Universität Regensburg

Fakultät für Sprach-, Literatur- und Kulturwissenschaften
Institut für Information und Medien, Sprache und Kultur (I:IMSK)
Lehrstuhl für Medieninformatik

Sichere Werkzeugketten und werkzeugunterstütztes Assessment

Verwendung komplexer Werkzeugketten für die Entwicklung und Bewertung sicherheitsrelevanter Software

Inaugural-Dissertation zur Erlangung der Doktorwürde
der Fakultät für Sprach-, Literatur- und Kulturwissenschaften
der Universität Regensburg

Vorgelegt von

Andreas Bärwald

aus Wassertrüdingen

2018

Erstgutachter: Prof. Dr. phil. Christian Wolff

Zweitgutachter: Prof. Dr. rer. nat. Jürgen Mottok

Abstract

Eine stark wachsende Zahl eingebetteter elektronischer Systeme implementiert sicherheitsrelevante Funktionen. Diese Funktionen werden zunehmend in Software implementiert. Die einschlägigen Normen zur Funktionalen Sicherheit erfordern den Einsatz von „vertrauenswürdigen“ Softwareentwicklungswerkzeugen und Werkzeugketten.
Diese Arbeit leitet einen generischen, projektunabhängigen Ansatz für die Qualifizierung von Softwarewerkzeugen her, die bei der Entwicklung sicherheitsrelevanter Systeme eingesetzt werden. Auf Basis dieser Methodik wird in einem zweiten Schritt ein Vorgehen entwickelt, dass es ermöglicht, die Bewertung der Funktionalen Sicherheit (Safety Assessment) werkzeugunterstützt durchzuführen. Dies führt zu einer signifikanten Effizienzsteigerung bei der Begutachtung sicherheitstechnischer Systeme.

Abstract

A growing number of embedded electronic systems are implementing safety-related functions. These functions are increasingly being implemented in software. All relevant standards for Functional Safety require the use of "trustworthy" software development tools and tool chains.

This thesis introduces a generic, project-independent approach for the qualification of software tools which are used in the development of safety-related systems. On the basis of this methodology, a procedure is developed that makes it possible to carry out the assessment of Functional Safety tool-based. This leads to a significant increase in efficiency when assessing safety-related systems.

Danksagungen

Diese Arbeit entstand berufsbegleitend in einem industriellen Umfeld im Bereich „Sicherheit und Elektronik“ bei TÜV SÜD. Die erarbeiteten Methoden wurden in zahlreichen Industrieprojekten zusammen mit Kunden erfolgreich angewendet und verfeinert. Ich bin dankbar, meine Ideen in realen Projekten mit realen Kunden leben zu dürfen.

Im Besonderen möchte ich meinem Doktorvater Prof. Dr. phil. Christian Wolff für die Unterstützung und den Rückhalt in allen Belangen danken. Ohne Prof. Dr. rer. nat. Jürgen Mottok wäre es nie zu diesem Vorhaben gekommen. Ich danke ihm für die Motivation, gemeinsame Veröffentlichungen, Feedback und tatkräftige Unterstützung.

Weiterer besonderer Dank gilt meinen Eltern, Prof. Dr.-Ing. habil. Werner Bärwald und Christine Bärwald, die mich immer unterstützt und gefördert haben. Meinem Vater danke ich für das konstruktive Feedback zu dieser Arbeit, dass er mit seinem umfangreichen akademischen Erfahrungsschatz gegeben hat.

Ein besonders herzlicher Dank gilt meinem langjährigen Freund Dipl. Inf. (Univ.) Andreas Zschocke, der dieses Vorhaben sehr intensiv begleitet hat. Ich danke ihm für die zahlreichen nächtlichen Fachdiskussionen, sein überaus hilfreiches Feedback und seine moralische Unterstützung.

Des Weiteren möchte ich mich bei meinen Kollegen bei TÜV SÜD bedanken: Dipl. Inf. (Univ.) Doris Wild, Dr. rer. nat. Bernd Spanfelner, Dipl. Ing. (FH) Nicole Pappler, Dipl. Ing. (Univ.) Manfred Inderst und Dr.-Ing. Detlev Richter. Ich lernte viel aus den gemeinsamen Projekten und danke für viele, sehr gute fachliche Gespräche, die stets Anregungen für den weiteren Fortschritt der Arbeit lieferten.

Prof. Dr.-Ing. Martin Hobelsberger danke ich für sein fachliches Feedback über Jahre und die inhaltlichen und motivierenden Impulse, die er stets gesetzt hat. Weiterer Dank gilt den Kollegen, mit denen ich im Rahmen von Projekten Veröffentlichungen publizierte. Besonderer Dank gilt dabei Christoph Bräuchle, Michael Beine und Harald Hauff.

Inhalt

Abbildungen

1 Einleitung

"… Menschen, Umwelt und technologische Entwicklungen für eine lebenswerte Zukunft in Einklang zu bringen."[1]

Technologischer Fortschritt bedeutet vor allem Chancen. Gleichwohl können durch neuartige technische Systeme auch zusätzliche Gefährdungen für Mensch und Umwelt entstehen. Diese Gefährdungen gilt es zu erkennen, zu analysieren und zu beherrschen. Genau das ist das primäre Ziel der Funktionalen Sicherheit elektrischer und elektronischer Systeme.

Diese Arbeit beschäftigt sich mit modernen Softwareentwicklungswerkzeugen und Werkzeugketten, die bei der Entwicklung sicherheitsgerichteter Systeme üblicherweise eingesetzt werden, und definiert Anforderungen an eine hinreichende Fehlerfreiheit[2] sowie die korrekte und zuverlässige Arbeitsweise dieser Werkzeuge, einen Ansatz für eine generische Werkzeugqualifizierung. Ausgehend von der Hypothese, dass Softwareentwicklungswerkzeuge für den Einsatz bei der Entwicklung sicherheitsgerichteter Systeme qualifiziert werden können, wird im zweiten Teil der Arbeit ein Konzept erarbeitet, indem die qualifizierte Werkzeugkette auch für die Sicherheitsbeurteilung (Assessment) verwendet wird.

In diesem einleitenden Kapitel werden die Motivation und die wissenschaftlichen Ziele dieser Arbeit beschrieben.

1.1 Motivation

Eine stark wachsende Zahl eingebetteter elektronischer Systeme implementiert sicherheitsrelevante Funktionen. Dabei nimmt die Komplexität dieser Systeme technologiebedingt stetig zu. Das ist auch auf eine steigende Vernetzung und den damit verbundenen Informationsaustausch und die funktionale Abhängigkeit von Subsystemen zurückzuführen. Dieser Trend ist industrieübergreifend zu beobachteten. Stellvertretend seien

[1] Mission Statement der TÜV SÜD AG aus dem Geschäftsbericht 2015 (TÜV SÜD AG, 2015)

[2] Eine hinreichende Fehlerfreiheit von Softwarewerkzeugen wird im Rahmen dieser Arbeit in den Kontext der Vertrauenswürdigkeit in eben diese Werkzeuge gesetzt.

hier die Automatisierungs- und Prozessleittechnik, Bahnanwendungen, Luftfahrt, Medizintechnik und vor allem stark wachsend, die Automotive-Anwendungen genannt. Megatrends, wie Industrie 4.0, Internet der Dinge (IoT), automatisiertes Fahren und Elektromobilität, aber auch Technologiesprünge, wie z.B. der zu erwartende Einsatz von Multicore-Technologie, können als Katalysator wirken. Darüber hinaus wird die technologische Entwicklung der „digitalen Transformation" in den nächsten Jahren und Jahrzehnten wesentlich durch Künstliche Intelligenz getrieben werden (Henning, 2016).

„Internet of Things & Services, M2M, or Cyber Physical Systems are much more than just buzzwords for the outlook connecting 50 billion devices in 2015."[3]

Bei der zunehmenden Vernetzung von Systemen wird das Thema Interoperabilität eine entscheidende Rolle spielen. Das wird die technologische Entwicklung industrieübergreifend in den kommenden Jahren signifikant beeinflussen (Miller & Lin, 2016). Im Energiesektor wird Interoperabiliät bereits heute bei Smart Grid Applikationen (Bärwald & Pfisterer, 2017) und im Bereich der Ladeinfrastruktur von Elektrofahrzeugen umgesetzt.
Ein besonders interessanter Aspekt all dieser technologischen Entwicklungen ist, dass die Zunahme der Komplexität meist maßgeblich in Software erfolgt, da innovative Funktionen überwiegend durch neuartige Softwarefunktionen realisiert werden. Dies ist besonders eindrucksvoll bei den modernen Automobilanwendungen zu beobachten. Systeme, wie elektro-mechanische Bremsen (EPB) und Lenkungen (EPS), Motorsteuerungen, Automatikgetriebe und Systeme, die in die Fahrdynamik des Fahrzeuges eingreifen, bestehen in den wesentlichen Teilen ihres funktionalen Verhaltens aus Software. Alle relevanten deutschen Fahrzeughersteller (OEM) setzen momentan einen maßgeblichen Entwicklungsfokus auf das Thema erweiterte Fahrerassistenzsysteme (ADAS) als Meilenstein hin zum automatisierten bzw. autonomen Fahren. Auch hier handelt es sich um Funktionen, die das Fahrzeugverhalten in der Längs- und Querdynamik beeinflussen und deren Fehlfunktion zu einer zusätzlichen technologiebedingten Gefährdung führen können. Daraus folgt, dass die Entwicklungsansätze, das Entwicklungsvorgehen

[3] Zitat von Dr. Stefan Ferber, Robert Bosch GmbH, in 2011 aus (Henning, 2016)

und daraus folgend auch die Verfahren für die Produktionsfreigabe solcher (Software-) Systeme, also das Safety-Assessment, an diese Herausforderungen angepasst werden müssen. Das beinhaltet auch die Verwendung von Softwarewerkzeugen, die zur Entwicklung solcher Systeme eingesetzt werden.

Die Folgen von fehlerhafter Software können sehr weitreichend sein. Der Verlust der Ariane 5 im Jahr 1996 z.B. ist maßgeblich auf die Verwendung von bewährter Software aus dem Ariane 4 Programm zurückzuführen, die nicht adäquat für den Einsatz in der neuen Systemumgebung und damit veränderten physikalischen Eigenschaften angepasst wurde. In Summe führte ein fehlerhafter Type-Cast und ein unvollständiger Testansatz bei der Wiederverwendung von Software-Bausteinen (Re-Use) zum Totalverlust eines komplexen technischen Systems, was zu großem wirtschaftlichen Schaden (ca. 500 Millionen US Dollar) führte, in anderen Fällen aber auch zu Gefahr für Leib und Leben oder zu massiven Umweltschäden führen kann. Der folgende Ausschnitt aus dem Quellcode zeigt die Problematik in diesem Beispiel.

```
01  declare
02    vertical_veloc_sensor: float;
03    horizontal_veloc_sensor: float;
04    vertical_veloc_bias: integer;
05    horizontal_veloc_bias: integer;
      ...
10  begin
11    declare
12      pragma suppress(numeric_error, horizontal_veloc_bias);
13    begin
14      sensor_get(vertical_veloc_sensor);
15      sensor_get(horizontal_veloc_sensor);
16      vertical_veloc_bias := integer(vertical_veloc_sensor);
17      horizontal_veloc_bias := integer(horizontal_veloc_sensor);
        ...
20    exception
21      when numeric_error => calculate_vertical_veloc();
22      when others => use_irs1();
23    end;
24  end irs2;
```

Abbildung 1: Fehlerhafte Stelle im Ariane 5 Ada-Quelltext nach (Riedl, 2002)

Die unabhängige Untersuchungskommission fand nach kurzer Zeit heraus, dass ein Konvertierungsfehler (siehe Zeilen 16 und 17 im Code-Beispiel) letztlich zur Explosion führte. Einige Sekunden nach dem Start begann das Trägheitskontrollsystem widersprüchliche Daten an den Bordcomputer zu senden, was letztendlich zur automatischen Selbstzerstörung der Rakete führte, da der Druck auf die Booster wegen fehlender Kurskorrekturen so groß wurde, dass sie abzubrechen drohten.

Eigentlich sollte das System die Beschleunigungswerte der Rakete als 64-Bit-Gleitkommazahlen verarbeiten. Allerdings empfing der Bordcomputer aber nur noch Fehlermeldungen (Exception-Handling in den Zeilen 20ff) vom Trägheitskontrollsystem, die final zur Abschaltung des Systems führten. Der Grund dafür war der explizite Type-Cast, der 64-Bit-Gleitkommazahlen in 16-Bit-Integer-Zahlen konvertierte (Zeilen 16 und 17), welche allerdings nur Zahlen zwischen -32768 und +32767 speichern können. Die Entwickler gingen dabei explizit davon aus, dass die Grenzen des Integer-Wertebereiches bei diesen Operationen nicht erreicht werden und begründeten dies mit der Verwendung dieses Softwareteiles in der Ariane 4 und der daraus resultierenden Betriebserfahrung (proofen in use). Dabei wurde nicht berücksichtigt, dass die neuentwickelte Ariane 5 sich wesentlich schneller bewegte. Sekundenbruchteile nach der Abschaltung der Trägheitskontrolle nahm ein redundantes System seine Arbeit auf. Dieses versagte ebenfalls, da die aufgespielte Software identisch mit der des Hauptsystems war. Es handelte sich also um einen systematischen Fehler, der durch den Einsatz von homogener Redundanz zur Laufzeit nicht erkannt und beherrscht werden konnte.

Dieses Beispiel zeigt die Komplexität moderner Softwaresysteme und den Einfluss kleinster Implementierungsfehler auf das Systemverhalten bis hin zum Totalverlust des technischen Systems im Worst-Case, was zu erheblichen finanziellen Schäden sowie massivem Reputationsverlust führen kann. Es geht also im Besondern um die Qualität der Entwicklung solcher Systeme. Denn eine hohe Entwicklungsqualität senkt die Wahrscheinlichkeit, dass Fehler im Zielsystem enthalten sind. Das schließt die Systemarchitektur, den Verifikationsansatz aber auch die Entwicklungsprozesse bis hin zur Verwendung geeigneter Softwareentwicklungswerkzeuge und der Werkzeugkette mit ein.

Gerade die Automobilindustrie ist von bisweilen sehr kostspieligen Rückrufaktionen betroffen. Das hat Bährle in seiner Masterarbeit (Bährle, 2016) am Beispiel der Automobilindustrie in den USA untersucht (siehe Abbildung 2). Dabei ist zu erkennen, wie die

Anzahl der Rückrufe aufgrund von Softwarefehlern seit 2012 massiv zunimmt. Allein in den USA waren in den Jahren 2014 und 2015 jeweils fast 6 Millionen Fahrzeuge betroffen. Diese Zunahme ist nahezu dramatisch, wenngleich der Anteil von Rückrufen mit Softwarebezug die 15% Marke noch nicht übersteigt.

