Gesellschaft für Informatik e.V. (GI)

publishes this series in order to make available to a broad public recent findings in informatics (i.e. computer science and information systems), to document conferences that are organized in cooperation with GI and to publish the annual GI Award dissertation.

Broken down into
• seminars
• proceedings
• dissertations
• thematics
current topics are dealt with from the vantage point of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure high quality contributions.

The volumes are published in German or English.

Information: http://www.gi.de/service/publikationen/lni/

ISSN 1617-5468
ISBN 978-3-88579-663-3

"Automotive 2017" is the seventh event in a conference series focusing on safety and security in the field of automotive and other critical software application domains.This volume contains the papers accepted for presentation at the conference.

269

P. Dencker, H. Klenk, H. Keller, E. Plödereder (Hrsg.):Automotive 2017

GI-Edition

Lecture Notes in Informatics

Peter Dencker, Herbert Klenk, Hubert Keller, Erhard Plödereder (Hrsg.)

Automotive – Safety & Security 2017

Sicherheit und Zuverlässigkeit für automobile Informationstechnik

30.–31. Mai 2017
Stuttgart

Proceedings

Peter Dencker, Herbert Klenk, Hubert B. Keller,
Erhard Plödereder (Hrsg.)

Automotive - Safety & Security 2017

Sicherheit und Zuverlässigkeit für automobile Informationstechnik

30. – 31.5.2017
Stuttgart, Germany

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings
Series of the Gesellschaft für Informatik (GI)

Volume P-269

ISBN 978-3-88579-663-3
ISSN 1617-5468

Volume Editors
Herbert Klenk
Airbus
Rechliner Str.
85077 Manching
E-Mail: herbert.klenk.external@airbus.com
Erhard Plödereder
Universität Stuttgart
Universitätsstr. 38
70569Stuttgart
E-Mail: ploedere@informatik.uni-stuttgart.de

printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort

Die 7. Tagung Automotive – Safety & Security 2017 - Sicherheit und Zuverlässigkeit für automobile Informationstechnik fand am 30. - 31. Mai 2017 an schon vertrautem Platz in Stuttgart-Feuerbach im Auditorium der Robert Bosch GmbH statt. Die Tagungsserie ist mit der Zuverlässigkeit und Sicherheit softwarebasierter Funktionen im Automotive-Bereich befasst. Zwei Tage lang wurden die neuesten Ideen und konkreten Lösungen für die drängenden Herausforderungen der Softwareentwicklung mit Schwerpunkt auf Sicherheit und Zuverlässigkeit sowie Qualität in den Endprodukten diskutiert. Hochkarätige eingeladene Hauptredner waren Franco Gasperoni, Christian Wieschebrink und Stefan Jähnichen.

Franco Gasperoni ist Mitbegründer der Firma AdaCore, die als Produzent der Ada und Spark Technologien weltweit bekannt ist und deren Software gerade in Bereichen mit höchster Zuverlässigkeit, wie in der Avionik, eingesetzt werden. Seit 2016 ist Franco Gasperoni auch Chief Executive Officer von AdaCore. In seinem Vortrag „Software Safety and Security in a World of Systems“ referierte er über die Möglichkeiten, mit technologischen Mitteln die Sicherheit und Zuverlässigkeit heutiger komplexer Systeme deutlich zu verbessern.

Christian Wieschebrink ist seit 2004 Referent im BSI, wo er sich vor allem mit Privatsphäre bewahrenden codebasierten kryptographischen Verfahren befasst. In seinem Vortrag „IT-Sicherheit für das vernetzte Fahren“ gab er Einblicke in die nötigen Elemente der IT-Sicherheit der kooperativen Systeme für vernetzte Fahrzeuge. Die konkrete Formulierung von IT-Sicherheitsanforderungen für Fahrzeugkomponenten und deren konstruktive Umsetzung war ein weiterer Aspekt.

Stefan Jähnichen leitete das Fraunhofer Institut FIRST und war Ordinarius an der Universität Berlin. Seit seiner Pensionierung engagiert er sich im neu entstehenden Einstein Center Digital Future. Leitend in vielen Gremien war er unter anderem Präsident der Gesellschaft für Informatik und ist Aufsichtsratsvorsitzender des Leibniz Zentrums für Informatik Schloß Dagstuhl. Er adressierte in seinem Vortrag „Anonymität und Authentifizierung im vernetzten Fahrzeug“ die immer wichtiger werdenden Themen der Privatheit von Daten und der Sicherheit im Zugang zu vernetzten Fahrzeugen.

Direkt nach der Konferenz fand am 31.5. ein Tutorial von Jan Pelzl, Hochschule Hamm-Lippstadt, zum Gebiet „Intrusion Detection and Prevention Systems for Automotive Systems“ statt. Seit 2015 hat Pelzl die Professur für Computer Security an der Hochschule Hamm-Lippstadt inne. Seit 1999 war er im Bereich der eingebetteten IT-Sicherheit tätig und setzte erfolgreich viele nationale und internationale Projekte um. Zuvor war er von 2007 bis 2014 technischer Geschäftsführer der ESCRYPT GmbH, einer Tochtergesellschaft der Robert Bosch GmbH. Seine Forschungsschwerpunkte liegen in dem Bereich der Embedded Security und Cyber Security, insbesondere Automation, Automotive und Medical. Er ist Autor des Lehrbuchs „Understanding Cryptography“.

Parallel zur Konferenz zeigte eine Ausstellung einschlägige Werkzeuge rund um die automotive Softwareentwicklung. Der Konferenz vorgeschaltet fanden bereits am 29. Mai Sitzungen von GI und VDI Fachausschüssen und –gremien statt, die auch fachliche Träger dieser Konferenz sind. Anschließend traf man sich zu einem informellen Get Together für

bereits anwesende Konferenzteilnehmer. Das Networking konnte in den Kaffee- und Mittagspausen der Konferenz und insbesondere beim Konferenzdinner am 30. Mai vertieft werden.

Die Automobilindustrie erfährt einen grundlegenden Wandel durch die rasch fortschreitende Digitalisierung, alternative Antriebskonzepte, Vernetzung von Fahrzeugen und Infrastrukturen sowie autonomen Fahrfunktionen. Die Weiterentwicklung klassischer Methoden und Vorgehensweisen zur Sicherstellung der erforderlichen Software-Qualität sicherheitskritischer Anteile wird aktuellen automobilen Anforderungen nicht mehr gerecht. Zukünftig ist insbesondere die Reduzierung der Komplexität unter Beachtung von Safety und Security Anforderungen, sowie deren Wechselwirkung wichtig zur Sicherstellung einer zuverlässigen Funktion unter allen sich ergebenden Situationen. Hinzu kommen die Anforderungen der Beherrschung des autonomen Fahrens.

Zwar hat die Diskussion über softwarebasierte „Defeat Switches" in Diesel-Fahrzeugen in den letzten Monaten die Schlagzeilen der Presse und damit auch die Aufmerksamkeit der Führungsetagen beherrscht, aber in den vorausblickenden Arbeiten der Entwicklungsabteilungen ist man sich inzwischen der Wichtigkeit der Abschirmung der automotiven Systeme gegenüber böswilligem Eindringen bewusst. Gleichzeitig macht der Übergang der Haftung auf den Hersteller bei autonomem Fahren klar, dass für die Zuverlässigkeit der Software auch eine dem Stand der Technik entsprechende Softwareentwicklung nötig sein wird. Neueste Untersuchungen belegen auch hier noch Defizite. Die Automotive – Safety&Security Konferenzserie liegt damit genau im Trend.

Im Vorwort der Konferenz im Jahr 2015 schrieben wir: „ *...Kritische Ereignisse in letzter Zeit zeigen drastisch, dass ohne die Berücksichtigung entsprechender Sicherheitsanforderungen sowohl aus dem Safety- als auch aus dem Security-Bereich erhebliche Risiken in der zukünftigen Entwicklung drohen. .... Autonomes Fahren unter massiver Vernetzung und hochkomplexer Informationsverarbeitung mit interpretativen Entscheidungen des Fahrzeugs erfordern höchste Zuverlässigkeit und gleichzeitig transparentes und nachvollziehbares Verhalten."*

Diese Sätze sind unverändert gültig. Allerdings können wir ein Wandern des Brennpunkts beobachten: Während sich die damaligen Beiträge vorrangig um technische Lösungen zur Vermeidung von Gefährdungen der Sicherheit (im Sinne der Security) bemühten, ist diesmal ein Trend zu erkennen, bereits in den frühen Phasen der Systementwicklung geeignete Analysen durchzuführen und Maßnahmen zu ergreifen, die das Risiko von Sicherheitsverletzungen und von funktionalem Versagen zu minimieren suchen. Ausgehend von der Grundforderung nach der *Zuverlässigkeit softwarebasierter Funktionen sind Safety und Security Anforderungen integriert zu betrachten und umzusetzen.*

An dieser Stelle möchten wir dem Programmkomitee danken, das zu den nötigen vielen Reviews bereit war, um die Begutachtung aller Beiträge durch mindestens vier Gutachter zu gewährleisten. Aus der Menge der eingereichten Papiere wurden die besten und einschlägigsten Beiträge für die Konferenz und die Proceedings ausgewählt, die erneut in der „Lecture Notes in Informatics"-Serie publiziert werden. Leider konnten nur rund die Hälfte der Einreichungen in das Konferenzprogramm aufgenommen werden. Wir danken allen Autoren für ihr Interesse an der Konferenz. In der Schlusssession der Konferenz war,

wie immer, die Spannung groß, wer die Preise für den besten Beitrag und für die beste Präsentation erhalten würde.

Die Tagung ist Teil einer Serie, die seit 2004 im zweijährigen Turnus veranstaltet wird. Die Tagungsbände der Serie sind Zeitzeugen eines hochinteressanten Wandels der Interessen rund um software-gestützte Funktionen im Automobil. Seit dem anfänglichen Ringen um bessere Zuverlässigkeit in den damals verbesserungsbedürftigen Systemen haben wir das Entstehen von einschlägigen Richtlinien und Standards, z. B. EN 26262, erlebt, deren Notwendigkeit 2004 noch sehr umstritten war. Seit 2012 ist zu beobachten, dass sich das Interesse der Sicherheit zuwendet, nun, da Automobile nicht mehr autarke Einheiten sind, sondern beginnen, in globale Kommunikationsnetze eingebunden zu sein. Die bislang oft ungesicherten Kommunikationsprotokolle im Auto müssen mit den nötigen technischen Mitteln gegen unautorisierte Zugriffe und Beeinflussungen abgesichert werden. Weltpolitische Ereignisse haben gezeigt, dass das Eindringen in Systeme nicht nur von einigen wenigen böswilligen Hackern in Garagen betrieben wird. Sicherlich haben sie dazu beigetragen, dass Sicherheit zu einem globalen Thema geworden ist. Hardware-orientierte Angriffe zeigen, dass Security von Grund auf in allen Ebenen der Systeme zu betrachten ist und auf technischer Ebene neue Sicherheits- und Zuverlässigkeitsfragen für die IT im Fahrzeug aufwirft. Aber auch die das Fahrzeug umgebenden Prozesse, von der Diagnostik in der Werkstatt und der Wartung über die Bereitstellungsmethoden von Telefonie und Infotainment, bis hin zur Erfüllung gesetzlicher Anforderungen rund um E-Call-Dienste, müssen in die Sicherheitsanalysen aufgenommen werden und zweifelsohne Neuerungen erfahren.

Die fachlichen Träger der Automotive-Tagung sind die Fachgruppen Ada, ASE, ENCRESS, EZQN, FERS und FoMSESS der Gesellschaft für Informatik in den Fachbereichen "Sicherheit - Schutz und Zuverlässigkeit" und "Softwaretechnik" sowie der Fachausschuss "Embedded Software" der VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik und der Förderverein Ada Deutschland e.V. Die Tagungsleitung hatten Hubert B. Keller, Karlsruher Institut für Technologie, und Klaus Fronius, ETAS GmbH, inne. Die wissenschaftliche Leitung erfolgte durch Hubert B. Keller, Karlsruher Institut für Technologie, und Erhard Plödereder, Universität Stuttgart. Unser Dank geht an das restliche Organisationsteam, bestehend aus Peter Dencker, Hochschule Karlsruhe (Ausstellung), Christoph Grein (Web), Reiner Kriesten, Hochschule Karlsruhe (Tutorials), Natascha Funk, ETAS GmbH (Tagungssekretariat und -vorbereitung), Herbert Klenk, Airbus (Finanzen und Tagungsband) und Klaus Fronius, ETAS GmbH (Organisation vor Ort).

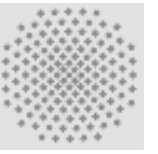
Der besondere Dank der Tagungsleitung geht an die ETAS GmbH für die großzügige Bereitstellung der Tagungsstätte und der Verpflegung der Tagungsteilnehmer und an Frau Cornelia Winter, Gesellschaft für Informatik, für die zügige Bearbeitung der Proceedings. Dank auch an den Förderverein Ada Deutschland e.V. für die Übernahme der finanziellen Verantwortung für die Gesamtveranstaltung.

Karlsruhe und Stuttgart, im Frühjahr 2017

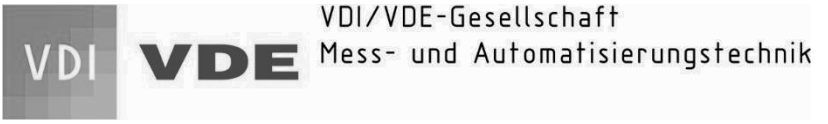
Hubert B. Keller Erhard Plödereder

Sponsoren und Aussteller

Wir danken den folgenden Unternehmen und Institutionen für die Unterstützung der Tagung.



Universität Stuttgart



Wissenschaftliche Leitung / Vorsitz Programmkomitee

Hubert B. Keller, Karlsruher Institut für Technologie (KIT)
Erhard Plödereder, Universität Stuttgart

Tagungsleitung

Hubert B. Keller, Karlsruher Institut für Technologie
Klaus Fronius, ETAS GmbH

Organisation

Klaus Fronius, ETAS GmbH (Lokale Organisation)
Herbert Klenk Airbus (Finanzen, Tagungsband)
Peter Dencker, Hochschule Karlsruhe (Ausstellung)
Reiner Kriesten, Hochschule Karlsruhe (Tutorien)
Christoph Grein (Web)

Programmkomitee

Gerhard Beck, Rohde & Schwarz SIT GmbH
Manfred Broy, TUM; Zentrum Digitalisierung.Bayern
Stefan Bunzel, Continental AG
Simon Burton, Robert Bosch GmbH
Peter Dencker, Hochschule Karlsruhe
Dirk Dickmanns, Airbus
Bernhard Fechner, FU Hagen
Hannes Federrath, Universität Hamburg
Felix Freiling, Universität Erlangen-Nürnberg
Klaus Fronius, ETAS GmbH
Simon Fürst, BMW Group
Rüdiger Grimm, Universität Koblenz-Landau
Erwin Großpietsch, EUROMICRO
Albert Held, Daimler AG
Bernhard Hohlfeld, TU Dresden
Dieter Hutter, DFKI / Universität Bremen
Stefan Jähnichen, TU Berlin
Jan Jürjens, TU Dortmund
Herbert Klenk, Airbus
Reiner Kriesten, Hochschule Karlsruhe
Thomas Kropf, Robert Bosch GmbH
Ulrich Lefarth, Thales Deutschland
Tobias Lorenz, Continental Automotive GmbH
Jürgen Mottok, OTH Regensburg
Francesca Saglietti, Univ. Erlangen-Nürnberg

Ina Schaefer, TU Braunschweig
Jörn Schneider, Hochschule Trier
Elmar Schoch, Audi AG
Claus Stellwag, Elektrobit Automotive GmbH
Werner Stephan, DFKI
Theodor Tempelmeier, Hochschule Rosenheim
Michael Weyrich, Universität Stuttgart
Hans-Jörg Wolff, Robert Bosch GmbH
Thomas Wollinger, escrypt GmbH

Fachliche Träger und Veranstalter

Gesellschaft für Informatik mit den Fachgruppen Ada, ASE, ENCRESS, EZQN, FERS, FoMSESS
VDI/VDE-GMA mit dem Fachausschuss 5.11 Embedded Software;
Förderverein Ada Deutschland. e.V.

Inhaltsverzeichnis

Funktionale Sicherheit in Automotive und Avionik: Ein Staffellauf

Andreas Schwierz[1] Georg Seifert[1] Sebastian Hiergeist[1]

Abstract: Der nachfolgende Bericht geht auf die gemeinsamen Interessen von sicherheitskritischen Systemen aus der Luftfahrt- und der Automobilbranche ein. Hierbei wird dargelegt, dass die Software-Funktionalität stark von der eingesetzten Hardware abhängig ist und Auswirkungen auf die gewünschte Sicherheit hat. In diesem Bereich können beide Branchen voneinander profitieren. Die Luftfahrt hat historisch gesehen schon früh angefangen, systematisch funktionale Sicherheit zu standardisieren, wohingegen die Automobilbranche seit 2011 nachzieht und mit ihrer großen Marktmacht auf die Hardwarehersteller einwirken kann. Hieraus könnte auch die Luftfahrtindustrie ihren Nutzen ziehen.

Keywords: Luftfahrt, Automobil, Funktionale Sicherheit, Echtzeitsysteme, WCET, Redundanzsystem, Interferenz, Zugriffskollisionen

1 Einleitung

Luftfahrtzeug- und Automobilhersteller teilen beide das Interesse, sichere Beförderungsmittel bereitzustellen. Getrieben durch den Wandel von mechanischen zu elektrischen/-elektronischen (e/e) Systemen begann in der Luftfahrt ab den 1980ern[2] eine strukturierte Auseinandersetzung mit funktionaler Sicherheit. Die Herausforderung war eine Beibehaltung bzw. Steigerung der Sicherheitsansprüche um ein Fail-Operational Verhalten garantieren zu können. Erreicht wurde dies durch die Entwicklung von Redundanzarchitekturen auf Systemebene. Verwendung fanden hierbei Mikroprozessoren (Micro Processor Unit, MPUs), die mit proprietärer Erweiterungen (ASICs, FPGAs, usw.) um die entsprechende Redundanzfunktionalität erweitert wurden. Im Vordergrund stand hier die Sicherheit des Gesamtsystems, während andere Faktoren wie Gewicht, Größe, Energieverbrauch und Kosten einen untergeordneten Stellenwert einnahmen.

Die Automobilbranche hat erst mit der Veröffentlichung des ISO 26262 [Te09] im Jahr 2011 ein Äquivalent zu den Avionik-Standards definiert. Bedingt durch die Anforderungen aus der Automobilbranche lag der Entwicklung des Standards eine andere Anforderungsbasis zugrunde. Der Hauptunterschied besteht darin, dass ein Fail-Safe Verhalten für solche Systeme ausreichend ist. Die oben genannten Faktoren wie Gewicht, Größe, Energieverbrauch und Kosten haben bei diesen Systemen einen ungleich höheren Stellenwert. Um diese Anforderungen erfüllen zu können wurde seitens der Chip-Hersteller viel

[1] Technische Hochschule Ingolstadt, Zentrum für Angewandte Forschung, Paradeplatz 13, 85049 Ingolstadt, Vorname.Nachname@thi.de

[2] Im Jahr 1982 wurde der Standard DO-178[Sp82] veröffentlicht.

Entwicklungs- und Forschungsaufwand betrieben, um die geforderten Safety-Aspekte im Rahmen des ISO 26262 zu erfüllen.

Dem historisch bedingten Erfahrungsvorsprung zum Trotz konnte die Luftfahrtindustrie hingegen keinen Einfluss auf die Entwicklung sicherheitskritischer und gleichzeitig hochintegrierter Hardware-Komponenten ausüben[FK06]. In sicherheitskritischen Avionik-Systemen muss dies durch den Einsatz von bewährten MPUs, in Kombination mit einer Redundanz auf Systemebene, gelöst werden. Zur Steigerung der Integrationsdichte wurde in der zivilen Luftfahrt der Integrated Modular Avionic (IMA) Ansatz entwickelt, der erstmals im Airbus A380 zum Einsatz kam. Diese Architektur ist auf die Verwendung in großen, komplexen Systemen ausgelegt und für den Einsatz in sicherheitskritischen Avionik-Systemen ungeeignet. In Anbetracht der anvisierten Einsatzszenarien von unbemannten Flugkörpern[Vo13] müssen deswegen funktional hochintegrierte Mikrocontroller (Microcontroller Unit, MCUs) verwendet werden.

Seitens der Automotive Mikrocontroller (Automotive Microcontroller Unit, AMCUs)-Hersteller standen hier domänenspezifischen Entwicklungsanforderungen im Vordergrund. Damit AMCUs in künftigen Avionik-Systemen eingesetzt werden können, muss die Zulassbarkeit argumentiert werden. Bei der Entwicklung des Avionik-Systems entstehen Commercial off-the-shelf (COTS)-spezifische herausforderungen, von denen drei in den nachfolgenden Kapiteln aufgegriffen werden.

Kapitel 2 erläutert den Stellenwert der **Qualität** bei der Avionik-Herstellung und stellt die Frage, ob Hardware-Komponenten, entwickelt nach ISO 26262, eine Erleichterung in der Flugzeugzulassung darstellen können. Trotz der umfangreichen Safety-Features innerhalb der AMCU wird die **Zuverlässigkeit** einer einzelnen AMCU nicht ausreichen, um den höchsten Sicherheitsansprüchen gerecht zu werden. Deswegen werden in Kapitel 3 aktuelle Sicherheitsbedenken untersucht und entsprechende Lösungsmöglichkeiten aufgezeigt. Als sinnvolle Lösung erscheint in diesem Zusammenhang die Realisierung eines Redundanznetzwerkes durch AMCU-eigenen Bordmittel. Aufgrund der dadurch stark steigenden Datenlast auf den einzelnen MCU müssen die Einflüsse von Eingabe/Ausgabe (E/A)-Datenflüssen auf die Software-Ausführungszeit detaillierter untersucht werden. Dieser Aspekt wird in Kapitel 4 ausführlich betrachtet.

2 Avionik-Entwicklung: Qualität von komplexen COTS

In der Luftfahrt und Automobilindustrie beschreiben domänenspezifische Standards die Entwicklung von sicherheitskritischer Hardware als Teil eines Systems. In der Luftfahrt ist dies der Standard DO-254[Sp00] – in der Automobilindustrie der ISO 26262[Te09]. Sie sind aus dem Bewusstsein entstanden, dass ein strukturierter Entwicklungsprozess notwendig ist um systematische Fehler bei komplexen Komponenten zu vermeiden. Dies muss das Ziel während der Entwicklung sein, da durch eine umfangreiche Verifikation nicht alle Entwicklungsfehler aufgedeckt werden können.

Diese Erkenntnis gilt für komplexe COTS-Komponenten. Deren Entwicklungsprozess bzw. die daraus resultierende Qualität muss zulassungskonform sein. Das heißt, es muss hierbei

eine qualitative Bewertung über den Entstehungsprozess der Komponente erstellt werden. Das daraus resultierende Ergebnis ist eine Vertrauensaussage über die Integrität bzw. Qualität der COTS-Komponente. Als Bewertungsgrundlage für die Qualität kann dabei der domänenspezifische Standard herangezogen werden. Dieser Vergleich ist gewinnbringender, wenn der COTS-Entwicklungsprozess auf einem verbreiteten Standard beruht. Da die gewonnenen Erkenntnisse auf einem Vergleich zwischen zwei Standards basieren ist das Abstraktionsniveau hoch. Die Wiederverwendung ist dadurch unabhängig von Hersteller und Produkt.

Der AMCU stellt für die Luftfahrtindustrie eine komplexe COTS-Komponente dar und erfüllt die Anforderung: Entwickelt nach einem verbreiteten Standard. Die Zielsetzung des ISO 26262 bzgl. der Entwicklung funktional sicherer Systeme ist eine Ausgangsvoraussetzung für einen Vergleich. Diese Halbleiterprodukte werden in Zukunft weitere sicherheitsrelevante Funktionen übernehmen und die Hersteller sind sich der steigenden Nachfrage bewusst. Mit diesem Anspruch wird die aktuelle Entwurfsversion des ISO 26262 aus dem Jahr 2016 um Teil 11[3] ergänzt, damit die domänenspezifischen Anforderungen an die Halbleiterentwicklung sichergestellt werden können. Ein domänenübergreifendes Qualitätsverständnis wird Einfluss auf die Entwicklung von künftigen Halbleiterprodukten haben, von denen beide Branchen profitieren.

Im weiteren Verlauf dieses Kapitels wird auf die aktuelle Zulassungssituation von COTS-basierter Avionik eingegangen. Zusätzlich soll dargestellt werden, welche Vorteile die Herstellung einer Vergleichbarkeit der branchenabhängigen Standards in Avionik-Projekten bringen kann.

2.1 Zulassung COTS-basierter Avionik

Einleitend ist zu klären, dass der Begriff *Zulassung* im Zusammenhang mit Avionik-Systemen der Lesbarkeit geschuldet ist. Tatsächlich wird ein Flugsystem von der Zulassungsbehörde als Ganzes genehmigt. Ein Avionik-System wird hierbei nicht einzeln betrachtet, doch muss es konform zu den Regularien entwickelt worden sein damit es *zulassbar* ist. Somit kann im folgenden Avionik-Zulassung als die *Zulassbarkeit* eines Avionik-Systems verstanden werden.

Noch bevor die Hardware von e/e Systemen im Fokus der funktionalen Sicherheit stand, beschrieb man mit dem DO-178[4] die Softwareentwicklung für Luftfahrtanwendungen. Die Absicht war, konkretere Zielvorgaben für die Software-Entwicklung zu definieren, um die abstrakteren Sicherheitsrichtlinien der Zulassungsbehörde zu erfüllen. Mit dem DO-254 [Sp00] wurde 2000 ein Standard veröffentlicht der dieses Ziel für die Hardware-Entwicklung verfolgt. Als übergeordnetes Bindeglied zwischen den DO-178B und dem DO-254 dient die aktuelle Veröffentlichung des Standards ARP4754A [SA10]. Er beschreibt den Entwicklungsprozess für Avionik-Systeme und definiert die beiden Standards als entsprechende Empfehlung für die Hardware und Software-Entwicklung.

[3] Leitfaden zur Anwendung des ISO 26262 für Halbleiter
[4] Aktuelle Veröffentlichung ist der DO-178C aus dem Jahr 2012.

COTS sind Komponenten entwickelt für verschiedene Kunden und unterschiedliche Anwendungen. Der Entwicklungsverlauf bzw. dessen Ergebnis bestimmt alleinig der Hersteller. Die Produkte sind an der Nachfrage am Markt und den akzeptierten Standards ausgerichtet. Der Endkunde hat keinen Einfluss auf den Entstehungsprozess und muss nach dessen Entwicklung nachweisen, ob die Anforderungen für seinen Anwendungsfall erfüllen werden. Mit dieser Tatsache wird in der Luftfahrtindustrie bereits seit Jahrzehnten erfolgreich umgegangen, was durch die im Einsatz befindlichen MPUs in Systemen bewiesen wird [Ye96].

Die heute anvisierten COTS-Komponenten sind komplexer, als die bereits im Einsatz befindlichen MPUs. Die entstandenen Erfahrungen bei der Entwicklung und Zulassung von COTS-basierter Avionik decken die Einschränkungen im praktischen Nutzen des DO-254 auf. Eine umfangreiche Offenlegung von Entwicklungsdaten durch den COTS-Hersteller ist nötig, widerspricht jedoch dem Schutz dieses Wissens. Versuche an dieses Wissen durch *Reengineering* zu gelangen, stellten sich als nicht praktikabel heraus [HB07].

Aus wirtschaftlichen Gründen streben Avionik-Hersteller keine Eigenentwicklung von zentralen Rechenkomponenten wie MPUs und MCUs an und verwenden stattdessen alternative Methoden. Diese sollen nachweisen, dass die COTS-Komponente den Anforderungen der Gesetzgebung und des Avionik-Systems entsprechen. Ein schadhaftes Fehlversagen muss extrem unwahrscheinlich sein. Dabei wird erwartet, dass die Komponente unter allen denkbaren Bedingungen wie beabsichtigt funktioniert und für unerwartete Ereignisse Maßnahmen ergriffen werden.

Aktuelle Ansätze über die Argumentation der Zulassbarkeit von COTS-Komponenten bauen auf einer Kombination folgender Bestandteile auf [Ce14], [Wi15]:

Vorhandene Entwurfsdaten Hat der Hersteller einen strukturierten und anforderungsbasierten Entwicklungsprozess verwendet, so können diese Daten – dessen Einverständnis vorausgesetzt – wiederverwendet werden.

Eigene Erzeugung von Entwurfsdaten Durch Reengineering-Maßnahmen können diese Daten auch nach der Entwicklung erhoben werden.

Betriebserfahrung Die Aussagekraft der Erfahrung kann die Produktreife untermauern, sodass keine systematischen Fehler bei einer ähnlichen Verwendung zum Tragen kommen.

Fehlermaskierung auf Systemebene Das Verhalten bei bestimmten Fehlerszenarien kann nicht in jedem Fall nachgewiesen werden. Durch Anpassung der Systemarchitektur (z.B. strukturelle Redundanz mit unterschiedlichen Komponenten, siehe Kapitel 3) können diese Fehler maskiert werden.

Detaillierte Entwurfsdaten sind notwendig, falls den Informationen zur COTS-Komponente nicht vertraut wird oder deren Aussagekraft nicht ausreicht. Damit die Betriebserfahrung glaubwürdig die Qualität einer Komponente unterstützt, muss diese aufwändig

ermittelt und bewertet werden. Dieses hier beschriebene Vorgehen der COTS-Nachweisführung ist anwendbar für MPUs oder MCUs mit niedriger Komplexität. Ein AMCU zählt nicht hierzu.

2.2 Zielsetzung des Qualitätsvergleichs

In Kapitel 2.1 werden Herausforderungen und aktuelle Lösungsansätze zur Zulassung von Avionik-Systemen beschrieben, wenn eine COTS-Hardware-Komponente eine zentrale Rolle im Systementwurf einnimmt. Die Zulassungsproblematik von COTS-Komponenten ist ein ständiger Wettlauf mit der fortschreitenden Technologie und ohne diese Produkte ein Innovationspotential künftiger sicherheitsrelevanter Avionik gehemmt wäre. Aktuell zulassungskonform entwickelte Komponenten beheben diesen Mangel nicht.

Bisher wurde die Vergleichbarkeit des Qualitätsniveaus zwischen den domänenspezifischen Standrads ISO 26262 Teil 5 und DO-254 nicht untersucht. Ähnliche erbrachte Vergleiche zwischen den Softwarestandards beider Domänen [GHW11], [Le12] geben dazu Anlass, diesen für Hardwarestandards mit einer zielgerichteten Verwertungsperspektive zu erbringen.

Ist das Qualitätsniveau zwischen den Domänen vergleichbar, werden folgende Auswirkungen erwartet:

- Die wiederkehrenden Kosten der Zulassung von COTS-basierter Avionik können reduziert werden. Diese Annahme beruht darauf, dass der Vergleich beider Standards generischer Natur (ohne Projektbezug) ist. So können diese einmal gewonnenen Erkenntnisse für eine Vielzahl von Avionik-Projekten wiederverwendet werden.
- Versuche der Einflussnahme der Luftfahrtindustrie auf die Entwicklung von COTS-Komponenten-Hersteller sind bekannt [CB04]. Ist den Herstellern bewusst, dass die notwendigen prozessspezifischen Anpassungen wirtschaftlich sind, werden sie umgesetzt. Das Risiko der Machbarkeit wird im Vorfeld dieser Forschung erbracht.

3 Sicherheitsbedenken aus dem Avionik-Sektor und Steigerung der Sicherheit

Wie bereits in Kapitel 2.1 beschrieben ist, liefert der DO-254 keine Hilfestellung für die zulassungskonforme Entwicklung von COTS-basierter Avionik. Um den Einsatz von COTS MCUs dennoch zu ermöglichen entstand durch die Federal Aviation Administration (FAA) zu diesem Thema in Zusammenarbeit mit namhaften Firmen aus dem Avionik-Bereich 2011 eine Forschungsarbeit die sich mit dieser Problematik befasst. Aus dem daraus resultierenden „Handbook for the Selection and Evaluation of Microprocessors for Airborne Systems“ [FA11] werden die Sicherheitsanforderungen behandelt die auch auf AMCUs angewendet werden müssen. In diesem Kapitel wird bewertet, ob diese von AMCUs erfüllt werden können. Die Beschreibung eines Redundanzkonzeptes ermöglicht die Erkennung und Toleranz weiterer Fehlerfälle.

3.1 Aktuelle Safety-Bedenken aus dem Avionik-Bereich

Im Rahmen der Untersuchungen seitens der FAA konnten insgesamt drei Hauptpunkte ermittelt werden, bei denen hinsichtlich des Einsatzes von AMCUs im Avionik-Bereich noch Sicherheitsbedenken bestehen.

Sichtbarkeit und Debugbarkeit AMCU-Hersteller erlauben in der Regel keinen Einblick in die internen Strukturen ihrer Produkte, da dieses Wissen als Firmengeheimnis angesehen wird. Dies hat den signifikanten Nachteil, dass sich die Hardware nicht mehr bis ins Detail analysieren lässt, um so exakte Vorhersagen hinsichtlich der Ausführungszeit treffen zu können. Durch die Integration der Systemarchitektur auf einem Chip wird es zudem unmöglich, gezielt zwischen den Komponenten Fehler einzuspeisen. Dadurch lassen sich die Sicherheits-Algorithmen nicht mehr, oder nur mit hohem Aufwand auf der Hardware selbst testen.

Konfigurationsprobleme Da es für einen AMCU-Hersteller nicht wirtschaftlich ist, für jeden Kunden eigene Produkte nach einem exakt vorgegebenen Funktionsumfang zu entwickeln, werden die Produkte für den breiten Markt konzipiert. Dieser beinhaltet einen Durchschnitt an Funktionen die in der entsprechenden Domäne üblicherweise benötigt werden. Da die Funktionen anwendungsspezifisch konfiguriert werden müssen, werden Software-Register zur Aktivierung bzw. Deaktivierung verwendet. Die Konfiguration der einzelnen Komponenten erfolgt dabei ebenfalls über in Software ansteuerbare Register. Bedingt durch Software-Fehler oder atmosphärische Einflüsse wie Single Event Upset (SEU) [5] können sich die Konfigurationen einzelner Register unbeabsichtigt ändern.

Gemeinsam genutzte Ressourcen Während die Anzahl der Komponenten innerhalb einer MCU immer weiter steigt, werden einige Komponenten weiterhin von mehreren Teilnehmern gleichzeitig benutzt. Hierzu zählt unter anderem der Hauptspeicher, der sowohl von Prozessoren als auch Direct Memory Access Controller (DMA-C) bedient wird. Hierbei kommt es, wie in Kapitel 4.1 beschrieben, zwangsweise zu Kollisionen, was starke Auswirkungen auf die Ausführungszeit haben kann.

3.2 Maßnahmen zur Steigerung der Safety auf Architekturebene

Ergänzend werden im Bericht der FAA Lösungsansätze behandelt. Hierbei soll durch Arbitrierungsverfahren für gemeinsam genutzte Ressourcen oder einem sogenannten Frame-Lock Ansatz [FA11] eine Entschärfung der Problematik bei den Ausführungszeiten erreicht werden. Trotz der vorgestellten Möglichkeiten kann durch das Fehlen von Entwurfsdaten nicht sichergestellt werden, dass alle Fehlerfälle abgedeckt sind. Hierzu werden zwingend weitere Maßnahmen auf Systemebene benötigt.

[5] Änderung von Werten/Zuständen innerhalb der MCU oder des Speichers durch geladene Teilchen aus der Atmosphäre.

Redundanz Der klassische Redundanzansatz in Form eines Triplex- [vN56] oder Quadruplex-Systems [AB10] bietet den größten Mehrgewinn bezogen auf die Sicherheit des Gesamtsystems. Hierdurch kann durch die Verschaltung mehrerer gleichartiger AMCUs, wie in Abb. 1 gezeigt, ein Fail-Operational-Verhalten erreicht werden – vorausgesetzt das Redundanznetzwerk selbst wurde entsprechend fehlertolerant ausgelegt.

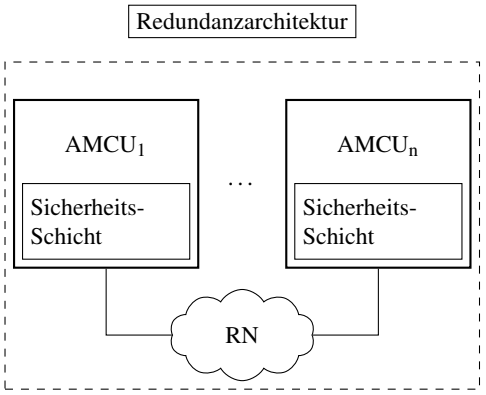


Abb. 1: Redundanzkonzept

Bezüglich der Robustheit gegenüber äußeren Einflüssen gibt es hier Seitens der FAA Vorschriften, welche im DO-160 [Sp10] konkretisiert sind. Für die Realisierung eines solchen Redundanznetzwerkes (in Abb. 1 als RN dargestellt) bietet es sich an, auf die standardmäßig vorhandenen Schnittstellen der verwendeten AMCUs zuzugreifen. Hierzu wurde bereits eine entsprechende Voranalyse in [HH16] durchgeführt. Durch den Aufbau eines solchen Netzwerkes können beispielsweise Konfigurationsprobleme toleriert werden, welche durch äußere Umwelteinflüsse ausgelöst wurden. Auch der komplette Ausfall eines MCU durch Überspannung oder Alterung, kann dadurch toleriert werden. Hierbei bringt es einen entsprechenden Mehrwert, wenn das System um den Aspekt der Hardware-Dissimilarität [Mo99] erweitert wird. Das aktuelle Ausschluss-Kriterium für solche Redundanzarchitekturen sind schlicht die Kosten (siehe [Gr15]), da alle Komponenten innerhalb des Steuergerätes mindestens dreimal vorhanden sein müssen. Andererseits wurden von der Automotive-Industrie bereits erste Bemühungen unternommen um mittels spezieller Duplex-Systeme und Rekonfigurationsmechanismen [Mu15] effizientere Lösungen erreichen zu können.

Dissimilarität Um die Auswirkung von Entwicklungsfehlern der AMCU ausschließen zu können, kann der Redundanzansatz mit verschiedenen MCUs und Intellectual Properties (IPs) verschiedener Hersteller verwendet werden. Der Grad einer hinreichenden Dissimilarität von MCUs muss dabei im Einzelfall geprüft werden.

Der dissimilare Ansatz führt jedoch zu einem signifikanten Anstieg der Entwicklungskosten, da eine Einarbeitung in mehrere MCUs erfolgen muss – zusätzlich zu Entwicklung und Wartung redundanter Software-Versionen für die verschiedenen MCUs.

4 Analyse von Zugriffsstrukturen

In Kapitel 3.2 wird dargestellt, dass einzelne MCUs den Safety-Anforderungen der Luftfahrt nicht genügen, wodurch ein redundanter Betrieb unerlässlich ist. Durch diese Forderung verschärfen sich die in Kapitel 3.1 genannten Bedenken über die Sichtbarkeit und Nutzung gemeinsamer Ressourcen. Die dadurch entstehenden hardwareseitigen Zugriffskonflikte können sich stark auf die Software auswirken, was wiederum zu einer Verletzung von zeitlichen Bedingungen führen kann.

Klassische Avionik-Systeme basieren aktuell auf diskreten CPUs bei denen der E/A-Bus und in vielen Fällen der System- bzw. Speicher-Bus offengelegt ist. Dies bedeutet nicht nur, dass das Busprotokoll (z.B. Arbitrierungsstrategie) zur Analyse offengelegt ist, sondern auch, dass mit externen Messgeräten die Kommunikation verifiziert werden kann.

Durch den Einsatz von hochintegrierten MCUs wird die Analyse des zeitlichen Systemverhaltens, sowie deren Einzelkomponenten und der Anwendungs-Software erheblich erschwert. Um Aussagen über die Zugriffspfade oder auftretender Interferenzen treffen zu können, muss auf integrierte Debug-Schnittstellen oder auf externe Messmethoden zurückgegriffen werden.

4.1 Allgemeine Problematik

In sicherheitskritischen Echtzeitsystemen, wie beispielsweise in Flugsteuerungen der Avionik, werden zyklische Regelschleifen verwendet. Neben dem Sensor-Input und der Aktor-Ansteuerungen werden bei hoch sicherheitskritischen Systemen mit mehrfach redundanter Hardware (vgl. Kapitel 3.2) zudem die Ein- und Ausgaben abgeglichen. In Abb. 2 wird eine typische Verarbeitungskette der Daten dargestellt. Die Ausführung der grau dargestellten Funktionen ist ein synchroner Prozess, für den sich die auftretenden Datenströme gut evaluieren lassen. Die Eingabe der angebundenen Sensoren hingegen läuft asynchron zur Regelschleife ab und steht somit in Konkurrenz zueinander.

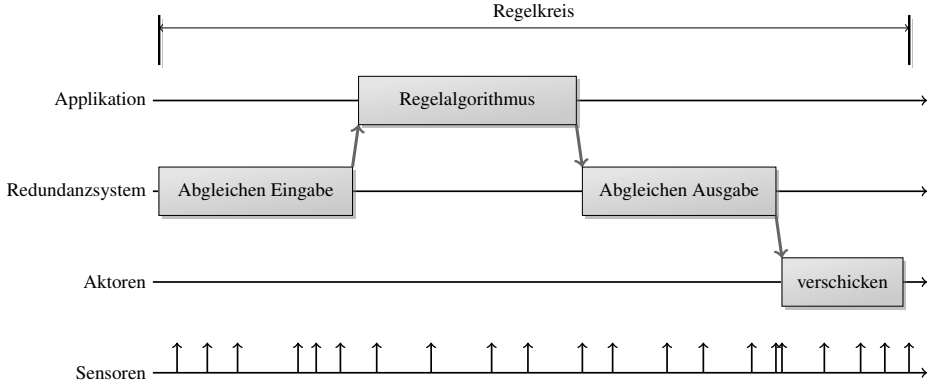
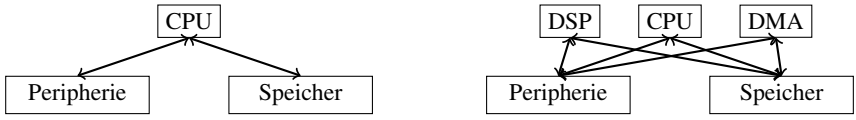


Abb. 2: Erweiterte Regelschleife

Um eine Analyse der Datenströme, insbesondere der Zugriffszeiten auf Register- und Speicherbereiche zu gewährleisten, ist in aktuellen Flugsteuerungsanwendungen das Einlesen der asynchronen Sensordaten ein Teil der (synchronen) Regelschleife. Dies stellt sicher, dass nur eine aktive Komponente (vgl. Abb. 3a) Zugriffe auf die verschiedenen Ressourcen verursacht. Dadurch entstehen keine Konflikte und die Zugriffszeiten auf Adress- und Registerbereiche lassen sich mit einem Maximum (Worst Case) angeben. Ein Nachteil dieser Implementierung ist, dass die Verarbeitung der Datenströme Ressourcen der CPU belegt.



(a) Konkurrenzfreier Zugriff (b) Konkurrierender Zugriff

Abb. 3: Zugriff auf Ressourcen

Diese einfache Datenverarbeitung lässt sich nur mit moderaten Datenraten und wenigen Schnittstellen realisieren. Durch den Anstieg der zu verarbeitenden Daten und abzudeckenden Schnittstellen[6] ist eine effizientere Verarbeitung der Eingabedaten nötig. Dies lässt sich durch den Einsatz spezialisierter Hardware wie DMA-C und digitaler Signalprozessors (Digital Signal Processor, DSPs) realisieren. Mittels eines selbstständigen Zugriffs auf Speicher und Peripherie sowie einer optionalen Datenverarbeitung kann die CPU entlastet werden.

Durch den Einsatz weiterer aktiver Komponenten (vgl. Abb. 3b) lässt sich ein kollisionsfreier Zugriff auf gemeinsame Ressourcen nicht mehr realisieren. Dies führt auch dazu, dass sich die Zugriffszeiten auf Speicher und Register nicht mehr verifizieren lassen. Es kann zwar immer noch eine obere Schranke angegeben werden, wobei diese nur vereinzelt oder in Extremsituationen auftritt und die Vorteile von Speicherdirektzugriff (Direct Memory Access, DMA) und DSP zunichtemacht.

4.2 Abschätzen von Datenströmen

Der pessimistische Ansatz, dass jeder Zugriff eine Kollision verursacht, tritt in modernen MCUs nicht immer auf. Wird von verschiedenen Komponenten auf Ressourcen im System zugegriffen, muss dadurch nicht immer ein Konflikt entstehen. Ein Beispiel hierfür ist die Verwendung eines Crossbar Switch als Verbindungsnetzwerk bei dem nur bei Zugriffen auf identische Zielressourcen Kollisionen auftreten.

[6] Implementierung von Redundanzsystemen mithilfe von on-Chip Schnittstellen (vgl. Kapitel 3.2), Radar- oder Kamera-basierte Fahrerassistenzsystemen (Advanced Driver Assistance Systems, ADAS), etc.

4.2.1 Analyse möglicher Kollisionspfade

Um einen Überblick über die möglichen Kollisionen und deren Eigenschaften zu erhalten lassen sich gezielte Messungen durchführen. Dafür werden einzelne Pfade gegeneinander vermessen und die daraus resultierenden Interferenzen festgehalten.

Als Testaufbau eignet sich ein minimales Programm, welches CPU-getrieben nur einen vorher festgelegten GPIO-Pin zyklisch umschaltet. Hierbei handelt es sich um eine kompakte Befehlsfolge, der im CPU-Cache gehalten wird. Ein Nachladen von Instruktionen aus dem Hauptspeicher ist nicht nötig. Neben den Zugriffen auf den GPIO-Pin werden asynchrone Transfers mithilfe des DMA-C induziert, um Daten beispielsweise von einem UART-Register in den SRAM zu übertragen.

Anhand des Datenbuchs (im aktuellen Beispiel wird ein TI Hercules TMS570LC4357 verwendet, vgl. Reference Manual [Te14]) lässt sich feststellen, dass sowohl das GPIO- als auch das UART-Register einen gemeinsamen Bus verwenden. Treten nun gleichzeitig Transferanfragen auf, so sind hier Kollisionen zu erwarten. Diese lassen sich mit Oszilloskop oder Logic Analyzer aufzeigen, sodass dadurch die daraus resultierende Wartezeit der CPU ausgemessen werden kann. Diese Analyseschritte müssen für jeden möglichen Pfad innerhalb des MCU wiederholt werden.

Neben externen Messmethoden lassen sich mithilfe von Tracing-Schnittstellen Informationen über das System herausführen um die internen Abläufe besser analysieren zu können. Hier kann es jedoch, je nach Implementierung der Tracing-Schnittstelle, zu Problemen kommen. Einige Implementierungen übergeben die Ereignisse, ohne einen internen Zeitstempel zu setzen, an einen Zwischenspeicher. Die gepufferten Nachrichten werden erst durch das Auftreten bestimmter Ereignisse durch den MCU verschickt, wodurch die externen Tracing-Hardware keine exakten Zeitstempel generieren kann. Im Mittel stimmen die interpolierte Ausführungszeiten je Instruktion, jedoch zeigt dies, dass hier der Fokus klar auf den durchschnittlichen Performance-Werten liegt und nicht auf die sicherheitsrelevante Worst Case Execution Time (WCET).

4.2.2 Analyse von Zugriffsmustern

Neben dem Wissen über mögliche Kollisionen sind für Timing-Analysen die tatsächlichen Zugriffsmuster von Applikationen notwendig. Hierunter wird die zeitliche Abfolge von Zugriffen auf bestimmte Adressbereiche verstanden. Ebenfalls können zyklisch wiederkehrende Zugriffsmuster (vgl. Regelschleife, Abb. 2) die Analyse erleichtert, da nur auf einem zeitlich begrenzten Muster Analysen erbracht werden müssen.

Neben den Zugriffsmustern der Applikation lassen sich zudem auch Zugriffsmuster seitens der E/A ermitteln. Können hier keine Muster abgeleitet werden, so lassen sich mit Hilfe der maximalen physikalischen Übertragungsrate einzelner Schnittstellen die Zugriffsraten ermitteln. Es kann davon ausgegangen werden, dass innerhalb der Zeitspanne einer Nachricht maximal eine definierte Anzahl an Unterbrechungen initiiert werden kann.

4.2.3 Abschätzen von Konflikten

Kombiniert man das Wissen aus der Analyse der Zugriffsmuster der Software und aus denen der Schnittstellen, so lassen sich die theoretische Anzahl an maximalen Konflikten errechnen. Dies reduziert die Überschätzung der WCET erheblich, da nur die maximal auftretenden Konflikte berechnet werden und nicht für jede Instruktion von einer Vielzahl von Kollisionen ausgegangen werden muss.

5 Schluss

Bei der Entwicklung von funktional sicheren Systemen wechselten sich die Luftfahrt- und Automobilindustrie, ähnlich eines Staffellaufs, chronologisch bei der Weiterentwicklung von Technologien und Methoden ab. So wie vor der Jahrtausendwende Fly-By-Wire-Systeme entstanden sind, die keiner mechanischen Absicherung bedürfen [Ye96], so hat die Automobilindustrie durch ihre Marktmacht den Sicherheitsgedanken bis zum Hardware-Komponenten-Hersteller, die zu AMCUs geführt haben, transportieren können.

Dieser Forschungsbeitrag versteht sich als Fortsetzung dieses Staffellaufes. Es werden Untersuchungen beschrieben die u. a. notwendig sind um AMCUs in sicherheitskritischen Avionik-Systemen, mit dem Anspruch der Zulassbarkeit, einsetzen zu können. Indem die Luftfahrtindustrie den Stab weiterträgt, werden ebenso neue Impulse für den Automobilmarkt gesetzt. Diese können einen Mehrwert für neue sicherheitsrelevante Automotive-Systeme im Bereich des autonomen Fahrens darstellen.

Danksagung

Diese Veröffentlichung wird unterstützt durch:

- das Projekt FORMUS^{3}IC “Multi-Core safe and software-intensive Systems Improvement Community”, Förderkennzeichen AZ-1165-15, der Bayerische Forschungsstiftung,
- das Open Innovation for RPAS (OPIRA) Projekt, finanziert durch das Luftfahrtforschungsprogramm V (LuFo V5-1), Förderlinie “Technologie” des Bundesministeriums für Wirtschaft und Energie und
- der von Airbus Defense and Space finanzierten Stiftungsprofessur “Systemtechnik für sicherheitskritische Software”, unterstützt durch den “Stifterverband für die Deutsche Wissenschaft e.V.”

Literaturverzeichnis

[AB10] Audsley, N. C.; Burke, M.: Distributed Fault-Tolerant Avionic Systems – A Real-Time Perspective. 2010.

[CB04] Cole, P.; Beeb, M.: Safe COTS graphics solutions: impact of DO-254 on the use of COTS graphics devices for avionics. In: The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576). Institute of Electrical and Electronics Engineers (IEEE), 2004.

[Ce14] Certification Authorities Software Team: Compliance to RTCA DO-254/ EUROCAE ED-80, "Design Assurance Guidance for Airborne Electronic Hardware", for COTS Intellectual Properties Used in Programmable Logic Devices. Bericht 33, Federal Aviation Administration, August 2014.

[FA11] FAA: Handbook for the Selection and Evaluation of Microprocessors for Airborne Systems. Bericht, 2011.

[FK06] Forsberg, Hakan; Karlsson, Kristoffer: COTS CPU Selection Guidelines for Safety-Critical Applications. In: 25TH Digital Avionics Systems Conference. Institute of Electrical and Electronics Engineers (IEEE), oct 2006.

[GHW11] Gerlach, Matthias; Hilbrich, Robert; Weißleder, Stephan: Can cars fly? from avionics to automotive: Comparability of domain specific safety standards. In: Proceedings of the Embedded World Conference. 2011.

[Gr15] Grave, Rudolf: Autonomous Driving – From Fail-Safe to Fail-Operational Systems. TechDay December2015, 2015.

[HB07] Hilderman, Vance; Baghi, Tony: Avionics certification: A complete guide to DO-178 (software), DO-254 (hardware). Avionics Communications, Leesburg, VA, 2007.

[HH16] Hiergeist, Sebastian; Holzapfel, Florian: Fault-tolerant FCC Architecture for future UAV systems based on COTS SoC. 2016.

[Le12] Ledinot, Emmanuel; Gassino, Jean; Blanquart, Jean-Paul; Boulanger, Jean-Louis; Quéré, Philippe; Ricque, Bertrand: A cross-domain comparison of software development assurance standards. ERTS, 2012.

[Mo99] Montenegro, Sergio: Sichere und fehlertolerante Steuerungen: Entwicklung sicherheitsrelevanter Systeme. Hanser, München [u.a.], 1999.

[Mu15] Much, Alexander: The Safe State: Design Patterns and Degradation Mechanisms for Fail-Operational Systems. safetronic.2015, 2015.

[SA10] SAE Aerospace: Guidelines for Development of Civil Aircraft and Systems. Bericht ARP4754, SAE International, Dezember 2010.

[Sp82] Special C. of RTCA: DO-178: Software Considerations in Airborne Systems and Equipment Certification. RTCA, Dezember 1982.

[Sp00] Special C. of RTCA: DO-254, Design Assurance Guidance for Airborne Electronic Hardware. RTCA, April 2000.

[Sp10] Special C. of RTCA: DO-160G: Environmental Conditions and Test Procedures for Airborne Equipment. RTCA, 2010.

[Te09] Technical Committee 22: ISO/DIS 26262 - Road vehicles – Functional safety. Bericht, International Organization for Standardization, Geneva, Switzerland, Juli 2009.

[Te14] Texas Instruments: . TMS570LC43x 16/32-Bit RISC Flash Microcontroller. Texas Instruments, Mai 2014.

[vN56] von Neumann, J.: Probabilistic Logics and Synthesis of Reliable Organisms from Unreliable Components. 1956.

[Vo13] Volpe National Transportation Systems Center: Unmanned Aircraft System (UAS) Service Demand 2015-2035: Literature Review and Projections of Future Usage. Bericht DOT-VNTSC-DoD-13-01, U.S. Department of Transportation, 2013.

[Wi15] Wilkinson, Chris: Obsolescence and Life Cycle Management for Avionics. Bericht, Federal Aviation Administration, November 2015.

[Ye96] Yeh, Y.C.: Triple-triple redundant 777 primary flight computer. In: IEEE Aerospace Applications Conference. Proceedings. Institute of Electrical and Electronics Engineers (IEEE), 1996.

Risk-Oriented Security Engineering

Christof Ebert and Dominik Lieckfeldt [1]

Abstract: Virtually every connected system will be attacked sooner or later. A 100% secure solution is not feasible. Therefore, advanced risk assessment and mitigation is the order of the day. Risk-oriented security engineering for automotive systems helps in both designing for robust systems as well as effective mitigation upon attacks or exploits of vulnerabilities. Security must be integrated early in the design phase of a vehicle to understand the threats and risks to car functions. The security analysis provides requirements and test vectors and adequate measures can be derived for balanced costs and efforts. The results are useful in the partitioning phase when functionality is distributed to ECUs and networks. We will show with concrete examples how risk-oriented cyber security can be successfully achieved in automotive systems. Three levers for automotive security are addressed: (1) Product, i.e., designing for security for components and the system, (2) Process, i.e., implementing cyber security concepts in the development process and (3) Field, i.e., ensuring security concepts are applied during service activities and effective during regular operations.

Keywords: Cyber Security, Safety, embedded systems, quality requirements, risk management, validation

1 Introduction

1.1 Automotive Connectivity and Cyber-Security

More than 20 years ago, the invention of the CAN bus built the basis of connectivity. In the beginning, only two to three ECUs (electronic control units) were connected. But nowadays we have complex networks of sensors and actors with different bus systems like CAN, LIN, FlexRay, MOST or Ethernet. The interaction of functions in this distributed network is an essential part for our today's modern cars with all features for safety and comfort.

Besides the further development of innovative sensors like radar and camera systems and the analysis of the signals in highly complex ECU systems, the connected cars will be a driving factor for tomorrow‘s innovation. Internet connections will not only provide the need for information to the passenger. Functions like eCall or communication between cars or car to infrastructure (car2x) shows high potential to revolution the individual traffic. This includes the improvement of the traffic flow controlled by intelligent traffic lights, warnings from roadside stations or brake indication of adjacent cars. This builds the basis for enhanced driver assistant systems and automated driving. But the connection to the outer world bears also the risk for attacks to the car (Fig. 1).

