Integration textueller Anforderungen und Modell-basiertem Testen mit SysML

Oliver Alt
Continental
Division Chassis & Safety
Electronic Brake Systems, ETC-PM
Guerickestrasse 7, D-60488 Frankfurt am Main
oliver.alt@contiautomotive.com

1 Einleitung

In der modernen System- und Softwareentwicklung bilden Anforderungen die Grundlage aller Entwicklungsaktivitäten. Diese Anforderungen liegen zumeist in nicht formaler Form als textuelle Anforderung oder Beschreibung von Anwendungsfällen vor. Neben diesen informellen Entwicklungsdokumenten gibt es im Rahmen einer Modell-basierten Entwicklung aber auch die formalen Systemmodelle, die das System in seiner Architektur und seinem Verhalten beschreiben.

Damit ergibt sich eine Lücke zwischen den Anforderungen und den Systemmodellen, die nur durch den menschlichen Entwickler gefüllt werden kann, der aus den Anforderungen das Systemmodell erstellt. Ein formale, im Sinne von rechnerunterstützter Überführung der Anforderungen in Modelle und umgekehrt ist aufgrund des rein informellen Charakters der Anforderungen nicht möglich.

Im Folgenden wird ein Verfahren vorgestellt, das es erlaubt, Anforderungen und Anwendungsfälle mit Hilfe von Modellierungsmustern in ein formales Systemmodell in Form von SysML-Aktivitätsdiagrammen zu überführen. Umgekehrt lassen sich aus diesen Diagrammen auch die informellen Anforderungen leicht gewinnen. Aus den erstellten Modellen lassen sich mit Hilfe von bereits erarbeiteten Verfahren Testfälle generieren und ausführen. Damit kann ein solches Modell als Testgrundlage in einem Testprozess Verwendung finden.

2 SysML

Die Systems Modeling Language (SysML) [OMG07] ist eine neue standardisierte grafische Modellierungssprache zur Erstellung von Systemmodellen. SysML basiert auf der Softwarebeschreibungssprache UML2. Viele bewährte Konzepte der UML2 werden wiederverwendet, aber auch neue Elemente und Diagramme hinzugefügt und Teile weg gelassen um die Erfordernisse des System Engineering optimal zu unterstützen.

Eine der wichtigsten Neuerungen der SysML ist sicherlich das Anforderungsdiagramm und das Anforderungselement. Damit lassen sich textuelle Anforderungen erstmals als Elemente im Systemmodell darstellen und mit anderen Modellelementen verknüpfen. Dies ermöglicht zum Beispiel eine Zuordnung von Anforderungen zu Architekturelementen oder Verhaltenselementen. Somit lässt sich eine durchgängige Tracebility zwischen Anforderungen und den Systemmodellelementen herstellen, wie sie von vielen Entwicklungsprozessstandards heute eingefordert wird. Durch den Einsatz von SysML kann damit auf der informellen Ebene bereits eine Integration zwischen Anforderungen und dem Systemmodell erzielt werden.

3 Umsetzung von Anforderungen in ein Systemmodell

Um eine solche Integration nicht nur informell zu erreichen, muss ein Verfahren gefunden werden mit dem ein Zusammenhang zwischen einer Anforderung und dem Systemmodell auf einer direktere Art und Weise hergestellt werden kann. Dazu wurden Anwendungsfälle, Anforderungen und Systemmodelle im Hinblick auf Gemeinsamkeiten und Abhängigkeiten analysiert.

Als Illustrationsbeispiel für das entwickelte Verfahren dient ein Tempomatsystem, also eine automatische Geschwindigkeitsregelung im Fahrzeug [MDC07].

