Safety in sich schnell ändernden Systemen
Praxistaugliche Anwendung von formalen Methoden

Christine Jakobs, Matthias Werner; TU Chemnitz

Der klassische Entwicklungsprozess für Software im Bereich eingebetteter Echtzeitsysteme geht von einem statischen System mit festgelegten Komponenten aus. Die Marktnachfrage führt zu einem Bedarf an stärkerer Dynamik. Dadurch erhöht sich die Komplexität und es wird wichtiger, den Erhalt von sicherheitskritischen Eigenschaften (safety) durch stringente Modellierung und Analyse abzusichern.

Eingebettete Systeme sind aus unserem täglichen Leben nicht mehr wegzudenken. Fahrerassistenzsysteme und intelligente Fabriksteuerungen sind Beispiele, in denen hochspezialisierte Computersysteme unser Leben durchdringen. Um diesen Anforderungen gerecht zu werden, müssen diese Systeme ihre Aufgaben meist unter Echtzeitanforderungen lösen. Dabei interagieren sie meist direkt mit der Umwelt und müssen dadurch hohe Ansprüche an ihre Sicherheit (safety) erfüllen.

Entwicklung von Software für eingebettete Systeme

Wie in vielen Industriebereichen wird auch bei eingebetteten Systemen die Software meist durch Domänenexperten programmiert. Diese müssen ihr Fachwissen übertragen und in die Programmierung mit meist hardwarenahen Programmiersprachen einbringen. Dabei muss die Software von eingebetteten Systemen hohen Anforderungen genügen. Im Vergleich zu Software für „general purpose computer“ sind die nichtfunktionalen Eigenschaften komplexer und der Prozess zu deren Nachweis wird durch zahlreiche Standards bestimmt.

Der klassische Entwicklungsprozess für Software in eingebetteten Systemen geht daher meist von einem statischen System aus. Die Softwarekomponenten und die Konfiguration des Gesamtsystems werden während des Entwicklungsprozesses festgelegt und hinsichtlich der funktionalen und nicht-funktionalen Eigenschaften analysiert. Dabei wird meist von der maximalen Ausstattung des Systems, also bei Fahrzeugen z. B. in der Komfortausführung, ausgegangen. Dieses Maximalsystem wird auf die Erfüllung der Sicherheitsanforderungen analysiert. Bei einem positiven Ergebnis wird davon ausgegangen, dass damit auch jedes andere System den Anforderungen gerecht wird. Problematisch ist dabei, dass diese statische Softwarearchitektur keine späteren Änderungen vorsieht. Damit muss im Aktualisierungsfall das System vollständig neu konfiguriert werden. Auf der anderen Seite führt die Analyse des Systems in Maximalausstattung auch zu einem anderen Problem: Kaum ein gekauftes System wird in dieser Ausstattung ausgeliefert. Demzufolge wird das System aufgrund einer permanenten Überschätzung aufgebaut.

Dynamische eingebettete Systeme

Neben den oben geschilderten Problemen im klassischen Entwicklungsprozess für eingebettete Systeme, findet derzeit ein Paradigmenwechsel statt, der durch den technischen Fortschritt und die mit ihm auftretenden Erwartungen und Anorderungen der Kunden ausgelöst wird. Z.B. wird in der Autoindustrie immer mehr Software eingesetzt um Fahrerassistenzsysteme anzubieten. Die Vernetzung zwischen Fahrzeugen und der Umwelt eröffnet einen neuen Angriffsvektor auf diese sicherheitskri-

tischen Systeme. Damit werden Softwareaktualisierungen zur Fehlerbehebung die Regel. Es reicht nicht mehr aus, das Fahrzeug bei Werkstattbesuchen zu aktualisieren. Vielmehr wird es notwendig, die Fahrzeuge jederzeit „over-the-air“ aktualisieren zu können. Des Weiteren gibt es zunehmend Bestrebungen, in eingebetteten Systemen nicht mehr speziell angefertigte Hardware zu nutzen, sondern auf „commercial-of-the-shelf“-Hardware zu wechseln. Damit wird die Möglichkeit eröffnet, Software per Hochintegration auf der Hardware zu platzieren. Diese Flexibilität in der Konfiguration durch Aktualisierungen und Hochintegration wird jedoch bezahlt mit erhöhter Komplexität, was u.a. den Nachweis der nicht-funktionalen Eigenschaften erschwert. Da durch diese Komplexität ein Nachweis durch einfaches Testen zunehmend schwieriger wird, muss auf Modellierung und darauf basierende Analyse dieser dynamisch komponierten Systeme zurückgegriffen werden.

