Umsetzung der Anforderungen aus der ISO 26262 bei der Entwicklung eines Steuergeräts aus dem Fahrerinformationsbereich.

Dr. Antje Gieraths[1], Christoph Müller-Albrecht[1], Sean Brown[2]

[1]Instruments & Displays, HMI Software
BMW Group
[2]Telemotive AG
antje.gieraths@bmw.de
christoph.mueller-albrecht@bmw.de
sean.brown@partner.bmw.de

Kurzfassung: Die seit November 2011 geltende ISO Norm 26262 gilt als die Norm zur Erreichung und Umsetzung funktionaler Sicherheit im Automobil. In diesem Artikel wird anhand eines konkreten Steuergeräts beschrieben wie die Anforderungen aus der ISO Norm für die ASIL Abstufungen A und B erreicht werden können. Das gewählte Steuergerät beeinhaltet dabei Sicherheitsanforderungen und Anforderungen aus dem Bereich der Fahrerinformationssysteme, die in Einklang gebracht werden müssen. Die ISO Norm gibt verschiedene Arbeitsschritte vor, beispielsweise die Definition der Anforderungen, die Implementierung der Software und die verschiedenen Testarten. In diesem Artikel soll besonders auf den Entwicklungsschritt der Softwarearchitekturdefinition eingegangen werden. Es wird gezeigt, wie die in der ISO Norm geforderte „Freedom from Interference“ durch Software Design umgesetzt werden kann. Die hierfür notwendigen Voraussetzungen, z.B. auf Betriebssystemseite werden erläutert. Insbesondere im Sicherheitsbereich spielt eine hohe Qualität der Software und eine kontinuierliche Rückmeldung darüber an den Projektleiter und die Entwickler ein wichtige Rolle. Entsprechende Metriken sind während der Softwareentwicklungsphase kontinuierlich zu erheben. In diesem Beitrag soll gezeigt werden wie Continuous Integration mit einem hohen Automatisierungsgrad zur Umsetzung der ISO Norm eingesetzt werden kann. Dabei wird auf die konkreten Implementierungsschritte und einige Werkzeuge eingegangen.

1 Einleitung

1.1 Prämissen

Seit November 2011 ist die ISO26262 in Kraft. Sie gilt als die Norm zur Erreichung und Umsetzung funktionaler Sicherheit im Automobilbereich. Sie leitet sich aus der IEC EN DIN 61508 ab. Die ISO26262 gibt ein Vorgehensmodell mit standardisierten

Arbeitsprodukten vor. Beispiele für die Arbeitsprodukte sind das technische Sicherheitskonzept, der Verification Plan oder der Verification Report. Die ISO Norm deckt die Konzeptphase, die Produktentwicklungsphase, die Produktion, die Prozesse sowie die Analysen als Arbeitsschritte ab. Für ein Steuergerät wird die Produktentwicklung auf Systemebene, auf Hardware und Software Ebene (Teil 5 und 6 der ISO Norm) beschrieben. Die ISO Norm beschreibt die Umsetzung der funktionalen Sicherheit für vier Abstufungen der Automotive Safety Integrity Level (ASIL) – A bis D. In diesem Artikel wird auf das ein Steuergerät aus dem Fahrerinformationsbereich eingegangen, das die Level A bzw. B zu erfüllen hat. Ausgehend von einem Steuergerät, das als Zwei-Prozessorsystem aufgebaut ist und das Headup-Display (HUD) sowie das Kombiinstrument ansteuert, wird erläutert, wie die Darstellung von sicherheitsrelevanten Anzeigen wie den Check Control Meldungen (z.B. „Motorhaube offen“) umgesetzt wird.