3.1 Externe und interne Systemsicht

Als Ergebnis der Analyse ergibt sich ein Systembild wie es in Abbildung 1 dargestellt ist. Ein Teil der Anforderungen beschreibt das System aus externer Sicht und sieht das System dabei als Black-Box an. Diese Sicht ist auch diejenige, die ein Systembenutzer hat. Er benutzt ausschließlich die zur Verfügung stehenden Schnittstellen, um mit dem System zu interagieren. Oftmals enthalten diese Anforderungen auch Bedingungen wann es einem Benutzer gestattet ist, eine bestimmte Aktion mit dem System durchzuführen. Bei den Anwendungsfällen findet sich so etwas als Beschreibung der Vorbedingung, die erfüllt ein muss, um den Anwendungsfall durchzuführen.

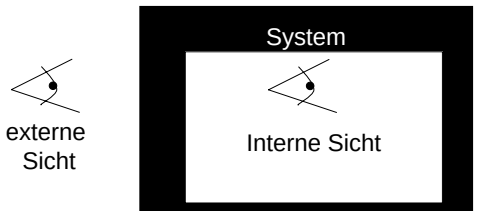

Abbildung 1: Externe und interne Systemsicht

Der externen Sicht gegenüber steht die interne Systemsicht, die beschreibt wie sich das System intern verhalten und funktionieren soll. Dort ist definiert welche Art von Aktionen das System ausführen muss, um die gewünschte Funktionalität zu erfüllen.

Als drittes findet sich die Beschreibung der Verknüpfung beider Sichten, das heißt, es wird beschrieben

wie sich eine Aktion eines externen Systembenutzers auf das Systemverhalten auswirkt und welche der internen Abläufe wie beeinflusst werden.

3.2 Verhaltensmodellierung mit SysML-Aktivitätsdiagrammen

Viele etablierte und auch neuere Verfahren im Bereich des Modell-basierten Testens benutzen Zustandsdiagramme, um das Systemverhalten zu beschreiben.

Im Gegensatz dazu wurde hier bewusst die SysML-Aktivitätsmodellierung ausgewählt und dies begründet sich wie folgt:

- Durch die Verwendung von Aktivitätsdiagrammen brauchen im Gegensatz zu Zustandsautomaten die Systemzustände nicht vor Modellerstellung gefunden werden, sondern ergeben sich aus dem Systemmodell implizit.
- Anforderungen und insbesondere Anwendungsfallbeschreibungen benutzen Aktivitäten, um die Systemfunktionalität zu beschreiben. Daher können solche Beschreibungen leicht in ein Aktivitätsdiagramm umgesetzt werden.

Die Elemente der Aktivitätsdiagramme wurden bereits mit der Version 2.0 der UML stark erweitert. Die SysML erweitert diese neuerdings um neue Elemente. Insbesondere führt die SysML einen neuen Typ von Aktion ein, den so genannten *Kontrolloperator*, der es ermöglicht, das Laufzeitverhalten einer Aktivität bzw. Aktion von Außen zu beeinflussen. Bislang war es nicht möglich Aktionen beispielsweise in ihrer Ausführung zu unterbrechen und an gleicher Stelle später fortzusetzen. Dies wird nun durch den Kontrolloperator möglich und erweitert damit die Aktivitätsmodellierung um ein Konstrukt, das wie sich zeigt für die Formalisierung von Anforderungen benötigt wird.

3.3 Definition der Modellierungselemente

Bevor mit der Verhaltensbeschreibung begonnen werden kann müssen die erforderlichen Elemente, also Aktivitäten und Objekte definiert werden. Um die externe von der internen Systemsicht abzugrenzen werden die jeweiligen Aktivitäten entsprechend gekennzeichnet:

Benutzeraktivitäten sind all die externen Aktivitäten, die ein Benutzer des Systems mit diesem ausführen kann (externe Sicht).

Systemaktivitäten sind die Aktivitäten die das System intern durchführt (interne Sicht).

Systemgrößen speichern als Instanzen (Objekte) interne Systemvariablen.

Abbildung 2 zeigt die definierten Aktivitäten und Systemgrößen für das Tempomat-Beispielsystem.