[1] Vector Consulting Services, Ingersheimer Straße 24, D-70499 Stuttgart, E-Mail: Christof.Ebert@vector.com

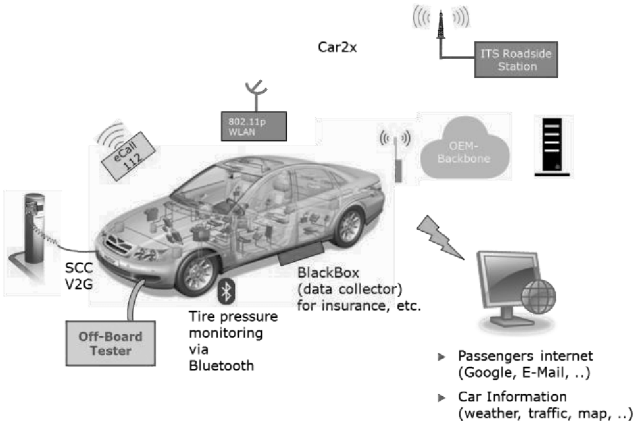


Fig. 1: Car with remote connections

The picture above shows several car connections that are already available today or will come up soon in the near future. Each connection to the car has a potential risk for an attack, regardless if it is wireless or wired. Just the threat is different. The access through a connector is only possible for a limited amount of cars, whereas a far field connection can be accessed from anywhere in the world. But also near field connections play an important role, such as tire pressure monitoring system, Bluetooth and wireless LAN. Security and reliability of these connections will be essential for the acceptance and success of these systems. With the introduction of this technology precautions must be taken to increase the reliability and to reduce the vulnerability to the system.

Obviously with growing connectivity functional safety needs security. Based on the specific challenges of automotive security, OEMs and suppliers have to realize an effective protection against manipulations of automotive E/E systems [EB2016]. Key points in the development of protected E/E systems are the proper identification of security requirements, the systematic realization of security functions, and a security validation to demonstrate that security requirements have been met. The following items need to be considered to achieve security in the car development process:

- Standardized process models for a systematic approach which is anchored in the complete development process. This starts on the requirements analysis through the design and development down to the test of components and the network.
- Quick software updates to close vulnerabilities in ECU software.
- Reliable protocols that are state-of-the-art and meet long-term security demands. Related to security this is often combined with cryptographic keys. So, a key management over the lifecycle of the vehicle must be maintained.

- In-vehicle networks and system architecture that provides flexibility and scalability and are designed under consideration of security aspects.

Based on our experiences in several client projects, we show which security engineering activities are required to create secure systems and how these activities can be performed efficiently in the automotive domain [EB2016]. In the following we want to take a view on each of these topics, what are the current activities, but also want to provide hints on how to mitigate the security risks.

1.2 Safety and Security

Night drive on the highway. The display suddenly flashes and the loudspeakers transmit a loud and painful sound. The driver is highly disturbed and tries to stop this annoyance. In doing so he is losing control over the car and causes an accident. Mere fiction? Not really. Continuously growing complexity within the electrical subsystems of the car, their interconnection by a variety of bus systems, and the use of standard components with open interfaces make networked systems within the car increasingly vulnerable. Such risks demand strong protection on various levels along the entire life-cycle of components and of the entire vehicle. Looking to past experiences with insufficient security in other domains, it is obvious that automotive security will determine which suppliers and electronic platforms (e.g. AUTOSAR) will capture the market for standard components. And it will determine how fast further communication systems (e.g., telematics with internet access) will be accepted by customers and policy makers.

What is cyber-security? Basically, security is a quality attribute which heavily interacts with other such attributes, such as availability, safety or robustness. Security is the sum of all attributes of an information system or product which contributes towards ensuring that processing, storing and communicating of information sufficiently protects integrity, availability and trust. Security implies that the product will not do anything with the processed or managed information which is not explicitly intended by its specification. If for instance the classic definition of a functional requirement meant that the car can be started by turning the key, but would also allow a variety of mechanisms to start it otherwise, maybe for diagnostic or repair services (who was not in such situation that he needed support on the road and the person would open the trunk and start the engine directly?). Security implies that the car cannot be started except for the defined scenarios and is therefore protected against theft or misbehaviors. It's growing relevance comes from the simple observation that by defining functionalities alone, there is nothing said about the correlation of features, specifically if one of the many components of the car malfunctions. Many drivers of cars of the first generation of highly interconnected electrical control units distributed across the car will recall strange behaviors, such as windows which would open when switching on the radio. Automotive security has to ensure that any such malfunction or misuse case will not happen.

There is a big difference when we contrast safety and security. Safety is built upon reliability theory and looks into statistical malfunctions of components with small

probabilities and how they will impact functionality. Security on the other hand has to deal with the worst cases with a probability of one because once known, they will be exploited. One might argue that safety is about criticality for the life and health of the system's user, while security is only about annoyances. It is however obvious that within a safety-critical system, such as a car, security meets safety because malfunctions can interact and cause disturbances that can result in accidents, as described in our introduction.

Vulnerability scenarios within cars have been changing fast over the past years. The increasing interconnection on different architectural layers (e.g., electrical control units, software components, configurations and their changes, communication inside and outside the car, diagnosis, telematics) has caused a level of complexity that was unknown so far. It is a mere question of time until the resulting loopholes and weaknesses are identified and abused. It was showed already that widely used automotive bus systems such as CAN and FlexRay can be brought from the outside – by connecting a device to any point of such bus systems – into overload conditions which will eventually cause malfunctions [EB2016].

State of the art communication systems increasingly offer open interfaces (e.g., DVDs, E-Mails, USB, Bluetooth, IP-based diagnostics) that allow to inject viruses and Trojan horses to the respective embedded operating systems. Also, defective code and configuration settings can create new and unknown vulnerabilities as we are used to from many information systems.

This is what drives the security attacks in our illustrative case study from the beginning of this article. Fig. 2 illustrates the primary sequence of incidents that caused the flashing display and the loudspeakers or head unit to transmit such loud signals.

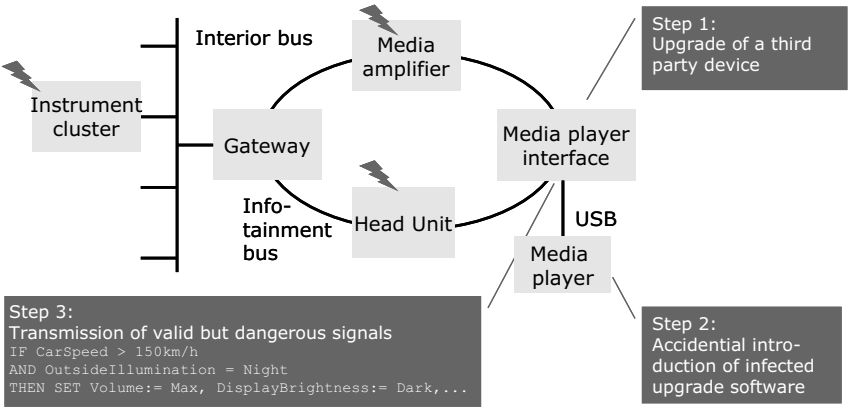


Fig. 2: „Night Drive" – How could it happen?

As so often in security attacks, the first step was just a normal upgrade of the multimedia equipment with a better device. Needless to say that it was not delivered and installed by the OEM, but came through an internet delivery for perceived cost reasons and enhanced functionality. In a second step infected software came into the multimedia devices, probably via an infected USB stick or from a media file. It could well be that software upgrades to one of the media devices also brought this infected software into the system. From here onwards it was just normal cause and effect, namely transmission of valid but dangerous signals on the infotainment bus which were triggered by listening to signals with speed and outside illumination. These signals are almost omnipresent in car networks due to many dependencies on these factors.

The different reasons for insufficient automotive security are illustrated with examples in Fig. 3. They are distinguished according to the different scenarios that cause the vulnerability or security problem within the product (e.g., functionality, architecture, configurations), in the process (e.g., design, development, validation, stakeholder communication), and in the field (e.g., maintenance, enhancements, diagnosis).

All these scenarios result from unawareness of security needs and security technology, be it by ignorance ("this won't matter in cars because all our critical electrical units are protected by cryptography"), arrogance ("security matters only in information systems") or naivety ("we have verified all requirements and components according to our established test strategy"). An overall security strategy is mostly missing.

Causes	Product / architecture	Development process	After-sales / field
By ignorance	- security is insufficiently supported by architecture - components are individually verified but not the system in which they operate	- missing security requirements - no abuse and misuse scenarios - insufficient verification during the entire component and system life-cycle - inadequate checklists and design guidelines to design for security	- changes and modifications are not sufficiently validated against security requirements and abuse / misuse scenarios - security impacts of changing a component or introducing a new version or variant of a software / hardware component are not analyzed
By arrogance	- experiences from other products, domains and markets (e.g., IT, telecommunications) are not considered - security is designed only on the basis of firewalls, gateways and protected components	- inadequate or missing communication between automotive electrical engineers and IT security experts - development processes specifically on system level without security requirements and checks	- new software releases are introduced during production and after-sales without considering security checks and qualifications
By naivety	- insufficient training on security for software and systems engineering - unknown misuse / abuse scenarios - automotive bus systems are active even when ignition is off	- unknown state of the practice for security verification tools and test methods - missing requirements and criteria for security - non-proprietary or open components are introduced without verifying security criteria on system level	- software components or media in use which have not been qualified - after-sales devices and components are integrated to the car without assessing the impacts on other subsystems such as network overload

Fig. 3: Causes for security vulnerabilities and issues in the life-cycle of a car

Typically, components are individually protected such as encrypted flashware for an engine controller. Critical functionality such as engine management, theft protection or engine diagnosis is hardened and verified. Increasingly secure networks and architectures are discussed and will certainly influence the design of cars ten years from now. Safety has received a lot of focus recently in automotive engineering and qualification of components and systems, such as processes to ensure proper handling and engineering according to SIL-levels. But safety and associated design rules are insufficient as we have learned before. They look to faults and their probabilities, while security has to deal with the worst case in scenarios where a probability is replaced by the willingness of the attacker to cause the worst possible damage.

Two aspects related to security in embedded systems have to be considered: (1) Attack scenarios go well beyond individual components and functions. (2) While safety deals with avoiding critical failure modes, security has to cope with intelligently introduced causes of faults, which is far more difficult, given that the attackers' intelligence, willingness, determinedness, and creativity often exceed that of the engineers looking to a problem from the – different – perspective of how to solve it, and not how to find loopholes and strange feature correlations.

Telecommunication and information systems have realized several years ago that isolated mechanisms (e.g., distributed functionality in proprietary subsystems, protection on component-level, gateways and firewalls between components, validation of critical functions) are insufficient. This article underlines together with concrete examples how automotive security can be achieved. We will take the three different perspectives that were introduced in Fig. 3, namely (1) the product and its architecture (e.g., specification of security requirements, misuse and abuse cases, vulnerability analysis, inherently secure architectures); (2) engineering for security during the development process (e.g., FMEA and hazard analysis as a basis for security, protection on component- and on system-level, systematic verification, code analysis, validation on product-level); and (3) the relevant after-sales activities in the field (e.g., fault analysis, patch and correction handling, emergency response and handling, distribution of corrections and protective mechanisms).

Security and related measures demand well-founded concepts all along the life-cycle of both components and the car itself, especially if their effectiveness has to be proven at a later point due to legal actions. With this article we strive not only to provide guidance for specific misuse cases but to change the mentality of engineers of embedded systems towards designing for security – rather than for functionality.

1.3 Risk-oriented Security

Developing secure software is challenging for several reasons, namely because increasingly systems are connected, most software is developed in a global context in heterogeneous teams with various skills, systems complexity is exploding with embedded and IT systems converging such as IoT, and both budget and cycle times are

continuously decreasing. For instance a modern car has almost hundred embedded microcontrollers on board and is connected over several external interfaces to a variety of cloud technologies. At the same time cyber-attacks and vulnerabilities are increasing. Therefore, software technology and the underlying security engineering have to be constantly improved.

While there is a movement towards better understanding security from the ground up, many of the existing approaches in managing security have been focused around encryption, developing malware software, and to detect attacks to networks and systems. Existing methods and tools are limited by large number of false positives and inability to consistently trace such issues to the root causes. In this article we will particular draw attention to all aspects of security from specification to design and life-cycle support.

Over the past decade trends like connected car and driver assistance systems among others have led to software and connectivity playing an increasingly important part in developing vehicles and also for business models of OEMs and suppliers likewise.

Devastating impact of security issues is already known from industrial sectors like IT-infrastructure, aviation, information technology and telecommunications, industrial control systems and energy and financial payments. Virtually every connected system will be attacked sooner or later. A 100% secure solution is not feasible. Therefore, advanced risk assessment and mitigation is necessary to protect assets. Consequently, the typical solution to security in these industries relies on suitable risk assessment that projects threats on assets of interests. Thereby cost of implementing specific security measures can be compared with the probability of a particular threat that they counter.

Asset-based risk assessment is a suitable tool for companies to steer efforts for security engineering in a systematic and comprehensive way and thereby involve all relevant stakeholders in the organization. For example, a CEO may not find it very helpful to have a long exhaustive list with every attack vector or potential threat – they need to be provided with a ranked listing and useful decision-support tools which clearly shows alternatives and consequences. From the view of an automotive system developer, a flat listing of potential threats might not help to improve the system. To really help, they need to be able to map security threats, countermeasures and requirements to system/architecture elements in their scope of the project.

The systematic management of security threats and associated security goals is essential to actually providing safe and competitive products, and to protect valuable assets and business models.

But what makes security engineering so complex? Automotive developers face the challenge of securing a system against attackers whose capabilities and intentions are at best partially known. Some attacks might today appear infeasible, but todays impossible attacks might become more likely in the near future. An example of this is attacking a vehicle simply by exploiting wireless interfaces, 20 years ago would have been extremely unlikely, however today a cheap software defined radio and accomplish these

types of attacks with little effort. On the other hand, an attacker might invest more effort into launching an attack the more valuable a successful attack is to him. Some attacks represent more effort to the attacker than others given the specific potential of the attacker. It is this risk/reward payoff that is analyzed in security engineering. Likewise during testing and verification, suitable methods to verify that the vehicle has the required security level and process goals like, test strategy and coverage, need to be chosen.

Furthermore, the assets to be protected from attacks are decided by stakeholders involved, e.g. drivers would indicate different assets of their vehicle to be protected compared with what an automotive developer considers an asset. However, customers/drivers need to be satisfied with their vehicle in order to buy another one from the same company. Consequently, security engineering must seek tradeoffs between cost of security measures and benefit to assets in order to make sustainable decisions.

Security concepts must balance the cost of not having enough security and thus being successful attacked with all damaging consequences and the cost spent to implement appropriate security mechanisms and keep them updated along the life-cycle of the car – well beyond end of production. We therefore introduce here a strict risk-oriented approach to security (Fig. 4).

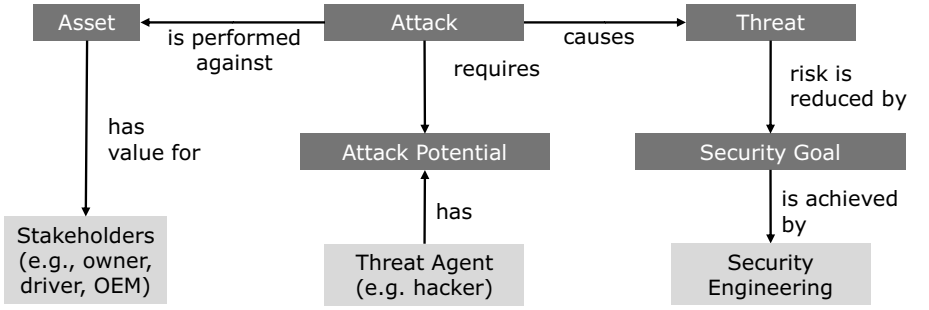


Fig. 4: Overview of risk-oriented cyber-security analysis process with the major steps of asset determination, attack potentials and security goals derived from threats

To summarize, the relationship between assets, attackers and threats is complex and dynamic (e.g. attacks are more probable the less effort is required and the more value successful attacks represent; attack vectors and effort change over time). Furthermore, common understanding of assets among all stakeholders of security engineering is mandatory in order to provide information for steering the security engineering.

Choosing the right set of security engineering methods for analysis, concept and testing is challenging but required in order to enable goal-oriented and manageable security engineering.

Risk-based Security Engineering combines state-of-the-art methods for automotive security risk assessment in a practical framework and supports all involved stakeholders to develop "secure-enough" products. The method and our approach for proposing a concrete technical security concept is based upon security best practices such as:

- SAE J3061 (Automotive cyber-security) being the first standard on the topic of automotive cyber-security but also being aware that it primarily enriches ISO 26262 towards security for functional safety [SAE2016].
- ISO 15408 (Evaluation criteria for IT security) with its focus on IT systems, specifically the 7 evaluation assurance levels (EAL) for security requirements and guidance on common criteria a standardized practice translated by Vector to automotive common criteria.
- ISO 27001 (Information security management systems) with its governance requirements for security engineering across the entire value chain.
- IEC 62443 (Industrial communication network security) with its strong view on distributed systems and necessary security technologies and governance.
- ISO 26262 (Automotive functional safety) using its clear focus on automotive electronic systems with good coverage of entire life-cycle; revision in 2016 [ISO2017].
- IEC 61508 (Functional safety for electronic systems) while being aware that it is only a high-level functional safety guidance for electronic systems.

Our Vector Security Check and underlying security engineering methods have adopted the state of the practice in security evaluation and proposed mitigation [EB2016]. It is using significant research work from our worldwide security projects. It also uses external best practices, such as "E-safety vehicle intrusion protected applications" (EVITA) funded by European Union [Ev2017], HEAVENS [Is2014], and other proposed methods for security risk assessment in automotive development [Se2017, Si2017, Pr2017, ETSI2010].

We will furtheron show by examples how to use the risk-oriented security concept covering the entire security life-cycle with focus on the upper left activities, namely

- Asset Definition and Threat and Risk analysis
- Security Goals
- Security Concept

1.4 Related Work in Automotive Security

The automotive industry is already engaged in security topics since several years. Several (EU-) funded projects had been launched for researches on Car2x. In the

following a few of them will be presented. The SEVECOM project (www.sevecom.org, [Se2017]) has analyzed risks and threats and has defined first general security architecture. A notably project for security was EVITA ("E-safety vehicle intrusion protected applications", www.evita-project.org, [Ev2017]; Fig. 5). The main objectives were to design a secure on-board network and the definition of building blocks to protect security relevant components and data inside a vehicle. One of the major outcomes was the definition of a hardware security module, defined in three versions: light, medium and full. Each version requires at least a hardware acceleration for data encryption (AES), secure key storage and a secure boot. These requirements show equivalences to the SHE (Secured Hardware Module) defined by the HIS (Hersteller Initiative Software, www.automotive-his.de).

HSM	EVITA full	EVITA medium	EVITA light
Internal NVM	Yes	Yes	Optional
Internal CPU	Programmable	Programmable	None
HW crypto algorithms (incl. key generation)	ECDSA, ECDH, AES/MAC, WHIRLPOOL/HMAC	AES/MAC, Key storage, Microcontroller.	AES/MAC
HW crypto acceleration	ECC, AES, WHIRLPOOL	AES	AES
RNG	TRNG	TRNG	PRNG w/ ext. seed
Counter	16x64bit	16x64bit	None
Intended use-case	C2x,...	Gateway, engine control, head unit,...	Sensors, actuators, ...

Fig. 5: EVITA classification for the hardware security module (HSM)

In-field tests for a Car2x communication was made in the Sim project (www.simtd.de, [Si2017]). In the area of Frankfurt, a field test was established with more than 100 test vehicles. Highways, country roads and city traffics were equipped with infrastructure to communicate with. A currently active project is PRESERVE (http://www.preserve-project.eu, [Pr2017]). The main objective is the design of security architecture for vehicle-to-infrastructure (V2x), to setup and test the system. This shall be achieved by setting up a fully operating security subsystem in a real environment with consideration of cost and performance. This includes also a further hardware environment with adequate performance.

These activities are important preconditions for a secure communication that is standardized in the Car2x area. This is essential for interoperability between cars from different car manufacturers and beyond national boundaries. In Europe there is the CAR2CAR consortium working on standards for vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I) and a cooperative intelligent transport system (C-ITS).

2 Security analysis

Security in a complex system cannot be achieved by applying countermeasures on single items. It requires an analysis of the complete functionality or system as a whole and to apply countermeasures as an integral part. First, you need to identify what are the assets I want to protect. Besides financial aspects also confidentiality and, especially for the automotive industry, safety functions must be considered carefully. The next step would be a threat analysis: who has access to my assets, what are potential attackers and where are my access points. A typical approach to this is the construction of a data flow diagram in which the assets are identified. It provides an overview of all connections and access points, where attacks and manipulations can be achieved. From the material above a risk assessment can be done to obtain the measurements and results in a classification of the risk. An example of such a risk assessment can be found in the picture below. Here, as an example, the classification was defined in three categories: Low, medium and high (Fig. 6).

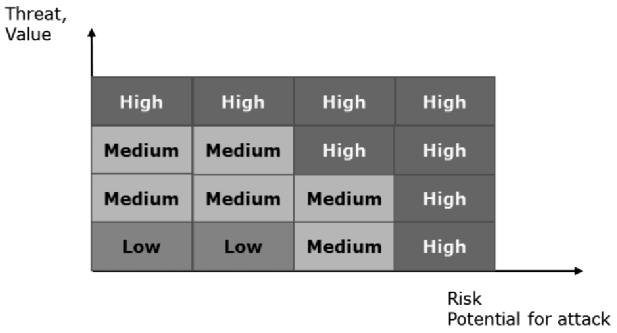


Fig. 6: Definition of security level derived from threat analysis and risk assessment.

This process provides systematic means to deal with the subject and results in a balanced trade-off for cost and efforts. Depending on the determined security level countermeasures can be defined on system level and further derived as security input requirements for ECUs. The analysis phase provides now also requirements for hardware extensions of the ECU, e.g. if hardware acceleration is needed for authentication or if a specific key management is required for higher security measures. The requirements are also an input to define test vectors on functional (for an ECU) and system level (for the vehicle). These tests, together with standard penetration tests, then will help to provide evidence for successful application of the security to the function and system.

3 In-vehicle security

The IT industry deals already since years with strategies for data protection and to

provide secured networks to prevent them against unauthorized access. Wide experiences are available here, that, with special considerations, can be adapted and are useful for the automotive industry as well. Similar activities can be seen such as the adaptation of the Ethernet when BroadR-Reach was introduced to the automotive area. This allows also taking over the proven software architecture of Ethernet, so that a number of approved protocols are available as well for a secured data transmission. Essentially, they are based on cryptography, software algorithms based on more or less complex mathematics. The algorithms itself are not the secret and are available to the public but keys provide the secret and they must be created, distributed and maintained carefully. A popular key management system used by the IT industry is the PKI (Public Key Infrastructure). It contains a hierarchical certificate management with associated keys and builds the basis for an authenticated communication between partners.

3.1 Security Engineering

While security requirements are concerned with what has to be protected, security engineering defines how the protection is realized. It affects all activities that are associated with "normal" engineering, such as system design, software construction, tests and after sales. We regard each of these activities in the following paragraphs.

In a recent client project, we had to identify and prioritize security requirements for an auto-motive E/E component. Based on the size and complexity of the component development project and the given capacity for security engineering, we selected an agile approach [EB2016] that was conducted in form of several workshops. The client's engineers provided expertise on the component's functions and their implementation. We moderated the workshop and provided expertise in security requirements analysis as well as knowledge on the used technologies' vulnerabilities and sensible protection mechanisms.

For better understanding we will show some hands-on examples from current security projects:

- Adapt the development processes to factor in security engineering activities. Security engineering activities are known, scheduled, and executed smoothly within the "normal" development, not in an ad hoc way. Security is considered from the beginning on through the complete project. Additionally, synergies can be exploited (e.g. a configuration management process can prevent quick fixes that have not been tested against security vulnerabilities).
- Systematically elicit security requirements. Elements that have to be protected are known from the beginning on, allowing for stringent realization of their security. Additionally, security requirements can be used to deduce test cases for security validation.

- Thoroughly review or test any security relevant arte-fact. Reviews of security engineering artefacts such as security requirements and security concept as well as simulations of security functionalities and code analyses allow for the identification of vulnerabilities at the earliest possible time.
- Use analysis and test tools. Automated tools reduce effort and allow for efficient and comprehensive analysis and (regression) testing. For instance the Vector PREEvision PLM and modelling environment provides a strong collaborative engineering backbone for ensuring application of above measures along the life-cycle.
- Manage embedded security competencies. Many activities of security engineering require a specific embedded security expertise, e.g. identification of vulnerabilities, design of the security architecture, secure implementation, performance of security tests, and review of security-related work products. Without this expertise, effective security engineering is near to impossible. Therefore, build up embedded security competence in your organizational unit or obtain it from internal or external providers.

We do not claim that these are the only valid solutions. Depending on e.g. corporate culture, existing experiences, and project size and complexity, other solutions may be preferred. Independent of the approach used, we noticed several activities that benefitted the introduction and the performance of security engineering in general [EB2016].

3.2 Software update and maintenance

To enable efficient after sales activities in spite of constraining security mechanisms, several aspects need to be addressed. How can software updates be performed in the field with both security against unauthorized manipulations and justifiable logistical effort? The association of German car manufacturers (HIS) has created specifications for secure flashing of ECU software, for which conforming flash bootloaders are available. However, the concrete realization of the related logistical infrastructure needs to be considered (Fig. 7).

An important aspect of after sales activities is the way OEMs and suppliers react when a security issue is detected in a fielded vehicle. Such scenarios have to be foreseen before the vehicle's SOP and procedures that define actions and responsibilities have to be set up. Actions to be planned are risk assessment of the issue, elimination of critical software vulnerabilities, and update of the software in the field. To achieve an efficient issue handling, a smooth cooperation between OEM and suppliers is required.

Certificates are building the basic concept for a secure and authenticated communication between the vehicle and the backend. They are managed in a PKI system installed and maintained by the OEM. This allows customer oriented service and maintenance with online connections to the vehicle. In case of a car problem, first diagnostic analysis could

be made by the OEM help center in case of a malfunction. It also can be used to report early recognized anomalies and with the collection and analysis of further data an early warning could be sent to the driver before a harmful damage occurs.

Optimal preparation of a service can be achieved when the car workshop reads out online car information. Finally, software updates can be initiated quick and easy without the need to enter the workshop. This can be particularly helpful if software problems are encountered. This is an important if not even an essential pre-requisite for connected cars and provides several advantages: if an attack has been, it can be quickly closed by a software update. Like on a PC, software patches and updates can be distributed to close the vulnerabilities. A secure vehicle-to-backend connection can also provide a secure way to communicate with the internet. It opens also the possibility to perform software updates to ECUs if, for example, a critical software failure in an ECU was encounter.

The advantage of software maintenance over the backend of the OEM is obvious. The communication between the two systems can be strongly restricted, so that other connections are simply not possible, because the firewall at this location does not need to allow any other accesses. The connections are restricted to those partners who have access to the keys and certificates. In addition, system information can be gathered in the field and (anonymously) transmitted, which helps improving car functions. Moreover, this can open new business divisions for an EOM like cloud-services, function enabling, secure internet access or software-as-a-product.

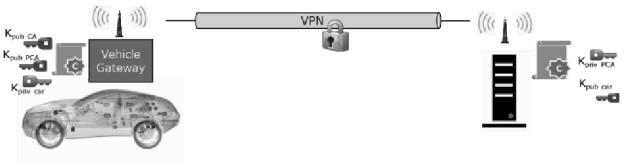


Fig. 7: Secured Remote-connection of the vehicle to the OEM backbone using certificates.

4 Case Study: Connectivity

A major objective in the IT industry is the provisioning of high performance and secure networks in enterprises. The location of the items on the network is just one aspect for the organization and operation of such a network. Considering security aspects in the basic structure from the very beginning can provide essential advantages for the flexibility and scalability of such a network. It can also reduce the risk for attacks. The major attack scenario sin automotive vehicles are described in Fig. 8

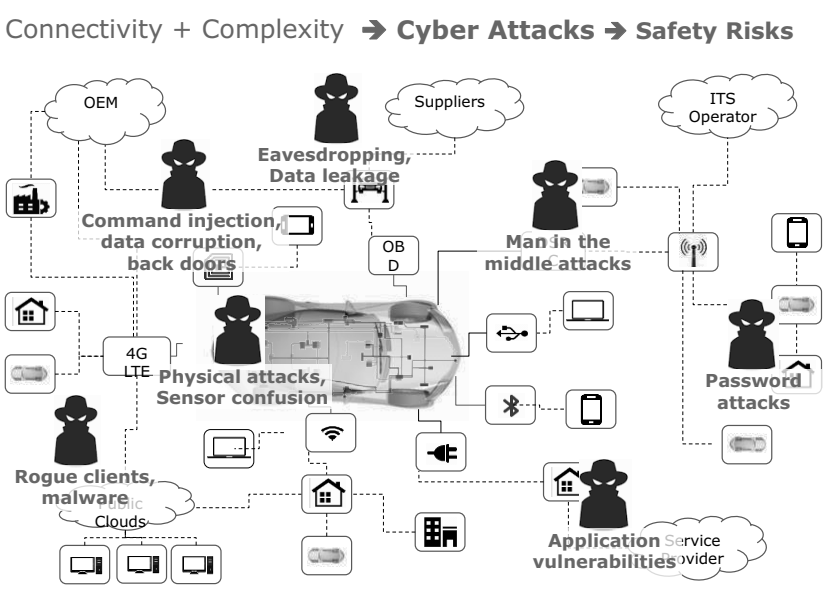


Fig. 8: Increasing connectivity drives complexity and enables multiple attack paths.

Security components like firewall and router are also important parts and are useful to separate networks. For example, account computers will not be connected to the same network as that from marketing or development. Instead, computers are grouped to separate networks depending on their use case and traffic. A router interconnects the networks and provides data exchange between them. It passes only the relevant and allowed data from one network to the other. The access to computers is already restricted by the structure of the network. The router with an integrated firewall also manages the access to the internet. This device observes the incoming and outgoing traffic and can be configured that only allowed traffic will pass. Additionally, maintenance can be done easily on the central part by applying patches or re-configure the device if needed.

Let's consider a car network under the aspects shown above. At the very beginning, a safety and security analysis has been performed and the networks are partitioned so that the connected items are grouped under functional, safety and security aspects. The different networks are interconnected by a gateway. The ECU with a remote connection is considered particularly as unsafe. Even if great care was taken during the software implementation, a failure cannot be excluded and the potential risk is too high that someone could capture and take control of the ECU from outside. To minimize the risk, this ECU should not have access to any other internal networks. We locate this function into a separate ECU (inter comm. module) and connect it through the gateway (Fig. 9).

The gateway can now contain a firewall that has separate filter rules for each subnet. Only those messages are passed to other networks that are allowed. The traffic inside a network is not restricted and affected by the firewall.

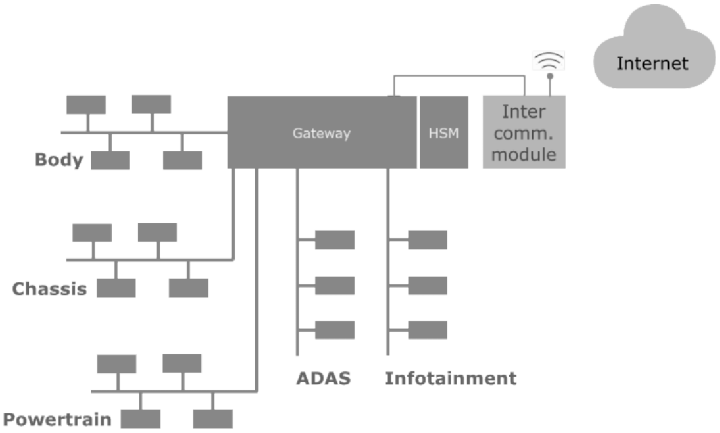


Fig. 9: Vehicle network with remote connectivity and gateway

We now take advantage of our threat analysis and risk assessment that has provided a detailed security analysis of our system. In the data flow diagram we saw the interaction of signals between partitioned functions separated in ECUs. This has already helped us to separate the ECUs to different networks, respectively partitioned the networks according to safety, security and functional aspects. We can now classify networks into security zones according to the safety and security requirements of the transmitted signals. If a network mainly contains signals with high security and safety requirements, that is classified as high security zone. The network with intermediate safety and security data is classified as a medium security zone. The network that contains just a few signals with safety and security requirements and many signals from remote connections is a low security zone. A network that contains an ECU with a remote connectivity must be treated as unsafe in principle and is therefore in a low security zone or even completely isolated (Fig. 10).

The origin and distribution of the signals influences the settings of the firewall in the gateway. The presence of signals from other security zones gives an indication to the security measures for the internal signals. If a network is physically isolated and signals from other networks are rarely used, the threat potentials are low. Unless other threats from adversaries[2] are identified, reduced measures can be applied for signals in such a security zone. This reduces efforts and costs for security measures of these ECUs. It shows how partitioning provides advantages. The signal flow from high to low security zones is not critical. However, if threats for data manipulation on the network are given, security measures like authentication or confidentiality can be added to the data. Greater care must be taken in the other direction. The risk potentials for these signals must be observed carefully. Also, if filter rules of the firewall can influence the complete security zone settings. For counter measures, authentication on signal may be required.

[2] For example, the manipulation of sensor signals for the engine control on the powertrain.

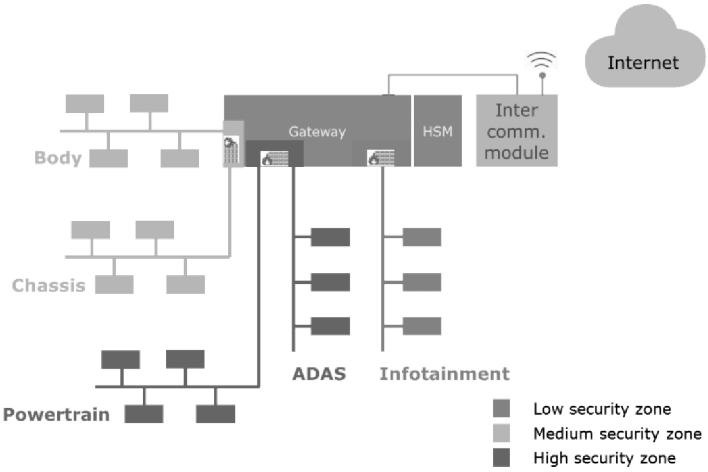


Fig. 10: Security zones in automotive networks

5 Conclusion

Automotive security has gained huge relevance in short time-frame. Attacks are reported today almost continuously and therefore systems must be protected and hardened. Safety needs security as a mandatory condition, which means that any safety-critical system as a minimum must also be protected for cyber security. Security must be integrated early in the design phase of a vehicle to understand the threats and risks to car functions.

Risk-oriented security helps to balance growing security threats with increasing complexity over the entire life-cycle. Unlike many previous attempts our research and many practice projects indicate that while design for security is good, it is not good enough. Effective security must handle the entire life-cycle (Fig. 11).

Security analysis provides requirements and test vectors and adequate measures can be derived for balanced costs and efforts. The results are useful in the partitioning phase when functionality is distributed to ECUs and networks. Networks isolated under security aspects helps to reduce the risks and efforts. Security key management will become an important part and requires a key infrastructure (PKI) managed by the OEM over the production and maintenance phase of the vehicle. Additionally the secure key handling inside an ECU and the usage in development, production and maintenance phase must be considered. The PKI must be online to allow access by the workshops. Additionally, an OEM backend is needed that allows flash programming over the air, at least to provide hot fixes and patches. Such a backend can provide additional security and features to the car owners, but can also open new business divisions for OEMs.

Fig. 11: Security Engineering along the Life-cycle

Companies urgently need to build up necessary basic security expertise and obtain adequate external support, specifically where security meets safety. Mature development processes provide a good basis but need to be amended with dedicated security engineering activities as we have showed in this article.

Bibliography

[EB2016] C. Ebert, A. Braatz: Automotive security engineering, Vector White Paper 2016.

[Se2017] SeVeCom (Secured Vehicular Communication) project: www.sevecom.org, Last accessed on 12.Mrc.2017.

[Si2017] SIMTD (Secure Intelligent Mobility): www.simtd.de, Last accessed on 12.Mrc.2017

[Pr2017] PRESERVE (Preparing Secure Vehicle-to-X Communication Systems): www.preserve-project.eu, Last accessed on 12.Mrc.2017

[Ev2017] EVITA (E-safety vehicle intrusion protected applications): www.evita-project.org, Last accessed on 12.Mrc.2017.

[ISO2017] ISO 26262, ed.2, draft - Road vehicles — Functional safety, Last accessed on 12.Mrc.2017, www.iso.org

[ETSI2010] ETSI TR 102 893, "Intelligent Transport Systems (ITS); Security; Threat, Vulnerability and Risk Analysis (TVRA)," 2010.

[Is2014] M. Islam et al.: Project overview HEAVENS - Healing Vulnerabilities to Enhance Software Security and Safety, Volvo AB, 2014.

[SAE2016] SAE International: “Cybersecurity Guidebook for Cyber-Physical Vehicle Systems, J3061_201601”, 2016, www.sae.org

Hacking Trucks – Cybersecurity Risks and Effective Cybersecurity Protection for Heavy-Duty Vehicles

Marko Wolf[1] and Robert Lambert[2]

Abstract: Similar to passenger cars, heavy-duty vehicles, such as commercial trucks and buses, are becoming increasingly software-driven, interconnected and semi-automated, and hence are also becoming increasingly susceptible to cybersecurity attacks. This article will identify and evaluate these cybersecurity threats and risks affecting the monetary business operation, reliability, and safety of heavy-duty vehicles, comparing them with similar cybersecurity risks for typical passenger vehicles. Based on this overall cybersecurity threat and risk analysis, the article will then present and explain our holistic and multi-layer protection approach to reduce such cybersecurity risks for heavy-duty vehicles.

Keywords: Cyber security, automotive, heavy-duty, security risk, threat, protection

1 Introduction and Motivation

Most automotive industry players agree [McK16] that three central technology trends – namely connectivity, electro-mobility, and autonomous driving – will determine the development of the automotive domain for next 10-15 years. According to Werner Bernhard [Ber16], Head of Daimler Trucks & Buses, significant change will affect commercial vehicles in particular which "will experience more changes within the next 10 years as we have seen in the last 50 years". As shown in Figure 1, the rise of these three game-changing technologies will accelerate the deployment of electronic control systems, greatly increase the amount of vehicular software, and compound the number of digital interfaces, all of which will in turn increase the degree of networking and the system complexity in general.

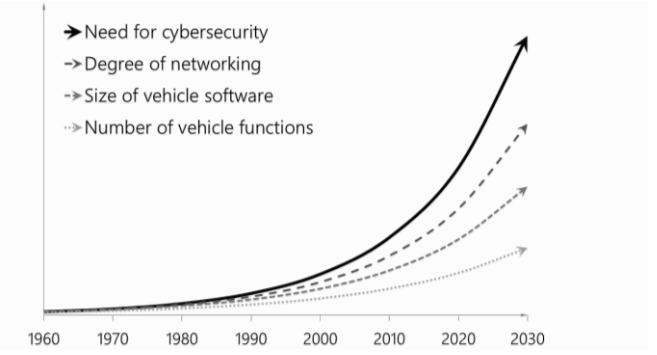


Figure 1: In order to improve fuel efficiency, fleet management, and safety, heavy-duty vehicles

[1]ESCRYPT GmbH, Munich, Germany (marko.wolf@escrypt.com
[2]ETAS Canada Inc., Kitchener, Canada (robert.lambert@etas.com)

will utilize – similarly to passenger vehicles - more electronic control systems, an increasing degree of networking, and a larger amount of software [Cha09]; this clearly also increases the need for proper cybersecurity protections.

However, since "complexity is the worst enemy of security" [Sch12] we will experience also more related cybersecurity risks & threats and hence we will also need more cybersecurity protection. In fact, compared with standard passenger vehicles, heavy-duty vehicles will be even more susceptible to cybersecurity threats since these vehicles:

- will use more complex and software-driven functionality (e.g., for platooning),
- will create, process, store and exchange more data internally and also externally via powerful, long-distance wireless communication channels (e.g., LTE interfaces for fleet management),
- will be more standardized, homogenous, and interoperable (e.g., use interchangeable engines, and employ the SAE J1939 in-vehicle network protocol),
- must often support multiple attachments (e.g., tractor implements) which, if they communicate with the vehicle, present a risk for virus and worm infection (especially since attachments will often be produced by multiple distinct manufacturers, so any weaknesses in communication protocols will take much coordination effort, and even more time, to fix satisfactorily),
- have greater value (typically > 100.000 €) and often carry valuable or dangerous loads (e.g., goods worth 1 million € per truck or hazardous chemicals),
- promise more gains from each attack and have larger potential attack benefits (e.g., systematic toll fraud, large-scale counterfeiting), and last but not least,
- are in motion up to 20 hours a day, with 3x the distance travelled, up to 5x the size and up to 30x the weight of a typical passenger car.

Considering these features together, we perceive how urgent the need for cybersecurity is. Cybersecurity considerations are just as critical as the usual safety considerations for heavy-duty vehicles, and in fact, security considerations are necessary to provide safety.

1.1 Our Contribution

This article will identify and evaluate potential cybersecurity threats and risks affecting the reliability, safety, and monetary business operation of heavy-duty vehicles in comparison with similar cybersecurity risks for typical passenger vehicles. Based on this overall threat and risk analysis, the article will then present and explain our holistic and multi-layer protection approach to reduce such cybersecurity risks for heavy-duty vehicles.

1.2 Related Work

While passenger vehicle security is already well covered in security engineering, security research, and the media (for instance by the notable publication [CMK11]), heavy-duty vehicle security, has up until now, been investigated or tackled only rarely. Some recent publications have begun to raise awareness of the problem, for example [OBr16] and [PSA16]. The currently most prominent publication regarding heavy-duty vehicle security, [BHM16], demonstrates several practical attacks on vehicle safety owing to the openness and easy (physical) access to a standardized in-vehicle network (via SAE J1939 protocol) used across all trucks and other heavy-duty vehicles in the USA. However, to the authors' knowledge there are virtually no publications providing detailed investigations into potential attackers, attack motivations, attack paths, damage potentials, or even potentially effective security protection for heavy-duty vehicles.

2 Cybersecurity Threats on Heavy-Duty Vehicles

While trucks and buses differ from standard passenger vehicles in size, weight, value, typical use, and, attraction to hackers (cf. Section 1), their internal E/E architecture is quite similar to passenger cars. As depicted in Figure 2, they also consist of about 50 distributed electronic control units (ECUs) that communicate with each other over standardized automotive bus networks such as CAN. They further provide various standardized communication interfaces to the outside world such as the physical on-board diagnosis interface (e.g., OBD port), short-range wireless communication interface (e.g., Wi-Fi), and long-distance mobile broadband communication (e.g., LTE). Hence, trucks and buses can also be susceptible to similar cybersecurity threats and risks as passenger cars.

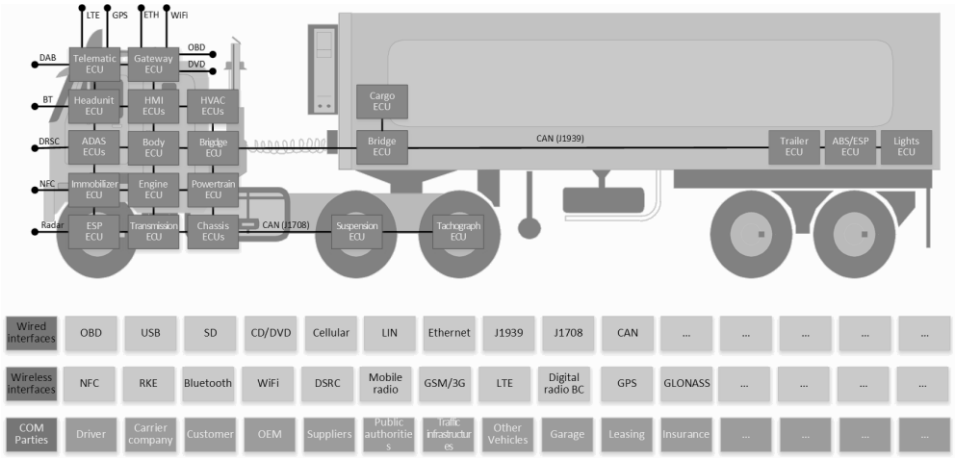


Figure 2: Typical heavy-duty vehicle E/E architecture with its various wired and wireless

interfaces

The next subsections will identify and evaluate current and future cybersecurity threats and risks affecting the monetary business operation, reliability, and safety of heavy-duty vehicles as compared with similar cybersecurity risks for typical passenger vehicles (where similar security threats exist). To this end, we provide exemplary (real-world) attacks; identify typical attackers and evaluate their individual attack potential. We further identify exemplary damaged parties; evaluate the damage potential of the attack; and calculate the resulting cybersecurity risk, which is then compared with similar cybersecurity threats for passenger vehicles (where similar security threats exist). For evaluation of the attack and damage potentials and the calculation of the resulting cybersecurity risk, we use a simplified version of the well-established security risk evaluation method as described in [SW12] and shown in Table 1.

Attack success probability ↓	**Security risk assessment**			
Certain	Medium	High	High	High
Possible	Small	Medium	High	High
Unlikely	Negligible	Small	Medium	High
Very rare	Negligible	Negligible	Small	Medium
Damage potential →	Insignificant	Significant	Critical	Catastrophic

Table 1: Simplified 4x4 automotive cybersecurity risk matrix according to [SW12]

The following sections analyze four important vehicular cybersecurity attack categories, which are physical theft, electronic manipulation, data theft, and safety attacks.

2.1 Physical Theft of Complete Vehicles or Valuable Vehicle Components

Physical theft of complete vehicles or valuable vehicle components is probably the oldest and most prominent vehicle security attack. Compared with passenger vehicles, heavy-duty vehicles are subject to a higher security risk because of the much higher attack gain of up to 1 million € for a truck with a valuable load.

	Passenger vehicle	**Heavy-duty vehicle**
Exemplary attacks	Theft of airbags, navigation systems, whole car	Theft of navigation system, tractor, load, or both
Typical attacker	Organized crime	Organized crime
Attack probability	Possible	Possible
Damaged party	Owner	Owner, operator, customer
Damage potential	Significant	Critical
Resulting cybersecurity risk	**Medium**	**High**

Table 2: Systematic derivation and comparison of cybersecurity risks for passenger vehicles and heavy-duty vehicles regarding vehicle theft or theft of valuable vehicle components

Truck vehicle thieves can abuse the known security vulnerabilities of many remote keyless entry (RKE) and immobilizer implementations, which are similar to those installed in today's passenger cars [GOD16]. Therefore, thieves can attack RKE by:

- simple jamming of the remote "lock" signal,
- calculating and sending the "unlock" signal based on a wiretapped "lock" signal,
- injecting the "unlock" signal into the unprotected onboard network via physically connecting to it through exposed and easily accessible physical bus interfaces (e.g., trailer hitch, external user interface)

If separation between internal and external networks is weak, even wireless interfaces like Wi-Fi or Bluetooth might be abused to inject "unlock" messages. Quick thefts of locked trucks raise suspicions that such thefts based on weak cybersecurity are still prevalent[1]. Truck component thieves in turn can abuse inherently limited physical protection (often put in place in order to enable easy interoperability and exchange of parts) and weak component authentication mechanisms (often not implemented at all) which could prevent the installation or the proper operation of vehicle components from unknown sources.

2.2 Manipulation Attacks on Electronic Vehicle Functionality and Vehicle Data

Together with physical thefts, unauthorized manipulations of in-vehicle data and functionality are probably the most common vehicle cybersecurity attacks. They are usually insider-attacks executed by the legitimate owner or driver of the truck, very often with professional support from specialized companies[2], which makes it particularly hard to defend against, especially since the truck manufacturers are seldom the damaged parties.

In fact, most manipulation attacks try to circumvent legal restrictions that protect the environment (e.g., disable exhaust gas treatment [Bo17]), driving safety (e.g., disable emergency brake system [Sta16]), traffic safety and fair competition (e.g., manipulated speedometers[3]) or try to betray the used-vehicle buyer (e.g., odometer manipulation). Damages to OEMs emerge mainly by warranty fraud due to out-of-specification usage (e.g., chip tuning) or manipulated lifetime counters (e.g., manipulated motor running time). However, with the continuously growing pay-on-demand economy (e.g., truck leasing, truck renting, or very costly special vehicles used only for a short time period such as agriculture vehicles), attacking such digital pay-on-demand (third-party) business models (e.g., pay-as-you-drive insurances) becomes a critical manipulation attack target as well [Law08].

[1] https://www.usatoday.com/story/news/crime/2016/07/05/clarkstown-cops-180k-truck-stolen-lot/86728206/
[2] http://www.allcartuning.com/chiptuning-lkw.html
[3] http://www.c-a-i.net/products.php?category=speedo

	Passenger vehicle	Heavy-duty vehicle
Exemplary attacks	Chip tuning, odometer manipulation, Pay-per-use bypassing, EDR manipulation	Chip tuning, tachograph manipulation, bypassing legal or safety limitations, Pay-per-use bypassing, manipulate vehicle/load monitoring
Typical attacker	Owner	Owner, driver, operator
Attack probability	Unlikely	Possible
Damaged party	OEM, third party, society	OEM, third party, society
Damage potential	Significant	Significant (at least)
Resulting cybersecurity risk	**Small**	**Medium**

Table 3: Systematic derivation and comparison of cybersecurity risks for passenger vehicles and heavy-duty vehicles regarding manipulation attacks on electronic vehicle functionality and data

With modern vehicle E/E architectures, virtually all manipulation attacks can be executed by electronic means alone, with only minimal or even no physical manipulation. The insider attacker will mainly use the easily accessible onboard diagnosis interface (OBD) which allows deep access to virtually all onboard ECUs. In order to manipulate certain data or functionality (usually via some variable control parameters stored in a table in ECU flash memory), the attacker needs to re-engineer some "hidden commands" or - for trucks even more simply – can make "use" of the standardized SAE J1939 protocol used in virtually all modern trucks [BHM16].

Even though most manipulations will cause "only" financial damages, deep software manipulation of today's complex E/E architectures, which control several critical driving functionalities, performed with home-brewed tools of dubious origin and quality, can clearly affect vehicle-driving safety as well, even though that might not have been intended. And here we see an elevated damage potential for heavy-duty compared to passenger vehicles. The attack potential for heavy-duty vehicles is rated higher than for normal passenger vehicles owning to many factors: the standardized, easy accessible J1939 interface and the increased number of promising attack targets that could work to the benefit of an owner, driver, or operator. This elevates the "medium" cybersecurity risk for trucks and buses.

2.3 Data Theft Attacks or Misuse of Digital Vehicle Data

Data theft or data misuse attacks might be expected to be rare events at a first consideration, but are already a multibillion-dollar real-world problem.

The most prominent data theft attacks are IP thefts employed to reduce engineering costs for competing products or to make counterfeit parts. According to the U.S. Federal Trade Commission, "counterfeiting represents a $12 billion per year problem for the entire automotive industry". However, it is not only a financial problem, but is very often also a safety problem. This is because counterfeit parts may not perform as well as legitimate

OEM aftermarket components, may be manufactured with less precision, or may use inferior materials. Truck braking systems are one of the components most likely to be counterfeited, and these fake braking parts result in a large number of deadly accidents [Cla14]. Other IP theft targets are costly to developed engine control software or exhaust-cleaning programs. Vehicular IP thefts and software piracy attacks are mainly insider attacks (i.e., attacks having complete physical control of the target vehicle) executed by dedicated experts that, for instance, simply dump ECU software binaries using an OBD command, re-enable fused debug interfaces, up to more sophisticated physical attacks that, for instance, de-package a chip and read-out memories with powerful microscopes [Sko01].

Like with passenger cars, other data theft attacks are privacy infringements that involve secretly collecting, storing, and transferring, for instance, vehicle location, vehicle operation, or driver's communications[4]. This data could then be used to monitor individual driving behavior (e.g., to defend warranty claims), enable individual marketing (e.g., location-based services), resell collected data to third parties[5] (e.g., Google maps), or – in the worst case – this secretly stored data used against the driver in case of an accident[6]. However, for commercial trucks, in addition to potential privacy infringements, economic espionage is much more likely, and the attack path is similar. In contrast to passenger cars, modern trucks often enable OEMs, logistic operators, carriers, and sometimes even customers to have considerable remote access to truck internal data, even in some cases allowing direct access to the CAN bus to monitor and control vehicle position, or to get information on how the vehicle has been loaded, or even how it is being driven. Competitors can try to hack into these remote interfaces to monitor (or disturb) their competition or might try to steal or purchase such data from third party application providers (e.g., digital toll applications) that collect, store, aggregate, and sell such data without the explicit knowledge and permission of the driver or operator.

While the attack probability for heavy-duty vehicles is already somewhat larger due to the broader deployment of remote access applications, the damage potential for trucks regarding espionage and safety is considerably larger, resulting in a high cybersecurity risk.

	Passenger vehicle	**Heavy-duty vehicle**
Exemplary attacks	IP theft, privacy invasions, counterfeit parts	IP or business secrets theft, privacy invasions, counterfeits parts, vehicle tracking, load control or navigation manipulation, operator/driver extortion
Typical attacker	Plagiarist, competitor, third parties (e.g., insurances), OEM	Plagiarist, competitor, third parties (e.g., insurance companies), OEM, government, organized crime
Attack probability	Possible	Possible

[4] https://www.adac.de/infotestrat/technik-und-zubehoer/fahrerassistenzsysteme/daten_im_auto/
[5] http://www.usatoday.com/story/money/cars/2013/03/24/car-spying-edr-data-privacy/1991751/
[6] https://netzpolitik.org/2016/bmw-speichert-keine-standortdaten-gibt-aber-bewegungsprofil-an-gericht/

Damaged party	Driver, owner, OEM	Driver, operator, customer, OEM, society
Damage potential	Significant	Critical
Resulting cybersecurity risk	**Medium**	**High**

Table 4: Systematic derivation and comparison of cybersecurity risks for passenger vehicles and heavy-duty vehicles regarding data theft attacks or misuse of digital vehicle data

2.4 Attacks on Vehicle Reliability and Vehicle Safety

Finally, yet importantly, truck reliability and safety are at least as endangered as it has been recently demonstrated with real-world passenger cars, where hackers where able to remotely hijack a Jeep over the Internet and have successfully attacked the Jeep's steering, acceleration, and braking systems[7]. In fact, due to the standardized J1939 protocol used in virtually all modern trucks, the (most) costly attack preparation step, the reverse engineering of the susceptible internal commands, would not be necessary, making such safety attacks against trucks and buses much easier. Researchers from Michigan University have already demonstrated such attacks in practice on a class-8 semi-tractor and a 2001 school bus [BHM16]. Even though they have not executed their attacks remotely, it is easy to imagine that hackers will find many similar remote entry points as have already been very successfully found into passenger cars [CMK11].

In real life, such safety attacks have not happened yet against cars nor against trucks, since these attacks are still quite costly to prepare and provide virtually no direct financial benefit, or would cause enormous search pressure if abused for extortion or even terrorism. Thus, we rate the attack probability for trucks "unlikely" in the first instance, while we inherently rate the potential damage of a 40-ton vehicle driving around at 60 mph without brakes "catastrophic", which results again in high cybersecurity risks for trucks compared with "medium" for passenger vehicles.

	Passenger vehicle	**Heavy-duty vehicle**
Exemplary attacks	Delete critical data, lock critical functions, hijack driving functionality	Delete critical data, lock critical functions, hijack driving functions
Typical attacker	Extortionist, terrorist, nation-state	Extortionist, terrorist, nation-state
Attack probability	Very rare	Unlikely
Damaged party	Driver, society	Driver, operator, customer, society
Damage potential	Catastrophic	Catastrophic
Resulting cybersecurity risk	**Medium**	**High**

Table 5: Systematic derivation and comparison of cybersecurity risks for passenger vehicles and heavy-duty vehicles regarding attacks on vehicle reliability and vehicle safety

[7] https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

3 Cybersecurity Protection for Heavy-Duty Vehicles

The following section will provide our holistic, systematic and multi-layer protection approach in order to reduce the cybersecurity risks for heavy-duty vehicles to a minimum. Our holistic approach assures vehicular security by applying the following three security principles:

(1) *Security for the entire heavy-duty vehicle system* (i.e., from individual ECU to connected cloud backend)
(2) *Security for the entire heavy-duty vehicle lifecycle* (i.e., from first requirements analysis to vehicle phase-out)
(3) *Security for the entire heavy-duty vehicle organization* (i.e., from security processes to security governance)

The next three subsections explain the realization of these three security principles in more detail. We do not have to start from scratch, but can benefit and reuse much of the already existing experience and security solutions from the passenger vehicle domain. In fact, most of the security approaches for passenger vehicles can be directly transferred to the heavy-duty vehicle domain.

3.1 Security for the Entire Heavy-Duty Vehicle System

For sustainable vehicular security, it is necessary to always consider the whole vehicle system starting from the individual ECU up to the connected services in the backend, since a smart attacker would also check the whole vehicle system for the weakest link at which to execute an attack most easily. Thus, for instance, even a perfectly secure encryption algorithm would lose all security if we use a global secret key for every truck, if that one key can be obtained from any ECU which uses the secure algorithm.

For sustainable vehicular security, we also need multiple lines of defense since – especially within the rather slow and costly to adapt vehicular security domain - we always have to assume that one of our protection measures might become weakened or even fail. Long term, real-world security experience forbids the typical “single point of failure” protection approaches which might have, for instance, only a single firewall gateway isolating a secure internal vehicle network from an insecure external one, and where a single vulnerability would compromise all vehicles of that type in the world completely and at once.

