Guidelines Are a Modeler's Best Friend

Ein Einstieg in die statische Modellanalyse

Dr. Simon Rösel, Model Engineering Solutions GmbH

Die modellbasierte Entwicklung ist eine moderne Methode, um eingebettete Regel- und Steuersysteme zu entwickeln. Das gewünschte Systemverhalten wird durch ein Modell als zentrales Entwicklungsartefakt beschrieben. Entsprechende Teile des Modells bilden den Ausgangspunkt für die automatisierte Codegenerierung für Steuergerätesoftware, die der Laufzeitumgebung angepasst wird und anschließend auf den Controller geladen werden kann. Dabei ist der Einsatz von Modellierungsrichtlinien im Hinblick auf Funktionale Sicherheit und Qualitätssicherung unverzichtbar.

Statische Modellanalyse: Die Anwendung von Richtlinien und Standards

Im Automotive-Bereich sind die meisten Steuergeräte-Funktionen, die mit Hilfe der modellbasierten Softwareentwicklung entstehen, sicherheitsrelevant. Standardisierte verlässliche Methoden sind somit unerlässlich. Für die Entwicklung elektrisch/elektronischer Kraftfahrzeugsysteme spielt die ISO 26262 („Road Vehicles – Funtional Safety") als zentrale ISO-Norm eine herausragende Rolle [1]. Die ISO 26262 widmet das komplette Kapitel 6 der Produktentwicklung auf Softwareebene und empfiehlt dabei explizit den Einsatz von semi-formalen Modellierungssprachen, wie z.B. Simulink.

Neben Fragen der Funktionalen Sicherheit steht in Softwareentwicklungsprozessen die Qualitätssicherung im Vordergrund. Beide Aspekte hängen wesentlich vom effizienten Einsatz von Modellierungs- und Konformitätsrichtlinien ab. Dabei gilt der Grundsatz, dass die Modellqualität für den gesamten Entwicklungsprozess entscheidend ist und damit auch die Qualität der generierten Software bestimmt. Weiter finden auch bekannte Elemente der statischen Quellcodeanalyse (z.B. Range Checking, Komplexitätsanalysen, Prüfung starker Typisierung) ihre Entsprechung in der statischen Modellanalyse.

Quellen für Modellierungsrichtlinien, Beispiele

Modellierungsrichtlinien lassen sich grob hinsichtlich des relevanten Modellierungsgegenstandes, der Werkzeugspezifik und der jeweiligen Ziele unterscheiden. Design-Aspekte von Simulations- und Controller-Modellen stehen bei den *MathWorks Automotive Advisory Board (MAAB)*-Regeln im Vordergrund und fördern durch die Einhaltung von Best Practices die Les- und Wartbarkeit [2]. Die *MISRA Simulink/Stateflow*- und *MISRA TargetLink-Regeln* stellen auf Sicherheitsaspekte der Modelle und des daraus zu generierenden Codes ab [3] [4].

Die *dSPACE TargetLink Modeling Guidelines* beziehen sich ebenfalls auf die effiziente Code-Generierung mit TargetLink [5]. Insbesondere schließen diese Richtlinien die Verwendung von Modellierungsmustern aus, die inkompatibel mit der automatischen Codegenerierung sind, oder zu ineffizientem Code führen. Mit der gleichen Intention werden auch einheitliche Festlegungen für Modellkonfigurationen und Code-Generator-Einstellungen getroffen.

Steht der Entwurf sicherheitsrelevanter Software im Vordergrund steht, finden die *MES Functional Safety Guidelines*, die aus den Anforderungen der ISO 26262 und anderen Sicherheitsstandards abgeleitet sind, Anwendung [6]. Zur optimalen Vorbereitung auf dynamische Modelltests dient das *MES Fit for Testing* Richtliniendokument [7]. Hier kommen insbesondere Regeln zum Tragen, die eine eindeutige Beziehung zwischen Modell- und Testobjekten sowie die automatische Testbed-Generierung unterstützen.

