Entwurfsabsicherung für eingebettete Mehrkernsysteme im Kontext der ISO 26262

Benedikt Bauer*, Jan Steffen Becker†, Thomas Peikenkamp†, Christof Schlaak†, Ingo Stierand†
*TWT GmbH Science & Innovation
benedikt.bauer@twt-gmbh.de
†OFFIS
becker,peikenkamp,schlaak,stierand@offis.de

Zusammenfassung—**Für rechenleistungsintensive Anwendungen wie Fahrerassistenzsysteme werden auch in eingebetteten Systemen zunehmend Mehrprozessor- und Mehrkernprozessorsysteme eingesetzt. In diesem Papier wird dargestellt, wie sich Anforderungen im Rahmen eines Mehrkernsystementwurfs unter Verwendung geeigneter, standardisierter Modellierungskonzepte ISO 26262 konform umgesetzt werden können. Dementsprechend wird aufgezeigt, wie sich sicherheitsrelevante Anforderungen der Systemebene auf die untergeordneten System-Komponenten (z.B. Regler) und letztendlich die HW/SW-Architektur heruntergebrochen werden. In sicherheitskritischen Anwendungsbereichen gibt es dort insbesondere auch Anforderungen und Einschränkungen, die bei der Partitionierung und Verteilung zu beachten sind. Für die Verteilung von Softwarekomponenten auf die einzelnen Prozessorkerne kommen hierbei meist Algorithmen zum Einsatz, da eine Verteilung unter Beachtung aller Zwangsbedigungen „von Hand“ nicht mehr zu leisten ist. Am Beispiel von Methoden und Werkzeugen, welche im AMALTHEA Projekt entwickelt wurden, wird gezeigt, welche Arten von Anforderungen dies sind und wie sie ermittelt werden können.**

Abstract*—*For computing-intensive applications such as driver assistance systems, multiprocessor and multi-core processor systems are increasingly being used in embedded systems. This paper shows how requirements for a multi-core system design can be implemented in compliance to ISO 26262 modeling concepts, i.e. safety standards. Accordingly, safety-relevant system-level requirements have to be broken down to the subordinate system components (e.g., controllers) as well as the hardware and software architecture. In safety-critical application domains, there are also requirements and limitations that must be taken into account during partitioning, mapping, and distribution processes. For the distribution of software components to the individual processing cores, various algorithms are mostly used, since a manual distribution under consideration of all constraints is error prone, time consuming, and ineffective. This paper finally presents a way to determine and identify requirement types along with methods and tools that are supported by the APP4MC platform.

I. Einleitung

Für rechenleistungsintensive Anwendungen wie Fahrerassistenzsysteme werden auch in eingebetteten Systemen zunehmend Mehrprozessor- und Mehrkernprozessorsysteme eingesetzt. Bei der Entwicklung eingebetteter Softwaresysteme muss im Automobilbereich insbesondere die Norm ISO 26262 [1] beachtet werden. Diese Arbeit beinhaltet eine Zusammenstellung von Ergebnissen nationaler und internationaler Forschungsprojekte (insbesondere Amalthea4public, Aramis II und ASSUME), die Fragen des sicherheitsgerichteten Entwurfs von Applikationen für Mehrkernsysteme untersuchen. Die Ergebnisse werden exemplarisch in einen ISO 26262-konformen Entwicklungsprozess eingeordnet.

Die Arbeiten wurden teilweise vom Bundesministerium für Bildung und Forschung (BMBF) unter den Förderkennzeichen 01IS14029A (Amalthea4Public), 01IS16025A (Aramis II) und 01IS15031H (ASSUME) gefördert.

Als laufendes Beispiel dient in dieser Arbeit ein ferngesteuerter, unbemannter Helikopter mit vier Rotoren, im folgenden Quadcopter genannt. Der Quadcopter dient zur Videoübertragung bei Ballspielen und ist daher mit einer schwenkbaren Kamera und Objekt-Tracking-Software ausgestattet. Die Flugsteuerung, insbesondere die Steuerung der Rotoren, wird als sicherheitskritisch betrachtet. Es wurde hier bewusst kein Beispiel aus dem Automobilbereich gewählt, um zu verdeutlichen, dass die vorgestellten Methoden auch in anderen Domänen als der ISO 26262 eingesetzt werden können. Der Aufbau dieses Artikels orientiert sich grob an der Struktur der ISO 26262. Abschnitt II stellt die Gefahren- und Risikoanalyse vor. Darauf aufbauend werden in Abschnitt III Anforderungen auf Systemebene und in Abschnitt IV. In Abschnitt V wird als Kernthema dieser Arbeit der Dienstbasierte Entwurf der technischen Architektur vorgestellt und dies in Abschnitt VI für die Hardware/Software-Schnittstelle eingesetzt. Die Arbeit schließt mit einem Ausblick in Abschnitt VII.

