Architekturvarianten sicherheitskritischer Echtzeitsysteme

Sichere und kostengünstige Lösungen systematisch ermitteln

Dr. Ulrich Becker, Method Park
Dr. Isabella Stilkerich, Schaeffler Technologies
Dr. Ralf Münzenberger, INCHRON

Immer mehr eingebettete Systeme sind sicherheitsrelevant, oft gekennzeichnet durch eine Kombination aus hohen Verfügbarkeitsanforderungen und harten Echtzeitanforderungen. Typisches Beispiel für diese Systemkategorie sind Sicherheits- und Assistenzfunktionen im Automobilbereich. Werden die Echtzeitanforderungen erst spät betrachtet, resultieren oft teure Änderungen an der System-, SW- und HW-Architektur. Dieser Artikel zeigt ein durchgängiges Vorgehen von der Sicherheitsanalyse über die funktionale Architektur bis hin zur technischen Architektur auf. Schon in der funktionalen Architektur werden sicherheitsrelevante Wirkketten identifiziert, End-to-End Echtzeitanforderungen zugewiesen und Zeitbudgets abgeleitet. Aus der funktionalen Architektur können verschiedene Varianten technischer Architekturen abgeleitet werden. Die Architekturvarianten können hinsichtlich ihrer Echtzeiteigenschaften simuliert und systematisch bewertet werden, so dass die bezüglich Sicherheit und Kosten optimale Variante ausgewählt werden kann.

Als Fallbeispiel dient ein fiktives Fußgänger-Airbag-System (FABSY), das Fußgänger vor körperlichen Schäden bei der Kollision mit einem Auto schützen soll. Das System erkennt, wenn eine Kollision des Fahrzeugs mit einem Fußgänger unausweichlich ist, und löst in diesem Fall einen in der Fahrzeugfront angebrachten Fußgänger-Airbag aus.

In der Betrachtung der funktionalen Sicherheit muss neben der *er*wünschten Auslösung des Airbags auch berücksichtigt werden, dass durch eine *uner*wünschte Auslösung die Fahrzeuginsassen und weitere Verkehrsteilnehmer gefährdet werden können. Aus der Analyse der FABSY-Funktion ergeben sich harte Echtzeitanforderungen, z.B. für den Zeitpunkt der Airbag-Auslösung.

Gefahren- und Risikoanalyse

In der Gefahren- und Risikoanalyse werden die Sicherheitsziele definiert, die während der gesamten weiteren Entwicklung zu beachten sind. Die Sicherheitsziele werden mit einem Indikator für die Kritikalität versehen. Dieser reicht von **A**utomotive **S**afety **I**ntegrity **L**evel (ASIL) A (wenig kritisch) bis ASIL D (sehr kritisch). Für FABSY heißt das konkret: "Verhindere ungewolltes Auslösen des Airbags" (Sicherheitsziel 1/ASIL D), und "Wenn eine Kollision mit einem Fußgänger unvermeidbar ist, garantiere das Auslösen des Airbags" (Sicherheitsziel 2/ASIL B).

Funktionale Architektur

In der funktionalen Architektur wird FABSY aus rein funktionaler Sicht in Funktionsblöcke zerlegt und deren Zusammenwirken beschrieben. Funktionsblöcke

sind beispielsweise die Akquisition eines Bildes oder das Erkennen eines Fußgängers im Bild.

Die funktionale Architektur abstrahiert von der konkreten technischen Realisierung. Es wird offengelassen, auf welchem Steuergerät und mit welcher Technologie ein Funktionsblock realisiert wird. So bleiben verschiedene Varianten für die technische Systemarchitektur möglich und können später systematisch verglichen werden.