Mit diesen Elementen lässt sich das Verhalten des Systems modellieren. Um die informellen Anforderungsdokumente zu formalisieren wurden unter Verwendung der Aktivitätsmodellierung drei Modellierungsmuster entwickelt,

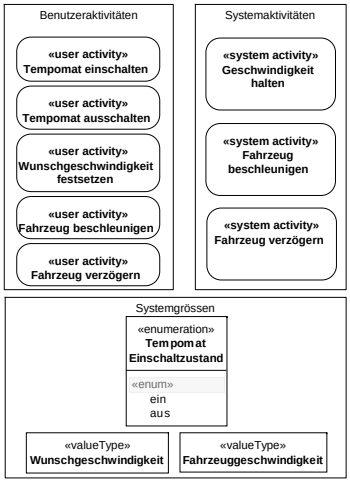

Abbildung 2: Definition der Modellelemente für das Beispielsystem

die die in Abschnitt 3.1 gefundenen Anforderungstypen umsetzten: Dies ist zum Einen die Beschreibung der Vorbedingung wann eine externe Benutzeraktion durchgeführt werden kann. Zum Anderen die Darstellung wie sich eine solche Durchführung auf das System auswirkt und zu dritten wie das interne Systemverhalten aussieht.

Im Folgenden werden diese drei Muster für das Beispielsystem anhand der Benutzeraktivität `Wunschgeschwindigkeit festsetzen` exemplarisch gezeigt.

Vorbedingungen von Benutzeraktionen

Für jede Benutzeraktivität wird eine Vorbedingung modelliert. Dies wird mit Hilfe von Entscheidungsknoten und Kontrollflusskanten mit den entsprechenden Bedingungen ausgedrückt. Wenn die Vorbedingung erfüllt ist soll die Benutzeraktivität ausgeführt werden. Dies wird dadurch ausgedrückt, dass eine `SendSignal`-Aktion als Instanz der Benutzeraktivität durch die Kontrollflüsse aktiviert wird.

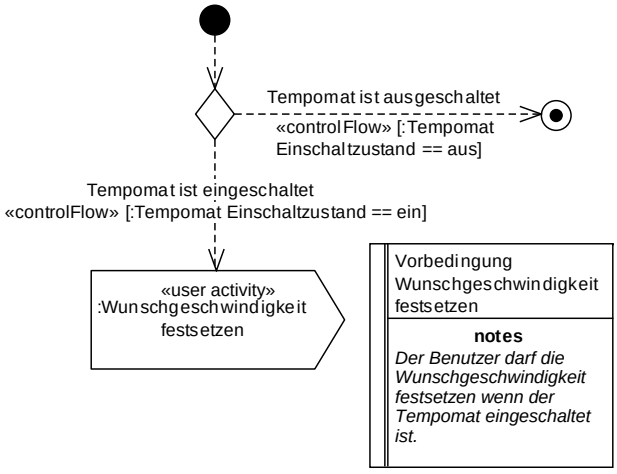

Abbildung 3: Modellierungsmuster für Vorbedingungen

Abbildung 3 zeigt die modellierte Vorbedingung für die Aktivität `Wunschgeschwindigkeit festsetzen` und die zugehörige äquivalente Anforderung. Die Aktivitätsmodellierung erfordert auch die Alternativpfade zu modellieren und geht damit sogar über die Anforderung hinaus (Nichterfüllung der Vorbedingung).

Verknüpfung von externem und internem Verhalten

Um das externe und interne Systemverhalten zu verknüpfen werden durch die AcceptEvent-Aktion als Signalempfänger und Gegenstück zur Vorbedingung entsprechende Abläufe zur Änderung des Systemzustands initiiert.