Unfortunately, until now exists no standardized vehicle security approach yet, but Figure 3 shows how a sustainable vehicular security approach might look from the technical perspective, where the vehicle system employs multiple lines of defense. Each line or layer uses different security mechanisms, assuming that not all security mechanisms would fail at once. Based on a secure trust anchor, usually realized with an automotive-capable hardware security module [WW12], we can assure the integrity (and confidentiality) of the ECU firmware which uses, for instance, secure boot or trusted

boot protection [WG11]. The protected ECU firmware in turn provides higher-level software-based security functions to enable secure onboard communication protocols such as the AUTOSAR-based "Secure Onboard Communication (SecOC)" protocol [AS15]. A secure in-vehicle E/E architecture further separates connected ECUs into three to ten mutually isolated sub-networks of different security and safety classes, which can communicate across subnets only via secure gateway processors enforcing strict firewalling rules [JSV13]. Vehicle-external communication is further protected by a central gateway (CGW) equipped with vehicular intrusion detection (IDS) and response (IRS) systems, which implement external communication security protocols for securing V2V (e.g., IEEE 1609) and V2I (e.g., Embedded TLS) communications [WSA15]. Finally, yet importantly, all relevant backend and infrastructure services such as key management and cloud services, but also connected IoT and cellular devices, need strong classical network security and mobile security solutions.

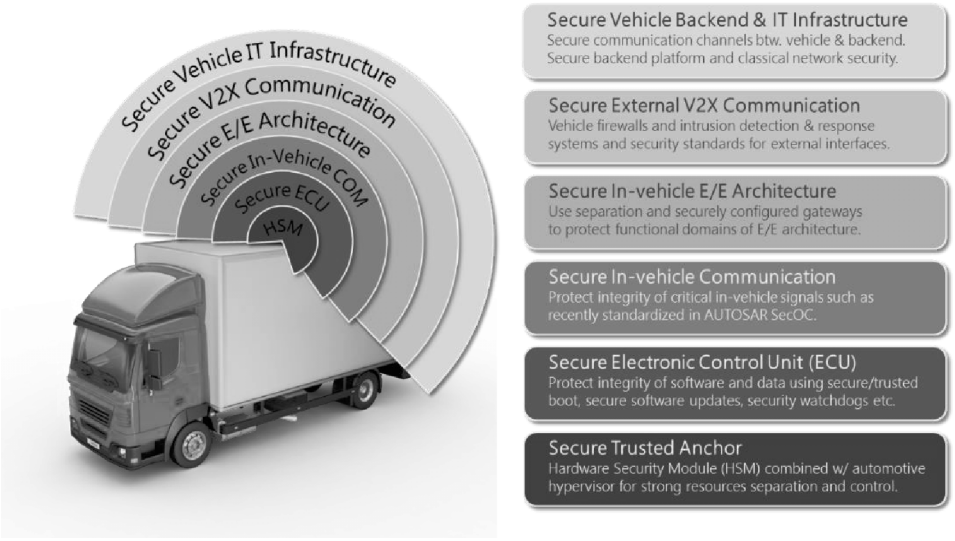


Figure 3: Multiple lines of defense protecting the entire heavy-duty vehicle system.

3.2 Security for the Entire Heavy-Duty Vehicle Lifecycle

In contrast to classical engineering, where the operational environment is mainly defined by natural laws and reliable statistics and where engineering processes usually end with the start of production, security engineering does not end until product phase-out. This is because the security environment is continuously changing, particularly in early production, or when newly identified attack paths, new vulnerabilities, or new security research are discovered.

Thus, security engineering uses a continuous vehicle security lifecycle [SAE16] that

provides security procedures for the whole vehicle lifecycle from requirements engineering until phase-out, as shown in Figure 4 (including some exemplary security procedures executed during each lifecycle phase).

Such a continuous lifecycle also has some additional technical and organizational implications, since for instance all development hardware, all tool chains, and at least some of the experts involved have to remain available until final phase-out, which means for heavy-duty vehicles: for up to 20 years.

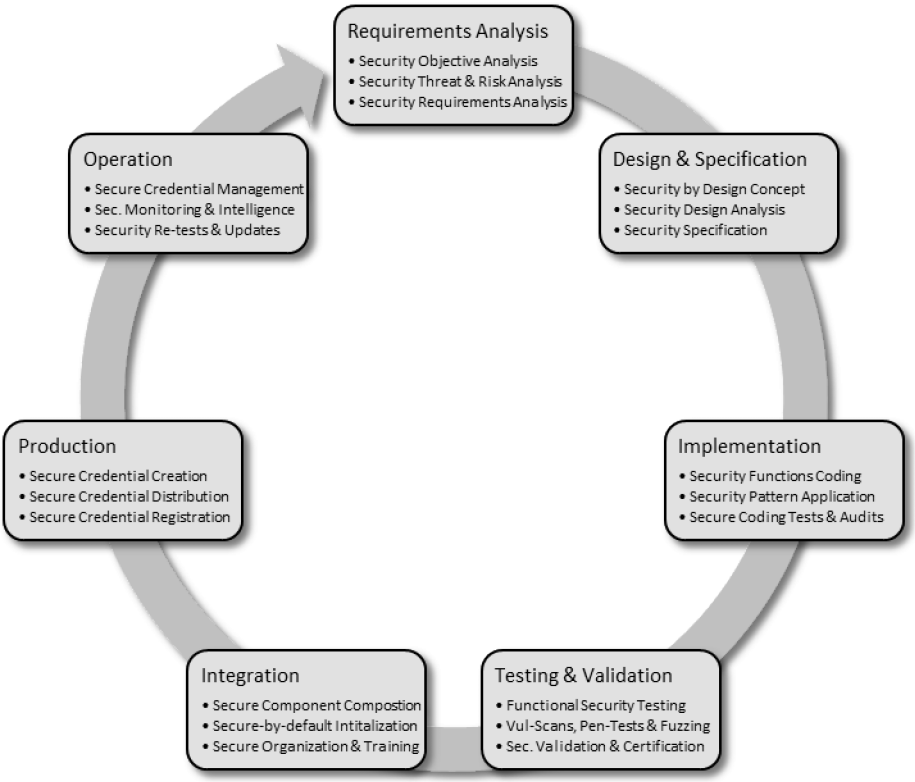


Figure 4: Continuous vehicle security lifecycle with exemplary security operations per lifecycle phase, which are executed continuously until product phase-out, to be able to react to the continuously changing security environment.

3.3 Security for the Entire Heavy-Duty Vehicle Organization

Vehicle security is indeed much more that “just another technical vehicle feature” developed by “just another company division”. In fact, sustainable vehicle security requires deep cross-divisional integration and strong commitment from the whole

organization. This is especially difficult since security, at first glance, creates neither new features nor new revenues, but only additional documentation, processes, and complexity without any immediately apparent benefits.

Without engaging the whole organization, the efforts for security can become quickly ineffective and bogged down by compatibility issues, insufficient resources, hard-wired dummy values, "secret" (debug) circumventions, or organizational process vulnerabilities such as insufficient access and usage control for important cryptographic secrets.

On the other hand, a well-engaged security organization helps a lot for instance to avoid inefficiency by several mutually incompatible isolated solutions (also known as "Insellösungen"). It also clearly reduces security risks by reducing complexity ("which is the worst enemy of security"), provides always a good system overview and ensures proper management of all security-critical functions and corresponding credentials. Moreover, well-organized vehicle security management can in fact increase security without extra costs, for instance, if small separate security mechanisms can together share a powerful high-security hardware crypto module.

Figure 5 gives a first overview on how a vehicle manufacturer or vehicle supplier could setup his vehicle security organizational structure, which is an independent and additional structure to the classical IT security organizational structure. Thus the vehicle security organizational structure shown in Figure 5 clearly focusses on the cybersecurity protection of the company's products, but does not replace classical organizational IT security, such as securing company networks or controlling access to the company's facilities.

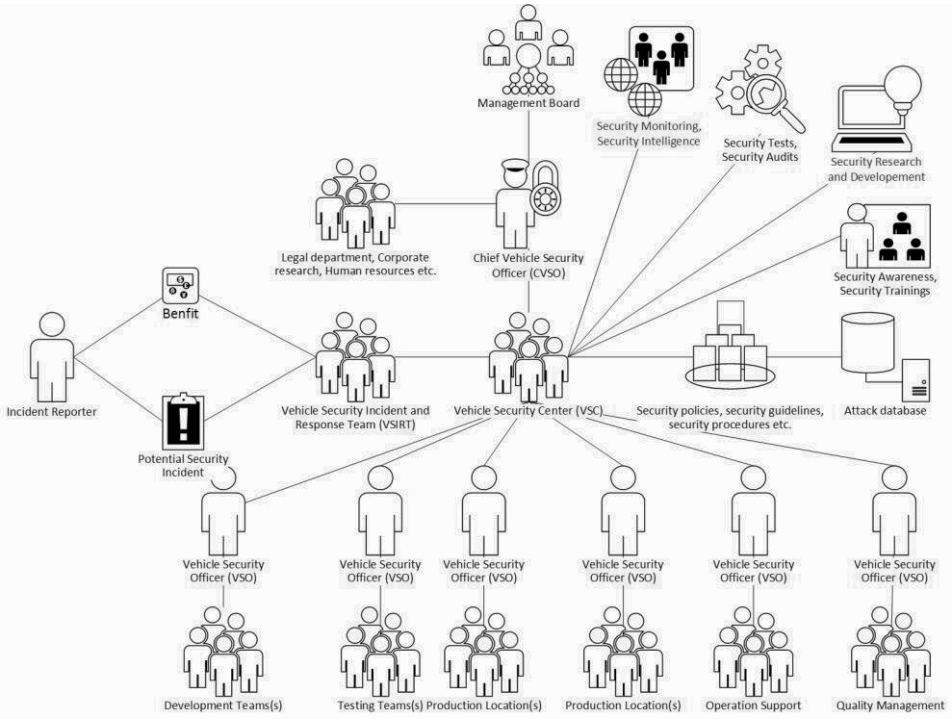


Figure 5: Roles and relations for implementing vehicle security within the organization.

In the following, a very short description of the different security roles and their responsibilities shown in Figure 5 is given.

Vehicle Security Officer (VSO) is an (additional) role of a team member in who is involved in virtually all organizational units participating in the vehicle product lifecycle, such as development, testing, production, and operation, but also in cross-divisional departments such as quality management. The VSO ensures, for instance, that his team members get sufficient cybersecurity training, comply with all relevant security rules and processes, apply up-to-date cybersecurity protection mechanisms, and report new cybersecurity risks and threats (if any), new security requirements, or potential improvements for vehicle cybersecurity protection. VSOs are steered by and report to the Vehicle Security Center.

Vehicle Security Center (VSC) is a team of dedicated vehicle security experts which develop and maintain the relevant cybersecurity procedures (e.g., security engineering process), guidelines (e.g., secure coding guideline), and policies (e.g., access control policy for software signing key) for ensuring sufficient cybersecurity protection of all company vehicle products through their entire lifecycle. The VSC works closely with the Vehicle Security Incident and Response Team (VSIRT) to evaluate (new) security risks

and threats and, if needed, coordinates the development and rollout of effective response measures such as security patches. The VSC further works closely with many other company departments, for instance with legal departments to keep their cybersecurity requirements up-to-date (e.g., new privacy protection laws), with cooperate IT for hosting security services (e.g., security credential management system), or with cooperate research to improve their knowledge about new security threats and effective protections. The VSC is further responsible for in-house security training and awareness, internal security tests and audits security monitoring and intelligence, and development of new cybersecurity protection measures. The VSC in turn is managed the Chief Vehicle Security Officer (CVSO) who will directly (and exclusively) report to the management board.

Vehicle Security Incident and Response Team (VSIRT) is a team of vehicle security experts focused on new cybersecurity risks and threats around the company's products and cybersecurity forensics. The VSIRT monitors press & media, attends relevant security conferences, boards, and committees, talks to customers, employees, and even competitors to learn about new security risks and threats. Sometimes they even provide "bug bounty" programs, which pay for security vulnerabilities detected and reported by so-called "white hackers". The VSIRT is also responsible for executing (e.g., revoking a certificate) or requesting (e.g., development of a security patch) effective response measures in case of a critical product security risk. The VSIRT is steered by the VSC.

Chief Vehicle Security Officer (CVSO) is a senior executive heading all vehicle security activities of a company. The CVSO decides about the vehicle security strategy, manages relevant cybersecurity risks, ensures cybersecurity governance, and makes decisions on all critical incident response measures (e.g., service shutdowns). Since cybersecurity is a cross-divisional function, the CVSO reports only directly to the management board and can thus push necessary cybersecurity protection requirements and measures through all other company departments.

4 Summary and Outlook

In this article, we have identified and evaluated potential cybersecurity threats and risks affecting the reliability, safety, and monetary business operation of heavy-duty vehicles in comparison with similar cybersecurity risks for typical passenger vehicles. Based on this analysis, we then presented and explained our holistic protection approach to reduce such cybersecurity risks for heavy-duty vehicles.

The analysis has shown that the cybersecurity risks for heavy-duty vehicles are often of higher risk when compared to typical passenger vehicles, since the corresponding attacks on heavy-duty vehicles could be executed easier or would have a larger damage potential. The analysis further shows that most of these cybersecurity threats are already realistic, in fact already executed, today and will become even more critical in the future.

But the article showed also that many effective cybersecurity protection measures already existing in the passenger vehicle domain, can very often be easily transferred to the heavy-duty vehicles. The next version of this article will further investigate the costs and efforts needed for implementing proper cybersecurity protections for heavy-duty vehicles and will show that the return on investment will be achieved even earlier and more easily when compared with the implementation and return expected in standard passenger vehicles.

5 References

[AS15] AUTOSAR, "Specification of the Secure Onboard Communication", In *AUTOSAR Release 4.2.2*, July 2015.

[Ber16] Wolfgang Bernhard, "Daimlers Lkw-Chef will autonome Trucks bis 2020", In *Manager Magazin*, June 2016.

[BHM16] Yelizaveta Burakova et al., "Truck Hacking: An Experimental Analysis of the SAE J1939 Standard", In *USENIX Workshop on Offensive Technologies*, August 2016.

[Bo17] Christian Bock, "Die Lüge vom sauberen LKW", In *ZDF Zoom*, January 2017.

[Cha09] Robert Charette, "This car runs on code" In *IEEE Spectrum 46.3*, 2009.

[Cla14] Jane Clark, "Are Your Aftermarket Truck Parts the Real Deal?", In *Truckinginfo.com*, March 2014.

[CMK11] Stephen Checkoway et al., "Comprehensive Experimental Analyses of Automotive Attack Surfaces", In *USENIX Security*, August 2011.

[GOD16] Flavio Garcia et al., "Lock It and Still Lose It–On the (In) Security of Automotive Remote Keyless Entry Systems", In *USENIX Security*. August 2016.

[JSV13] James Joy et al., "Gateway Architecture for Secured Connectivity and in Vehicle Communication", In *VDI Wissensforum*, October 2013.

[Law08] Nate Lawson, "Highway to Hell: Hacking Toll Systems", In *Blackhat*, August 2008.

[McK16] McKinsey, "Automotive Revolution Perspective Towards 2030", In *Advanced Industries*, January 2016.

[OBr16] Chris O'Brien, "Long-Haul Trucking Connectivity Brings Hacking Risks", In *Trucks.com Trucking Technology*, May 2016.

[PSA16] FBI & NTHSA, "Motor Vehicles Increasingly Vulnerable to Remote Exploits", In *Public Service Announcements* I-031716-PSA, March 2016.

[SAE16] SAE J3061 Vehicle Cybersecurity Systems Engineering Committee, "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems", In *SAE International*, January 2016.

[Sch12] Bruce Schneier, „Complexity the Worst Enemy of Security", In *Schneier Security Blog*, December 2012.

[Sko01] Sergei P. Skorobogatov, "Copy Protection in Modern Microcontrollers", In http://www.cl.cam.ac.uk/~sps32/mcu_lock.html, November 2001.

[Sta16] Kristina Staab, "Wenn der Lkw die Notbremse zieht", In *SWR Aktuell Hintergrund*, August 2016.

[SW12] Michael Scheibel et al., "A Systematic Approach to a Quantified Security Risk Analysis for Vehicular IT Systems", In *Automotive Safety & Security*, November 2012.

[WG11] Marko Wolf et al., "Design, Implementation, and Evaluation of a Vehicular Hardware Security Module", In *International Conference on Information Security and Cryptology*, November 2011.

[WSA15] Yaron Wolfsthal et al., "Solution for Detecting Cyber Intrusions to Connected Vehicles", In *IBM Security Intelligence*, September 2015.

[WW12] André Weimerskirch et al., “Hardware Security Modules for Protecting Embedded Systems”, In *ESCRYPT Security Whitepapers*, May 2012.

Extending a Compiler Backend for Complete Memory Error Detection

Norman A. Rink[1] Jeronimo Castrillon[2]

Abstract: Technological advances drive hardware to ever smaller feature sizes, causing devices to become more vulnerable to faults. Applications can be protected against errors resulting from faults by adding error detection and recovery measures in software. This is popularly achieved by applying automatic program transformations. However, transformations applied to intermediate program representations are fundamentally incapable of protecting against vulnerabilities that are introduced during compilation. In particular, the compiler backend may introduce additional memory accesses. This report presents an extended compiler backend that protects these accesses against faults in the memory system. It is demonstrated that this enables the detection of all single bit flips in memory. On a subset of SPEC CINT2006 the runtime overhead caused by the extended backend amounts to 1.50*x* for the 32-bit processor architecture *i386*, and 1.13*x* for the 64-bit architecture *x86_64*.

Keywords: transient hardware faults, soft errors, memory errors, error detection, fault tolerance, resilience, compiler backend, code generation, intermediate representation (IR), LLVM

1 Introduction

Aggressive technology scaling increases the rates of hardware faults [Sh02, Bo05, Bl06, Ba05], and faults cause erroneous application behavior with non-negligible probabilities [SPW09, NDO11]. Transient hardware faults, also known as *soft errors*, are commonly attributed to cosmic radiation [Ba05]. However, due to shrinking feature sizes, devices are also becoming more vulnerable to variations in supply voltage and temperature, which reduces reliability [Bo05, Sh14]. Moreover, the current trends toward lowering energy consumption and temperature dissipation further reduce reliability [Es11, Ta12, Sh14].

In safety-critical automotive applications, faults that go undetected can pose a danger to human life. Therefore, software that is designed for applications with strict safety and reliability requirements must incorporate measures to tolerate hardware faults [Pa08]. Software can be made fault-tolerant by adding integrity checks to programs. When a check fails, an error has been detected and suitable measures can be taken to recover from it. To enable checks, and hence error detection, some form of redundancy must be added to programs. This can be done conveniently by applying automatic program transformations, such as source-to-source transformations, cf. [Re99, BSS13, KF15, Ka16]. With the rising popularity of the LLVM framework and intermediate representation (IR) [LA04],

[1] Center for Advancing Electronics Dresden, Technische Universität Dresden, Chair for Compiler Construction, 01062 Dresden, norman.rink@tu-dresden.de
[2] Center for Advancing Electronics Dresden, Technische Universität Dresden, Chair for Compiler Construction, 01062 Dresden, jeronimo.castrillon@tu-dresden.de

many fault tolerance schemes have appeared that are implemented as IR transformations, e.g. [FSS09, Sc10, Fe10, Zh10, Ri15, CNV16]. Operating on IR has the advantages of target-independence and increased productivity compared with operating on machine instructions. However, when transformations are applied to programs at an abstraction level above machine instructions, the compiler backend may introduce new vulnerabilities to faults. Specifically, the backend introduces numerous additional memory accesses, cf. Figure 1, which are then not protected against faults in the memory system.

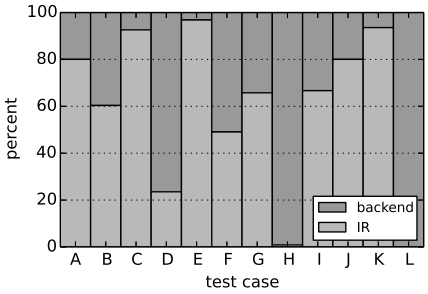


Fig. 1: Dynamic load operations present in the intermediate representation (IR) of programs or inserted by the compiler backend. The test programs labeled A–L are introduced in Section 4, cf. Table 1.

In this work we present a modified compiler backend that adds error detection measures to the memory accesses it inserts. By combining this backend with any fault tolerance scheme that operates on program IR or source code, errors can be detected in all memory accesses that occur in the final machine code. Our compiler backend implements error detection by *dual modular redundancy* (DMR), i.e. data words in memory are duplicated. Whenever a data word is loaded from memory, the duplicated copies are compared. If disagreement is found, an error has been detected. Thus, DMR is capable of detecting any number of flipped bits within a data word. For complete protection of all memory accesses in the final machine code of programs, the extended compiler backend will be combined with the *AN encoding* error detection scheme at the IR level [Br60, Fo89, FSS09, Ri15].

Many fault tolerance schemes circumvent the problem of memory errors by assuming that memory is protected against faults by hardware measures, such as memory modules equipped with *error correcting codes* (ECC) [Re05, YGS09, MPC14, DS16]. This assumption, however, is problematic for two reasons. First, cost and area considerations may rule out using ECC memory at all levels in the memory hierarchy, especially in on-chip caches and load-store queues. In fact, the need to protect a processor's load-store queue against faults has recently been stressed [DS16]. Second, it has been found that the widely used *single error correcting, double error detecting* (SECDED) codes are incapable of handling large fractions of error patterns that occur in practice [HSS12].

This article is structured as follows. Section 2 introduces memory faults and error detection schemes, including the AN encoding scheme. Section 3 identifies the memory accesses that are inserted by the compiler backend and explains how they are equipped with error

detection measures. Section 4 evaluates our error detection scheme. Section 5 discusses related work, and Section 6 summarizes and discusses the findings of the present work.

2 Background

Faults in memory are a major cause of erroneous application behavior and service disruptions [SPW09, HSS12]. Typical faults in memory cells are bit flips caused by energetic particles that originate from cosmic radiation [Ba05]. However, motivated by the current trend toward reducing energy consumption, it has been suggested that the operating voltage of SRAM be lowered [Es12], and that refresh cycles of DRAM modules be extended [Li11, We15]. Both suggestions reduce the capability to retain data and hence increase the probability of memory faults.

The probability that a data word is corrupted by a fault increases with the time that the data word spends in memory. When considering fault tolerance measures for the memory system, main memory is targeted first since this is where the lifetimes of data will generally be the longest. This also means that when, say, ECC are implemented in hardware, on-chip memories, such as low cache levels or load-store queues, may not be protected, cf. [DS16].

2.1 DMR-based error detection

Error detection schemes work by maintaining redundant information that is used to check the integrity of data. This is most evident in error detection schemes based on DMR, where two copies are kept of each data word. If the two copies disagree, an error must have occurred. By comparing the two copies, all single bit flips can be detected. Multiple bit flips can also be detected, provided they do not affect the two copies in identical ways. In particular, multiple bit flips can always be detected if they occur in only one of the copies.

Note that applying error detection by DMR to multi-threaded applications can be problematic. If all memory accesses are duplicated non-selectively, care must be taken to avoid race conditions when different threads access redundant copies of data.

2.2 Error detection by encoding

An alternative approach to error detection is based on encoding data. If the set of valid code words is a small subset of all possible data words, a fault is likely to produce a data word that is not a valid code word. Hence, errors can be detected by checking whether data words are also valid code words. Parity checking is an example of this: in valid code words, the parity bit equals the parity of the code word. This enables the detection of single bit flips. ECC memory typically uses more sophisticated codes, which can also correct errors.

When data is encoded, additional bits are required to represent code words. Although these bits contain redundant information, no data is duplicated explicitly. Therefore, encoding-based error detection schemes can immediately be applied to multi-threaded applications.

A simple, yet effective, encoding-based error detection scheme for integer values can be defined by decreeing that the valid code words are precisely the multiples of a fixed integer constant A. This is known as *AN encoding* [Br60, Fo89]. Variants of AN encoding are popularly used in software-implemented error detection [CRA06, FSS09, Sc10, KF15, Ri15, Ka16]. While AN encoding has the advantage that its capability to detect complex error patterns can be adjusted flexibly by varying the encoding constant A, not all values of A are equally well-suited to error detection [Ho14].

2.3 Memory error detection by AN encoding

We now describe a scheme for detecting errors in memory. The scheme is based on AN encoding, and the key idea is that only valid code words are kept in memory. To achieve this, an integer value m must be *encoded* before being stored:

$$m_{encoded} = m \cdot A. \tag{1}$$

Consequently, whenever a value $m_{encoded}$ is loaded from memory, it must be *decoded* before further processing takes place:

$$m = m_{encoded}/A. \tag{2}$$

Errors can be detected by evaluating the following boolean expression for a value n that has been loaded from memory:

$$n \bmod A = 0. \tag{3}$$

In the absence of errors, the value n is a valid code word, and expression (3) evaluates to `TRUE`. Hence, if expression (3) evaluates to `FALSE`, an error must have occurred.

The presented AN encoding scheme has been implemented by instrumenting load and store instructions in the LLVM IR of programs. Every store instruction is preceded by a multiplication with the constant A, cf. (1). Following every load instruction there is a modulo operation for error checking, cf. (3), and a division for decoding, cf. (2). Figure 1 proves that this approach to memory error detection has its limitations since the compiler backend inserts additional load instructions when lowering the IR to machine code. Errors in memory that affect these load instructions cannot be detected at the IR level.

3 The Extended Compiler Backend

To overcome the limitations of IR-based error detection, the additional memory accesses that are inserted by the compiler backend must be protected against faults. Backends for the C programming language insert memory accesses for the following purposes: to handle register spills (*spill*); to save and restore callee-saved registers (*csr*), the frame pointer (*fptr*), and the return address (*return*); to pass function arguments (*arg*); to access jump tables (*jt*). Since all of these memory accesses, apart from jump table accesses, operate

on the local program stack, duplicating these accesses causes no issues for multi-threaded applications that use shared memory. Since jump table accesses are read-only, their duplication is safe too. We have extended the LLVM backend [LA04] for the *x86* architecture to implement DMR-based error detection for backend-inserted memory accesses.

3.1 Register spills

The *x86* machine code in Listings 1 and 2 illustrates how DMR-based error detection works for register spills. Originally, cf. Listing 1, the register `eax` is spilled to a stack slot at offset –0x30 from the frame pointer (in register `ebp`). The extended backend allocates a second stack slot at offset –0x34, cf. Listing 2. When the register `eax` is restored, the values at the two stack slots are compared. If disagreement is found, control is transferred to an error handler. For the purpose of the present work, error handling consists of exiting the program with a special exit code that indicates that an error has been detected.

List. 1: Register spill and restore.

```
mov   eax,-0x30(ebp)
  ...
mov   -0x30(ebp),eax
add   eax,esi
```

List. 2: Duplicated spill and error checking.

```
mov   eax,-0x34(ebp)
mov   eax,-0x30(ebp)
  ...
mov   -0x30(ebp),eax
cmp   -0x34(ebp),eax
jne   <error_handler>
add   eax,esi
```

3.2 Other stack accesses

The memory accesses *csr*, *fptr*, *return*, and *arg* are analogous to register spills in that they also save values to the program stack and later restore these values to registers. DMR-based error detection is added by duplicating the values on the stack, completely analogously to Listing 2. Full implementation details of DMR-based error detection for these memory accesses can be found in the accompanying technical report [RC16]. Here we only discuss the subtleties of the *return* and *arg* accesses.

On the *x86* architecture, the return address is always passed on the stack. Thus, given the possibility of memory faults, it can never be assumed that the return address is correct. To obtain a copy of the return address that is guaranteed to be correct, the calling convention has been modified so that the return address is passed in a register. Note that on architectures with a designated return register, e.g. ARM or MIPS, protecting the return address against memory faults does not require that the calling convention be modified.

To detect errors in function arguments that are passed on the stack, the calling convention has been modified so that a duplicated copy of the argument sequence is put on the stack immediately above the original sequence, as in Figure 2. When a function argument is loaded into a register, error checking is performed by comparing its value with the corresponding value in the duplicated argument sequence.

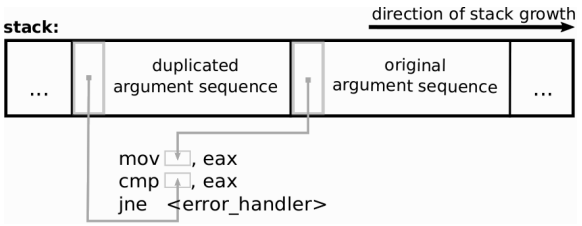


Fig. 2: Original and duplicated function arguments on the stack.

The obligation to implement our modified calling convention rests entirely with the caller. This means that, if a callee chooses not to perform error detection on the return address or on its stack arguments, this does not break function calls. In particular, library functions can still be called fully transparently from within protected functions.

3.3 Jump tables

Jump tables are arrays of addresses of basic blocks, and they reside in the program code segment. To protect jump tables against errors, the extended backend duplicates each jump table in the code segment. Before transferring control to an address that is stored in a jump table, error checking is performed by comparing the address with the corresponding entry in the duplicated jump table.

4 Evaluation

Since faults occur rarely in individual devices, one must actively inject faults into systems or programs to evaluate the effectiveness of fault tolerance schemes. In this work, the test programs from Table 1 are used for this purpose. Some of the test programs (C, E, K) appear in the MiBench suite [Gu01], and similar programs are often used to evaluate fault tolerance schemes [Re99, OSM02, KF15, Ri15, DS16]. The programs represent typical algorithmic tasks, such as sorting, tree and graph traversal, manipulation of bit patterns, and linear algebra. In test program L, a switch statement selects one of the many arguments of the enclosing function; this test has been included here since it is the only one that passes function arguments on the stack for the 64-bit calling convention on *x86*.

Binaries have been generated from the test programs on the *i386* architecture (the 32-bit version of *x86*) and on *x86_64* (the 64-bit version of *x86*). Figure 3 depicts the work flow for this. Program IR is generated by the Clang compiler frontend, which is part of the LLVM infrastructure. Binaries are evaluated on both *i386* and *x86_64* since these architectures have different numbers of general purpose registers: while *x86_64* has sixteen, *i386* has only eight. As a consequence, there will be more backend-inserted memory accesses in *i386* binaries. All binaries have been generated at optimization level `-O3`, and the constant $A = 58659$ has been used for AN encoding, cf. [Ho14].

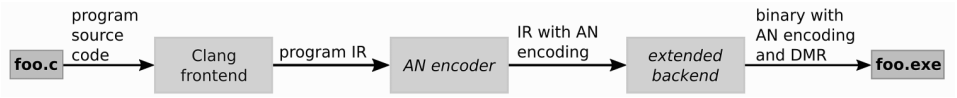


Fig. 3: Generation of binaries with memory error detection measures.

	description
A	array reduction
B	bubblesort
C	cyclic redundancy checker (CRC-32)
D	DES encryption algorithm
E	Dijkstra's algorithm (shortest path)
	arithmetic expression interpreter
F	recursive expression tree evaluation
G	token lexer for arithmetic expressions
H	arithmetic expression parser
I	matrix multiplication
J	array copy
K	quicksort
L	switch

Tab. 1: Suite of test programs.

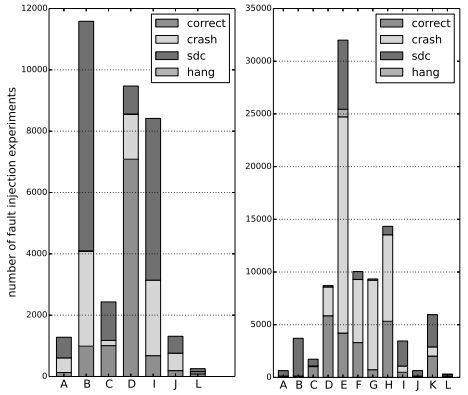
It should be noted that AN encoding produces incorrect programs if the bit width of the constant A is so large that not all encoded values fit into the machine data word. This problem occus on the *i386* architecture when high addresses, e.g. addresses in the stack area, are encoded: multiplying high addresses with the constant A results in a value that cannot be represented by 32 bits. For this reason, AN encoding of the test programs E, F, G, H, K fails on the *i386* architecture. On *x86_64* this is not a problem since pointers are only 48 bits wide and the chosen constant $A = 58659$ is represented by 16 bits.

In evaluating error detection schemes, it is common practice to inject single bit flips, e.g. [YGS09, Fe10, DS16]. To study how programs respond to memory errors, we inject single bit flips into the data words resulting from load operations. This is facilitated by the Intel Pin tool [Lu05] for dynamic binary instrumentation: during the execution of a binary, a single load operation is instrumented with an xor-operation that flips one of the bits in the result of the load. We refer to the execution of a binary with a flipped bit as a *fault injection experiment*. The outcome of a fault injection experiment is determined by the program's response to the injected bit flip. The following responses can occur:

1. *correct*: The program terminates normally and produces correct output.
2. *hang*: The program runs for longer than $10x$ its normal execution time, and is therefore deemed to hang. In practice, especially in safety-critical embedded applications, a hardware watchdog may terminate and restart long-running programs.
3. *crash*: The program terminates abnormally, e.g. due to a segmentation fault.
4. *sdc*: Silent data corruption occurs when the program terminates normally but produces incorrect output.

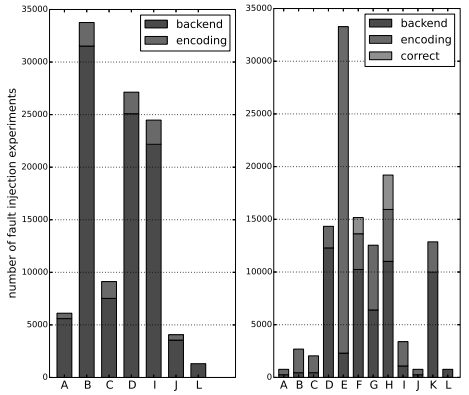
5. *encoding*: The fault is detected by AN encoding.
6. *backend*: The fault is detected by the extended backend's DMR measures.

The fault injection experiments conducted in this work cover all possible patterns in which single bit flips in memory can affect the binaries generated from the test programs. In the following, we therefore report absolute numbers of program responses.



a: *i386* binaries. b: *x86_64* binaries.

Fig. 4: No error detection.



a: *i386* binaries. b: *x86_64* binaries.

Fig. 5: AN encoding and DMR in the backend.

4.1 No error detection

Figure 4 summarizes the fault injection experiments for the *plain* binaries generated from the programs in Table 1, i.e. the binaries without any form of error detection. Only the test programs for which AN encoding produces correct programs appear in Figure 4a.

While *crash* responses indicate that something has gone wrong, when *sdc* occurs in practice, one has no reason to believe that the computed output is incorrect. Therefore, one is often particularly interested in the proportion of *sdc* [Fe10, KF15, DS16]. For the *i386* binaries the number of *sdc* is generally larger than for *x86_64*. This is to be expected given that *i386* has fewer registers: more data words that are relevant for the program output will, at least temporarily, reside in memory and hence be vulnerable to faults.

Figure 4 shows that there is indeed a need for error detection schemes. While some faults do not affect program behavior, thus leading to *correct* responses, there is always a large fraction of malignant program responses, i.e. *crash*, *hang*, and *sdc*.

4.2 AN encoding with DMR in the compiler backend

When AN encoding at the IR level is combined with the extended compiler backend, all single bit flips in memory are detected, as evidenced by Figure 5. For the binaries F and

H on *x86_64* there are a number of *correct* responses, which occur when faults affect load operations that are part of a call to the `memcpy` library function. Although this function call is present in the IR, it is not protected by our AN encoding scheme since no data is loaded into the program. The response *correct* ensues when faults affect only those portions of the copied data that are subsequently not used and hence not loaded into the program.

While Figure 5a, for the 32-bit binaries, is dominated by *backend* responses, this is not the case for Figure 5b. Since *x86_64* has more general purpose registers than *i386*, there is lower register pressure, causing the backend to insert fewer additional memory accesses to handle *spill*, *csr* etc. Also note that the total numbers of fault injection experiments in Figure 5a are about twice as high as in Figure 4a. This is due to the dominating *backend* responses in Figure 5a and the fact that the extended backend duplicates memory accesses.

Figure 5 proves that our approach to detecting memory errors is effective: no malignant program responses remain. In particular, DMR-based error detection in the extended compiler backend succeeds at removing the vulnerabilities that Figure 1 hints at.

4.3 Runtime overheads

Fault tolerance comes at the price of performance penalties since some form of redundancy is required. The runtimes of the test programs from Table 1 are depicted in Figure 6, where geometric means across all test programs are shown. Runtimes have been normalized to the *plain* binaries, without any error detection measures.

The largest fraction of overhead is due to AN encoding, which is plausible given the high latency of integer multiplication, division, and modulo, cf. (1)–(3). In fact, AN encoding is known to introduce large overheads, cf. [FSS09, Ri15, Ka16]. Given that AN encoding is responsible for detecting only a small fraction of errors in the *i386* binaries, cf. Figure 5a, the overhead that AN encoding introduces may not be justifiable on *i386*.

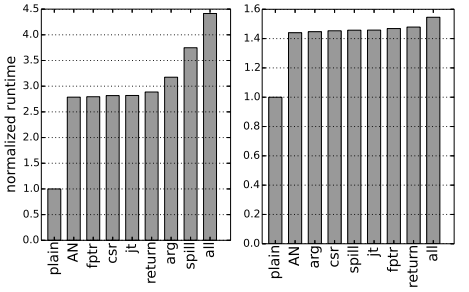
The overhead that the extended backend causes on *i386* is dominated by *spill*, followed by *arg*. This is in agreement with the fact that *i386* has relatively few registers and uses a calling convention by which all arguments are passed on the stack. Neither of these observations apply to the *x86_64* architecture, and hence the overhead introduced by the extended backend is considerably lower.

The runtime overhead introduced by the extended backend has also been evaluated on a subset of the SPEC CINT2006 suite. The subset consists of those C benchmarks that are unaffected by our modified calling convention, which are: `400.perlbench`, `401.bzip2`, `429.mcf`, `445.gobmk`, `458.sjeng`, `462.libquantum`. Geometric means across these benchmarks are shown in Figure 7, where runtimes have again been normalized to the *plain* binaries, to which the backend has not applied any DMR-based error detection measures. Note that the overhead introduced by duplicating function arguments is lower in Figure 7a than in Figure 6a. An explanation for this is that functions in the SPEC benchmarks have larger bodies, and hence longer execution times, than in the test programs from Table 1. Therefore, the overhead introduced by duplicated function arguments car-

ries less weight. When *all* DMR measures are applied by the backend, the resulting mean overheads are 1.50*x* on *i386* and 1.13*x* on *x86_64*.

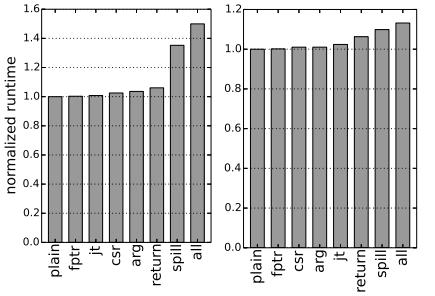
From Figures 6 and 7 it is clear that the handling of register spills is the dominant source of overhead among the DMR-based measures in the compiler backend. Comparing the results for the *i386* and *x86_64* architectures, it can be concluded, perhaps unsurprisingly, that memory error detection is more efficient on architectures with many registers. It should also be noted that the overhead of handling return addresses can be reduced, if not entirely avoided, on architectures that pass the return address in a register, e.g. ARM or MIPS.

All runtime measurements were conducted on an Intel Core i7-4790 CPU (3.6GHz), with 32GB of main memory. The operating system is Ubuntu 16.04.1 LTS, with a 4.4.0 kernel.



a: *i386* binaries. b: *x86_64* binaries.

Fig. 6: Mean overheads for test programs.



a: *i386* binaries. b: *x86_64* binaries.

Fig. 7: Mean overheads for SPEC.

5 Related work

Fault tolerance schemes that operate on program source code appeared early, and this approach is still pursued [Re99, BSS12, BSS13, KF15, Ka16]. Low rates of silent data corruption can be achieved despite the fact that the compiler backend may introduce new vulnerabilities after these schemes have been applied. However, a considerable proportion of faults still lead to program crashes, e.g. [KF15].

With the advent of super-scalar processors it became viable to implement DMR-based error detection schemes by duplicating machine instructions [OSM02]. Subsequently proposed fault tolerance schemes were also implemented by modifying compiler backends, e.g. [Re05, YGS09, MPC14, DS16]. Unlike in the present article, these schemes assume that memory is protected by hardware measures, e.g. ECC, and hence memory operations are not accompanied by error detection. The only exception to this is the nZDC scheme [DS16], where memory accesses are duplicated since the processor's load-store queue is assumed to be vulnerable to faults. Since the nZDC scheme duplicates all load operations, and not just those that access local memory, it is limited in handling multi-threaded applications correctly, as already noted in [DS16].

The popularity of the LLVM compiler framework and IR [LA04] has led to many IR-based fault tolerance schemes [FSS09, Sc10, Fe10, Zh10, Ri15, CNV16]. The DMR-based schemes [Fe10, Zh10, CNV16] assume, once again, that memory is protected against faults by hardware measures. The encoding-based schemes [FSS09, Sc10, Ri15] do not make this assumption, but they suffer from the observed shortcoming that the memory accesses that are introduced by the compiler backend are left unprotected.

That return addresses and frame pointers need protection was already observed in the context of protecting an operating system against hardware faults [BSS13]. The fault tolerance scheme in [BSS13] was implemented based on *aspects* [SGSP02]. Conceptually, aspects operate on program source code, but their implementation requires interaction with the compiler. Thus, the implementation of aspects may introduce new vulnerabilities.

AN encoding was originally introduced in [Br60] and studied in detail, among other *arithmetic error codes*, by [Ga66, Av71]. Protecting processors by AN encoding was suggested in [Fo89], where the ANB and ANBD schemes were also introduced. IR-based implementations of AN encoding appeared in [FSS09, Ri15]. As in the present article, other fault tolerance schemes also combine encoding with DMR [OMM02, CRA06, KF15]. Here we applied DMR very selectively, only to local memory accesses inserted by the compiler backend, which has the advantage that duplication is safe in multi-threaded programs.

6 Summary and Discussion

Fault tolerance schemes that are applied to programs at the level of intermediate representation (IR) cannot address vulnerabilities resulting from later stages of the compilation process. Specifically, the compiler backend introduces additional, unprotected memory accesses to implement, e.g., register spills. In this article we have presented an extended backend that adds error detection by dual modular redundancy (DMR) to the memory accesses it inserts. It has been shown that this, combined with an IR-based AN encoding scheme, succeeds at detecting all errors resulting from single bit flips in memory.

The extended backend introduces an average runtime overhead of $1.50x$ for binaries from SPEC CINT2006 running on *i386*, and $1.13x$ for the corresponding binaries running on *x86_64*. This is in agreement with the expectation that there is less need to protect against faults in the memory system on machines with more registers, i.e. on *x86_64*. The reported runtime overheads are noticeably lower than for the nZDC scheme, which also duplicates memory accesses [DS16]. This is unsurprising since, in the present work, error detection has been applied to memory accesses more selectively.

Implementing fault tolerance schemes at the IR level enables target-independence and enhances productivity. The latter is particularly important for relaxed fault tolerance schemes, where some amount of vulnerability is accepted in exchange for reduced overhead [Fe10, KF15, Ri15]. In quantifying the vulnerabilities of a relaxed scheme, meaningful results can only be obtained if one is guaranteed that the compilation process following the application of the fault tolerance scheme does not introduce new vulnerabilities. The extended backend we have presented here gives this guarantee.

Due to strict safety and reliability requirements, automotive applications may not be able to rely on relaxed fault tolerance schemes. Schemes based on IR transformations only are also not an option due to the remaining vulnerabilities. The extended compiler backend overcomes this problem and thus facilitates complete memory error detection.

7 Acknowledgments

This work was funded by the German Research Council (DFG) through the Cluster of Excellence 'Center for Advancing Electronics Dresden' (cfaed). The authors acknowledge useful discussions with Sven Karol and Tobias Stumpf.

References

[Av71] Avizienis, A.: Arithmetic Error Codes: Cost and Effectiveness Studies for Application in Digital System Design. IEEE Trans. on Computers, C-20(11):1322–1331, 1971.

[Ba05] Baumann, R.: Soft Errors in Advanced Computer Systems. IEEE Design & Test of Computers, 22(3):258–266, 2005.

[Bl06] Blome, J. A.; Gupta, S.; Feng, S.; Mahlke, S.: Cost-efficient soft error protection for embedded microprocessors. In: Proc. Int'l Conf. Compilers, Architecture and Synthesis for Embedded Systems. CASES'06, pp. 421–431, 2006.

[Bo05] Borkar, S.: Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. IEEE Micro, 25(6):10–16, 2005.

[Br60] Brown, D. T.: Error Detecting and Correcting Binary Codes for Arithmetic Operations. IRE Trans. Electronic Computers, pp. 333–337, 1960.

[BSS12] Borchert, C.; Schirmeier, H.; Spinczyk, O.: Protecting the Dynamic Dispatch in C++ by Dependability Aspects. In: Proc. 1st Workshop Software-Based Methods for Robust Embedded Systems. SOBRES'12, 2012.

[BSS13] Borchert, C.; Schirmeier, H.; Spinczyk, O.: Return-Address Protection in C/C++ Code by Dependability Aspects. In: Proc. 2nd Workshop Software-Based Methods for Robust Embedded Systems. SOBRES'13, 2013.

[CNV16] Chen, Z.; Nicolau, A.; Veidenbaum, A. V.: SIMD-based Soft Error Detection. In: Proc. ACM Int'l Conf. Computing Frontiers. CF'16, pp. 45–54, 2016.

[CRA06] Chang, J.; Reis, G. A.; August, D. I.: Automatic Instruction-Level Software-Only Recovery. In: Int'l Conf. Dependable Systems and Networks. DSN'06, pp. 83–92, 2006.

[DS16] Didehban, M.; Shrivastava, A.: nZDC: A compiler technique for near Zero Silent Data Corruption. In: Proc. Design Automation Conf. DAC'16, 2016.

[Es11] Esmaeilzadeh, H.; Blem, E.; Amant, R. St.; Sankaralingam, K.; Burger, D.: Dark silicon and the end of multicore scaling. In: Proc. 38th Ann. Int'l Symp. Computer Architecture. ISCA'11, pp. 365–376, 2011.

[Es12] Esmaeilzadeh, H.; Sampson, A.; Ceze, L.; Burger, D.: Architecture Support for Disciplined Approximate Programming. In: Proc. 17th Int'l Conf. Architectural Support for Programming Languages and Operating Systems. ASPLOS'12, pp. 301–312, 2012.

[Fe10] Feng, S.; Gupta, S.; Ansari, A.; Mahlke, S.: Shoestring: Probabilistic Soft Error Reliability on the Cheap. In: Proc. 15th Int'l Conf. Architectural Support for Programming Languages and Operating Systems. ASPLOS'10, pp. 385–396, 2010.

[Fo89] Forin, P.: Vital Coded Microprocessor Principles and Applications for Various Transit Systems. In: Control, Computers, Communications in Transportation: Selected Papers from the IFAC/IFIP/IFORS Symposium. pp. 79–84, 1989.

[FSS09] Fetzer, C.; Schiffel, U.; Süßkraut, M.: AN-Encoding Compiler: Building Safety-Critical Systems with Commodity Hardware. In: Proc. 28th Int'l Conf. Computer Safety, Reliability, and Security. SAFECOMP'09, pp. 283–296, 2009.

[Ga66] Garner, H. L.: Error Codes for Arithmetic Operations. IEEE Trans. Electronic Computers, EC-15(5):763–770, 1966.

[Gu01] Guthaus, M. R.; Ringenberg, J. S.; Ernst, D.; Austin, T. M.; Mudge, T.; Brown, R. B.: MiBench: A Free, Commercially Representative Embedded Benchmark Suite. In: Proc. IEEE Int'l Symp. Workload Characterization. IISWC'01, pp. 3–14, 2001.

[Ho14] Hoffmann, M.; Ulbrich, P.; Dietrich, C.; Schirmeier, H.; Lohmann, D.; Schröder-Preikschat, W.: A Practitioner's Guide To Software-based Soft-Error Mitigation Using AN-Codes. In: Proc. 15th Int'l Symp. High-Assurance Systems Engineering. 2014.

[HSS12] Hwang, A. A.; Stefanovici, I. A.; Schroeder, B.: Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design. In: Proc. 7th Int'l Conf. Architectural Support for Programming Languages and Operating Systems. ASPLOS'12, pp. 111–122, 2012.

[Ka16] Karol, S.; Rink, N. A.; Gyapjas, B.; Castrillon, J.: Fault Tolerance with Aspects: A Feasibility Study. In: Proc. 15th Int'l Conf. Modularity. 2016.

[KF15] Kuvaiskii, D.; Fetzer, C.: Δ-encoding: Practical Encoded Processing. In: Proc. 45th Ann. Int'l Conf. Dependable Systems and Networks. DSN'15, 2015.

[LA04] Lattner, C.; Adve, V.: LLVM: a Compilation Framework for Lifelong Program Analysis & Transformation. In: Proc. Int'l Symp. Code Generation and Optimization. CGO'04, p. 75, 2004.

[Li11] Liu, S.; Pattabiraman, K.; Moscibroda, T.; Zorn, B. G.: Flikker: saving DRAM refresh-power through critical data partitioning. In: Proc. 16th Int'l Conf. on Architectural Support for Programming Languages and Operating systems. ASPLOS'11, pp. 213–224, 2011.

[Lu05] Luk, C.-K.; Cohn, R.; Muth, R.; Patil, H.; Klauser, A.; Lowney, G.; S. Wallace, Steven; Reddi, V. J.; Hazelwood, K.: PIN: Building Customized Program Analysis Tools with Dynamic Instrumentation. In: Proc. Conf. Programming Language Design and Implementation. PLDI'05, pp. 190–200, 2005.

[MPC14] Mitropoulou, K.; Porpodas, V.; Cintra, M.: DRIFT: Decoupled CompileR-Based Instruction-Level Fault-Tolerance. In: Proc. 26th Int'l Workshop Languages and Compilers for Parallel Computing. LCPC'13, pp. 217–233, 2014.

[NDO11] Nightingale, E. B.; Douceur, J. R.; Orgovan, V.: Cycles, Cells and Platters: An Empirical Analysis of Hardware Failures on a Million Consumer PCs. In: Proc. 6th Conf. on Computer Systems. EuroSys'11, pp. 343–356, 2011.

[OMM02] Oh, N.; Mitra, S.; McCluskey, E. J.: ED4I: Error Detection by Diverse Data and Duplicated Instructions. IEEE Trans. Computers, 51(2):180–199, 2002.

[OSM02] Oh, N.; Shirvani, P. P.; McCluskey, E. J.: Error Detection by Duplicated Instructions in Super-Scalar Processors. IEEE Trans. Reliability, 51(1):63–75, 2002.

[Pa08] Panaroni, P.; Sartori, G.; Fabbrini, F.; Fusani, M.; Lami, G.: Safety in Automotive Software: An Overview of Current Practices. In: Proc. 32nd Ann. IEEE Int'l Computer Software and Applications Conf. COMPSAC'08, pp. 1053–1058, 2008.

[RC16] Rink, N. A.; Castrillon, J.: Comprehensive Backend Support for Local Memory Fault Tolerance. Technical Report TUD-FI-16-04, Technische Universität Dresden, 2016.

[Re99] Rebaudengo, M.; Reorda, M. S.; Torchiano, M.; Violante, M.: Soft-error Detection through Software Fault-Tolerance techniques. In: Int'l Symp. Defect and Fault Tolerance in VLSI Systems. DFT'99, pp. 210–218, 1999.

[Re05] Reis, G. A.; Chang, J.; Vachharajani, N.; Rangan, R.; August, D. I.: SWIFT: Software Implemented Fault Tolerance. In: Int'l Symp. Code Generation and Optimization. CGO '05, pp. 243–254, 2005.

[Ri15] Rink, N. A.; Kuvaiskii, D.; Castrillon, J.; Fetzer, C.: Compiling for Resilience: The Performance Gap. In: Proc. Mini-Symp. Energy and Resilience in Parallel Programming. ERPP'15, 2015.

[Sc10] Schiffel, U.; Schmitt, A.; Süßkraut, M.; Fetzer, C.: ANB- and ANBDmem-Encoding: Detecting Hardware Errors in Software. In: Proc. 29th Int'l Conf. Computer Safety, Reliability, and Security. SAFECOMP'10, pp. 169–182, 2010.

[SGSP02] Spinczyk, O.; Gal, A.; Schröder-Preischkat, W.: AspectC++: An aspect-oriented extension to the C++ programming language. In: Proc. 40th Int'l Conf. Tools Pacific: Objects for internet, mobile and embedded applications. CRPIT'02, pp. 53–60, 2002.

[Sh02] Shivakumar, P.; Kistler, M.; Keckler, S. W.; Burger, D.; Alvisi, L.: Modeling the effect of technology trends on the soft error rate of combinational logic. In: Proc. Int'l Conf. Dependable Systems and Networks. DSN'02, pp. 389–398, 2002.

[Sh14] Shafique, M.; Garg, S.; Henkel, J.; Marculescu, D.: The EDA Challenges in the Dark Silicon Era: Temperature, Reliability, and Variability Perspectives. In: Proc. 51st Ann. Design Automation Conf. DAC'14, pp. 1–6, 2014.

[SPW09] Schroeder, B.; Pinheiro, E.; Weber, W.-D.: DRAM Errors in the Wild: A Large-scale Field Study. In: Proc. 11th Int'l joint Conf. Measurement and Modeling of Computer Systems. SIGMETRICS'09, pp. 193–204, 2009.

[Ta12] Taylor, M. B.: Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse. In: Proce. 49th Ann. Design Automation Conf. DAC'12, pp. 1131–1136, 2012.

[We15] Weis, C.; Jung, M.; Ehses, P.; Santos, C.; Vivet, P.; Goossens, S.; Koedam, M.; Wehn, N.: Retention Time Measurements and Modelling of Bit Error Rates of WIDE I/O DRAM in MPSoCs. In: Proc. Design, Automation & Test in Europe Conf. & Exhibition. DATE'15, pp. 495–500, 2015.

[YGS09] Yu, J.; Garzarán, M. J.; Snir, M.: ESoftCheck: Removal of Non-vital Checks for Fault Tolerance. In: Proc. 7th Ann. Int'l Symp. Code Generation and Optimization. CGO'09, pp. 35–46, 2009.

[Zh10] Zhang, Y.; Lee, J. W.; Johnson, N. P.; August, D. I.: DAFT: Decoupled Acyclic Fault Tolerance. In: Proc. 19th Int'l Conf. Parallel Architectures and Compilation Techniques. PACT'10, pp. 87–98, 2010.

Exploring and Understanding Multicore Interference from Observable Factors

Benjamin Lesage[1], David Griffin[1], Iain Bate[1] and Frank Soboczenski[1]

Abstract:

Multi-core processors bring a wide variety of challenges to the development, maintenance and certification of safety-critical systems. One of the key challenges is to understand how tasks sharing the processing resource affect one another, and to build an understanding of existing or new platforms. Industry reports that interference can lead to large variations in execution times which can lead to a wide variety of problems including timing overruns. To support performance improvements, debugging and timing analysis, a framework is presented in this paper for reliably establishing the interference patterns of tasks using simple contenders. These contenders systematically manipulate the shared resources so the effect on interferences can be understood and analysed. The approach relies on guided exploration of the interference space and existing performance monitoring infrastructure. It has been implemented on a Tricore AURIX platform to analyse the behaviour of multiple real and kernel applications.

Keywords: Inter-core interferences, timing analysis, Shared resources, measurement-based, performance monitoring

1 Introduction

The drive for performance in modern systems has to face an increasing energy wall; merely increasing clock speeds to achieve higher performance is no longer a viable solution. On the other hand, multicore platforms offer improved performances through the use of multiple processing units on the same chip. Tasks are running concurrently on different cores while sharing off-core resources such as lower levels of the memory hierarchy or external IO channels. The introduction of resources shared by concurrent tasks introduce new sources of interferences. Accesses to a shared resource may suffer arbitration delay when multiple cores compete for said resource. The behaviour of a request may also be altered due to interferences, e.g. the eviction of a cache block by a concurrent task.

Preemption-related interferences, similarly to inter-core interferences, exhibit both direct and delayed effects after the occurrence of an interference event. Inter-core interferences however do not occur as a single point of interference, as opposed to a preemption, but interleave with the normal execution of a task. During inter-core interference analysis, this requires to more pessimistic assumptions where each request is assumed to be in conflict with a request from a co-runner [Al15], or complex models of the underlying architecture requiring precise knowledge of its behaviour [JHH15]. Various mechanisms have been

[1] Department of Computer Science, University of York, York, United Kingdom

proposed instead to limit or preclude interferences in time and space between concurrent cores, such as cache partitioning [SM08] or leaky bucket schemes [Ji13]. Such schemes require often costly support from the underlying platform. To identify the solution adapted to a specific task, a necessary step is the building of an understanding of its sensitivity to interferences and the underlying factors.