Die Funktionsblöcke erben ihren ASIL von den Sicherheitszielen, an deren Umsetzung sie beteiligt sind – so erhält die Personenerkennung den ASIL D. Eine Funktion wie die Personenerkennung lässt sich über Kamera oder Radar aber nicht mit der für ASIL D erforderlichen Zuverlässigkeit gewährleisten. Schon in der funktionalen Architektur kann darauf mit der Architekturmaßnahme "ASIL Dekomposition" reagiert werden: Die Kette von der Akquisition des Bildes bis zur Entscheidung den Airbag auszulösen wird redundant mit unterschiedlichen Technologien realisiert. Die Auslösung des Airbags erfolgt nur, wenn radarbasierte *und* kamerabasierte Erkennung zur gleichen Entscheidung kommen. Wenn sich die radar- und kamerabasierten Teilsysteme nicht ungewollt gegenseitig beeinflussen ("freedom from interference“), lässt sich der ASIL dieser Funktionsblöcke von ASIL D auf ASIL B (D) reduzieren.

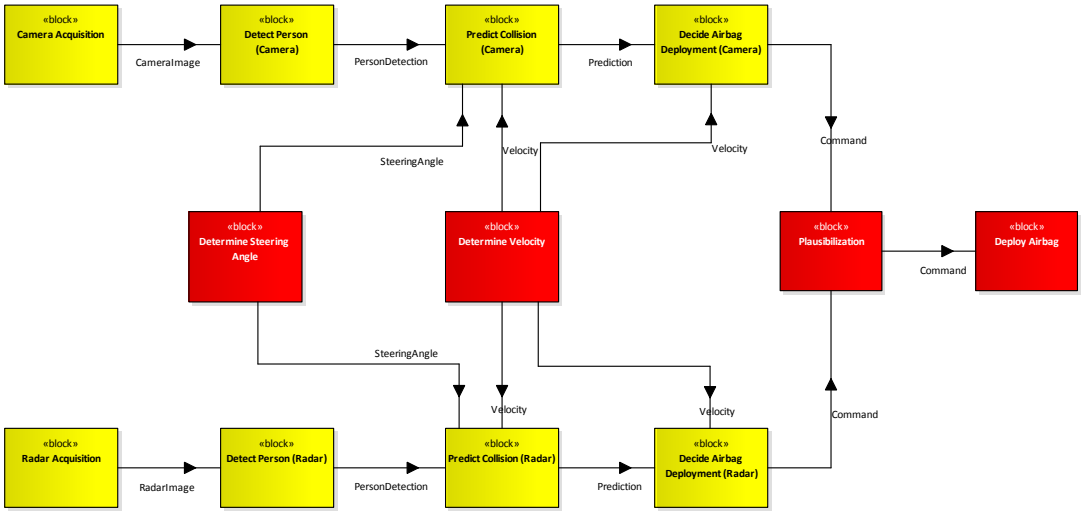

Bild 1: Funktionale Systemarchitektur von FABSY nach der ASIL Dekomposition. Den Funktionsblöcken ist ein ASIL B (gelb) bzw. ASIL D (rot) zugewiesen.

Technische Architektur

Die technische Architektur beschreibt die konkrete technische Realisierung und ist Grundlage für das technische Sicherheitskonzept. Sie beschreibt, welche Steuergeräte beteiligt und wie diese vernetzt sind; sie zeigt auch, wie die Funktionsblöcke auf die Steuergeräte verteilt werden.

Im vorherigen Abschnitt zur funktionalen Architektur wurde bereits die Rückwirkungsfreiheit angesprochen, die nun in der technischen Architektur hergestellt werden muss. Sie ist Grundlage für Redundanztechniken und die ASIL Dekomposition. Sie bezieht sich auf drei Aspekte, die auch in Teil 6 der ISO 26262 aufgelistet werden:

1. Räumliche Isolation der Komponenten
2. Zeitliche Isolation der Komponenten
3. Sicherer Austausch von Informationen zwischen den Komponenten

Isolation bedeutet, dass sich mögliche Fehler einer Komponente nicht auf andere Komponenten ausbreiten können. Die Isolationsdomänen können sowohl durch physische Trennung als auch durch software- und hardwarebasierte Techniken hergestellt werden. Die Auswahl geeigneter Mechanismen liegt im Bereich der Architekturentscheidungen in den verschiedenen Disziplinen *System*, *Software* und *Hardware*.