II. HARA & Ableitung Sicherheitsziele

Die Anforderungen, die bei der Implementierung einer sicherheitskritischen Systemfunktion (sog. *Item*) zu erfüllen sind, leiten sich nach ISO 26262 aus der Gefahren- und Risikoanalyse (engl.: *Hazards analysis and risk assessment*, kurz: HARA) und den daraus bestimmten Sicherheitszielen ab. Im Rahmen der Gefahren- und Risikoanalyse wird untersucht, welche Fehlfunktionen des Items in welchen Nutzungssituationen zu einer Gefährdung von Menschen führen können. Die Gefährdung wird in den Kategorien Schwere der zu erwartenden Verletzungen (S0–S3), Kontrollierbarkeit (C0–C3) und Exposition (E0–E4) bewertet und daraus eine Kritikalitätsbewertung in Form eines *Automotive Safety Integrity Level* (ASIL) gebildet. Auf den Gefährdungsereignissen basierend werden Sicherheitsziele als höchstrangige Sicherheitsanforderungen definiert und den Gefährdungsereignissen

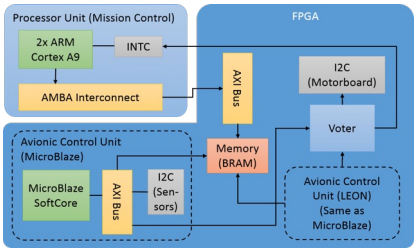

Abbildung 1. Ausschnitt aus der HW-Architektur des Quadcopters

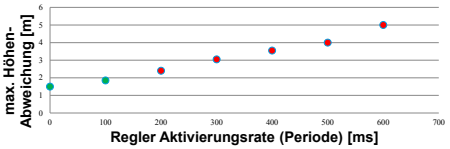

Abbildung 2. Abhängigkeit der durch einen Windstoß verursachten Flughöhenabweichung von der Aktivierungsrate des Höhenreglers, ermittelt in 7 Simulationsläufen (grün = innerhalb, rot = außerhalb des erlaubten Bereiches).

zugeordnet, deren Eintritt sie verhindern sollen. Die Sicherheitsziele erhalten nach ISO 26262 neben anderen Informationen einen ASIL und eine Fehlertoleranzzeit, sofern anwendbar, die sich aus den Bewertungen der jeweils zugeordneten Gefährdungsereignisse ergeben. Wenn sichere Zustände bekannt sind, sollen auch diese angegeben werden. Ein Beispiel anhand des Quadcopters ist in Tabelle I dargestellt.

Wird die Systemfunktion in Teilfunktionalitäten zerlegt, so ist für jede daraus resultierende Anforderung zu prüfen, welche Sicherheitsziele durch Fehler der Teilfunktionalitäten verletzt werden können. Entsprechend erben abgeleitete Anforderungen nach gewissen Regeln ASIL und Fehlertoleranzzeit von den übergeordneten Anforderungen. Hierbei gilt, dass eine Anforderung mit einem bestimmten ASIL durch zwei parallel durchgeführte Funktionalitäten mit dann niedrigerem ASIL erfüllt werden. In letzterem Fall ist sicherzustellen, dass die Stärke der Segregation zwischen den beiden parallelen Funktionalitäten dem ASIL der ursprünglichen, nicht dekomponierten Anforderung entspricht.