Modern architectures exhibit performance monitoring counters (PMCs) as a tool to evaluate the behaviour of a task. PMCs provide an estimate of the occurrences of low-level events on the platform. The information they expose can be used in particular to evaluate how often and how much a task makes use of particular resource, private or shared [An97], with minimal knowledge on the platform. As an example, consider an architecture with a single private instruction cache on top of a shared SRAM memory. Without further knowledge of the platform, one can safely assume that a task with a large observed miss rate is more likely to access the shared memory and be sensitive to inter-core interferences. Some factors are thus more pertinent than others to the execution time variations of a task. Moreover, there are typically more events to be counted than registers available to track those events.

This paper proposes an approach to identify the main factors of variability in the temporal behaviour of a task. The analysis operates without knowledge of the underlying platform and the implemented policies for shared resources. The technique exposes the PMCs which are tied to variations in the execution time of a task. Through systematic and reproducible exploration of the interference space, this allows the isolation of the effect of interferences. The selection of most relevant factors provides feedback on the sources of interferences that need to be tackled to improve the predictability of a task's behaviour and the robustness a task or the system as a whole w.r.t. interferences. Focused testing on the factors identified by the analysis can further provide for a partial multi-variate model of the temporal behaviour of a task in relation to said factors. We provide general guidelines on how synthetic contenders can exercise inter-core interference in the analysed system.

2 Related Work

Current state of the art techniques for the analysis of the effects of inter-core interferences focus on either taking into account possible effects or precluding them. The work of Altmeyer and al. [Al15] belongs to the first category. The authors propose a generic framework to compute the response time of task, taking into account the delays that might rise from the arbitration of accesses to the shared bus. While focused on a single shared resources, the technique needs to take into account a wide range of effects, such as memory refreshes, cache hits and misses, to build a reasonable model of interleaved accesses. More integrated approaches [JHH15] further improve the precision of the estimates but require an extensive knowledge of the underlying platform or complex models capturing a large portion of the system state.

Approaches such as partitioning divide the shared resource space into segments dedicated to the sole benefit of a single task. They focus on precluding interferences at the expense of limiting the resources available to each task [Al14; SK11; SM08]. Leaky bucket schemes [Ji13]

offer an alternative solution to reduce interferences. Accesses to a shared resource by a task are budgeted to limit the amount of interferences they might generate in a given time interval. In either case, the interference problem becomes an optimisation one to derive the budget allocated to each task in the system, satisfy the system's constraints, and maximise the use of the shared resource. Such techniques should therefore be applied with care, and their underlying assumptions and impact empirically validated. As an example, considerable interferences might still occur in partitioned caches due to shared miss handling status registers [VYF16].

Radojković et al. [Ra12] evaluate the effect of inter-core interference from co-runners through empirical experiments. Their work demonstrates that shared resources can contribute to large variations in the temporal behaviour of a task. Their evaluation relies on resource stressing kernels, thus identifying the slowdown which may be induced by a specific component on the platform. We instead aim at identifying which components or combinations thereof contribute to variations in the behaviour of a specific task. This reduces the pessimism of the following analyses by focusing on the components known to impact the analysed task. Testing can then proceed by focusing on said components, e.g. allowing for the empiric validation of selected inter-core interference management methods. Our approach also relies on a wider exploration of the interference space as the worst-case scenarios do not stem from stressing a single resource.

3 Overview

Our approach to evaluate the impact of inter-core interferences on the execution of a task can be broken down into simple steps. Data is first collected by running the task of interest against selected competitor tasks while collecting data related to both execution time and as many performance counters as possible. All collected measurements capture the end-to-end behaviour of the analysed task; instrumentation primitives surround the analysed task. We then identify a set of representative factors to understand which factors drive variations in the execution of the analysed task, and whether or not they relate to inter- or intra-core effects.

Without prior knowledge of the usefulness of PMCs, it is necessary to build a small dataset with all PMCs in order to determine their usefulness. To this end in Section 6, Principal Components Analysis is used for an automatic feature selection phase, i.e. to find a set of PMCs which is capable of representing the variability of the data. The identification of the PMCs relevant to the analysed task can help direct later testing phases, to evaluate selected inter-core interferences management approaches or build a model of the analysed task. Further data collection phases and the exercised contenders can be focused on the factors known to contribute to variability in the behaviour of the analysed task. Without refining the selection of components exercised during analysis, more different types of contenders need to be considered to exercise all possible sources of interference. This is turn leads to more testing or less significant data available.

While the factor selection is platform-agnostic (§ 6), the available factors and their interpretation depend on the underlying system. The methods thus relies on platform-specific

instrumentation. Instrumentation and contenders are expected to respectively capture the available PMCs alongside timing information and exercise the different sources of variability in the platform. We discuss the requirements inherent to those steps in Section 5. To illustrate our approach, we focus in the following on an Infineon AURIX Tricore platform as presented in the next section.

4 Evaluation Platform

Our evaluation platform is composed of the OSEK/VDX compliant Erika Enterprise [En16] real-time operating system running on top of an Infineon AURIX Tricore TC27x [In14]. Figure 1 outlines the architecture of the AURIX platform. The AURIX cores have different capabilities and fulfil different roles in the system:

- Core 0: Error checking, Energy Efficient Tricore 1.6E core
- Core 1: Error checking, High Performance Tricore 1.6P core
- Core 2: No error checking, High Performance Tricore 1.6P core

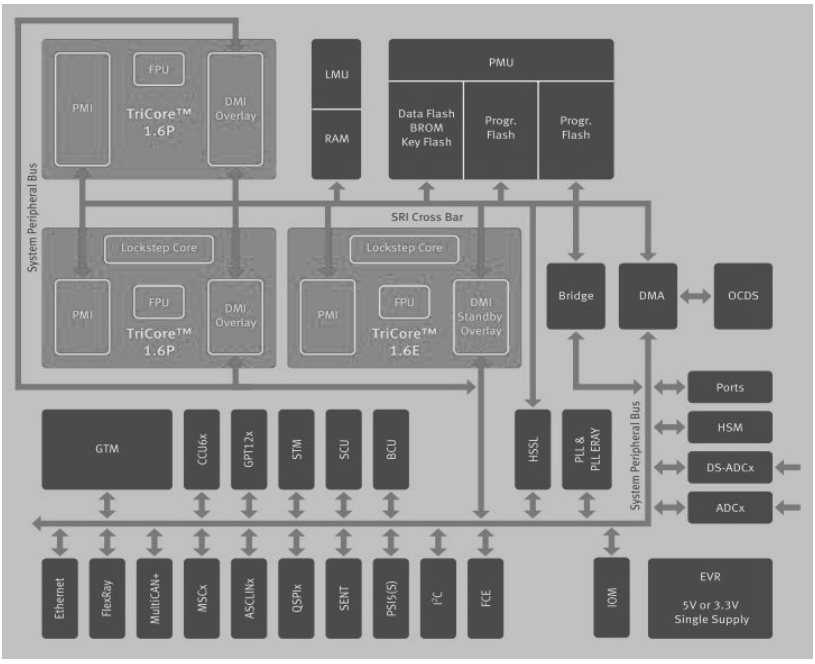


Fig. 1: Overview of the Infineon AURIX Tricore platform.

Each core has access to a crossbar which connects a SRAM unit, flash memories, and external peripherals through a bridge. Interference across cores typically stem from concurrent accesses to either of those resources, e.g. congestion on the flash due simultaneous

requests requiring arbitration. In the following, we focus on interferences caused by either the SRAM and one of the code ROM. The default memory mapping in Erika does not map data into the secondary ROM or segments located in remote scratchpads.

Core 1 and 2 expose 12 PMCs, 9 in the case of Core 0, which have the capability to monitor performance metrics such cache hits/misses, executed branches, or stalls in the pipeline. Only 3 registers are available per core to track PMCs, and each PMC can only be tracked by a specific register. This restricts the set of PMCs which can be monitored during a single run. For example, there is no configuration of said registers which allows tracking both data and instruction cache misses for Core 1.

In the best case, to capture data on all PMCs from the AURIX, it would be necessary to run each test four times. This is undesirable in that it increases the amount of testing that is required from the user to understand the impact of interferences on a task. Other platforms may expose more PMCs which renders this approach infeasible for large sets of tests[3]. Redundancy between factors or a lack of correlations to interferences further reinforces the need to reduce the set of collected PMCs to a significant one. Consider for example a computationally intensive program mapped into its core local scratchpads. Monitoring its cache hits will not yield conclusive results.

5 Data Collection

Our approach relies on task-level instrumentation to capture end-to-end timing traces. The collected traces include both the execution time of the analysed task and related metrics pertinent to the behaviour of the platform. We thus rely on the PMCs exposed by the hardware. PMCs expose counts for specific events or latencies suffered by a task, e.g. the number of executed branches acts as an observable proxy for the path executed during an observation. Increases in stalls suffered by the load/store unit can be indicative of increased contention in the shared memory. Our implementation automates the whole trace collection process, allowing for the automated collection of traces capturing all available PMCs on the platform or a selection thereof.

The PCA requires the presence for each run of all available factors, PMCs on the target platform, to establish their relation to the execution time of the analysed task. The analysed task is ran multiple times, under the same inputs, merging the results obtained for identical runs but different configurations of PMCs. While we aimed at improving the reproducibility of the framework between runs, the platform still exhibit some uncontrolled sources of variability, e.g. uninitialised values on processor start. Those prevent the perfect reproduction of the runs of a task. We validated that the error between reproduced runs is both minimal and characterised as random noise, by fixing a performance counter and comparing its value across identical runs and varying PMCs configurations. The error is minimal ($< 5\%$). The use of the Wald-Wolfowitz [St06] further confirmed that it could be reasonably characterised as random noise and would not introduce systemic failings [St06].

[3] For example, the P4080 platform [Se], which we have also applied the technique to, exposes approximately 128 PMCs with 4 registers per core, and would require each experiment to be repeated 32 times.

A dedicated instrumentation buffer is used to log the timing and PMCs values on each core. A full instrumentation buffer interrupts all execution on the platform and triggers the collection of the data through the debugger interface. The same debug interface is used to configure the PMCs exercised during a set of runs. A single binary and test vector can thus be used to collect different PMCs. Instrumentation buffers are mapped into a debug memory segment, itself mapped onto local scratchpads during analysis or unmapped on a deployed system. Writes to the unmapped debug segment are simply discarded by the platform. The event instrumentation routines can therefore be kept in the deployed system. Each request for an instrumentation point is broadcast to all cores in the system to capture PMCs across all cores.

5.1 Synthetic contenders

We developed a set of synthetic contenders to drive the exploration of the possible inter-core interference configurations. The contenders aim to exert the variability inherent to the analysed task in reaction to inter-task conflicts. Knowledge about the potential sources of variability in a system is required to exert them in a significant way. As such, contenders are strongly platform-dependent. While contenders for one platform may not apply to another, similar principles apply, e.g. varying accesses across cache lines, cache sets or physical pages, interleaving sequences of reads or writes, etc.. The use of the complete system as deployed would help understand the main sources of variability in the analysed task, but may not highlight the impact of contention should it suffer a constant interference rate.

A single set of platform-wide contenders has been derived for the AURIX. Controlled accesses to the shared memory segments, through non-cacheable addresses, are used to generate interferences. Those are restricted to the Shared RAM and a segment of flash. Given the default memory mapping implemented by the OSEK/VDX-compliant Erika OS, a core is restricted to either its local scratchpads, the shared memory, or one of the flash segments. While the method is not restricted to a specific platform, taking into consideration its underlying restrictions helps reducing the configurations that need to be considered during testing. Contending accesses interleave with non-interfering one; accesses by a core to its scratchpad do not contribute to the overall inter-core interferences. Each contender loops upon a determined access sequence to generate a controlled, continuous amount of contention. To preclude any impact on the functional behaviour of the analysed tasks and preserve data coherency, there is no sharing of data between contenders and analysed tasks.

The level of interference generated by a contender is expressed as and controlled by the portion of its instructions generating contention. Both the interference level and pattern exercised by a contender are set dynamically within user-defined bounds. An interference level is first randomly selected. Then a permutation of instructions is generated to select the conflicting ones. The code of each contender is stored in a local scratchpad such that it can be rewritten to enforce the selected interference pattern. The interference patterns exercised by a contender during an experiment can easily be reproduced.

Contenders run on all cores effectively acting as an idle task in the system. On each core a preemptive task with the lowest priority runs the main loop of a contender. Variation in the

observed interference patterns relies on periodic reconfiguration of the contenders, upon a signal from the analysed task. The process can proceed without explicit synchronisation between contenders and analysed task. This signal is triggered at the end of the periodic analysed task allowing for the reconfiguration to occur between activations of the task. Like the instrumentation routines, the primitives can thus be kept into the deployed system.

6 Feature Selection

Due to the impracticality of capturing vast amounts of data for each and every PMC, as well as a desire to focus on high-quality PMCs, it is necessary to reduce the number of observed PMCs[4]. The goal for this step is to identify the PMCs which are correlated, and then select a set of representative PMCs which can be captured in a single configuration while still describing the majority of the data. While it is inevitable that some detail in the data will be lost at this stage, the reduction in the amount of effort required to get a single data point enables more data to be collected.

In order to accomplish this, we use the technique of PCA [Jo02]. PCA is a technique which identifies correlations within a dataset by finding the *Principal Components* (PCs) of the data. Each PC describes one of the main axes of variance in the analysed dataset such that variations on each axis can be attributed to a specific set of factors. Correlated factors are thus captured as part of the same PC. For example, the main axis of variation on a data-sensitive application, its main PC, will include factors such as hits in the cache or stalls in the memory units. This is further illustrated in Figure 2, which shows the main axes of variations in a 2-dimensional dataset; PC1 captures the axis along which the majority of the variance in the data occurs.

Additional metrics are attached to the PCs and each factor inside a PC to measure their respective impact on the whole dataset and the PC. The PCs specify a loading on each factor that indicates the weighting that must be assigned to its observations such that they lie on the axis defined by the PC. In other words, the loading of a factor on a PC captures its correlation to the PC, how it evolves alongside the axis defined by the PC. A loading of near 0 indicates that the observations are not correlated on the PC, whereas loadings of 1 and -1 indicate perfect positive and negative correlation respectively, i.e. a factor which values respectively increase or decrease with values along the PC. Considering the same example of a memory intensive task, accesses to the bus from the task or contenders are likely to increase its execution time, thus being positively correlated to the principal components. Conversely, a decrease in executed integer instructions may be correlated to an increase in the memory traffic, and the task's execution time. In the Figure 2 example, x has a high loading in PC1, indicating a high degree of correlation to PC1, but a much lower loading on PC2.

PCs themselves have an overall magnitude assigned to them which can be seen as a proxy for their contribution to the variations in the dataset, i.e. the amount of variance in the dataset captured upon the axis of the PC. The right-hand side of Figure 2 shows how it is

[4] In statistical literature, this is commonly referred to as *dimensionality reduction* or *feature selection*.

possible to reduce the 2-dimensional dataset to a single dimension on which accounts for the majority of the variability.

A standard use of PCA is to identify which PCs do not significantly contribute to the overall distribution of a dataset using their respective loadings. For example, if a PC accounts for less than 10% of the variance in the entire dataset, as accounted for by its loading, then variation along that PC can be simply dismissed as sampling error and the PC ignored; this quickly and simply reduces the number of dimensions in the analysed data set. However, in this application it may be necessary to reduce the number of factors further, and select only the high quality PMCs which yield information on the observed interferences multiplier. Hence it is necessary to filter the PMCs further, such that only the highest quality PMCs are used.

The first step to finding the highest quality PMCs is to remove all PCs which are not correlated to the execution time of the task under analysis, as these are unlikely to yield useful information. This relies on the absolute loading of the analysed task's execution time on the PC to measure their correlation. In addition, components which explain a low amount of the overall variance are removed, as these represent factors which are unlikely to have a high impact on the result. The remaining components are weighted by the amount of variance they explain, through their respective loadings, and then the PMCs with the highest degree of correlation to these components are selected. For example, if 2 components A, B remain, of which A explains 60% of the variance and B 30%, then for each PMC that is selected correlated to B, two PMCs will be selected that correlate to A. The exact PMCs selected will be those with the highest correlation to the components A and B respectively.

It is also possible to add additional constraints to factor selection. Depending on the platform, and the amount of effort a user is prepared to undertake when gathering data, it may be desirable to place a restriction on the number of runs required to gather the data. This may not always be the case, or the user may not be able to expend the additional effort to capture them. Therefore, if the user wishes to impose additional constraints to reduce the burden of data gathering, these constraints should be formulated and applied at this stage. This is implemented by using a Integer Linear Programming (ILP) solver [Gu15] to perform the maximisation step, and so any ILP constraints can be used.

A pseudocode implementation of the application of PCA in our approach to select n relevant factors is given in Algorithm 1. Once PCA has identified which PMCs must be collected, the main data collection process can now take place and is only required to record values for these PMCs, which reduces the burden of instrumentation.

Quality of the selected factors and contenders

One issue that may be encountered during feature selection is the selection of poor quality factors. This can happen if user constraints prevent high quality factors from being selected, or high quality features simply don't exist. If this is the case then the outcome of the algorithm may be a limited number of factors ($< n$) or factors with a low correlation to the execution time of the analysed task.

```
Function GetBestPMCs(dataset, n)
    P ← PCs of dataset given by PCA
    discard any c ∈ P not correlated with execution time
    discard any c ∈ P not accounting for a substantial amount of variance (e.g. < 10%)
    compute relative weighting of remaining c ∈ P
    select n PMCs maximising the sum of loadings subject to the weightings (and additional user
      constraints) by ILP solver
    return PMCs
end
```

Algorithm 1: Pseudo-code implementation of PCA as used to extract the best possible PMCs for instrumentation

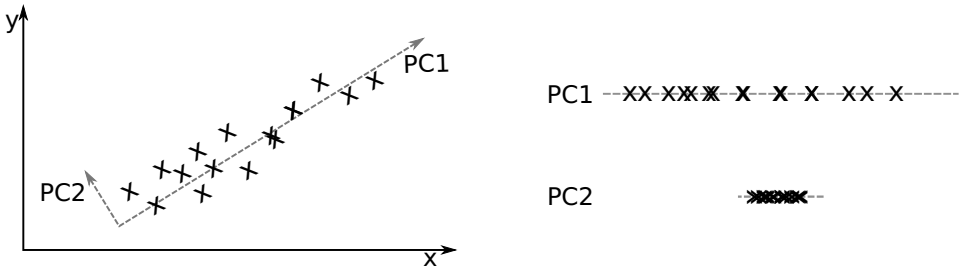


Fig. 2: Graphical Example of PCA

The use case for the analysis also drives requirements on the exercised contenders during the construction of the analysed dataset. If the observations focus on a subset of shared resources of interest, the selected factors then offers a relative classification of those resources which contribute the most to variations in the analysed task. Similarly for a conclusive feature selection, it is important for a contender to exercise various sources of interferences; focus on a single of the shared resources of interest, e.g. the shared memory, may lead to orthogonal variations of available PMCs hindering the analysis process.

Variability within the analysed dataset is an important factor to discriminate the PMCs that correlate to the behaviour of the analysed task. The exercised contenders should aim not only at generating worst-case interference scenarios but also produce a wide gamut of scenarios. Our feature selection focuses on factors correlated to the execution time of the analysed task. Therefore, variation on the inputs, executed paths, and observed scenarios is only captured by the analysis if it has a noticeable impact on the analysed task's temporal behaviour.

7 Evaluation

We evaluated our approach on various benchmarks deployed on the Erika OS running atop the AURIX platform, as described in Section 4. Two sets of benchmarks were investigated: simple examples from the Taclebench suite [Co] and three real-world applications. The

collection of end-to-end timing and PMCs values relies on the instrumentation of the analysed task, i.e. the insertion of a call to the instrumentation routine before and after calls to the analysed task. We use the Rapita Verification Suite [Ra] to that purpose. All benchmarks are set to run in a single periodic task concurrently with our synthetic contenders. Input vectors are either provided as part of the benchmark [Co] or randomly generated during our experiments. Similarly, explored interference levels and patterns are randomly generated for each execution before each execution of the analysed task. To enforce the occurrences of inter-core interferences, portions of the data manipulated by some benchmarks have been mapped into the shared memory.

7.1 Understanding the sources of execution time variability

We present the factors identified as relevant to the variability of a selection of benchmarks in Table 1. The temporal variability for a benchmark is captured by the ratio between the maximum and minimum observed execution time during our experiments (in the last column). The real-world benchmarks investigated were *missile-c*, a missile control program converted from Ada to C [Hi], and the *powerwindow* and *lift* benchmarks from the Taclebench [Co] suite. ALU, LSU, LU stalls represents stalls in the Arithmetic, Load/Store and Loop unit respectively. Each factor is prefixed by its measuring core.

The largest ratios between maximum and minimum execution times are observed as an example for *anagram*, *binarysearch*, *dijkstra*, and *missile_c*. Our analysis selects in such cases factors such as the number of executed branches, multiple issues, or stalls in the ALU, as most relevant to the observed variations. This suggests that variability in these benchmarks stems first and foremost from the observations capturing different execution paths; variations in the execution time of such tasks is not driven by the variations in inter-core interferences but changes in executed paths and input vectors. The analysis of the impact of inter-core interferences on those benchmarks should therefore distinguish between different execution scenarios. Considering *dijkstra* as an example, the path searching algorithm includes a short, special case if the source and target nodes are the same.

The identification of factors from other cores as relevant, e.g. C2 Data Memory Stalls for *compressdata* or *matmult*, is an indicator of the sensitivity of an application to variations in the behaviour of concurrent tasks as triggered by our contenders. The strong contribution of factors such as memory stalls further identifies those tasks as data intensive applications. On *compressdata*, variations due to contentious data accesses further trumps those due to small variations in the execution path. Further testing should therefore focus on contenders exercising the shared memory to exploit variability in these benchmarks.

The results of the application of the analysis to different benchmarks and cores also points towards an asymmetry on the platform. Tasks running on Core 1 are more likely to be impacted by contenders on Core 2 than on Core 0; factors from Core 2 are more often selected on their own for benchmarks running on Core 1 than factors from Core 0. This may stem from lower contention levels from the energy efficient Core 0, or asymmetry in the arbitration of accesses to the shared resources.

Tab. 1: Representative factors identified for each benchmark.

Benchmark	Core	Select Factors		Max/Min runtime ratio
MÄLARDALEN				
adpcm_encoder	C0	C0 Data Memory Stalls	C0 ALU Stalls	1.00276
		C0 Executed branches	C2 LSU Stalls	
adpcm_encoder	C1	C1 Data Memory Stalls	C1 Executed branches	1.00412
		C2 Data Memory Stalls	C2 ALU Stalls	
anagram	C1	C1 Data Memory Stalls	C1 LSU Stalls	3.4214
		C1 Multiple instructions issue	C1 Executed branches	
binarysearch	C0	C0 Data Memory Stalls	C0 ALU Stalls	1.85784
		C0 Executed branches	C2 Data Memory Stalls	
binarysearch	C1	C0 Data Memory Stalls	C1 Data Memory Stalls	1.9
		C1 ALU Stalls	C1 Executed branches	
bitcount	C0	C0 Data Memory Stalls	C0 ALU Stalls	1.2179
		C0 Executed branches	C1 LSU Stalls	
bitcount	C1	C1 ALU Stalls	C1 LSU Stalls	1.22467
		C1 Executed branches	C2 LSU Stalls	
codecs_dcodrle1	C0	C0 Data Memory Stalls	C0 LSU Stalls	1.43177
		C0 Executed branches	C2 Data Memory Stalls	
codecs_dcodrle1	C1	C1 Data Memory Stalls	C1 LSU Stalls	1.53956
		C1 Multiple instructions issue	C1 Executed branches	
compressdata	C0	C0 Data Memory Stalls	C0 Executed branches	1.55328
		C1 Data Memory Stalls	C2 Executed branches	
compressdata	C1	C0 Data Memory Stalls	C1 Data Memory Stalls	1.63359
		C1 Executed branches	C2 Executed branches	
countnegative	C0	C0 Executed branches	C1 Data Memory Stalls	1.25762
		C1 LSU Stalls	C2 Data Memory Stalls	
countnegative	C1	C1 LSU Stalls	C1 Multiple instructions issue	1.06209
		C1 Executed branches	C2 LSU Stalls	
dijsktra	C0	C0 ALU Stalls	C0 Executed branches	1577.23
		C1 LSU Stalls	C2 LSU Stalls	
dijsktra	C1	C1 Data Memory Stalls	C1 LSU Stalls	147.27
		C1 Multiple instructions issue	C1 Executed branches	
duff	C0	C0 Data Memory Stalls	C0 ALU Stalls	1.51672
		C1 Data Memory Stalls	C2 Data Memory Stalls	
duff	C1	C0 Data Memory Stalls	C1 Data Memory Stalls	1.51595
		C1 ALU Stalls	C2 Data Memory Stalls	
matmult	C0	C0 Data Memory Stalls	C1 Data Memory Stalls	1.32437
		C2 Data Memory Stalls	C2 Multiple instructions issue	
matmult	C1	C1 Data Memory Stalls	C1 LSU Stalls	1.31893
		C2 Data Memory Stalls	C2 Multiple instructions issue	
ndes	C0	C0 Data Memory Stalls	C0 ALU Stalls	1.09869
		C0 LSU Stalls	C1 Data Memory Stalls	
ndes	C1	C1 Data Memory Stalls	C1 ALU Stalls	1.112
		C1 LSU Stalls	C2 Data Memory Stalls	
qurt	C0	C0 Data Memory Stalls	C0 ALU Stalls	1.35984
		C0 Executed branches	C2 Data Memory Stalls	
qurt	C1	C1 Data Memory Stalls	C1 ALU Stalls	1.33716
		C2 Data Memory Stalls	C2 Executed branches	
rijndael_encoder	C0	C0 Data Memory Stalls	C2 Data Memory Stalls	1.03466
		C2 Multiple instructions issue	C2 Executed branches	
rijndael_encoder	C1	C1 Data Memory Stalls	C1 LSU Stalls	1.0344
		C2 Data Memory Stalls	C2 Multiple instructions issue	
statemate	C0	C0 Data Memory Stalls	C0 LSU Stalls	1.00501
		C0 Executed branches	C2 LSU Stalls	
statemate	C1	C1 LSU Stalls	C1 Multiple instructions issue	1.00509
		C1 Executed branches	C2 LSU Stalls	
st	C0	C0 Data Memory Stalls	C0 ALU Stalls	1.10449
		C1 Data Memory Stalls	C1 Multiple instructions issue	
st	C1	C1 Data Memory Stalls	C2 Data Memory Stalls	1.13961
		C2 Multiple instructions issue	C2 Executed branches	
REAL-WORLD EXAMPLE				
lift	C0	C0 Data Memory Stalls	C0 ALU Stalls	1.46235
		C0 Executed branches	C2 Multiple instructions issue	
lift	C1	C1 ALU Stalls	C1 LU Stalls	
		C1 Multiple instructions issue	C1 Executed branches	1.48444
missile_c	C0	C0 Data Memory Stalls	C0 ALU Stalls	24.3645
		C0 LSU Stalls	C0 Executed branches	

7.2 Exploring the impact of inter-core interferences

We focus in the following on the *matmult* benchmark running on Core 0. The application comprises a single path computing the multiplication of two matrices mapped into the main memory. Variability in the temporal behaviour of the benchmark is thus mostly related to inter-core interferences and accesses to the shared main memory, as captured by our method in Table 1. Using our framework, we collect observations under different configurations of interference levels, i.e. the portion of accesses to the shared memory, from each core. The collected execution times are then normalised over the execution time for *matmult* in isolation, without contenders. The median of the observed execution times for each inte rference level are presented in Figure 3.

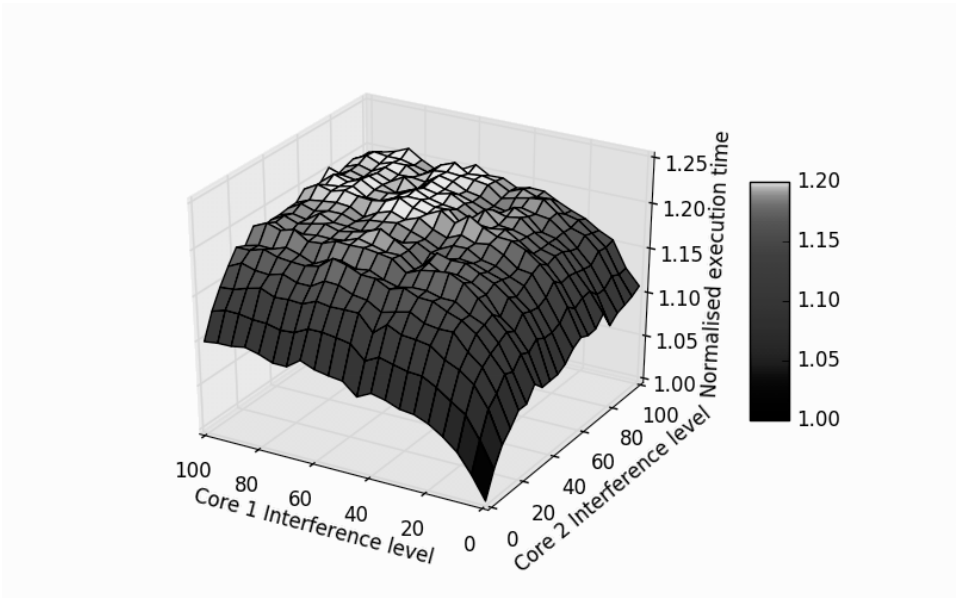


Fig. 3: Normalised execution time for *matmult* under inter-core interferences

As expected, the execution time of the *matmult* benchmark increases alongside the interferences generated by the synthetic contenders. This is however not a strictly increasing curve; high execution times are more likely to be observed when about 80% of Core 1 memory accesses hit the shared memory, and at least 60% of Core 2 do. This reinforces the observation that the arbitration policy on the AURIX shared memory may be asymmetric.

Furthermore, maximising the interferences generated by other cores does not guarantee maximising the impact on the analysed task as contending cores start interfering with themselves, restricting their maximum bandwidth to the main memory, and each other. Similarly, focusing solely on the impact of a single source of interferences at a time, e.g. Core 1, does not lead to maximised observed execution time. To understand the impact of interferences, we advocate the need to explore a wide variety of configurations, in terms of types of interferences but also strength of those interferences.

8 Conclusion

This paper introduces a feature selection approach to understand the main source of variability in an application. Our approach draws the relation between the temporal behaviour of an application and the observable factors on the platforms through the performance monitoring infrastructure. We focus on the impact of inter-core interferences stemming from the use of shared resources by concurrent tasks. To exercise a sufficient level of variability, we rely on synthetic, configurable contenders to exercise different interference patterns, sources, and levels.

We implemented our approach on the OSEK/VDX-compliant Erika OS running atop the Tricore AURIX TC277x platforms. Our framework allows the joint collection of timing and PMCs information, under user-controlled interference ranges. The evaluation demonstrates that the method is able to classify the main sources of variability in different categories of applications, from control code to more data-centric kernels. Using such a simple kernel, we further illustrated the importance of variability in the test conditions to highlight variability and the worst-case configurations in the analysed task.

We evaluated our process on other platforms, such as the Freescale P4080, and plan to expand to other architectures or sources of interferences. Our approach requires only minimal knowledge of the underlying platform. Namely, potential sources of interferences need to be identified and contenders designed to exercise them. Work is further required to interpret the meaning behind the PMCs selected by the analysis. However, similar design principles still apply across platforms, such as varying access patterns, data-centric kernels, etc., and similar performance monitoring infrastructure are available.

Acknowledgments

This work was partially funded by EU FP7 IP PROXIMA (611085), and the UK EPSRC Project MCCps (EP/P003664/1). EPSRC Research Data Management: No new primary data was created during this study.

References

[Al14] Altmeyer, S.; Douma, R.; Lunniss, W.; Davis, R. I.: Evaluation of Cache Partitioning for Hard Real-Time Systems. In: 2014 26th Euromicro Conference on Real-Time Systems (ECRTS). July 2014.

[Al15] Altmeyer, S.; Davis, R. I.; Indrusiak, L.; Maiza, C.; Nelis, V.; Reineke, J.: A Generic and Compositional Framework for Multicore Response Time Analysis. In: Proceedings of the 23rd International Conference on Real Time and Networks Systems. RTNS, 2015.

[An97] Anderson, J. M.; Berc, L. M.; Dean, J.; Ghemawat, S.; Henzinger, M. R.; Leung, S.-T. A.; Sites, R. L.; Vandevoorde, M. T.; Waldspurger, C. A.; Weihl, W. E.: Continuous Profiling: Where Have All the Cycles Gone? ACM Trans. Comput. Syst. 15/4, Nov. 1997.

[Co] Contributors: Taclebench Benchmark Suite.

[En16] Enterprise, E.: ERIKA Enterprise | Open source RTOS Osek/VDX Kernel, 2016, URL: http://erika.tuxfamily.org/drupal/.

[Gu15] Gurobi Optimization, I.: Gurobi Optimizer Reference Manual, 2015, URL: http://www.gurobi.com.

[Hi] Hilton, A.: SPARK Missile Guidance Simulator.

[In14] Infineon: Aurix (TM) Family TC27xT Documentation, 2014, URL: www.infineon.com/aurix.

[JHH15] Jacobs, M.; Hahn, S.; Hack, S.: WCET Analysis for Multi-core Processors with Shared Buses and Event-driven Bus Arbitration. In: Proceedings of the 23rd International Conference on Real Time and Networks Systems. RTNS, ACM, New York, NY, USA, 2015.

[Ji13] Jing, W.: Performance Isolation for Mixed Criticality Real-time System on Multicore with Xen Hypervisor, MA thesis, Uppsala University, Department of Information Technology, 2013.

[Jo02] Jolliffe, I.: Principal component analysis. Wiley Online Library, 2002.

[Ra] Rapita Systems: Rapita Verification Suite, https://www.rapitasystems.com/.

[Ra12] Radojković, P.; Girbal, S.; Grasset, A.; Quiñones, E.; Yehia, S.; Cazorla, F. J.: On the Evaluation of the Impact of Shared Resources in Multithreaded COTS Processors in Time-critical Environments. ACM Transactions on Architecture and Code Optimization 8/4, 34:1–34:25, Jan. 2012.

[Se] Semiconductor, F.: EREF: A Programmer's Reference Manual for Freescale Embedded processors.

[SK11] Sanchez, D.; Kozyrakis, C.: Vantage: scalable and efficient fine-grain cache partitioning. In: SIGARCH Computer Architecture News. Vol. 39. 3, ACM, pp. 57–68, 2011.

[SM08] Suhendra, V.; Mitra, T.: Exploring locking: partitioning for predictable shared caches on multi-cores. In: Design Automation Conference (DAC). 45th ACM/IEEE. June 2008.

[St06] Stephens, L. J.: Schaum's Outlines: Beginning Statistics. McGraw-Hill, 2006.

[VYF16] Valsan, P. K.; Yun, H.; Farshchi, F.: Taming Non-Blocking Caches to Improve Isolation in Multicore Real-Time Systems. In: IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). Apr. 2016.

A Testing Framework Architecture Concept for Automotive Intrusion Detection Systems

Christopher Corbett[1], Tobias Basic[2], Thomas Lukaseder[3], Frank Kargl[3]

Abstract: Vehicles are the target of a rising number of hacking attacks. The integration of in-vehicle intrusion detection systems is a common approach to increase the overall system security. However, testing and evaluating these systems is difficult due to the lack of tools to generate realistic benign and malicious workloads as well as sharing these workloads with other researchers. Currently, testing tools are predominantly intended for Network Intrusion Detection System (NIDS) in company or industrial networks where their usefulness became apparent. Yet, in the automotive domain, development of testing tools is still in the early stages. Existing non-commercial automotive tools only focus on one specific bus technology each. However, in-vehicle communication exceeds bus technology boundaries and a testing tool must cover multiple technologies. We propose a framework architecture concept for in-vehicle NIDS testing and evaluation to enable the creation of realistic network traffic and attacks in consideration of automotive specific challenges. Our concept provides the opportunity to share data without additional anonymization effort therefore improving cooperation and reproducibility of testing results.

Keywords: automotive, network, IDS, evaluation, security, framework

1 Introduction

Automotive networks are essential for both driver assistance and future trends such as autonomous driving. With increasing complexity and rising numbers of network devices, the possible impact of malicious manipulation and malfunction also increases. Additional network bandwidth is mandatory to cover new functional requirements and cannot be met with traditional bus systems such as the Controller Area Network (CAN). Automotive Ethernet is designed to tackle these problems and — in addition to the necessary bandwidth — provides greater flexibility with regard to higher layer protocols.

A rising number of attacks on vehicles (e.g. [MV15],[RM15]) emphasizes the need for more security precautions and extended protection mechanisms in upcoming automobiles. Embedding Intrusion Detection Systems (IDS) into in-vehicle networks is an applicable approach to enhance overall vehicle security complementary to encryption and authentication mechanisms. Evaluating and testing IDS is a difficult task. Realistic datasets that are compliant to the automotive domain specific requirements are necessary for testing but hard to obtain. Furthermore, there is no standardized methodology for the evaluations which in turn leads to a lack of comparability of the results.

[1] Audi AG, 85045 Ingolstadt, christopher.corbett@audi.de
[2] TU Darmstadt, Department of Computer Science, fi59eged@rbg.informatik.tu-darmstadt.de
[3] Ulm University, Institute of Distributed Systems, {firstname}.{lastname}@uni-ulm.de

In this paper, we show that most commonly available tools do not meet the requirements for automotive NIDS evaluations and we introduce our architecture concept to cover those needs. With our approach, we are not only able to test network intrusion detection systems or generate custom network traffic, but — through separation of the evaluation scenario definition from specific network parameters — scenarios can be shared among interest groups without the necessity to anonymize traces or the risk of exposing real network topologies and information.

The remainder of this paper is structured as follows: In Section 2, we provide an overview of automotive domain specific protocols and topologies. We present related work in Section 3 and then give an overview of common in-vehicle network attack scenarios in Section 4. Necessary intrusion detection evaluation steps are described in section 5 and the derived requirements can be found in Section 6. Our framework architecture concept is described in Section 7; followed by our conclusion in Section 8.

2 Background

The automotive industry introduced a variety of bus technologies over the years. Local Interconnect Network (LIN), Media Oriented Systems Transport (MOST), Controller Area Network (CAN) and Flexray are well established in in-vehicle networks. With new feature sets, bandwidth requirements increased rapidly and therefore new technologies such as the enhanced CAN — Controller Area Network Flexible Data Rate (CAN-FD) — and the Ethernet (IEEE 802.3) protocol gained attention. As in-vehicle networks are very heterogeneous, data exchange between Electronic Control Units (ECUs) exceeds bus technology boundaries and translations (e.g. transporting CAN frames via Ethernet) are commonly used. To start off with a decent framework feature set to generate testing workload we examined attributes, parameters and characteristics of automotive Ethernet, CAN and CAN-FD.

2.1 Automotive Protocols

With each bus technology and feature set, new automotive protocols were introduced or enhanced over time, from which some are used in industrial networks (e.g. CAN protocols) and for others it is thinkable to be used in company networks (e.g. remote vehicle diagnostics). Figure 1 shows the classification of CAN and Ethernet protocols in the layers of the Open Systems Interconnection Model (OSI).

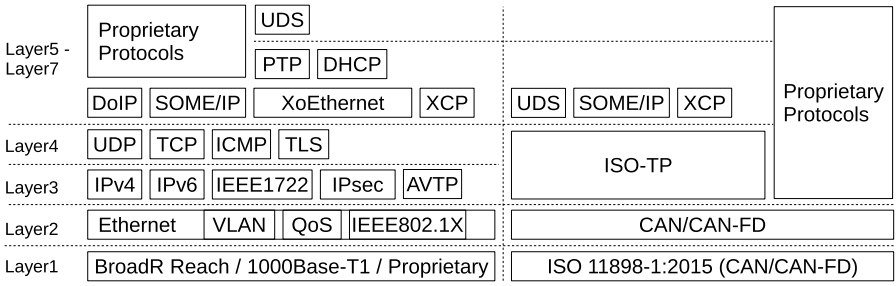


Figure 1: Protocol overview

Zimmermann and Schmidgall [ZS14] give an in depth overview of most of the standardized automotive protocols. In Table 1 we provide a brief summary of protocols we analyzed to derive requirements for the testing framework architecture.

Protocol	**Standard**	**Description**
Unified Diagnostic Services (UDS)	ISO 14229	An application client-server protocol to remotely call diagnose procedures in ECUs and transport information back to the requester.
ISO Transport Protocol (ISO-TP)	ISO 15765-2	Protocol to transport payloads larger than the maximum payload size on CAN/CAN-FD.
Diagnose over Internet Protocol (DoIP)	ISO 13400	An automotive transport layer protocol to transport UDS messages between the client and the server using port 13400 (UDP/TCP).
Universal Measurement and Calibration Protocol (XCP)	ASAM MCD-1 XCP V1.3.0 [AS15]	A bus-independent master-slave communication protocol to exchange information between ECUs and a calibration software on a external device (e.g. PC, Vehicle Tester, Laptop).
Scalable Service-Oriented Middleware over Internet Protocol (SOME/IP)	AUTOSAR [AU15][AU14]	A service oriented protocol that supports remote procedures calls, data serialization and a service discovery and publish/subscribe mechanism.
XoverEthernet	AUTOSAR [AU15][AU14] / Proprietary	Frames of one bus system are transported as payload via Ethernet (e.g. CAN over Ethernet).
Proprietary	Proprietary	Besides standardized and established protocols, car manufacturers make use of own or third party proprietary protocols.

Table 1: List of analyzed protocols

2.2 Communication patterns

Communication patterns in in-vehicle networks are based on design rules which rest upon applied bus technologies as well as application and protocol requirements. Therefore, each Original Equipment Manufacturer (OEM) network acts differently and a general descrip-

tion can not be given. However, several suppliers provide development tools to cover requirements across OEMs and provide a decent overview of CAN bus communication patterns in their documentation (e.g. Vector Informatik GmbH [Ve15]). These patterns are extended by common Ethernet behavior (e.g. fire and forget) as it is not an exclusive replacement for legacy bus systems in the foreseeable future. As a result we derived several factors from CAN and Ethernet communications that result in different patterns. These are:

- time triggering
- events
- fire and forget
- request and response
- state premises (stateful)
- no state premises (stateless)

3 Related Work

3.1 Automotive IDS

The challenges of designing tests of intrusion detection systems are widely understood. Milenkoski et al. [Mi15] provide a very extensive survey of common practices for tests of different kinds of intrusion detection systems. They discuss the three main components of IDS testing: workloads, metrics and measurement methodology. For each of the components, the authors provide a common terminology. For our work, we adopt this terminology and propose an architecture for workload generation in automotive networks.

There has been previous work that deals with designing intrusion detection systems for in-vehicle networks that employ CAN as main bus technology [Ha14][SKK16][KK16][CS16]. However, to the best of our knowledge, there is no work that deals with intrusion detection in modern in-vehicle architectures that also employ Ethernet as a backbone technology for the in-vehicle network. Nonetheless, Herold et al. [He16] have explored anomaly detection for the Scalable Service Oriented Middleware over Internet Protocol (SOME/IP) using complex event processing. To test their anomaly detection regarding performance, they implemented a SOME/IP packet generator, featuring four kinds of simple attacks: 1) malformed packets 2) protocol violations 3) system-specific violations and 4) timing issues. However, they only investigated anomaly detection for SOME/IP, which is an application layer protocol that is employed in upcoming Ethernet-based vehicle networks. The automotive protocol stack for Ethernet-based networks is much more diverse. In our work, we look at all the protocols and designed an architecture that is able to test an NIDS in modern Ethernet-based in-vehicle networks.

Moreover, there has been a lot of works that deal with the generation of workloads for testing IDS [Mi15]. Antonatos et al. [AAM04] have provided an extensible framework for the generation of realistic workloads in their work. Their generator is, however limited to

application layer traffic, and focuses on the generation of payloads for these protocols. We do aim for a similar approach for our architecture, but want to provide more flexibility regarding developing and describing different scenarios. Our architecture also supports the generation of traffic down to layer 2. Furthermore, we extend our architecture to fulfill automotive requirements (cf. Section 6).

3.2 State of the Art Tools

There is a variety of tools which can be used to perform specific attacks or scans, such as Nmap, Nessus and Metasploit. However, most of them are limited to very specific use cases such as port scanning in the case of Nmap. While Metasploit can be used to generate traffic, its main purpose is to generate pure malicious traffic with the help of its integrated exploit database.

Manual testing is a very time consuming task. It involves manually generating traffic with a collection of tools, capture the traffic using e.g. Wireshark, and then modifying and replaying the traffic. Furthermore, none of these tools have been adapted for the automotive domain. The manually generated traffic would have to be adjusted to represent realistic traffic in an in-vehicle network for effective testing of automotive NIDS. This adjustment process can also take a substantial amount of time as the traffic model has to be modified for every model and vehicle setup.

Packet generation tools are meant as a solution to the manual generation problem. They facilitate the automated generation of packets, which can be used to test intrusion detection systems. However, they do not provide the functionality required to reliably test automotive NIDS. Most tools do not fulfill our requirements as they do not support traffic generation, modification, and forwarding across multiple interfaces, which e.g. allows man-in-the-middle attacks on layer 2.

We have explored the feature sets of 10 existing packet generation tools and have found that none of them provide support for automotive protocols such as SOME/IP or UDS and DoIP. None of them provide the ability to prioritize traffic when capturing and modifying the response, or when capturing and sending a response to a captured packet. Additionally, a large chunk of the tools did not provide the flexibility to write scripts in order to automate certain tasks and re-use them for further tasks.

Due to the cyclic nature of a lot of messages sent in an in-vehicle network, we also require a packet generation tool to send packets at steady intervals with an insignificant amount of jitter. None of the tools provided a similar feature except Ostinato and packETH. Ostinato only allowed setting an interval of a packet per X seconds; packETH, however, offered millisecond and even nanosecond resolution for interval generation, but lacks other features, such as multiple interface support. Moreover, as the Ethernet layer is more important in in-vehicle networks compared to company networks, we need full flexibility when crafting and modifying Ethernet packets. Only some of the packet generation tools allowed receiving packets on layer 2. A summary of our findings can be found in Table 2.

	PCAP Replay	Multiple Interface Handling	Scripting	Layer 2 Support	IPv6 Support	Automotive Protocols	Generate Mixed Traffic	Priority Handling	Capture Traffic	Packet modification	Packet sending interval	Automation
Tomahawk	✓						✓					✓
Bit-Twist	✓			✓			✓		✓			✓
Hping2							✓					✓
Hping3			✓				✓		✓			✓
Nemesis				✓	✓		✓					✓
Ostinato	✓	(✓)	✓	✓	✓		✓		✓		(✓)	
packETH	✓			✓	✓		✓				✓	
Yersinia				✓			(✓)		✓			
netsniff-ng	✓			✓	✓		✓		✓			
pktgen	✓	(✓)	✓	✓	✓		✓		✓			✓

Table 2: An overview of available packet generation tools and their capabilities.

4 In-Vehicle Network Attack Scenarios

There are different types of attack scenarios for in-vehicle networks. Figure 2 shows how we set up an example network topology with a centralized component (e.g. a gateway or a routing unit) and four network participants. The following scenarios are feasible ways to inject malicious traffic or to modify existing network traffic in vehicular networks.

1. **Man in the middle**: In this scenario a malicious network participant (E) is positioned between the devices d and r to eavesdrop, manipulate or forge network traffic.
2. **Compromised device**: In this scenario, device (a) gets compromised with a piece of malicious software (F) to forge authentic communication or modify communication behavior.
3. **Attached device**: A new malicious network participant (G) is attached to the network and forges network traffic on an existing connection between network participants b and r.

4. **Device replacement**: An existing device (c) gets replaced by a malicious device (H).
5. **Compromised central network device**: Similar to scenario 2) a malicious piece of software is placed into a central network device (I) to modify network behavior or traffic.

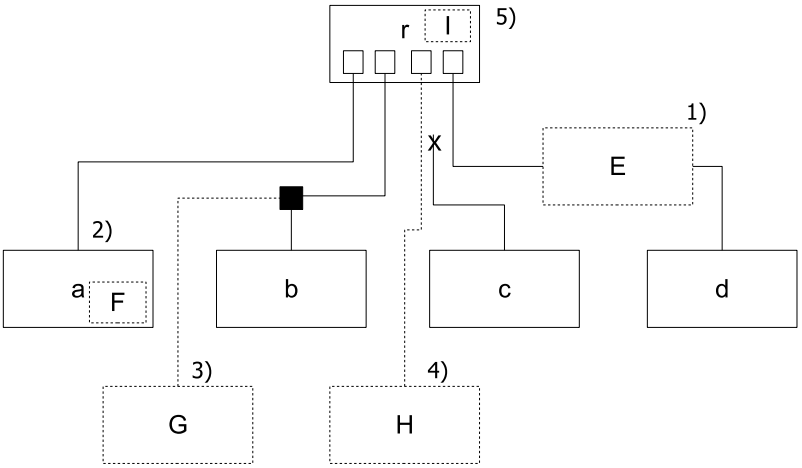


Figure 2: Overview of in-vehicle network attack scenarios.

5 IDS Testing Parameters and Metrics

This section gives a short introduction to different types of intrusion detection systems, general testing metrics and parameters as well as automotive domain specific parameters.

5.1 Categorization

Fallstrand et al. [FL15] state that intrusion detection and prevention systems are commonly categorized based on four properties:

- Scope — What kind of entity or entities does the system protect?
- Location and Distribution — Where and how are the system components deployed?
- Detection method — How does the system identify intrusions?
- Post-detection — How does the system respond to detected intrusions?

In this paper, we focus both on centralized and distributed in-vehicle network intrusion detection systems with a focus on misuse or anomaly based detection algorithms.

5.2 Metrics

IDS testing metrics can be categorized into performance- and security-related metrics which can again be grouped by commonly used basic attributes such as false-negative, true-positive, false-positive, true-negative, positive predictive value and negative predictive value, but also in composite values such as expected cost and intrusion detection capability [MC14]. In addition to the metrics used in prior research, we add detection latency to the list of metrics. Especially for automotive intrusion detection systems, timeliness of the attack detection can be crucial for the applicability in the field.

5.3 Requirements for NIDS Testing

The design of NIDS depends on numerous factors such as the network topology, the protocols, or the detection mechanisms used. These factors also need to be considered when designing a testing architecture. The testing architecture needs to be able to be agnostic to differences in NIDS architectures — i.e. it should not inherently favor one architecture over the other — while still acknowledging specific strengths and weaknesses of NIDS architecture types. For instance, some machine learning based NIDS need a certain time to build their specific neuronal network. A data set with a certain minimal size with labels for both benign traffic and attacks needs to be available both for training and testing the NIDS.

Milenkoski et al. [MC14] showed that workloads are mandatory for intrusion detection system testing and can be divided into three different types: purely benign, purely malicious and mixed workload sets. The acquirement or generation of workloads are either achieved by executables (e.g. manual generation, exploit databases, vulnerability and attack injection or workload drivers) or traces (e.g. through acquisition or generation). Usable training data is scarce and — as these data sets are recorded from real networks — they also show the characteristics of the original network without the possibility to adjust to the network configuration of the NIDS application site. Therefore, a dynamic testing system that can generate traffic on the fly and can generate an unlimited amount of data is advantageous.

Test runs have to be repeatable to ensure scientifically valid results, while the test environment also needs to offer the dynamic of real networks in the form of random changes in the network behavior. For this to work efficiently, automation must be possible.

5.4 Automotive Specific Challenges

An in-vehicle network combines different bus technologies which cannot be strictly separated and influence each other. Therefore, a comprehensive NIDS needs to consist of a combination of bus specific NIDS which adds complexity to the NIDS itself and to the tests of such a system. Currently, research focuses more on NIDS and Network Intrusion

Detection and Prevention Systems (NIDPS) for CAN networks while Ethernet is starting to gain some attention.

The unavailability of automotive specific attacks is another factor that complicates testing of an automotive NIDS. In comparison to attacks on company networks, attacks on vehicles are very vehicle and OEM specific. There is no comprehensive database of known attacks available that could be shared among car manufacturers.

To develop, test, and evaluate automotive NIDS — independent of the chosen detection method — valid and realistic data sets of network traffic must be available. Usually, traces of existing traffic or manually generate traffic are used. However, if no real traffic trace is available, the generated traffic cannot be proven to be realistic.

6 Framework Requirements

Considering attack scenarios, traffic generation, and IDS metrics, we derived a set of essential requirements. The architecture has to meet these requirements to facilitate the generation of realistic automotive workloads.

Protocol support The architecture must provide the ability to parse, manipulate and forge packets sent using protocols used in the automotive protocol stack described in Section 2. This facilitates the communication with other members in the automotive network as a legit as well as a malicious entity, depending on the scenario.

Frame manipulation A lot of network management, such as Virtual Local Area Network (VLAN) segmentation, happens on the Ethernet layer. Therefore, in addition to the previous requirement, the given tool must be able to manipulate packets on layer 2, including the VLAN tag.

Response time In order to deal with real-time applications in automotive networks, the tool has to be able to respond to a packet within the defined deadline for the corresponding vehicle domain. This ranges from 10ms for safety-critical sytems up to 150ms for audio/video streams in the infotainment domain [LP13].

Bandwidth The tool must provide a bandwidth of 100 Mbit/s (better yet 1 Gbit/s). Current automotive applications employ 100 MBit Ethernet, however, in future applications Gbit Ethernet is going to be employed in vehicles.

Time interval support ECUs are very sensitive regarding the interval at which they expect a certain signal or packet to arrive at their interface. Hence, the tool must be able to send and forward packets and frames at steady intervals while keeping the jitter as low as possible.

Fuzzing support Due to the long lifetime of automobiles, they are continuously exposed to new kinds of attacks. The tool must provide a fuzzing functionality to be able to simulate previously unknown scenarios and attacks.

State handling State handling is the ability to establish a certain state in a protocol. For example, messages A,B,C are sent according to specification and then deviate from the specification or modify messages. The state machine is also required to perform e.g. Transmission Control Protocol (TCP) Session Hijacking attacks.

Frame and packet scheduling The prioritization of packets and frames is of higher importance in automotive networks compared to company networks. The architecture therefore both has to be able to deal with incoming packets of different priority, and has to be capable to prioritize their processing accordingly.

Multiple interface support Several interfaces must be usable in parallel. Some devices communicate on several buses such as Ethernet and CAN. The architecture and tool must be able to replicate this behavior.

Scenario and parameter separation The separation of evaluation scenarios and data or value sets is important to enable the exchange and verification of results with and by third parties.

7 Testing Tool Architecture Concept

We propose an architecture for a testing tool that facilitates the proper evaluation of automotive IDS by satisfying all the requirements which we have defined in the previous section. Fulfilling the requirements ensures the generation of realistic automotive workloads. It also overcomes various shortcomings of existing tools. While our goal was to design a tool for the evaluation of automotive IDS, it can also be used to perform functional testing as well as security testing of an (automotive) network.

7.1 Architecture

The tool architecture is divided into three layers: user, developer, and system. This makes the tool's underlying framework easy to extend for those, who have the technical knowledge and easy to use for those, who just want to set up a test quickly using the pre-defined scenarios.

From a user's perspective, either pre-defined testing and attack scenarios or a self-designed scenario description can be used and configured. The configuration file contains various parameters exposed by the scenario, such as interfaces, protocols, layers, and packet values. Furthermore, it includes timing as well as priority information, if needed. Additional parameters can be exposed through the developer layer.

The tool's developer layer offers an extensible framework with which testing and attack scenarios can be developed. It provides three basic modules: **function blocks**, **core** and **network abstraction**. A developer can implement a function block (e.g. SYN scan attack) with custom logic and a defined parameter set. These function blocks can then be used by users to describe scenarios, which resemble malicious, benign, or mixed traffic.

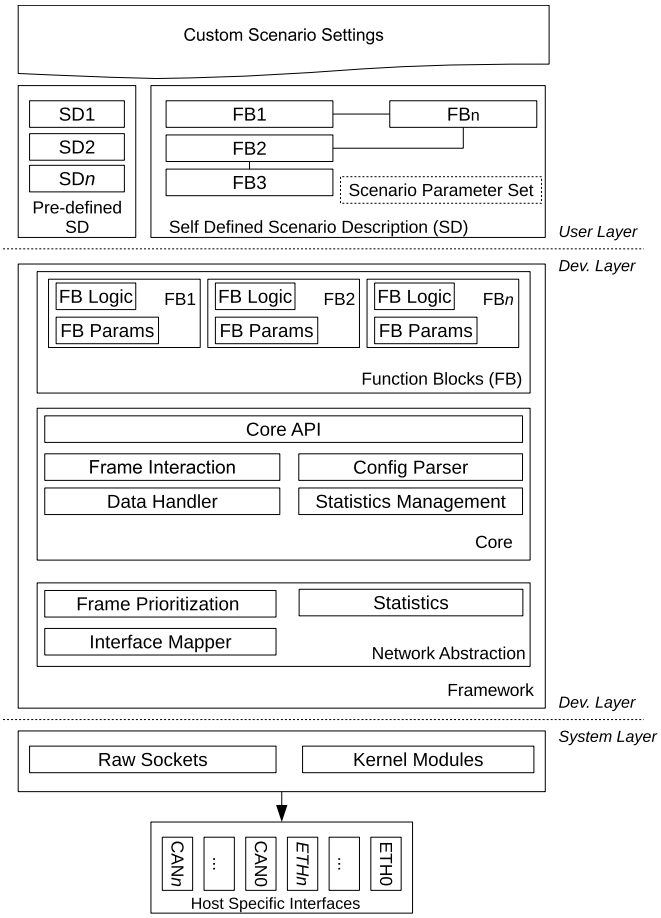


Figure 3: An architecture for an automotive testing tool.

