, Informationsspeicher, Bedingung, oder eine physische Beschaffenheit.
- ♦ Ereignis: Ist eine zeitliche Referenz. Das kann eine Änderung einer Bedingung bzw. eines Zustandes oder generell ein Signal zur Synchronisation sein.

Daneben werden noch drei weitere Element-Typen vorgeschlagen [Dun16], die in Systemspezifikationen eine zentrale Rolle spielen:

- ▣ Plan: Verkörpert jeweils ein Diagramm des Modells.
- ↯ Anforderung: Beschreibt das physische oder funktionale Bedürfnis, das der bestimmte Entwurf, das Produkt oder der Prozess realisieren muss.
- ✶ Merkmal: Spiegelt die kennzeichnende Charakteristik eines Systems wieder, zuweilen als Alleinstellungsmerkmal.

Konsolidierung: In der Konsolidierung werden identische Modellelemente in Diagrammen gleichen oder unterschiedlichen Typs identifiziert und einem Element im Modellkern zugewiesen. Kommt also ein Modellelement mehrfach vor, wird es zu einer einzigen Objektinstanz im Modellkern zusammengeführt. Der Modellkern vermeidet somit redundante Informationen.

Vernetzung: Hierbei werden Modelelemente miteinander in Beziehung gesetzt. Die logischen Beziehungen werden automatisch oder manuell gesetzt.

Je nach Diagrammtyp werden bestimmte Beziehungen zwischen Modellelementen automatisch abgeleitet. Beispielsweise enthält eine Funktionshierarchie *Akteure*, die wiederum *Akteure* enthalten; zwischen ihnen wird also eine „*enthält*“ Beziehung angelegt. Manche Beziehungen lassen sich nicht automatisch identifizieren, wie zum Beispiel eine „*erfüllt*“ Beziehung zwischen einer *Anforderung* und einem *Akteur* (Funktion), und sind folglich manuell anzulegen.

Verwendung eines vereinbarten Vokabulars: Eine gemeinsame Begriffswelt (auch Glossar, Terminologie oder Ontologie genannt) erleichtert den Datenaustausch zwischen Systemen, indem Objekte und Attribute automatisch zugeordnet werden können. Es

unterstützt auch die Kommunikation zwischen Projektpartnern, indem Begriffe mit vereinbarter Bedeutung verwendet werden [Spec17].

Zusammenfassend sei gesagt, dass das *Verstehen* der Modelle die ersten drei Prinzipien der Modellintegration betrifft. Das vierte Prinzip dient dem *Verknüpfen* der Modell-Elemente.

3 Anwendungsbeispiel Kaffeemaschine

Anhand des Tool-Vendor-Project des TdSE wird gezeigt, wie Teilmodelle maschinell verstanden und verknüpft werden: Ein Kaffeevollautomat am Flughafen soll verschiedene Kaffeesorten zu verschiedenen Preisen zubereiten. Aus den vorgegebenen Anforderungen entstand eine Anforderungsliste, aus der die Funktionshierarchie und die Systemstruktur abgeleitet wurden. Anschließend entstanden ein Zustands- und ein Aktivitätsmodell. Nachfolgend werden die Teilmodelle kurz beschrieben.

Anforderungsliste

Die Anforderungen aus dem Tool-Vendor-Project wurden um einige Punkte erweitert und mit MS-Excel erfasst. Ein Auszug der Anforderungsliste ist in Bild 2 zu sehen, wobei mit Rücksicht auf die Lesbarkeit einige in dieser Betrachtung weniger wichtige Spalten nicht gezeigt werden.