Die vollständige Abdeckung aller relevanten Aspekte erfordert eine sinnvolle Kombination der verschiedenen Regelwerke.

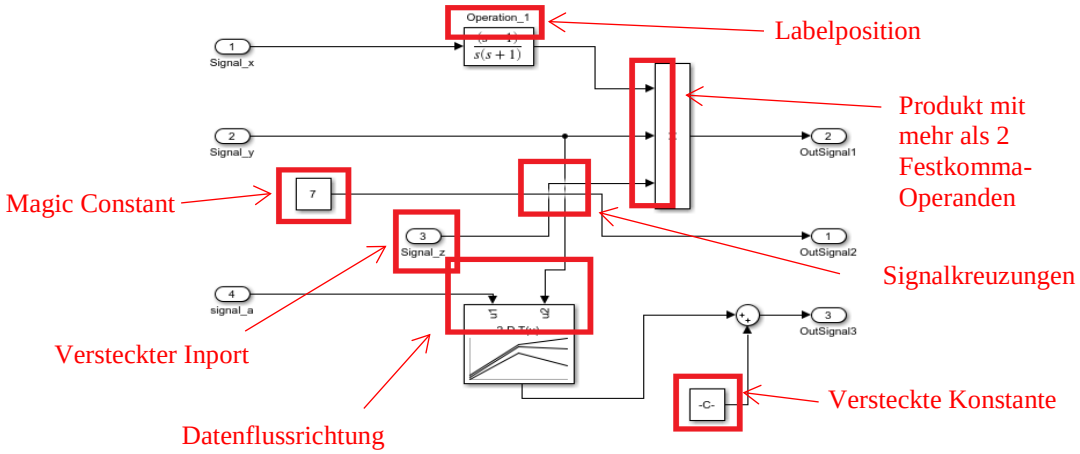

Abb. 1: Simulink-Modell vor Anwendung von Richtlinien

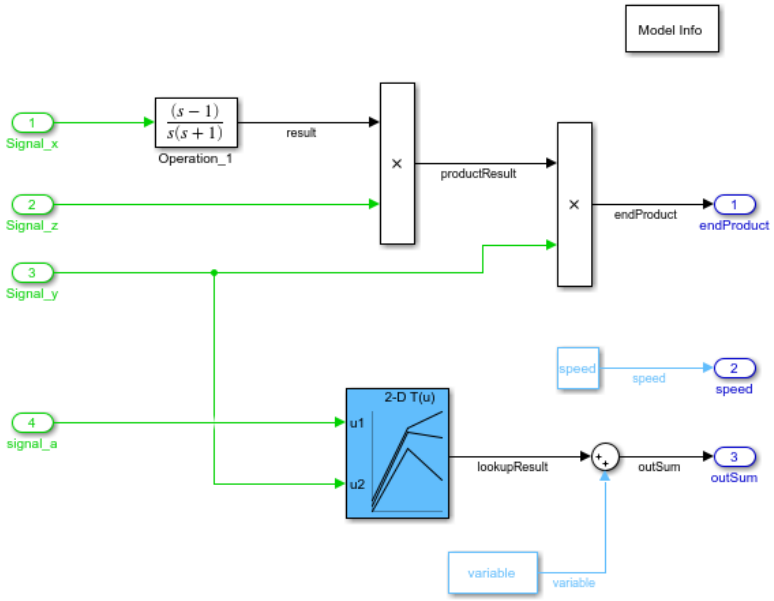

Abb. 2: Simulink-Modell nach Anwendung von Richtlinien

Ziele für den Einsatz von Modellierungsrichtlinien
1) Vermeidung nicht-robuster Modellierungstechniken:
Um nicht-robuste Modellierungstechniken zu umgehen, ist vor allem die Definition eines „Safe Subset“, d.h., die Festlegung einer Untermenge sicherer syntaktischer Elemente der jeweiligen Modellierungssprache notwendig.

