Typisierung und Verifikation zeitlicher Anforderungen automotiver Software Systeme

Matthias Gehrke, Martin Hirsch,
Wilhelm Schäfer
Software Quality Lab (s-lab)
Universität Paderborn
D-33095 Paderborn
[mgehrke|mahirsch|wilhelm]@upb.de

Oliver Niggemann, Dirk Stichling

dSPACE GmbH
Technologiepark 25
D-33100 Paderborn
[ONiggemann| DStichling]@dspace.de

Ulrich Nickel
Hella KGaA
Rixbecker Str. 75
D-59552 Lippstadt
Ulrich.Nickel@hella.com

Zusammenfassung: Ein wesentliches Problem der Integration von Steuergeräten im Automobil besteht darin, dass die einzelnen Steuergeräte in ihrer Funktionalität zwar spezifiziert werden, dass aber mögliche Inkompatibilität und daraus resultierende Fehler insbesondere im Hinblick auf zeitliche Anforderungen an das Gesamtsystem erst während des Integrationstest erkannt werden (können). Wir schlagen ein modellbasiertes Vorgehen vor, das basierend auf einer AUTOSAR-kompatiblen Komponentenarchitektur die Spezifikation der Funktionalität und der zeitlichen Randbedingungen unterstützt und darüber hinaus die komponentenübergreifende automatische Überprüfung von zeitlichen Anforderungen an das Gesamtsystem auf der Basis der Modelle der einzelnen Steuergeräte ermöglicht.

1 Einleitung

Die in Kraftfahrzeugen eingesetzten Steuergeräte werden insbesondere durch den verstärkten Einsatz von Software immer leistungsfähiger. Darüber hinaus wird durch einen massiven Ausbau ihrer Vernetzung untereinander wesentlich weitergehende Funktionalität und damit zusätzlicher Komfort und mehr Sicherheit möglich und realisiert. Ein Beispiel für diesen Trend sind moderne Fahrerassistenzsysteme. Für einen Automobil-Zulieferer wie die Hella KGaA Hueck & Co führt dieser bekannte Trend zu immer größeren Projekten und vor allem immer umfangreicher werdenden Komponentenspezifikationen, da immer komplexere Funktionen in die einzelnen Steuergeräte zu integrieren sind. Bei Herstellern wird der Aufwand für die Integration der einzelnen Steuergeräte dementsprechend immer aufwendiger und auch schwieriger.

Ein wesentliches Problem dieser Integration besteht darin, dass die einzelnen Steuergeräte in ihrer Funktionalität zwar spezifiziert werden, dass aber oft eine genaue Spezifikation der zeitlichen Abläufe und Restriktionen fehlt oder nur in Form von natürlich sprachlichem Text ausgedrückt wird. Selbst wenn diese Spezifikation in formaler Notation vorliegt, sind komponentenübergreifenden Prüfungen auf Widerspruchsfreiheit oder Vollständigkeit aufgrund der Komplexität solcher Systeme oftmals nur schwer möglich. Bei der Integration der Steuergeräte, d.h. beim ersten Test stellt sich dann erst heraus, ob die in den einzelnen Steuergeräten realisierten Funktionen und ihre zeitlichen Randbedingungen miteinander kompatibel sind und nicht sogar zu Fehlfunktionen führen (können).

Wir schlagen ein modellbasiertes Vorgehen vor, dass basierend auf einer AUTOSAR-kompatiblen Komponentenarchitektur die Spezifikation der Funktionalität und der zeitlichen Randbedingungen unterstützt und darüber hinaus die komponentenübergreifende automatische Überprüfung von zeitlichen Anforderungen an das Gesamtsystem auf der Basis der Modelle der einzelnen Steuergeräte und somit weit vor einer Testphase ermöglicht.