1.2 Motivation

Aus anderen Anforderungen, die beispielsweise die passive und aktive Sicherheit beim Fußgängerschutz betreffen, entstehen Entwicklungen wie eine elektronische Verriegelung der Motorhaube im Fahrzeug. So kann bei einem Zusammenprall mit einem Fußgänger oder Radfahrer die Motorhaube aufgestellt werden. Die Schwere der (Kopf-) Verletzungen kann durch diese Maßnahme gemindert werden. Zur Umsetzung dieser Maßnahme ist es jedoch erforderlich, dass die Motorhaube geschlossen ist und ein Fehler dem Fahrer zur Anzeige gebracht wird. Hier findet der Austausch einer mechanischen durch eine elektonische Komponente statt. Dies hat jedoch zur Folge, dass daraus Anforderungen an Anzeigeinstrumente zur ISO26262-konformen Software Entwicklung entstehen. Das heißt, es muss eine Anzeige geben, die den Fahrer über den Zustand der elektronischen Motorhaubenverriegelung sicher und zuverlässig informiert (Check Control „Motorhaube offen.“, siehe Abbildung 1).

Abbildung 1 - Motorhaube offen - Checkcontrol Meldung im freiprogrammierbaren Kombiinstrument.

2 Vorgehensmodell

Die ISO Norm gibt ein V-Modell zur Umsetzung der Software Anforderungen vor. Es beginnt mit dem Systementwurf („Initiation of Product Development“) und der Spezifikation der Sicherheitsanforderungen. Jeder Abschnitt in der ISO Norm enthält nach den ASIL Leveln gestaffelte Maßnahmen. Diejenigen Schritte, die mit „++“ bzw. highly recommended gekennzeichnet sind, sind als verpflichtend für das entsprechende ASIL Level anzusehen. Zu den ersten Schritten gehört beispielsweise auch die Definition von Modellierungs- und Codierungsrichtlinien (vgl. Abbildung 2). Für jeden Prozessschritt entstehen so konkrete Anforderungen, die überprüfbar und mit Hilfe von Werkzeugen umgesetzt werden können.

Table 1 — Topics to be covered by modelling and coding guidelines

	Topics	ASIL			
		A	B	C	D
1a	Enforcement of low complexity[a]	++	++	++	++
1b	Use of language subsets[b]	++	++	++	++
1c	Enforcement of strong typing[c]	++	++	++	++
1d	Use of defensive implementation techniques	o	+	++	++
1e	Use of established design principles	+	+	+	++
1f	Use of unambiguous graphical representation	+	++	++	++
1g	Use of style guides	+	++	++	++
1h	Use of naming conventions	++	++	++	++

[a] An appropriate compromise of this topic with other methods in this part of ISO 26262 may be required.

[b] The objectives of method 1b are
- Exclusion of ambiguously defined language constructs which may be interpreted differently by different modellers, programmers, code generators or compilers.
- Exclusion of language constructs which from experience easily lead to mistakes, for example assignments in conditions or identical naming of local and global variables.
- Exclusion of language constructs which could result in unhandled run-time errors.

[c] The objective of method 1c is to impose principles of strong typing where these are not inherent in the language.

Abbildung 2 Anforderungen an die Modellierungsrichtlinien aus dem Teil 6 der Software Anforderungen. Quelle[ISO26262]

3 Umsetzung von Architekturanforderungen

3.1. Signalfluss im Kombiinstrument:

Das als Kombiinstrument bezeichnete Steuergerät besteht aus zwei Prozessoren – der automotive Prozessor (MCU) und dem Grafikprozessor (GPU). Das Steuergerät wird bei BMW als big-little System entworfen, d.h. man geht von einem relativ kleinen automotive Kern aus und einem großen Grafikprozessor, der seinerseits einen Grafikkern und eine kleine CPU enthält. Auf der MCU läuft AUTOSAR als Betriebssystem und verschiedene Software Komponenten, beispielsweise

Signalverarbeitungsbibliotheken und Treibersoftware. Des weiteren gibt es Komponenten, die das Flashen und die Diagnosefähigkeit umsetzen. Abbildung 3 zeigt einen groben Überblick über den Signalfluss innerhalb der Software. Ein Signal, z.B. das der Ganganzeige kommt über den CAN Bus an MCU an, durchläuft die verschiedenen AUTOSAR Schichten und gelangt dann zu den Signalverarbeitungsbibliotheken, dort wird das Signal gefiltert (wenn notwendig) und auf Fehler überprüft. Die Kommunikation zwischen den Modulen findet über die AUTOSAR RTE (Run Time Environment) statt. Die Treiber, die beispielsweise das Backlight des Displays ansteuern, tauschen ihre Daten mit anderen Software Komponenten auf der MCU über die RTE aus. Das Signal, das ursprünglich über den CAN auf dem Steuergerät ankam, gelangt dann über die Interprozessor Kommunikation auf die GPU. Dort wird es von der HMI (Human Machine Interface) Software dargestellt.