Die Entscheidung für die eine oder andere Variante wird durch die funktionalen und nichtfunktionalen Eigenschaften der jeweiligen Lösungen sowie durch die Betrachtungen der funktionalen Sicherheit im Rahmen der Failure-Mode-and-Effects Analyse (FMEA) getroffen. Im Folgenden nehmen wir an, dass diese Analysen bereits durchgeführt wurden.

Im Rahmen des Fallbeispiels werden nun exemplarisch eine integrierte Architektur und eine föderierte Architektur als Alternativen betrachtet:

- Variante "Dedizierte ECUs": In dieser föderierten Architektur werden die kamera- und radarbasierten Pfade auf den dedizierten Steuergeräten Camera ECU und Radar ECU ausgeführt. Die Plausibilisierung der beiden Auslöse-Entscheidungen und die eigentliche Auslösung des Airbags erfolgen auf der FABSY ECU. Verwendet man dedizierte ECUs, dann können die beiden Pfade hardwareseitig sehr gut isoliert werden.
- Variante "CDAC ECU": In dieser integrierten Architektur wird ein leistungsfähiges Steuergerät CDAC (Central Driver Assistance Controller) eingesetzt. Die gesamte Verarbeitung von der Akquisition des Bildes bis zur Entscheidung über die Auslösung des Airbags wird auf der CDAC ECU ausgeführt, und zwar für den kamera- und den radarbasierten Pfad. Die Plausibilisierung der Entscheidungen und die Auslösung des Airbags erfolgen wie in der Variante "Dedizierte ECUs" auf der FABSY ECU. Die Konsolidierung der Radar- und Kameraverarbeitung sowie evtl. weiterer Fahrer-Assistenzfunktionen auf einem leistungsfähigen Steuergerät kann Vorteile bezüglich Kosten, Gewicht und Bauraum haben, erfordert jedoch Maßnahmen in der System- und Software-Architektur zur Isolation der Pfade.

Die technischen Architekturvarianten können bezüglich der Erfüllung der Echtzeitanforderungen, Hardware-Kosten und weiterer Kriterien bewertet werden. Für die Umsetzbarkeit einer preislich günstigeren Variante ist jedoch entscheidend, dass alle Echtzeitanforderungen erfüllt werden.

Echtzeitanforderungen
Das FABSY System muss harte Echtzeitanforderungen erfüllen: Für bestimmte Aufgaben gibt es eine Deadline, zu der die Aufgabe abgearbeitet sein *muss*, da sonst die Systemfunktion und die damit verbundenen Sicherheitsziele nicht gewährleistet sind. Beispielhaft betrachten wir folgende Echtzeitanforderungen:

1. Das korrekte Scheduling muss sichergestellt und Mehrfachaktivierungen verhindert werden.

2. Die Latenzzeit zwischen dem Eintritt eines Fußgängers in den Fahrweg des Autos und dem Auslösen des Airbags darf maximal 150 ms betragen.

Durch eine modellbasierte Analyse des dynamischen Verhaltens lässt sich frühzeitig sicherstellen, dass keine systematischen zeitlichen Fehler in einer Architektur enthalten sind.

Analyse auf unterschiedlichen Architekturebenen

Eine erste Bewertung der Varianten kann schon auf der Ebene der funktionalen Architektur erfolgen: Aus der funktionalen Architektur werden Wirkketten abgeleitet und den einzelnen Funktionsblöcken Zeitbudgets und Aktivierungen zugewiesen. Bereits mit einem solchen hardware-unabhängigen Timing-Modell auf Basis einer Wirkkette ist eine erste Überprüfung von Anforderung 2 möglich.
Eine detaillierte Analyse der technischen Architektur erfolgt hardware-abhängig, indem u.a. folgende Eigenschaften in einem Timing-Modell berücksichtig werden:

- Busse (FlexRay, CAN)
- ECUs, Prozessoren und Cores
- Zuordnung von Software-Komponenten und Tasks
- Tasks, Mapping von Tasks/ISRs auf CPUs und Cores
- Scheduling-Eigenschaften von Tasks

Die ersten drei Punkte müssen gemäß ISO 26262 Part 4 bei einer Verifikation des Systementwurfs berücksichtigt werden, Punkt vier und fünf gemäß ISO 26262 Part 6 bei der Verifikation des Software-Entwurfs.
Im Folgenden wird beschrieben, wie die Varianten der technischen Architektur analysiert, optimiert und verifiziert werden.