Beim Quadrocopter wird das Sicherheitsziel SZ2 „Die Motorsteuerung muss ausfallsicher sein“ beispielsweise dadurch erfüllt werden, dass die Steuerungskomponenten redundant ausgelegt werden. Die relevante HW-Architektur ist in Abb. 1 dargestellt. Die Software für die Motorsteuerung wird standardmäßig sowohl auf den MicroBlaze- und LEON-Softcores als auch dem ARM-Hardware-Kern ausgeführt. Die Sensorwerte werden über die I2C-Schnittstelle gelesen und die resultierenden Motorstellwerte in den BRAM geschrieben. Die Voter-Komponente vergleicht die berechneten Motorsignale, und leitet bei Übereinstimmung zweier Signale dieses an das Motorboard weiter. Gemäß der ASIL-Dekomposition kann das ASIL für die Stellwertberechnung nun herabgesetzt werden. Für die ECUs MicroBlaze, LEON ASIL und die Mission Control Unit ergibt sich ASIL A(C)[1] und für den Voter ASIL B(C). Die maximale Signallaufzeit Sensoren → Rechenkerne → Rotoren kann experimentell ermittelt werden, wie in Abschnitt III dargestellt.

III. Berücksichtigung von Mehrkernarchitekturen bei der funktionalen Dekomposition

Die Tiefe der funktionalen Dekomposition sollte zu einer Verfeinerung führen, die es gestattet, Teilfunktionen eindeutig Elementen der Zielplattform zuordnen zu können. Mehrkernsysteme bieten kostengünstige Hardware mit viel Rechenleistung und spielen dadurch bei komplexen Eingebetteten Systemen eine zunehmend wichtigere Rolle. Bei der Realisierung einer Systemfunktion mit mehreren parallel arbeitenden Berechnungselementen entstehen allerdings neue Herausforderungen. Bei zeitkritischen Funktionen werden Garantien benötigt, die die Verfügbarkeit von Ergebnissen zu einem gewissen Zeitpunkt zusichern. Scheduling- und Mapping-Entscheidungen sollen vom Entwickler so gewählt werden, dass die benötigten Berechnungsergebnisse auch rechtzeitig bereit stehen. Damit diese Entscheidungen richtig getroffen werden können, werden konkrete technische Anforderungen (z.B. an die Aktivierungsperiode von periodisch durchgeführten Berechnungen) benötigt, welche zwar implizit in den Anforderungen auf Systemebene enthalten sind, aber daraus zunächst noch abgeleitet werden müssen. Für diese Dekomposition der System-Anforderungen werden zusätzliche Informationen beispielsweise über die eingesetzte Hardware benötigt. Diese fließen dann in die abgeleiteten Anforderungen der Teilfunktionen mit ein.

In Sicherheitsziel SZ1 (siehe I) wurde identifiziert, dass der Quadcopter gegen Windstöße robust sein muss. In Abb. 2 wurden die Auswirkungen verschiedener Berechnungszeiten simuliert. So können konkrete technische Anforderung an die untergeordneten Systemkomponenten (Scheduling der Regler) abgeleitet werden.

IV. Ableitung von Softwareanforderungen

Parallel zur funktionalen Dekomposition werden neben Hardwareanforderungen auch feingranulare Softwareanforderungen aus den Systemanforderungen und den Sicherheitszielen abgeleitet. Für einen ISO 26262-konformen und effizienten Entwicklungsprozess ist es wichtig, dass Anforderungen über die verschiedenen Entwurfsebenen und -sichten hinweg nachverfolgt werden können. Um hohe Qualitätsstandards sicherzustellen und unnötige Kosten durch Fehler in späteren Entwicklungsschritten zu vermeiden, sollten Anforderungen möglichst früh während der funktionalen Dekomposition auf Konsistenz überprüft werden. Dabei helfen formale Spezifikationssprachen wie das *Simplified Universal Pattern* [2], [3]. Diese erlauben durch ihre eindeutige Semantik automatisierte (Konsistenz-)Analysen und Testfallgenerierung. In [4] wird eine integrierte Toolchain vorgestellt, die Formalisierung, Konsistenzanalyse, Testfallgenerierung und Qualitätsmanagement abdeckt.

[1]Das C in Klammern ist eine Referenz auf den ASIL des Sicherheitsziels.