3.2 Umsetzung

In [ISO26262], Teil 6, Abschnitt 4 heißt es:

When claiming compliance with ISO 26262, each requirement shall be complied with, unless one of the following applies:

a) tailoring of the safety activities in accordance with ISO 26262-2 has been planned and shows that the requirement does not apply, or

b) a rationale is available that the non-compliance is acceptable and the rationale has been assessed in accordance with ISO 26262-2.

Das bedeutet, es gibt drei Möglichkeiten die Anforderungen der ISO Norm zu erfüllen:

1. Alle Software ist nach dem geforderten ASIL Level zu entwickeln.
2. Alle Software wird nach dem geforderten ASIL Level entwickelt und es werden Argumentationen und Begründungen gegeben, die nach der ISO Norm akzeptiert werden.
3. Die Software wird so partionioniert, dass nur die Teile, die für die funktionale Sicherheit relevant sind, nach ASIL qualifiziert werden und die sogenannte „Freedom from Interference“ , d.h. Rückwirkungsfreiheit gewährleistet ist.

Die Software Architektur, in der die Komponenten und ihre Funktionalität und ihr Zusammenspiel mithilfe dynamischer und statischer Sichten festgelegt wird, ist also von immenser Wichtigkeit bei der Umsetzung der ISO Norm [BS2012]. Die Methode der „Decomposition“ bei der Betrachtung einer Komponente als „Safety Element out of Context“ kann sehr hilfreich sein. Hier kann ein System so entworfen werden, dass aus zwei Komponenten, die ASIL A erfüllen, Anforderungen nach ASIL B erfüllt werden können – durch geschicktes Systemdesign und eine entsprechende Aufteilung.

Der in 3.1 beschriebene Signalfluss muss nun nach den in der ISO Norm beschriebenen Kriterien abgesichert werden. Auf der MCU bedeutet dies ein entsprechendes Betriebssystem – AUTOSAR mit ASIL B einzusetzen, das Speicherschutz und Laufzeitgarantie bietet. Über CRC[1] Checks und Alive Counter wird die Aktualität und Validität des Signals überprüft. Dies gilt ebenfalls für die Übertragung zur GPU.

In der HMI Software sollen nun verschiedene Anzeigen mit der ASIL Einstuftung A integriert mit anderen Anzeigen (Tacho, Drehzahlmesser, etc.) ohne ASIL Einstufung angezeigt werden.

Als Prämisse für die vorgestellte Lösung gilt außerdem, dass der verwendete Grafikprozessor mehrere Hardware Grafiklayer zur Verfügung stellt. Dies ist notwendig um sicherzustellen, dass die sicherheitsrelevanten Anzeigen auf einem separaten Layer angezeigt werden können. Dieser muss auch so konfiguriert werden können, dass er nicht überschrieben werden kann.

Im folgenden wird beschrieben, wie das Konzept der Freedom from Interference auf der GPU Seite umgesetzt werden kann.

Als zentrale Idee werden sicherheitsrelevante und nicht-sicherheitsrelevante Software Anteile von einander getrennt, damit sie sich nicht negativ beeinflussen können. Es wird eine sogenannte SafetyHMI entwickelt, die nur die ASIL-relevanten Anzeigen enthält. Die anderen Anzeigen werden in der regulären HMI Software zusammengefasst und müssen die Sicherheitsanforderungen nicht erfüllen. Dazu ist ein Betriebssystem notwendig, welches Laufzeit- und Speicherschutz bietet. Die Inhalte für die funktionale Sicherheit befinden sich in einem eigenen (virtuellen) Adressraum und werden über einen ebenfalls nach ISO26262 entwickelten Grafiktreiber angezeigt. Durch die verschiedenen Grafiklayer, welche die GPU Hardware bereitstellt, ist es möglich sicherheitsrelevante Anzeigen auf einem Layer zu konzentrieren. Die Layer werden dabei so konfiguriert, dass der oberste Layer mit seinen Elementen immer sichtbar bleibt.