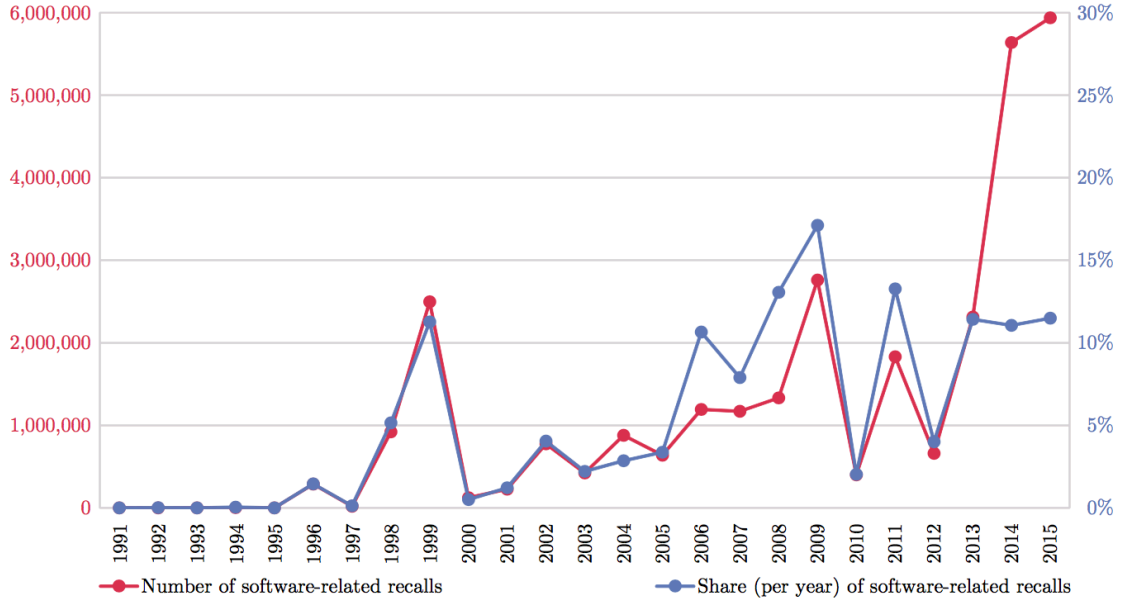

Abbildung 2: Anzahl und Verteilung von Rückrufen aufgrund von Softwarefehlern in der Automobilindustrie in den USA von 1991 bis 2015 nach (Bährle, 2016)

Bei Rückrufen spielen vor allem juristische Aspekte im Hinblick auf die Produkthaftung eine entscheidende Rolle, da die Systemhersteller im Schadensfall in der juristischen Auseinandersetzung nachweisen müssen, dass bei der Entwicklung der Systeme der Stand der Wissenschaft und Technik berücksichtigt wurde. In Deutschland ist dies im Produkthaftungsgesetz (§1 Abs. 1 ProdHaftG)[4] und im BGB (§823 Abs. 1 BGB)[5] geregelt. Es gilt die sogenannte Verkehrssicherungspflicht des Herstellers in Bezug auf Konstruktion, Fabrikation, Inverkehrbringen und Produktbeobachtung (after sales). Der Maßstab ist immer der Stand von Wissenschaft und Technik, siehe z.B. das BGH-Urteil vom 16.06.2009 (VI ZR 107/08) – „Airbag-Urteil“ (Bundesgerichtshof, 2008). Ein Indiz für die Erfüllung des Standes der Technik kann die erfolgreiche Anwendung eines international

[4] §1 Abs. 1 ProdHaftG: “Wird durch den Fehler eines Produktes jemand getötet, sein Körper oder seine Gesundheit verletzt oder eine Sache beschädigt, so ist der Hersteller des Produktes verpflichtet, dem Geschädigten den daraus entstehenden Schaden zu ersetzen.”

[5] §823 Abs. 1 BGB: “Wer vorsätzlich oder fahrlässig das Leben, den Körper, die Gesundheit, die Freiheit, das Eigentum oder rein sonstiges Recht eines anderen widerrechtlich verletzt, ist dem anderen zum Ersatz des daraus entstehenden Schadens verpflichtet.”

gültigen Standards sein (Klindt, 2012). Bei der heutigen technologischen Innovationsgeschwindigkeit stößt dieser Ansatz allerdings an seine Grenzen, da der Stand der Normung den technischen Entwicklungen zeitlich immer nachläuft. Diese Thematik wird unter Juristen bisweilen kontrovers diskutiert:

„Das ist jedenfalls bei komplexen innovativen Produkten, etwa bei Autos, eine sehr verkürzte Betrachtungsweise. Denn Autos verkaufen sich heute fast nur noch über technische Innovationen, für die es keinen gesicherten und anerkannten Kenntnisstand in breiteren Fachkreisen gibt. Die vom BGH verworfene „Branchenüblichkeit" dominiert oft über die strengeren Anforderungen an den „neuesten Stand von Wissenschaft und Technik"."[6]

Wird berücksichtigt, dass der Umfang und die Komplexität von Software im Automobil in den vergangenen 10 Jahren signifikant zugenommen haben, ist es wichtig, sich mit der Problematik „Vermeidung systematischer Fehler in der Softwareentwicklung" zu beschäftigen, im Besonderen bei sicherheitsrelevanten Funktionen. Dieses Ziel verfolgen die einschlägigen Standards zur Funktionalen Sicherheit, die in dieser Arbeit im Detail betrachtet werden. Ein zentraler Aspekt ist dabei die geeignete Verwendung von Softwareentwicklungswerkzeugen und Werkzeugketten. Deren Komplexität lässt sich eindrucksvoll am Beispiel der automatischen Codegenerierung zeigen (siehe Abbildung 3), die im Kontext modellbasierter Entwicklungsmethoden gerade im Automotive-Umfeld eine zunehmende Verbreitung findet. Der Einsatz von Softwareentwicklungswerkzeugen erfordert insbesondere bei der Entwicklung von sicherheitsrelevanten Systemen ein strukturiertes Vorgehen, um einen definierten Vertrauensgrad in die eingesetzte Werkzeugkette sicherzustellen, da fehlerhaftes Verhalten eines automatisierten Entwicklungsschrittes zu signifikanten Fehlfunktionen der Produktivsoftware und somit zu sicherheitskritischen Systemzuständen des Zielsystems führen kann.

[6] Zitat aus (Helmig, 2013)

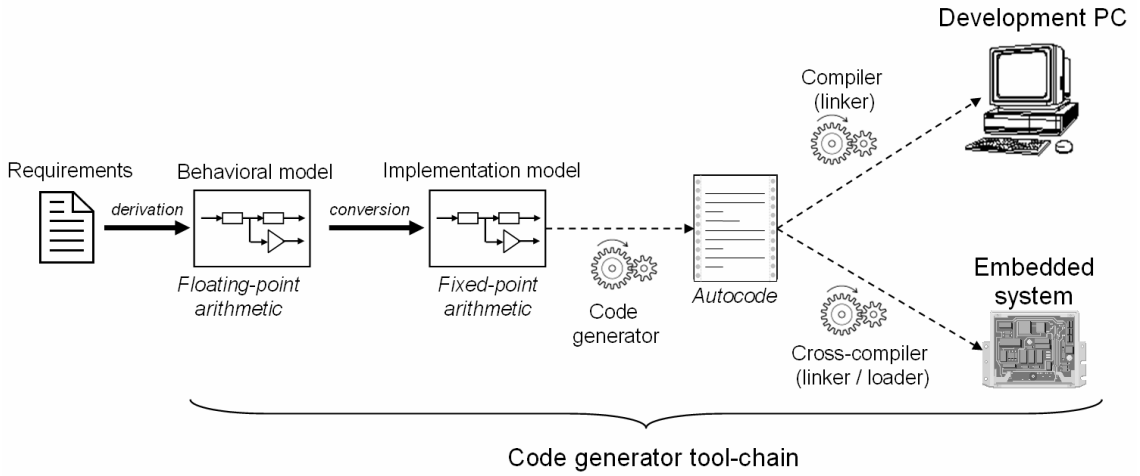

Abbildung 3: Prinzipien der modellbasierten Codegenerierung für embedded systems nach (Fey & Stürmer, 2008)

In dieser Arbeit werden u.a. auch die Anforderungen für die Verwendung von Code-Generatoren bei der Entwicklung sicherheitsgerichteter Systeme in einer Fallstudie untersucht.

1.2 Wissenschaftliche Ziele dieser Arbeit

Diese Arbeit beschäftigt sich mit zwei wesentlichen Schwerpunkten bei der Entwicklung von komplexen Softwaresystemen, die besonders im Umfeld von sicherheitsrelevanten Systemen bedeutend sind:

1. **Sichere Werkzeuge und Werkzeugketten:** Bei der Entwicklung von moderner Software wird eine bisweilen komplexe Kette von diversen Softwareentwicklungswerkzeugen verwendet. Wenn die Werkzeuge durch Fehlfunktion zusätzliche, in den Verifikationsschritten nicht erkennbare, Fehler in die Produktivsoftware einbringen, kann dies zu erheblichen Sicherheitsproblemen des Zielsystems und damit zu einer Gefährdung für Leib und Leben von Personen führen. Dieses Risiko gilt es zu minimierten. Verschiedene Sicherheitsstandards geben Hinweise, wie dies erfolgen könnte. Die vorliegende Arbeit zeigt einen generischen Ansatz zur Bewertung der Vertrauenswürdigkeit in Werkzeuge und Werkzeugketten unter Verwendung der bekannten Ansätze zur Abschätzung der Kritikalität von Werkzeugfehlern auf.

2. **Werkzeugunterstützte Sicherheitsnachweise**[7]**:** Basierend auf dem ersten Teil dieser Arbeit wird die in Punkt 1 beschriebene Verwendung von komplexen Werkzeugketten für die Softwareentwicklung als Stand der Technik angenommen. Bei der Bewertung von sicherheitsrelevanten Systemen, die unter Zuhilfenahme dieser Werkzeugketten entwickelt werden, werden die Daten meist nicht direkt aus den Entwicklungssystemen verwendet, sondern Safety-Case-Dokumentensätze aufwendig durch Export aus diesen erzeugt. Im zweiten Teil dieser Arbeit wird ein integraler Ansatz hergeleitet, der diesen Systembruch verhindert und die Safety-Assessments effizienter und präziser macht.

1.2.1 Stand der Wissenschaft und Technik

Um den Stand der Wissenschaft und Technik im Rahmen dieser Arbeit zu untersuchen, wurde fortlaufend mit folgenden Datenbanken für wissenschaftliche Veröffentlichungen recherchiert (Stand Februar 2018):

- IEEE Xplore Digital Libary[8],
- ACM Digital Libary[9] und
- Google Scholar[10].

Dabei wurden für die Recherche folgende Schlagwortkombinationen verwendet:

Tabelle 1: Schlagwortkombinationen zur Recherche in Datenbanken für wissenschaftliche Publikationen

(Functional) Safety	Assessment	-
		Software
		Tool
		Tool based
		Plug-in
	Safety case	-
		Tool
		Tool Based

[7] Tool based assessment
[8] http://ieeexplore.ieee.org/
[9] http://dl.acm.org
[10] http://scholar.google.de/

	Qualification, Qualify	-
		Software
		Tool
		Safety Manual
	Reference Workflow	-
	Security	-
	Process	-
	Certification	-
	SPICE	-
	Open Source	-

Zu Beginn der Recherche im Jahr 2010 gab es kaum Veröffentlichungen, die für die Zielsetzungen dieser Arbeit - die generische, projektunabhängige und standardübergreifende Qualifizierung von Software-Werkzeugen sowie die Verwendung einer Werkzeugkette für den Sicherheitsnachweis - wissenschaftliche Beiträge oder technische Lösungen thematisierten.

In (S. Schneider & Mai, 2009) wird ein Ansatz beschrieben, wie eine Werkzeugkette beim Automobilhersteller (OEM) BMW bestehend aus den Einzelwerkzeugen dSPACE TargetLink (Codegenerator), WindRiver C Cross-Compiler und WindRiver Cross-Linker mittels einer Validierungssuite qualifiziert wurde. Der Beitrag (Frank et al., 2008) auf der Konferenz „Automotive Safety und Security 2008“ beschreibt aus der Anwendersicht am Beispiel eines Compilers die verschiedenen Möglichkeiten, die Werkzeugqualität abzusichern. Das Fazit dieses Beitrages ist, dass eine Kombination verschiedener Testansätze, wie z.B. Back-To-Back-Test und Compliance Suites, in den Entwicklungsprozess des sicherheitsrelevanten Systems integriert werden müssen. Stürmer beschreibt in seiner Disseration (Stürmer, 2006) einen „Test Suite-orientierten Ansatz zur Absicherung einer Model-basierten Codegenerierung“.

Im weiteren Verlauf der Recherchen manifestierte sich, dass die Softwareentwicklungswerkzeuge nach den einschlägigen Publikationen meist nur projekt- oder anwenderbezogen qualifiziert werden. Später gab es hierzu einige praktische Ansätze und Veröffentlichungen. Exemplarisch seien an dieser Stelle (Slotosch & Reiling, 2012) und

(Beemster, 2013) genannt, die im Wesentlichen Automotive-Anwendungen adressieren. In (Asplund, El-khoury, & Törngren, 2012) wird ein Ansatz vorgestellt, der die Qualifikation der Softwareentwicklungswerkzeuge auf die Systemebene hebt und die Anforderungen an die Werkzeugkette als projektspezifische Sicherheitsziele in den Sicherheitsnachweis integriert.

Im Luftfahrtbereich gibt es im Umfeld der modellbasierten Entwicklung schon länger Überlegungen, mit qualifizierten Werkzeugketten zu arbeiten (Federal Aviation Administration, 2005). Allerdings hat die Werkzeugqualifizierung auch hier immer einen klaren Projektbezug. (Slotosch, 2012) beschreibt einen modellbasierten Werkzeugqualifizierungsansatz für die Eclipse-Entwicklungsumgebung inkl. verwendeter Plug-In's, der durch die Anforderungen aus dem Luftfahrtbereich nach der DO-330 (International Standard, 2011a) motiviert ist, aber auch im Automotive-Umfeld seine Anwendung finden kann. Diese Methode kann also generisch genutzt werden. Die Qualifizierung fokussiert dabei auf einen Validierungsansatz (validation suite), der modellbasiert und automatisiert durchgeführt werden kann. Eine weitere Veröffentlichung aus dem Luftfahrtbereich (Pothon, Pomies, Comar, & Brosgol, 2013) adressiert die Anforderungen an das „Tool Qualification Document" nach der DO-330, ebenfalls mit klarem Projektbezug.