2) Erhöhung von Effizienz und Sicherheit:
Wesentliche Maßnahmen, um dieses Ziel zu erreichen, sind die einheitliche Modell- und Werkzeugkonfiguration sowie einheitliche, gesicherte Einstellungen für Codegenerierung und -optimierung. Des Weiteren können bekannte ineffiziente oder funktional riskante Modellierungsmuster, beispielsweise die Verwendung von Gleitkommavariablen für Boolesche Signale, durch Modellierungsrichtlinien verboten werden.

3) Bessere Lesbarkeit, Wiederverwendung, Erweiterbarkeit, Wartbarkeit:
Diese Ziele können durch Vorgaben, die das graphische Layout, Namenskonventionen sowie spezifische Restriktionen für die Verwendung von Modellierungselementen betreffen, erreicht werden. Insbesondere wird dadurch die verteilte Entwicklung, beispielsweise zwischen OEM und Zulieferer, gefördert.
4) Einhaltung von Sicherheitsstandards:

Die Einhaltung von Sicherheitsstandards (z.B. IEC 61508, ISO 26262, ISO 25119, DO-178C) stellt ein übergeordnetes Ziel dar (s.u.).

Modellstruktur-Analyse und Komplexitätsreduzierung
Die Wart- und Testbarkeit von Modellkomponenten hängt wesentlich von einer sinnvoll gewählten Modellarchitektur ab. Die Herausforderung liegt hierbei darin, Funktionalitäten so zu kapseln, dass Module beherrschbarer Komplexität entstehen, die durch effektive Interfaces gekoppelt sind. Ein weiterer Aspekt, der bei der Modellstrukturanalyse eine Rolle spielt, ist die Vermeidung von *Model Clones* [8]. Solche semantisch äquivalenten Teilstrukturen reduzieren die Wartbarkeit des Modells deutlich und sollten deshalb vermieden werden. Um Modelle automatisiert hinsichtlich dieser Fragestellungen zu analysieren, kommen in der Praxis verschiedene Metriken zum Einsatz. Neben der Anzahl relevanter Sprachkonstrukte gehören die Metriken *local complexity* und *global complexity* sowie die für Modelle adaptierte *Halstead-Metrik* zu den wichtigsten Vertretern [9].

Vorgaben von Standards wie ISO 26262 und IEC 61508 zu Richtlinien und Designprinzipien
Als generische Norm für sicherheitsrelevante E/E/PE-Systeme definiert die IEC 61508 eine Reihe von Maßnahmen, die sowohl den Entwicklungsprozess als auch das Produkt betreffen, um die Sicherheit des Systems zu gewährleisten. Als kraftfahrzeugspezifische Auslegung fordert die ISO 26262 in Kapitel 6 („Product Development at the Software Level“) die Berücksichtigung einer Reihe von Aspekten (*Topics*), die je nach *Automotive Safety Integrity Level (ASIL)* der zu entwickelnden Funktion unterschiedliche Relevanz haben [1].

Daher muss beim Einsatz von Richtlinien für sicherheitsrelevante Softwarekomponenten beachtet werden, dass die von der ISO genannten *Topics* adressiert werden, in dem die abgeleiteten Regeln am Modell (und im Code)

umfassend geprüft werden, um, falls nötig, entsprechende Anpassungen vorzunehmen. Die speziell für Modellierungs- und Coderichtlinien relevanten Aspekte aus der ISO 26262 sind in Abbildung 3 zusammengefasst. Teil 6 der ISO 26262 enthält darüber hinaus auch konkrete Implementierungsvorgaben zum Design von Softwaremodulen. Diese dienen hauptsächlich dazu, die korrekte Ausführung von Programmteilen innerhalb von Modulen sicherzustellen sowie die Schnittstellenkonsistenz zwischen Modulen zu erreichen. Die für die modellbasierte Entwicklung relevanten *Topics* sind in Abbildung 4 zusammengefasst.