Abbildung 3 zeigt den Aufbau der „Safety HMI“ und der HMI innerhalb des Gesamtaufbaus. Um die Komplexität in der SafetyHMI so gering wie möglich zu halten, werden Texte als vorgerenderte Bilder im Speicher gehalten.

Wenn die Safety HMI nach allen Regeln der ISO Norm entwickelt wird, dann können mit der oben genannten Trennung alle sicherheitsrelevanten Anzeigen abgedeckt und die Konformität mit der ISO26262 sichergestellt werden.

[1] CRC = Cyclic Redundancy Check

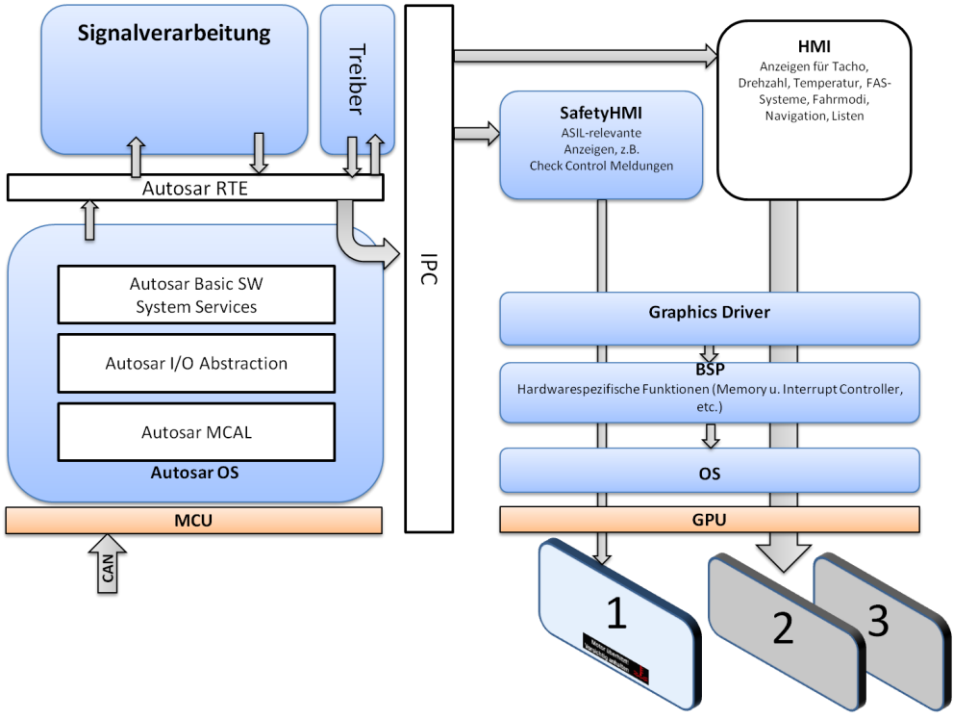

Abbildung 3 Signalfluss Diagramm mit SafetyHMI und HMI. Die sicherheitsrelevanten Anzeigen werden immer im vordersten Layer angezeigt. Die beiden HMI Softwaren laufen parallel, aber in verschiedenen Adressräumen. Alle hellblau markierten Flächen müssen nach der ISO Norm 26262 entwickelt werden. Die Betriebssysteme auf dem Grafikcore und dem Automotive Core müssen Laufzeitgarantie und Speicherschutz bieten. Der erste Layer ist der unüberschreibbare Hardware Layer, welcher die sicherheitsrelevante Anzeigen darstellt. Der zweite Layer dient zur Darstellung der regulären HMI Software mit Tachometer, Drehzahlmesser, den Fahrmodi, der Navigation und den Listen. Der dritte Layer stellte die Anzeigeelemente auf dem Head Up Display (HUD) dar.