Werkzeugunterstützte Sicherheitsnachweise, die auf der Verwendung einer qualifizierten Werkzeugkette beruhen, in der die Artefakte der Entwicklung hinterlegt sind, werden hingegen in der Literatur im Kontext der Funktionalen Sicherheit bis dato nicht beschrieben. Es gibt Veröffentlichungen, die sich mit der Erstellung von Sicherheitsnachweisen (safety case) und den Implikationen für die Bewertung der Funktionalen Sicherheit (independend safety assessment) beschäftigen (Birch et al., 2013), die aber nicht den Fokus auf der Verwendung eines integrierten Softwarewerkzeuges für das Assessment haben.

Für die gesamtheitliche Betrachtung von Safety (Funktionale Sicherheit) und Security (Informationssicherheit) gibt es zahlreiche Arbeiten. Gerade im Bereich Automation, Industrie 4.0, Industrial Internet of Things (IIoT) (Industrial Internet Consortium, 2016b) aber auch im Automotive-Umfeld (Burton, Likkei, Vembar, & Wolf, 2015) liegen umfangreiche Publikationen vor.

Für das Forschungsfeld der Integration von Reifegradmodellen (z.B. Automotive SPICE) und der Funktionalen Sicherheit gibt es zahlreiche Veröffentlichungen, z.B. (Dold, Frank, Gantner, & Pohl, 2012), (Messnarz, Habel, & Ross, 2010), (Messnarz, Sokic, Habel, König, & Bachmann, 2011), (König, Pappler, & Wild, 2013) und (Ekert & Hagenmeyer, 2009). Diese Veröffentlichungen legen den Schwerpunkt lediglich auf Anforderungen an sicherheitsrelevante Systeme, nicht aber auf Anforderungen an Softwareentwicklungswerkzeuge im Detail.

1.2.2 Abgrenzung dieser Arbeit

Der Schwerpunkt dieser Arbeit liegt auf Anwendungen für sicherheitsgerichtete Systeme, die den Anforderungen des generischen Standards zur Funktionalen Sicherheit, der IEC 61508, genügen müssen, sowie Anwendungen im Automotive-Umfeld, wo die ISO 26262 die normative Grundlage bildet.

Im Rahmen dieser Arbeit ist der deutsche Begriff Sicherheit immer äquivalent zum englischen Wort Safety – die Freiheit von Gefahren - und damit abgegrenzt zum Wort Security – Informationssicherheit.

1.2.3 Forschungshypothesen

Auf Basis der vorstehend formulierten wissenschaftlichen Ziele und der offenen Fragen, die sich aus dem Stand von Wissenschaft und Technik ableiten lassen, können folgende Forschungshypothesen aufgestellt werden:

1. **Sichere Werkzeuge und Werkzeugketten**
    a. Verschiedene internationale Standards stellen Anforderungen an Softwareentwicklungsprozesse. Sicherheitsnormen verlangen den vertrauenswürdigen Umgang mit den in der Systementwicklung eingesetzten Softwareentwicklungswerkzeugen. Hier gibt es Überschneidungen, die für einen generischen Ansatz genutzt werden können.
    b. Die Qualifizierung von Softwareentwicklungswerkzeugen (Tool Qualification) muss unabhängig von
        i. konkreten Projekten,
        ii. Domänen und
        iii. ggf. Standards erfolgen,
        iv. d.h. eine generische Werkzeugqualifizierung ist möglich.

c. Zahlreiche Softwareprodukte haben eine langjährige Historie. Bei der Qualifizierung von Softwareentwicklungswerkzeugen muss ein geeigneter Methodenmix angewendet werden, der sich am individuellen Produkt ausrichtet.

2. **Werkzeugunterstützte Sicherheitsnachweise**
   a. Qualifizierte Softwarewerkzeuge und Werkzeugketten, die in der Entwicklung sicherheitsrelevanter Systeme eingesetzt werden, sollten durch eine geeignete Methodik auch für die Beurteilung der Funktionalen Sicherheit (Assessment) und die Freigabe solcher Systeme verwendet werden.
   b. Moderne Konfigurationsmanagement-Systeme mit integriertem Anforderungsmanagement (sogenannte ALM[11]-Werkzeuge) bilden die Klammer um alle Dokumente im Softwareentwicklungsprozess und können somit durch eine geeignete Erweiterung der Funktionalität auch für das Assessment verwendet werden.
   c. Werkzeugunterstütztes Assessment reduziert die Aufwände erheblich und verbessert die Qualität und Vollständigkeit von Assessments.

[11] Application Lifecycle Management

1.3 Aufbau der Arbeit

Im Folgenden werden die Gliederung der Arbeit und das wissenschaftliche Vorgehen sowie die verwendeten Methoden beschrieben.

1.3.1 Gliederung der Arbeit

Diese Arbeit gliedert sich wie folgt (siehe Abbildung 4):

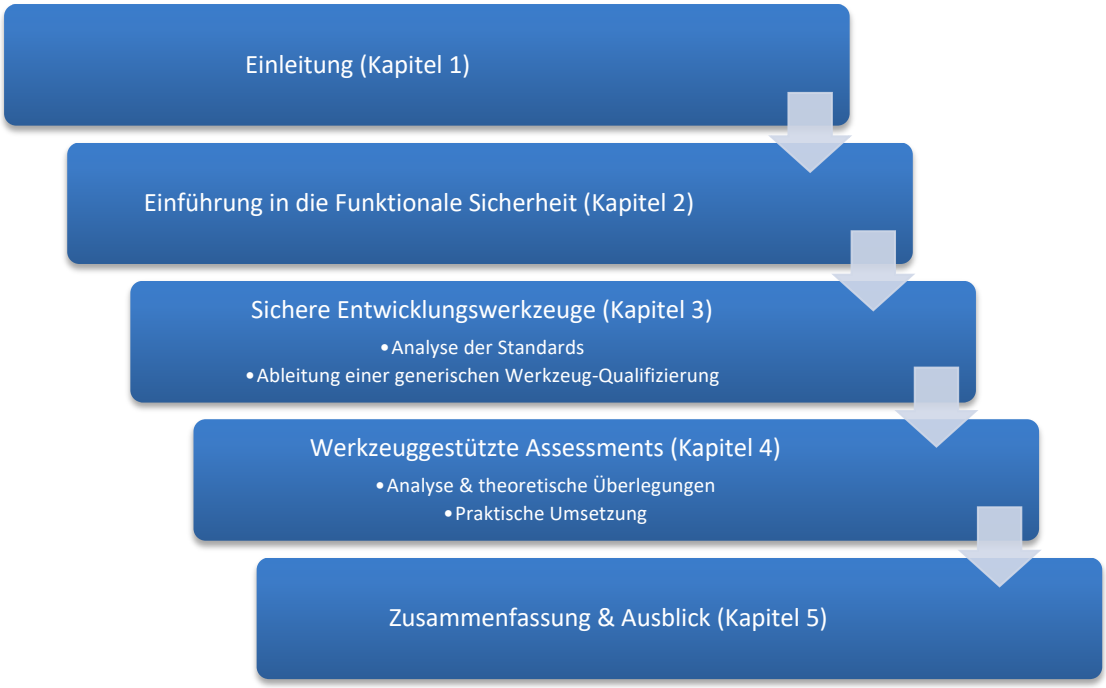

Abbildung 4: Aufbau der Arbeit

Nach einer Einleitung in Kapitel 1 werden in Kapitel 2 die Grundlagen der Funktionalen Sicherheit erläutert. Daraus werden im Kapitel 3 die in modernen Softwareentwicklungen typischerweise verwendeten Werkzeuge und Werkzeugketten untersucht und die Anforderungen aus den relevanten Standards analysiert. Auf dieser Basis wird ein Vorschlag für eine generische Qualifikation von Softwarewerkzeugen hergeleitet.

Im Kapitel 4 wird initial unterstellt, dass alle Fragestellungen bzgl. Werkzeugqualität des Kapitels 3 vollständig gelöst sind. Auf Basis einer integren Werkzeugkette wird zunächst ein theoretischer Ansatz entwickelt, das Assessment der Funktionalen Sicherheit in die Softwarewerkzeugkette zu integrieren. Auf der Grundlage dieser Überlegungen wurde auf Basis eines in der Praxis weitverbreiteten Werkzeuges zum Projektdatenhandling ein Demonstrator implementiert.

Das Kapitel 5 gibt eine Zusammenfassung und einen Ausblick.

1.3.2 Vorgehen und Methoden

Nachdem die Kapitel 1 und 2 die Arbeit einführen und die relevanten Normen vorstellen, werden in den Kapiteln 3 und 4 die Implikationen für die in Kapitel 1.2.3 formulierten Forschungshypothesen erarbeitet. Das Vorgehen ist dabei wie folgt: Im ersten Schritt erfolgt eine Analyse der Anforderungen und des Standes der Wissenschaft und Technik. Auf dieser Basis wird eine Lösung mit dem Ziel hergeleitet, die Forschungshypothesen zu belegen. Jeder Lösungsansatz wird mit Fallstudien in Kapitel 3 bzw. einer Beispielimplementierung unterlegt und somit die Relevanz in der Praxis gezeigt.

1.4 Veröffentlichungen im Rahmen dieser Arbeit

Im Rahmen dieser Arbeit gab es zahlreiche Veröffentlichungen, die entweder publiziert wurden (siehe Kapitel 1.4.1) oder als Konferenzbeiträge eingereicht und gehalten wurden (siehe Kapitel 1.4.2).

Im Laufe der Zeit wurden die Konzepte der generischen Werkzeugqualifizierung und des werkzeugunterstützten Assessments zunehmend verfeinert und in aktualisierter Form zyklisch veröffentlicht und der Fachwelt zur Diskussion gestellt.

Darüber hinaus war der Autor u.a. in folgenden Arbeitskreisen mit Fokus Funktionale Sicherheit und Software tätig:

- ***NA 052-00-32-08-01/2 bzw. vormals NA 052-01-26-01/2***: deutsches Spiegelgremium zum **ISO/TC 022/SC 03/WG 16** „Functional Safety" – „Funktionssicherheit", verantwortlich für die Entwicklung des Standards ISO 26262
- **DKE *K134:*** „Gebrauchsfähigkeit und Qualität bei erneut verwendeten Teilen und Geräten der Elektrotechnik"
- ***DKE UK966.1:*** deutsches Spiegelgremium zum ***CENELEC SC 31-9*** „Electrical apparatus for the detection and measurement of combustible gases to be used in industrial and commercial potentially explosive atmospheres" - „Mess- und Warngeräte für gefährliche Gase"
- **IEC *TC31*** „Equipment for explosive Atmospheres"

1.4.1 Publikationen

Die im Rahmen dieser Arbeit veröffentlichten Publikationen sind in Tabelle 2 aufgelistet. In der rechten Spalte ist dargestellt, welchen direkten Bezug die einzelnen Titel zur Bearbeitung der einzelnen Forschungshypothesen haben.

Tabelle 2: Publikationen im Rahmen dieser Arbeit

Titel	Autoren	Jahr	Verweis	Forschungs-hypothese
„Konzept einer PDM-Tool gestützten Erstellung von Softwaresystemen und einer Zertifizierungs-Suite“, White-Paper im Rahmen des Forschungsprojekts SPES 2020	Andreas Bärwald, Manfred Inderst, Dominik Soyer	2012	(Bärwald, Inderst, & Soyer, 2012)	2a,b
„Konsequent - Interview: ISO 26262 fordert Aktivitäten im kompletten Entwicklungszyklus“, IX 01-02/2012 Extra Seite VIff,	Andreas Bärwald, Barbara Lange	2012	(Bärwald & Lange, 2012)	1a-c
„Simulation im Zertifizierungsprozess von Produkten: Anforderungen an die Simulationsmodelle und die Simulationswerkzeuge (Certification of Embedded Software System: State of the Art and Future Requirements)“, White-Paper im Rahmen des Forschungsprojekts SPES 2020	Andreas Bärwald, Manfred Inderst, Roland Rosen, Erwin Smit-Wiesner, Philipp Emanuel Stelzig, Jan Christoph Wehrstedt, Doris Wild	2011	(Bärwald, Inderst, Rosen, Smit-Wiesner, et al., 2011)	1a,b 2a
„Zertifizierung von Automatisierungssoftware: Ist-Analyse und Anforderungen“, White-Paper im Rahmen des Forschungsprojekts SPES 2020	Andreas Bärwald, Manfred Inderst, Roland Rosen, Erwin Smit-Wiesner, Philipp Emanuel Stelzig, Jan Christoph Wehrstedt, Doris Wild	2011	(Bärwald, Inderst, Rosen, Stelzig, et al., 2011)	1a-c 2a,b
„Dependability-Betrachtung von Multicore-Scheduling“, HANSER automotive 11/2010, Seite 24ff	Andreas Bärwald, Michael Deubzer, Jürgen Mottok	2010	(Bärwald, Deubzer, & Mottok, 2010)	-
„Sichere Codegenerierung“, HANSER automotive 01-02/2010, Seite 30ff	Andreas Bärwald, Michael Beine	2010	(Bärwald & Beine, 2010)	1a-c

Titel	Autoren	Jahr	Verweis	Forschungs-hypothese
„Qualifizierung und Zertifizierung von Software-Entwicklungswerkzeugen“, HANSER automotive 01-02/2010, Seite 34ff	Andreas Bärwald, Harald Hauff, Jürgen Mottok	2010	(Bärwald, Hauff, & Mottok, 2010)	1a-c
„Qualification and Certification of Development Tools for Safety-Critical Applications”, im Band der “Automotive - Safety & Security 2010 - Sicherheit und Zuverlässigkeit für automobile Informationstechnik”, Seite 135ff	Andreas Bärwald, Jürgen Mottok, Harald Hauff	2010	(Bärwald, Mottok, & Hauff, 2010)	1a-c
„Qualification and Certification of Development Tools for Safety-Critical Applications”, White-Paper für dependability@Siemens'2009, Siemens AG	Andreas Bärwald, Harald Hauff, Jürgen Mottok	2009	(Bärwald, Hauff, & Mottok, 2009)	1a-c
„IEC 61508 and MISRA C - The Benefits of Utilising IEC 61508 and MISRA C for Automotive Applications” Band zur “The 1st IEE Automotive Electronics Conference, London”, Seite 7ff	Andreas Bärwald	2005	(Bärwald, 2005)	-

1.4.2 Konferenzbeiträge

Neben den in Kapitel 1.4.1 aufgelisteten Publikationen wurden die Forschungshypothesen in zahlreichen Konferenzbeiträgen bearbeitet und auf einschlägigen Fachkonferenzen einem Expertenpublikum zur Diskussion gestellt. Da Ergebnisse wissenschaftlicher Arbeit erst nach einiger Zeit ihren Weg in Standardisierung bzw. Normung finden, ist dieser Austausch zu technischen Entwicklungen in der globalen „Community“ der Experten für Funktionale Sicherheit sehr wichtig. In der Tabelle 3 sind die für diese Arbeit relevanten Konferenzbeiträge unter Mitwirkung des Autors dieser Arbeit zusammengefasst und in Relation zu den Forschungshypothesen gesetzt. Speziell für die Bearbeitung der Forschungshypothesen 1a-c gibt es eine Vielzahl von Beiträgen, die ihren Anfang im Jahr 2006 auf der safetronic mit der Vorstellung der Idee für eine Werkzeugzertifizierung hatte. Diese Idee wurde dann ab 2009/10 bis 2016 kontinuierlich weiterentwickelt und verfeinert.

