Architekturvarianten im Kontext von 26262 und harter Echtzeit

Die Anzahl aktiver Sicherheits- und Assistenzsysteme mit hohen Verfügbarkeitsanforderungen und harten Echtzeitanforderungen nimmt stark zu. Werden die Echtzeitanforderungen erst spät betrachtet, resultieren daraus oft teure Änderungen an der System-, Software- und Hardware-Architektur. Wie diese vermieden werden können, zeigt dieser Beitrag.

Als Fallbeispiel soll ein fiktives Fußgänger Airbag System (FABSY) dienen, das Fußgänger vor körperlichen Schäden bei der Kollision mit einem Auto schützt. 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 erwünschten Auslösung des Airbags auch berücksichtigt werden, dass durch eine unerwünschte Auslösung die Fahrzeuginsassen und weitere Verkehrsteilnehmer gefährdet werden können. Aus der Analyse der FABSY-Funktion ergeben sich harte Echtzeitanforderungen, etwa 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 Automotive Safety Integrity Level (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

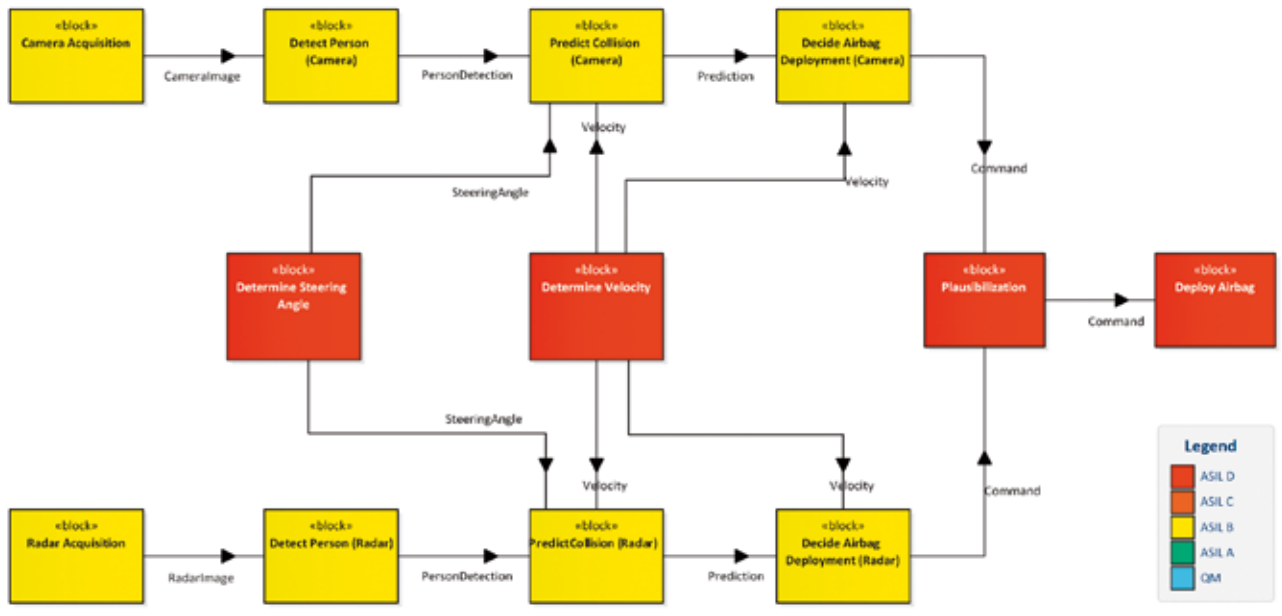

Bild 1: Funktionale Systemarchitektur von FABSY nach der ASIL-Dekomposition. Den Funktionsblöcken ist ein ASIL B (gelb) bzw. ASIL D (rot) zugewiesen. (© Method Park)

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).