4 Continuous Integration zur Umsetzung der ISO Norm.

Continuous Integration und auch Continuous Deployment sind Schlagwörter, die in den letzten Jahren zunehmend an Bedeutung gewonnen haben [QT2012]. In der Software Entwicklung für eingebetteten Systeme sind schnelle und flexible Entwicklungszyklen mit einer raschen Rückmeldung an die Entwickler genau so wünschenswert wie im Bereich der Websoftware.

Zur Umsetzung der ISO26262 müssen u.a. die folgenden drei Abschnitte in der Softwareentwicklung durchgeführt werden:

1. Software Design und Implementierung

2. Software Modul Test

3. Software Integration und Test

In diesem Abschnitt soll darauf eingegangen werden, wie die Anforderungen, die sich aus der ISO26262 ergeben, mit Continuous Integration nachhaltig erfüllt und welche Tools dabei genutzt werden können.

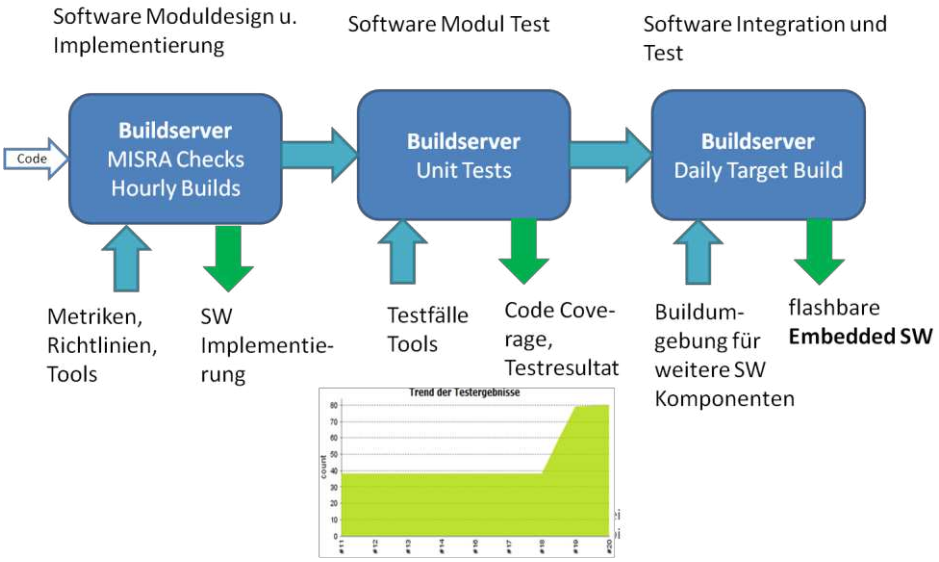

Abbildung 4 Buildkette für die Erstellung der eingebetteten Software. Als Ergebnisse der einzelnen Schritte erhält man die implementierte Software, die Testresultate und die Code Coverage sowie aus dem letzten Schritt die flashbare Software für die Target Hardware.

4.1 Software Design und Implementierung

In diesem Schritt werden die Software Module erstellt und die Funktionalität implementiert. Es gilt nach vorher festgelegten Regeln zu programmieren und die Einhaltung dieser Regeln messbar zu überprüfen.

Die Software muss kompilierbar sein und auf eventuelle Fehler der Entwickler muss schnell reagiert werden können.

Eine Continuous Integration Umgebung, beispielsweise der Jenkins Server [Jenkins] wird dazu genutzt, mehrfach pro Tag, idealerweise nach jeder Veränderung des Quellcodes in der Versionsverwaltung die Software zu kompilieren. Parallel geben Werkzeuge zur statischen Code Analyse, die automatisiert auf dem Server gestartet werden, den Entwicklern und dem Projektleiter eine Rückmeldung bezüglich der Verletzung von vorher festgelegten Programmierrichtlinien. Auch diverse Metriken, wie die Anzahl der Codezeilen, die zyklomatische Zahl oder die Kommentarfrequenz können

so erfasst werden. Bei Verletzung der Regeln erhalten die Entwickler direkt Rückmeldung über den Server.

Als Ergebnis dieses Schrittes der Continuous Integration Kette erhält man die implementierte und geprüfte Software (vgl. Abbildung 4).