Tabelle 3: Konferenzbeiträge im Rahmen dieser Arbeit

Titel / Veranstaltung	Autoren	Jahr	Verweis	Forschungs-hypothese
„Open Source Licensing and Governance"	Dr. Catharina Maracke, Nicole Pappler, Kayoko Takanishi, Andreas Bärwald	2018	(Bärwald, Maracke, Takanishi, & Pappler, 2018)	-
„Free and Open Source Software (FOSS) sicher in IoX-Projekten verwenden", Internet of Things - vom Sensor bis zur Cloud 2017, München	Andreas Ainöder, Andreas Bärwald, Nicole Pappler	2017	(Ainöder, Bärwald, & Pappler, 2017)	-
„IEC 61850 Interoperability & Integration Integrity Checks", DISTRIBUTECH 2017, San Diego	Andreas Bärwald, Peter Pfisterer	2017	(Bärwald & Pfisterer, 2017)	-
„Software Tool Qualification and Certification for Functional Safety", IIC Q2/2016-Meeting, Tokyo	Andreas Bärwald	2016	(Bärwald, 2016)	1a-c
„Toolunterstützte Assessments", safe.tech 2015, München	Andreas Bärwald, Christoph Bräuchle, Jürgen Mottok	2015	(Bärwald, Bräuchle, & Mottok, 2015b)	2a-c
„Tool based assessments", IQPC Embedded Conference 2015, Berlin	Andreas Bärwald, Christoph Bräuchle, Jürgen Mottok	2015	(Bärwald, Bräuchle, & Mottok, 2015a)	2a-c
„Functional Safety in the Automotive Industry - ISO 26262 - Status quo of the ISO 26262: A snapshot from the point of view of an assessment and certification authority", TÜV SÜD safe future now conference 2014, Tokyo	Andreas Bärwald	2014	(Bärwald, 2014a)	1a-c

Titel / Veranstaltung	Autoren	Jahr	Verweis	Forschungs-hypothese
„Functional Safety in the automotive industry - Status quo & upcoming trends from the point of view a certification authority", IQPC Automotive Embedded Multi-Core Systems 2014, Sindelfingen	Andreas Bärwald	2014	(Bärwald, 2014b)	1a-c 2a
„Functional Safety in the Automotive Industry - Status quo & upcoming trends from the point of view of a certification authority", The 2nd International Symposium on Road Vehicles - Functional Safety Standards and its Application, Shanghai	Andreas Bärwald	2013	(Bärwald, 2013)	1a-c
„Reproduzierbare und effiziente Toolqualifizierung und TCL-Avoidance", safe.tech 2013, München	Andreas Bärwald, Christoph Bräuchle	2013	(Bärwald & Bräuchle, 2013)	1a-c
„Toolunterstützung für die Entwicklung von eingebetteten Multicore-Systemen mit Safety- und Echtzeitanforderungen", safe.tech 2012, München	Andreas Bärwald, Michael Deubzer, Martin Hobelsberger, Jürgen Mottok,	2012	(Bärwald, Deubzer, Hobelsberger, & Mottok, 2012)	1a
„Softwarewerkzeuge für sicherheitsgerichtete Entwicklungen - Anforderungen und Lösungsansätze", Funktionale Sicherheit – Fachkonferenz, Düsseldorf	Andreas Bärwald, Bernd Spanfelner	2011	(Bärwald & Spanfelner, 2011)	1a,b
„Softwarewerkzeuge für sicherheitsgerichtete Entwicklungen - Anforderungen und Lösungsansätze", 2. FuSi Forum, Osnabrück	Andreas Bärwald, Doris Wild	2011	(Bärwald & Wild, 2011)	1a,b
„ISO 26262 - Der neue Automotive Standard für Funktionale Sicherheit", Bayerisches IT-Sicherheitscluster: Automotive Forum AFS3, Regensburg	Andreas Bärwald	2010	(Bärwald, 2010b)	1a,b
„Zertifizierung von Werkzeugen und Werkzeugketten", Software im Automobil, Stuttgart	Andreas Bärwald, Jürgen Mottok	2010	(Bärwald & Mottok, 2010)	1a-c

Titel / Veranstaltung	Autoren	Jahr	Verweis	Forschungs-hypothese
„Certification of safety relevant systems - Benefits of using pre-certified components", Wind River Industrial Event, Paris	Andreas Bärwald	2010	(Bärwald, 2010a)	1a,b
„Creating Safety Relevant Systems by using Generic Software Components (for example an RTOS)", Embedded Systems Conference ESC UK, Farnborough	Andreas Bärwald, Richard Barry	2009	(Bärwald & Barry, 2009)	1a
„Hypervisor - Virtualisierungsplattform zur Trennung von Sicherheitsfunktionen verschiedener Safety Integrity Level", SIL Konferenz, Basel	Andreas Bärwald, Andreas Buchwieser	2009	(Bärwald & Buchwieser, 2009)	1a
„Einsatz zertifizierter Codegenerierungswerkzeuge in sicherheitsgerichteten Entwicklungen", safetronic'09, München	Andreas Bärwald, Michael Beine	2009	(Bärwald & Beine, 2009)	1a-c
„Softwarebus für sicherheitsgerichtete Anwendungen", Konferenz Sicherheitsgerichtete Systeme, Stuttgart	Andreas Bärwald, Cecil Bruce-Boye	2008	(Bärwald & Bruce-Boye, 2008)	-
„Hypervisor - Virtualisierungsplattform zur Trennung von Sicherheitsfunktionen verschiedener Safety Integrity Level", safetronic'08, München	Andreas Bärwald, Andreas Buchwieser	2008	(Bärwald & Buchwieser, 2008)	-
„Implementierung eines sicherheitsgerichteten Software-Entwicklungsprozesses am Beispiel eines Kfz-Zulieferers", safetronic'08, München	Andreas Bärwald, Sahin Duygun	2008	(Bärwald & Duygun, 2008)	1a,b
„Werkzeuge in einem sicherheitsgerichteten Entwicklungsprozess", safetronic'06, München	Andreas Bärwald	2006	(Bärwald, 2006)	1a,b
„MISRA C und IEC61508 - Zertifizierung einer IEC 61508-konformen Software-Entwicklungsprozess mit MISRA C", MISRA DAY 2004, Ludwigsburg	Andreas Bärwald	2004	(Bärwald, 2004)	-

Die Konferenzbeiträge liegen in den meisten Fällen in Form von Foliensätzen in den Tagungsunterlagen vor. „Ausformulierte Paper“ sind in diesem Kontext eher unüblich. Die veröffentlichten Ausarbeitungen in Volltext sind in Kapitel 1.4.1 aufgeführt.

2 Einführung in die Funktionale Sicherheit

"Das Verhüten von Unfällen darf nicht als eine Vorschrift des Gesetzes aufgefasst werden, sondern als ein Gebot menschlicher Verpflichtung und wirtschaftlicher Vernunft"[12]

Schon Werner von Siemens formulierte diesen Anspruch Ende des 19. Jahrhunderts. Es geht also darum, die Gefahren für Leib und Leben auf ein gesellschaftlich akzeptiertes Maß zu reduzieren (siehe Abbildung 7 im folgenden Kapitel). Moderne Sicherheitsstandards verfolgen als Nachweismethodik für die erreichte Risikominimierung einen probabilistischen Ansatz, d.h. es wird das Ausfallverhalten der Systeme zusätzlich zu qualitativen Maßnahmen quantitativ berechnet (siehe Kapitel 2.1.).

Im Kontext dieser Arbeit ist der Begriff Sicherheit mit dem englischen Wort Safety gleichzusetzen, d.h. die Sicherheit vor Gefährdungen für Leib und Leben und/oder katastrophalen Auswirken für die Umwelt. Davon abgegrenzt werden muss der Begriff Security, also Sicherheit im Sinne von Informationssicherheit, u.a. dem Schutz vor Manipulationen, die Gewährleistung von Vertraulichkeit und Verfügbarkeit. In zunehmend vernetzten Systemen kann ein Security-Vorfall zu einem erheblichen Sicherheitsproblem im Sinne von Safety führen, so dass die Security-Eigenschaften eines Systems auch Safety-Anforderungen sein können. Das wird in modernen Safety-Standards, z.B. IEC 61508 Ed. 2 (Deutsche Norm, 2010a), zumindest adressiert. Für die Umsetzung wird explizit aber auf geeignete Standards aus dem Security-Umfeld verwiesen, z.B. IEC 62443 bzw. ISO 2700X. Ein integraler Ansatz, wie Security-Anforderungen aus der Prozesssicht in den Lebenszyklus der Funktionalen Sicherheit eingebettet werden können, wurde u.a. von Burton, Likkei, Vembar und Wolf (Burton et al., 2015) vorgestellt:

"… the safety and security of the system depends on a good system design, which also manages to resolve potentially conflicting requirements of the two. The approach described here, however ensures that sufficient rigor is applied during the design process that all potential risks are analyzed and adequately mitigated against in a manner that is open to professional scrutiny. The combined approach helps to identify synergies and

[12] Werner von Siemens, Zitat von 1880 (Siemens AG, 2016)

potential conflicts at an early design phase and ensures that the evaluation assurance level achieved for security threats is appropriate to the potential safety risk, measures in ASILs resulting from those threats."

Es wird also ein integraler Ansatz vorgeschlagen, den Sicherheitszielen der Funktionalen Sicherheit (Safety Goals) entsprechende Security Goals gegenüberzustellen und die Betrachtung in einen gemeinsamen Prozessüberbau einzubetten. Dieses Vorgehen wird in der Abbildung 5 schematisch dargestellt.

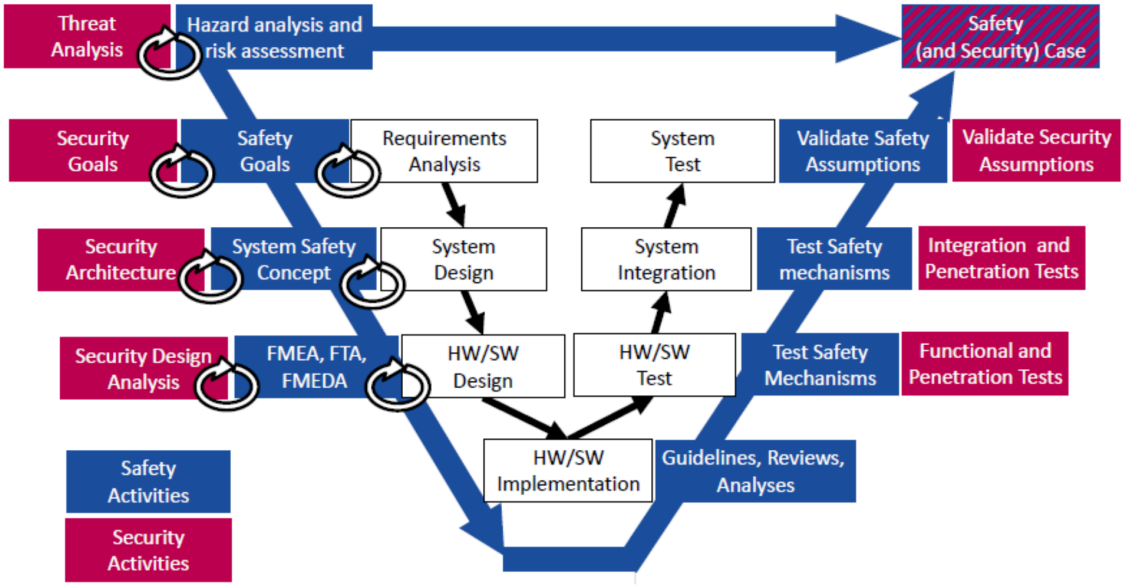

Abbildung 5: Vorschlag eines um Security Aktivitäten erweiterten Prozessmodells für Funktionale Sicherheit nach (Burton et al., 2015)

Typischerweise sind moderne Prozessmodelle phasenorientiert und basieren auf den Grundüberlegungen des V-Modells (siehe Kapitel 2.4.2.1). In diesem Fall wurden zwei V's ineinander verschachtelt; eins für den Safety-Anteil und das zweite für den Security-Teil.

Auch im Industrial Internet Consortium[13] wird im „Industrial Internet Security Framework" (IISF) (Industrial Internet Consortium, 2016b) auf Basis der „Industrial Internet

[13] Das Industrial Internet Consortium (IIC) ist eine weltweite Vereinigung von Industrieunternehmen und Forschungseinrichtungen mit 258 Mitgliedorganisationen (Stand November 2016). Ziel des IIC ist die schnelle Entwicklung, Adoption und Verbreitung von vernetzten Maschinen und Geräten sowie intelligenter Nutzung der anfallenden Daten (data analytics) im Kontext „Industrial Internet". Gegründet wurde das IIC im März 2014 durch die Firmen AT&T, Cisco, General Electric, IBM und Intel. (http://www.iiconsortium.org)

Reference Architecture“ (IIRA) (Industrial Internet Consortium, 2016a) domänenübergreifend die Brücke zwischen Safety und Security geschlagen.
Untersuchungen bzgl. der Security-Anforderungen an sicherheitsgerichtete Systeme sind nicht im weiteren Betrachtungsumfang dieser Arbeit enthalten.

2.1 Ziele der Funktionalen Sicherheit

„Das Risiko, das mit einem bestimmten technischen Vorgang oder Zustand verbunden ist, wird zusammenfassend durch eine Wahrscheinlichkeitsaussage beschrieben, die

- die zu erwartende Häufigkeit des Eintritts eines zum Schaden führenden Ereignisses und
- das beim Ereigniseintritt zu erwartende Schadensausmaß

berücksichtigt.“[14]

Zum besseren Verständnis der Ziele der Funktionalen Sicherheit werden am Anfang dieses Kapitels drei Begriffe eingeführt:

1. **Sicherheit (Safety)**: wird in diesem Kontext als „Freiheit von inakzeptablen Risiken” definiert,
2. **Risiko (Gefährdung):** ist die Kombination der Auftrittswahrscheinlichkeit und des Schadensausmaßes sowie
3. **Schadensausmaß:** damit ist primär die Verletzung oder der Tod von Menschen im Fokus. Es kann aber auch die Gefährdung der Umwelt oder der Schutz von Produktionsmitteln adressiert werden.