Hazard	Fehlerfall	Situation	Sicherheitsziel	ASIL
Unkontrolliertes Flugverhalten	Abweichung $> Y$ der Flugsteuerung	Windstärke $\leq X$, im Flug	SZ1: Die Positionsabweichung durch Seitenwinde $\leq X$ darf maximal Y betragen.	E4,C3,S2 $\Rightarrow$ C
Absturz	Ausfall der Motorsteuerung	Im Flug	SZ2: Die Motorsteuerung muss ausfallsicher sein.	E4,C3,S2 $\Rightarrow$ C

Tabelle I
TEIL DER HARA FÜR DEN QUADCOPTER

V. NACHVERFOLGUNG VON ANFORDERUNGEN ENTLANG DES HARDWARE/ SOFTWARE ENTWURFS

Die technische Realisierung des Systems besteht einerseits aus der Entwicklung einer Software- aus der Systemarchitektur, und andererseits aus der Auswahl einer geeigneten Hardwarearchitektur, auf der die Software schließlich ausgerollt wird. Die ISO 26262 fordert, dass in diesem Prozess die Sicherheitsanforderungen der Systemarchitektur und daraus abgeleitete Fehlermodelle in der technischen Realisierung nachverfolgt werden. Eine dienst-basierte Modellierung soll diese Nachverfolgbarkeit stark vereinfachen. Hierbei werden die Funktionen mit Diensten (im Sinne von Fähigkeiten) assoziiert, die sie für die Umsetzung ihrer Aufgabe benötigen. Zum Beispiel muss für eine Funktion, die für ihre Operation Eingabedaten benötigt, ein entsprechender Dienst zum Erhalt dieser Daten zur Verfügung stehen. Ähnlich werden Funktionen, die als Softwarekomponenten realisiert werden, entsprechend die Fähigkeit zur Ausführung durch die Hardware benötigen. Ein konzeptuelles Metamodell zur Modellierung von Diensten ist in Abb. 3 dargestellt, welches als Erweiterung des AMALTHEA Metamodells angelegt ist.

Vermöge dieses Dienstkonzeptes kann nunmehr auch die Fehlermodellierung erfolgen, indem Fehler nicht mehr direkt mit den Funktionen sondern mit Diensten assoziiert werden. Fehler von Diensten, die für die Realisierung der Schnittstellen benötigt werden, werden ebenfalls an diesen Schnittstellen sichtbar. Fehler von Ausführungsdiensten werden als interne Systemfehler der Funktion sichtbar. Die Identifikation von Fehlerpropagationen erfolgt mit etablierten Methoden, beispielsweise durch entsprechende Fehlermodellierung (z.B. HIP-HOPS [5], AltaRica [6]). Sobald entsprechende Ausführungsmodelle der Funktionen existieren, sollte dies durch direkte Fehlerinjektion erfolgen (z.B. [7]). Hierdurch kann sichergestellt werden, dass die Fehlermodellierung die relevanten Fehlerphänomene tatsächlich abdeckt.

Darüber hinaus bietet dieser Ansatz die Möglichkeit, bereits während der Entwicklung der Funktionsarchitektur (abstrakte) Hardware-Ressourcen mit den Diensten zu assoziieren (vgl. *uses* Beziehung zwischen *ServiceCapability* und *Resource*). Hierdurch können unter anderem räumliche Segregationanforderungen ausgedrückt werden, indem Funktionen Dienste mit unterschiedlichen assoziierten Resourcen benutzen. Schließlich zeigt Abb. 3 eine Möglichkeit, Spezifikationen zu erstellen, die die Benutzung der Dienste genauer formulieren. Hiermit können dann zeitliche Segregationeigenschaften ausgedrückt werden, indem die maximalen Interferenzen durch gemeinsame Benutzung der assoziierten Ressourcen festgelegt werden.

Der dargestellte Ansatz wird für die Softwarearchitektur ebenfalls angewendet. Dabei werden bei dem Softwareentwurf die verwendeten Schnittstellen definiert und ebenso wie für die Ausführung der Funktionen mit den dafür notwendigen Dienste assoziiert. Im Zuge dieses Entwurfsschrittes wird die Nachverfolgbarkeit der in der Funktionsarchitektur vorgenommenen Fehlermodellierung durch eine Allokation der entsprechenden Dienste sichergestellt. In Abb. 4 ist dies im unteren Bereich durch gestrichelte Linien angedeutet. Diese Beziehung ermöglicht es dem Entwickler unter anderem nachzuvollziehen, ob die Kombination aus HW/SW Fehlern durch die identifizierten Systemfehler abgedeckt sind. Ebenso kann auf diese Weise die Konsistenz der Fehlermodellierung im Hinblick auf Fehlerpropagationen überprüft werden. Insbesondere können bereits Verletzungen von bestimmten Unabhängigkeitseigenschaften überprüft werden, die durch die Allokation von mehreren Diensten der Funktionsarchitektur auf Dienste der Softwarearchitektur entstehen können (in Abb. 4 z.B. durch die Allokation der Dienste *ServiceA* der Funktionen auf *ServiceA'*).