4.2. Software Modul Test

Eine ausreichende Testabdeckung ist essentiell zur Erreichung einer zufriedenstellenden Qualität und vor allem Stabilität im Software Entwicklungsprozess. Auch hier kann Continuous Integration unterstützen. Die Testfälle und entsprechende Werkzeuge, wie das entsprechende Testframework (CUnit, Googletest, etc.) laufen automatisiert auf dem Buildserver. Auch hier erhalten Entwickler und Tester schnell eine Rückmeldung über den Erfolg oder Mißerfolg der geschriebenen Tests. Dabei ist dieses Feedback allgemeingültiger als lokal ausgeführte Tests auf dem jeweiligen Rechner des Entwicklers, weil beispielsweise nur lokale Einstellungen, die nur für einen Rechner bzw. Entwickler gelten, schnell aufgedeckt werden [QT2012].

Weiterhin wird automatisiert ermittelt welcher Prozentsatz der Funktionalität innerhalb der Software durch die Module tatsächlich abgedeckt wird. Dabei wird zunächst der Quellcode instrumentiert und dann mit den Modultests ermittelt welche Funktionen oder Statements damit getestet werden. Dies wird dann mit der insgesamt geforderten Code Coverage ins Verhältnis gesetzt.

Das Ergebnis dieses Schrittes sind also die Testergebnisse sowie die Code Coverage der Modultests (vgl. Abbildung 4).

4.3. Software Integration und Test

Ein entscheidender Teil des Software Entwicklungsprozesses bei der Entwicklung eingebetteter Systeme ist die Software Integration und der Test. Hierbei wird die entwickelte Funktionalität auf der Zielhardware integriert – meist werden dazu andere Compiler und Werkzeuge genutzt als im täglichen Entwicklungsprozess auf den Computern der Entwickler. Sehr oft ist auch mehr als eine Softwarekomponente zu integrieren. Beispielsweise wird die HMI Software für ein Fahrerinformationssystem an einem Ort entwickelt, die Systemfunktionen von einem anderem Team und/oder einer anderen Firma an einem anderen Ort und das Betriebssystem kommt von einer dritten Partei. Hier muss eine Umgebung geschaffen werden, die es ermöglicht z.B. die Anzeigesoftware, die großen optischen Veränderungen und Anpassungen unterworfen ist, auf der Zielhardware komplett zu integrieren und zu testen. Das Kompilat für die Zielhardware wird automatisiert und täglich erstellt. So erhalten die Entwickler zeitnah Rückmeldung über Warnungen und Fehler mit dem Ziel-Compiler und können diese viel schneller beheben, als wenn der Integrationsschritt immer am Schluss der Entwicklung erfolgt. Die Rückmeldung vom Buildserver erfolgt in der Regel per Email direkt an den jeweiligen Entwickler.

Als Ergebnis dieses Schritts in der Continuous Integration Kette steht eine funktionsfähige flashbare Software für die Zielhardware zur Verfügung.

5 Zusammenfassung

In diesem Artikel wird beschrieben, wie funktionale Sicherheit für ein Steuergerät aus dem Bereich der Fahrerinformation, d.h. mit grafischer Anzeige umgesetzt werden kann. Hierbei wird die Trennung der sicherheitsrelevanten und nicht sicherheitsrelevanten Software als zentrale Möglichkeit im Software Architektur Design genutzt.

Im zweiten Teil des Artikels wird ausgeführt wie für drei verschiedene Schritte in der ISO-konformen Softwareentwicklung Continuous Integration für die Entwicklung von eingebetteten Systemen genutzt wird.

Literatur

[ISO26262] ISO Norm 26262 zur Fahrzeugsicherheit.

[BS2012] Becker, Ulrich und Sechser, Bernhard, Sicherheit entwerfen statt testen., Automotive 11, 2012

[Jenkins] http://www.jenkins.org

[QT2012] Klaus Quibeldey, Christoph Thelen, Continuous Deployment, Informatik Spektrum, Springer Verlag 2012, https://www.gi.de/service/informatiklexikon/detailansicht/article/deployment-continuous.html