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 Rahmen des Fallbeispiels werden zwei Alternativen betrachtet:

- „Dedizierte ECUs":
  Kamera- und radarbasierten Pfade werden auf dedizierten Steuergeräten, Kamera-ECU und Radar-ECU, ausgeführt.
- „CDAC ECU":
  Ein leistungsfähiges Steuergerät CDAC (Central Driver Assistance Controller) ist für die kamera- und radarbasierten Pfade verantwortlich.

Die Architekturvarianten können bezüglich der Erfüllung der Echtzeitanforderungen, Hardware-Kosten und weiterer Kriterien bewertet 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 werden folgende Echtzeitanforderungen betrachtet werden:

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 der zweiten Anforderung 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,
- Scheduling-Eigenschaften von Tasks.

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.

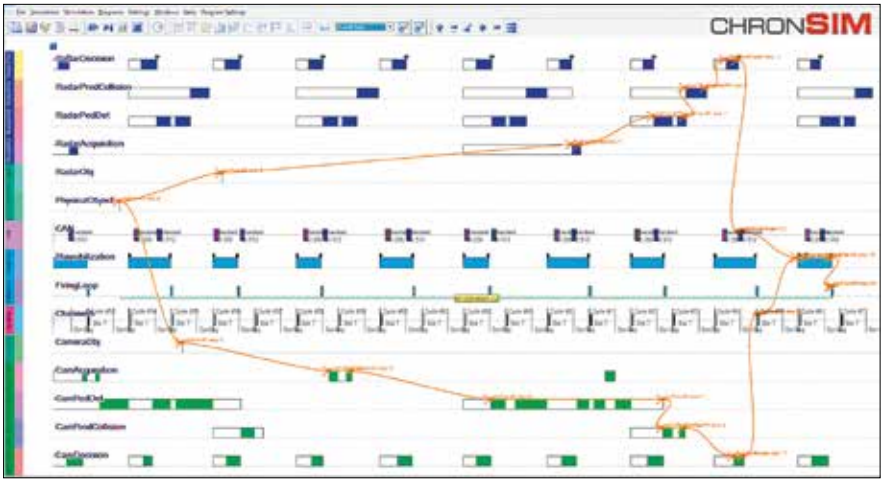

Bild 2: Analyse der Radar- und Kamerawirkkette für die Variante „Dedizierte ECUs“ einschließlich der Plausibilisierung mit dem Echtzeitsimulator chronSIM.
(© Method Park)

Um bei der detaillierten Analyse das Scheduling zu berücksichtigen, werden die Ausführungszeiten der Tasks als Nettozeiten spezifiziert, also die reine Ausführungszeit eines Task ohne Unterbrechungen.

Ein Timing-Modell auf Basis der technischen Architektur ermöglicht es, mit einem geeigneten Analysewerkzeug die Anforderungen noch vor der Implementierung zu verifizieren. Die Verifikation von Anforderung 1 (korrektes Scheduling, keine Mehrfachaktivierungen) ist ein direktes Ergebnis einer Scheduling-Analyse. Die weiteren Anforderungen werden auf der Grundlage von Wirkketten überprüft.

In Bild 2 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.

Das vorgestellte Vorgehen ermöglicht 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. Vorgegebene Zeitbudgets können zudem auf Basis von Messdaten konkretisiert werden.

Fazit

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

» **www.methodpark.de**

» **www.hanser-automotive.de/4239662**

Hier finden Sie die zusätzlich zu diesem Beitrag eine ausführliche Langversion.

Dr. Ulrich Becker, **Christian Lederer** und **Frank Pinecker** sind bei Method Park als Consultants, Trainer und Coach tätig. **Dr. Isabella Stilkerich** ist Software-Architektin im BMBF-Forschungsprojekt ARAMiS 2 bei Schaeffler Technologies. **Dr. Ralf Münzenberger** ist Mitgründer der INCHRON GmbH. **Philip Rehkop** arbeitet bei INCHRON als Professional Services Engineer.