Im folgenden Kapitel wird hierzu das Konzept des so genannten „Master“-Automaten vorgestellt. Dieser stellt die für eine Überprüfung einer Eigenschaft des Gesamtsystems minimal notwendige Verschaltung der einzelnen Automaten, die die Funktionalität eines einzelnen Steuergerätes bzw. einer einzelnen Komponente spezifizieren, dar. In Kapitel 3 wird gezeigt, wie dieser Automat auf der Basis einer einfachen parametergesteuerten Eingabe der vom Gesamtsystem gewünschten Eigenschaft automatisch erzeugt werden kann und wie diese gewünschte komponentenübergreifende Eigenschaft dann automatisch durch Model Checking überprüft wird. Zusammenfassung und Ausblick in Kapitel vier schließen das Papier ab.

2 Komponentenbasierter Entwurf

Die Architekturmodellierung eines automotiven Softwaresystems in [GH+06] basiert auf einem Komponentenansatz gemäß der AUTOSAR-Spezifikation [AUT]. Die damit modellierten Software-Komponenten kapseln Teilfunktionalitäten oder sogar die vollständige Funktionalität eines Steuergeräts oder Steuergerätenetzwerkes. Um mit anderen Komponenten zu kommunizieren, besitzt jede Komponente Ports. Die Ports verschiedener Komponenten können miteinander verbunden werden, wenn die an den Ports angebotenen Schnittstellen kompatibel sind. Kompatibel in diesem Zusammenhang bedeutet, dass die zur Verfügung gestellten Daten diejenigen sind, die ein verbundenes Interface benötigt. In Abbildung 1 ist ein Beispiel der Software-Architektur einer Diebstahlwarnanlage gegeben.

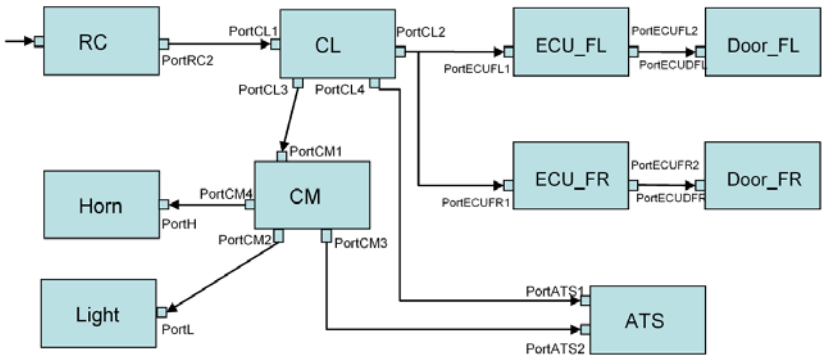

Abbildung 1: Software-Architektur einer Diebstahlwarnanlage

Das Verhalten einer einzelnen Komponente wird in [GH+06] mit Hilfe eines „Automotive Specific Automata“ spezifiziert (siehe Abbildung 2). Dies sind einfache Automaten die um Zeitannotationen erweitert wurden, wobei die Semantik über eine Abbildung auf Timed Automata [AD94] definiert ist. In Abbildung 2 ist ein „Automotive Specific Automata“ modelliert, der die drei Zustände *running*, *shutting_down* und *not_running* besitzt. Die Verweildauer in den Zuständen kann durch annotierte Zeitintervalle spezifiziert werden und die Aktivierung von Transitionen wird durch Nachrichten bzw. Guards modelliert.

Die formale Abbildung auf Timed Automata wird bei der Verifikation ausgenutzt, die mittels des Modelcheckers UPPAAL[1] durchgeführt wird. UPPAAL ist ein expliziter Modelchecker der als Eingabemodelle Timed Automata verwendet. Er ist frei verfügbar und wird im s-lab bereits seit einiger Zeit erfolgreich eingesetzt.