Modellierung und Analyse von eingebetteten Systemen
Ausgehend von den Industriestandards, beispielsweise der ISO 26262 für den Automobilbereich, können die Modellierungs- und Analysemethoden für eingebettete Systeme in drei Gruppen eingeteilt werden:

- Informale Methoden: Syntax und Semantik sind nicht vollständig definiert, z. B. Diagramme, Bilder.
- Semi-formale Methoden: Die Syntax ist vollständig definiert, jedoch ist die Semantik unvollständig definiert, z. B. UML.
- Formale Methoden: Syntax und Semantik sind vollständig definiert, z. B. Z-Notation, TLA+.

Die Autoindustrie legt vier verschiedene Level (ASIL A-D) fest, welche die Kritikalität der Komponente beschreiben. Im Falle von Hochintegration von Software bestimmt die Softwarekomponente mit der höchsten Stufe die Methoden des Nachweises für alle Komponenten, falls nicht eine Rückwirkungsfreiheit des Systems bewiesen werden kann.

Die Methoden, welche zum Nachweis der nicht-funktionalen Eigenschaften angewendet werden können, sind vielfältig und abgesehen von informalen Methoden können diese für alle Sicherheitsstufen verwendet werden. Informale und semi-formale Modellierungsmethoden werden als einfacher und damit günstiger in der Anwendung erachtet. Auf den ersten Blick ist dies auch der Fall, da sie eine kurze Einarbeitungszeit brauchen. Die Praxis zeigt, dass meist auf mehrere Modellierungs- und Analysemethoden zurückgegriffen wird, um die verschiedenen Stufen des Entwicklungsprozesses zu unterstützen. Dabei ist viel Aufwand zur Übersetzung der verschiedenen Modelle nötig. Dennoch führt dies meist zu unübersichtlichen Modellen, welche sich teilweise überschneiden und damit zum Verlust des Überblicks über den Gesamtprozess führen. Die geforderte Dynamik erhöht die Komplexität zusätzlich, was die derzeit eingesetzten Modellierungsmethoden nicht oder nicht im notwendigen Maße unterstützen.

Temporallogik als toolübergreifende Metasprache
Während der klassische Anwendungsfall formaler Verfahren die Spezifikation und anschließende Verifikation eines Systems bzw. einzelner Komponenten ist, schlagen wir einen ganzheitlichen Ansatz vor, in dem klassische Verifikationsverfahren ge-

zielt für die modellbasierte Entwicklung eingesetzt werden. Konkret nutzen wir TLA, eine Temporallogik.

Temporallogiken gehören zu den Modallogiken. Aussagen in diesen Logiken gelten nur unter bestimmten Bedingungen/in bestimmten Modi. Innerhalb dieser Modi gilt die bekannte Aussagenlogik, während global das Extensionalitätsprinzip nicht gilt. Temporallogiken treffen Aussagen über die zeitliche Ordnung von Aussagen, ohne ein bestimmtes zu Grunde liegendes Zeitmodell. Vielmehr werden Aussagen wie „es gilt immer, dass...“, „irgendwann gilt, dass...“ oder „es gilt ... bis ... gilt“ getroffen. Dies erleichtert die Analyse von Systemen mit verschiedenen Zeitmodellen (z. B. verteilte Systeme).

TLA, bzw. die Spezifikationssprache TLA+ wird bereits erfolgreich industriell eingesetzt. Lamport entwickelte TLA, um sowohl modulare Entwicklungsprozesse, als auch die Komposition von verschiedenen Zeitmodellen zu unterstützen. Hierfür erlaubt TLA dem Modell zu „stottern“. Das bedeutet, dass es Schritte gibt, in denen sich die betrachteten Systemvariablen nicht verändern. Dadurch kann bei der Komposition von Modellen eine Verknüpfung der Ablaufbeschreibung ausbleiben. Die Modelle können vielmehr konkurrierend analysiert werden, da während das eine Modell stottert, ein anderes die Möglichkeit hat Fortschritt zu generieren.