VI. HW/SW SCHNITTSTELLE

Die HW/SW-Schnittstelle stellt für die Umsetzung der ISO 26262 ein Schlüsselement in dem Sicherheitsprozess der technischen Realisierung dar, da durch sie die Beziehungen zwischen auftretenden HW Fehlern und ihrer Wirkung auf die SW sichtbar wird. In MC Systemen wird diese Aufgabe dadurch erschwert, dass aufgrund der gemeinsamen Ressourcennutzung komplexe Wechselbeziehungen entstehen können.

Um Anforderungen der ISO in einem modellbasierten Ansatz zu adressieren, wird die Hardware mit ihren bereitgestellten Schnittstellen und den möglichen Fehlertypen modelliert. Auch hier kann das oben beschriebene dienstbasierte Vorgehen genutzt werden. Auf Basis dieser Modelle ist es dann möglich, bei der Allokation der Softwarekomponenten auf die vorhandene Hardware auch die Zwangsbedingungen zu berücksichtigen, die sich aus den Anforderungen der Funktionalen Sicherheit ergeben. Die oben eingeführten Konzepte werden mit in dem Projekt AMALTHEA4public entwickelten Modellierungskonzepten kombiniert. Die Modellierungssprache erlaubt die Definition von sogenannten *Zugriffspfaden*, mit deren Hilfe Abläufe innerhalb der Hardware dargestellt werden können. Ein Beispiel aus der Flugsteuerung des Quadcopters zeigt Abb. 5, in dem der Zugriffspfad vom LEON-Softcore über den Systembus (AXI) zum Datenspeicher (BRAM) definiert ist.