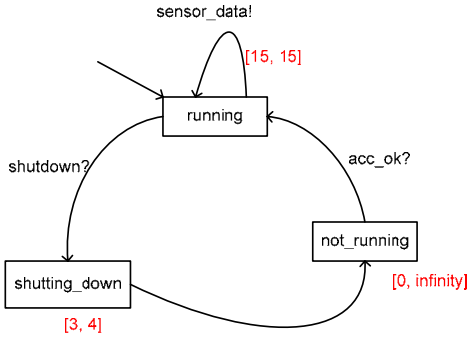

Abbildung 2:Beispiel eines Automotive Specific Automata

[1] http://www.uppaal.com

Für die Verifikation der Eigenschaft „Wie lange braucht das System, bis die Nachricht speed_ok gesendet wird“ muss diese Anfrage nun entsprechend formalisiert werden, damit die Verifizierung mit Hilfe des Modelcheckers UPPAAL möglich ist. Die Mächtigkeit der in UPPAAL verwendeten TCTL reicht hierfür jedoch nicht aus. Der Ansatz, das Verhalten aller Automaten global in einem Automaten zusammenzufassen und nur eine globale Uhr zu verwenden, ist nicht sinnvoll, da der dann gebildete Produktautomat ein Verhalten beschreibt, welches durch entstehende Seiteneffekte möglicherweise nicht mehr mit dem spezifizierten Verhalten einzelner Automaten übereinstimmt.

Um dieses Problem zu lösen wurde das Prinzip der szenariobasierten Verifikation angewandt [GH04, KM+02]. Hierbei wird für eine zu verifizierende Eigenschaft ein so genannter „Master“-Automat formuliert, welcher parallel zu den eigentlichen Automaten ausgeführt wird. Ein „Master“-Automat observiert das Verhalten der anderen Automaten (Timed Automata), wodurch dieser auf Einhaltung bestimmter Eigenschaften (durch temporallogische Formeln spezifiziert) überprüft werden kann. Da der „Master“-Automat genau das minimal benötige Verhalten aller vernetzten Komponenten für die zu überprüfende Eigenschaft in einem Automaten zusammenfasst, ist hierdurch eine korrekte, effiziente Verifikation möglich.

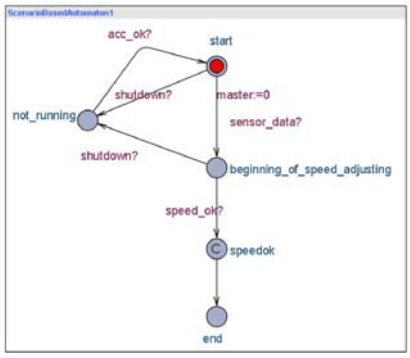

Abbildung 3: Beispiel für einen „Master-Automaten“

In Abbildung 3 ist ein Beispiel für einen „Master“-Automaten gegeben. Dieser beinhaltet z.B. den Zustand *not_running*, welcher dem Zustand *not_running* aus Abbildung 2 entspricht. Die anderen Zustände stammen aus anderen Timed Automata bzw. wurden explizit hinzugefügt.

3. Verifikation zeitlicher Anforderungen

Die manuelle Konstruktion sowohl der „Master“-Automaten, als auch der temporallogischen Formeln, ist sehr aufwändig und fehleranfällig. In beiden Fällen wäre es daher hilfreich einen entsprechenden Formalismus zu haben, der sowohl den „Master“-Automaten, als auch die temporallogischen Formeln automatisch generiert.

Die grundlegende Idee eine solche Generierung umzusetzen besteht darin, die verschiedenen Zeit-Anforderungen zu typisieren. Für jeden Typ ist es dann möglich, entsprechende Generierungsvorschriften zu definieren. In diesem Kapitel wird eine Typisierung der Zeit-Anforderungen für den Bereich der Karosserieelektronik im Bereich der Automobilindustrie vorgenommen. Als Grundlage für die Analyse der Zeit-Anforderungen werden Informationen verwendet, die aus den dort erstellten Pflichtenheften hervorgehen. Diese Zeit-Anforderungen kann man, abstrahiert betrachtet, auf die Zeitdauer zwischen Ereignissen beziehen. Es geht also immer um Ereignisse, ihr Auftreten, und die Zeitspanne zwischen ihrem Auftreten.