Some simple example scenarios can be:

1. **Pure benign traffic:** We simulate an ECU that sends out a signal at a pre-defined interval.
2. **Pure malicious traffic:** We send out a TCP SYN scan to determine open ports on an ECU.
3. **Mixed traffic:** We combine function blocks 1) and 2) to disguise our attack in regular traffic, and provide a more realistic and challenging scenario.

The core of the framework provides essential functionality such as frame interaction, a configuration parser, a statistics manager, and a data handler. An Application Programming Interface (API) can be used by function blocks to interact with the core's submodules. The data handler submodule provides means to interact with a list of provided payloads for a

function block. Moreover, the frame interaction submodule enables parsing, crafting, and manipulation of frames and packets.

The framework's network abstraction module provides functionality to interface with different kinds of networking sockets, such as raw sockets or custom implementations (e.g. ring-buffer based implementations such as PF_RING[PF]) of the system layer. Furthermore, it takes care of the prioritization of frames to meet defined timing constraints. It also maps the physical interfaces of the executing host to logical interfaces defined in the scenario description.

Figure 3 shows the architecture of our proposed testing and workload generation framework for the evaluation of automotive NIDS.

7.2 Discussion

Existing tools shown in 3.2 cover only some features required to create workloads for an automotive NIDS testing. Either several tools must be combined or they lack necessary protocol support. Our architecture concept remedies these shortcomings and provides the ability to generate, modify, and analyze automotive-compliant traffic. Through scenario descriptions, it is possible to generate both benign and malicious traffic, and easily apply these scenarios to different vehicle setups. As our approach supports several interfaces, it is possible to implement more complex scenarios, such as man-in-the-middle attacks on layer 2. Additionally, setups can be shared among other research groups to verify results or to be used in their own research.

8 Conclusion

Evaluating and testing network intrusion detection systems is essential for improving NIDS. Workloads are necessary for testing the NIDS' crucial detection capabilities. For automotive Ethernet, such workloads are not currently available. For our malicious workload model, we consider five attack scenarios as presented in Section 4. We then derived several requirements that are necessary to be able to generate realistic traffic in Ethernet-based in-vehicle networks. In particular, the most important requirements are the ability to handle multiple interfaces and the ability to prioritize the handling of different streams of traffic.

We have analyzed several packet generation tools. We have found that none of the tools we analyzed fulfilled our requirements. The most prominent finding from our analysis showed that none of the tools provided support for multiple interfaces or traffic prioritization. With an extensive amount of features missing in all analyzed tools, we have come to the conclusion, that extending existing tools is not a viable option and that a new architecture has to be designed with the specific challenges of in-vehicular networks in mind.

We have proposed a novel architecture concept for a tool that remedies these shortcomings. Furthermore, we made sure that our proposed architecture is extensible. New scenarios

and attacks can be added easily through scenario descriptions. The provided functionality can be extended through additional function blocks, or by extending the framework. Our architecture fulfills the set requirements described in Section 6 by providing the necessary modules in the framework.

A proof of concept implementation of the framework has to be provided to determine whether our architecture proves usable in a realistic scenario. Said implementation then has to be evaluated with regard to our identified requirements by implementing the attack scenarios as described in Section 4. Furthermore, using said implementation to evaluate a given automotive NIDS requires implementation of further scenarios to build a realistic workload model. All these steps are left for future work.

References

[AAM04] Antonatos, Spyros; Anagnostakis, Kostas G; Markatos, Evangelos P: Generating realistic workloads for network intrusion detection systems. In: ACM SIGSOFT Software Engineering Notes. volume 29. ACM, pp. 207–215, 2004.

[AS15] ASAM MCD-1 XCP V1.3.0: Universial Measurement and Calibration Protocol (XCP), 2015.

[AU14] AUTOSAR 4.2 Rev. 1: Example for a Serialization Protocol (SOME/IP), 2014.

[AU15] AUTOSAR 4.2 Rev. 2: Specification of Service Discovery, 2015.

[CS16] Cho, Kyong-Tak; Shin, Kang G: Fingerprinting electronic control units for vehicle intrusion detection. In: 25th USENIX Security Symposium (USENIX Security 16). USENIX Association, pp. 911–927, 2016.

[FL15] Fallstrand, Daniel; Lindström, Viktor: Applicability analysis of intrusion detection and prevention in automotive systems. Master's thesis, Department of Computer Science and Engineering; Chalmers University of Technology;Göteborg Sweden, 2015.

[Ha14] Han, Song; Xie, Miao; Chen, Hsiao-Hwa; Ling, Yun: Intrusion Detection in Cyber-Physical Systems: Techniques and Challenges. In: IEEE SYSTEMS JOURNAL. volume 8. IEEE, 2014.

[He16] Herold, Nadine; Posselt, Stephan-A; Hanka, Oliver; Carle, Georg: Anomaly detection for SOME/IP using complex event processing. In: Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP. IEEE, pp. 1221–1226, 2016.

[KK16] Kang, Min-Joo; Kang, Je-Won: Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security. PloS one, 11(6):e0155781, 2016.

[LP13] Lee, Youngwoo; Park, KyoungSoo: Meeting the real-time constraints with standard Ethernet in an in-vehicle network. In: Intelligent Vehicles Symposium (IV), 2013 IEEE. IEEE, pp. 1313–1318, 2013.

[MC14] Mitchell, Robert; Chen, Ing-Ray: A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. volume 46. ACM Computing Surveys, 2014.

[Mi15] Milenkoski, Aleksandar; Vieira, Marco; Kounev, Samuel; Avritzer, Alberto; Payne, Bryan D: Evaluating Computer Intrusion Detection Systems: A Survey of Common Practices. ACM Computing Surveys (CSUR), 48(1):12, 2015.

[MV15] Miller, Charlie; Valasek, Chris: Remote Exploitation of an Unaltered Passenger Vehicle. Black Hat, 2015.

[PF] PF_RING: http://www.ntop.org/products/packet-capture/pf_ring/. Last accessed 2016-12-12.

[RM15] Rogers, Marc; Mahaffey, Kevin: How to Hack a Tesla Model S. DEF CON 23, 2015.

[SKK16] Song, Hyun Min; Kim, Ha Rang; Kim, Huy Kang: Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In: 2016 International Conference on Information Networking (ICOIN). IEEE, pp. 63–68, 2016.

[Ve15] Vector Informatik: Vector: CANoe Interaction Layer. 2015. http://vector.com/portal/medien/cmc/application_notes/SN-IND-1-011_InteractionLayer.pdf. Last accessed 2016-12-12.

[ZS14] Zimmermann, Werner; Schmidgall, Ralf: Bussysteme in der Fahrzeugtechnik. Springer Vieweg, 2014.

Peter Dencker et. al.(Hrsg.): Automotive Safatey & Security 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 103

Adapting Organic Computing Architectures to an Automotive Environment to Increase Safety &Security

Kevin Lamshöft, Robert Altschaffel, Jana Dittmann [1]

Abstract: Modern cars are very complex systems operating in a diverse environment. Today they incorporate an internal network connecting an array of actuators and sensors to ECUs (Electronic Control Units) which implement basic functions and advanced driver assistance systems. Opening these networks to outside communication channels (like Car-to-X-communication) new possibilities but also new attack vectors arise. Recent work has shown that it is possible for an attacker to infiltrate the ECU network insides a vehicle using these external communication channels. Any attack on the security of a vehicle comes implies an impact on the safety of road traffic. This paper discusses the possibilities of using architectures suggested by Organic Computing to reduce these arising security risks and therefore improve safety. A proposed architecture is implemented in a demonstrator and evaluated using different attack scenarios.

Keywords: Automotive, System Architectures, Organic Computing

1 Introduction

Modern cars are complex systems containing a wide array of actuators, sensors and ECUs (Electronic Control Units). Some of these components are essential for the basic function of a car while others provide assistance or amnesties to the driver. All these components are connected to an internal network. With the growing number of external means to connect to this network, like Car-to-X-Communication (C2X), In-Car Internet or remote diagnostics an inherent risk of attacks on these networks arise. Recent work [MV15] has proven this assumption. Any attack on the security of a vehicle carries the same implication on the safety of road traffic as error and faults of individual vehicular components. They can lead to dangerous situations either through direct means (e.g. failure of brakes), interruption of an assistance function the driver relies on (e.g. ABS) or distraction (e.g. Multimedia). The complex interplay of all these components is bound to further increase with the introduction of autonomous vehicles. This complexity challenges classical engineering approaches. Hence this paper explores the possibilities of using

[1] Otto-von-Guericke Universität, AMSL, Universitätsplatz 2, 39102 Magdeburg,
Kevin.Lamshoeft|Robert.Altschaffel|Jana.Dittmann@iti.cs.uni-magdeburg.de

architectures derived from the field of Organic Computing in order to cope with the growing complexity.
During this paper section 2 handles a brief introduction on the specifics of automotive IT and gives an overview on the topic of Organic Computing in general and the observer/controller architecture in particular. Section 3 will discuss how the naive implementation of Organic Computing architectures would perform in an automotive environment while section 4 presents and discusses changes to such an architecture proposed by us. Section 5 describes the implementation of a demonstrator used for evaluation of the concepts proposed in this work. Section 6 discusses practical scenarios in which an attacker injects spoofed messages in a malicious manner. Here it is evaluated if and how the demonstrator could reduce the impact of an attack. Section 7 concludes this paper with summary and outlook.

2 State of the Art

This section gives a brief overview on the state of automotive IT focused on the main vehicular network - the CAN bus. Further introduction is given on the principles of Organic Computing (OC) and the means to achieve them by using various architectures. Finally, information on anomaly-based intrusion detection will be presented since it will be used in the demonstrator used for evaluating the concepts presented in this work.

2.1 Specifics of automotive IT

Modern cars consist of dumb and smart components. The dumb (or passive) components are all parts without electronics. Smart (or active) components consist of:

- **Sensors** measure the conditions of the vehicle's systems and environment (e.g. pressure, speed, rain intensity etc.) but can also capture user input requests.
- **Actuators** are units that perform a mechanic actuation.
- **Electronic Control Units (ECUs)** perform the electronic processing of input signals, which are acquired via different types of sensors and relay commands to the actuators. Some units control critical systems of the vehicle such as the engine and safety systems (ABS, Airbag) while others control comfort units such as the door

control units. The number of ECUs embedded with a vehicle is still rising to more than 100 in 2010 [Su14].

- **Direct analogue cable connections** are used to carry measured signals (from sensors to ECU) or actuation impulses (from ECU to actuators).
- **Shared Digital Bus Systems** are used for communication among ECUs. Beyond the direct connections between ECUs and sensors/actuators, ECUs are additionally connected amongst each other via digital field bus systems [Tr09]. This shared medium is used to exchange required information, like forwarding digitized sensor signals, exchanging current operating parameters, remote actuation requests or diagnostic requests for maintenance purposes. In modern cars, several different technologies for digital automotive field bus systems are used. The most common automotive field bus system is the Controller Area Network (CAN) [Bos91], which is the core network of the vehicle systems communication. This CAN network is divided into sub-networks such as powertrain/engine, diagnostics, comfort or infotainment. ECUs are connected to each sub-network depending on their functions and these sub-networks interconnect in a ECU device called CAN Gateway which handles the routing of messages to different sub-networks. The specific sub-networks offer a shared medium for the exchange of CAN messages. The CAN message consists of several flags without further importance to this paper, the CAN ID and the payload. The CAN ID represents the type of a message and implies a certain sender and receiver for the message - hence any ECU on the specific bus will receive all messages but discard the ones with CAN ID unimportant to it. It is assumed that a message with the corresponding ID is send by the ECU normally responsible for this message. In addition, the CAN ID serves as priority. Since there is no sender verification it is very easy to insert crafted packets into CAN networks once access to an entity able to send on the bus (an ECU or a tap) is established.

Recent trends show the increased communication between cars and other cars (Car-to-Car, C2C) or infrastructure (Car-to-Infrastructure, C2I [Ka08]). These communication channels not only increase functionality but also carry the risk of new attack vectors, as demonstrated [MV15].
In an automotive environment, an attack on the security of the car is bound to also become a safety risk due to the already dangerous nature of road traffic. If in classical desktop IT a system crashes the system simple crashed. If the IT system in a moving vehicle crashes you have a moving vehicle that does not act reliable to user input and might as well crash in a more dramatic manner.

2.2 General Introduction to Organic Computing

Organic Computing is an approach to deal with the complexity regarding systems of systems. With the ever-growing amount of different participating systems, scenarios and circumstances classical engineering approaches are reaching their limits. OC aims at viewing the system of systems as an organic whole with the user only formulating aims or tasks while the subsystems themselves deal with the realization of these tasks, as evident in the quote from [Wu08]:
In organic computing, the only task humans hold on to is the setting of goals. As the machine is autonomously organizing, detailed communication between programmer and machine is restricted to the fundamental algorithm, which is realizing system organization.

A fundamental aspect of OC is emergence. [WH05] define emergence as:
A system exhibits emergence when there are coherent emergents at the macro-level that dynamically arise from the interactions between the parts at the micro-level. Such emergents are novel w.r.t. the individual parts of the system.

Hence emergence describes a macro behaviour of a complex systems not inherent in the behaviour of the specific components. A system based on OC principles is henceforth called an organic system and should fulfil several properties. These properties are known as self-x-properties. These include self-adaption as the core property of any organic system [MSU11]. Other examples for self-x-properties in their work are include self-configuration, self-optimization and self-healing, which are specialized types of self-adaption. Furthermore, self-perception was identified as a basic requirement for organic systems [Al14].

2.3 Generic Observer/Controller Architecture in Organic Computing

In order to guarantee that the macro behaviour of a decentralized self-organizing system meets the intended purpose a so called observer/controller is used. A suggestion for this central concept is the Generic Observer/Controller Architecture as introduced in [Ri06] and is

used the foundation for the approach presented in this work. The architecture consists of three major entities: A multi-agent system, called System under Observation and Control (SuOC), an observer and a controller. The Observer is monitoring the SuOC with sensors, processes the data and passes accumulated information on the state to the controller which then evaluates possible actions and might control the SuOC. The observer/controller pattern is built on top of the SuOC - if the observer/controller fails, the SuOC will retain its self-organizing structure.

The observer consists of different modules, which define the observation process:

- **O1 Monitor**: gains raw data from the underlying SuOC
- **O2 Log file**: saves data from each iteration which might be used for predictions
- **O3 Pre-Processor**: prepares data for analyses and prediction
- **O4 Data analyser**: applies a set of detectors on the pre-processed data; result is reflecting the current state of the SuOC
- **O5 Predictor**: predicts future system states based on raw data, history data and analyser data
- **O6 Aggregator**: accumulates data which is then passed to the controller

The controller receives aggregated data from the observer and compares it to the goals for the organic system by the external user. This component directs emergent behaviour of the SuOC in order to achieve desired emergent behaviour or disrupt or prevent undesired emergent behaviour. Three types of control can be applied by the controller: Influencing local decision rules, influencing the system structure and influencing the environment. The controller uses an internal action selector which selects best suited action based on the current situation of the SuOC (mapping) and forwards this decision towards its actuators. The applied action is saved in a history file and is in a next step evaluated by comparing the new system state with the state before. Depending on how much the action influenced the system state of the SuOC the fitness value for the mapping is updated. Hence the mapping is improved over time. This learning process can be enhance using machine learning techniques, for example, by using evolutionary algorithms, learning classifier systems, reinforcement learning or neural networks.

2.4 Anomaly-based Intrusion Detection

The use of use Shannon entropy for detection of anomalies in in-vehicle networks was proposed in 2011 [MN11]. This work uses entropy which is the expected average value of information that a message carries in a message flow. Entropy can be calculated for a single message, specific CAN IDs or the whole bus traffic. The entropy of the usual behaviour is determined a priori in a learning phase. For intrusion detection, the entropy is calculated in fixed intervals and compared to the entropy of normal behaviour. An anomaly is found if the difference if the expected and the current value differ more than a threshold. This approach has been successfully tested and proved useful [MSGC16]. An Extension [CK16] proposes the use of fingerprinting techniques known from conventional IT in automotive networks. Here the clock skew estimation is used to localise affected ECUs.

3 Adopting the Generic Controller/Observer Architecture to Automotive Bus Networks

As mentioned in section 2.1 defending against advanced attacks on automotive bus networks is a non-trivial task.
However, most attacks have one aspect in common: On the lowest level, they are sending forged messages on automotive bus networks. Such a message does look legitimate in CAN bus networks since there is no authentication. While fuzzing and replay attacks reportedly can be detected in parts by automotive IDS [MSGC16], more sophisticated, targeted attacks can be recognized only by looking at the result in the overall system – in this case the car.

For example, prior experiments with several cars have shown that an attacker is able to lock the doors permanently by sending forged messages on the CAN bus. This prevents the passengers from leaving the car. In addition, the heating can be turned on and air conditioning can be turned off while preventing user input. This can lead to serious disturbance or even bodily harm. For this attack, an attacker only needs a few forged messages. Each message by itself is inconspicuous and seems legitimate and therefore

should not raise any alarm. However, the behaviour of the overall system, caused by these accumulated messages, is suspicious and harmful.

As shown before a car is a system of system and hence this undesired behaviour akin to an undesired emergent behaviour. As shown in section 2.3 the Generic Observer/Controller Architecture forms the part of the Organic Computing approach aiming to observe emergent effects and either taking actions to achieve desired emergence or preventing/disrupting undesired emergence. By adapting this architecture to the requirements of cars and implementing the concept we might be able to detect such advanced attacks and mitigate their impacts with low-cost hardware. In contrast to Intrusion Detection Systems the presented approach goes further and does not only detect anomalies but also takes actions to counter attacks. This is achieved by not only looking for anomalies but aggregating data from multiple sources in order to get a better understanding of the systems state and reducing the number of false-positives. By using methods derived from the field of machine learning the system learns with each incident and gets better over time.

The following section will deal with adapting this generic approach to the automotive domain.

3.1 Theoretical Considerations

The first step on adapting the Generic Observer/Controller Architecture to the automotive domain is a the definition of system boundaries for the SuOC. Since this approach aims to increase robustness against attacks, or in fact general malfunctions as well, we want to achieve desired and disrupt undesired emergent behaviour of the whole system – technically speaking the whole car. Considerations on the feasibility of defining a car as SuOC rely on the definition of a SuOC presented in section 2.3. Here a SuOC is defined as a multi-agent, self-organizing system.

Automotive IT consists of a network of ECUs, sensors and actuators. As these ECUs are autonomous entities, communicating and interacting with each other, observing and acting in an environment to achieve goals they can be considered agents. Technically speaking a set of ECU networks can

therefore considered a multi-agent system. Following the definition given by [MWJ+07] a self-organizing system is self-managing, structure-adaptive and employs decentralized control. The network of ECUs is self-managing in the sense of adapting to different I/O requirements without an explicit external control input on how to achieve this. The driver gives a general objective (I/O requirement) towards the car (e.g. Acceleration) causing multiple ECUs to work together in order to achieve that goal. Automotive IT is structure-adaptive as the ECUs maintain their structure and provide the systems primary functionality. As there is no central ECU that controls the others it employs decentralised control – leading to the conclusion that the network of ECUs can be seen as a self-organizing system.

3.2 Adapting the Observer to Automotive Bus Networks

We aim at not only detecting irregularities in the function but the car but also on reaction towards these irregularities. Therefore, we propose the usage of multiple cooperating Observer/Controller instances (as defined in the Generic Observer/Controller Architecture). One major task of the observer is, similar to IDSs known from conventional IT, detecting anomalies - called "symptoms" in the MAPE cycle [IBM06]. These symptoms might lead to an undesired emergent behaviour. Going beyond the possibilities of an IDS in this approach the Observer considers the symptoms more thoroughly by aggregating data from several data analysers (resp. emergence detectors or IDSs), Log files, Predictors and communicating with other Observers. This helps to reduce false-positives and get a better understanding of the symptom before applying control actions.

In order to achieve this the Observer presented in section 2.3 is adapted to an automotive environment as follows:

- **O1 Monitor**: In the OC architecture the observer is monitoring the influence of the agents on their surroundings. In the case of automotive IT this means that the Observer is not monitoring the ECUs directly but their influence on the car. Hence this means the monitoring of specific actuators. As it would require a broad range of sensors to monitor the actual behaviour of all actuators we propose a different approach in not monitoring the physical actions of the actuators but in monitoring the communications on the automotive bus networks. This is feasible since actuators

at this moment do not have any computing power themselves and just act on the orders given by them from the ECUs. Hence, if there is an action, there is also a message causing this action. This might not hold true for the future, though, so future work will add sensors which directly monitor the physical actions of the actuators. Even then the primary sensor will most likely still be the networking interface which allows the observer to read all BUS communication.

- **O2 Log file**: Since the log files function is basically to save data for further iterations and predictions no adaption to an automotive context is needed.
- **O3 Pre-Processor**: This step prepares the raw data supplied by the monitor for the following steps of analysing and prediction. Depending on the Data Analysers and Predictors the specific tasks of the Pre-Processor might vary. For bus networks the main part of the Pre-Processor is filtering (e.g. leave out keep-alive-messages or duplicates) and prioritising (e.g. error messages) of the network traffic. Aggregation and counting of reoccurring messages might be useful as well.
- **O4 Data Analyser**: This is one of most important parts in the approach presented in this work. It applies a set of detectors on the pre-processed data in order to identify undesired behaviour of the car. One trivial approach would be to identify error frames (e.g. failure of signal lights) on the bus network. This relies on the affected ECU detecting such failures. It hence would not work in scenarios where no malfunction of specific components are caused but rather a harmful macro behaviour, like in the scenario introduced in section 3. Therefore, a deeper analysis of the network traffic is needed in order to detect undesired emergent behaviour. Two different approaches are suggested here: detect unusual behaviour and detect illogical behaviour and detect rule violations. An example for unusual behaviour might be the toggling of certain features multiple times in short intervals (e.g. recurring signals to close windows). This might also point to a component failing to react on a given input, like repeated pushing of a brake pedal without reaction of the brake. In these cases further analysis will be conducted. As the log file stores information about the system state the Analyser is able to check if the speed reduced in case of the repeatedly triggered brake pedal. Examples for illogical behaviour are opening the trunk while driving, pushing brake and accelerator pedals at same time causing invalid system states. The third option is to define correct behaviour a priori by rules (e.g. doors need to be closed while driving) and monitor violations.
- **O5 Predictor**: Based on data from the analyser and log file the predictor calculates possible future system states which will be passed to the controller. This enables the controller to take preventative actions. For example, the predictor extrapolates the speed for a time t+1, based on the current speed at time t and the speed of the state before at time t-1.
- **O6 Aggregator**: The aggregator accumulates information of the analysers detectors and the predictor to give a most accurate evaluation on the current and future system states to the controller which then based on that information takes actions to influence the environment towards a desired behaviour. In order to identify and

analyse the symptom the Aggregator needs to interpret the data coming from the data analysers. Hence a semantic lookup table (which message is representing what information) is needed but could also be implemented at an earlier stage in the Data Analyser. The data coming from the Observer needs to be encoded in a way that the Controller can map it to an action. One naive approach is to pass raw information coming from the analysers without any semantics towards the Controller. For example, the corresponding CAN IDs which show anomalies could be mapped to an action (e.g. a detected anomaly at 0x172 would be mapped to an action which opens the windows). Depending on the Data Analysers and information gathered by the Aggregator more precise information could be passed. An exemplary data set for the given scenario could be *state = [{Entropy Anomaly Detector -> Affected ID: 0x172}, {ECU Fingerprint Anomaly Detector -> Anomaly Source: central lock ECU}, {Aggregator -> Result: doors do not open]* and would map *actions = [{flash central lock ECU}, {open windows}, {open trunk}]*. We recommend using multiple Observers communicating with each other, getting additional information and do cross checking, to specify the systems state more accurately. For example, an Observer monitoring unusual behaviour of the doors (e.g. a door is opening and closing repeatedly) can request additional information from another Observer which is monitoring different parts of the vehicle like the powertrain bus in order to determine if the vehicle is moving.

3.3 Adapting the Controller to Automotive Bus Networks

The Controller's main task is to take actions based on information it receives from the Observer. The Controller takes actions by sending messages to the agents (ECUs) which then applies actions to the car. Therefore, a mapping of the environmental states and actions is required by the Controller.

When the Controller gets information by the Observer it applies a set of classifiers (rules) to map environmental states into actions. Each classifier has a fitness value/reward that defines the quality of the action and is updated when the Controller gets feedback on how good the applied action has performed.
If multiple classifiers fit a given environmental state the one with highest fitness/reward is selected. The structure of a classifier is given by*{State -> Action : Fitness}*. One basic example for a classifier would be {*0x172 -> 0x172#1122 : 42*}, where *0x172* marks the CAN ID, which shows anomalies, *0x172#1122*the action (the CAN message to be sent by the controller) and *42* the fitness value. Before applying actions, we suggest

reporting to the driver. The user should be informed that an anomaly has been detected, what causes are probable and which influences are detected and expected. The driver should have the option to mark the anomaly as false-positive. Causes for false-positives are found in the data analysers as well as in unexpected behaviour of the driver or passengers. In a next step the driver should be informed on planned actions and asked for permission. There might be situations in which the driver has to act by himself (e.g. applying manual handbrake to slow the car in case of brake failure). In that case, the controller only gives a warning and recommendations on how to act. The rules for reaction can be added manually or generated by machine learning.

4 Implementation

The approach presented in this paper has been evaluated in a demonstration in order to examine its merits and drawbacks. For our experimental work we used the electronics of an Audi Q7 built in 2008. These electronics have been extracted after a crash test. Due to the crash, most parts of its drive train were missing. Hence, we focused on ECUs communicating on the comfort bus. The CAN bus was accessed directly by a Raspberry Pi using CANtact interfaces. CANtact was chosen for using the SocketCAN driver of can-utils which allows to receive all frames from the bus and send arbitrary messages. A Raspberry Pi was used for implementing the Observer/Controller functionality.
The following subsections give an overview on how the concepts developed in section 3 have been implemented for the demonstrator.

4.1 Observer Implementation

This subsection describes how the observer was implemented in the demonstrator.

- **O1 Monitor:** The Monitor was implemented on a Raspberry Pi using Python. It uses a SocketCAN interface to communicate with the CANtact board and the python-can package to monitor traffic on the bus.
- **O2 Log File:** The demonstrator saves general information on the vehicle state, e.g. the status of ignition and door locks.

- **O3 Pre-Processor:** Received CAN frames are striped to CAN ID and payload only, removing timestamps since these would cause bogus results during entropy calculation.
- **O4 Data Analysers:** An entropy-based anomaly detection approach was implemented. (see section 2.4). The demonstrator implements a combination of entropy calculation for the whole bus and dedicated calculations for specific IDs. The entropy analyser has to go through a learning phase before it can be used. In that learning phase the average entropy μ, standard deviation σ, a model parameter k and time windows t are calculated and defined. A target space which marks the usual behaviour as a range$[\mu - \sigma, \mu + \sigma]$and a acceptance space which allows minor deviation of the usual behaviour (reduces false-positives) as range $[\mu - k\sigma, \mu + k\sigma]$, where k is a model parameter are defined. Figure 1 shows the learned parameter settings used in the experiments.

Entropy Calculation	Controller Area Network ID	average entropy μ	standard deviation σ	model parameter k	time / message window t
Comfort Bus	-	5.94	0.1	4.7	5 (seconds)
Locks	0x171	1.385	0.252	1.9	30 (frames)
Windows	0x182	0.11	0.33	0.6	30 (frames)
Multimedia	0x5C4	0.941	0.02	1.95	30 (frames)

Fig. 1: Learned parameter settings for the demonstrator

- **O5 Predictor:** Prediction is not included in this demonstrator.
- **O6 Aggregator:** The Aggregator fetches data from the entropy analysers and general information of the log file and passes them towards the Controller module. In this demonstrator the Observer reports where an anomaly is detected (whole bus and/or CAN IDs), the value of constraint violation (distance of measured entropy from acceptance space) and a human readable state of the system (e.g. *car standing, doors locked, anomaly regarding locks detected*) in order to inform the driver and for debugging purposes.

4.2 Controller Implementation

For the implementation of the controller a pre-defined classifier set is used. Using fitness values and wildcards for the classifiers the controller is able to map any state given as input from the Observer. Before taking any action the controller reports the aggregated information of the observer and the mapped action to the user (in the current version via CLI) and asks for permission. After taking the action the user is asked whether the problem

persists (in future versions the controller evaluates that by itself using data from the observer). If the problem is solved the classifier gets a reward which increases its fitness. If the anomaly persists the controller takes another matching action if available. If that is not the case the controller notifies the user to stop the car.

5 Evaluation

This section describes how the demonstrator was evaluated.

5.1 Adversary Model and Attack Scenarios

Prior research has shown several physical and remote attack surfaces and vectors [MV15]. In this evaluation scenario we selected an attacker who is able to send arbitrary messages with spoofed IDs. This allows the attacker to toggle certain features of the car in with the aim of making driving impossible or at least very uncomforting. This type of attack is typical for ransomware campaigns from other computational domains. We define three different attack scenarios for our experiments:

- **S1 Lockout** The attacker locks the driver out of his car. This attack can be implemented for the Q7 with a minimum effort. When the car gets locked or unlocked by the remote a corresponding sequence of messages is transmitted on the comfort bus. The attacker monitors the bus for the first message of this sequence. Upon detection the attacker immediately sends the sequence to lock the doors. That procedure is sufficient to lock the doors permanently. As the lock is purely electric a manual opening with the key can be overridden as described before.
- **S2 Nuisance** The attacker randomly toggles warning and turning lights, continuously raises the volume of the multimedia system and opens and closes the windows in a random way. This can again be done by inserting CAN messages to the comfort bus.
- **S3 Confinement** The attacker waits on the driver to turn off the car and release the key. Then the central locks are applied and windows are closed followed by a Denial-of-Service attack. This results in locked in passengers as doors and windows are closed an no user input is registered by the ECU.

These three scenarios are implemented in Python using the python-can package for receiving and transmitting messages. A second Raspberry Pi is used for this. In each Scenario the BUS traffic is monitored (O1) and pre-processed (O2).

5.2 Test Results

- **S1 Lockout** When the user tries to open to the car the doors are locked immediately again. That incident alone does not raise any alarm. Subsequent tries to open the car led to an alarm triggered by the detector of the corresponding ID and the complete bus (O4). The Aggregator (O6) fetches data related to the doors from the log file (O2) and passes information to the controller that anomalies were detected on the comfort. It also reports that the doors are probably closed. Since door locks are a concern for security the controller reports to the user over a CLI that unusual behaviour is observed related to the car locks and asks the user if he is trying to open the car. Depending on user input the observer opens the doors.
- **S2 Nuisance** The ransomware used in this scenario has several phases. The ransomware toggles the warning lights **(S21**). This is not sufficient to raise alarm. In the next step the ransomware quickly raises and lowers the volume leading to distracting noise (**S22**). The bus detector (O4) raises alarm due to low entropy values but the ID detector does not. The detected anomaly and information of the log file (O2) are not sufficient for the controller to take any actions. Several runs with different parameters for the ID detector have been performed. Large k values used in the ID detector imply low false-positives rates but does not lead to the detection the attack. Low k values lead to detection but brings high false-positive rates. A productive use is not possible at this moment. However, given the attack is detected the controller turns off the audio system as countermeasure. In the next phase the ransomware raises and lowers the windows in a random way **(S23**). This behaviour carries the sample implications like S22. In total, the current implementation cannot detect the attack reliable without producing high rates of false-positives. In future implementations, multiple analysers, for example ECU Fingerprinting or Artificial Neural Networks, should be used. Moreover, the aggregator could be improved by using a predictor (O5) and more detailed logs (O2), e.g. by building a model of the car that reflects its status.
- **S3 Confinement** The Denial-of-Service attack is detected by ID detectors of windows and locks as well as the bus detector (O4).

Moreover, the Observer reports to the controller that doors and windows are closed. The controller opens the trunk and notifies the user to leave the car through the trunk.

6 Summary and Outlook

This work discusses the possibilities of bringing Organic Computing Observer / Controller architectures into an automotive environment. It is shown that such an adaption is possible and even enhances security and safety of an automotive. The main contribution is the discussion and exploration of necessary adaptation as well as limitation and merits. A first demonstrator and its efficiency in hampering such complex exemplary attack scenarios has been shown. The approach presented in this work allows a reaction on complex threats which single constituent parts would not itself register as a threat. This approach seems well suited for use in other scenarios with similar threat scenarios and basic architecture. In essence all systems which consistent of actuators, sensors, communication bus and computational units carry the same risk of being attacked by injected bus messages. If the system is complex enough that a single injected message does not trigger any alarm or harm by itself while a sequence of injected messages provokes a harmful macro behaviour the same threat scenarios arise. The fact that these system of systems show an observable macro behaviour implies that they influence their surrounding and hence cause inherent safety risk once their security is compromised. One example for such a system would be industrial automation. Industrial automation shares the same basic components like automotive IT and faces the same problems with ever growing complexity. It is without doubt that industrial automation systems are systems of systems. In addition, attacks on these cyber-physical systems also often consist of a sequence of bus messages to the actuators which are each by itself without harm but in their entirety cause a harmful macro behaviour. Examples of malicious attacks on these types of systems might be similar to these presented in here. In the lockout scenario (S1) an attacker would manipulate an industrial system in a way to prohibit its normal function. This might include erratic moving of the industrial robots so a maintenance crew might only be able to approach the robot after powering it down. In either case the functionality is denied to the user and a safety risk arises from the security risk. The possibility of

causing nuisance (S2) by using industrial actuators and HMI is obvious, while a locked in scenario (S3) would require doors to be attached to the industrial system. This mighty only be relevant in risk zones separated by doors and more a topic of smart home automation. However, an adaptive system observing the macro behaviour of such a system again enables the suppression of this threat.

Acknowledgments

The work on organic computing in automotive environments is supported by German Research Foundation project ORCHideas (DFG GZ: 863/4-1). The application to industrial control systems is funded in parts by the German Federal Ministry of Economic Affairs and Energy (BMWi, project no. 1501502B). The authors thank all project staff members and involved students as well as the reviewers for their help.

References & Acknowledgments

[MV15] Miller, C.; Valesek C.: Remote Exploitation of an Unaltered Passenger Vehicle. Black Hat USA

[Su14] Sugimura, T: Junction Blocks Simplify and Decrease Networks When Matched to ECU and Wire Harness. Encyclopedia of Automotive Engineering. 1–7

[Tr98] Trautmann, T.: Grundlagen der Fahrzeugmechatronik: EinepraxisorientierteEinführungfürIngenieure, Physiker und Informatiker, 2009

[Ka08] Kargl,F.; Papadimitratos, P.; Buttyan, L.; Müter, M.; Schoch, E.; Wiedersheim, B.; Thong, T.; Calandriello, G.; di Torino, P.; Held, A.; Kung, A.; Hubaux, J.: Secure Vehicular Communication Systems: Implementation, Performance, and Research Challenges. Communications Magazine, Volume 46(11), 110-118

[Wu08] R.P. Würtz (ed.): Organic Computing. Understanding Complex Systems, doi: 10.1007/978-3-540-77657-4 1, © Springer-Verlag Berlin Heidelberg 2008

[WH05] De Wolf T., Holvoet T. (2005) Emergence Versus Self-Organisation: Different Concepts but Promising When Combined. In: Brueckner S.A., Di MarzoSerugendo G., Karageorgos A., Nagpal R. (eds) Engineering Self-Organising Systems. ESOA 2004. Lecture Notes in Computer Science, vol 3464. Springer, Berlin, Heidelberg

[MSU11] Müller-Schloer, C.; Schmeck, H.; Ungerer, T.: Organic Computing — A Paradigm Shift for Complex Systems, Springer, ISBN: 978-3-0348-0129-4, 2011.

[Al14] Altschaffel, R; Hoppe, T.; Kuhlmann, S.; Dittmann, J.: Towards more Secure Metrics for Assessing the Fitness Level of Smart Cars. Proceedings of the 3rd International Conference on Connected Vehicles & Expo, 149-154

[Ri06] Richter, U.; Mnif, M.; Branke, J.;Christian Muller-Schloer, C.;Schmeck, H.:Towards a generic observer/controllerarchitecture for Organic Computing. GI Jahrestagung (1) 93 (2006): 112-119.

[MWJ+07] Mühl, G., Werner, M., Jaeger, M., Herrmann, K. &Parzyjegla, H. (2007). On the definitions of self-managing and self-organizing systems, KiVS 2007 Workshop: Selbstorganisierende, Adaptive, KontextsensitiveverteilteSysteme (SAKS 2007).

[Bos91] Robert Bosch GmbH, CAN Specification 2.0, 1991, http://www.bosch-semiconductors.de/media/ubk_semiconductors/pdf_1/canliteratur/can2spec.pdf, (last checked: 18.10.2016)

[MN11] Müter, Michael, and NaimAsaj. "Entropy-based anomaly detection for in-vehicle networks." Intelligent Vehicles Symposium (IV), 2011 IEEE. IEEE, 2011.

[CK16] Cho, Kyong-Tak, and Kang G. Shin. "Fingerprinting electronic control units for vehicle intrusion detection." 25th USENIX Security Symposium (USENIX Security 16). USENIX Association, 2016.

[MSGC16] Marchetti, M., Stabili, D., Guido, A., &Colajanni, M. (2016, September). Evaluation of anomaly detection for in-vehicle networks through information-theoretic algorithms. In Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on (pp. 1-6). IEEE.

[IBM06] COMPUTING, Autonomic, et al. An architectural blueprint for autonomic computing. IBM White Paper, 2006, 31. Jg.

Peter Dencker et. al. (Hrsg.): Automotive Safety & Security 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 121

Ontologiebasierte Abhängigkeitsanalyse im Projektlastenheft

Konstantin Zichler[1] und Steffen Helke[2]

Abstract: Zu Beginn eines Projekts dokumentieren interdisziplinäre Domänen-Experten die Anforderungen an alle Lebensphasen eines Nutzfahrzeugs und die entsprechenden Realisierungskonzepte im Projektlastenheft. Die Kenntnis der Abhängigkeiten zwischen Anforderungen bietet den Vorteil, fehlerhafte Produktkonzepte bereits in der frühen Projektphase zu vermeiden. Bei der Durchführung von Abhängigkeitsanalysen besteht für die Experten der einzelnen Abteilungen die Schwierigkeit darin, von den dokumentierten Einzelbeiträgen auf domänenübergreifende Abhängigkeiten zwischen den Anforderungen zu schließen. Bisher werden diese Analysen für gewöhnlich manuell durchgeführt, da es dafür kaum Werkzeugunterstützung gibt. Wir stellen ein neuartiges Verfahren vor, bei dem das für die Abhängigkeitsanalyse erforderliche, fachspezifische Wissen zu einer gemeinsamen Wissensbasis in Form einer Ontologie aggregiert wird. Zusammen mit Axiomen, einem Reasoner und Werkzeugen aus dem Natural Language Processing wird eine automatisierte Abhängigkeitsanalyse im Projektlastenheft realisiert, mit der es möglich ist, bisher nicht berücksichtigte Abhängigkeiten zwischen Anforderungen zu identifizieren.

Keywords: Anforderungsmanagement, Abhängigkeitsanalyse, Ontologie, Reasoner, Natural Language Processing.

1 Einleitung

Die Entwicklung von Nutzfahrzeugen startet gewöhnlich nach einem Frontloading-Prinzip. Ziel bei diesem Ansatz ist es, bereits in einer frühen Phase der Produktentstehung Fahrzeugkonzepte so zu entwickeln, dass diese im späteren Projektverlauf kaum noch verändert werden müssen. Dies wird dadurch erreicht, dass alle relevanten Marktanforderungen an ein Produkt von Beginn an bei der Konzeption berücksichtigt werden. Dieses Vorgehen birgt vor allem den Vorteil, dass nach Abschluss der Frontloading-Phase nur noch über ökonomisch sinnvolle und technisch realisierbare Alternativen von Fahrzeugkonzepten entschieden werden muss. Dadurch werden viele Verzögerungen im weiteren Prozessverlauf vermieden und die gesamte Produktentwicklung stabiler und kostengünstiger.

Als Voraussetzung für die Entwicklung marktgerechter Produkte gilt das Anforderungsmanagement. In der frühen Phase des Projekts erheben Projektteams, bestehend aus interdisziplinären Domänen-Experten, zunächst die Anforderungen an alle

[1] Daimler AG, T/OGP, Mercedesstr. 132/1, 70546 Stuttgart, konstantin.zichler@daimler.com
[2] Brandenburgische Technische Universität Cottbus-Senftenberg, steffen.helke@b-tu.de

Lebensphasen eines Nutzfahrzeugs. Jeder Domänen-Experte definiert die Anforderungen, die aus Sicht der eigenen Abteilung erforderlich sind. Dabei handelt es sich um funktionale und nichtfunktionale Anforderungen. Anschließend entwickelt das Projektteam auf Basis der Anforderungen Realisierungskonzepte und bestätigt die technische Machbarkeit, sowie die Wirtschaftlichkeit des Projekts.

Die frühe Phase in der Produktentstehung von Nutzfahrzeugen erfordert insbesondere eine präzise Abstimmung zwischen den Domänen-Experten, wobei unterschiedlichste Einflussfaktoren berücksichtigt werden müssen. Als maßgebliches Instrument dient dabei das Projektlastenheft. Darin dokumentieren Projektteams Anforderungen, Realisierungskonzepte und andere für die Projektabwicklung wichtige Informationen. Das Projektlastenheft dient vor allem als Kommunikationsgrundlage zwischen den Domänen-Experten und soll ein gemeinsames Verständnis für das angestrebte Projektziel schaffen. Um ein Verständnis der Inhalte für alle Projektteammitglieder gleichermaßen zu ermöglichen, werden Anforderungen und Realisierungskonzepte in natürlicher Sprache in Office-Dokumenten dokumentiert. Das Projektlastenheft ist ferner die Grundlage für den weiteren Spezifikationsprozess und die anschließende Produktentwicklung. Alle Abhängigkeiten zwischen den Anforderungen müssen durch sorgfältige Analysen identifiziert und bewertet werden, da sie zu Widersprüchen und damit fehlerhaften Fahrzeugkonzepten führen können.

Bisher suchen Projektteams manuell nach Abhängigkeiten im Projektlastenheft, da es hierfür kaum Werkzeugunterstützung gibt. Neben der Tatsache, dass das Projektlastenheft ein sehr umfangreiches Dokument ist, stellen die unterschiedlichen Perspektiven der Fachexperten dabei eine der wesentlichen Herausforderungen für eine erfolgreiche Abhängigkeitsanalyse dar. Das Wissen über Zusammenhänge im Projekt liegt verteilt vor, hauptsächlich in den Köpfen der Domänenexperten. Mit seinem Beitrag im Projektlastenheft dokumentiert der jeweilige Fachexperte nur einen Teil seines Wissens. Für das Projetteam besteht die Herausforderung folglich darin, von den dokumentierten Einzelbeiträgen auf domänenübergreifende Abhängigkeiten zwischen den Anforderungen zu schließen. Ein weiterer Faktor sind die impliziten Annahmen der einzelnen Projektteammitglieder. Pohl et al. verweisen bspw. darauf, dass der häufigste Grund für unvollständige Anforderungen falsche Annahmen der Stakeholder sind. Sie setzten z.B. voraus, dass bestimmte Informationen selbstverständlich sind und deshalb gar nicht explizit genannt werden müssen [PR11].

Anhand des folgenden Beispiels wird diese Problematik deutlich. Ein Plug-in-Hybrid-Fahrzeug soll auf der Grundlage einer vorhandenen Fahrzeugplattform entwickelt werden. Das Projektteam entscheidet sich bei der Definition des Realisierungskonzepts dafür, das Bordnetz der vorhandenen Fahrzeugplattform zu übernehmen. Wenn das Bordnetz zuvor in einem Fahrzeug mit Verbrennungsmotor eingesetzt wurde, wäre die Lebensdauer des Bordnetzes für ein Plug-in-Hybrid-Fahrzeug nicht ausreichend. Der Grund dafür ist, dass Teile des Bordnetzes eines Plug-in-Hybrid-Fahrzeugs auch im Stillstand des Motors aktiv sind, nämlich während der Beladung des Akkumulators über eine externe Stromquelle. In diesem Beispiel muss das Projektteam also die

Abhängigkeiten zwischen den Anforderungen an die Nutzung des Plug-in-Hybrid-Fahrzeugs, wie das Fahren und das Laden des Akkumulators, und den Anforderungen an die Lebensdauer des Bordnetzes ausreichend berücksichtigen. Da diese Anforderungen aber in einem realen Projekt von unterschiedlichen Domänen-Experten aus Vertrieb, Entwicklung und Qualität erhoben werden, können bei einer manuellen Analyse solche Abhängigkeiten auch leicht unerkannt bleiben.

Um derartige Anforderungen besser aufdecken zu können, schlagen wir vor, das verteilte Wissen über die Produktentstehung von Nutzfahrzeugen in einer Wissensbasis zu konzentrieren und basierend auf den gesammelten Informationen domänenübergreifende Abhängigkeiten zwischen Anforderungen abzuleiten. Das Wissen soll dabei mit Hilfe von Ontologien repräsentiert werden. Eine Ontologie ist eine formale, explizite Spezifikation einer gemeinsamen Konzeptualisierung [SBF98]. Sie kann als eine Wissensbasis verstanden werden, die das Wissen einer Anwendungsdomäne beschreibt. Mit Hilfe von Methoden zur Schlussfolgerung ist es außerdem möglich, neue, implizite Informationen aus dem bereits spezifizierten Wissen abzuleiten [HKR+08]. Für die Prüfung der Ontologie auf Konsistenz und zur Ableitung von implizitem Wissen werden Inferenzmaschinen (engl. *Reasoner*) eingesetzt.

Der Einsatz von Ontologien im Anforderungsmanagement wurde in der Vergangenheit bereits in einigen Arbeiten vorgestellt, vgl. dazu [FMK+11], [SB16], [SP14], [Si14] und [SHS14]. Dabei fokussieren diese Ansätze eher auf detaillierte und vor allem technische Anforderungen, wie sie für Feinspezifikationen im späteren Verlauf der Produktentstehung typisch sind. Demgegenüber enthält ein Projektlastenheft vor allem grobe Anforderungen und strategische Zielbeschreibungen an alle Produktlebensphasen. Das bedeutet, dass dabei auch Anforderungen aus den Abteilungen, wie Vertrieb, Produktion, Logistik und After-Sales berücksichtigt werden müssen. Schraps und Peters [SP14] schränken in ihrer Arbeit bspw. die Satzstruktur der Anforderungen bereits bei ihrer Formulierung durch eine formale Grammatik ein. Die so formulierten Anforderungen werden mit Hilfe von semantischen Mustern in eine Ontologie überführt. In der Arbeit von Farfeleder et al. [FMK+11] werden Schablonen (engl. *boilerplates*) zur standardisierten Beschreibung von Anforderungen verwendet. Über eine graphische Schnittstelle wird vom Nutzer zunächst eine vorgegebene Schablone ausgewählt. Anschließend bekommt er für die Befüllung der Leerfelder vom Werkzeug Vorschläge, die aus einer Anforderungsontologie stammen. Solche Vorgehen erfordern grundsätzlich gut strukturierte Spezifikationsdokumente und viel Erfahrung der Nutzer im Umgang mit Anforderungsmanagement. Die frühe Phase in der Produktentstehung erfordert jedoch Lastenhefte, deren Erstellung möglichst flexibel gehandhabt werden kann, da große Teile eines Fahrzeugkonzepts durch Beschluss wieder entfallen können. Insbesondere durch die häufigen Änderungen am Fahrzeugkonzept würde eine exakte Dokumentation ohnehin nur unnötig die Arbeit der Domänenexperten behindern. Daher kann im vorliegenden Anwendungsfall von einer Struktur, wie sie bspw. bei den Spezifikationen mit hohem Detaillierungsgrad vorhanden ist, nicht ausgegangen werden. Eine Modellierung der Anforderungen in Form einer Ontologie durch Domänen-Experten muss aufgrund des zu hohen Aufwands ebenfalls ausgeschlossen werden.

Aus diesem Grund schlagen wir für die Abhängigkeitsanalyse im Projektlastenheft eine Kombination aus einer Ontologie in Verbindung mit einem Reasoner und Natural Language Processing (NLP) vor. Nach unserem Konzept wird ein Teil der Ontologie, das sogenannte Domänenmodell, welches das für die Produktentstehung erforderliche Wissen enthält, von Mitarbeitern aus einem zentralen Projektmanagement Office manuell erstellt. Dieses Domänenmodell beschreibt vor allem die Klassenebene der Ontologie. Die Instanzebene des Domänenmodells soll unter Zuhilfenahme von NLP vollautomatisiert erstellt werden. Damit soll es möglich sein, in einem gegebenen Projektlastenheft eine volltautomatisierte Abhängigkeitsanalyse durchzuführen. Diese Analyse wird von einem zentralen Bereich durchgeführt. Das Projektteam bekommt eine Auswertung über die ermittelten Abhängigkeiten und kann diese bei der Konzeption des Fahrzeugs berücksichtigen. Da es für diesen Anwendungsfall noch keine dedizierte Werkzeugunterstützung gibt, nutzen wir Plug-ins innerhalb der bekannten Architekturen Protégé [Mu15] und GATE [CTR+13], um die grundsätzliche Machbarkeit einer vollautomatischen Abhängigkeitsanalyse im Projektlastenheft nachzuweisen.

Das Papier gliedert sich wie folgt. In Abschnitt 2 werden die Grundlagen zu Ontologien und NLP erläutert. In Abschnitt 3 stellen wir unseren Ansatz für eine ontologiebasierte Abhängigkeitsanalyse anhand einer prototypischen Werkzeugkette vor. In Abschnitt 4 beschreiben wir erste Experimente mit der prototypischen Werkzeugkette und diskutieren die Ergebnisse. Dabei weisen wir die grundsätzliche Machbarkeit einer ontologiebasierten Abhängigkeitsanalyse im Projektlastenheft nach. Abschnitt 5 befasst sich mit verwandten Arbeiten. Unsere Ergebnisse fassen wir in Abschnitt 6 zusammen und geben darüber hinaus einen Ausblick über unser weiteres Vorgehen.

2 Grundlagen

Ziel dieses Kapitels ist es, Grundlagen über Aufbau, Erstellung und Anwendung von Ontologien zu vermitteln, sowie bekannte Methoden und Werkzeuge aus dem Natural Language Processing (NLP) vorzustellen.

2.1 Erstellung und Anwendung von Ontologien

Der größte Mehrwert von Ontologien in Verbindung mit Methoden zur Schlussfolgerung ist die Ableitbarkeit von neuem Wissen. Die Grundlage dafür ist eine Wissensbasis – die Ontologie. Eine Ontologie beschreibt das Wissen einer Anwendungsdomäne durch die Begriffe (Konzepte), die innerhalb dieser Domäne genutzt werden und ihre Beziehungen (Relationen) zueinander. In Abb. 1 ist eine schematische Darstellung einer Ontologie zu sehen. Dabei sei zunächst auf die Klassen der Ontologie verwiesen, die hier mit schwarzen Kreisen markiert sind. Klassen beschreiben die übergeordneten Begriffe einer Domäne. Klassen können Unterklassen besitzen. Im abgebildeten Beispiel hat die Klasse *Vehicle* eine Unterklasse *Distribution*. Diese Unterklasse spezifiziert eine bestimmte Art eines Fahrzeugs, nämlich die eines Fahrzeugs für den Verteilerverkehr. Klassen können

über Relationen miteinander verknüpft werden. Zwei Klassen, und die sie verbindende Relation bilden ein sogenanntes Tripel. Das Beispiel in Abb. 1 zeigt bspw. das Tripel (*CANBus, isPartOfVehicle, Distribution*) und drückt damit aus, dass ein CAN-Bus immer in der *Distribution* eines Fahrzeugs enthalten sein muss. Eine konkrete Ausprägung einer Klasse wird als Instanz oder Individuum bezeichnet. Instanzen sind in Abb. 1 durch schwarze Rauten gekennzeichnet. *EBus2X* ist bspw. ein fiktiver Eigenname für eine Instanz der Klasse *CAN-Bus*.

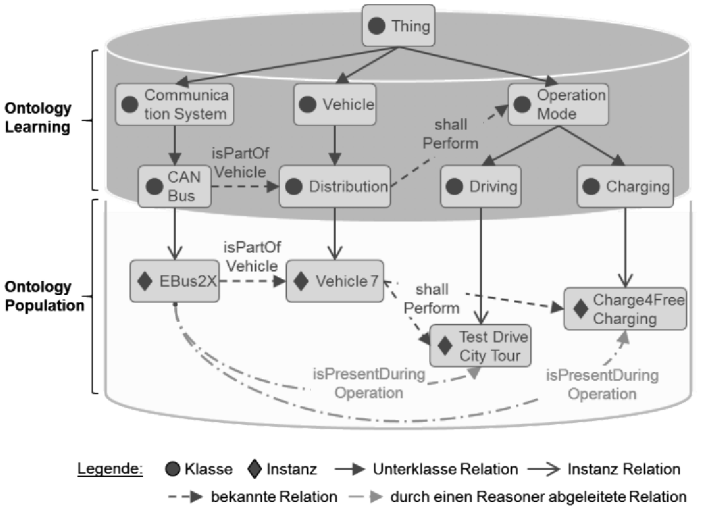


Abb. 1: Schematische Darstellung einer Ontologie

Ontologien werden in Ontologiesprachen beschrieben, wie bspw. OWL 2 (Web Ontology Language). OWL basiert auf der Prädikatenlogik erster Stufe [HKR+08].

Die Erstellung von Ontologien gliedert sich in zwei Teile, das *Ontology Learning* und das *Ontology Population* (siehe Markierungen in Abb. 1). *Ontology Learning* beschreibt die Erzeugung neuer Klassen (Konzepte) und Relationen in der Ontologie. Dadurch wird die innere Struktur der Ontologie erweitert. Demgegenüber zielt *Ontology Population* auf die Instanziierung dieser Klassen und Relationen [RMC+11].

Für gewöhnlich werden Ontologien manuell erstellt, was durchaus aufwändig sein kann. Für die Erstellung werden Ontologie-Editoren genutzt. Ein bekanntes Beispiel ist der Ontologie-Editor der Stanford University Protégé [Mu15]. Weiterhin gibt es Möglichkeiten, Ontologien oder Teile davon automatisiert zu erstellen. In der Vergangenheit wurden dazu NLP-Techniken innerhalb der General Architecture for Text Engineering (GATE) genutzt, vgl. dazu [WKR10] und [MFP09]. Dabei ist bisher vor allem die automatisierte Erstellung von flachen Klassenhierarchien mit nicht immer ausreichender Genauigkeit möglich. Im Gegensatz zum *Ontology Learning*, lässt sich *Ontology Population*, also die Anreicherung einer Ontologie mit Instanzen und den

Beziehungen zwischen ihnen, weitestgehend automatisieren.

In einer Ontologie liegt Wissen in einer formalisierten Form vor. Die Maschinenlesbarkeit dieses Wissen ermöglicht es, Schlussfolgerungen über die Inhalte der Ontologie automatisiert zu ziehen. Zu diesem Zweck werden die bereits erwähnten Reasoner genutzt. Sie prüfen die Wissensbasis auf Konsistenz und leiten neue Informationen aus dem bereits spezifizierten Wissen ab. Zu diesem Zweck werden Axiome und Ausdrücke definiert. Auch Relationen können genauer spezifiziert werden, um bspw. auszudrücken, dass zwei Klassen Synonyme sind, wenn diese in der betrachten Domäne dieselbe Semantik besitzen. In Abb. 1 kommt die Relation *isPresentDuringOperation* zweimal vor und ist im Gegensatz zu den anderen Relationen durch eine Strichpunktlinie dargestellt. Diese besondere Notation zeigt an, dass diese Relation auf der Grundlage des nachfolgenden Object-Subproperty-Axioms mit Hilfe eines Reasoners abgeleitet worden ist:

```
SubObjectPropertyOf(ObjectPropertyChain a:isPartOfVehicle
a:shallPerform)a:isPresentDuringOperation)
```

Dieses Axiom wird auch als *Property Chain* bezeichnet und sagt aus, dass beim direkten Aufeinanderfolgen von zwei vorgegebenen Relationen (im Beispiel *isPartOfVehicle* und *shallPerform*) die Endpunkte der entstehenden Kette mit einer dritten Relation (im Beispiel *isPresentDuringOperation*) zu verknüpfen sind.