Die Spezifikation von Systemkomponenten mit TLA+ ermöglicht ihre Analyse hinsichtlich ihrer funktionalen und nicht-funktionalen Eigenschaften. Hierfür bietet TLA+ die Möglichkeit, Kriterien wie z. B. Lebendigkeit und Erreichbarkeit direkt zu verifizieren.

Werden zusätzlich auch die Spielregeln der Systemkomposition mittels TLA+ spezifiziert, so erlaubt eine anschließende Analyse die Generierung einer Softwarearchitektur, welche diesen Regeln und damit den geforderten nicht-funktionalen Eigenschaften der Architektur entspricht. So können beispielsweise aus den echtzeitkritischen Softwarekomponenten mittels Spezifikation der Schedulingkriterien die entsprechenden Konfigurationen für das Betriebssystem generiert werden.

Dieser ganzheitliche Ansatz ermöglicht eine Anwendung während des gesamten Entwicklungsprozesses. Dies begründet sich in der gezielten Unterstützung von Modularisierung und Komposition in der Spezifikationssprache. Dadurch kann flexibel zwischen den Ebenen der modellbasierten Entwicklung gewechselt werden.

Anwendungsbeispiele von Google und Amazon zeigen die Eignung von TLA+ als Spezifikationssprache sowohl für kleine Softwarekomponenten als auch für große verteilte Systeme. Eigene Beispielfälle zeigen, dass dieses „Auf den Kopf stellen“ der Spezifikationssprache zur Generierung von Architekturen mit einfachen Mitteln die gewünschten Ergebnisse bringt. Unter Anwendung von PlusCal als Algorithmensprache für TLA+ können mit Hilfe einer C-ähnlichen Syntax die verwendeten Kompositionsoperatoren einfach spezifiziert, direkt in TLA+ übersetzt und analysiert werden.

Zusammenfassung und Ausblick

Die Welt der sicherheitskritischen eingebetteten Systeme befindet sich derzeit im Wandel. Annahmen eines statischen Systems gehören durch die Forderungen nach

einfachen Möglichkeiten zur Aktualisierung von Software der Vergangenheit an. Dadurch erhöht sich die Komplexität in der Erstellung von Software für eingebettete Systeme sowie der Nachweis der funktionalen und nicht-funktionalen Eigenschaften.

Temporallogiken, speziell TLA+ ermöglichen einen ganzheitlichen Ansatz zur Spezifikation und Analyse solcher Systeme während des gesamten Entwicklungsprozesses. Dabei können sowohl einzelne Softwarekomponenten als auch die Kompositionsoperatoren modelliert werden. Dadurch wird die Softwarearchitektur generativ gebildet und garantiert dadurch inhärent die geforderten nicht-funktionalen Eigenschaften.

Autoren

Christine Jakobs studierte BWL und Informatik und forscht seit ihrer Masterarbeit im Bereich von cyber-physischen Systemen. Dabei ist sie insbesondere an Eigenschaften wie Echtzeitfähigkeit und Verlässlichkeit interessiert. Im Rahmen ihrer Promotion verfolgt sie das Ziel mit Hilfe formaler Methoden künftige sicherheitskritische Systeme, insbesondere aus dem Transportbereich, auch unter den Bedingungen der heute durch den Markt geforderten Flexibilität beherrschbar zu halten.

Prof. Matthias Werner hat an der HU Berlin E-Technik und Regelungstechnik studiert und dort auch promoviert. Nach Forschungsaufenthalten in Austin und Cambridge (UK) habilitierte er an der TU Berlin zu Eigenschaften verlässlicher Systeme. Er hat die Professur für Betriebssysteme an der TU Chemnitz inne. Zu seinen Forschungsinteressen zählen Entwurf und Analyse von Laufzeitsystemen unter Berücksichtigung von Eigenschaften wie Echtzeitfähigkeit, Mobilität oder Verlässlichkeit.

Kontakt
Internet: osg.informatik.tu-chemnitz.de
Email: [bartch|werner]@informatik.tu-chemnitz.de