3.1 Typisierung der Zeit-Anforderungen

Die Zeit-Anforderungen für den Bereich Karosserieelektronik lassen sich in vier verschiedene Typen einteilen, wobei diese hierarchisch organisiert sind. Die vier Typen sind: „Maximale Verarbeitungszeit“, „Exakte Ausführungszeit“, „Synchronizität“ und „Konditionale Anforderungen“.

Maximale Verarbeitungszeit

Die maximale Verarbeitungszeit beschreibt, wie viel Zeit maximal zwischen zwei geordneten Ereignissen vergehen darf. Das folgende Beispiel verdeutlicht dies:

„Die Verarbeitungszeit zwischen Erkennen einer Bremslichtanforderung durch ein Lichtsteuergerät und dem Ansteuern der Ausgänge für die Bremsleuchten darf höchstens 50 ms betragen.“

Die maximale Verarbeitungszeit ist wie folgt definiert:

Eingabe: Start-Ereignis E_1
End-Ereignis E_2 wobei E_1 zeitlich vor E_2 eintritt
Zeit t_{max}

Bedingung: Die Zeit t zwischen dem Auftreten von E_1 und E_2 darf nicht größer sein als ein vorgegebener Wert t_{max}, also $t <= t_{max}$.

Exakte Ausführungszeit

Die exakte Ausführungszeit beschreibt, dass ein bestimmtes Ereignis genau nach einer festgelegten Zeit eintreten muss. Zusätzlich kann eine zeitliche Toleranz angegeben werden. Das folgende Beispiel verdeutlicht dies:

„Die Zentralverriegelung muss das Heckschloss bei Anforderung für eine Zeit von 600 ms ansteuern.“

Die Exakte Ausführungszeit ist wie folgt definiert:

Eingabe: Start-Ereignis E_1

End-Ereignis E_2 wobei E_1 zeitlich vor E_2 eintritt
Zeit t_{exakt}
Toleranz $t_{tol} >= 0$

Bedingung: Die Zeit t zwischen dem Eintreten von E_1 und dem Eintreten von E_2 muss sich im Intervall $[t_{min}, t_{max}]$ befinden, wobei $t_{min}=t_{exakt}-t_{tol}$ und $t_{max}=t_{exakt}+t_{tol}$ ist.

Sychronizität

Die Synchronizität beschreibt, dass mehrere Ereignisse ungeordnet innerhalb eines festgelegten Zeitintervalls eintreten müssen. Das folgende Beispiel verdeutlicht dies:

„Alle Leuchten (einschließlich Kontrolllampen), die gemeinsam in Betrieb sind, müssen gleichzeitig ein- bzw. ausgeschaltet werden. Es darf zu keinem vom Betrachter wahrnehmbaren Zeitversatz zwischen den einzelnen Lampen kommen."

Die Synchronizität ist wie folgt definiert:

Eingabe: Menge von Ereignisse $\{E_1, ..., E_n\}$
End-Ereignis E_2 wobei E_1 zeitlich vor E_2 eintritt
Zeit $t_{synchron}$

Bedingung: für alle i,j aus $\{1, ..., n\}$ gilt: $|t_i-t_j| <= t_{synchron}$ mit t_i ist Zeitpunkt des Eintretens von E_i

Konditionale Anforderungen

Konditionale Anforderungen beschreiben laufzeitabhängige Szenarien. Hier kommt es darauf an, was während der Laufzeit passiert, wann welches Ereignis eintritt. Das folgende Beispiel verdeutlicht dies:

„Werden von dem gleichen Schließzylinder innerhalb einer parametrierbaren Zeit zwei Entriegelungsbefehle abgegeben, wird das Fahrzeug komplett entriegelt."

Die konditionale Anforderung ist wie folgt definiert:

Eingabe: Start-Ereignis E_1
Bedingungs-Ereignis E_2 wobei E_1 zeitlich vor E_2 eintritt
Folge-Ereignis E_3
Zeit t_{cond}

Bedingung: Falls $t_2-t_1<=t_{cond}$ dann muss E_3 nach Ablauf der Zeit t_{cond} eingetreten sein; t_i ist der Zeitpunkt des Eintretens von E_i

Diese vier hier vorgestellten Typen sind ausreichend, um den Großteil der Zeit-Anforderungen automotiver Softwaresysteme zu beschreiben.