2.2 Natural Language Processing

Die Automatisierung von *Ontology Population* erfolgt in der vorliegenden Arbeit mit NLP innerhalb von GATE. Die General Architecture for Text Engineering (GATE) ist eine Open Source Software, deren Hauptzweck darin besteht, Dokumente zu annotieren. Diese Annotationen können in GATE automatisch oder manuell erstellt werden. Die automatische Erstellung von Annotationen im Rahmen unseres Ansatzes erfordert unter anderem die folgenden Anwendungen, die in GATE als *Processing Resources* bezeichnet werden: *Tokeniser*, *Sentence Splitter*, *Part of Speech (POS) Tagger*, *Gazetteer*, *JAPE Transducer*. In [Cu14] werden diese wie folgt erläutert:

- *Tokeniser* spalten Text in sehr kleine Token, wie Nummern, Satzzeichen und Worte verschiedener Art.
- *Gazetteer* bestehen aus Listen, die Namen von Entitäten, wie bspw. Namen von Städten, Organisationen oder Wochentagen enthalten. Diese Listen werden dazu genutzt Begriffe wie Eigennamen im Text zu suchen (Named Entity Recognition). Alle Zeichenketten im Text, die mit dem Eigennamen einer der genutzten *Gazetteer*-Liste übereinstimmen, werden mit der Annotation Lookup markiert.
- *Sentence Splitter* segmentieren, wie der Name schon sagt, Text in Sätze. Dabei wird eine *Gazetteer*-Liste mit Abkürzungen verwendet, um diese von Satzzeichen zu unterscheiden, die ein Satzende markieren.

- *Part-of-Speech Tagger* erstellen Annotationen zu jedem Wort oder Symbol im Text, die angeben, um welche Wortart es sich bei dem jeweiligen Wort oder Symbol handelt.

Die so erstellten Annotationen können anschließend mit dem *JAPE Transducer* verändert werden. JAPE (Java Annotation Patterns Engine) erlaubt die Erkennung von Regulären Ausdrücken in Annotation, die auf einem Text erstellt wurden. Für die Anwendung von JAPE werden im ersten Schritt Grammatiken erstellt. Eine JAPE Grammatik besteht aus Phasen, von welchen jede aus Muster-Aktion-Regeln besteht. Die Phasen laufen sequentiell durch und stellen eine Kaskade von Transduktoren über Annotationen dar. Eine Regel hat eine linke und eine rechte Seite. Die linke Seite der Regeln beschreibt jeweils Muster von Annotationen, die gesucht werden sollen. Die rechte Seite besteht aus einer Anweisung für die Manipulation dieser Annotationen. Annotationen, die in das Muster der linken Seite der Regel passen, werden mit einem Label versehen. Dadurch kann auf der rechten Seite der Regel auf dieses Label Bezug genommen werden [Cu14]. Mit dem *JAPE Transducer* ist es möglich, bestehende Annotationen zu verändern oder nach Textbestandteilen zu suchen und diese zu annotieren. Nachfolgend ist eine Regel dargestellt, die im Text nach einem Token mit der Zeichenkette *Vehicle* gefolgt von einer Zahl sucht und diesen als *DistributionTruck* annotiert:

```
Rule: DistributionTruck
(
     {Token.string == Vehicle} {Token.kind == number}
):tag
-->
:tag.DistributionTruck = {rule = "DistributionTruck"}
```

Mit dieser Regel könnte bspw. nach Fahrzeugmodellen im Text gesucht werden. Alle *Processing Resources* werden in einer *Processing Pipeline* angeordnet und von GATE sequentiell ausgeführt. Am Ende liegt ein annotierter Text vor, der Aufschluss über die Semantik der Textbestandteile gibt. Diese Informationen können dazu verwendet werden, die Inhalte des Texts gezielt auszuwählen und weiterzuverarbeiten.

3 Ontologiebasierte Abhängigkeitsanalyse im Projektlastenheft

In diesem Kapitel präsentieren wir eine prototypische Umsetzung für die Abhängigkeitsanalyse im Projektlastenheft. Weiterhin stellen wir Überlegungen über die Einbindung des Lösungsansatzes in die organisatorischen Abläufe eines Industrieunternehmens an.

3.1 Prototypische Werkzeugkette

Für die Lösung der Aufgabenstellung verwenden wir eine prototypische Werkzeugkette,

die im Wesentlichen aus Protégé [Mu15], GATE [CTR+13] und dem OwlExporter [WKR10] besteht. Die Abhängigkeitsanalyse mit dieser Werkzeugkette erfordert ein bereits bestehendes Domänen-Modell in Form einer Ontologie. Dieses Domänen-Modell wird manuell erstellt (*Ontology Learning*) und besteht aus Klassen, Relationen, Axiomen und Ausdrücken. Die Begriffe (Konzepte) für die Ontologie stammen aus den an der Projektabwicklung beteiligten Domänen. Sie werden aus bereits vorhandenen Projektlastenheften und anderen Projektdokumenten gesammelt, in eine hierarchische Beziehung gebracht und über Relationen miteinander in Beziehung gesetzt. Grundsätzliche Zusammenhänge, wie bspw., dass alle Fahrzeuge auf Verkehrswegen fahren, werden über Axiome und Ausdrücke in OWL 2 formuliert und in der Ontologie hinterlegt. Die ontologiebasierte Abhängigkeitsanalyse unter Verwendung von NLP ist in Abb. 2 dargestellt. Nachfolgend werden die Schritte der Analyse näher beschrieben:

1. Das zuvor manuell erstellte Domänen-Modell wird als RDF/XML-Dokument exportiert und in einem Ordner gespeichert. Dieses Modell ist die Grundlage für die spätere *Ontology Population*.

2. Projektlastenhefte werden in GATE importiert. Es können gleichzeitig mehrere Dokumente importiert werden. Alle Dokumente werden automatisch in sogenannte GATE-Dokumente konvertiert.

3. Aus den GATE-Dokumenten wird ein Korpus erstellt. An dem Korpus erfolgt die anschließende sprachliche Analyse.

4. Der Korpus durchläuft die Komponenten *Document Reset PR*, *English Tokeniser*, *Gazetteer*, *Sentence Splitter*, *POS Tagger*, *NE Transducer* und OrthoMatcher aus der Anwendung ANNIE [CMB+02]. Die einzelnen Bestandteile des Textes werden entsprechend ihrer Zugehörigkeit von ANNIE annotiert. Als Ergebnis liegt ein annotierter Korpus vor. Zur besseren Übersichtlichkeit wurden die Komponenten für die sprachliche Analyse in Abb. 2 zu einem Schritt zusammengefasst.

5. Ausgehend von zuvor definierten JAPE-Regeln, sucht der *JAPE Transducer* nach Annotationen im Korpus. Die JAPE-Regeln sind Teil der OwlExporter Demo [WKR10] und wurden für den vorliegenden Anwendungsfall angepasst. Das Ziel dieses Schrittes ist es, Instanzen für die *Ontology Population* zu identifizieren. Der *JAPE Transducer* erstellt zusätzliche Annotationen, die die Begriffe für den anschließenden Export markieren.

6. Der OwlExporter, ein Java Plug-in, identifiziert die Instanzen und Relationen, die exportiert werden sollen, anhand der Annotationen im Korpus. Danach exportiert der OwlExporter Instanzen und Relationen in die Ontologie. Für diesen Schritt wird die zuvor erstellte Ontologie im RDF/XML-Format benötigt. Sie dient als Ausgangsdokument für die *Ontology Population*. Der OwlExporter positioniert die im Korpus gefundenen Instanzen und Relationen automatisch an die richtige Stelle in der Ontologie. Das Ergebnis ist eine Ontologie im RDF/XML-Format.

7. Die Ontologie wird mit Protégé geöffnet.
8. Der Reasoner Fact++ [TH06] wird nun verwendet, um die Konsistenz der Ontologie zu prüfen und um neue (implizite) Abhängigkeiten in der Ontologie abzuleiten.
9. Mit dem SPARQL Query Panel werden die Abhängigkeiten innerhalb der Ontologie abgefragt. Als Ausgabe werden sogenannte Tripel, also Instanz-Paare mit jeweils einer Relation abgefragt.

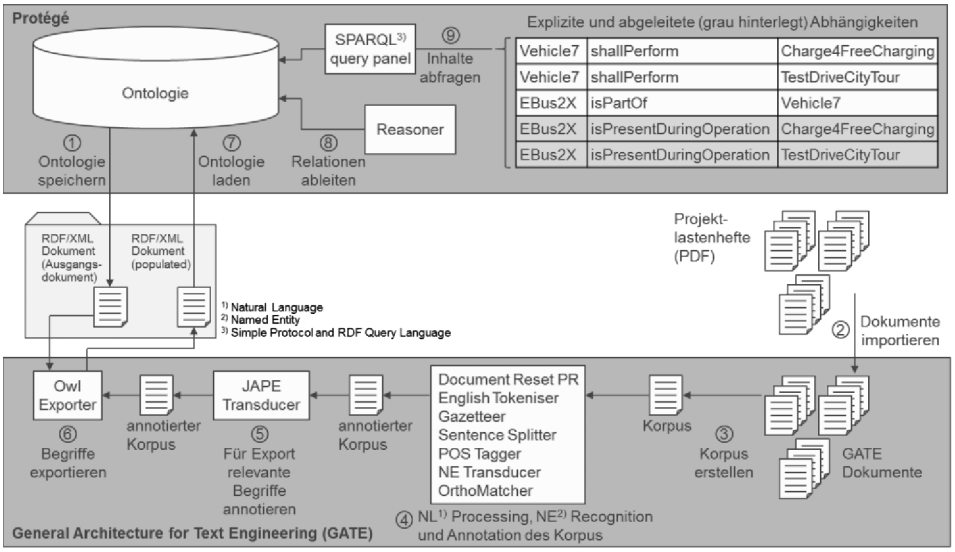


Abb. 2: Prototypische Werkzeugkette

3.2 Organisatorische Umsetzung

Die Randbedingungen in einem Industrieunternehmen lassen es nicht zu, dass Domänen-Experten eigenständig Wissen in Form einer Ontologie modellieren und sprachliche Analysen mit GATE durchführen. Neben dem Faktor Zeit spielen dabei auch noch die fehlenden Kenntnisse der Domänen-Experten im Bereich von Ontologien und NLP eine entscheidende Rolle. Aus diesem Grund schlagen wir vor, einen zentralen Bereich für Projektmanagementunterstützung, ein sogenanntes Project Management Office (PMO), mit der Durchführung der ontologiebasierten Abhängigkeitsanalyse zu beauftragen. In einigen Projekten der Fahrzeugentstehung übernimmt der Projektmanagementunterstützer (PMU) eine dem Requirements Engineer ähnliche Rolle. Er leitet das Projektteam bei der Erstellung des Projektlastenhefts an und hilft bei ausgewählten Tätigkeiten des Anforderungsmanagements. In seiner Querschnittsrolle hat er außerdem einen guten domänenübergreifenden Überblick und Zugriff auf erforderliche Projektunterlagen. Außerdem ist er als neutrale Person geeignet, um Schwachstellen im Konzept

aufzuzeigen, da er keine der am Projekt beteiligten Abteilungen vertritt und damit mit mehr Akzeptanz beim Aufzeigen von Fehlern rechnen kann. Eine organisatorische Umsetzung unseres Vorgehens ist in Abb. 3 dargestellt.

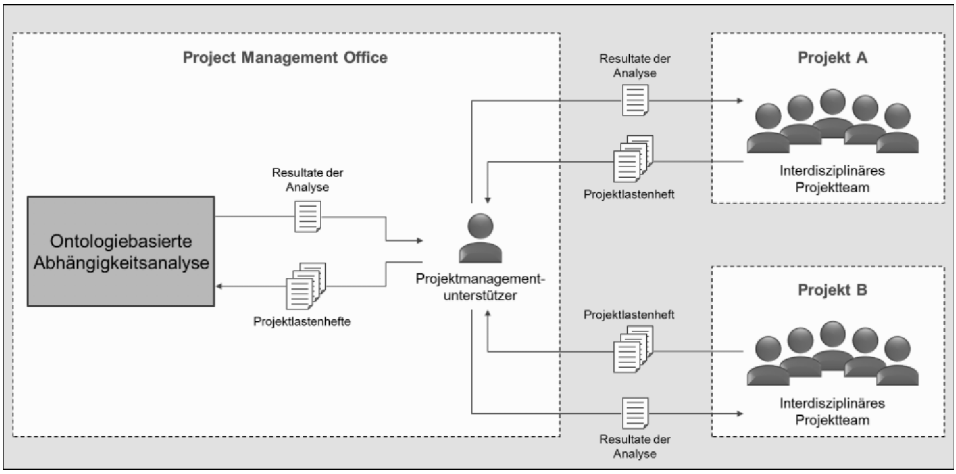


Abb. 3: Organisatorische Umsetzung der ontologiebasierten Abhängigkeitsanalyse

Dabei ist ein Projektmanagementunterstützer derjenige, der das Werkzeug für die ontologiebasierte Abhängigkeitsanalyse verwaltet. Er erstellt die Wissensbasis (Ontologie) und versorgt die Projektteams mit den Ergebnissen der ontologiebasierten Abhängigkeitsanalyse. Der Ablauf gliedert sich wie folgt:

1. Projektteams übergeben ihre Projektlastenhefte an den PMU.
2. Der PMU führt die Abhängigkeitsanalyse durch. Dabei filtert er die ermittelten Abhängigkeiten, um die Projektteams nicht mit unnötig vielen Informationen zu belasten.
3. Der PMU leitet die relevanten Ergebnisse an das jeweilige Projektteam weiter. Die Domänen-Experten werden über die Abhängigkeiten informiert und können Anforderungen ergänzen oder ändern und Anpassungen am Produktkonzept vornehmen.

4 Praktische Experimente und Diskussion der Ergebnisse

Zum Nachweis der prinzipiellen Machbarkeit unseres Ansatzes, haben wir einen Text aus fiktiven Anforderungen zusammengestellt. In Abb. 4 ist dieser Text bereits in GATE importiert. Dabei ist zu sehen, dass die für den Export relevanten Begriffe mit Hilfe der adaptierten OwlExporter Demo im Text identifiziert und für den Export mit der Annotation *OwlExportClassDomain* und *OwlExportRelationDomain* annotiert wurden.

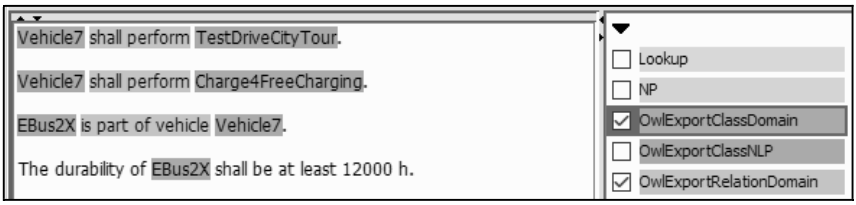


Abb. 4: Annotierte Begriffe für den Export mit OwlExporter Screenshot GATE

Im nächsten Schritt werden diese Begriffe von dem OwlExporter in das RDF/XML-Dokument exportiert (*Ontology Population*). Dieses Dokument enthält die angereicherte Ontologie, die im nächsten Schritt mit Protégé geladen wird.

In Abb. 5 ist das Ergebnis des *Ontology Population* und der Schlussfolgerung durch den Reasoner zu sehen. Die beiden Instanzen *EBus2X* und *Vehicle7* wurden zusammen mit der Relation *isPartOfVehicle* aus dem Text exportiert. Daneben wurden auch die Instanzen *TestDriveCityTour* und *Charge4FreeCharging* sowie die Relation *shallPerform* aus dem Text extrahiert, die hier aber aus Gründen der Übersichtlichkeit nicht dargestellt sind.

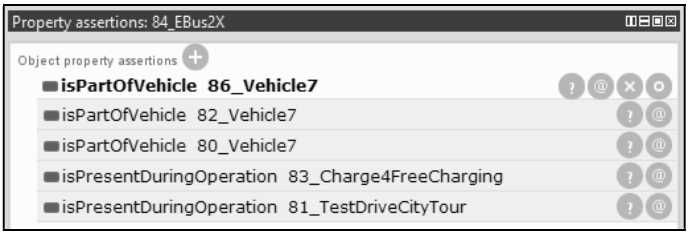


Abb. 5: Darstellung der Inferenz in Protégé

Aufgrund des zuvor definierten Axioms leitet der Reasoner aus dem vorhandenen Wissen, zusätzlich zwei neue Abhängigkeiten her:

```
(EBus2X, isPresentDuringOperation, TestDriveCityTour)
(EBus2X, isPresentDuringOperation, Charge4FreeCharging).
```

Diese zusätzlichen Relationen sind nun explizit und werden den Domänen-Experten angezeigt. Damit ist für das gesamte Projektteam klar, dass *EBus2X* durch diese beiden Operationen ebenfalls betroffen ist. Als Folge können die Domänen-Experten die Anforderung an die Lebensdauer des Bordnetzes und anschließend das Produktkonzept bereits in der frühen Projektphase anpassen.

Dieses Beispiel zeigt nur einen kleinen Ausblick dessen, was möglich ist. Die Instanzen *TestDriveCityTour* und *Charge4FreeCharging* könnten ihrerseits weitere Verknüpfungen zu anderen Einflussfaktoren aufweisen. Diese könnten ebenfalls automatisiert mit *EBus2X* und anderen Konzepten der Ontologie verknüpft werden, so dass sich noch mehr Abhängigkeiten ergeben. Durch diese Art der Analyse kann die

Qualität eines Produktkonzepts in der frühen Phase der Produktentstehung zusätzlich gesteigert werden. Der manuelle Aufwand, der für die Modellierung der Klassen und Relationen anfällt, ist dabei verhältnismäßig gering. Außerdem kann das manuell erstellte Domänen-Modell mehrfach wiederverwendet werden. Der größte Teil der Ontologie, wird mit Hilfe des *Ontology Population* automatisiert erstellt. Speziell die laufenden Änderungen auf Instanzebene, können mit Hilfe des *Ontology Population* ebenfalls automatisiert werden. Dieses Vorgehen erlaubt die Analyse von sehr umfangreichen Dokumenten in kürzester Zeit. Da Teile eines Nutzfahrzeugs häufig parallel in unterschiedlichen Projekten entwickelt werden, können so die Projektlastenhefte von mehreren Projekten gleichzeitig auf Abhängigkeiten analysiert werden. Damit könnten nicht nur domänenübergreifende, sondern auch projektübergreifende Abhängigkeiten gefunden werden.

Bevor jedoch diese Werkzeugkette genutzt werden kann, muss einige Vorarbeit geleistet werden. Das Domänen-Modell muss erstellt und validiert werden. Weiterhin müssen die NLP-Komponenten angepasst werden. Es müssen bspw. JAPE-Regeln formuliert und Gazetteer-Listen erstellt werden. Von der Qualität dieser Vorarbeit ist folglich auch die Trefferquote bei der sprachlichen Analyse im Text abhängig. Daneben bleibt der Aufwand für die Pflege und Wartung der Wissensbasis und der Werkzeuge für die sprachliche Analyse.

5 Verwandte Arbeiten

Schraps und Bosler stellen einen Ansatz vor, mit dem sie das Wissen aus Software-anforderungen in eine Anforderungsontologie überführen. Dazu werden die Anforderungen im ersten Schritt mit Hilfe von NLP-Techniken annotiert. Anschließend sucht eine Mustererkennung nach vordefinierten Mustern in der Grammatik der Anforderungen. Teile der Anforderungen, die in diese Muster fallen, werden entsprechend ihrer Semantik in die Anforderungsontologie überführt [SB16]. In ihrer Arbeit präsentiert Siegemund eine Methode für eine automatisierte Validierung und Messung des Anforderungswissens. Mit der prototypischen Implementierung OntoReq demonstriert Siegemund, wie unvollständige und inkonsistente Anforderungen und Qualitätsfehler mit Hilfe einer Anforderungsontologie automatisch identifiziert werden. Außerdem wird der Requirements Engineer bei der Lösung dieser Fehler durch wissensbasierte Vorschläge angeleitet [Si14]. Schraps und Peters haben eine Methode entwickelt, bei der Anforderungen mit Hilfe von semantischen Mustern in eine Ontologie überführt werden. Diese Ontologie nutzen sie anschließend für die Prüfung der Konsistenz innerhalb des Spezifikationsdokuments. Ihre Methode basiert nicht auf NLP. Stattdessen schränken sie die Satzstruktur der Anforderungen bereits bei der Formulierung durch eine formale Grammatik ein [SP14]. Soomro et al. präsentieren eine ontologiebasierte Visualisierung von unterschiedlichen Abhängigkeitsbeziehungen zwischen Anforderungen. In ihrem Ansatz behandeln sie speziell die Auswirkungen von Änderungen der Anwenderanforderungen auf andere Anforderungen. Soomro et al.

gehen davon aus, dass Software-Teams Abhängigkeiten zwischen Anforderungen übersehen, weil diese nicht deutlich genug dargestellt werden [SHS+14]. Farfelder et al. nutzen eine Ontologie, um qualitativ hochwertige Anforderungen bereits bei ihrer Erhebung zu formulieren. In ihrem System kann der Requirements Engineer über eine graphische Benutzerschnittstelle aus einer Menge von vorgegebenen Boilerplates wählen. Das System nutzt eine Domänenontologie, um eine Liste mit Vorschlägen für die Einzelteile einer Anforderung zur Verfügung zu stellen, die der Requirements Engineer für die Definition der Anforderungen nutzen kann [FMK+11].

Im Vergleich zu den hier vorgestellten Ansätzen bietet unsere Vorgehensweise die Möglichkeit, weitgehend vollautomatisch die Abhängigkeiten in Projektlastenheften zu finden. Werden mehrere Projektlastenhefte analysiert, können sogar Abhängigkeiten zwischen verschiedenen Projekten identifiziert werden. Unser Ansatz bietet den Vorteil, dass die Ersteller von Projektlastenheften kein spezielles Know-how zu *Natural Language Processing* oder *Ontology Engineering* benötigen und auch keine formale Grammatik erlernen müssen. Die Analyse wird in ein zentrales Project Management Office ausgelagert, wodurch eine Entlastung der Projektmannschaft erreicht wird. Eine solide sprachliche Qualität der Projektlastenhefte steigert die Leistungsfähigkeit unserer Methode. Wir arbeiten aktuell an Erweiterungen, die auch bei Rechtschreib- oder Grammatikfehlern akzeptable Ergebnisse liefern.

6 Zusammenfassung und Ausblick

In dem vorliegenden Papier wurde ein neuartiges Verfahren für die Abhängigkeitsanalyse im Projektlastenheft vorgestellt. Das für die Abhängigkeitsanalyse erforderliche, fachspezifische Wissen ist dabei zu einer gemeinsamen Wissensbasis in Form einer Ontologie aggregiert. Zusammen mit Axiomen, einem Reasoner und Werkzeugen aus dem Natural Language Processing wurde von uns eine automatisierte Abhängigkeitsanalyse im Projektlastenheft realisiert, mit der es möglich ist, bisher nicht berücksichtigte Abhängigkeiten zwischen Anforderungen zu identifizieren. Mit unseren Ergebnissen aus ersten Tests haben wir die grundsätzliche Machbarkeit des Verfahrens nachgewiesen. Zukünftig gilt es sicherzustellen, dass das im Domänen-Modell hinterlegte fachspezifische Wissen korrekt ist. Aus diesem Grund arbeiten wir derzeit an einem Verfahren, mit dem es möglich ist, eine Ontologie zu validieren. Dabei wollen wir die spezifischen Rahmenbedingungen innerhalb der Produktentstehung von Nutzfahrzeugen und die Fähigkeiten der interdisziplinären Domänen-Experten berücksichtigen. Weiterhin wollen wir quantitative Aussagen über im Projektlastenheft tatsächlich vorhandene und durch das vorgestellte Verfahren gefundene Abhängigkeiten treffen können. Zu diesem Zweck werden ausführliche Tests mit der vorgestellten Werkzeugkette durchgeführt.

Literaturverzeichnis

[CMB+02] Cunningham, H.; Maynard, D.; Bontcheva, K.; Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proc. of the 40th Annual Meeting of the Association for Computational Linguistics. 2002.

[CTR+13] Cunningham, H.; Tablan, V.; Roberts, A.; Bontcheva, K.: Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics. PLoS Computational Biology 9(2), 2013.

[Cu14] Cunningham, et al.: Developing Language Processing Components with GATE Version 8. University of Sheffield, Department of Computer Science. 17 Nov. 2014.

[FMK+11] Farfeleder, S.; Moser, T.; Krall, A.; St˚alhane, T.; Omoronyia, I.; Zojer, H.: Ontology-Driven Guidance for Requirements Elicitation, Springer-Verlag, Berlin u.a., 2011.

[HKR+08] Hitzler, P.; Krötzsch, M.; Rudolph, S.; Sure, Y.: Semantic Web: Grundlagen. Springer-Verlag, 2008.

[MFP09] Maynard, D.; Funk, A.; Peters, W.: SPRAT: a tool for automatic semantic pattern-based ontology population. Proc. of the Int. Conf. for Digital Libraries and the Semantic Web, Trento, Italy, 2009.

[Mu15] Musen, M., A.: The Protégé project: A look back and a look forward. AI Matters. ACM 1(4), June 2015.

[PR11] Pohl, K.; Rupp, C.: Basiswissen Requirements Engineering – Aus- und Weiterbildung zum Certified Professional for Requirements Engineering. dpunkt-Verlag, 2011.

[RMC+11] Ruiz-Martínez, J., M.; Minarro-Giménez, J., A.; Castellanos-Nieves, D.; García-Sanches, F.; Valencia-García, R.: Ontology Population: An Application for the E-Tourism Domain. Int. Journal of Innovative Computing, Information and Control, 7(11), 2011.

[SB16] Schraps, M.; Bosler, A.: Knowledge Extraction from German Automotive Software Requirements using NLP-Techniques and a Grammar-based Pattern Detection. The 8th Int. Conf. on Pervasive Patterns and Applications, pp. 17-21, 2016.

[SBF98] Studer, R., V.; Benjamins, R.; Fensel, D.: Knowledge engineering: Principles and methods. Data & Knowledge Engineering, 25(1-2):161–197, 1998.

[Si14] Siegemund, K.: Contributions to Ontology-Driven Requirements Engineering, 2014.

[SHS+14] Soomro, S.; Hafeez, A.; Shaikh, A.¸Musavi, S.: Ontology Based Requirement Interdependency Representation and Visualisation. In Communication Technologies, Information Security and Sustainable Development, Springer-Verlag, pp. 259-270, 2014.

[SP14] Schraps, M.; Peters, M.: Semantic Annotation of a Formal Grammar by Semantic-Patterns, IEEE 4th Int. Workshop on Requirements Patterns, 2014.

[TH06] Tsarkov, D.; Horrocks, I.: FaCT++ description logic reasoner: System description. In Proc. of the Int. Joint Conf. on Automated Reasoning, 2006.

[WKR10] Witte, R.; Khamis, N.; Rilling, J.: Flexible Ontology Population from Text: The Owl-Exporter, Int. Conf. on Language Resources and Evaluation, pp. 3845--3850, 2010.

Performing a More Realistic and Complete Safety Analysis by Means of the Six-Variable Model

Nelufar Ulfat-Bunyadi[1] Denis Hatebur[2] Maritta Heisel[3]

Abstract: Safety analysis typically consists of hazard analysis and risk assessment (HARA) as well as fault tree analysis (FTA). During the first, possible hazardous events are identified. During the latter, failure events that can lead to a hazardous event are identified. Usually, the focus of FTA is on identifying failure events within the system. However, a hazardous event may also occur due to invalid assumptions about the system's environment. If the possibility that environmental assumptions turn invalid is considered during safety analysis, a more realistic and complete safety analysis is performed than without considering them. Yet, a major challenge consists in eliciting first the 'real' environmental assumptions. Developers do not always document assumptions, and often they are not aware of the assumptions they make. In previous work, we defined the Six-Variable Model which provides support in making the 'real' environmental assumptions explicit. In this paper, we define a safety analysis method based on the Six-Variable Model. The benefit of our method is that we make the environmental assumptions explicit and consider them in safety analysis. In this way, assumptions that are too strong and too risky can be identified and weakened or abandoned if necessary.

Keywords: safety analysis, hazard analysis, risk analysis, fault tree analysis, assumption, environment, six-variable model

1 Introduction

The focus of FTA is usually on identifying the failure events within the system that can lead to a hazardous event (cf. e.g. [IS11], [RS15]). Yet, a system is always embedded in an environment. It is designed for this environment and satisfies its requirements during operation only if the assumptions about this environment that were made during development are valid [ZJ97]. A hazardous event may also occur due to environmental assumptions that turn out to be invalid. Van Lamsweerde describes in his book [La09] several accidents in which wrong or invalid assumptions about the environment led to a hazardous event, while the system behaved as specified. A famous example is the Lufthansa A320 flight to Warsaw. The plane ran off the end of the waterlogged runway resulting in injuries and loss of life. The reverse thrust was disabled for up to nine seconds after landing. The reason was the following. The autopilot had the task to enable reverse thrust only if the plane is moving on the runway. Since the wheels were not turning due to aquaplaning, the autopilot 'assumed' that the plane is not moving on the runway and thus disabled reverse thrust. So, the system behaved as specified but the hazardous event occurred nevertheless

[1] University of Duisburg-Essen, Oststrasse 99, 47057 Duisburg, nelufar.ulfat-bunyadi@uni-due.de
[2] Institut für technische Systeme GmbH, Emil-Figge-Str. 76, 44227 Dortmund, d.hatebur@itesys.de
[3] University of Duisburg-Essen, Oststrasse 99, 47057 Duisburg, maritta.heisel@uni-due.de

due to the wrong assumption. Due to such accidents, we argue that safety analysis should not only focus on hazardous events resulting from failure events within the system. Wrong or invalid environmental assumptions should also be considered as possible causes of a hazardous event. This will result in a more realistic and more complete safety analysis. To realize that it is necessary (i) to make the environmental assumptions explicit and (ii) to take the probability for their invalidity into account in safety analysis. In previous work, we developed the Six-Variable Model which provides the required support in making the environmental assumptions explicit. In this paper, we present a safety analysis method that is based on the Six-Variable Model and extends traditional safety analysis in order to consider also invalid environmental assumptions as possible causes of hazardous events.

The paper is structured as follows. In Section 2, we first introduce the Six-Variable Model. In Section 3, we briefly describe some fundamentals that we use. Our extensions to safety analysis are then presented in Section 4. To demonstrate that our method results in a more realistic safety analysis, we compare our method to traditional safety analysis in Section 5. Finally, we discuss related work in Section 6 and draw conclusions in Section 7.

2 The Six-Variable Model

The problem with assumptions is that developers do not always document the assumptions they make. Frequently, they are even not aware of them. Even if they are aware of the importance to document assumptions, they are left alone with the question which information to document. Our Six-Variable Model provides support in this regard.

The Six-Variable Model [UBMH16] is based on the well-known Four-Variable Model [PM95] and focuses on control systems. A control system consists of some control software which uses sensors and actuators to monitor/control certain quantities in the environment. In vehicles and air planes, we find a lot of control systems, e.g. ACC (Adaptive Cruise Control) or ESP (Electronic Stability Program). The Four-Variable Model defines the content of software documentation for control systems. Therein, Parnas and Madey differentiate between four variables. Monitored variables m are environmental quantities the control software monitors through input devices. Controlled variables c are environmental quantities the control software controls through output devices. Input variables i are data items that the control software needs as input and output variables o are quantities that the control software produces as output.

However, frequently, it is not possible to monitor/control exactly those variables one is interested in. Instead, a different set of variables is monitored/controlled, whose variables are related to the ones of real interest. The Six-Variable Model demands that the variables of real interest should be documented as well (beside the classical four variables). In addition, it should also be documented how the variables of real interest are related to the monitored/controlled ones. The Six-Variable Model is depicted in Figure 1 as a problem diagram. We first explain the notation and then the content of the model.

Problem diagrams have been introduced by Jackson [Ja01]. A problem diagram shows the so called machine (i.e. the software-to-be), its environment, and the requirement to be sat-

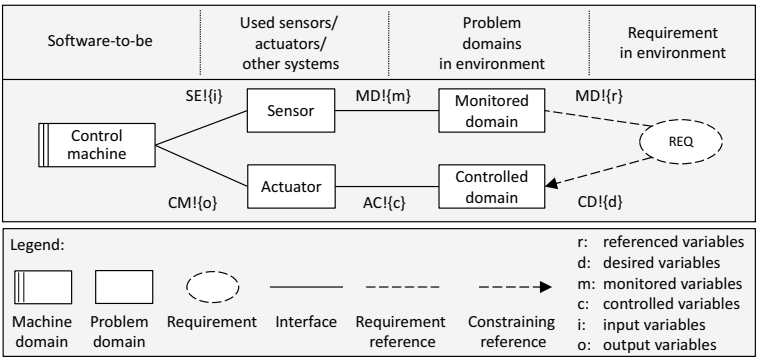


Fig. 1: The Six-Variable Model [UBMH16]

isfied. A problem diagram contains the following modelling elements: a machine domain, problem domains, a requirement, interfaces, requirement references, and constraining references. A problem domain represents any material or immaterial object in the machine's environment. The requirement is to be satisfied by the machine together with the problem domains. Interfaces exist between machine domain and problem domains or among problem domains. At the interfaces, phenomena (e.g. events, states, values) are shared. Sharing means that one domain controls a phenomenon, while the other observes it. At an interface, not only the phenomena are annotated but also an abbreviation of the domain controlling them followed by an exclamation mark (e.g. CM!). A requirement is connected to problem domains by means of a requirement reference or a constraining reference. A requirement reference expresses that the requirement refers somehow to the domain phenomena, while a constraining reference expresses that the requirement constrains (i.e. influences) them.

In the Six-Variable Model (Figure 1), the machine is the control software. It uses sensors and actuators to monitor/control certain phenomena of environmental domains (*m* and *c* variables). The requirement is shown on the right-hand side of the model. It refers to and constrains certain phenomena of the environmental domains. As stated above, these phenomena are not necessarily the same phenomena as the controlled/monitored ones. We call these phenomena the *r* and *d* variables. *r* (referenced) variables are environmental quantities that should originally be observed in the environment. Originally means before deciding which sensors/actuators/other systems to use for monitoring/controlling. As explained above, this decision frequently results in a different set of variables which are monitored/controlled. *d* (desired) variables are environmental quantities that should originally be influenced in the environment.

The benefit of making the six variables explicit is, among others, that we can make the environmental assumptions explicit based on them. Usually, we are the developers of the machine but the sensors/actuators are developed by other parties. We depend on them for satisfying the requirement *REQ* (see Figure 1). We assume, for example, that the sensors provide an *i* variable which actually reflects the corresponding *m* variable. Figure 2 shows the environmental assumptions for the Six-Variable Model. We differentiate between two

types of assumptions (based on [La09]): domain hypotheses and expectations. A domain hypothesis is a descriptive statement about the environment which needs to be valid. An expectation is a prescriptive statement to be satisfied by an environmental agent like a person, sensor, actuator. So, *DH-MD* is a hypothesis about the monitored domain, which needs to be true. *Exp-SE* is an expectation to be satisfied by the sensors, *Exp-AC* is an expectation to be satisfied by the actuators, and *Exp-CD* is an expectation to be satisfied by the controlled domain. *SOF* represents the software requirements which are to be satisfied by the control machine. The requirements *REQ* can only be satisfied, if *DH-MD* is valid and *Exp-SE, SOF, Exp-AC* as well as *Exp-CD* are satisfied.

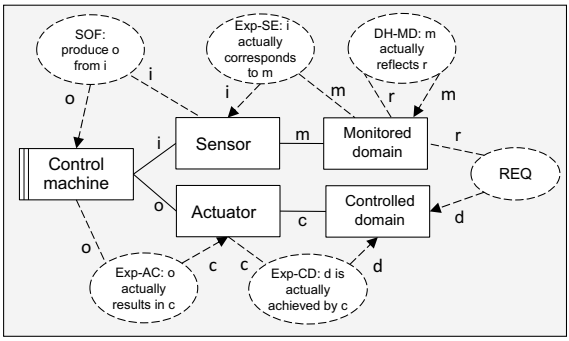


Fig. 2: Assumptions regarding the six variables [UBMH16]

Note that there may be several connection domains (i.e. a chain of sensors or a chain of actuators) between the machine and the environmental domains (Figure 1), especially in embedded systems. The existence of connection domains means that there are not only six variables to be documented but even $4+n$ variables. However, our Six-Variable Model supports that (see [UBMH16] for more details).

In case of the Lufthansa air plane (see Figure 3), for example, a referenced variable is the plane's movement on the runway. This phenomenon was observed by monitoring the turning of the wheels. A desired variable was the deceleration of the plane. This should be achieved by enabling the reverse thrust (controlled variable). Note that we show the same plane domain twice in Figure 3 and annotated it with an asterisk. Usually, this is not allowed in problem diagrams (i.e. a domain is only shown once in a diagram) but we did it here to enable the reader to see the similarity to Figure 1. Figure 3 shows also the environmental assumptions for the Lufthansa autopilot. Unfortunately, such a model was not created for the autopilot. Yet, making environmental assumptions explicit helps already since developers then start reflecting on them. However, we go one step further and ask for estimating the probability for their invalidity and taking these values into account in safety analysis. If it then turns out that assumptions are too strong and too risky, they can be weakened or abandoned. In this way, hazardous events resulting from invalid environmental assumptions can be prevented.

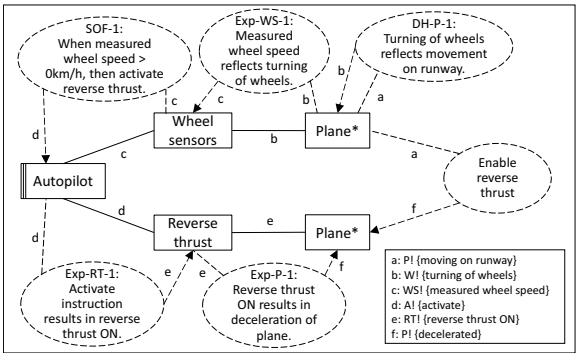


Fig. 3: Six variables in case of the Lufthansa autopilot

3 Fundamentals

Beckers et al. [Be13] have developed a hazard analysis and risk assessment (HARA) method which is compliant to the ISO 26262 Standard [IS11]. We use and extend this method in Section 4. Therefore, we introduce it here briefly. The goal of the HARA method is to identify potential hazards for the considered system and to formulate safety goals related to the prevention or mitigation of these hazards in order to achieve an acceptable residual risk. The method consists of seven steps. First, a so called context diagram[4] is created and a list of high-level requirements is defined. Second, for each high-level requirement, possible faults are identified using certain guide words (e.g. no, unintended, early, late) as a support. Third, for each high-level requirement, situations are selected from a given list of possible situations (during a vehicle's life) that are considered to be relevant for the requirement (e.g. parking manoeuvre, braking situation, highway situation). Fourth, for each fault/requirement combination, all situations that could lead to a potential hazard are identified in the list of situations being relevant (result from Step 3). Fifth, each hazard is classified according to its severity, exposure, and controllability in order to assess the required level of risk reduction (ASIL). The possible ASILs are: QM, ASIL A, ASIL B, ASIL C, and ASIL D. ASIL D is the highest level requiring the highest risk reduction and QM is the lowest level expressing that the normal quality measures applied in the automotive industry are sufficient. Sixth, safety goals are defined to address the hazards (i.e. to prevent or mitigate them). Seventh, the results of the hazard analysis and risk assessment are reviewed by an independent party.

4 Our Safety Analysis Method

Traditional safety analysis focuses on the system (cf. e.g. [IS11]). The system is defined as a set of elements that relates at least a sensor, a controller and an actuator with one another [IS11]. During traditional safety analysis, hazardous events occurring at the actuators are

[4] Context diagrams have also been introduced by Jackson [Ja01]. A context diagram is similar to a problem diagram but it shows only the machine and the problem domains connected to it.

identified (see Figure 1) and their causes are analysed until arriving at the sensors. Possible causes of a hazardous event (i.e. *c* variable is not achieved/effected) are then: a fault in the actuators, a fault in the machine, or a fault in the sensors.

Yet, actually, hazardous events occur at the controlled domains in the environment (see Figure 1). For example, in case of the Lufthansa air plane, the controlled variable was 'reverse thrust ON'. Yet, the desired variable was 'plane decelerated' (see Figure 3). This is what we actually wanted to achieve in the environment. During HARA, we should therefore identify situations in which this goal is not achieved (i.e. *d* variable is not achieved). Such hazardous events represent situations, in which assumptions of type *Exp-CD* (see Figure 2) turn out invalid. Furthermore, FTA should not stop at the sensors because there are further possible causes for a hazardous event. As pointed out in Section 2, we make the assumption that the monitored variables *m* reflect the referenced variables *r* (assumptions of type *DH-MD* in Figure 2). Yet, these are assumptions that can also turn invalid. Actually, this was the case in the Lufthansa example. The turning of the wheels did not reflect the plane's actual movement on the runway in that situation, i.e. *DH-P-1* (see Figure 3) turned out invalid. We need to consider the possibility that such assumptions turn invalid in the FTA. Therefore, FTA should not stop at the sensors but at the monitored domains. Further possible causes of a hazardous event are then: a 'fault' in the monitored domain, i.e. a wrong/invalid domain hypothesis.

In the following, we describe a safety analysis method that performs HARA and FTA in the way we just suggested. Our method is compliant to the ISO 26262 Standard [IS11]. It consists of the five steps depicted in Figure 4. The figure provides an overview of the method steps as well as inputs and outputs of each step. They are explained in the following.

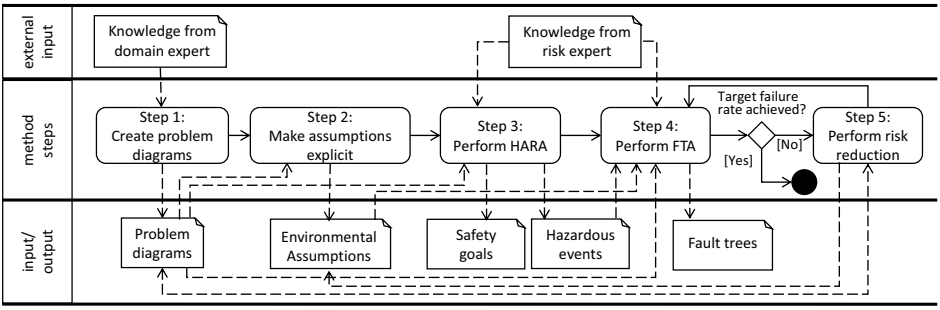


Fig. 4: Overview of our method

Step 1: Create problem diagrams. As a first step, one or (if necessary) several problem diagrams need to be created for the system's main functionality based on our Six-Variable Model presented in Section 2. In the ISO 26262 Standard, the creation of a functional architecture showing the main system components is suggested. Instead of the functional architecture, we suggest using problem diagrams as a foundation for HARA because they show not only the system but also the environment. For the problem diagrams it is essentially important that all domains connecting the machine to the real world are actually modelled in the problem diagrams even if they only transmit data without processing it

(e.g. in case of a chain of sensors or a chain of actuators). The reason is that failure events could also occur here.

Step 2: Make assumptions explicit. For each problem diagram from Step 1, the assumptions need to be made explicit as shown in Figure 2. Making the assumptions explicit has the benefit that we can assess the probability for their invalidity and consider them in the FTA (Step 4).

Step 3: Perform HARA. During this step, the HARA is mainly performed in the way described by Beckers et al. in [Be13] except that we identify hazardous events at the controlled domains (i.e. situations in which d variable is not achieved) and not at the actuators. Input to this step are the problem diagrams from Step 1. For each requirement in a problem diagram, possible faults are identified. For each fault/requirement combination, all situations that could lead to a potential hazard are identified. Each hazard is classified according to its severity, exposure, and controllability in order to assess the required level of risk reduction (ASIL). Finally, safety goals are defined for preventing or mitigating the hazards. In order to evaluate whether the residual risk of safety goal violations is sufficiently low, the ISO 26262 Standard requires for a system with ASIL C or ASIL D to determine the PMHF (cf. [IS11]). PMHF stands for 'Probabilistic Metric for random Hardware Failures' and is used to evaluate the probability of violation of the considered safety goal using, for example, quantified FTA and to compare the result of this quantification with a target value. This is what we do in Step 4.

Step 4: Perform FTA. For each hazardous event, we identify possible causes and document the failure events and faults in a fault tree based on the corresponding problem diagram from Step 1. The hazardous event occurs at the controlled domain in the problem diagram. To identify faults, we traverse the problem diagram, starting at this controlled domain and stopping at the monitored domain(s). Note that it might be necessary to traverse the same problem diagram for several hazardous events. We document the identified failure events and faults in a fault tree. A fault tree (FT) is a directed acyclic graph with two types of nodes: events and gates [RS15]. The notation is shown in Figure 6. The event at the top of a fault tree is the hazardous event being analysed. Events that are not further decomposed are basic events, i.e. either faults or situations. Gates represent how failures propagate, i.e. how failures can combine to cause a hazardous event. There are two types of gates. An AND gate means that the output event occurs if all of the input events occur. An OR gate means that the output event occurs if any of the input events occurs.

The fault tree allows one to calculate the probabilities of failure occurrence. We calculate the probability that a failure occurs in one hour of operation time, assuming that this is the time of a typical drive cycle (see [IS11], Part 5, Section 9.4.2.3, Note 2). We also assume a linear failure rate distribution. Therefore, our values for the probabilities in the fault tree and for the failure rates are the same. Failure rates are annotated at the leaves of the tree (based on expert knowledge) and are calculated bottom up to the root according to FTA rules for AND and OR gates (cf. [RS15]). As stated above, the resulting value (i.e. the actually achieved failure rate) must be compared to a target value. One possibility to obtain PMHF target values is given by means of Table 6 in Part 5 of the ISO 26262 Standard [IS11]. It requires for ASIL D a failure rate smaller than $10^{-8}h^{-1}$ and for ASIL C a failure

rate smaller than $10^{-7}h^{-1}$. Note that these PMHF target values only cover the hardware parts of the system. The ISO 26262 Standard does not consider faults that occur in the environment. Since we think that the environment should be considered, we would like the environment failure rate to have the same value as the PMHF target value. This means that the overall maximum failure rate for violating the safety goal would be $2 \cdot 10^{-8}h^{-1}$ for ASIL D and $2 \cdot 10^{-7}h^{-1}$ for ASIL C.

To use the same fault tree for calculating the actually achieved failure rate (in order to show compliance to ISO 26262), we set the failure rates of the environment events to 1, if they are connected with an AND gate, and to 0, if they are connected with an OR gate in order to ignore them. The actually achieved failure rate is compared to the target value (i.e. $2 \cdot 10^{-8}h^{-1}$ for ASIL D or $2 \cdot 10^{-7}h^{-1}$ for ASIL C). Only if the calculated value for the tree is higher than the target value, Step 5 needs to be performed.

Step 5: Perform risk reduction. During this step, modifications are made to the problem diagrams from Step 1, for example, a sensor is added to increase reliability of monitored information, or a different procedure is considered (e.g. calculation or estimation). By means of such modifications, the assumptions of type *DH-MD, Exp-SE, Exp-AC, Exp-CD* may change as well, i.e. they are strengthened, weakened, or abandoned. Based on these modified problem diagrams, Step 4 is performed again. This is done until the target failure rate is achieved.

5 Comparison of Traditional FTA and Our Method

To illustrate that performing safety analysis based on the Six-Variable Model (as we do in our method) results in a more realistic and complete safety analysis, we compare two types of fault trees: Fault Tree 1, which focusses on the system (result of traditional FTA) and Fault Tree 2, which considers also the environment with the *r* and *d* variables. As an example we consider the ACC system described in [Ro03]. This is a simple version of an ACC, which mainly supports cruise control. It maintains the desired speed entered by the driver. If it detects vehicles ahead, it adapts the speed of the ACC vehicle accordingly.

Performing traditional FTA. The system design for the ACC is given as a SysML internal block diagram in Figure 5. The ACC uses data from a long range radar sensor (LRR) and ESP sensors to identify vehicles ahead on the same lane. For adapting the speed of the ACC vehicle, it uses the ESP system and the engine management system.

One hazardous event that may occur at the ESP (actuator) is *unintended braking*. We perform traditional FTA, i.e. we start analysis at the ESP and stop at the sensors (LRR and ESP sensors) focussing on the identification of system failures. The result, Fault Tree 1, is given in Figure 6. The hazardous event *unintended braking* may be caused by two events: the ESP system performs deceleration on its own (F ESP DEC) or it performs deceleration due to the input it received (FA ESP DEC). The first event represents a fault in the ESP system and is not further analysed. The second event may have different causes and is therefore further analysed. First, the CAN bus may have requested erroneously to decelerate, i.e. there was a CAN fault, (F CAN DEC1) or CAN requests for deceleration due

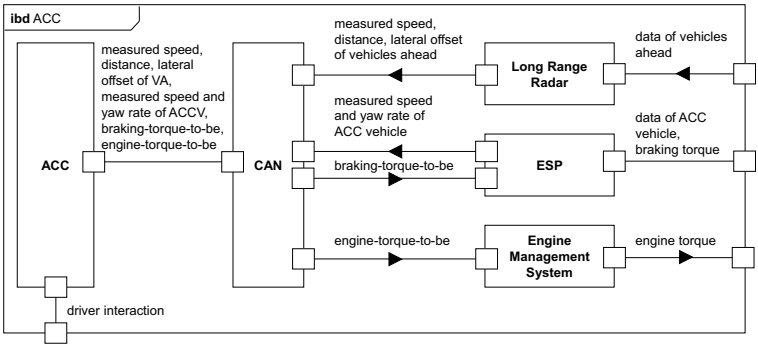


Fig. 5: ACC system design

to the input it receives from ACC (FA CAN DEC1). The latter event has again two possible causes: an ACC fault (F ACC DEC) or ACC requests deceleration due to the input it receives from CAN (FA ACC DEC). The latter event has again two possible causes: a CAN fault (F CAN DEC2) or CAN requests deceleration due to the input it receives from sensors. The latter event has again two possible causes: the ESP sensors send wrong information (wrong speed and/or wrong yaw rate) due to a fault in the ESP sensors or the long range radar sends wrong information (wrong speed and/or distance and/or lateral offset) due to a fault in the long range radar sensor.

Once the fault tree is created, failure rates must be assigned to the leaves of the fault tree and the failure rates must be calculated bottom up. Due to the experience one of the authors had in the automotive domain, the values were assigned by him in our example. We used the tool Reliability Workbench 11[5] to calculate the failure rates bottom up. To check the values calculated by the tool, we calculated them also manually. Overall, we achieve a failure rate of $2.04 \cdot 10^{-6} h^{-1}$.

Application of our method. We create first a problem diagram based on the Six-Variable Model. This is given in Figure 7. In contrast to Figure 5, Figure 7 shows the environment and the r and d variables. Referenced variables are speed, distance, and lane of vehicles ahead as well as the lane of the ACC vehicle. Monitored variables are speed, distance, and relative position of vehicles ahead as well as the course of the ACC vehicle. The desired variable is the speed adaptation of the ACC vehicle. The requirement in Figure 7 is 'Maintain desired speed and keep safety distance to vehicles ahead'. This requirement can be decomposed into the assumptions (expectations and domain hypotheses) and software requirements given below the problem diagram in Figure 7.

Instead of the hazardous event *unintended braking*, we consider the hazardous event that occurs at the controlled domain: *unintended speed reduction*. Based on the problem diagram in Figure 7, we create the fault tree. It is given in Figure 8. Compared to Fault Tree 1, Fault Tree 2 starts with a different root node due to the different hazardous event that is considered. This event has two possible causes. First, the ACC vehicle may decelerate

[5] http://www.isograph.com/software/reliability-workbench/

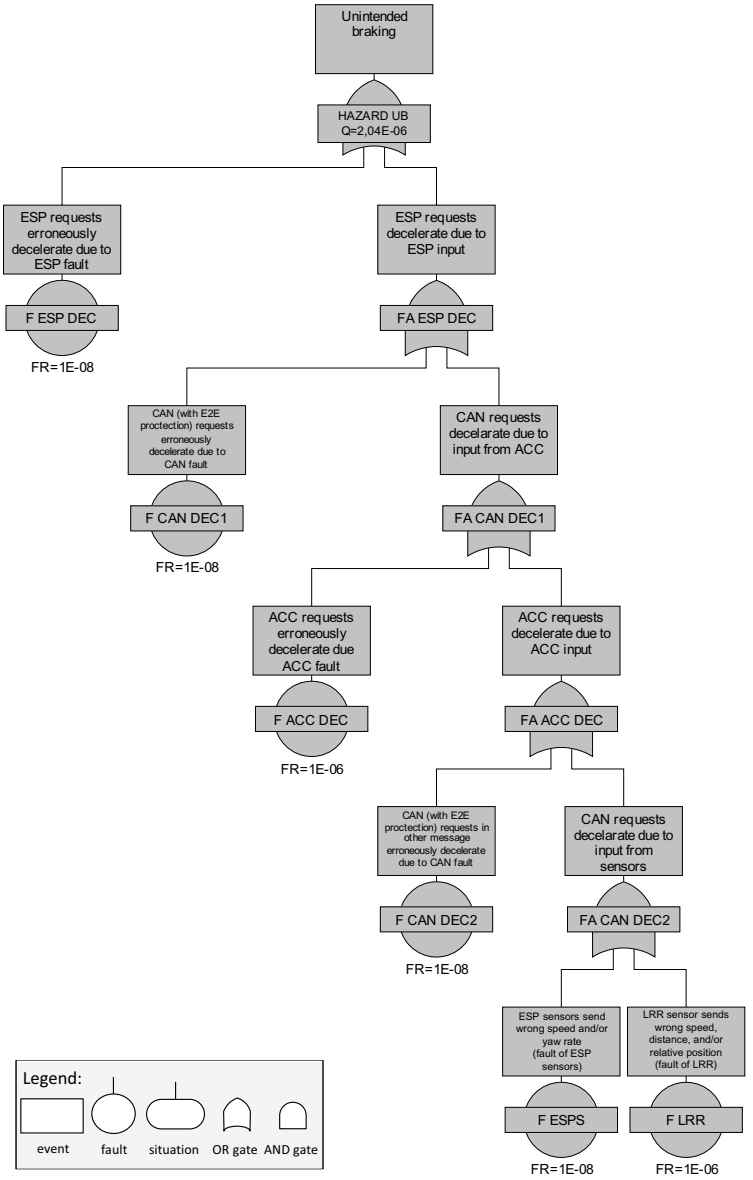
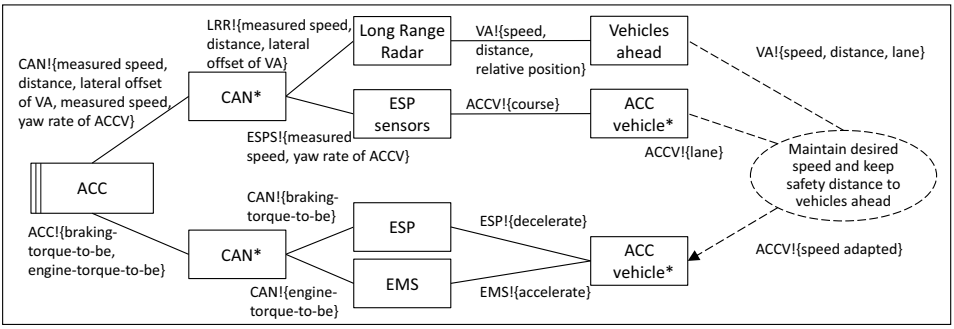


Fig. 6: Fault Tree 1 – Result of Traditional FTA



DH1-ACCV:	The course of the ACC vehicle reflects its lane.
DH1-VA:	Speed, distance, and relative position of a vehicle ahead reflect its speed, distance, and lane.
Exp1-LRR:	Measured speed, distance, and lateral offset of a vehicle ahead reflect its actual speed, distance, and relative position.
Exp1-ESPS:	Measured speed and yaw rate of the ACC vehicle reflect its actual course.
Exp1-CAN:	Speed, distance, and lateral offset of vehicles ahead as well as speed and yaw rate of ACC vehicle transmitted by CAN correspond to the speed, distance, and lateral offset measured by the long range radar and the speed and yaw rate measured by the ESP sensors.
SOF1-ACC:	Determine course of the ACC vehicle based on the measured speed and yaw rate of the ACC vehicle. Calculate relative position of vehicle ahead based on determined course and lateral offset of vehicle ahead. Calculate engine- or braking-torque-to-be.
Exp2-CAN:	Transmitted engine- and braking-torque-to-be correspond to engineand braking-torque-to-be provided by ACC.
Exp1-ESP:	Decelerate instruction reflects braking-torque-to-be.
Exp1-EMS:	Accelerate instruction reflects engine-torque-to-be.
Exp1-ACCV:	Speed of ACC vehicle is adapted according to decelerate instruction.
Exp2-ACCV:	Speed of ACC vehicle is adapted according to accelerate instruction.