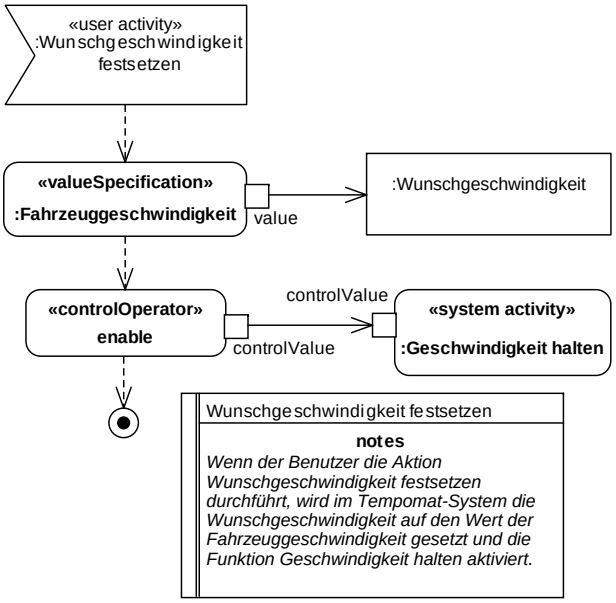

Abbildung 4: Zuordnung von externen Aktionen zu internem Verhalten

Die Verknüpfung des externen und internen Systemverhaltens, die zugehörige Anforderung und die entsprechende Formalisierung durch die Modellierung zeigt Abbildung 4.

Im Beispiel werden durch die Benutzeraktion eine `valueSpecification`- und eine `control operator`-Aktion gestartet. Die valueSpecification ist in der Lage einen Objektwert zu ändern. Der Kontrolloperator erlaubt eine Änderung des Laufzeitzustandes einer Aktion von außen ohne einen expliziten Kontrollfluss (starten, stoppen, anhalten, fortsetzten etc.).

Mit Hilfe des Kontrolloperators und der valueSpecification kann der Systemzustand, der sich aus den Systemgrössen und den Laufzeitzuständen der Systemaktivitäten zusammensetzt durch die Benutzeraktionen beeinflusst werden.

Modellierung des internen Verhaltens

Als letztes muss noch das interne Verhalten der Systemaktion(en) beschrieben werden. Dies geschieht neuerlich durch ein Aktivitätsdiagramm und unter Verwendung der Kontrolloperatoren und valueSpecification-Aktionen.

Abbildung 5 zeigt das interne Verhalten der durch dir Benutzeraktion gestarteten Systemaktion `Geschwindigkeit halten`.

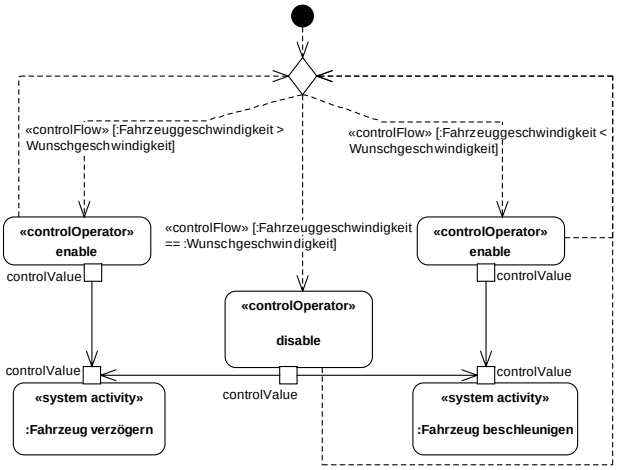

Abbildung 5: Internes Verhalten der Systemaktivität `Geschwindigkeit halten`

Ein solches Diagramm ergibt sich aufgrund der Verzweigungen und unterschiedlichen Pfade aus mehreren Anforderungen, die die Systemaktion beschreiben. Die drei zugehörigen Anforderungen sind in Abbildung 6 gegeben.