Das Ziel der Funktionalen Sicherheit ist es, das Restrisiko eines elektrischen/elektronischen Systems auf ein gesellschaftlich akzeptiertes Niveau zu bringen.

„Die Problematik der rechtlichen Umsetzung technischer Risiken besteht also in der Festlegung einer Grenze, ab der das juristische Risiko zu einer Gefahr wird (Gefahrenschwelle). Wie die Abbildung 6 zeigt, teilt aus juristischer Sicht die Gefahrenschwelle den gesamten Risikobereich in den Bereich der gesellschaftlich unerwünschten

[14] Zitat aus (International Standard, 1987) Kapitel 2.2

Gefahren und den Bereich des gesellschaftlich zu akzeptierenden Restrisikos.“[15] (siehe Abbildung 6)

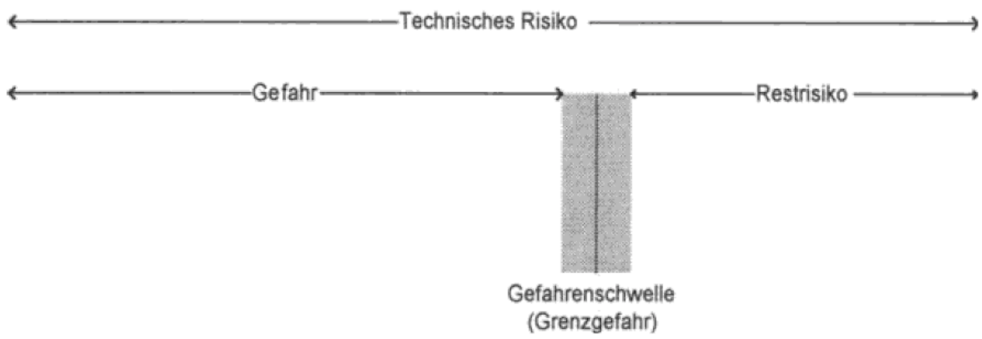

Abbildung 6: Definition des Restrisikos nach (Becker, 1995) **Seite 30**

Es kann also folgende Risikoformel formuliert werden nach (Bundesministerium für Umwelt für Naturschutz und Reaktorsicherheit, 2004) Seite 13:

$$R = S \, x \, E$$

mit den Kennzahlen

R = Risiko, Risikozahl als Maß für das Risiko

S = Eintrittshäufigkeit des Schadensereignisses

E = Schadensausmaß, Konsequenz.

Die Risikoreduzierung erfolgt auf Basis einer Gefährdungsanalyse (siehe Kapitel 2.2), die das Ausgangsrisiko eines Systems anhand standardisierter Parameter ermittelt und daraus resultierend die notwendige Risikominimierung definiert (siehe Abbildung 7).

[15] Zitat aus (Deutsche Norm, 2009) Seite 30

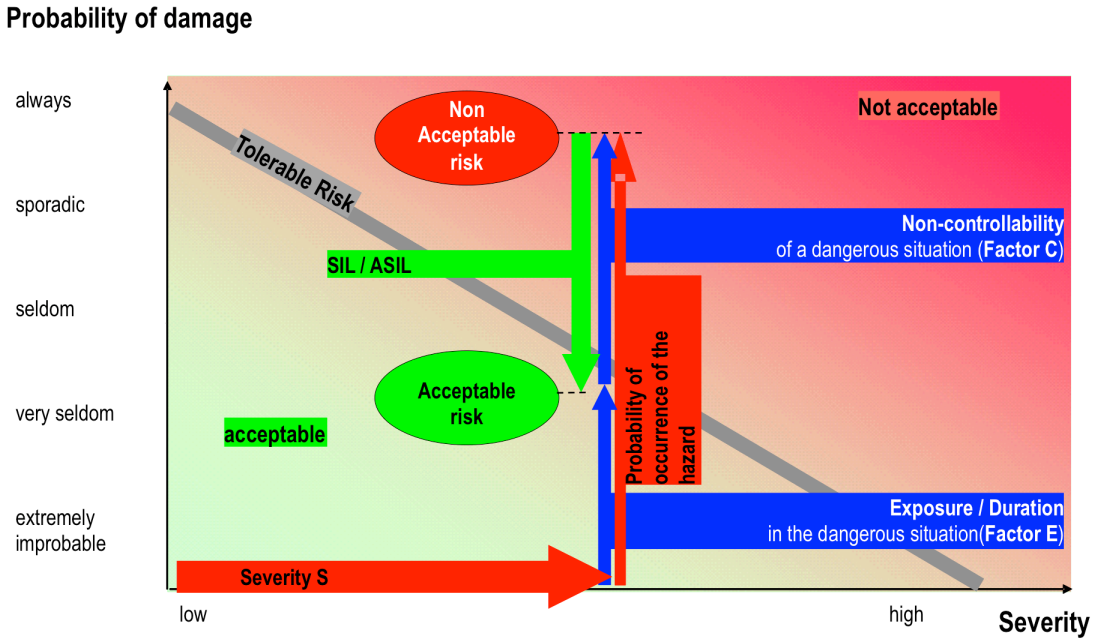

Abbildung 7: Gefährdungsbeurteilung und Risikominimierung auf Basis der Parameter S, E und C basierend auf IEC 61508, ISO 26262 und ISO 12100 (Quelle: TÜV SÜD Automotive GmbH, Functional Safety Training)

In Abhängigkeit der Parameter Schadensausmaß S (severity) und der Auftrittswahrscheinlichkeit, die sich aus den Parametern Häufigkeit E (exposure) und Beherrschbarkeit C (controllability) zusammensetzt, wird das potentielle Risiko des Systems ermittelt. Also kann die Risikoformel um die Beherrschbarkeit wie folgt ergänzt werden:

$$R = S \, x \, E \, x \, C$$

Im Beispiel in Abbildung 7 liegt dieses Risiko oberhalb des gesellschaftlich akzeptierten Risikos (tolerable risk). Deshalb ist eine Risikoreduzierung (SIL/ASIL, siehe Kapitel 2.2) in diesem Beispiel gefordert, um das Restrisiko unterhalb der Grenze für das tolerierte Risiko zu drücken.

2.2 Die Gefährdungsbeurteilung

Am Anfang eines Projektes zur Funktionalen Sicherheit steht immer eine Analyse der potentiellen Gefährdungen, die von einer Fehlfunktion des Systems ausgehen können. Daraus kann das Maß der nötigen Risikominderung abgeleitet werden (siehe Kapitel 2.1 und Abbildung 7). Dies ist am Beispiel der ISO 26262 (International Standard, 2011b) der sogenannte ASIL (Automotive Safety Integrity Level). In anderen Standards zur Funktionalen Sicherheit gibt es ähnliche Ansätze, wie z.B. Performance Level und Kategorien

in der Welt der Maschinensicherheit, z.B. ISO 12100 (Deutsche Norm, 2010d), oder den SIL (Safety Integrity Level) der Basisnorm IEC 61508 (Deutsche Norm, 2010a).

Im Folgenden wird das Vorgehen zur Ermittlung des ASIL am Beispiel der ISO 26262 vorgestellt. Die Risikoanalyse wird immer durch ein Expertengremium durchgeführt, welches gemeinsam die Klassifizierung der einzelnen Parameter vornimmt.

Zur Bestimmung des ASIL werden die bereits eingeführten drei Parameter

1. **Severity (S)**, das Schadensausmaß
2. **Exposure (E)**, die Auftrittswahrscheinlichkeit bzw. Häufigkeit und
3. **Controllability (C)**, die Beherrschbarkeit

verwendet. Dabei werden aber normativ die Klassifizierungen der einzelnen Parameter vorgegeben. Der Parameter S klassifiziert den Grad der möglichen Unfallfolgen zwischen S0 und S3 (siehe Abbildung 8).

Class	S0	S1	S2	S3
Description	No injuries	Light and moderate injuries	Severe and life-threatening injuries (survival probable)	Life-threatening injuries (survival uncertain) or fatal injuries

Abbildung 8: Beispielhafte Klassifikation des Schadensausmaßes nach ISO 26262-3 Tabelle B.1 (International Standard, 2011b)

Der Parameter E bestimmt, wie hoch die Auftrittswahrscheinlichkeit des Schadensszenarios ist. Dafür wird eine Skala von E0 (unwahrscheinlich) bis E4 (höchstwahrscheinlich) verwendet (siehe Abbildung 9).

	Class				
	E0	E1	E2	E3	E4
Description	Incredible	Very low probability	Low probability	Medium probability	High probability

Abbildung 9: Beispielhafte Klassifikation der Auftrittswahrscheinkeit nach ISO 26262-3 Tabelle B.2 (International Standard, 2011b)

Im letzten Schritt wird die Beherrschbarkeit in der Fehlersituation durch einen „Normalfahrer“ bewertet (siehe Abbildung 10). Die Skala reicht von C0 (immer beherrschbar) bis C3 (weniger als 90% der Fahrer können das Schadensereignis verhindern).

Driving factors and scenarios	Class of controllability (see Table 3)			
	C0	C1	C2	C3
	Controllable in general	99 % or more of all drivers or other traffic participants are usually able to avoid harm	90 % or more of all drivers or other traffic participants are usually able to avoid harm	Less than 90 % of all drivers or other traffic participants are usually able, or barely able, to avoid harm

Abbildung 10: Beispielhafte Klassifikation der Beherrschbarkeit nach ISO 26262-3 Tabelle B.4 (International Standard, 2011b)

Für die Parameter E und C sind in der ISO 26262 Band 3 Tabelle B.2 und B.4 entsprechende Beispiele für die Automotive-Domäne gelistet.

Wurden die Parameter S, E und C nach oben genanntem Vorgehen ermittelt, können in der Risikomatrix (siehe Abbildung 11) die entsprechenden Automotive Safety Integrity Level (ASIL) abgelesen werden. QM bedeutet, dass ein etabliertes Qualitätsmanagement-System[16] für dieses Restrisiko ausreichend ist. Der ASIL A stellt die niedrigsten Anforderungen, am anderen Ende der Skala steht der ASIL D.

		Controllability C		
Severity S	Exposure E	C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	ASIL A
	E4	QM	ASIL A	ASIL B
S2	E1	QM	QM	QM
	E2	QM	QM	ASIL A
	E3	QM	ASIL A	ASIL B
	E4	ASIL A	ASIL B	ASIL C
S3	E1	QM	QM	ASIL A
	E2	QM	ASIL A	ASIL B
	E3	ASIL A	ASIL B	ASIL C
	E4	ASIL B	ASIL C	ASIL D

Abbildung 11: Risikomatrix nach ISO 26262-3 Tabelle A.4 (International Standard, 2011b)

Der ASIL definiert die entsprechenden technischen und prozessualen Anforderungen für die Entwicklung des sicherheitsrelevanten Systems.

[16] Gängige Standards für Qualitätsmanagement-Systeme sind die ISO 9001 (Deutsche Norm, 2008b) und im Automotive-Umfeld im Besonderen die ISO TS 16949 (Deutsche Norm, 2009).

2.3 Erreichung der Funktionalen Sicherheit

Ein System ist funktional sicher, wenn zufällige Fehler während der Laufzeit beherrscht und systematische Fehler während der Entwicklung vermieden werden (siehe Abbildung 12).

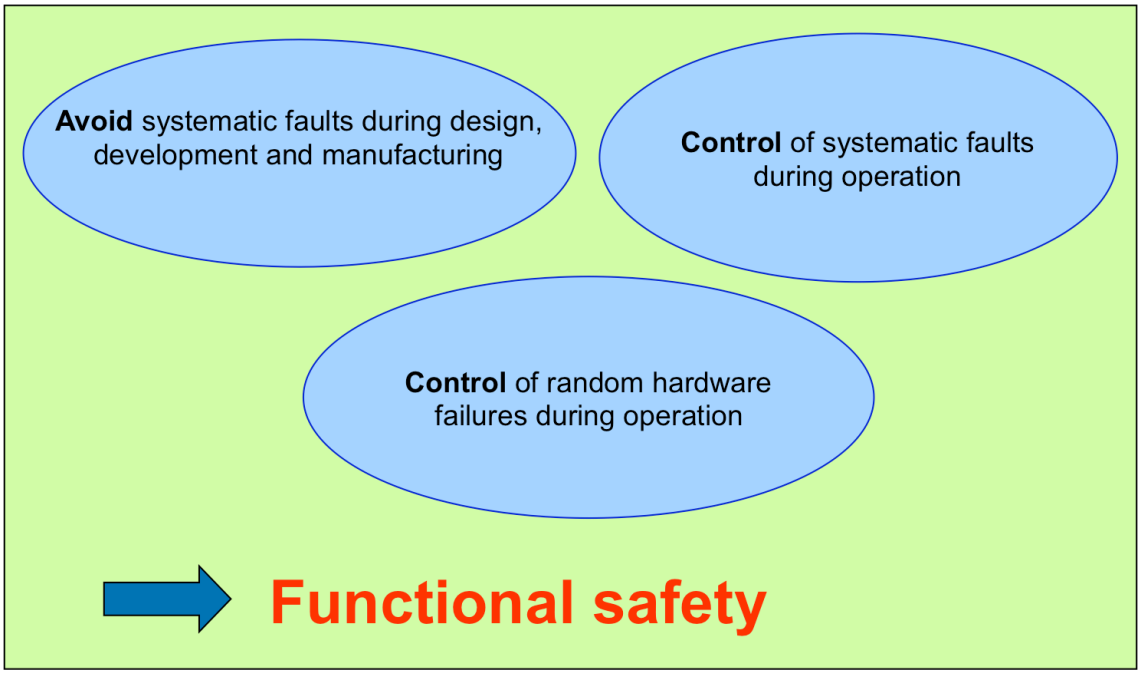

Abbildung 12: Aspekte der Funktionen Sicherheit in Anlehnung an IEC 61508 (Quelle: TÜV SÜD Automotive GmbH, Functional Safety Training)

Wird davon ausgegangen, dass Software keine zufälligen Fehler hat, sondern alle Softwarefehler bei der Entwicklung des Systems bereits eingebracht werden, müssen Maßnahmen definiert werden, die Einfluss auf die Qualität und Zuverlässigkeit von Software haben. Das sind im Wesentlichen geeignete Softwareentwicklungsprozesse, angereichert mit einigen Methodenempfehlungen. Dies ist der allgemeine Ansatz in den verschiedenen Sicherheitsstandards, die im Weiteren untersucht werden.