Fig. 7: Problem diagram for the ACC and environmental assumptions

although its input does not require deceleration (F ACCV1). This event covers exactly the situation when the c variable does not result in the d variable, i.e. negation of an assumption of type Exp-CD, here Exp1-ACCV. The ACC vehicle shall not decelerate but it does. The controlled domain ACC vehicle does not behave as expected. This situation may be seldom but nevertheless we consider it as a possible cause of the hazardous event. In Fault Tree 1, this event was not considered. The other possible cause of the root node is that the ACC vehicle decelerates due to the input it receives (FA ACCV DEC). This event is then further decomposed. The middle of the tree resembles Fault Tree 1. Yet, at the bottom of the tree, we find again differences, since we consider the possibility that the domain hypotheses DH1-ACCV and DH1-VA are wrong/invalid. There may be situations in which the course estimated by the ESP sensors does not correspond to the actual lane of the ACC vehicle (F ACCV2). In a curve, for example, the estimation of the own lane and the lane of vehicles ahead may be difficult. For example, if both cars are driving close to the road marking, the ACC may assume (based on the determined course of the ACC vehicle and the calculated relative position of the vehicle ahead) that they are driving on the same lane although they are actually on two different lanes. So, there may be situations in which the relative position of vehicles ahead does not reflect their lane. This is covered by event F LRR. Fault Tree 2 results in an overall failure rate of: $2.02 \cdot 10^5 h^{-1}$.

Comparison of the results. The comparison of the two fault trees reveals that, of course, the consideration of assumptions in Fault Tree 2 results in a higher value for the overall failure rate since more risks are considered than in Fault Tree 1. Yet, they are the assumptions that we actually make. Therefore, the overall failure rate of Fault Tree 2 is more realistic.

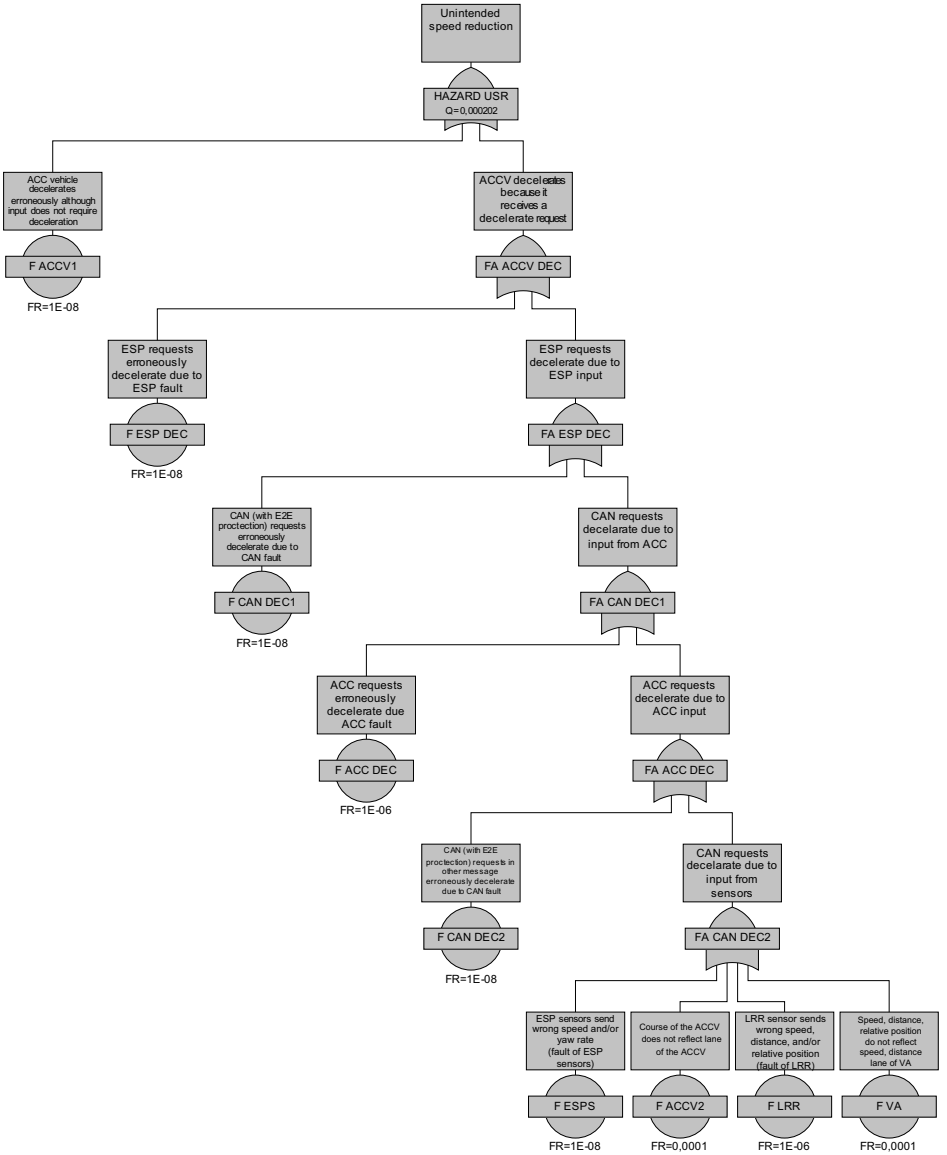


Fig. 8: Fault Tree 2 – Result of Our Method

Furthermore, it is more complete since we do not neglect failure events that may occur due to wrong/invalid assumptions (Exp1-ACCV, DH1-ACCV, DH1-VA).

6 Related Work

A method that is related to our work is the KAOS method described by van Lamsweerde [La09]. He assumes Jackson's model of the world and the machine and suggests a goal-oriented method. He also calls for documenting environmental assumptions (expectations and domain hypotheses) and for checking whether they are too strong and too risky. However, he focuses on the classical four variables and thus does not provide support in making the 'real' environmental assumptions explicit.

Another work that is closely related to ours is the work of Tun et al. [Tu15]. They suggest a method for identifying and classifying environmental assumptions whose violation is known from experience to have prevented the requirements from being satisfied. The idea is to reuse this knowledge of past failures to prevent failures in the future. The approach is quite interesting because it also considers possible mismatches between assumptions the software makes about the environment and the actual environmental reality. Yet, the method is reactive, i.e. failures must have been occurred in the past to be considered in the future. In contrast to that, our method is proactive. We provide support in making the environmental assumptions explicit and in considering the possibility for their violation already during safety analysis, i.e. before failures actually occur. The failures can then be prevented, for example, by changing the system design (e.g. adding sensors) and thus weakening assumptions.

Another related work is HAZOP (Hazard and Operability Studies) [IE01]. HAZOP is a technique for examining a system in order to identify potential hazards in the system as well as potential operability problems with the system. A key characteristic of HAZOP is that it provides a core set of guide words which are intended to stimulate the imagination of the team (performing the HAZOP study) in a systematic way to identify hazards and operability problems. We use these HAZOP guide words in our method (Step 3: Perform HARA) as well.

Finally, STPA (Systems-Theoretic Process Analysis) [Le11] is a further important related work. While fault tree analysis is based on reliability theory, STPA is based on systems thinking. This means, the cause of an accident/hazardous event is not understood as a series of failure events but as the result of a lack of constraints imposed on the development, design, and operation of the system. Safety is therefore viewed as a control problem, i.e. hazardous events occur when component failures, external disturbances, and/or dysfunctional interactions among system components are not adequately handled. Preventing hazardous events requires thus designing a control structure that enforces the necessary constraints on system development and operation. STPA is an interesting method and, since it is based on systems thinking, is closer related to our Six-Variable Model than fault tree analysis. Yet, fault tree analysis is a well-established technique that is frequently used in industry, especially in the automotive domain. Therefore, our aim was to extend fault tree analysis. Yet, we consider extending STPA with our Six-Variable Model in future work.

7 Conclusion and Future Work

In this paper, we presented an ISO-26262-compliant method for performing hazard analysis and fault tree analysis. The main contribution of our method is that it supports the identification of failure events beyond the system border. More precisely, it supports developers in considering the risk associated with possibly too strong environmental assumptions in the hazard analysis as well as the fault tree analysis. We think that it is important to consider such risks because there are numerous real accidents that were caused by such assumptions turning out to be invalid in certain operational situations while the system behaved as specified. Too strong assumptions can be weakened or abandoned by changing the system design. Therefore, they should be detected early on in the development process. Our method supports that. In future work, we plan to extend STPA (Systems-Theoretic Process Analysis) with our Six-Variable Model and to compare the resulting method with the one described in this paper with regard to the applicability, usability, etc.

References

[Be13] Beckers, K.; Heisel, M.; Frese, T.; Hatebur, D.: A Structured and Model-based Hazard Analysis and Risk Assessment Method for Automotive Systems. In: Proc. of ISSRE 2013. pp. 238–247, 2013.

[IE01] IEC 61882 Hazard and Operability Studies – Application Guide, 2001.

[IS11] ISO 26262 Road Vehicles – Functional Safety, 2011.

[Ja01] Jackson, M.: Problem Frames – Analysing and Structuring Software Development Problems. Addison-Wesley, 2001.

[La09] Lamsweerde, A. Van: Requirements Engineering – From System Goals to UML Models to Software Specifications. John Wiley and Sons, 2009.

[Le11] Leveson, N.: Engineering a Safer World – Systems Thinking Applied to Safety. The MIT Press, 2011.

[PM95] Parnas, D. L.; Madey, J.: Functional Documents for Computer Systems. Science of Computer Programming, 25(1):41–61, 1995.

[Ro03] Robert Bosch GmbH: ACC Adaptive Cruise Control - The Bosch Yellow Jackets. Edition 2003 edition, 2003.

[RS15] Ruijters, E.; Stoelinga, M.: Fault Tree Analysis: A Survey of the State-of-the-Art in Modeling, Analysis and Tools. Computer Science Review, 15(1):29–62, 2015.

[Tu15] Tun, T.; Lutz, R.; Nakayama, B.; Yu, Y.; Mathur, D.; Nuseibeh, B.: The Role of Environmental Assumptions in Failures of DNA Nanosystems. In: Proc. of COUFLESS 2015. pp. 27–33, 2015.

[UBMH16] Ulfat-Bunyadi, N.; Meis, R.; Heisel, M.: The Six-Variable Model – Context Modelling Enabling Systematic Reuse of Control Software. In: Proc. of ICSOFT-PT 2016. pp. 15–26, 2016.

[ZJ97] Zave, P.; Jackson, M.: Four Dark Corners of Requirements Engineering. ACM Transactions on Software Engineering and Methodology, 6(1):1–30, 1997.

Using STPA in Compliance with ISO 26262 for Developing a Safe Architecture for Fully Automated Vehicles

Asim Abdulkhaleq[1], Stefan Wagner[2], Daniel Lammering [3], Hagen Boehmert[4] and Pierre Blueher [5]

Abstract: Safety has become of paramount importance in the development lifecycle of the modern automobile systems. However, the current automotive safety standard ISO 26262 does not specify clearly the methods for safety analysis. Different methods are recommended for this purpose. FTA (Fault Tree Analysis) and FMEA (Failure Mode and Effects Analysis) are used in the most recent ISO 26262 applications to identify component failures, errors and faults that lead to specific hazards (in the presence of faults). However, these methods are based on reliability theory, and they are not adequate to address new hazards caused by dysfunctional component interactions, software failure or human error. A holistic approach was developed called STPA (Systems-Theoretic Process Analysis) which addresses more types of hazards and treats safety as a dynamic control problem rather than an individual component failure. STPA also addresses types of hazardous causes in the absence of failure. Accordingly, there is a need for investigating hazard analysis techniques like STPA. In this paper, we present a concept on how to use STPA to extend the safety scope of ISO 26262 and support the Hazard Analysis and Risk Assessments (HARA) process. We applied the proposed concept to a current project of a fully automated vehicle at Continental. As a result, we identified 24 system-level accidents, 176 hazards, 27 unsafe control actions, and 129 unsafe scenarios. We conclude that STPA is an effective and efficient approach to derive detailed safety constraints. STPA can support the functional safety engineers to evaluate the architectural design of fully automated vehicles and build the functional safety concept.

Keywords: STAMP/STPA Safety Analysis, ISO 26262, Functional Safety, Autonomous Vehicles.

1 Introduction

Nowadays, innovations in software and technology lead to increasingly complex automotive systems such as self-parking vehicles, the use of smartphones to park vehicles and, more recently, fully automated driving vehicles. As a new technology, the fully automated driving vehicles may bring a new safety risk and threats to our society which have to be controlled during their development. Hence, the safety analysis becomes a great challenge in the development of safety-critical systems. In the past, failures of the automotive systems beyond separate component malfunction like interface problems led to safety issues. The automotive industry started to pay attention to the functional safety of vehicle electronic control systems and to introduce new standards to address the growing complexity of its systems. The safety standard ISO 26262 [IS11] "Road vehicles – Functional

[1] Institute of Software Technology, University of Stuttgart, Asim.Abdulkhaleq@informatik.uni-stuttgart.de
[2] Institute of Software Technology, University of Stuttgart, Stefan.Wagner@informatik.uni-stuttgart.de
[3] Continental, Regensburg, Germany, Daniel.Lammering@continental-corporation.com
[4] Continental, Frankfurt am Main, Germany, Hagen.Boehmert@continental-corporation.com
[5] Continental, Frankfurt am Main, Germany, Pierre.Blueher@continental-corporation.com

safety" is an international risk based safety standard to describe state-of-the-art for the development of safety-relevant vehicle functions and addresses possible hazards caused by malfunctioning behaviour of electrical/electronic systems.

1.1 Problem Statement: The ISO 26262 is a document intended to achieve functional safety regarding an E/E component. It specifically addresses hazards resulting from the presence of failures and malfunctions emanating from hardware (HW) and software (SW). These may be introduced by random (HW-related) failures/malfunctions and/or systematic SW failures. The intention is to give arguments for functional safety if "best practice" guidelines have been followed during concept and development phases. The Hazard Analysis and Risk Assessment (HARA) [IS11] defines possible hazards; deductive and inductive analyses look for E/E faults and failures that lead to these hazards. In contrast to the general assumption, following this guideline does not mean that the product is "safe". It is the authors' belief that a "safe" product does not only result from the absence of hazards during the presence of malfunctions, but also the absence of hazards in the absence of malfunctions. That bears the question **how these hazards can be identified and what their cause might be**.

1.2 Research Objectives: This research work answers that question by using the STPA method as an approach to identify the potential hazards of fully automated vehicles in compliance with ISO 26262 at an early concept phase and provide safety constraints on how the risk for an accident can be mitigated by avoiding those hazards.

1.3 Contribution: We provide guidance on how to use STPA in compliance with ISO 26262. We apply STPA to the existing architecture design of a fully automated driving vehicle to develop a safety concept to enhance the architecture design.

1.4 Context: This work was conducted in the form of a cooperation between Continental, which is a German automotive manufacturing company, and the University of Stuttgart, during the development process of a fully automated driving vehicle project.

1.5 Terminology:

Functional Safety is "Absence of unreasonable risk due to hazards caused by malfunctioning behavior of Electrical/Electronic systems"[IS11].

Operational Safety (Roadworthiness): is "a property or ability of any kind of automobile to be in a suitable operating condition or meeting acceptable standards for safe driving and transport of people, baggage or cargo in roads or streets" [Go14].

Item: "is a system or array of systems or a function to which ISO 26262 is applied" [IS11].

2 Background

2.1 Hazard Analysis Approach: STPA

STPA (Systems-Theoretic Processes Analysis) [Le11] was developed by Leveson in 2004 based on the STAMP (Systems-Theoretic Accident Model and Processes) causality accident model for identifying system hazards and safety-related constraints necessary to ensure acceptable risk in complex systems. STPA helps to identify causal factors and unsafe scenarios in which the safety constraints can be violated. STPA results in identifying a larger set of causes, many of them not involving failures or unreliability, while traditional techniques were designed to prevent hazards due to component failure accidents (caused by one or more components that fail). The main steps of STPA are divided into three sub-steps: (1) Establish the fundamentals of the analysis (e.g. system-level accidents and the associated hazards) and draw the control structure diagram of the system. (2) Use the control structure diagram to identify the potentially unsafe control actions. (3) Determine how each potentially unsafe control action (accident causes) could occur by identifying the process model and its variables for each controller and analysing each path in the control structure diagram.

STPA has been successfully applied and extended in different domains such as STPA for automotive systems [AW13], STPA for cybersecurity [YL14] and STPA for software safety [AWL15].

2.2 ISO26262 Safety Standard

ISO 26262 (Road vehicles functional safety) [IS11] is an international functional safety standard, which provides guidance, recommendation and argumentation for a safety-driven product development in the automotive area. Safety classification and suggestions for specific safety development processes may aid to stipulate functional safety for each new product as State-of-the-Art.

ISO 26262 is structured into 10 parts and describes the safety activities in 7 parts (3–9). Part 3 specifics the concept phase (as shown in Fig. 2) which starts with defining the item (e.g. system, array or function) and performing the hazard and risk analysis for the item. The results are the safety goals for all hazards which are derived and classified with an ASIL (Automotive Safety Integrity Level) rating which is a risk classification scheme defined by the ISO26262 standard. Part 4 specifies the requirements for product development at the system level for automotive applications. Parts 5 and 6 specify the requirements for product development at the hardware level and software level for automotive applications. Part 7 specifies the requirements for production, operation, service and decommissioning.Part 8 specifies the supporting processes (e.g. hardware and software tool qualification). Part 9 specifies the automotive safety integrity level (ASIL)-oriented and safety-oriented analysis.

3 Related Work

Hommes [Ho12, Ho15] highlighted the benefits of applying STPA in the automotive domain and using STPA as a hazard analysis technique in the ISO 26262 lifecycle, especially in the concept phase (ISO26262 part 3). Hommes also provided an assessment review of the ISO 26262 standard's ability to address the challenges in ensuring the safety of complex software intensive E/E systems. Hommes mentioned that there is a lack in guidance on hazard identification and elimination in the concept phase, which makes the ISO 26262 standard not sufficient to provide safety assurance. That gives us a strong motivation to investigate the use of STPA in compliance with the ISO 26262 standard to gain a deeper understanding about the benefits and limitations of using STPA as a hazard analysis approach to support the HARA process in ISO 26262.

Recently, Mallya et al. [Ma16] analysed how STPA can be used in an ISO 26262 compliant process. They mapped every relevant activity and artifact required or recommended by the HARA process which can be satisfied by applying STPA. They emphasized that the key difference between STPA and HARA is the risk assessment process. However, both STPA and HARA have different base assumptions for identifying hazards. Based on this work, We explored how STPA can extend the safety scope of ISO 26262 by considering different factors that cause inadequate controls during the operational time of a vehicle (e.g. human, environment). We also proposed a concept on how to use STPA to support HARA activities in ISO 26262 instead of mapping STPA activities onto the HARA process.

We have applied STPA to a well-known example of a safety-critical systems in the automotive domain [AW13]: the Adaptive Cruise Control system (ACC). This case study was performed based on an existing case study with MAN Truck & Bus AG [Wa10] in which the authors conducted an exploratory case study applying safety cases for the ACC system. We compared the results of STAMP/STPA with the safety cases on the same system. In [AW14], we proposed a safety verification methodology based on STPA safety analysis. We applied STPA to vehicle cruise control software to identify the software safety requirements at the system level and verify these safety requirements at the design level. Recently, we proposed a safety engineering approach for software-intensive systems based on STPA [AWL15], called *STPA SwISs* which combines the activities of safety engineering and software engineering. Thus, in turn, it shall help to reduce the associated software risks to a low level. This approach can be applied in compliance with the ISO 26262 part 6 at the software level.

4 The Concept of Using STPA in the ISO 26262 Lifecycle

The main goal of STPA is to identify inadequate control scenarios which can lead to accidents and develop detailed safety constraints to eliminate and control these unsafe scenarios. The main starting point of STPA is to identify potential accidents and hazards at the system level and draw hierarchical safety control structure of the system. The main activities of the concept phase in ISO 26262 are defining an item, identifying the hazardous events that need to be eliminated or controlled and developing the safety concept at the

Tab. 1: The terms used in STPA and the terms used in ISO 26262 part 3: concept phase

STPA Terminologies [Le11]	ISO 26262 Terminologies [IS11]
Accident: Accident (Loss) results from inadequate enforcement of the behavioural safety constraints on the process.	**No corresponding term**
No corresponding term	**Harm:** is a physical injury or damage to the health of persons.
Hazard is a system state or set of conditions that, together with a particular set of worst case environmental conditions, will lead to an accident.	**Hazard** is a potential source of harm caused by malfunctioning behaviour of the item.
No Corresponding	**Item** is a system or array of systems to implement a function at the vehicle level, to which ISO 26262 is applied.
Unsafe Control Actions are the hazardous scenarios which might occur in the system due to a provided or not provided control action when it was required.	**No corresponding term**
No corresponding term	**Malfunctioning Behaviour**: is a failure (termination of the ability of an element to perform a function as required) or unintended behavior of an item with respect to its design intent
Safety Constraints are the safeguards which prevent the system from leading to losses (accidents)	**Functional Safety Requirements**: are specifications of implementation-independent safety behaviour, or implementation-independent safety measures, including its safety-related attributes.
Causal Factors are the accident scenarios that explain how unsafe control actions might occur and how safe control actions might not be followed or executed.	**No corresponding term**
No corresponding term	**Hazardous Events** are combinations of a hazard and an operational situation.
Corresponding safety Constraints are top-level safety constraints which are derived based on the unsafe control actions	**Safety Goals** are top-level safety requirements as a results of the hazard analysis and risk assessments
No Corresponding term	**Functional Safety Concept** consists of functional requirements and preliminary architectural assumptions.
Process Model is a model required to determine the environmental & system variables (process model variables) that affect the safety of the control actions.	(Partially) **Operation Situation** is a scenario that can occur during a vehicle's life. **Operating Mode** is perceivable functional state of an item or element. **Safe State** is operating mode of an item without an unreasonable level of risk (e.g. switched-off mode, intended operating mode, degraded operating mode)
No Corresponding term	**ASIL** is one of four levels to specify the item's or element's necessary requirements of ISO 26262 and safety measures to apply for avoiding an unreasonable residual risk, with D repre-

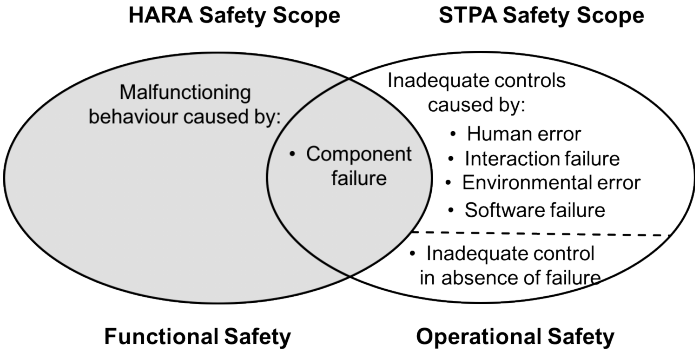


Fig. 1: The safety scope of STPA and HARA in ISO 26262

system level. The Hazard Analysis and Risk Assessment (HARA) process in ISO 26262 consists of the following activities [IS11]: (1) Situation Analysis and Hazard identification, (2) Hazard classification, (3) Hazard determination, and (4) Safety goal determination. In this paper, we used STPA as a hazard analysis technique to support HARA by defining an item and identifying the hazards and unsafe scenarios of the item at the system level. The term ”item” is used here to refer to the system.

Figure 1 shows the safety scope of both STPA and HARA. STPA focuses on identifying the potential inadequate controls that could lead to the hazards. The inadequate control can be caused by human error, interaction failure, environmental, software failure. STPA also focuses on identifying the inadequate controls in absence of the individual component failures (e.g. dysfunctional interactions or unhandled conditions). The safety scope of the HARA in ISO 26262 is to identify the possible hazards caused by the malfunctioning behaviour of electronic and electrical systems (individual components). To use STPA in the ISO 26262 lifecycle, we first define the important terms of STPA and ISO 26262 (shown in Table 1). Based on our expertise in STPA and ISO 26262, we map the terms of STPA and ISO 26262 which have the same meaning. In the following, we summarise the main steps to use STPA in compliance with ISO 26262 in the part 3 concept phase (shown in Fig. 2):

1. Apply STPA Step 0 (Fundamentals Analysis):
   - 1.1. Identify Accidents and system-level hazards.
   - 1.2. Identify the high-level system safety constraints.
   - 1.3. Draw the control structure diagram of the system
2. Use the results of STPA Step 0 to define an item and item information needed (e.g. purpose, content of item, functional requirements etc.). The control structure diagram in STPA Step 0 shows the main components which form a system under analysis. This diagram contains information to help the functional safety engineer to define an item and its boundaries.

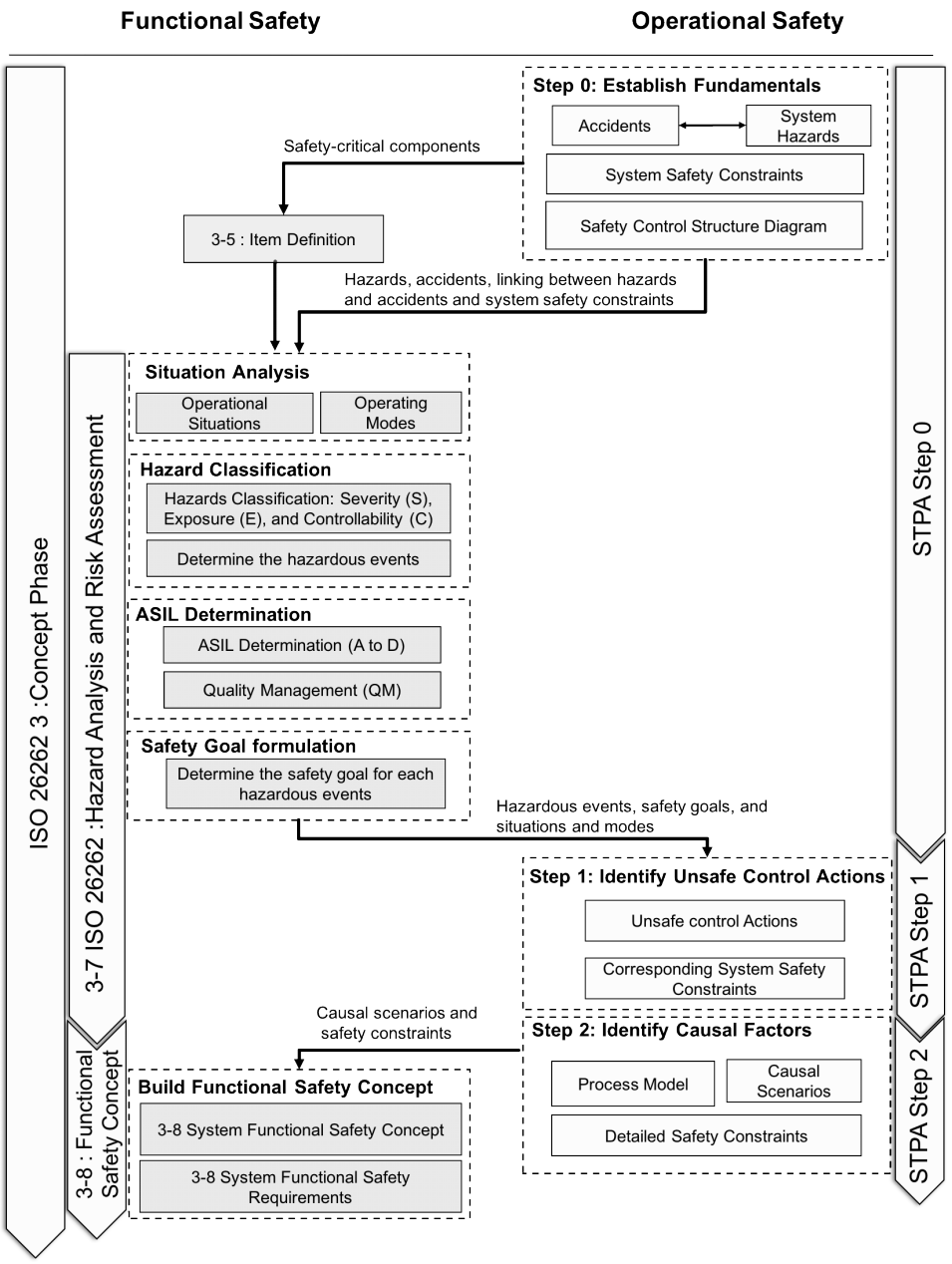


Fig. 2: Integration of STPA into the ISO26262- part 3 concept phase

3. Use the list of hazards, accident, the high-level system safety constraints identified in STPA Step 0 as an input to the HARA approach.

4 Apply the HARA approach:

4.1 Determine the operational situations and operating modes in which an item's malfunctioning behaviour may lead to potential hazards.

4.2 Classify the hazards identified in Step 0 based on the estimation of three factors: Severity (S), Probability of Exposure (E) and Controllability (C)

a) Identity the hazardous events by considering the hazards in different situations.

4.3 Determine ASIL (Automotive Safety Integrity Levels) for each hazardous event by using four ASILs: A (the lowest safety integrity level) to D (the highest one). If the hazardous event is not unreasonable, we refer it as QM (Quality Management).

4.4 Formulate the safety goal for each hazardous event.

5. Use the hazardous events, safety goals, situations and modes as input to the STPA Step 1.

6. Apply STPA Step 1 to identify the unsafe control actions of an item

7. Apply STPA Step 2 to identify the causal factors and unsafe scenarios of each unsafe control action identified in STPA Step 1.

8. Use the results of STPA Step 1 & 2 to develop the system functional safety concept and safety requirements at this level.

5 Application Example: Fully Automated Driving Vehicle

Study Object: Semi-automatic and fully automated driving requires compliance with essential system features like reliability, availability, security and safety. A fully automated driving system (SAE Level 5) [SA16] involves the act of navigating the car without any input from the human driver through the use of sensing the environment, performing and calculating a desired driving path (trajectory) and sending the desired controls to the actuators (as shown in Fig. 3). Therefore, the required components for a fully automated driving system can be classified in three main groups: 1) **Sense**: Several sensors are necessary to gather information of the environment, perception of the vehicle state and traffic participants. For example, wheel speed sensors, cameras, short and long range radars and even lidar technology are used to get a fine-grained environment model; 2) **Plan:** The planning component consist of multiple levels of decision making. The driving strategy plans upcoming maneuvers and is a core element of the car's calculated behaviour. It serves as an input for the trajectory planning sub-module. The function of the trajectory planning is processing a safe vehicle trajectory with respect to the surroundings (collision avoidance) and to the given maneuver. The motion controller calculates the motion requests from the desired trajectory. 3) **Act:** In the Act group all needed actuators, e.g. brake system, steering system and engine control, for longitudinal and lateral movement can be summarized. The output of the motion controller provides the torque requests for the actuators.

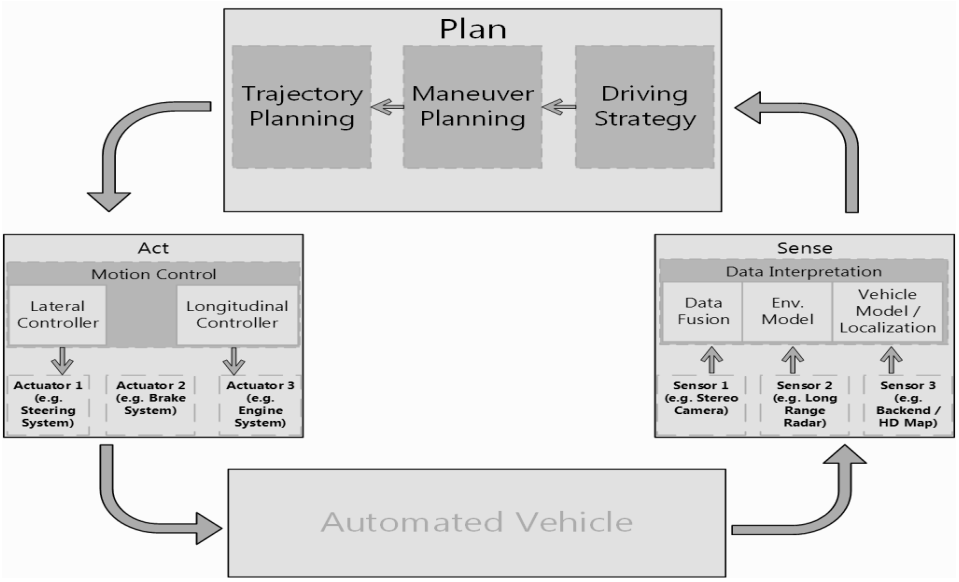


Fig. 3: Functional Architecture of Fully Automated Driving Vehicles

Tab. 2: Examples of the system level accidents

ID	Accident
1	AD vehicle lost the steering and collided into an object moving in front on a highway.
2	AD vehicle lost the steering/braking suddenly while the vehicle moving up in the hill and made an accident.
3	AD vehicle made a collision due to loss the communication signals with Back-end.
4	AD vehicle collided into an object or vehicle due to a wrong driving strategy

Apply the STPA Step 0: Fundamentals Analysis First, we do the first part of STPA Step 0 (identifying the system-level accidents). As a result, we identified 24 system-level accidents which the fully automated driving system can lead or contribute to. Table 2 shows examples of the system level accidents. For example, *AC.1: The fully automated vehicle collided into an object moving in front on a highway.* Second, we do part 2 of Step 0 (identifying hazards). As a result, we identified 176 hazards which can lead to these accidents. For example, a hazard can lead to accident *AC.1*, *HA.1: The fully automated vehicle lost steering control because it received wrong ego longitudinal torque.* Third, we do part 3 of Step 0 (identifying the system-level safety constraints). An example for a high-level system safety constraint is *SC.1: The fully automated driving vehicle must receive correct data all the time while driving on a road.*

Tab. 3: Examples of the system level hazards

ID	Hazards
1	The AD vehicle lost steering control because it received wrong ego longitudinal torque.
2	The AD vehicle does not detect a moving obstacles in the front.
3	The AD vehicle moves with no data of prediction of situation and scenario for traffic participants.
4	The AD vehicle receives wrong environmental model data.

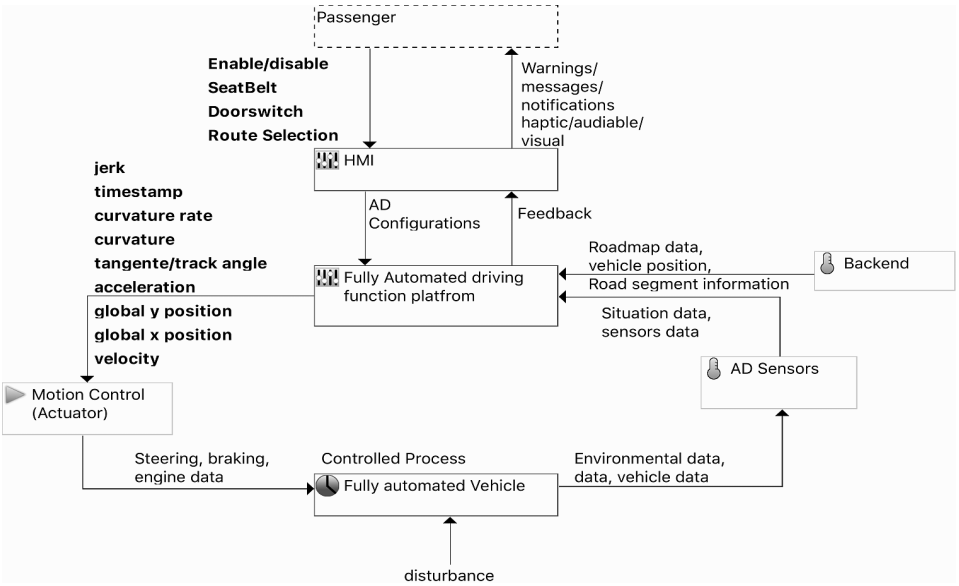


Fig. 4: The safety control structure diagram of fully automated driving system

Fourth, we do the part 4 of Step 0 (drawing the high-level safety control structure diagram). Figure 4) shows the control structure diagram of the fully automated driving system at the architectural design level. The control structure diagram shows the main components which interact with the fully automated driving system in the vehicle. We used the results of the STPA Step 0 to define the item. For example, the fully automated driving function platform is an item in which the ISO 26262 can be applied. The control structure diagram shows the boundary of the fully automated driving function platform and its interfaces. The purpose of the fully automated driving function platform is to control the autonomous vehicle by issuing the control actions to the motion control and receiving the feedback from the different sensors.

Using the STPA Step 0 results: We also used the results of the STPA Step 0 (list of accidents, list of hazards) as an input to the HARA approach. First, we determine the operational situations and operating modes. For example, the operation situation from the

accident AC.1 can be determined as *OS1: crashing on a highway*. The operating mode is *OM.1: driving*. We classified each hazard identified in the STPA Step 0 with two factors (severity and exposure). For example, we estimated the severity for the hazard HA1 as S3 (Life-threatening injuries or fatal injuries), the probability of exposure of the operational situation (on highway) as E3 (Medium probability). Then, we identified an hazardous event HE.1 from the hazard HA.1 and the accident AC.1 as *HE.1: The fully automated vehicle lost control of steering while driving on a highway.* Next, we estimated the controllability for each hazardous event (C0: simply controllable to C3: difficult to control). The controllability [MHM15] is a way of assessing the likelihood that the hazardous situation is usually controllable or not. As assumptions of the fully automated vehicles (SAE level 5), the driver is not expected to take control at any time. Therefore, we assigned the controllability as C3 (Difficult to control or uncontrollable) for each hazardous events of fully automated driving system. For example, the controllability of the hazardous event HE.1 is C3. We also assigned an ASIL to each hazardous event. For example, the ASIL of the hazardous event HE.1 is ASIL C. We formulated the safety goal for each hazardous events. For example, the safety goal for the hazardous event HE.1. is *SG.1: the fully automated driving vehicle must not lose the steering control while driving on a highway*.

Apply the STPA Steps 1 & 2: We used the output of the HARA approach (e.g. list of hazardous events and operational situations) as an input to the STPA Step 1 to formulate the unsafe control actions. We used the control structure diagram to identify the unsafe control actions of the fully automated driving vehicle (STPA Step 1). To identify unsafe control actions, we first identified the critical safety control actions at a high level of abstraction. For example, the fully automated driving function platform has a control action called *trajectory*. Primarily, the trajectory contains a time sequence of state-space points with *timestamp, x and y position, velocity, acceleration, track angle, jerk, curvature and curvature rate*. The trajectory is issued by the automated driving function platform to the motion controller.

We evaluated each of these control actions within four general hazardous types [Le11] (e.g. not providing, providing incorrect, providing at wrong timing/order, and stopped too soon/applied too long) to check whether or not they lead to hazardous events. We identified 27 unsafe control actions. For example, *UCA-1: The fully automated driving function platform does not provide a valid trajectory to motion control while driving too fast on a highway*. This unsafe control action (malfunctioning behaviour) can lead to the potential hazard HA.1.

To generate the corresponding safety constraints, we translated each unsafe control action into a corresponding safety constraint by using the guide words e.g. “shall” or “must”. For example, a corresponding safety constraint SC-1 for unsafe control action UCA-1 is: *The fully automated function platform must always provide a valid trajectory to motion control while driving on a highway*. We used the results of the situation analysis to determine the process model of the Automated Driving (AD) function platform. For example, a process model variable of the AD function platform is the *road_type* which has the following values: *highway, parking, intersection, mountain, city, urban*. We used the results of STPA Steps 0 & 1 as input to STPA Step 2 to identify the causal factors and scenarios. We

also determined the accident causes (STPA Step 2) for each unsafe control action to get a deeper understanding on how they could occur in the fully automated driving vehicle. For example, a causal scenario for the unsafe control action UCA-1 is: *The fully automated driving function platform receives wrong signals from backend due to the lack of communication while driving too fast on a highway*. Then, a new safety constraint can be derived as *The fully automated driving function platform shall receive correct data from the backend without delay during driving.*

Using the STPA Step 2 results: We used the results of the STPA Step 2 to build the safety concept and addressed the new safety requirements. For example, the causal scenario CS.1 of UCA-1 is: *The AD function platform does not provide a valid trajectory to motion control while the system is active and the vehicle is moving and there is traffic ahead on highway*. Then, we identified the safety constraints (SC) for each causal scenario. For example, the safety constraint (SC.1) for CS.1 is: *the AD function platform must always provide the trajectory to enable motion control to adjust the throttle position and apply brake friction when the vehicle is moving and there is traffic ahead to avoid a potential collision*. We used the results of the STPA Step 2 to build the functional safety concept and determine the functional safety requirements.

6 Discussion

Based on our work, we found that STPA and HARA have different base assumptions. HARA has two parts: 1) Hazard analysis which aims at identifying the hazards that lead to harm. However, these hazards are related to the individual component failures and they are not described in terms of other accident causes such as interaction failures between vehicle and its environment and driver (a passenger in a fully automated vehicle). The second part is the risk assessment which aims at identifying risks of each identified hazard. Whereas STPA focuses on control problems, not component failures. STPA aims at identifying inadequate control caused by component failures, human errors, and component interaction errors among the system components. It is also able to identify hazards that arise due to unsafe interactions among the system components in the absence of component failures. Therefore, STPA can identify more types of hazards and not only hazards which may occur due to the component failures, but also the hazards which may occur in the absence of component failures. To fill this gap between HARA and STPA, we showed how to use the results of STPA as in input to support the HARA process activities instead of mapping the HARA activities and artifact to STPA. Moreover, STPA does not support risk analysis while HARA supports this kind of activities. The work here shows that STPA and HARA are complementary to each other and STPA can be used to extend the safety scope of ISO 26262.

Another gap in the concept phase in ISO 26262 is that there is no systematic way to define the item [Ka15]. We found that STPA can fill this gap by applying the STPA Step 0 before starting the concept phase. The STPA Step 0 can define the item and establish the information needed for the item. Moreover, the STPA Step 0 can define the safety constraints for each item.

The STPA Step 2 requires that defining the process model of the controller, which determines the current state of the controlled process and how the control actions will be issued by the controller. This model also is used to determine the causal scenarios of each unsafe control action identified in STPA Step 1. However, there is no guidance on how to define the process model and its variables which should be augmented in the process model. We figured out that HARA can fill this gap by using the situation analysis which determines the operational situations and operating modes in which an item's malfunctioning behaviour will result in a hazardous event. These situations and modes can be used as input to the process model into STPA.

An assumption of STPA is that it can be applied at all stages of system development process, especially at an early stage [Le11]. This assumption is similar to the assumption of the HARA process which can also be applied at an early stage of system development. However, STPA helps to derive more detailed safety requirements, not only the functional safety requirements which are the main output of the concept phase in ISO 26262. Therefore, mapping the results of STPA Step 1 and Step 2 to build the functional safety concept requires high expertise in both STPA and HARA. Furthermore, STPA is a top-down process and the detailed design of item is not necessary to be known before applying STPA to the item. This assumption is also similar to the assumption of the HARA process which can be applied with a little bit of knowledge about the detailed design of the item.

In conclusion, we believe that STPA can support the HARA process in ISO 26262 activities and help the functional safety engineers to develop the functional safety requirements for each item identified at an early stage in the concept phase based on the results of the STPA Step 0. Indeed, the integration of STPA into HARA activities still needs modification in the assumptions and terms of both STPA and HARA to directly map the results of STPA into HARA.

7 Conclusion

In this paper, we explored the use of the STPA approach as hazard analysis in compliance with ISO 26262 to improve the safety architecture of the fully automated driving vehicle project at Continental. Our work showed that STPA is a powerful hazard analysis technique which can be used to support the safety lifecycle and HARA process in ISO 26262 by providing a systematic guidance on defining an item, deriving detailed safety constraints and developing safety goals and a functional safety concept. That helps us to evaluate the architectural design of the new fully automated driving system at an early stage of the development process. As future work, we plan to explore the use of the STPA approach in compliance with ISO 26262 at different levels of the fully automated driving architecture (e.g. software level) to develop detailed safety requirements. We plan also to conduct an empirical case study evaluating our proposed concept with functional safety engineers at Continental to understand the benefits and limitations of using STPA to support the HARA process and to extend the safety scope of ISO26262. We plan also to develop an extension to XSTAMPP [AvW] to support the HARA process activities.

References

[AvW] Abdulkhaleq , A.; Wagner, S.: XSTAMPP 2.0: new improvements to XSTAMPP Including CAST accident analysis and an extended approach to STPA. STAMP 2015, MIT, USA.

[AW13] Abdulkhaleq, A.; Wagner, S.: Experiences with Applying STPA to Software-Intensive Systems in the Automotive Domain. 2013 STAMP Conference at MIT, Boston, USA, 2013.

[AW14] Abdulkhaleq, Asim; Wagner, Stefan: A software safety verification method based on system-theoretic process analysis. In: International Conference on Computer Safety, Reliability, and Security. Springer, pp. 401–412, 2014.

[AWL15] Abdulkhaleq, Asim; Wagner, Stefan; Leveson, Nancy: A Comprehensive Safety Engineering Approach for Software-Intensive Systems Based on STPA. Procedia Engineering, 128:2 – 11, 2015. Proceedings of the 3rd European STAMP Workshop 5-6 October 2015, Amsterdam.

[Go14] Gov.UK: , Guide to Maintaining Roadworthiness: Driver and Vehicle Standards Agency and Department for Transport, 2014.

[Ho12] Hommes, Qi Van Eikema: Review and Assessment of the ISO 26262 Draft Road Vehicle-Functional Safety. In: SAE Technical Paper. 2012.

[Ho15] Hommes, Q.V. E.: , Safety Analysis Approaches for Automotive Electronic Control Systems, 2015.

[IS11] ISO: International Organization for Standardization, International Standard 26262: Road vehicles â Functional safety. International Standard. ISO, First edition, Nov. 2011.

[Ka15] Kannan, S Manoj; Dajsuren, Yanja; Luo, Yaping; Barosan, Ion: Analysis of ISO 26262 Compliant Techniques for the Automotive Domain. In: Proceedings of the International Workshop on Modelling in Automotive Software Engineering co-located with ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2015), Ottawa, Canada. volume 1487, 2015.

[Le11] Leveson, N.: Engineering a Safer World: Systems Thinking Applied to Safety. Engineering Systems. MIT Press, 2011.

[Ma16] Mallya, Archana; Pantelic, Vera; Adedjouma, Morayo; Lawford, Mark; Wassyng, Alan: Using STPA in an ISO 26262 Compliant Process. In: Computer Safety, Reliability, and Security: 35th International Conference, SAFECOMP 2016, Trondheim, Norway, September 21-23, 2016, Proceedings. Springer International Publishing, Cham, pp. 117–129, 2016.

[MHM15] Monkhouse, Helen; Habli, Ibrahim; Mcdermid, John: The Notion of Controllability in an autonmous vehicle context. In: CARS 2015-Critical Automotive applications: Robustness & Safety. 2015.

[SA16] Automated Driving Levels of Driving Automation are defined in new SAE International Standard J3016.

[Wa10] Wagner, S.; Schatz, B.; Puchner, S.; Kock, P.: A Case Study on Safety Cases in the Automotive Domain: Modules, Patterns, and Models. In: 2010 IEEE 21st International Symposium on Software Reliability Engineering. pp. 269–278, Nov 2010.

[YL14] Young, William; Leveson, Nancy G.: An Integrated Approach to Safety and Security Based on Systems Theory. Commun. ACM, 57(2):31–35, February 2014.

Towards the Use of Controlled Natural Languages in Hazard Analysis and Risk Assessment

Paul Chomicz[1] Armin Müller-Lerwe[2] Götz-Philipp Wegner[2] Rainer Busch[2]
Stefan Kowalewski[1]

Abstract: New safety-critical and software-controlled systems of automobiles have to be developed according to the functional safety standard ISO 26262. A hazard analysis and risk assessment has to be performed for such systems. The sub-activities of this analysis technique are defined by the standard, but informative definitions leave room for subjective variation, and documentation details are left to the car manufacturer. Usually, natural languages are used for the documentation, which are powerful and expressive but also complex and ambiguous. We propose the usage of controlled natural languages for the documentation of the results of the hazard analysis and risk assessment. In a first step, we developed a controlled natural language for the description of the hazardous events. The language reduces ambiguity and improves the consistency across hazard analyses and risk assessments.

Keywords: Controlled Natural Language, Hazard Analysis and Risk Assessment, Functional Safety, ISO 26262, Hazardous Event

1 Introduction

In the automotive industry, new safety-critical functions that are realized by software-controlled electric or electronic systems have to be developed in accordance with the functional safety standard ISO 26262 [IS11]. Its safety lifecycle encompasses principal safety activities during the concept phase, product development, and after start of production.

The hazard analysis and risk assessment (HARA) is a safety activity which is performed during the concept phase. The objective is to identify and categorize the potential hazards of functions and to derive safety goals for the prevention or mitigation of these hazards. The hazard analysis and risk assessment comprises three steps. The first step is the situation analysis and hazard identification. Potential unintended behaviors have to be identified that could lead to a hazard within a specific situation (hazardous event). Afterwards, the risk classification of the hazardous events takes place. The risk of each hazardous event is classified by determining the severity (S), the probability of exposure (E), and the controllability (C). According to these three parameters, the required automotive safety integrity level (ASIL) is assigned in the last step. The ASIL specifies the level of risk reduction for achieving an acceptable residual risk with ASIL D representing the highest and ASIL A the lowest level [IS11].

[1] RWTH Aachen University, Lehrstuhl Informatik 11 – Embedded Software, Ahornstraße 55, 52074 Aachen, {chomicz, kowalewski}@embedded.rwth-aachen.de
[2] Ford Research & Innovation Center Aachen, Süsterfeldstraße 200, 52072 Aachen, {amuell12, gwegner2, rbusch1}@ford.com

The ISO 26262 standard describes which sub-activities are part of the hazard analysis and risk assessment, but it does not describe, for example, how to record the unintended behaviors or how to determine one of the risk parameters. The standard defines severity, exposure, and controllability in a qualitative way that leaves room for subjective interpretation. Due to the fact that usually mutiple new functions use the same actuators, malfunctions could often cause similar hazards. Since new functions and systems are developed by different teams, it is a challenge to assure consistency of the risk classifications between the hazard analyses and risk assessments developed for different vehicle functions. Inconsistency might lead to different levels of safety measures for similar hazardous events.

In analysis techniques like hazard analysis and risk assessment, a natural language is usually used for the documentation. On the one hand, natural languages are powerful and expressive, but on the other hand, they are complex and ambiguous. Same or similar hazardous events and rationales for the classification are often described using different wordings and phrases. This makes it difficult to check for consistency especially across HARAs that are developed by different teams. In order to approach these problems, controlled natural languages are a promising way.

A controlled natural language (CNL) is a subset of a natural language [Ku14]. CNLs are obtained by restricting the vocabulary or the grammar. These restrictions aim to increase terminological consistency and to reduce ambiguity and complexity. Numerous controlled natural languages have been developed for various domains, e.g. Airbus Warning Language [SBC03], Attempto Controlled English (ACE) [FKK08], or Bio-Query-CNL [EY09].

The controlled natural languages for the hazard analysis and risk assessment should allow the description of hazardous events and the rationales of the risk parameters in such a way that it supports the engineers in the development of HARAs, e.g. by a more efficient search for existing ratings of similar hazardous events. It should also reduce the possibility of formulating similar hazardous events or rationales with different wordings and phrases, and additionally, the language should enable or simplify an automatic consistency check between different HARAs.

The remainder of this paper is structured as follows. The next section describes a particular controlled natural language in detail, which was developed and put into operation at Ford Motor Company. Then, the formalization of the hazardous events is explained including the process to accomplish it. The fourth section gives details on the evaluation of the newly created language and first experiences about a productive usage. The last section presents an outlook on future work.

2 Related Work

There are numerous controlled natural languages in various domains [Ku14]. In general, CNLs can be divided into general-purpose languages and domain-purpose languages [Sc10]. A general-purpose language has not been designed for a specific scenario or application domain. An example of such a language is Attempto Controlled English (ACE)

[FKK08]. On the contrary, domain-purpose languages have been developed for a particular application area. The Airbus Warning Language [SBC03] and the Bio-Query-CNL [EY09] are such languages.

To the best of our knowledge, no controlled natural language has been developed specifically for hazard analysis and risk assessment. Some work was done in the area of applying CNLs to (safety) requirements engineering in the automotive domain [PMP11, HMM11, FH14]. In the following, a controlled natural language will be introduced that was developed at the Ford Motor Company. The structure of the language is similar to ours, and especially, the experiences that were made during the development and the productive usage at Ford are of great value.

The Standard Language (SLANG) is a controlled natural language that was developed to write process build instructions for a vehicle [Ry02]. Before using this controlled language, build instructions were written in a natural language causing problems such as ambiguity and inconsistency. Furthermore, Ford's vehicle assembly plants are spread all over the world, and threrefore, several different natural languages were used for the process sheets.

In addition to the language, the Direct Labor Management System (DLMS) was developed to address these problems [O'89]. The tool assists the vehicle assembly process planning. As input, process sheets written in the Standard Language are taken to produce detailed work tasks for each step of the assembly process. Before releasing these tasks to the assembly plants, they are translated into the corresponding language automatically by the system. The usage of the controlled natural language enables or simplifies the automatic machine translation and the precise determination of all work tasks, since the language is constructed in such a way that it can be processed by the system.

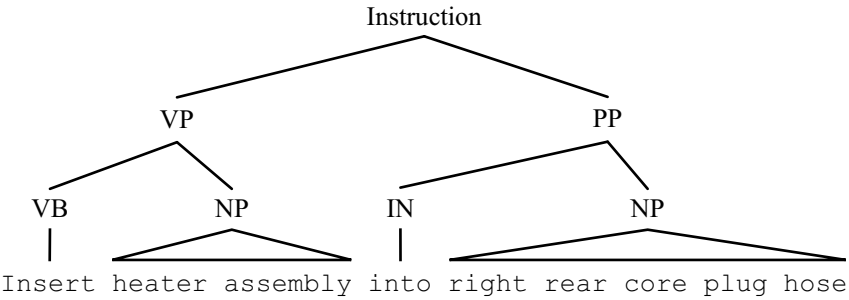


Fig. 1: Instruction written in SLANG [Ry05]

Instructions written in the Standard Language are in an imperative form including a verb phrase (VP), which contains a noun phrase (NP) that serves as the object of the verb. The level of detail can be increased by adding adverbs, adjuncts, or prepositional phrases (PP). The verb is the key word of every instruction, and every verb describes a particular, precisely defined action. The number of verbs in the vocabulary is limited, and certain prepositions have also a specific meaning that will be handled in a pre-determined manner by the system. Every part of the text that is delimited by brackets is ignored by the system

and does not have to conform to the Standard Language. Thus, it is possible to add comments or remarks to the build instructions. Figure 1 shows an example instruction written in SLANG.

The vocabulary of the language is limited to a certain set of words. Since, the vehicle assembly process is very dynamic, it is necessary to extend the vocabulary along with new vehicles and assembly plants that are added to the system. Therefore, process engineers have to request changes which need to be approved by an internal Ford systems organization before they will be added into the language and the system.

As the system simply flags any errors, the process engineers need an enhanced knowledge about the language to be able to fix the errors or to write correct build instructions in the first place. Along with the introduction of the controlled natural language for the productive usage, the process engineers were trained to write the process sheets in SLANG. During the initial implementation, the users resisted to use the language until they were trained and learned how to use it effectively. Another problem that was encountered was the misuse of the commentary function. This feature was used by users to bypass the process of adding new terms into the language [Ry06].

The Standard Language is a controlled natural language that was never designed to produce correct grammatical sentences with respect to the English language [Ry02]. The goals of the language were to develop consistent and precisely defined means of communicating and to enable or to simplify machine translation. Therefore, the process sheets get more precise and simpler in terms of automatic processing, but the language is less expressive and loses a part of its naturalness compared to the English language. As a consequence, instructions become less readable and less understandable for humans, especially for untrained users. However, the restrictions on the vocabulary and the grammar improve the quality of the translations. To counteract the negative effects of the newly created language, effort needs to be made to train the process engineers.

SLANG is a domain-purpose language. It is too domain-specific so that it could be reused for our purpose. Whereas, the usage of a general-purpose language would be in general possible, but it would have also some drawbacks. Such a language is not optimized to our domain-specific application. Certainly, the vocabulary would have to be adapted to the automotive domain. Furthermore, the structure of the language might not be suitable with respect to how the descriptions were written before by the safety engineers. Therefore, we decided to develope an own language that is close to existing hazardous event descriptions.

3 Controlled Natural Language for Hazardous Event Descriptions

In this section, the process of the formalization of the hazardous event descriptions and the formalization itself are described. For this purpose, we have analyzed existing hazard analysis and risk assessment documents provided by the Ford Motor Company. Based on this analysis, the controlled natural language was created. Furthermore, the translation of hazardous event descriptions that do not conform to the CNL is explained by means of two examples.

3.1 Analysis Process

To achieve a formalization of the hazardous event description, existing hazard analyses and risk assessments provided by the Ford Motor Company were analyzed. Our approach is bottom-up and iterative. In the first iteration, nine HARA documents have been analyzed. The documents describe the hazard analysis and risk assessment of, among others, an emergency braking system and an electronic controlled differential. The HARAs were performed by different teams and in different countries. However, the English language was used in all cases for the documentation.

From the provided documents, 208 different hazardous event descriptions were extracted, and the structure of the descriptions and the used wording were analyzed to create the controlled language. Table 1 contains an exemplary set of such descriptions that represents how hazardous events are currently described.[3] A hazardous event description consists of at least one hazardous event, which might be caused by another event. Thus, it is possible to construct causal chains.