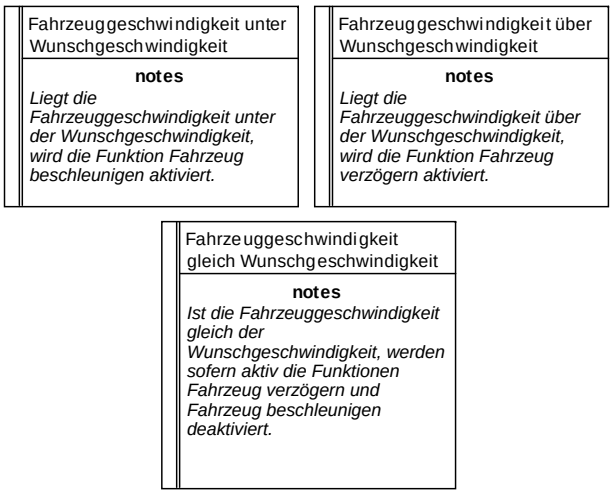

Abbildung 6: Beschreibung des internen Verhaltens als Anforderungen

Die gezeigten Anforderungen in den Beispielen sind etwas idealisiert um die Zuordnung der Diagramme zu den Texten deutlich zu illustrieren. Jedoch haben sich das vorgestellte Verfahren und die Muster bereits in der Praxis, bei der Formalisierung von Anforderungen bewährt. Mit etwas Übung gelingt es leicht auch aus etwas anders formulierten Anforderungen solche formalen Modelle aufzubauen.

4 Modell-basierte Testfallgewinnung

Mit einem nach den oben beschriebenen Mustern aufgebautem Modell lassen sich mit Hilfe des in früheren Arbeiten entwickelten Verfahrens (vgl. [SA07], [Alt07]) nun Zustandsautomaten und letztendlich ausführbare Testfälle für den Systemtest gewinnen.

Das Verfahren basiert darauf, das eine sukzessive Ausführung der Benutzeraktivitäten die möglichen Systemzustände nacheinander hervorruft. Ein Zustandsautomat bildet das Ergebnis einer solchen Modellsimulation. Mit Hilfe bekannter Verfahren lassen sich aus einem solchen Graphen dann Testfälle nach vordefinierten Überdeckungskriterien ableiten.

Mit den vorgestellten Modellierungsmustern kann damit auf einfache Art und Weise ein Verhaltensmodell aus Anforderungen aufgestellt werden, das nicht nur als formalere Dokumentation, sondern auch als Grundlage für einen Modell-basierten Test und die Testfallgenerierung dienen kann. Auch der Umgekehrte Weg der Ableitung von Anforderungen aus solchen Verhaltensmodellen ist möglich und kann dazu beitragen alle notwendigen Anforderungen die das modellierte Systemverhalten beschreiben zu erfassen. Dies kann beispielsweise dann eine Rolle spielen wenn innerhalb eines Entwicklungsprozesses textuelle Anforderungen gefordert werden.

5 Verwandte Arbeiten

Ein werkzeuggestütztes Verfahren zur Extraktion von Testinformationen aus Anforderungsspezifikationen beschreibt Sneed in [Sne05]. Bei diesem Verfahren werden bestimmte Schlüsselworte erfasst und daraus Testinformationen abgeleitet.

Einen ähnlichen Mechanismus, allerdings mit dem Fokus auf Anwendungsfallbeschreibungen und deren Formalisierung beschreibt Friske in [FP05] und [FS07]. Dabei werden Anwendungsfallbeschreibungen halbautomatisch in UML-Aktivitätsdiagramme umgesetzt. Das beschriebene Verfahren ist sicherlich mit dem hier vorgestellten Ansatz kombinierbar, da es so abgewandelt werden kann ,dass die vorgestellten Aktivitätsmuster berücksichtigt werden.

Grünbauer beschreibt in [Grü08] ein Verfahren um Abhängigkeiten zwischen verschiedenen Systemfunktionen zu modellieren und für die Verifikation der Systeme zu nutzen. Im Gegensatz zum hier beschriebenen Verfahren kommt dabei jedoch keine Standardsprache wie SysML sondern ein eigens entwickeltes Modellierungskonzept zum Einsatz.