Bewertung der beiden Varianten

Häufig treten beim ersten Entwurf einer System- oder Software-Architektur Echtzeitfehler auf. Dabei sind die Zuordnung von Software-Komponenten und Tasks sowie die Scheduling-Eigenschaften der Tasks Freiheitsgrade, die relativ einfach geändert werden können. Gleichzeitig haben diese Eigenschaften starken Einfluss auf die Einhaltung der Echtzeitanforderungen.

In einem effizienten Entwicklungsprozess werden dieser Eigenschaften modellbasiert analysiert und optimiert. Die Erstellung eines Timing-Modells, die Verifikation der Anforderungen und notwendige Optimierungen der Architektur sind ein integraler Bestandteil des Entwicklungsprozesses.

Um bei der detaillierten Analyse das Scheduling zu berücksichtigen, werden die Ausführungszeiten der Tasks als Nettozeiten spezifiziert, also die reine Ausführungszeit eines Tasks ohne Unterbrechungen. Häufig können die Netto-Ausführungszeiten der Software während der Entwurfsphase nicht auf dem Zielprozessor gemessen werden, da eine Implementierung noch nicht vorliegt. In diesem Fall werden die Netto-Ausführungszeiten als Zeit-Budgets festgelegt. Grundlage hierfür können Vorgaben, Expertenschätzungen oder Messungen aus Vorprojekten sein.

Ein Timing-Modell auf Basis der technischen Architektur ermöglicht es, mit einem geeigneten Analysewerkzeug die Anforderungen noch vor der Implementierung zu verifizieren (Verification of system design, ISO 26262-4). Hierbei ist die Verifikation von Anforderung 1 (korrektes Scheduling, keine Mehrfachaktivierungen) ein direktes Ergebnis einer Scheduling-Analyse. Hingegen wird für Anforderungen 2 eine Wirkkette modelliert, die abstrakt den Datenfluss beschreiben. Hierbei ist jeder Wirkkettenschritt ein relevanter Kommunikationspunkt, an dem Daten gepuffert, gelesen und anschließend verarbeitet oder geschrieben werden. Für jedes auslösende Ereignis (bei Anforderung 2 ist dies das Auftreten eines Objektes) wird während der Simulation eine neue Instanz der Wirkketten erzeugt und seine Anforderungen verifiziert.

In **Fehler! Verweisquelle konnte nicht gefunden werden.** ist die Analyse des dynamischen Verhaltens als Ausgabe des Echtzeitsimulators chronSIM zu sehen. Dargestellt ist eine Instanz der Wirkkette (ockerfarbene Linien) von Anforderung 2: Nach dem Eintritt eines Objektes folgen die Verarbeitung der Radar- (oberer Teil der Wirkkette) sowie der Kameradaten (unterer Teil der Wirkkette), das Senden über einen CAN- bzw. FlexRay-Bus zur FABSY ECU, die Plausibilisierung und schließlich das Auslösen des Airbags.