Typ	Titel	Beschreibung	Quelle	Funktion
Quality	Sicherheit (Geld)	Das Geld (EURO) muss sicher verwahrt werden	Systemarchitekt	
Quality	EC-Karte	Kunde möchte mit EC Karte bezahlen können	Kunde	EC einlesen
Quality	Kreditkarte	Kunde möchte mit Kreditkarte bezahlen können	Kunde	Kreditkarte einlesen
Quality	Geldscheine	Das Gerät muss Geldscheine bis 10 Euro annehmen können	Kunde	Scheine zählen, Scheine validieren
Quality	Bedienung	Die Bedienung muss für alle Reisenden und Mitarbeiter schnell und einfach zu bedienen sein.	Marketing	
Quality	H-Milch	Kunde möchte auch H-Milch haben.	Kunde	
Quality	Gemahlter Kaffe	Der Kaffee muss frisch gemahlen werden.	Kunde	Mahlwerk ausführen
Function	Becherausgabe	Vollautomatische Lösung mit Becherausgabe (600 Stück)	Entwicklungsabteilung	Kaffeebecher ausgeben
Function	Bechererkennung	Erkennung von Bechern. Bei Nichtexistenz wird ein Becher ausgegeben.	Entwicklungsabteilung	Kaffeebecher erkennen
Quality	Wasserfiltersyster	BRITA Wasserfiltersystem.	Entwicklungsabteilung	
Constraints	Kaffeebohnensort	Arabica (Standard).	Entwicklungsabteilung	

Bild 2: Auszug der Anforderungsliste.

Für die Modellintegration mit SpecIF wird zunächst anhand der ersten Zeile ein Objekt-Typ ↯ *Anforderung* angelegt mit je einem Attribut pro Spalte, wobei der Spaltenkopf in Zeile 1 als Attributname („*title*“) herangezogen wird.

Ab der zweiten Zeile wird jeweils eine Instanz des angelegten Typs mit den Spalten-inhalten als Attributwert („*value*“) erzeugt. In der Spalte *Funktion* werden jene aufgeführt, die die betreffende Anforderung erfüllen soll; aus dieser Angabe wird im Modellkern zwischen der betreffenden Anforderung und der genannten Funktion eine Beziehung „*erfüllt*“ angelegt („manueller Tracelink“).

Funktionshierarchie

Die Funktionshierarchie bricht die Gesamtfunktion des Systems systematisch auf atomare Teilfunktionen herunter. Aus jedem Knoten wird ein Objekt des Typs „■ *Akteur*“ erzeugt, wenn nicht vorhanden. Vom Planobjekt wird eine Beziehung *„zeigt“* zu jedem Knoten hinterlegt und aus jeder Kante (hier Verbindung durch Pfeil) wird automatisch eine Beziehung des Typs „*enthält*“ abgeleitet; beides automatisch.

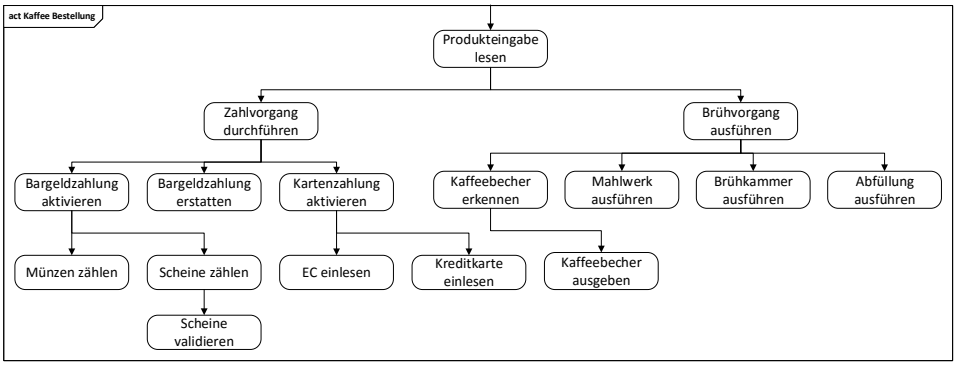

Bild 3: Funktionshierarchie unterhalb der Funktion *Produkteingabe lesen*.

Systemstruktur

Die Systemstruktur gliedert das Gesamtsystem hierarchisch in Komponenten bis zu Bauelementen auf. Aus jedem Knoten wird ein Objekt des Typs „■ *Akteur*“ erzeugt, sofern nicht vorhanden. Vom Planobjekt wird eine Beziehung vom Typ *„zeigt“* zu jeder vorkommenden Komponente angelegt und aus jeder Kante wird eine Beziehung des Typs „*enthält*“ gebildet; wiederum beides automatisch.