Topic	ASIL A	ASIL B	ASIL C	ASIL D
Enforcement of low complexity	++	++	++	++
Use of language subsets	++	++	++	++
Enforcement of strong typing	++	++	++	++
Use of defensive implementation techniques	o	+	++	++
Use of established design principles	+	+	+	++
Use of unambiguous graphical representation	+	++	++	++
Use of style guides	+	++	++	++
Use of naming conventions	++	++	++	++

Abb. 3: ISO 26262-6, § 5.4.7, Tabelle 1 (o keine Empfehlung / + empfohlen / ++ sehr empfohlen)

Des Weiteren enthält ISO 26262-6 abstrakte Prinzipien zum Entwurf der Softwarearchitektur, die darauf abzielen, Fehler, die durch hohe Komplexität entstehen, zu reduzieren. In vielen Fällen können auch hier konkrete Richtlinien für den modellbasierten Ansatz abgeleitet werden, die für eine Konformitätsprüfung bezüglich der Prinzipien der ISO herangezogen werden können [9].

Topic	ASIL A	ASIL B	ASIL C	ASIL D
Initialization of variables	++	++	++	++
No multiple use of variable names	+	++	++	++
Avoid global variables or else justify their usage	+	+	++	++
No implicit type conversion	+	++	++	++
No hidden data flow or control flow	+	++	++	++

Abb. 4: ISO 26262-6, § 8.4.4, Ausschnitt Tabelle 8 mit Aspekten, die für die modellbasierte Softwareentwicklung relevant sind (+ empfohlen / ++ sehr empfohlen)

Continuous Integration von statischer Modellanalyse
Für den praktischen Einsatz von Modellierungsrichtlinien sind statische Modellanalysen fortlaufend während des Entwicklungsprozesses automatisiert zu integrieren. Dies gilt insbesondere für Prüfroutinen zur Richtlinienkonformität. Als Grundlage der Automatisierungs-Infrastruktur werden in der Regel Jenkins-Server eingesetzt. Diese sind für die Ausführung verschiedener Aufträge von Modell-Ingenieuren, Modul-Testern und Software-Testern konfiguriert. Insbesondere können statische Modellanalysen, häufig im nahtlosen Zusammenspiel mit Versionsverwaltungssystemen, automatisiert ausgewertet werden und somit Qualitätsschranken ressourcenschonend und effizient überprüft werden.

Literaturverzeichnis

[1] International Organization for Standardization, ISO 26262: Road vehicles - Functional safety, 2011.

[2] MathWorks Automotive Advisory Board (MAAB), „Control Algorithm Modeling Guidelines Using MATLAB, Simulink, and Stateflow (Version 3.0)“, 2012.

[3] MIRA Limited, „MISRA AC SLSF: Modelling design and style guidelines for the application of Simulink and Stateflow“, 2009.

[4] MIRA Limited, „MISRA AC TL: Modelling style guidelines for the application of TargetLink in the context of automatic code generation“, 2007.

[5] dSpace GmbH, „Modeling Guidelines for dSpace TargetLink (Version 4.0.3)“, 2016.

[6] Model Engineering Solutions GmbH, „Functional Safety Modeling Guidelines“, 2015.

[7] Model Engineering Solutions GmBH, „MES Fit for Testing,“ 2018.

[8] E. Salecker und I. Stuermer, „JUST SIMPLIFY: Clone Detection for Simulink Controller Models“, *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.*, 2016.

[9] F. Bachmann und H. Dörr, „Analysis and Improvement of Model Architectures for Safety Related Systems“, *SAE Technical Paper*, 2018.

Autor

Dr. Simon Rösel ist seit 2017 Software Engineer für den MES Model Examiner (MXAM). Er hat an der Humboldt-Universität zu Berlin im Bereich Mathematische Optimierung promoviert. Im Zentrum seiner Tätigkeit bei MES stehen die Entwicklung von Checks zur automatisierten Richtlinienüberprüfung, z.B. im ISO 26262-Kontext, und die Unterstützung von Kunden- und Forschungsprojekten. Dabei gilt sein besonderes Interesse der Frage, wie Modelle effizient in Entwicklungsprozessen eingesetzt werden können.

Kontakt

Internet: https://model-engineers.com/de/
Email: simon.roesel@model-engineers.com