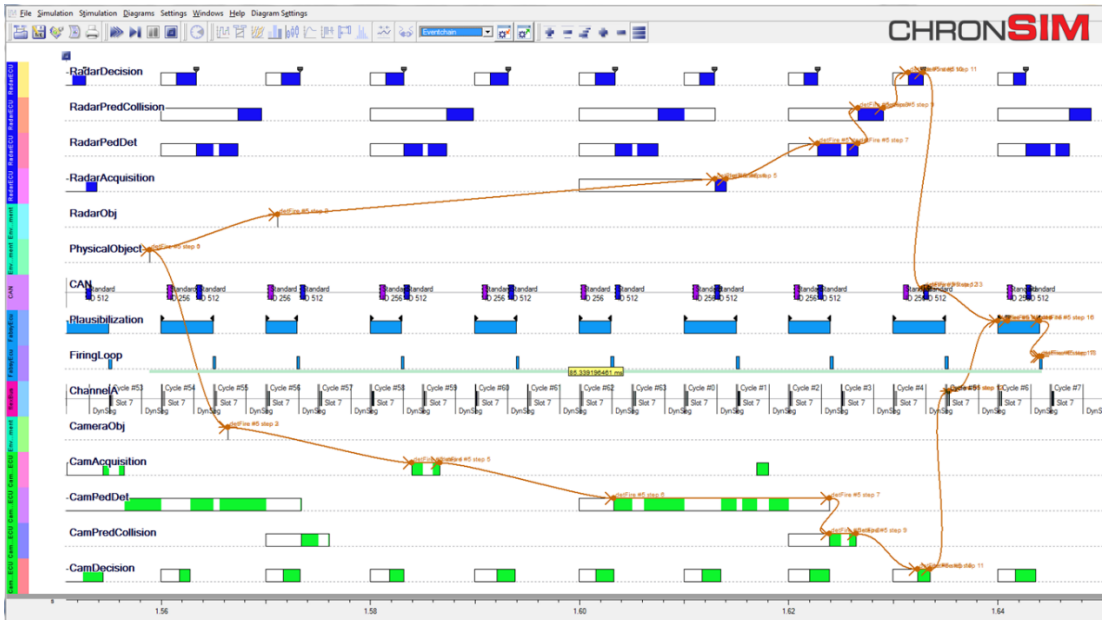

Bild 2: Analyse der Radar- und Kamerawirkkette für die Variante „Dedizierte ECUs“ einschließlich der Plausibilisierung mit dem Echtzeitsimulator chronSIM

Das vorgestellte Vorgehen erlaubt eine effiziente Optimierung der technischen Architektur. Schon nach wenigen Optimierungsschleifen zeigt sich, dass beide Varianten („Dedizierte ECU“ und „CDAC ECU") ihre Echtzeitanforderungen einhalten. Die Entscheidung kann anhand weiterer Kriterien, wie etwa Kosten, getroffen werden.

Mit weiteren Schritten kann das Echtzeitmodell verfeinert werden. Der vorgestellte Top-Down Spezifikationsansatz, der die Task-Eigenschaften und das Scheduling betrachtet, lässt sich durch eine Bottom-Up Analyse ergänzen. So kann man mit

verschiedenen Konfigurationen zur Ressourcenverteilung experimentieren. Bezieht man anwendungsspezifische Informationen ein, lassen sich optimistischere Aussagen über die Echtzeiteigenschaften des Programms auf einer konkreten Hardware treffen. Vorgegebenen Zeitbudgets können zudem auf Basis von Messdaten konkretisiert werden.

Fazit

Das vorgestellte Vorgehen ermöglicht es, kostengünstig verschiedene Architekturvarianten, Software- und Hardware-Konfigurationen und Implementierungen im Kontext von ISO 26262 und harter Echtzeit zu vergleichen. Der beschriebene Ansatz beeinflusst Sicherheitsanforderungen, Qualitätseigenschaften sowie Kostenbetrachtungen positiv.

Autoren

Dr. Ulrich Becker ist als Trainer, Berater und Coach bei Method Park tätig. Er arbeitet in verschiedenen Kundenprojekten zu den Themen Software-Architektur, Application Lifecycle Management und Verbesserung von Entwicklungsprozessen.

Dr. Isabella Stilkerich ist als Software-Architektin und Forscherin im Bereich E-Mobilität der Schaeffler Technologies AG & Co. KG tätig. Ihre Themenschwerpunkte sind Echtzeitsysteme, eingebettete Systeme, funktionale Sicherheit und AUTOSAR.

Dr.-Ing. Ralf Münzenberger ist bei der INCHRON GmbH für den Bereich Professional Services verantwortlich. In mehr als 160 Projekten hat er Kunden rund um das Thema Timing und Performance, Architekturoptimierung und funktionale Sicherheit beraten.