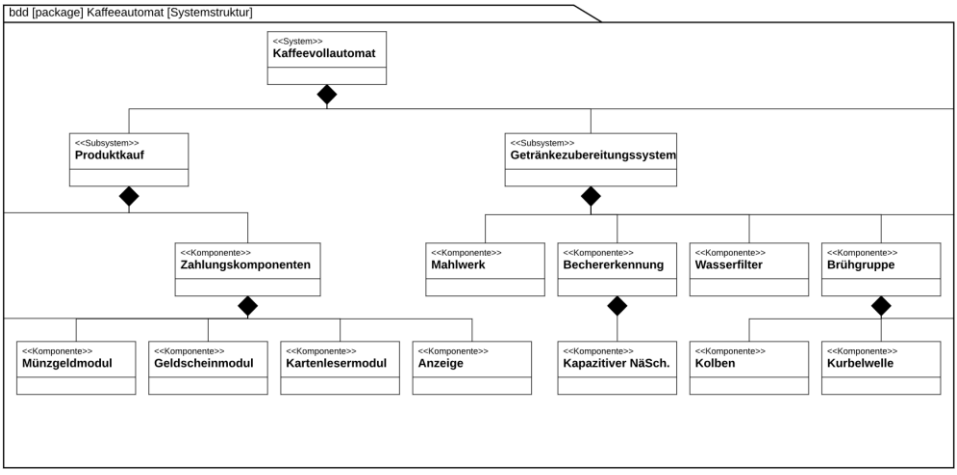

Bild 4: Auszug der Systemstruktur.

Zustandsdiagramm

Zustandsdiagramme betreffen das Verhalten und beschreiben die funktionalen Abläufe im System. Bild 5 zeigt, welche Funktionen beim Zustandsübergang auszuführen sind, die natürlich auch in der Funktionshierarchie zu finden sind. Zustände werden mit Objekttyp „● *Zustand*“ und Funktionen im Zustandsübergang als „■ *Akteur*“ im Modellkern zugeordnet bzw. neu angelegt. Weiterhin werden folgende Beziehungen automatisch hinterlegt: Das Planobjekt *„zeigt“* jeden Zustand und jede Funktion im Zustandsübergang, ein Zustand „*erlaubt*“ eine Funktion und eine Funktion „*resultiert in*“ einem Zustand.

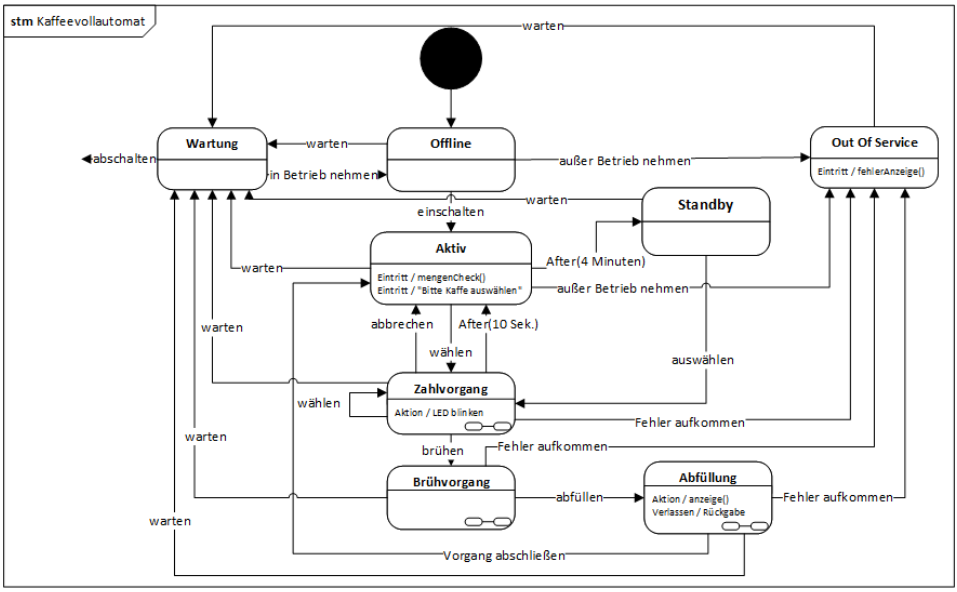

Bild 5: Zustandsdiagramm der TdSE Kaffeemaschine.