Die grundlegenden Arbeiten zum Umgang mit zufälligen Hardwarefehlern, deren Diagnose und Beherrschung stammen aus den 1980iger Jahren von Hölscher/Rader (Hölscher & Rader, 1984) und Hedtke (Hedtke, 1984). Diese Ansätze haben auch heute noch für moderne Systeme Gültigkeit. Beispielsweise wurden Methoden für RAM-, ROM- und Befehlssatztests von Mikroprozessorsystemen von diesen Autoren etabliert. Brown stellte 1998 einen Ansatz für einen CPU-Selbsttest für eine Motorola 68332 CPU unter dem Titel „Solving the software safety paradox“ (Brown, 1998) vor:

„Embedded systems running safety-critical applications face a similar paradox. How can the software know that it is operating correctly, especially if it isn't?"

Die Herausforderung besteht also darin, die Funktionalität der CPU zur Laufzeit zu testen und bei erkanntem Fehlerverhalten den „sicheren Zustand" einzuleiten. Dabei wird aus den Mikrobefehlen eine Art Code-Labyrinth aufgebaut (siehe Abbildung 13), deren Ausgang nur erreicht wird, wenn alle Operationen fehlerfrei ausgeführt wurden. Dabei werden die verschiedenen Befehlsgruppen sequentiell getestet. Begonnen wird mit den unbedingten Sprüngen (Zeile 02). Dann folgen bedingte Sprünge (Routine cpu0 ab Zeile 12) usw. bis hin zu den Tests der Registeroperationen (Routine cpu32 ab Zeile 35). Kommt es zu einem Fehler, wird das Programm in eine Endlosschleife geführt (Routine fail ab Zeile 07) und hardwarenahe Sicherungsmaßnahmen, wie ein Watchdog, führen das System in einen definierten, sicheren Zustand.[17]

```
01  _cpu_test:
02  /* Branch Always Test */
03  bra cpu0 /* branch always */
04  move.l #FALSE,d0 /* if falls through, error */
05  rts /* exit */
06  /* Any failures from the tests below branch to this fail location. */
07  fail:
08  movem.l (sp)+,d0-d7/a0-a6 /* restore registers */
09  /* d0 was included for testing*/
10  move.l #FALSE,d0 /* error return */
11  rts
12  cpu0: movem.l d0-d7/a0-a6,-(sp) /* push registers - test later*/
13  /* d0 is included to test later */
14  /* Branch on equ/ne zero Test */
15  clr.l d0 /* clear a register */
16  bne fail /* should not branch */
17  beq cpu10 /* should branch */
18  bra fail /* go fail */
19  cpu10: /* Move / Compare / Branch on Conditions / Jmp Tests */
20
21
```

[17] Diese Methode kann auf einkanaligen Systemen nie zu einer Betriebsart „fail-operation" führen, sondern solche sind lediglich „fail-safe", d.h. die Systeme schalten im Fehlerfall kontrolliert ab.

```
22  move.l #LONGPOSVALUE,d0 /* set value in register */
23  cmp.l #LONGPOSVALUE,d0 /* equal ? */
24  bne fail /* should not branch */
25  bge cpu20 /* should branch */
26  bra fail /* go fail */
27  cpu20: bgt fail /* should not branch */
28  cmp.l #0,d0 /* equal ? */
29  beq fail /* should not branch */
30  blt fail /* should not branch */
31  bgt cpu30 /* should branch */
32  bra fail /* go fail */
33  cpu30: bge cpu31 /* should branch */
34  bra fail /* go fail */
35  cpu31: jmp cpu32 /* should jump */
36  bra fail /* go fail */
37  cpu32: /* Test movem.l and cmpm */
38  cmp.l 4(sp),d1 /* regs d1 - a6 must compare */
39  bne fail /* if no compare - fail */
40  cmp.l 8(sp),d2
41  bne fail
42  cmp.l 12(sp),d3
43  bne fail
44  cmp.l 16(sp),d4
45  bne fail
46  cmp.l 20(sp),d5
47  bne fail
48  cmp.l 24(sp),d6
49  bne fail
50  cmp.l 28(sp),d7
51  bne fail
52  cmpa.l 32(sp),a0
53  bne fail
54  cmpa.l 36(sp),a1
…   bne fail
    cmpa.l 40(sp),a2
    bne fail
```

```
    cmpa.l 44(sp),a3
    bne fail
    …
```

Abbildung 13: Auszug des CPU Tests für einen Motorola 68332 Mikrocontroller nach (Brown, 1998)

Dieser Ansatz kann leicht auch für andere embedded Plattformen wie z.B. MIPS adaptiert werden. Mit einem Simulator wie SPIM (Larus, 1997) kann dann die Wirksamkeit der Methodik leicht gezeigt werden.[18]

2.4 Überblick über die relevanten Sicherheitsstandards

Im Bereich der Elektrotechnik und Elektronik ist im Wesentlichen eine Standardisierungsorganisation auf der internationalen, weltweiten Ebene von Bedeutung: die International Electrotechnical Commission (IEC). Die Grundnorm zur Funktionalen Sicherheit, die IEC 61508 (Deutsche Norm, 2010a), wird so beispielsweise aus der Arbeitsgruppe IEC/SC 65A WG 14 vorangetrieben. Applikationsstandards für die jeweiligen Industrien werden von der International Organization for Standardization (ISO) getrieben, z.B. die ISO 26262 (International Standard, 2011b) für die Funktionale Sicherheit von Straßenfahrzeugen im Gremium ISO/TC 022/SC 03/WG 16. Auf nationaler Ebene in Deutschland werden die Standards für Elektrotechnik und Elektronik von der Deutsche Kommission Elektrotechnik Elektronik Informationstechnik (DKE) und sonstige Standards vom Deutschen Institut für Normung (DIN) in den entsprechenden Spiegelgremien zu den internationalen Arbeitsgruppen entwickelt. So ist beispielsweise beim DIN der Arbeitskreis NA 052-00-32-08-01 das Spiegelgremium zum ISO/TC 022/SC 03/WG 16 auf internationaler Ebene. DIN vertritt die deutschen Interessen in den entsprechenden ISO Arbeitsgruppen. Auf europäischer Ebene gibt es darüber hinaus das Europäische Komitee für elektrotechnische Normung (CENELEC).

Die Betrachtung der einschlägigen Normen, die Anforderungen an Software für (sicherheitsgerichtete) eingebettete Systeme beinhalten, führt zu einer Unterscheidung zwischen generischen und anwendungsabhängigen Standards. In Abbildung 14 ist eine Übersicht über die relevanten Sicherheitsnormen in diesem Zusammenhang dargestellt. Der generische Standard für Funktionale Sicherheit, die Basisnorm, ist die IEC 61508.

[18] SPIM ist unter http://spimsimulator.sourceforge.net verfügbar (Larus, 2018)

Davon gibt es zahlreiche Ableitungen für verschiedene Industrien. Für die Luftfahrt und die Kernenergie gibt es solche Regelwerke bereits seit Mitte der 1980er Jahre. Diese Normen haben allerdings keinen direkten Bezug zur IEC 61508. Der Vollständigkeit halber sind in der Übersicht die ISO IEC 12207 (Softwarelebenszyklus) (International Standard, 2008) sowie die ISO IEC 15504 (SPICE) (International Standard, 2004) enthalten. Diese Normen beinhalten zwar keine spezifischen Anforderungen für sicherheitsgerichtete Systeme bzw. deren Entwicklungsprozesse, sind aber auch im Safety-Umfeld Stand der Technik und bilden häufig die Basis für die Implementierung von geeigneten Prozessen.

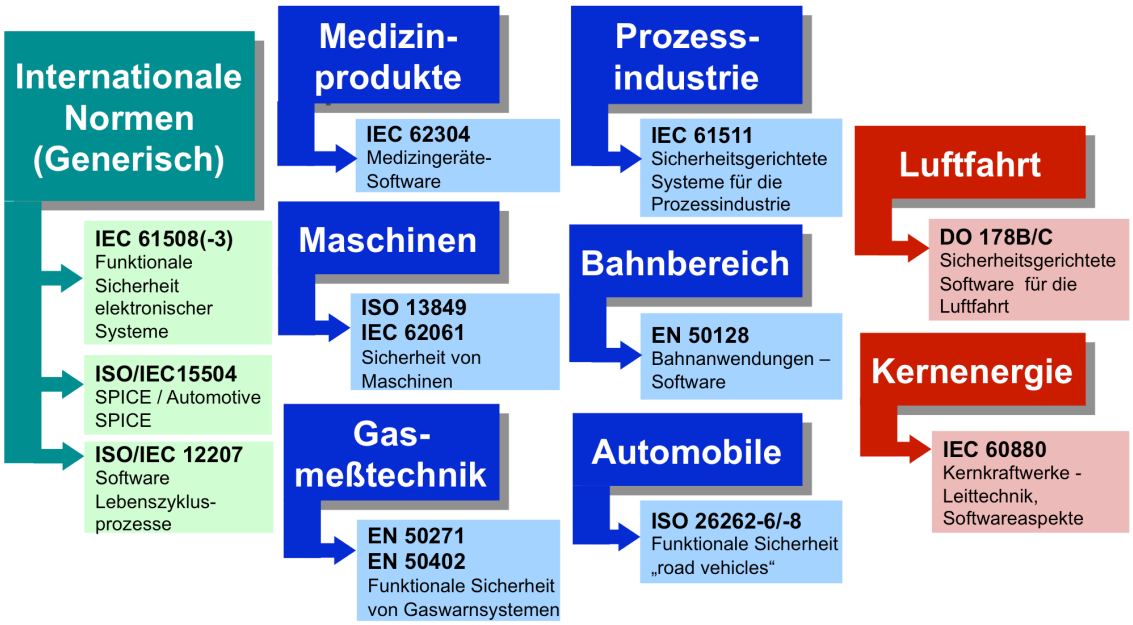

Abbildung 14: Übersicht über relevante Safety-Standards

Die IEC 61508 beschreibt die Anforderungen an die Entwicklung von sicherheitsgerichteten Systemen applikationsunabhängig. Das beinhaltet Anforderungen an die Prozesslandschaft (Funktional Safety Management), technische Anforderungen an die Hardware und Software sowie Maßnahmen und Methoden zur Sicherung der Integrität und Sicherheitsanalysen (wie z.B. FMEA, FMEDA, etc.). Die Anforderungen an den Softwareentwicklungsprozess überlappen sich mit den Anforderungen aus anderen gängigen Standards zur Softwareentwicklung, z.B. (Automotive-)SPICE.

Grundsätzlich muss in den einzelnen Standards zwischen zwei verschiedenen Anforderungsklassen unterschieden werden:

1. Anforderungen an den Entwicklungsprozess, also prozessuale Anforderungen und
2. technische Anforderungen, z.B. die Anwendung empfohlener Techniken und Maßnahmen.

Im Rahmen der Überarbeitung der ISO 26262 wurde im Unterarbeitskreis „Software und Prozesse“ des deutschen Normungskomitees[19] die erste Edition des Standards einer Analyse unterzogen, bei der ermittelt werden sollte, ob es sich bei den softwarebezogenen Anforderungen insbesondere in den Bänden 6 und 8 um prozessuale Anforderungen oder technische Anforderungen handelt. Das Ergebnis dieser Analyse der applikationsspezifischen Norm für den Automobilbereich ist beispielhaft für sämtliche Standards in Bereich der Funktionalen Sicherheit, wo die prozessualen Anforderungen dominieren. Wird berücksichtigt, dass Software nur systematische Fehler beinhalten kann, aber keine zufälligen, ist diese Gewichtung naheliegend. Die adäquate Maßnahme zur Vermeidung systematischer Fehler ist ein umfassend definierter und entsprechend geeignet implementierter (Software-)Entwicklungsprozess.

2.4.1 Normen zur Funktionalen Sicherheit

Wie in Abbildung 14 in Kapitel 2.4 dargestellt, gibt es Zusammenhänge und Gemeinsamkeiten zwischen den verschiedenen Sicherheitsstandards, auf die im folgenden Kapitel detailliert eingegangen wird. Die generische Grundnorm für Funktionale Sicherheit ist die IEC 61508, die explizit so verfasst ist, dass auf ihrer Basis applikationsspezifische Normen für verschiedene Domänen (z.B. Automotive, Medizin, Automatisierungstechnik, Maschinensicherheit, etc.) entstehen können.

In ihrer Historie älter sind die Standards für den Luftfahrtbereich (DO-178x) und für Kernenergie (IEC 60880), die keine direkte Verbindung zur IEC 61508 haben, aber für das in dieser Arbeit untersuchte Themenfeld „Anforderungen an die Verwendung von Softwareentwicklungswerkzeugen“ wertvolle Impulse liefern. In Kapitel 3.1 werden in diesem Zusammenhang ausgewählte Standards bzgl. ihrer Anforderungen an die Verwendung von Softwareentwicklungswerkzeugen untersucht.

[19] NA 052 DIN-Normenausschuss Automobiltechnik (NAAutomobil): NA 052-00-32-08-02 AK Arbeitskreis Software und Prozesse

2.4.1.1 Generische Grundnorm: IEC 61508

Die IEC 61508 (Deutsche Norm, 2010a) beschreibt generisch die Anforderungen an Sicherheitssysteme domänenunabhängig. Adressiert werden Geräte und Sensoren mit Mikroprozessoren für Sicherheitsaufgaben. Die IEC 61508 gilt deshalb als Grundnorm für verschiedene applikationsspezifische Sicherheitsstandards.

Die Norm besteht aus den folgenden 7 Bänden:

- Teil 1: Führt in das Konzept der Funktionalen Sicherheit ein und gibt einen Überblick über die Normen der Reihe IEC 61508.
- Teil 2: Anforderungen an sicherheitsbezogene elektrische/elektronische/programmierbare elektronische Systeme
- Teil 3: Anforderungen an Software
- Teil 4: Begriffe und Abkürzungen
- Teil 5: Beispiele zur Ermittlung der Stufe der Sicherheitsintegrität
- Teil 6: Anwendungsrichtlinie für Teil 2 und Teil 3
- Teil 7: Anwendungshinweise über Verfahren und Maßnahmen

In Band 3 werden Anforderungen an die Entwicklung von sicherheitsgerichteter Software, einschließlich der Anforderungen an den Softwareentwicklungsprozess und die dabei eingesetzten Werkzeuge definiert. Die IEC 61508 liegt seit 2010 in der Edition 2 vor. Die erste Veröffentlichung datiert aus dem Jahr 1998.