3.3 Generierung des „Master“-Automaten und der temporallogischen Formeln

Für jeden Anforderungstyp existiert eine Vorlage, die beschreibt, wie der jeweilige „Master“-Automat erstellt wird. Der Anwender muss lediglich den Typ der Anforderung und die Eingabedaten, wie zum Beispiel die Start- und Endereignisse, spezifizieren.

Im Folgenden ist beispielhaft für den Typ „Maximale Verarbeitungszeit“ und der Eigenschaft, dass 5ms nachdem die Fernbedienung betätigt wurde die Zentralverriegelung eingeschaltet ist, dargestellt, wie der „Master“-Automat generiert wird.Das Startereignis E_1 sei das Erkennen des Betätigens der Fernbedienung (`Remote_Control_portRC2_lock`), das Endereignis E_2 sei der Befehl zum Schärfen der Diebstahlwarnanlage (`Central_Locking_portCL4_ATSSharpening`) und die Zeit t_{max} sei 5 ms. In den Zuständen *Start* und *Waiting* wird nacheinander auf die entsprechenden Nachrichten gewartet. Der „Master“-Automat ist auf der rechten Seite der Abbildung 3 dargestellt.

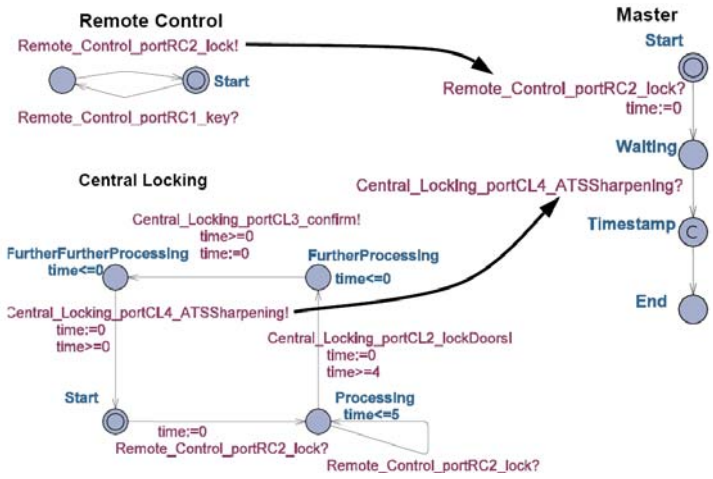

Abbildung 3: Generierung eines "Master"-Automaten

Nachdem der „Master“-Automat generiert wurde, muss nun noch die temporallogische Formel (in diesem Fall TCTL) generiert werden. Hierfür benötigt man lediglich die entsprechende Zeitangabe. Im obigen Beispiel sind dies t_{max} = 5 ms. Die folgende TCTL-Formel würde für das Beispiel wie folgt aussehen:

```
E<>Master.End
Master.Timestamp ~> Master.time <= 5
```

Der verwendete Modelchecker UPPAAL kann nun prüfen, ob der letzte Zustand immer erreicht wird und ob die Zeit zwischen den Ereignissen immer maximal 5 ms beträgt.

Äquivalent zur „Maximalen Verarbeitungszeit“ lassen sich auch für die anderen Anforderungstypen entsprechende Vorlagen zur automatischen Generierung des Master-Automaten angeben.

4. Zusammenfassung und Ausblick

Die Verifikation automotiver Software-Architekturen bereits in frühen Phasen der Software-Entwicklung wird in Zukunft immer mehr an Bedeutung gewinnen. Die Verifikation von Echtzeitaspekten spielt dabei eine besondere Rolle, da zeitliches Fehlverhalten von zusammenhängenden Funktionen ansonsten erst nach der Integration der einzelnen Funktionsimplementierungen erkannt werden kann. Dieses Fehlverhalten kann unter Umständen zu einem Redesign der gesamten Software führen und hat somit entschiedenen Einfluss auf Projektkosten und –laufzeiten.