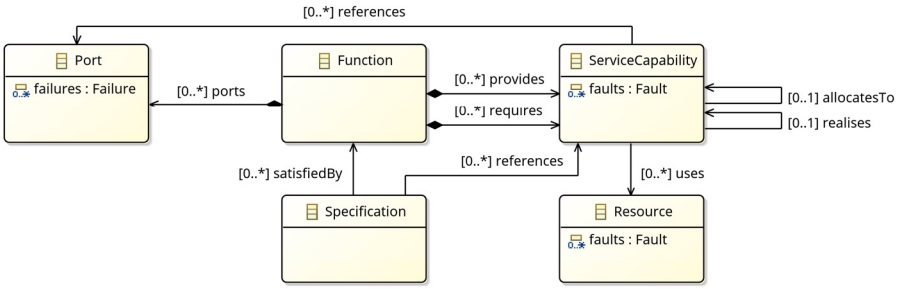

Abbildung 3. Konzeptuelles Metamodell für Dienste und Ressourcen

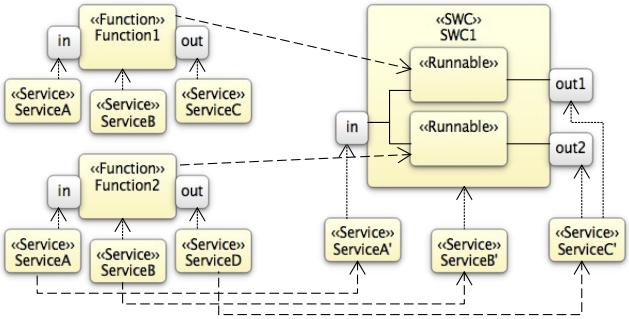

Abbildung 4. Allokation von Diensten zwischen Funktionen und Software

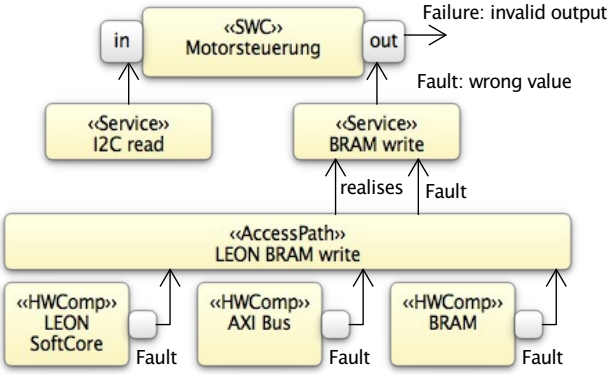

Abbildung 5. Modellierung der HW/SW Architektur

Die HW/SW-Schnittstelle wird hierbei mit Hilfe der Dienste realisiert. In dem Beispiel wird angenommen, dass Ausgaben der Softwarekomponente über einen Dienst realisiert werden, bei dem die Daten in den lokalen Datenspeicher geschrieben werden. Umgesetzt werden diese Dienste durch die „Benutzung“ dieser Zugriffspfade, was durch eine entsprechende Realisierungsbeziehung in Abb. 5 angedeutet ist.

Der Modellierungsansatz adressiert mehrere Anforderungen eines ISO 26262 konformen Entwicklungsprozesses. Erstens erlaubt dies die Analyse von Segregationseigenschaften, indem untersucht werden kann, welche Elemente auf die verschiedenen Hardwarewaressourcen zugreifen. Durch eine verfeinerte Spezifikation des zeitlichen Aspekts, d.h., wann und wie häufig Dienste von Softwarekomponenten genutzt werden, kann die zeitliche Segregation durch eine entsprechend durchgeführte Schedulinganalyse überprüft werden.

Zweitens erlaubt der Modellierungsansatz die detaillierte Analyse von Fehlereffekten der Hardware auf die Softwarekomponenten. Entsprechende Fehlermodelle für die einzelnen Hardwarekomponenten vorausgesetzt, erlaubt die Modellierung von Zugriffspfaden die Ableitung von entsprechenden Fehlermodellen für die Zugriffspfade selbst und damit indirekt für die damit assoziierten Dienste.

VII. Zusammenfassung und Ausblick

Das Papier zeigt wesentliche Anforderungen an ISO26262 konforme Entwicklungsprozesse auf, wobei insbesondere die durchgängige und nachverfolgbare Absicherung von Sicherheitszielen bei dem Entwurf der verschiedenen Architekturebenen (System, Hardware, Software) für Mehrkernsysteme Berücksichtigung findet.

Neben der notwendigen Integration der beschriebenen Konzepte in das AMALTHEA Metmodell ist zu bemerken, dass mit Zugriffspfaden und entsprechenden Fehlermodellen angereicherte Hardwaremodelle anwendungsunabhängig und damit wiederverwendbar sind. Es wäre daher wünschenswert, wenn solche Modelle von den Hardwareherstellern bereitgestellt würden.

Literatur

[1] *ISO 26262:Road vehicles—Functional safety*, 1st ed., International Organization for Standardization, 11 2011.

[2] T. Bienmüller, T. Teige, A. Eggers, and M. Stasch, “Modeling requirements for quantitative consistency analysis and automatic test case generation,” in *Workshop on Formal and Model-Driven Techniques for Developing Trustworthy Systems*, 2016.

[3] T. Teige, T. Bienmüller, and H. J. Holberg, “Universal pattern: Formalization, testing, coverage, verification, and test case generation for safety-critical requirements,” in *MBMV*, 2016.

[4] J. S. Becker, V. Bertram, T. Bienmüller, U. Brockmeyer, H. Dörr, T. Peikenkamp, and T. Teige, “Interoperable toolchain for requirements-driven model-based development,” in *ERTS*, 2018.

[5] Y. Papadopoulos and J. A. McDermid, *Hierarchically Performed Hazard Origin and Propagation Studies*, 1999.

[6] F. Kuntz, S. Gaudan, C. Sannino, É. Laurent, A. Griffault, and G. Point, “Model-based diagnosis for avionics systems using minimal cuts,” in *DX*, M. Sachenbacher, O. Dressler, and M. Hofbaur, Eds., Germany, 2011, pp. 138–145.

[7] T. Peikenkamp, A. Cavallo, L. Valacca, E. Böde, M. Pretzer, and E. M. Hahn, *Towards a Unified Model-Based Safety Assessment*, 2006.