2.4.1.2 Automotive: ISO26262

Die ISO 26262 „Road vehicles – Functional safety“ (International Standard, 2011b) ist eine internationale Norm für sicherheitsrelevante elektrische/elektronische Systeme in Kraftfahrzeugen. Im Fokus der Norm liegen Fahrzeuge bis 3500 kg zulässiger Gesamtmasse, nicht jedoch Prototypen oder spezielle Fahrzeuge wie beispielsweise Umbauten für Behinderte. Die Norm ist eine Adaption der IEC 61508 an die spezifischen Gegebenheiten im Automobilbereich. Die ISO 26262 definiert ein Vorgehensmodell zusammen mit geforderten Aktivitäten und Arbeitsprodukten (Work Products) sowie anzuwendenden Methoden in Entwicklung und Produktion.

Die Arbeiten an diesem Standard begannen bereits im Jahr 2003 in Deutschland unter Koordination des Verbandes der deutschen Automobilindustrie (VDA) im damaligen VDA Arbeitskreis AK16. Die Bände 1 bis 9 wurden final am 14. November 2011 in Kraft

gesetzt. Der informative Band 10 folgte dann im Jahr 2012. Im Januar 2015 startete die dreijährige Überarbeitungsphase der ISO 26262, so dass die „Second Edition“ im Laufe des Jahres 2018 erwartet wird. Die Änderungen für die ISO 26262:2018 werden in dieser Arbeit nicht berücksichtigt.

Die zehn Bände der ISO 26262 haben folgende Inhalte:

- Band 1: Vokabular
- Band 2: Management der Funktionalen Sicherheit
- Band 3: Konzeptphase
- Band 4: Produktentwicklung: Systemebene
- Band 5: Produktentwicklung: Hardwareebene
- Band 6: Produktentwicklung: Softwareebene
- Band 7: Produktion, Betrieb und Außerbetriebnahme
- Band 8: Unterstützende Prozesse
- Band 9: ASIL- und sicherheitsorientierte Analysen
- Band 10: Guideline (nur informativ)

Die Anforderungen an den Softwareentwicklungsprozess werden in Band 6 definiert. Die Anforderungen an die bei der Entwicklung sicherheitsrelevanter Systeme eingesetzten Werkzeuge befinden sich in Band 8 Kapitel 11.

2.4.1.3 Gasmesstechnik: EN 50402 und EN 50271

Die Normen EN 50402 aus dem Jahr 2009 (in aktueller Version aus 2014) und EN 50271 aus dem Jahr 2002 (aktuelle Version aus 2010) formulieren Anforderungen an die Funktionale Sicherheit von Gasmessgeräten. Die EN 50271 ist eine harmonisierte Norm im Rahmen der ATEX-Direktive[20]. Die Anforderungen der Norm entsprechen dem Sicherheitsniveau SIL1 nach IEC 61508. Der Softwarelebenszyklus wurde an die in diesem Umfeld üblichen Entwicklungsmethoden angepasst. Dabei wurden ein pragmatisches Tailoring der Grundnorm und eine Ergänzung um die gasmesstechnikspezifischen Anforderungen eingearbeitet. Die EN 50402 ist auf Gasmesstechnik mit einem Sicherheitsniveau ab SIL2 ausgelegt, basiert ebenfalls auf der IEC 61508 und beinhaltet industrie- und applikationsspezifische Anforderungen.

[20] Richtlinie 94/9/EG des Europäischen Parlaments und des Rates vom 23. März 1994 zur Angleichung der Rechtsvorschriften der Mitgliedstaaten für Geräte und Schutzsysteme zur bestimmungsgemäßen Verwendung in explosionsgefährdeten Bereichen

In Deutschland wird diese Normung beim VDE/DKE im UK996.1[21] vorangetrieben.

2.4.1.4 Bahnanwendungen: EN 50128

Die EN 50128 (Deutsche Norm, 2001) aus dem Jahr 2001 gilt für jegliche sicherheitsrelevante Software der Eisenbahn und ist eine Ableitung der IEC 61508. In Deutschland hat das Eisenbahn-Bundesamt (EBA) mit seiner Verwaltungsvorschrift für die Abnahme von Eisenbahnfahrzeugen gemäß § 32 Abs. 1 EBO (Eisenbahn-Bundesamt, 2018) im Zuständigkeitsbereich des Eisenbahn-Bundesamt die Anwendung der EN 50128 ausdrücklich für sicherheitsrelevante Software an Bord der Eisenbahnfahrzeuge verbindlich gemacht. Die Anwendung der Norm ist in einem Leitfaden detailliert beschrieben (Pickert & Kohlschein, 2005). Die EN 50128 stellt Anforderungen an den Einsatz von Softwarewerkzeugen, die im weiteren Verlauf dieser Arbeit analysiert werden. Die erste Version der EN 50128 wurde im Jahr 2001 veröffentlicht. Aktuell liegt Version 2 aus dem Jahr 2012 vor.

2.4.1.5 Maschinensicherheit: EN ISO 13849 und IEC 62061

Die EN ISO 13849-1 (Deutsche Norm, 2008a) ist die zentrale Norm für sicherheitsgerichtete Steuerungen im Bereich „Maschinensicherheit". Die EN ISO 13849-1 (inzwischen in der aktualisierten Version 2009) wurde 2006 als Europäische Version verabschiedet und ist seit 2008 als deutsche Norm verfügbar, die EN 954-1 abgelöst hat. Außerdem ist die EN ISO 13849-1 als harmonisierte Norm unter der Maschinenrichtlinie im Amtsblatt der EU veröffentlicht (Europäische Kommission, 2015).

Die IEC 62061 (Deutsche Norm, 2005b) von 2005 stellt eine sektorspezifische Norm unterhalb der IEC 61508 dar. In ihr werden ebenfalls die Anforderungen an die Realisierung sicherheitsrelevanter elektrischer Steuerungssysteme von Maschinen beschrieben. Sie betrachtet den gesamten Lebenszyklus von der Konzeptphase bis zur Außerbetriebnahme. Basis bilden quantitative und qualitative Betrachtungen von Sicherheitsfunktionen. Sie richtet sich an Planer, Errichter und Nutzer sicherheitstechnischer Systeme.

Die EN IEC 62061 ist als harmonisierte Norm unter der EG-Maschinenrichtlinie gelistet, so dass auch bei ihrer Anwendung die so genannte „Vermutungswirkung" eintritt.

[21] DKE/UK 966.1: Mess- und Warngeräte für gefährliche Gase

2.4.1.6 Medizinanwendungen: IEC 62304

Die IEC 62304 (Deutsche Norm, 2007a) ist eine in Europa harmonisierte Norm für „Medizingeräte-Software“. Sie stellt Mindestanforderungen an die wichtigsten Prozesse des Softwarelebenszyklus in der Softwareentwicklung. Die IEC 62304 basiert auf der IEC 61508 und verweist in wesentlichen Teilen auf die Grundnorm. Der internationale Standard wurde 2006 veröffentlicht und bis 2016 einer Überarbeitung unterzogen.

2.4.1.7 Prozessindustrie: IEC 61511

Die Norm IEC 61511 (Deutsche Norm, 2005a) legt die Mindestanforderungen an sicherheitstechnische Systeme in der Prozessindustrie fest. Sie baut auf der IEC 61508 auf, wurde jedoch für die Prozessindustrie zugeschnitten und 2005 veröffentlicht. Der Standard gliedert sich wie folgt:

- Teil 1: Allgemeines, Begriffe, Anforderungen an Systeme, Software und Hardware
- Teil 2: Anleitungen zur Anwendung des Teils 1
- Teil 3: Anleitung für die Bestimmung der erforderlichen Sicherheits-Integritätslevel

2.4.1.8 Kernenergie: IEC 60880 und IEC 61513

Seit 2001 ist IEC 61513 „Kernkraftwerke - Leittechnik für Systeme mit sicherheitstechnischer Bedeutung - Allgemeine Systemanforderungen“ (Deutsche Norm, 2011) verfügbar. Im Jahr 2011 wurde deren Edition 2 veröffentlicht. Diese Norm steht im Zusammenhang zur Basisnorm IEC 61508 und beschreibt allgemeine Sicherheitsanforderung für die Leittechnik von Kernkraftwerken auf System- bzw. Gesamtarchitektur-Ebene. Da in diesem Standard keine detaillierten Anforderungen an den Softwareentwicklungsprozess definiert sind und somit auch keine Anforderungen an die bei der Entwicklung eingesetzten Werkzeuge ableitbar sind, wurde diese Norm im Rahmen dieser Arbeit nicht näher analysiert.

Das Zusammenspiel der einzelnen Normen im Umfeld der Leittechnik für Kernkraftwerke ist in Abbildung 15 dargestellt.

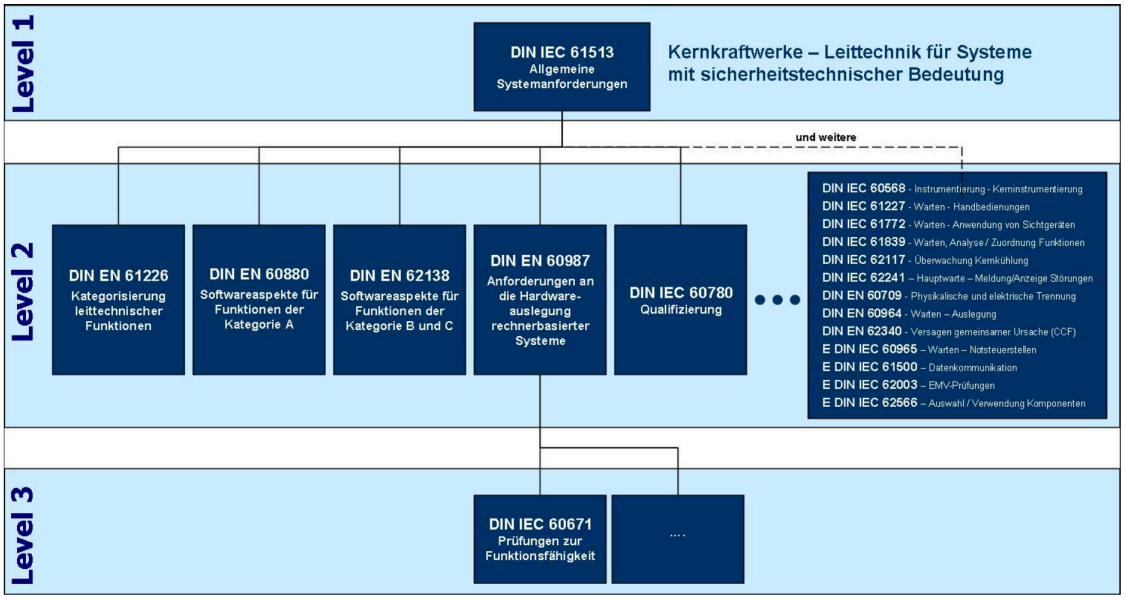

Abbildung 15: Zusammenhang der verschiedenen Sicherheitsnormen für Leittechnik in Kernkraftwerken nach (Bundesamt für Strahlenschutz, 2011)

Die Softwareaspekte werden in der IEC 60880 (Deutsche Norm, 2007b) für die Sicherheitskategorie A und der IEC 62138 (Deutsche Norm, 2010c) für die Kategorien B und C betrachtet. Die Kategorisierung erfolgt anhand der IEC 61226 (Deutsche Norm, 2010b), wobei die Kategorie A die höchste Kritikalität abbildet (World Nuclear Association, 2015).

Im Rahmen dieser Arbeit wird die IEC 60880 exemplarisch für diese Industrie bzgl. ihrer Anforderungen an Softwarewerkzeuge untersucht. Die IEC 60880 und die IEC 62138 stehen in der Normenpyramide in direktem Bezug zur IEC 61513. Die IEC 60880 liegt aktuell in der Edition 2 aus dem Jahr 2006 vor. Der erste Stand wurde 1986 veröffentlicht.

2.4.1.9 Luftfahrt: DO-178B und DO-178C

Die Norm DO-178B (International Standard, 1992) wurde im Dezember 1992 als Weiterentwicklung der DO-178A (International Standard, 1985) aus dem März 1985 veröffentlicht. Die Nachfolgenorm DO-178C / ED-12C (International Standard, 2012) ist seit Januar 2012 verfügbar.

Der Standard thematisiert die Anforderungen an die Softwareentwicklung im sicherheitskritischen Bereich der Luftfahrt (Software Considerations in Airborne Systems).

Entwickelt wurde die Norm von RTCA[22]. Durch die EUROCAE[23] wurde der Standard in den europäischen Geltungsbereich als ED-12B übernommen. Die amerikanische Luftfahrtbehörde FAA[24] und die europäische EASA[25] fordern für den Softwareentwicklungsprozess die Erfüllung der DO-178B/C.

Während im Luftfahrtbereich nur komplette Flugzeuge zertifiziert werden, werden (Software-)Komponenten einer Qualifizierung unterzogen. Voraussetzung für eine Qualifizierung von Software für den Einsatz in der Luftfahrt ist ein Sicherheitsnachweis auf Basis einer hinreichend vollständigen Dokumentation (Safety Case).

Entsprechend der Sicherheits-Integritäts-Level (SIL) in der IEC 61508 gibt es in der Luftfahrt die fünf Design Assurance Level (DAL) A bis E.

Schon in der DO-178B gibt es detaillierte Anforderungen an Softwareentwicklungswerkzeuge und deren Qualifizierung, die im Rahmen dieser Arbeit untersucht wurden.

2.4.2 Allgemeine Software Standards

Nach (Ross, 2014) ab Seite 23 kann die historische Entwicklung der in der Softwareentwicklung verwendeten Prozess- bzw. Vorgehensmodelle in 9 Phasen unterteilt werden:

1. Erster Ansatz zur Entwicklung übersichtlicher Programme (Dijkstra, 1968)
2. Entwicklung von Software-Engineering-Prinzipien (1968-74)
3. Entwicklung von phasenspezifischen Software-Engineering-Methoden (1972-1975)
4. Entwicklung von phasenspezifischen Werkzeugen (1972-1975)
5. Entwicklung von phasenübergreifenden (integrierten) Software-Engineering-Methoden (ab 1980)
6. Entwicklung von phasenübergreifenden (integrierten) Werkzeugen (ab 1980)
7. Definition verschiedener, konkurrierender objektorientierter Methoden (OO) (ab 1990)

[22] Die Radio Technical Commission for Aeronautics (RTCA) mit Sitz in Washington D.C. (USA) gibt Empfehlungen im Bereich der Kommunikation, Navigation und Überwachung des Flugverkehrsmanagements.

[23] Die European Organization for Civil Aviation Equipment (EUROCAE) ist die europäische Entsprechung der RTCA mit Sitz in Malakoff (Frankreich).