Aktivitätsdiagramm

Aktivitätsdiagramme zeigen, in welcher zeitlichen Reihenfolge untergeordnete Funktionen im Übergang zwischen zwei Zuständen ausgeführt werden. Ein bestimmter Ablauf wird von einem Ereignis im Startzustand ausgelöst; hier mit dem Symbol des Splittings dargestellt. Aus jedem Zustand wird ein Objekt des Typs „● *Zustand*" erzeugt oder einem vorhandenen zugeordnet. Ereignisse erhalten den Objekttyp „♦ *Ereignis*“, während die abzuarbeitenden Aktivitäten vom Typ „■ *Akteur*“ sind. Zwischen Zustand und Ereignis wird die Beziehung des Typs „*erfährt*“ verwendet. Zwischen Ereignis und Funktion entsteht eine Beziehung „*löst aus*“. Eine Funktion „*führt zu*“ der nächsten Funktion. Eine Aktivität endet, wenn ein Zielzustand erreicht wird. Sie erhält die Beziehung „*resultiert in*“.

Natürlich müssen sich die aus verschiedenen Plänen (hier Funktionshierarchie, Zustands- und Aktivitätsdiagrammen) abgeleiteten Beziehungen sinnvoll ergänzen. Tatsächlich kann automatisch geprüft werden, ob die jeweiligen Angaben zueinander passen.

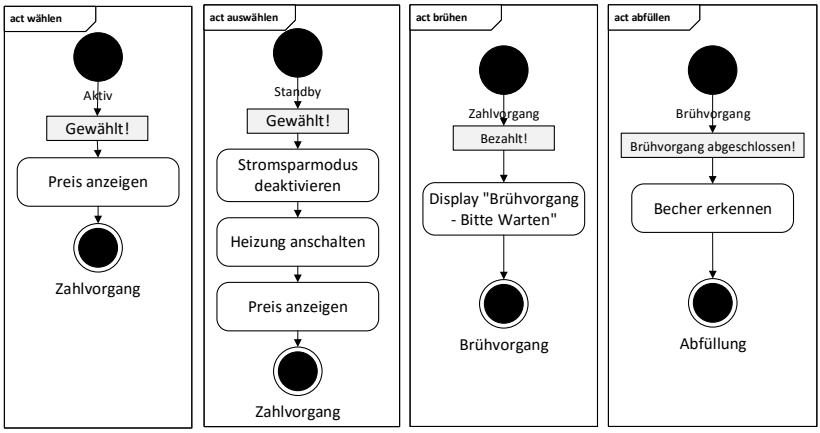

Bild 6: Eine Auswahl von Aktivitätsdiagrammen.

Ergebnis

Mit einer prototypischen Software wurden die Anforderungen aus MS-Excel und die Modelldiagramme aus MS-Visio automatisch in ein Integrationsmodell gemäß SpecIF überführt. Bild 7 stellt die Beziehungen des Modellelements „*Brühvorgang*“ vom Typ „● *Zustand*" nach Import in das „Interaktive Lastenheft“ dar: Von welchen Diagrammen es gezeigt wird, zu welchen Aktivitäten er potentiell führen kann, welche Aktivitäten in ihm resultieren, et cetera.