In der Arbeit von Bramsiepe, Sikora und Pohl [BSP08] werden Anforderungen aus Zielen und Szenarien abgeleitet. Dabei werden keine Aktivitätsdiagramme wie hier, sondern Sequenzdiagramme eingesetzt. Die hier verwendeten Aktivitätsdiagramme haben den Vorteil, dass auch Alternativen in einem Diagramm durch den Einsatz von Verzweigungen übersichtlich beschrieben werden können. Mit den Aktivitäsdiagrammen lassen sich somit prinzipiell auch mehrere solcher Szenarien beschreiben oder daraus ableiten.

6 Zusammenfassung und Ausblick

Mit Hilfe von SysML und den hier beschriebenen Modellierungsmustern wird die Formalisierung von Anforderungen erleichtert und damit die Lücke zwischen Anforderungen und formaler Systemmodellierung ein Stück weit verkleinert. Durch die Verwendung von SysML lassen sich darüber hinaus die Anforderungen auch ins Systemmodell integrieren und eine durchgängige Tracebility zwischen Anforderung, Modell und Testfall herstellen.

Weiterhin ermöglichen die vorgestellten Modellierungsmuster auch eine Herleitung von textuellen Anforderungen aus dem Systemmodell. An Konzepten eine solche Ableitung (teilweise) zu automatisieren, und damit das Modell in den Mittelpunkt der Entwicklung zu rücken, wird momentan gearbeitet. Damit könnte die Lücke zwischen Modell-basierter Entwicklung und Requirement Engineering mittelfristig zumindest in die Richtung der Anforderungen geschlossen werden.

Literatur

[Alt07] O. Alt. Deriving reusable system test cases from SysML models. In *Proceedings of the Software and Systems Quality Conference*, Düsseldorf, 2007.

[BSP08] N. Bramsiepe, E. Sikora und K. Pohl. Ableitung von Systemfunktionen aus Zielen und Szenarien. In GI, Hrsg., *Softwaretechnik-Trends*, Jgg. 28, Seiten 13–14, 2 2008.

[FP05] Mario Friske und Holger Pirk. Werkzeuggestützte interaktive Formalisierung textueller Anwendungsfallbeschreibungenfür den Systemtest. In *Beiträge der 35. Jahrestagung der Gesellschaft für Informatik (Band 2)*. A. B. Cremers and R. Manthey and P. Martini and V. Steinhage, 9 2005.

[FS07] M. Friske und H. Schlingloff. Generierung von UML-Modellen aus formalisierten Anwendungsfallbeschreibungen. In M. Conrad, H. Giese, B. Rumpe und B. Schätz, Hrsg., *MBEES - Model Based Engineering of Embedded Systems III*. Dagstuhl-Workshop, TU Braunschweig Report TUBS-SSE 2007-01, 1 2007.

[Grü08] J. Grünbauer. Erkennung von Feature Interactions auf Nutzungsebene - Modellierung und Verifikation von Abhängigkeiten. In GI, Hrsg., *Softwaretechnik-Trends*, Jgg. 28, Seiten 5–6, 2 2008.

[MDC07] MDC. *Bedienungsanleitung MDC Tempomat für smart for4 & Mitsubishi Colt CZ3/CZ30/CZT Version 1.3*, 2007. http://www.misterdotcom.de/BedienungsanleitungTempomat-smartfor4-Colt.pdf.

[OMG07] OMG. OMG Systems Modeling Language (OMG SysML^{TM}) - V 1.0. formal/2007-09-01, Object Management Group, 9 2007.

[SA07] S. Schlecht und O. Alt. Strategien zur Testfallgenerierung aus SysML Modellen. In W.-G. Bleek, H. Schwetner und H. Züllighoven, Hrsg., *Software Engineering 2007 - Beiträge zu den Workshops*, Seiten 101–106, 3 2007.

[Sne05] Harry M. Sneed. Reverse Engineering deutschsprachiger Fachkopnzepte. *Softwaretechnik-Trends*, 25(2):45, Mai 2005.