The structure of the hazardous event descriptions can be divided into two categories. The first category contains descriptions that were written in a bullet-point manner. The hazardous events in the second category were formulated using full sentences. The descriptions 1 and 4 of Table 1 are written in a bullet-point manner, and the other two are part of the second category. Overall, 141 descriptions belong to the bullet-point manner category (67.8 %) and 45 descriptions are part of the full sentence category (21.6 %). 22 hazardous events were formulated using both full sentences and bullet-point descriptions (10.6 %). The categorization was performed manually.

No.	*Hazardous Event Description*
1	`Fire outside passenger compartment.`
2	`The driver is not alerted to a credible threat.`
3	`The system is active at high speed and may not detect objects in relevant distance (due to sensor performance).`
4	`Unintended and unlimited system activation leading to loss of vehicle steerability due to blocked wheels without ABS.`

Tab. 1: Exemplary set of hazardous event descriptions

The intermediate formalization for the first set of HARA documents was reviewed in a second iteration, where seven different documents have been analyzed. Again, the provided data fulfilled the same properties as the first one, e.g. the HARAs were performed by different teams. The data set contains the documentation of the hazard analysis and risk assessment for an electronic clutch and a park assist.

[3] The examples are slightly modified to avoid the disclosure of proprietary information about the analyzed systems.

93 additional hazardous events that are different compared to the first set were extracted from this set. The structure of the descriptions is the same as in the first set, and the used wording is similar apart from system-related words. 76 hazardous events are written in a bullet-point manner (81.7 %) and 12 descriptions are formulated using full sentences (12.9 %). Again, a small portion of the descriptions is formulated using both full sentences and bullet-point descriptions (5.4 %).

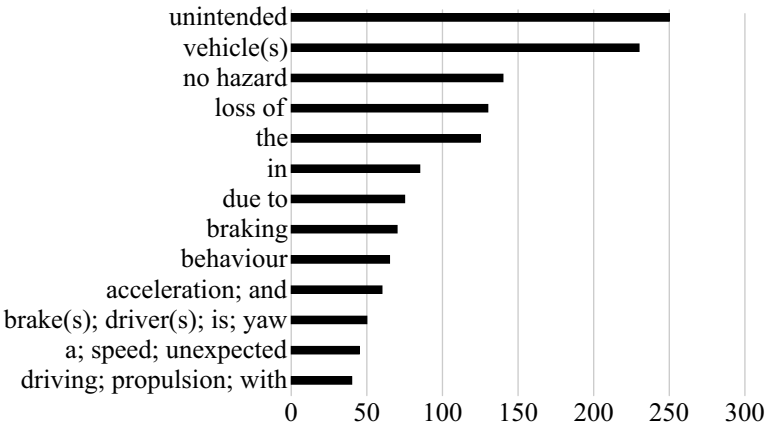


Fig. 2: Most frequently used words and phrases in hazardous event descriptions

Figure 2 depicts a frequency count of the words and phrases that were mostly used for the description of hazardous events. Most of the words are conjunctions, prepositions, adjectives, and nouns. Verbs were rarely used for the descriptions since most of them were written in a bullet-point manner and not as full sentences.

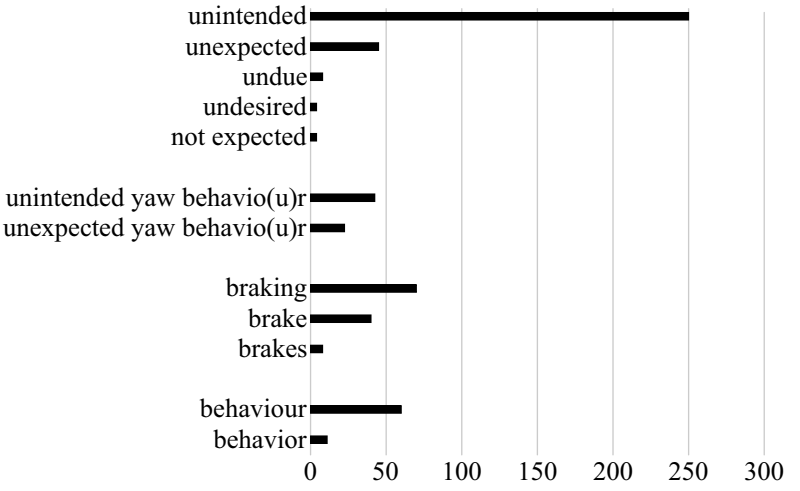


Fig. 3: Synonyms and similar words and phrases in hazardous event descriptions

In addition to this frequency count, an exemplary set of synonyms and words with the same or similar meaning but different spelling, e.g. American and British English, are

depicted in Figure 3. The first group contains words and phrases that are synonyms. The meaning might be slightly different between these words and phrases, but in our context, the differences can be ignored. An example is contained in the second group. This group consists of two different hazardous event descriptions that share nearly the same meaning. Another syntactically different description with the same semantic that is not contained in the analyzed HARAs is “unintended steering input”.

The third group contains words with the same word stem. The hazardous event “unintended braking” and “unintended brake activation” describe the same event using different wordings based on a word with the same word stem. The last group shows an example of the difference between American English and British English. Same words with slightly different spelling are contained in either language. This is an additional fact that has to be considered during the creation of a CNL.

3.2 Formalization

In this subsection, the results of the formalization of the hazardous event descriptions are presented. The restrictions for the developed controlled natural language are made on both the structure of the descriptions, so the grammar of the language, and the vocabulary. The language was developed in a bottom-up approach, and therefore, it is closely related to the provided data.

The grammar only allows to write the hazardous event descriptions in a bullet-point manner. Noun phrases are used to describe an event or a characteristic of a system. The phrase has a noun as its headword and can contain additional adnominals, like adverbs, adjectives, or noun adjuncts. The usage of pronouns and clauses in these noun phrases is prohibited. Certain prepositions and conjunctions are used to connect single hazardous events to build up more complex descriptions and to be able to create causal chains. The prepositions are divided into semantical categories, e.g. the prepositions “between” and “in front of” indicate position information or “by” and “to” indicates the point of view.

The controlled natural language does not provide the possibility to use full sentences for the descriptions. Therefore, verbs are not needed and not part of the language. This restriction further reduces the complexity of the language in comparison to the complete English language. Without verbs, the distinction between active and passive voice does not have to be considered, and further on, grammatical tenses are also omitted.

The first example from Table 1 conforms already to the grammar of the controlled language. The description consists of a noun phrase with a single noun and a prepositional phrase giving additional position information. Figure 4 depicts the example along with the classification of the part of speeches.

The last example from Table 1 is almost a correct description according to the controlled language. The description starts with a noun phrase consisting of additional adjectives followed by two causal phrases (CP). Causal phrases are phrases that start with a causal

linking phrase (CLP) followed by a noun phrase. Such phrases enable the user to formulate causal relationships.

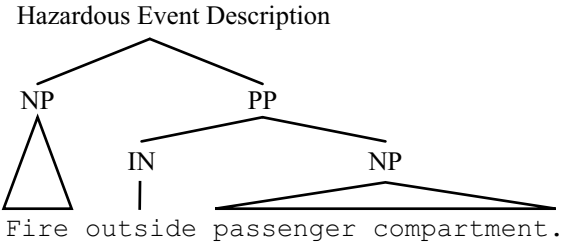


Fig. 4: Correct example with regard to the CNL

In this case, two different causal linking phrases have been used describing slightly different causal relationships. “A leading to B” expresses that an event A is the cause of an event B, whereas “A due to B” expresses that an event B is the cause of an event A. Both can be used interchangeably. However, describing a situation with more than one cause and using both expressions together might lead to a lack of causal relationship information.

In the example depicted in Figure 5, the structure of the description is “A leading to B due to C”. From this formulation, it can be interpreted that the events A and C are the causes of event B but nothing is said about the relationship between A and C. Considering the events in more detail, the event A (“unintended and unlimited system activation”) happens before the event C (“blocked wheels without ABS”). To avoid such a lack of information and to reduce misunderstandings, we restrict the controlled natural language to use only the causal linking phrase “due to”.

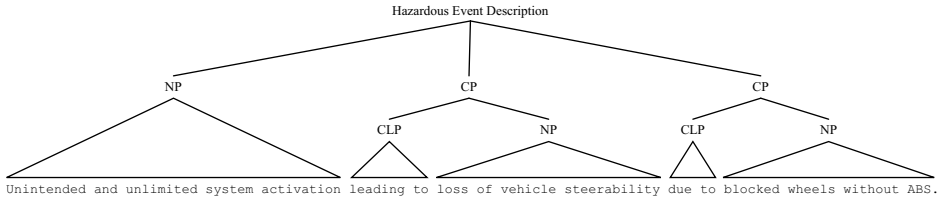


Fig. 5: Almost correct example with regard to the CNL

Since the usage of such a new controlled natural language might be difficult at the beginning, a commentary functionality is introduced to enable the user to write parts of the description using the complete English language. The users should be able to write the first ideas of the description or parts of it in the language they know and afterwards to translate it into the required form. This functionality is rather intended for the productive usage of the language than being a part of the language itself. Still it can be used to provide additional information, which is related to the hazardous event but is not part of its description.

The vocabulary of the CNL is restricted to the words that have been used in the provided documents and which were not removed during the formalization. In this context, words have been identified that share the same semantics. For example, the words “unintended”,

"unexpected", "unwanted", and "undesired" are semantically equivalent in our context as already mentioned above. Only one of the words ("unintended") is contained in our vocabulary, and the other synonyms are prohibited.

3.3 Translation

Hazardous event descriptions that were formulated using full sentences can be translated into bullet-point manner descriptions which are semantically equal and conform to the developed controlled natural language. In this subsection, the hazardous events 2 and 3 of Table 1 will be exemplarily translated into our language.

The general translation procedure is as follows. First, the determination of the part of speech of every word has to be performed [Br00]. Based on the given parts of speech, the translation is performed by removing words with certain parts of speech or transforming words into related words with a different part of speech. Some of the transformations will be explained by the two following examples.

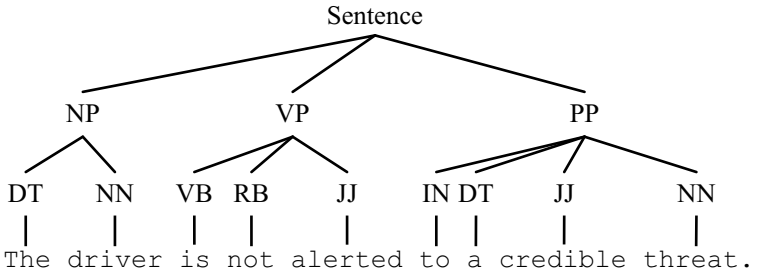


Fig. 6: Incorrect hazardous event description as a full sentence

The first example is shown in Figure 6 along with the classification of the parts of speech. The first part of the description is a simple sentence with a subject, verb, and adjective. In such a case, the "to be" verb can be simply removed and the order of noun and adjective switched resulting in "not alerted driver". If the adjective is modified by an adverb, then this will be moved along with the adjective just like in this example. The prepositional phrase consisting of a preposition and a noun phrase can remain unchanged. As a result, the hazardous event description "not alerted driver to a credible threat." conforms to the controlled natural language.

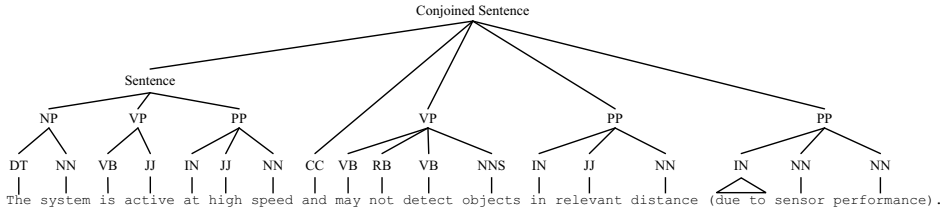


Fig. 7: Incorrect hazardous event description as a conjoined sentence

Figure 7 displays a more complex example. The first part of the conjoined sentence can be translated accordingly to the first example to “active system at high speed”. The second part contains an object after the verb followed by a prepositional phrase. The verb needs to be translated into an adjective that describes the effect on the object. The object is used as the noun phrase resulting in a passive description. This leads to “undetected objects”. The prepositional phrase can be simply taken as it is and added at the end. The last part of the description is written in brackets. The intention behind the usage of the brackets is not defined and gives room for various interpretations. In this case, it might be additional information or an assumption for the root of the hazard. A complete translation of this example might be “active system at high speed and undetected objects in relevant distance by the system due to sensor performance.”.

4 Evaluation

In total, 301 hazardous event descriptions were extracted from 16 different hazard analysis and risk assessment documents. 217 descriptions were already written in a bullet-point manner (72.1 %), and 57 hazardous events were described in full sentences (18.9 %). The remaining 27 descriptions were written in a mixed version (9.0 %).

The newly created controlled natural language for the hazardous event descriptions has been evaluated against the provided data. 156 out of the 217 descriptions that were written in a bullet-point manner are already in line with the CNL (71.9 %). Another 48 hazardous events could be translated into a correct form by replacing a synonym with the correct word that is part of the vocabulary (22.1 %). The other descriptions, including those that were written in full sentences or a mixed version, could be all translated into semantically equivalent hazardous event descriptions that conform to the language, as exemplarily shown in subsection 3.3. These results show that the language is closely related to the provided data but still expressive enough to describe every hazardous event that has arised in the considered HARAs.

Furthermore, the new language was prototypically applied in hazard analyses and risk assessments for new systems to make first experiences in a productive usage. Different engineers use the CNL during the development of three new HARAs within the domains steering, fuel cell, and powertrain. By extending the vocabulary with domain-specific terms, it was feasible to describe all hazardous events according to the CNL. It turned out that using the language leads to more consistent descriptions within the HARAs compared to existing HARAs.

On the other hand, it is difficult to write valid descriptions according to the CNL without good knowledge about the vocabulary and the grammar. Therefore, instantaneous support while entering data seems to be essential to enable engineers to use the controlled language effectively when performing hazard analyses and risk assessments.

As a last point, two examples are presented to show in which way the controlled natural language is able to avoid inconsistency. The vocabulary does not contain synonyms. Therefore, it reduces the possibility to write semantically equal descriptions syntactically

differently. For example, the two descriptions “unintended acceleration.” and “unwanted speed-up.” have the same meaning. The words “unintended” and “acceleration” are part of the vocabulary, but “unwanted” and “speed-up” are not, since these are synonyms of the other two words.

The second example concerns the structure of the descriptions. Using the English language, the two descriptions “vehicle may pull towards the opposite lane or the side of the road due to understeering behavior.” and “lane departure due to understeering.” with the same meaning were permitted, but only the second one conforms to the controlled natural language. The restrictions on the vocabulary and on the structure of the descriptions are intended to unify the descriptions and to reduce ambiguity and complexity.

5 Conclusion and Outlook

The formalization of the hazardous event descriptions is the first step towards the utilization of controlled natural languages for the hazard analysis and risk assessment according to ISO 26262. The controlled natural language defines a restricted common structure for the descriptions along with a limited vocabulary. Therefore, the complexity and ambiguity are reduced in the documentation of the HARA resulting in less inconsistency. Furthermore, the common structure simplifies the search for existing same or similar hazardous event descriptions.

The evaluation shows that all existing hazardous event descriptions of the provided HARA documents can be translated into the controlled language. Furthermore, a large portion of the descriptions was already compliant with the language. The CNL was prototypically applied to three newly created HARAs at the Ford Research & Innovation Center Aachen. It turned out that the language was applicable after extending the vocabulary with domain-specific terms, and all hazardous events could be described in that language. Currently, the vocabulary is restricted to the used words in the analyzed HARAs, and it needs to be extended beyond the scope of the provided documents.

Besides the description of the hazardous events, the rationales for the ratings of the parameters severity, exposure, and controllability are an essential part of the documentation of the hazard analysis and risk assessment. Therefore, controlled natural languages for the three rationales shall be provided to complete the set of languages. After the completion, all the languages shall be implemented in a prototype tool to further examine the usage of such languages for the HARA. Based on the prototype tool, a case study will be performed to gather more user experiences that shall help to improve the languages and their usage.

References

[Br00] Brill, Eric: Part-of-Speech Tagging. Handbook of Natural Language Processing, pp. 403–414, 2000.

[EY09] Erdem, Esra; Yeniterzi, Reyyan: Transforming Controlled Natural Language Biomedical Queries into Answer Set Programs. In: Proceedings of the Workshop on Current Trends

in Biomedical Natural Language Processing. Association for Computational Linguistics, pp. 117–124, 2009.

[FH14] Fockel, Markus; Holtmann, Jörg: A Requirements Engineering Methodology Combining Models and Controlled Natural Language. In: Model-Driven Requirements Engineering Workshop (MoDRE), 2014 IEEE 4th International. IEEE, pp. 67–76, 2014.

[FKK08] Fuchs, Norbert E.; Kaljurand, Kaarel; Kuhn, Tobias: Attempto Controlled English for Knowledge Representation. In: Reasoning Web, pp. 104–124. Springer, 2008.

[HMM11] Holtmann, Jörg; Meyer, Jan; Meyer, Matthias: A Seamless Model-Based Development Process for Automotive Systems. In: Software Engineering (Workshops). pp. 79–88, 2011.

[IS11] ISO: , ISO 26262-3: Road Vehicles – Functional Safety – Part 3: Concept Phase, 2011.

[Ku14] Kuhn, Tobias: A Survey and Classification of Controlled Natural Languages. Computational Linguistics, 40(1):121–170, 2014.

[O'89] O'Brien, John; Brice, Henry; Hatfield, Scott; Johnson, Wayne P; Woodhead, Richard: The Ford Motor Company Direct Labor Management System. In: Innovative Applications of Artificial Intelligence. volume 1, 1989.

[PMP11] Post, Amalinda; Menzel, Igor; Podelski, Andreas: Applying Restricted English Grammar on Automotive Requirements – Does It Work? A Case Study. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, pp. 166–180, 2011.

[Ry02] Rychtyckyj, Nestor: An Assessment of Machine Translation for Vehicle Assembly Process Planning at Ford Motor Company. In: Conference of the Association for Machine Translation in the Americas. Springer, pp. 207–215, 2002.

[Ry05] Rychtyckyj, Nestor: Ergonomics Analysis for Vehicle Assembly Using Artificial Intelligence. AI Magazine, 26(3):41, 2005.

[Ry06] Rychtyckyj, Nestor: Standard Language at Ford Motor Company: A Case Study in Controlled Language Development and Deployment. Cambridge, Massachussets, 2006.

[SBC03] Spaggiari, Laurent; Beaujard, Florence; Cannesson, Emmanuelle: A Controlled Language at Airbus. Proceedings of EAMT-CLAW03, pp. 151–159, 2003.

[Sc10] Schwitter, Rolf: Controlled Natural Languages for Knowledge Representation. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters. Association for Computational Linguistics, pp. 1113–1121, 2010.

GI-Edition Lecture Notes in Informatics

P-1 Gregor Engels, Andreas Oberweis, Albert Zündorf (Hrsg.): Modellierung 2001.

P-2 Mikhail Godlevsky, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications, ISTA'2001.

P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Lan-guage to Information Systems, NLDB'2001.

P-4 H. Wörn, J. Mühling, C. Vahl, H.-P. Meinzer (Hrsg.): Rechner- und sensorgestützte Chirurgie; Workshop des SFB 414.

P-5 Andy Schürr (Hg.): OMER – Object-Oriented Modeling of Embedded Real-Time Systems.

P-6 Hans-Jürgen Appelrath, Rolf Beyer, Uwe Marquardt, Heinrich C. Mayr, Claudia Steinberger (Hrsg.): Unternehmen Hochschule, UH'2001.

P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods – Countering or Integrating the extremists, pUML'2001.

P-8 Reinhard Keil-Slawik, Johannes Magenheim (Hrsg.): Informatikunterricht und Medienbildung, INFOS'2001.

P-9 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Innovative Anwendungen in Kommunikationsnetzen, 15. DFN Arbeitstagung.

P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.

P-11 Michael Weber, Frank Kargl (Hrsg.): Mobile Ad-Hoc Netzwerke, WMAN 2002.

P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.

P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahni_ (Hrsg.): The Changing Universities – The Role of Technology.

P-14 Robert Tolksdorf, Rainer Eckstein (Hrsg.): XML-Technologien für das Semantic Web – XSW 2002.

P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.

P-16 J. Felix Hampe, Gerhard Schwabe (Hrsg.): Mobile and Collaborative Business 2002.

P-17 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Zukunft der Netze –Die Verletzbarkeit meistern, 16. DFN Arbeitstagung.

P-18 Elmar J. Sinz, Markus Plaha (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2002.

P-19 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3. Okt. 2002 in Dortmund.

P-20 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3. Okt. 2002 in Dortmund (Ergänzungsband).

P-21 Jörg Desel, Mathias Weske (Hrsg.): Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.

P-22 Sigrid Schubert, Johannes Magenheim, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur "Didaktik der Informatik" – Theorie, Praxis, Evaluation.

P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 – Fortschritt durch Beständigkeit

P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen

P-25 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003

P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web

P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin

P-28 Ulrich Reimer, Andreas Abecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement – Er-fahrungen und Visionen

P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB'2003: Natural Language Processing and Information Systems

P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications

P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometrics and Electronic Signatures

P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003

P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft

P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)

P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)

P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenberg (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik

P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik

P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003

P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003

P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004

P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Edts.): ARCS 2004 – Organic and Pervasive Computing

P-42 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste

P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistance, Scalability, Transactions – Database Mechanisms for Mobile Applications

P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning. E-Services

P-45 Bernhard Rumpe, Wofgang Hesse (Hrsg.): Modellierung 2004

P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment

P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society

P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications

P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven

P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-52 Gregor Engels, Silke Seehusen (Hrsg.): DELFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik

P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004

P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration

P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Adhoc-Netze Medienintegration

P-56 Fernand Feltz, Andreas Oberweis, Benoit Otjacques (Hrsg.): EMISA 2004 – Informationssysteme im E-Business und E-Government

P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen

P-58 Sami Beydeda, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality

P-59 J. Felix Hampe, Franz Lehner, Key Pousttchi, Kai Ranneberg, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments

P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für inforrmatische Bildung

P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen

P-62 Federrath, Hannes (Hrsg.): „Sicherheit 2005“ – Sicherheit – Schutz und Zuverlässigkeit

P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and ist Applications

P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005

P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolffried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web

P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): DeLFI 2005: 3. deutsche e-Learning Fachtagung Informatik

P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)

P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)

P-69 Robert Hirschfeld, Ryszard Kowalcyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005

P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)

P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005

P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment

P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): “Heute schon das Morgen sehen“

P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology

P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture

P-76 Thomas Kirste, Birgitta König-Riess, Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz

P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006

P-78 K.-O. Wenkel, P. Wagner, M. Morgenstern, K. Luzi, P. Eisermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel

P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006

P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce

P-81 Wolfgang Karl, Jürgen Becker, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle (Hrsg.): ARCS´06

P-82 Heinrich C. Mayr, Ruth Breu (Hrsg.): Modellierung 2006

P-83 Daniel Huson, Oliver Kohlbacher, Andrei Lupas, Kay Nieselt and Andreas Zell (eds.): German Conference on Bioinformatics

P-84 Dimitris Karagiannis, Heinrich C. Mayr, (Hrsg.): Information Systems Technology and its Applications

P-85 Witold Abramowicz, Heinrich C. Mayr, (Hrsg.): Business Information Systems

P-86 Robert Krimmer (Ed.): Electronic Voting 2006

P-87 Max Mühlhäuser, Guido Rößling, Ralf Steinmetz (Hrsg.): DELFI 2006: 4. e-Learning Fachtagung Informatik

P-88 Robert Hirschfeld, Andreas Polze, Ryszard Kowalczyk (Hrsg.): NODe 2006, GSEM 2006

P-90 Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, Klaus Turowski (Hrsg.): Integration, Informationslogistik und Architektur

P-91 Henrik Stormer, Andreas Meier, Michael Schumacher (Eds.): European Conference on eHealth 2006

P-92 Fernand Feltz, Benoît Otjacques, Andreas Oberweis, Nicolas Poussing (Eds.): AIM 2006

P-93 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 1

P-94 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 2

P-95 Matthias Weske, Markus Nüttgens (Eds.): EMISA 2005: Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen

P-96 Saartje Brockmans, Jürgen Jung, York Sure (Eds.): Meta-Modelling and Ontologies

P-97 Oliver Göbel, Dirk Schadt, Sandra Frings, Hardo Hase, Detlef Günther, Jens Nedon (Eds.): IT-Incident Mangament & IT-Forensics – IMF 2006

P-98 Hans Brandt-Pook, Werner Simonsmeier und Thorsten Spitta (Hrsg.): Beratung in der Softwareentwicklung – Modelle, Methoden, Best Practices

P-99 Andreas Schwill, Carsten Schulte, Marco Thomas (Hrsg.): Didaktik der Informatik

P-100 Peter Forbrig, Günter Siegel, Markus Schneider (Hrsg.): HDI 2006: Hochschuldidaktik der Informatik

P-101 Stefan Böttinger, Ludwig Theuvsen, Susanne Rank, Marlies Morgenstern (Hrsg.): Agrarinformatik im Spannungsfeld zwischen Regionalisierung und globalen Wertschöpfungsketten

P-102 Otto Spaniol (Eds.): Mobile Services and Personalized Environments

P-103 Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, Christoph Brochhaus (Hrsg.): Datenbanksysteme in Business, Technologie und Web (BTW 2007)

P-104 Birgitta König-Ries, Franz Lehner, Rainer Malaka, Can Türker (Hrsg.) MMS 2007: Mobilität und mobile Informationssysteme

P-105 Wolf-Gideon Bleek, Jörg Raasch, Heinz Züllighoven (Hrsg.) Software Engineering 2007

P-106 Wolf-Gideon Bleek, Henning Schwentner, Heinz Züllighoven (Hrsg.) Software Engineering 2007 – Beiträge zu den Workshops

P-107 Heinrich C. Mayr, Dimitris Karagiannis (eds.) Information Systems Technology and its Applications

P-108 Arslan Brömme, Christoph Busch, Detlef Hühnlein (eds.) BIOSIG 2007: Biometrics and Electronic Signatures

P-109 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 1

P-110 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 2

P-111 Christian Eibl, Johannes Magenheim, Sigrid Schubert, Martin Wessner (Hrsg.) DeLFI 2007: 5. e-Learning Fachtagung Informatik

P-112 Sigrid Schubert (Hrsg.) Didaktik der Informatik in Theorie und Praxis

P-113 Sören Auer, Christian Bizer, Claudia Müller, Anna V. Zhdanova (Eds.) The Social Semantic Web 2007 Proceedings of the 1st Conference on Social Semantic Web (CSSW)

P-114 Sandra Frings, Oliver Göbel, Detlef Günther, Hardo G. Hase, Jens Nedon, Dirk Schadt, Arslan Brömme (Eds.) IMF2007 IT-incident management & IT-forensics Proceedings of the 3rd International Conference on IT-Incident Management & IT-Forensics

P-115 Claudia Falter, Alexander Schliep, Joachim Selbig, Martin Vingron and Dirk Walther (Eds.) German conference on bioinformatics GCB 2007

P-116 Witold Abramowicz, Leszek Maciszek (Eds.) Business Process and Services Computing 1st International Working Conference on Business Process and Services Computing BPSC 2007

P-117 Ryszard Kowalczyk (Ed.) Grid service engineering and manegement The 4th International Conference on Grid Service Engineering and Management GSEM 2007

P-118 Andreas Hein, Wilfried Thoben, Hans-Jürgen Appelrath, Peter Jensch (Eds.) European Conference on ehealth 2007

P-119 Manfred Reichert, Stefan Strecker, Klaus Turowski (Eds.) Enterprise Modelling and Information Systems Architectures Concepts and Applications

P-120 Adam Pawlak, Kurt Sandkuhl, Wojciech Cholewa, Leandro Soares Indrusiak (Eds.) Coordination of Collaborative Engineering - State of the Art and Future Challenges

P-121 Korbinian Herrmann, Bernd Bruegge (Hrsg.) Software Engineering 2008 Fachtagung des GI-Fachbereichs Softwaretechnik

P-122 Walid Maalej, Bernd Bruegge (Hrsg.) Software Engineering 2008 - Workshopband Fachtagung des GI-Fachbereichs Softwaretechnik

P-123 Michael H. Breitner, Martin Breunig, Elgar Fleisch, Ley Pousttchi, Klaus Turowski (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Technologien, Prozesse, Marktfähigkeit
Proceedings zur 3. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2008)

P-124 Wolfgang E. Nagel, Rolf Hoffmann, Andreas Koch (Eds.)
9th Workshop on Parallel Systems and Algorithms (PASA)
Workshop of the GI/ITG Speciel Interest Groups PARS and PARVA

P-125 Rolf A.E. Müller, Hans-H. Sundermeier, Ludwig Theuvsen, Stephanie Schütze, Marlies Morgenstern (Hrsg.)
Unternehmens-IT:
Führungsinstrument oder Verwaltungsbürde
Referate der 28. GIL Jahrestagung

P-126 Rainer Gimnich, Uwe Kaiser, Jochen Quante, Andreas Winter (Hrsg.)
10th Workshop Software Reengineering (WSR 2008)

P-127 Thomas Kühne, Wolfgang Reisig, Friedrich Steimann (Hrsg.)
Modellierung 2008

P-128 Ammar Alkassar, Jörg Siekmann (Hrsg.)
Sicherheit 2008
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
2.-4. April 2008
Saarbrücken, Germany

P-129 Wolfgang Hesse, Andreas Oberweis (Eds.)
Sigsand-Europe 2008
Proceedings of the Third AIS SIGSAND European Symposium on Analysis, Design, Use and Societal Impact of Information Systems

P-130 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
1. DFN-Forum Kommunikations-technologien Beiträge der Fachtagung

P-131 Robert Krimmer, Rüdiger Grimm (Eds.)
3rd International Conference on Electronic Voting 2008
Co-organized by Council of Europe, Gesellschaft für Informatik and E-Voting. CC

P-132 Silke Seehusen, Ulrike Lucke, Stefan Fischer (Hrsg.)
DeLFI 2008:
Die 6. e-Learning Fachtagung Informatik

P-133 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 1

P-134 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 2

P-135 Torsten Brinda, Michael Fothe, Peter Hubwieser, Kirsten Schlüter (Hrsg.)
Didaktik der Informatik –
Aktuelle Forschungsergebnisse

P-136 Andreas Beyer, Michael Schroeder (Eds.)
German Conference on Bioinformatics GCB 2008

P-137 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2008: Biometrics and Electronic Signatures

P-138 Barbara Dinter, Robert Winter, Peter Chamoni, Norbert Gronau, Klaus Turowski (Hrsg.)
Synergien durch Integration und Informationslogistik
Proceedings zur DW2008

P-139 Georg Herzwurm, Martin Mikusz (Hrsg.)
Industrialisierung des Software-Managements
Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschaftsinformatik

P-140 Oliver Göbel, Sandra Frings, Detlef Günther, Jens Nedon, Dirk Schadt (Eds.)
IMF 2008 - IT Incident Management & IT Forensics

P-141 Peter Loos, Markus Nüttgens, Klaus Turowski, Dirk Werth (Hrsg.)
Modellierung betrieblicher Informations-systeme (MobIS 2008)
Modellierung zwischen SOA und Compliance Management

P-142 R. Bill, P. Korduan, L. Theuvsen, M. Morgenstern (Hrsg.)
Anforderungen an die Agrarinformatik durch Globalisierung und Klimaveränderung

P-143 Peter Liggesmeyer, Gregor Engels, Jürgen Münch, Jörg Dörr, Norman Riegel (Hrsg.)
Software Engineering 2009
Fachtagung des GI-Fachbereichs Softwaretechnik

P-144 Johann-Christoph Freytag, Thomas Ruf, Wolfgang Lehner, Gottfried Vossen (Hrsg.)
Datenbanksysteme in Business, Technologie und Web (BTW)

P-145 Knut Hinkelmann, Holger Wache (Eds.)
WM2009: 5th Conference on Professional Knowledge Management

P-146 Markus Bick, Martin Breunig, Hagen Höpfner (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Entwicklung, Implementierung und Anwendung
4. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2009)

P-147 Witold Abramowicz, Leszek Maciaszek, Ryszard Kowalczyk, Andreas Speck (Eds.)
Business Process, Services Computing and Intelligent Service Management
BPSC 2009 · ISM 2009 · YRW-MBP 2009

P-148 Christian Erfurth, Gerald Eichler, Volkmar Schau (Eds.)
9th International Conference on Innovative Internet Community Systems
I^{2}CS 2009

P-149 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
2. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-150 Jürgen Münch, Peter Liggesmeyer (Hrsg.)
Software Engineering
2009 - Workshopband

P-151 Armin Heinzl, Peter Dadam, Stefan Kirn, Peter Lockemann (Eds.)
PRIMIUM
Process Innovation for Enterprise Software

P-152 Jan Mendling, Stefanie Rinderle-Ma, Werner Esswein (Eds.)
Enterprise Modelling and Information Systems Architectures
Proceedings of the 3rd Int'l Workshop EMISA 2009

P-153 Andreas Schwill, Nicolas Apostolopoulos (Hrsg.)
Lernen im Digitalen Zeitalter
DeLFI 2009 – Die 7. E-Learning Fachtagung Informatik

P-154 Stefan Fischer, Erik Maehle Rüdiger Reischuk (Hrsg.)
INFORMATIK 2009
Im Focus das Leben

P-155 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2009:
Biometrics and Electronic Signatures
Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-156 Bernhard Koerber (Hrsg.)
Zukunft braucht Herkunft
25 Jahre »INFOS – Informatik und Schule«

P-157 Ivo Grosse, Steffen Neumann, Stefan Posch, Falk Schreiber, Peter Stadler (Eds.)
German Conference on Bioinformatics 2009

P-158 W. Claupein, L. Theuvsen, A. Kämpf, M. Morgenstern (Hrsg.)
Precision Agriculture Reloaded – Informationsgestützte Landwirtschaft

P-159 Gregor Engels, Markus Luckey, Wilhelm Schäfer (Hrsg.)
Software Engineering 2010

P-160 Gregor Engels, Markus Luckey, Alexander Pretschner, Ralf Reussner (Hrsg.)
Software Engineering 2010 – Workshopband
(inkl. Doktorandensymposium)

P-161 Gregor Engels, Dimitris Karagiannis Heinrich C. Mayr (Hrsg.)
Modellierung 2010

P-162 Maria A. Wimmer, Uwe Brinkhoff, Siegfried Kaiser, Dagmar Lück-Schneider, Erich Schweighofer, Andreas Wiebe (Hrsg.)
Vernetzte IT für einen effektiven Staat
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2010

P-163 Markus Bick, Stefan Eulgem, Elgar Fleisch, J. Felix Hampe, Birgitta König-Ries, Franz Lehner, Key Pousttchi, Kai Rannenberg (Hrsg.)
Mobile und Ubiquitäre Informationssysteme
Technologien, Anwendungen und Dienste zur Unterstützung von mobiler Kollaboration

P-164 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2010: Biometrics and Electronic Signatures Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-165 Gerald Eichler, Peter Kropf, Ulrike Lechner, Phayung Meesad, Herwig Unger (Eds.)
10th International Conference on Innovative Internet Community Systems (I^2CS) – Jubilee Edition 2010 –

P-166 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
3. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-167 Robert Krimmer, Rüdiger Grimm (Eds.)
4th International Conference on Electronic Voting 2010
co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-168 Ira Diethelm, Christina Dörge, Claudia Hildebrandt, Carsten Schulte (Hrsg.)
Didaktik der Informatik
Möglichkeiten empirischer Forschungsmethoden und Perspektiven der Fachdidaktik

P-169 Michael Kerres, Nadine Ojstersek Ulrik Schroeder, Ulrich Hoppe (Hrsg.)
DeLFI 2010 - 8. Tagung der Fachgruppe E-Learning der Gesellschaft für Informatik e.V.

P-170 Felix C. Freiling (Hrsg.)
Sicherheit 2010
Sicherheit, Schutz und Zuverlässigkeit

P-171 Werner Esswein, Klaus Turowski, Martin Juhrisch (Hrsg.)
Modellierung betrieblicher Informationssysteme (MobIS 2010)
Modellgestütztes Management

P-172 Stefan Klink, Agnes Koschmider Marco Mevius, Andreas Oberweis (Hrsg.)
EMISA 2010
Einflussfaktoren auf die Entwicklung flexibler, integrierter Informationssysteme
Beiträge des Workshops der GI-Fachgruppe EMISA (Entwicklungsmethoden für Informationssysteme und deren Anwendung)

P-173 Dietmar Schomburg, Andreas Grote (Eds.)
German Conference on Bioinformatics 2010

P-174 Arslan Brömme, Torsten Eymann, Detlef Hühnlein, Heiko Roßnagel, Paul Schmücker (Hrsg.)
perspeGKtive 2010
Workshop „Innovative und sichere Informationstechnologie für das Gesundheitswesen von morgen“

P-175 Klaus-Peter Fähnrich, Bogdan Franczyk (Hrsg.)
INFORMATIK 2010
Service Science – Neue Perspektiven für die Informatik
Band 1

P-176 Klaus-Peter Fähnrich, Bogdan Franczyk (Hrsg.)
INFORMATIK 2010
Service Science – Neue Perspektiven für die Informatik
Band 2

P-177 Witold Abramowicz, Rainer Alt, Klaus-Peter Fähnrich, Bogdan Franczyk, Leszek A. Maciaszek (Eds.)
INFORMATIK 2010
Business Process and Service Science – Proceedings of ISSS and BPSC

P-178 Wolfram Pietsch, Benedikt Krams (Hrsg.)
Vom Projekt zum Produkt
Fachtagung des GI-Fachausschusses Management der Anwendungsentwicklung und -wartung im Fachbereich Wirtschafts-informatik (WI-MAW), Aachen, 2010

P-179 Stefan Gruner, Bernhard Rumpe (Eds.)
FM+AM`2010
Second International Workshop on Formal Methods and Agile Methods

P-180 Theo Härder, Wolfgang Lehner, Bernhard Mitschang, Harald Schöning, Holger Schwarz (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW)
14. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme“ (DBIS)

P-181 Michael Clasen, Otto Schätzel, Brigitte Theuvsen (Hrsg.)
Qualität und Effizienz durch informationsgestützte Landwirtschaft, Fokus: Moderne Weinwirtschaft

P-182 Ronald Maier (Hrsg.)
6th Conference on Professional Knowledge Management
From Knowledge to Action

P-183 Ralf Reussner, Matthias Grund, Andreas Oberweis, Walter Tichy (Hrsg.)
Software Engineering 2011
Fachtagung des GI-Fachbereichs Softwaretechnik

P-184 Ralf Reussner, Alexander Pretschner, Stefan Jähnichen (Hrsg.)
Software Engineering 2011
Workshopband
(inkl. Doktorandensymposium)

P-185 Hagen Höpfner, Günther Specht, Thomas Ritz, Christian Bunse (Hrsg.)
MMS 2011: Mobile und ubiquitäre Informationssysteme Proceedings zur 6. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2011)

P-186 Gerald Eichler, Axel Küpper, Volkmar Schau, Hacène Fouchal, Herwig Unger (Eds.)
11th International Conference on Innovative Internet Community Systems (I^2CS)

P-187 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
4. DFN-Forum Kommunikations-technologien, Beiträge der Fachtagung 20. Juni bis 21. Juni 2011 Bonn

P-188 Holger Rohland, Andrea Kienle, Steffen Friedrich (Hrsg.)
DeLFI 2011 – Die 9. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. 5.–8. September 2011, Dresden

P-189 Thomas, Marco (Hrsg.)
Informatik in Bildung und Beruf
INFOS 2011
14. GI-Fachtagung Informatik und Schule

P-190 Markus Nüttgens, Oliver Thomas, Barbara Weber (Eds.)
Enterprise Modelling and Information Systems Architectures (EMISA 2011)

P-191 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2011
International Conference of the Biometrics Special Interest Group

P-192 Hans-Ulrich Heiß, Peter Pepper, Holger Schlingloff, Jörg Schneider (Hrsg.)
INFORMATIK 2011
Informatik schafft Communities

P-193 Wolfgang Lehner, Gunther Piller (Hrsg.)
IMDM 2011

P-194 M. Clasen, G. Fröhlich, H. Bernhardt, K. Hildebrand, B. Theuvsen (Hrsg.)
Informationstechnologie für eine nachhaltige Landbewirtschaftung
Fokus Forstwirtschaft

P-195 Neeraj Suri, Michael Waidner (Hrsg.)
Sicherheit 2012
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 6. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)

P-196 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2012
Proceedings of the 11th International Conference of the Biometrics Special Interest Group

P-197 Jörn von Lucke, Christian P. Geiger, Siegfried Kaiser, Erich Schweighofer, Maria A. Wimmer (Hrsg.)
Auf dem Weg zu einer offenen, smarten und vernetzten Verwaltungskultur
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2012

P-198 Stefan Jähnichen, Axel Küpper, Sahin Albayrak (Hrsg.)
Software Engineering 2012
Fachtagung des GI-Fachbereichs Softwaretechnik

P-199 Stefan Jähnichen, Bernhard Rumpe, Holger Schlingloff (Hrsg.)
Software Engineering 2012
Workshopband

P-200 Gero Mühl, Jan Richling, Andreas Herkersdorf (Hrsg.)
ARCS 2012 Workshops

P-201 Elmar J. Sinz Andy Schürr (Hrsg.)
Modellierung 2012

P-202 Andrea Back, Markus Bick, Martin Breunig, Key Pousttchi, Frédéric Thiesse (Hrsg.)
MMS 2012:Mobile und Ubiquitäre Informationssysteme

P-203 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
5. DFN-Forum Kommunikations-technologien
Beiträge der Fachtagung

P-204 Gerald Eichler, Leendert W. M. Wienhofen, Anders Kofod-Petersen, Herwig Unger (Eds.)
12th International Conference on Innovative Internet Community Systems (I2CS 2012)

P-205 Manuel J. Kripp, Melanie Volkamer, Rüdiger Grimm (Eds.)
5th International Conference on Electronic Voting 2012 (EVOTE2012)
Co-organized by the Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-206 Stefanie Rinderle-Ma, Mathias Weske (Hrsg.)
EMISA 2012
Der Mensch im Zentrum der Modellierung

P-207 Jörg Desel, Jörg M. Haake, Christian Spannagel (Hrsg.)
DeLFI 2012: Die 10. e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V.
24.–26. September 2012

P-208 Ursula Goltz, Marcus Magnor, Hans-Jürgen Appelrath, Herbert Matthies, Wolf-Tilo Balke, Lars Wolf (Hrsg.) INFORMATIK 2012

P-209 Hans Brandt-Pook, André Fleer, Thorsten Spitta, Malte Wattenberg (Hrsg.) Nachhaltiges Software Management

P-210 Erhard Plödereder, Peter Dencker, Herbert Klenk, Hubert B. Keller, Silke Spitzer (Hrsg.) Automotive – Safety & Security 2012 Sicherheit und Zuverlässigkeit für automobile Informationstechnik

P-211 M. Clasen, K. C. Kersebaum, A. Meyer-Aurich, B. Theuvsen (Hrsg.) Massendatenmanagement in der Agrar- und Ernährungswirtschaft Erhebung - Verarbeitung - Nutzung Referate der 33. GIL-Jahrestagung 20. – 21. Februar 2013, Potsdam

P-212 Arslan Brömme, Christoph Busch (Eds.) BIOSIG 2013 Proceedings of the 12th International Conference of the Biometrics Special Interest Group 04.–06. September 2013 Darmstadt, Germany

P-213 Stefan Kowalewski, Bernhard Rumpe (Hrsg.) Software Engineering 2013 Fachtagung des GI-Fachbereichs Softwaretechnik

P-214 Volker Markl, Gunter Saake, Kai-Uwe Sattler, Gregor Hackenbroich, Bernhard Mit schang, Theo Härder, Veit Köppen (Hrsg.) Datenbanksysteme für Business, Technologie und Web (BTW) 2013 13. – 15. März 2013, Magdeburg

P-215 Stefan Wagner, Horst Lichter (Hrsg.) Software Engineering 2013 Workshopband (inkl. Doktorandensymposium) 26. Februar – 1. März 2013, Aachen

P-216 Gunter Saake, Andreas Henrich, Wolfgang Lehner, Thomas Neumann, Veit Köppen (Hrsg.) Datenbanksysteme für Business, Technologie und Web (BTW) 2013 – Workshopband 11. – 12. März 2013, Magdeburg

P-217 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.) 6. DFN-Forum Kommunikations-technologien Beiträge der Fachtagung 03.–04. Juni 2013, Erlangen

P-218 Andreas Breiter, Christoph Rensing (Hrsg.) DeLFI 2013: Die 11 e-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (GI) 8. – 11. September 2013, Bremen

P-219 Norbert Breier, Peer Stechert, Thomas Wilke (Hrsg.) Informatik erweitert Horizonte INFOS 2013 15. GI-Fachtagung Informatik und Schule 26. – 28. September 2013

P-220 Matthias Horbach (Hrsg.) INFORMATIK 2013 Informatik angepasst an Mensch, Organisation und Umwelt 16. – 20. September 2013, Koblenz

P-221 Maria A. Wimmer, Marijn Janssen, Ann Macintosh, Hans Jochen Scholl, Efthimios Tambouris (Eds.) Electronic Government and Electronic Participation Joint Proceedings of Ongoing Research of IFIP EGOV and IFIP ePart 2013 16. – 19. September 2013, Koblenz

P-222 Reinhard Jung, Manfred Reichert (Eds.) Enterprise Modelling and Information Systems Architectures (EMISA 2013) St. Gallen, Switzerland September 5. – 6. 2013

P-223 Detlef Hühnlein, Heiko Roßnagel (Hrsg.) Open Identity Summit 2013 10. – 11. September 2013 Kloster Banz, Germany

P-224 Eckhart Hanser, Martin Mikusz, Masud Fazal-Baqaie (Hrsg.) Vorgehensmodelle 2013 Vorgehensmodelle – Anspruch und Wirklichkeit 20. Tagung der Fachgruppe Vorgehensmodelle im Fachgebiet Wirtschaftsinformatik (WI-VM) der Gesellschaft für Informatik e.V. Lörrach, 2013

P-225 Hans-Georg Fill, Dimitris Karagiannis, Ulrich Reimer (Hrsg.) Modellierung 2014 19. – 21. März 2014, Wien

P-226 M. Clasen, M. Hamer, S. Lehnert, B. Petersen, B. Theuvsen (Hrsg.) IT-Standards in der Agrar- und Ernährungswirtschaft Fokus: Risiko- und Krisenmanagement Referate der 34. GIL-Jahrestagung 24. – 25. Februar 2014, Bonn

P-227 Wilhelm Hasselbring,
Nils Christian Ehmke (Hrsg.)
Software Engineering 2014
Fachtagung des GI-Fachbereichs Softwaretechnik
25. – 28. Februar 2014
Kiel, Deutschland

P-228 Stefan Katzenbeisser, Volkmar Lotz, Edgar Weippl (Hrsg.)
Sicherheit 2014
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 7. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
19. – 21. März 2014, Wien

P-229 Dagmar Lück-Schneider, Thomas Gordon, Siegfried Kaiser, Jörn von Lucke,Erich Schweighofer, Maria A.Wimmer, Martin G. Löhe (Hrsg.)
Gemeinsam Electronic Government ziel(gruppen)gerecht gestalten und organisieren
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI)
2014, 20.-21. März 2014 in Berlin

P-230 Arslan Brömme, Christoph Busch (Eds.)
BIOSIG 2014
Proceedings of the 13th International Conference of the Biometrics Special Interest Group
10. – 12. September 2014 in Darmstadt, Germany

P-231 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
7. DFN-Forum Kommunikationstechnologien
16. – 17. Juni 2014
Fulda

P-232 E. Plödereder, L. Grunske, E. Schneider, D. Ull (Hrsg.)
INFORMATIK 2014
Big Data – Komplexität meistern
22. – 26. September 2014
Stuttgart

P-233 Stephan Trahasch, Rolf Plötzner, Gerhard Schneider, Claudia Gayer, Daniel Sassiat, Nicole Wöhrle (Hrsg.)
DeLFI 2014 – Die 12. e-Learning Fachtagung Informatik
der Gesellschaft für Informatik e.V.
15. – 17. September 2014
Freiburg

P-234 Fernand Feltz, Bela Mutschler, Benoît Otjacques (Eds.)
Enterprise Modelling and Information Systems Architectures
(EMISA 2014)
Luxembourg, September 25-26, 2014

P-235 Robert Giegerich, Ralf Hofestädt, Tim W. Nattkemper (Eds.)
German Conference on Bioinformatics 2014
September 28 – October 1
Bielefeld, Germany

P-236 Martin Engstler, Eckhart Hanser, Martin Mikusz, Georg Herzwurm (Hrsg.)
Projektmanagement und Vorgehensmodelle 2014
Soziale Aspekte und Standardisierung
Gemeinsame Tagung der Fachgruppen Projektmanagement (WI-PM) und Vorgehensmodelle (WI-VM) im Fachgebiet Wirtschaftsinformatik der Gesellschaft für Informatik e.V., Stuttgart 2014

P-237 Detlef Hühnlein, Heiko Roßnagel (Hrsg.)
Open Identity Summit 2014
4.–6. November 2014
Stuttgart, Germany

P-238 Arno Ruckelshausen, Hans-Peter Schwarz, Brigitte Theuvsen (Hrsg.)
Informatik in der Land-, Forst- und Ernährungswirtschaft
Referate der 35. GIL-Jahrestagung
23. – 24. Februar 2015, Geisenheim

P-239 Uwe Aßmann, Birgit Demuth, Thorsten Spitta, Georg Püschel, Ronny Kaiser (Hrsg.)
Software Engineering & Management 2015
17.-20. März 2015, Dresden

P-240 Herbert Klenk, Hubert B. Keller, Erhard Plödereder, Peter Dencker (Hrsg.)
Automotive – Safety & Security 2015
Sicherheit und Zuverlässigkeit für automobile Informationstechnik
21.–22. April 2015, Stuttgart

P-241 Thomas Seidl, Norbert Ritter, Harald Schöning, Kai-Uwe Sattler, Theo Härder, Steffen Friedrich, Wolfram Wingerath (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2015)
04. – 06. März 2015, Hamburg

P-242 Norbert Ritter, Andreas Henrich, Wolfgang Lehner, Andreas Thor, Steffen Friedrich, Wolfram Wingerath (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2015) – Workshopband
02. – 03. März 2015, Hamburg

P-243 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
8. DFN-Forum Kommunikationstechnologien
06.–09. Juni 2015, Lübeck

P-244 Alfred Zimmermann, Alexander Rossmann (Eds.)
Digital Enterprise Computing (DEC 2015)
Böblingen, Germany June 25-26, 2015

P-245 Arslan Brömme, Christoph Busch , Christian Rathgeb, Andreas Uhl (Eds.)
BIOSIG 2015
Proceedings of the 14th International Conference of the Biometrics Special Interest Group
09.–11. September 2015
Darmstadt, Germany

P-246 Douglas W. Cunningham, Petra Hofstedt, Klaus Meer, Ingo Schmitt (Hrsg.)
INFORMATIK 2015
28.9.-2.10. 2015, Cottbus

P-247 Hans Pongratz, Reinhard Keil (Hrsg.)
DeLFI 2015 – Die 13. E-Learning Fachtagung Informatik der Gesellschaft für Informatik e.V. (GI)
1.–4. September 2015
München

P-248 Jens Kolb, Henrik Leopold, Jan Mendling (Eds.)
Enterprise Modelling and Information Systems Architectures
Proceedings of the 6th Int. Workshop on Enterprise Modelling and Information Systems Architectures, Innsbruck, Austria
September 3-4, 2015

P-249 Jens Gallenbacher (Hrsg.)
Informatik allgemeinbildend begreifen
INFOS 2015 16. GI-Fachtagung Informatik und Schule
20.–23. September 2015

P-250 Martin Engstler, Masud Fazal-Baqaie, Eckhart Hanser, Martin Mikusz, Alexander Volland (Hrsg.)
Projektmanagement und Vorgehensmodelle 2015
Hybride Projektstrukturen erfolgreich umsetzen
Gemeinsame Tagung der Fachgruppen Projektmanagement (WI-PM) und Vorgehensmodelle (WI-VM) im Fachgebiet Wirtschaftsinformatik der Gesellschaft für Informatik e.V., Elmshorn 2015

P-251 Detlef Hühnlein, Heiko Roßnagel, Raik Kuhlisch, Jan Ziesing (Eds.)
Open Identity Summit 2015
10.–11. November 2015
Berlin, Germany

P-252 Jens Knoop, Uwe Zdun (Hrsg.)
Software Engineering 2016
Fachtagung des GI-Fachbereichs Softwaretechnik
23.–26. Februar 2016, Wien

P-253 A. Ruckelshausen, A. Meyer-Aurich, T. Rath, G. Recke, B. Theuvsen (Hrsg.)
Informatik in der Land-, Forst- und Ernährungswirtschaft
Fokus: Intelligente Systeme – Stand der Technik und neue Möglichkeiten
Referate der 36. GIL-Jahrestagung
22.-23. Februar 2016, Osnabrück

P-254 Andreas Oberweis, Ralf Reussner (Hrsg.)
Modellierung 2016
2.–4. März 2016, Karlsruhe

P-255 Stefanie Betz, Ulrich Reimer (Hrsg.)
Modellierung 2016 Workshopband
2.–4. März 2016, Karlsruhe

P-256 Michael Meier, Delphine Reinhardt, Steffen Wendzel (Hrsg.)
Sicherheit 2016
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 8. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
5.–7. April 2016, Bonn

P-257 Paul Müller, Bernhard Neumair, Helmut Reiser, Gabi Dreo Rodosek (Hrsg.)
9. DFN-Forum Kommunikationstechnologien
31. Mai – 01. Juni 2016, Rostock

P-258 Dieter Hertweck, Christian Decker (Eds.)
Digital Enterprise Computing (DEC 2016)
14.–15. Juni 2016, Böblingen

P-259 Heinrich C. Mayr, Martin Pinzger (Hrsg.)
INFORMATIK 2016
26.–30. September 2016, Klagenfurt

P-260 Arslan Brömme, Christoph Busch, Christian Rathgeb, Andreas Uhl (Eds.)
BIOSIG 2016
Proceedings of the 15th International Conference of the Biometrics Special Interest Group
21.–23. September 2016, Darmstadt

P-261 Detlef Rätz, Michael Breidung, Dagmar Lück-Schneider, Siegfried Kaiser, Erich Schweighofer (Hrsg.)
Digitale Transformation: Methoden, Kompetenzen und Technologien für die Verwaltung
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2016
22.–23. September 2016, Dresden

P-262 Ulrike Lucke, Andreas Schwill, Raphael Zender (Hrsg.)
DeLFI 2016 – Die 14. E-Learning Fachtagung Informatik
der Gesellschaft für Informatik e.V. (GI)
11.–14. September 2016, Potsdam

P-263 Martin Engstler, Masud Fazal-Baqaie, Eckhart Hanser, Oliver Linssen, Martin Mikusz, Alexander Volland (Hrsg.)
Projektmanagement und Vorgehensmodelle 2016
Arbeiten in hybriden Projekten: Das Sowohl-als-auch von Stabilität und Dynamik
Gemeinsame Tagung der Fachgruppen Projektmanagement (WI-PM) und Vorgehensmodelle (WI-VM) im Fachgebiet Wirtschaftsinformatik
der Gesellschaft für Informatik e.V., Paderborn 2016

P-264 Detlef Hühnlein, Heiko Roßnagel, Christian H. Schunck, Maurizio Talamo (Eds.)
Open Identity Summit 2016
der Gesellschaft für Informatik e.V. (GI)
13.–14. October 2016, Rome, Italy

P-265 Bernhard Mitschang, Daniela Nicklas,Frank Leymann, Harald Schöning, Melanie Herschel, Jens Teubner, Theo Härder, Oliver Kopp, Matthias Wieland (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2017)
6.–10. März 2017, Stuttgart

P-266 Bernhard Mitschang, Norbert Ritter, Holger Schwarz, Meike Klettke, Andreas Thor, Oliver Kopp, Matthias Wieland (Hrsg.)
Datenbanksysteme für Business, Technologie und Web (BTW 2017)
Workshopband
6.–7. März 2017, Stuttgart

P-267 Jan Jürjens, Kurt Schneider (Hrsg.)
Software Engineering 2017
21.–24. Februar 2017
Hannover

P-268 A. Ruckelshausen, A. Meyer-Aurich, W. Lentz, B. Theuvsen (Hrsg.)
Informatik in der Land-, Forst- und Ernährungswirtschaft
Fokus: Digitale Transformation – Wege in eine zukunftsfähige Landwirtschaft
Referate der 37. GIL-Jahrestagung
06.–07. März 2017, Dresden

P-269 Peter Dencker, Herbert Klenk, Hubert Keller, Erhard Plödereder (Hrsg.)
Automotive – Safety & Security 2017
30.–31. Mai 2017
Stuttgart

The titles can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14 · D-53117 Bonn
Fax: +49 (0)228/9898222
E-Mail: druckverlag@koellen.de