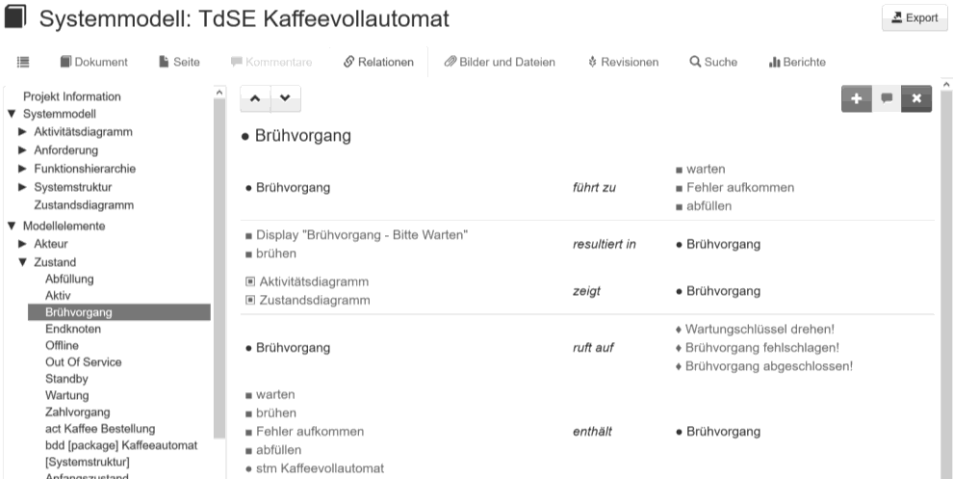

Bild 7: Automatisch erzeugte Relationen („Tracelinks“) des Modellelements „Brühvorgang“.

Das vollständige semantisch vernetzte Modell kann von den Autoren angefordert werden.

4 Nachvollziehbarkeit der Zusammenhänge

Traceability ist die explizite Nachvollziehbarkeit der Zusammenhänge zwischen Produktinformationen und gilt als eine wichtige Voraussetzung für die Entwicklung hochqualitativer Produkte. Auch die Einhaltung von Normen (u.a. ISO 15504, ISO 26262) und Richtlinien setzt Traceability voraus: Durch Beziehungen zwischen den Artefakten („Tracelinks“) werden komplexe Zusammenhänge in der Systementwicklung für die Beteiligten explizit verfügbar: Welche Funktionen sind verantwortlich für die Erfüllung einer Anforderung? Welche Tests verifizieren die Eignung eines Bauteils und was sind die entscheidenden Testparameter? Wie stellt ein Softwareentwickler sicher, dass die Entwicklungstätigkeiten in anderen Domänen, z. B. Mechanik und Elektronik, keine Konflikte mit dem Code verursachen?

Tracelinks lassen sich in zwei Hauptkategorien einteilen: Struktur- bzw. verhaltensbezogene Beziehungen einerseits sowie logische Zusammenhänge andererseits. Strukturbezogene Beziehungen stellen Hierarchie und Zugehörigkeit von Modellelementen dar. Verhaltensbezogene Beziehungen betreffen kausale und temporale Abhängigkeiten zwischen Zuständen, Ereignissen und Aktivitäten. Logische Zusammenhänge hingegen zeigen technische Beziehungen, wie „*satisfy“*, „*verify“*, „*derive“*, und „*refine"* [OMG15].

Die prototypische Anwendung zur Modellintegration hat gezeigt, dass struktur- und verhaltensbezogene Tracelinks, automatisch aus Diagrammen zu ermitteln und im SpecIF Integrationsmodell darstellbar sind (Beziehungstypen *„enthält“, „löst aus“* u.a.). Dabei spielt es keine Rolle, ob die Autorensysteme selbst Semantik kennen.

Logische Zusammenhänge sind hingegen zumeist manuell anzulegen. Als Beispiel wurde in der Anforderungsliste (Excel-Datei) eine Spalte vorgesehen, um für die jeweilige Anforderung betroffene Funktionen eintragen zu können. Dadurch konnten im Integrationsmodell die Tracelinks zwischen Anforderungen und Funktionen (Beziehungstyp „*satisfies*“ = „*erfüllt*“) hinterlegt werden.