In diesem Artikel wurde gezeigt, wie die Verifikation von Zeit-Anforderungen eines auf Komponenten basierenden Softwaresystems erfolgen kann. Dazu wird das zeitliche Verhalten der einzelnen Komponenten unabhängig voneinander mittels „Timed Automata“ bzw. einer Spezialisierung namens „Automotive Specific Automata“ beschrieben. Aus den einzelnen Timed Automata der Komponenten wird dann ein „Master“-Automat generiert. Dieser „Master“-Automat wird verwendet, um die Zeit-Anforderungen mittels temporallogischer Formeln komponentenübergreifend zu verifizieren. Sowohl der „Master“-Automat als auch die temporallogische Formel werden dabei automatisch generiert.

Bisher nicht berücksichtigt wurde die Verteilung der Software-Komponenten auf unterschiedliche Steuergeräte und die damit zusammenhängenden Laufzeiten der Netzwerkkommunikation oder aber Schedulingeffekte des Betriebssystems. Die Verifikation kann daher schon in frühen Phasen des Entwicklungsprozesses, zum Beispiel beim Fahrzeughersteller, eingesetzt werden. Sollen in späteren Phasen auch die o.a. Effekte nachgebildet werden, müssen entsprechend komplexere Automaten oder zusätzliche Automaten generiert werden, die z.B. die Laufzeiten von Nachrichten auf den Bussen berücksichtigen. Die Spezifikation des zeitlichen Verhaltens der einzelnen Komponenten selbst bleibt davon aber weiterhin unabhängig.

Eine weitere mögliche Erweiterung des Ansatzes besteht darin, die zu prüfenden Zeit-Anforderungen nicht in den temporallogischen Formeln (TCTL) zu codieren, sondern diese direkt in den „Master–Automaten“ als Invarianten zu annotieren. Der Vorteil hierbei wäre, dass die TCTL-Formeln deutlich kleiner werden und damit die Hoffnung besteht, das Modelchecking effizienter zu gestalten.

Acknowledgement

Für ihre Mitarbeit möchten wir Frau Petra Nawratil und Frau Renate Ristov herzlich danken. Sie haben mit ihren Diplom- bzw. Studienarbeiten entscheidende Beiträge für dieses Papier geliefert. Darüber hinaus bedanken wir uns bei den Mitarbeitern der Firma dSPACE GmbH und der Hella KGaA für Ihre tatkräftige Unterstützung.

Literaturverzeichnis

[AUT] http://www.autosar.org

[AD94] R. Alur and D. Dill. A theory of timed automata. In Theoretical Computer Science, 126(2): 183-235, 1994.

[GH04] S. Graf and J. Hooman. Correct Development of Embedded Systems. In Flavio Oquendo, Brian Warboys, and Ron Morrision, editors, *Proc. of the First European Workshop on Software Architecture, EWSA2004*, volume 3047 of *Lecture Notes in Computer Science*, pages 241-249, St Andrews, UK, May 21-22 2004, Springer Verlag.

[GH+06] M. Gehrke, M. Hirsch, P. Nawratil, O. Niggemann, and W. Schäfer: Scenario-Based Verification of Automotive Software Systems: In *Proc. 2nd Workshop on Modellbasierte Entwicklung eingebetteter Systeme (MBEES), Schloss Dagstuhl, Germany* (Holger Giese, Bernhard Rumpe, and Bernhard Schätz, eds.), volume Informatik-Bericht 2006-01, Technische Universität Braunschweig, pages 35-42, January 2006.

[KM+02] A. Knapp, S. Merz, and C. Rauh. Model Checking timed UML State Machines and Collaborations. 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT 2002), Oldenburg, September 2002, Lecture Notes in Computer Science, volume 2469, pages 395-414, Springer-Verlag, 2002.