5 Modellbasierte vs. dokumentenbasierte Entwicklung

Die Vorteile des modellbasierten Entwickelns sind in Wissenschaft und Industrie bestens bekannt. Die steigende Produktkomplexität durch zunehmende Kundenansprüche und Vernetzung zwingen Produktentwickler zu einem Paradigmenwechsel in ihren bisherigen Methoden, Werkzeugen und Vorgehensweisen. Studien [Kö12] und unsere Erfahrungen in vielen Projekten belegen jedoch, dass trotz des breiten Produktspektrums für modellbasiertes Entwickeln weiterhin sog. „Office-Lösungen“ verbreitet im Einsatz sind.

Dafür gibt es etliche Gründe. Erstens erfordert modellbasiertes Entwickeln ein erhebliches Investment in Ausbildung von Mitarbeitern, Entwicklungsprozesse und IT-Systeme, zweitens ist sind bestimmte Aspekte eines Produktes (z. B. Anforderungen, Stücklisten) mittels Dokumenten ausreichend zu beschreiben und schließlich sind Office-Lösungen an jedem Arbeitsplatz verfügbar und stellen eine niedrige Eintrittshürde dar.

6 Zusammenfassung und Ausblick

Einen modellbasierten Ansatz in der Produktentwicklung einzuführen erfordert ganzheitliches Umdenken, das Implementieren neuer IT-Systeme und Methoden, eine Änderung der Entwicklungs- und Freigabeprozesse bis hin zur Dokumention. Wir postulieren, dass ein modellbasiertes Arbeiten auch durch automatisches Auswerten und Integrieren von Diagrammen und anderen Inhalten aus Office-Tools möglich ist. Dies verringert die wirtschaftlichen, organisatorischen und technischen Hürden beim Paradigmenwechsel von dokumentenbasierter zu modellbasierter Entwicklung.

In diesem Beitrag zeigen wir, wie aus Diagrammen und Listen unterschiedlicher Notationen ein integriertes Systemmodell erzeugt und in ein Zielsystem überführt werden kann. Neben den Inhalten sind auch die Traceability Informationen darstellbar und übertragbar.

Zukünftige Forschung- und Entwicklungsaktivitäten haben zum Ziel, weitere Diagrammtypen der SysML, BPMN und andere Notationen zu interpretieren. Systematische Abbildungsvorschriften zwischen gebräuchlichen Notationen und SpecIF sind zu erarbeiten. Auch die Fähigkeit, die Modelle und Dokumente in andere Zielsysteme zu überführen, wird einer unserer Forschungsschwerpunkte sein.

Literaturverzeichnis

[Knö05] Knöpfel, A.; Gröne, B.; Tabeling, P.: Fundamental Modelling Concepts – Effective Communication of IT Systems. ISBN-13: 978-0-470-02710-3. John Wiley & Sons, Chichester, 2005.

[Kö12] Königs, Simon Frederick; Beier, Grischa; Figge, Asmus; Stark, Rainer (2012): Traceability in Systems Engineering – Review of industrial practices, state-of-the-art technologies and new research solutions. In: Advanced Engineering Informatics 26 (4), S. 924–940. DOI: 10.1016/j.aei.2012.08.002.

[Dun14] Dungern, O.v.: Übergreifende Konzeption von Geräten für die Gebäudeautomation – Methodik und Management. TdSE Tag des Systems Engineering der GfSE, November 2014. http://specif.de/files/resources/enso-m/documents-de/TdSE-2014_Dungern_Uebergreifende-Systemkonzeption_(Text).pdf.

[OMG15] SysML 1.4 Specification. September 2015. http://www.omg.org/spec/SysML/1.4/

[Dun16] Dungern, O.v.: Semantic Model Integration for System Specification - Creating a Common Context for Different Model Types. TdSE Tag des Systems Engineering der GfSE, November 2016. http://specif.de/files/resources/enso-m/documents-en/TdSE-2016_Dungern_Semantic-Model-Integration-for-System-Specification_(Text).pdf.

[Spec17] SpecIF – Open SE Models.: Systemmodelle zusammen führen mit SpecIF. 2017. http://specif.de/.