[24] Federal Aviation Administration (FAA), Bundesluftfahrtbehörde der USA mit Sitz in Washington D.C. (USA)

[25] Die European Aviation Safety Agency (EASA) ist die Flugsicherheitsbehörde der Europäischen Union für die zivile Luftfahrt mit Sitz in Köln (Deutschland).

8. Integration der OO-Methoden zur UML (Unified Modeling Language) (ab 1995)
9. UML 2.0 (aktuelle Version 2.5 vom März 2015 (Object Management Group, 2015))

Wird die Entwicklung über diese einzelnen Phasen betrachtet, zeigt sich, dass am Anfang die Fehlerarmut bei der Implementierung von Software im Fokus stand. Im Laufe der Zeit spielt aber der Aspekt „Umgang mit komplexen Systemen“ zunehmend eine Rolle. Stand der Technik heute ist eine integrierte Prozess- und Werkzeugkette, die alle Phasen des Softwareentwicklungsprozesses abdeckt. Die gängigen Prozessmodelle werden in den folgenden Kapiteln eingeführt.

2.4.2.1 V-Modell XT

Das V-Modell (Bröhl & Dröschel, 1995) ist ursprünglich als ein Vorgehensmodell speziell in Bereich der Entwicklungsprozesse für Software (Software Engineering) im Auftrag des Bundes konzipiert (siehe (Diesterer, Fels, & Hausotter, 2003) Seite 543ff) worden und eignet sich darüber hinaus hervorragend für die Projektplanung und Projektdurchführung. Es ist aus dem Wasserfallmodell durch Integration der Qualitätssicherung entwickelt worden. Grundgedanke des V-Modells ist es, den Entwicklungsprozess in definierte Phasen zu gliedern. Das V-Modell impliziert somit ein methodisches, phasenorientiertes Vorgehen mit integrierter Qualitätssicherung und den dazugehörigen Testmaßnahmen. Der grundlegende Aufbau des V-Modells ist aus der Abbildung 16 zu ersehen.

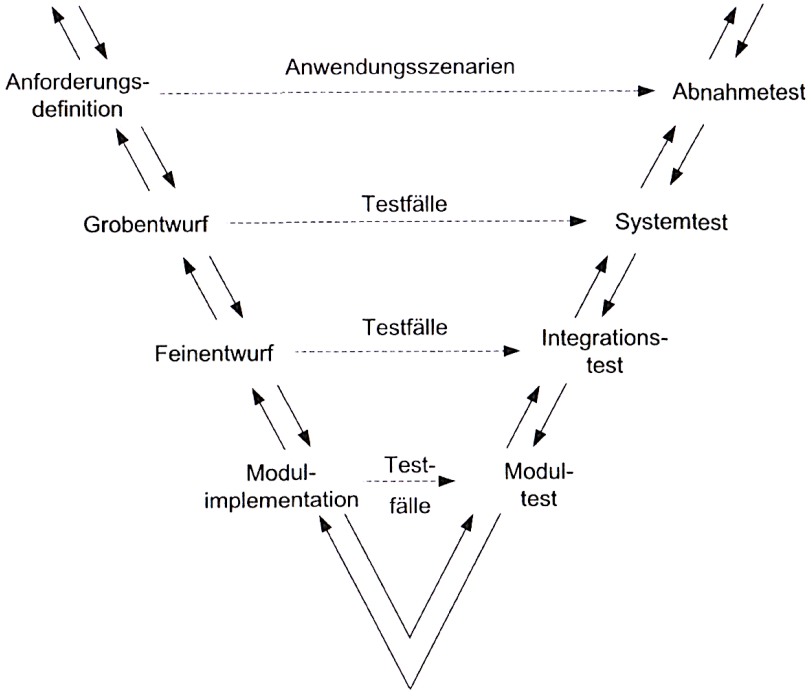

Abbildung 16: V-Modell nach (Diesterer et al., 2003) **S. 543**

Zu erkennen ist auf der linken Seite des Modells eine immer tiefere Detaillierung, bis letztlich in der Spitze des V die Implementierung erfolgt. Auf der rechten Seite wird diese Implementierung entsprechenden den links vorgenommenen Spezifikationen getestet. Mit den Tests lassen sich phasenbezogene Ergebnisse abheben, die gemäß Ergebnis entsprechende Vorgaben für die darunter liegende Projektphase enthalten.
Den Projektphasen stehen die jeweiligen Testphasen gegenüber. Das V-Modell ist allgemeingültig verfasst (Friedrich, Kuhrmann, Sihlin, & Hammerschall, 2009) und ist immer den konkret gegebenen Projektbedingungen anzupassen. Die aus dem wehrtechnischen Vorgehensmodell des Bundes abgeleitete „zivile“ Version mit ihren Entwicklungsphasen ist in der Abbildung 17 dargestellt. Es wird für die Anwendung des Modells davon ausgegangen, dass Softwareprodukte auf der Basis einer detaillierten Beschreibung der Anforderungen entwickelt werden.

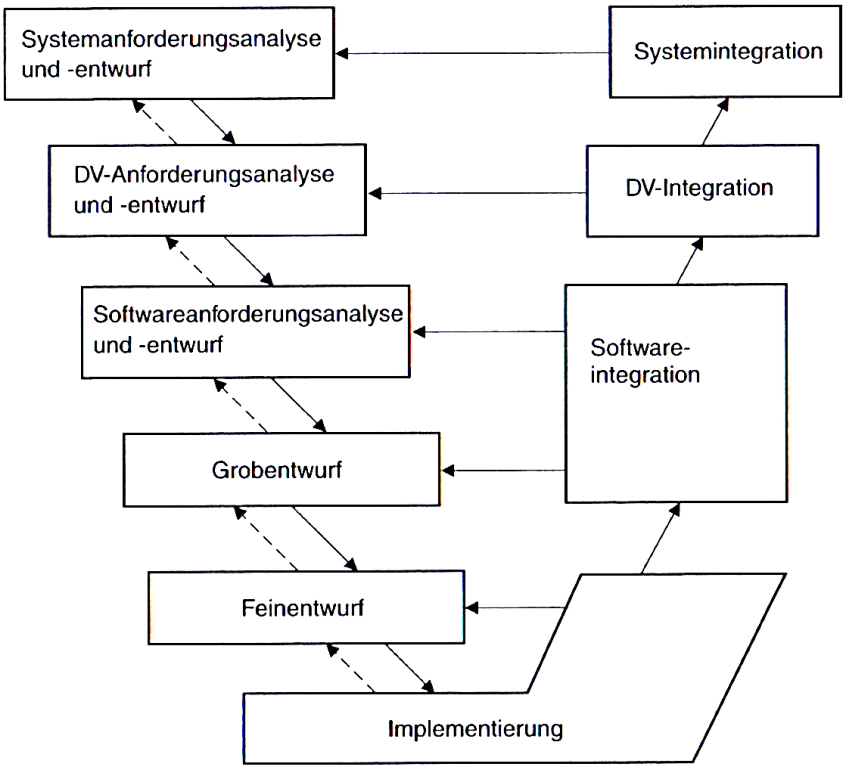

Abbildung 17: V-Modell des Bundes aus (U. Schneider & Werner, 2001) **Seite 333**

Das V-Modell XT ist eine Weiterentwicklung des 1997 veröffentlichten V-Modell 97 und besteht aus 8 Bänden (siehe Abbildung 18). Die aktuelle Version des V-Modell XT ist Version 1.4 (IT-Beauftragte der Bundesregierung, 2012), die 2012 veröffentlicht wurde.

Die in dem V-Modell enthaltenen vier Submodelle:

- Systemerstellung (SE)
- Qualitätssicherung (QS)
- Konfigurationsmanagement (KS)
- Projektmanagement (PM)

werden im Folgenden nach den Vorgaben des V-Modells XT auf die lösende Aufgabe bezogen betrachtet.

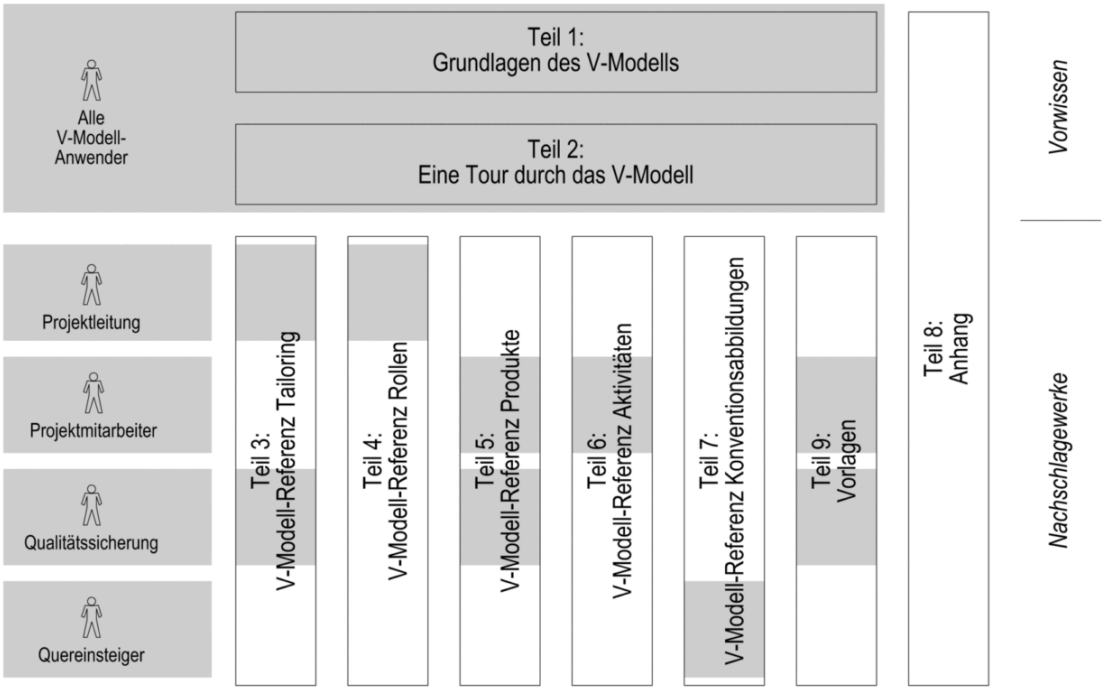

Abbildung 18: Aufbau des V-Modell XT 1.3 nach (IT-Beauftragte der Bundesregierung, 2006)

Das V-Modell hat folgende Zielsetzungen (IT-Beauftragte der Bundesregierung, 2006) ab Seite 1-7:

- Minimierung der Projektrisiken
- Verbesserung und Gewährleistung der Qualität
- Eindämmung der Gesamtkosten über den gesamten Projekt- und Systemlebenszyklus
- Verbesserung der Kommunikation zwischen allen Beteiligten

Das V-Modell definiert Rollen und Anforderungen an die Entwicklungsprozesse für komplexe Projekte. Für die Abwicklung öffentlicher Aufträge ist das V-Modell Standard in Deutschland und ist über das Bundesverwaltungsamt - Bundesstelle für

Informationstechnik (BIT)[26] verfügbar. Die „Entwicklungsstandards für IT-Systeme des Bundes“ wurden auf Basis des V-Modells entwickelt[27]. Mittlerweile hat sich das V-Modell zum Industriestandard entwickelt und zahlreiche Normen beeinflusst, die Entwicklungsprozesse adressieren. Somit ist das V-Modell XT kompatibel zu Standards wie ISO 9001 (Deutsche Norm, 2008b), CMMI und SPICE.

Seine Ursprünge hat das V-Modell bereits Ende der 1970er Jahre. Die erste Version wurde 1986 von der damals bundeseigenen IABG[28] entwickelt und war initial für IT-Projekte der öffentlichen Hand vorgesehen.

2.4.2.2 Softwarelebenszyklus: ISO IEC 12207

Die ISO IEC 12207 (International Standard, 2008) erschien in ihrer ersten Ausgabe im Jahr 1995 und adressiert Anforderungen an den Softwarelebenszyklus („Systems and software engineering — Software life cycle processes“). Aktuell liegt der Standard in der Ausgabe von 2008 vor.

Die in der ISO IEC 12207-2008 definierten Prozesse sind:

1. Einigungsprozesse (Agreement Processes)
2. Organisatorische Prozesse zur Projektunterstützung (Organizational Project-Enabling Processes)
3. Projektbezogene Prozesse (Project Processes)
4. Technische Prozesse (Technical Processes)
5. Software-Entwicklungsprozesse (Software Implementation Processes)
6. Software-Unterstützungsprozesse (Software Support Processes)
7. Software-Wiederverwendungsprozesse (Software Reuse Processes)

Ähnlich wie beim V-Modell werden folgende Inhalte adressiert: Dokumentation, Konfigurationsmanagement, Qualitätssicherung, formale Überprüfung der Prozesse (Verifikation), inhaltliche Überprüfung (Validierung), Review, Audit und Problembehandlung.

[26] http://www.bva.bund.de/

[27] geregelt in den Kriterien für Ausschreibungen und Bewertungen von IT-Leistungen des Beschaffungsamtes des Bundesministeriums des Innern (UfAB IV) (Beschaffungsamt des Bundesministeriums des Innern, 2015)

[28] Industrieanlagen-Betriebsgesellschaft mbH, http://www.iabg.de

2.4.2.3 Reifegradmodelle: CMMI, ISO IEC 15504 (SPICE) und Automotive-SPICE, ISO IEC 3300x-Normenreihe

Mit Reifegradbewertungen kann die Fähigkeit einer Entwicklungsorganisation beurteilt werden, qualitativ hochwertige und möglichst fehlerarme Software zu entwickeln. Dies wird in der Praxis z.B. bei der Bewertung und Auswahl von Zulieferfirmen genutzt. Ein Lieferant mit einem höheren Reifegrad liefert also mit einer höheren Wahrscheinlichkeit fehlerarme Software. Die gängigen Reifegradmodelle werden im Folgenden eingeführt.

CMMI (CAPABILITY MATURITY MODEL INTEGRATION)

CMMI (CMMI Product Team, 2010) ist ein Modell zur Reifegradbewertung von Entwicklungsprozessen und unterscheidet sich somit von den Prozessmodellen wie der ISO IEC 12207 (International Standard, 2008) (siehe Kapitel 2.4.2.2). Es dient maßgeblich zur Bewertung der Prozesslandschaft, aber auch zur Identifikation von Verbesserungspotentialen.

Das Modell geht auf eine Initiative des Verteidigungsministeriums der USA zurück und wurde initial am Software Engineering Institute (SEI) an der Carnegie Mellon University in Pittsburgh (USA) im Jahr 1986 entwickelt. Aktuell liegt der Stand 1.3 aus dem Jahr 2010 vor.

Im CMMI Modell gibt es folgende Prozessbereiche (Process Areas):

- Projektmanagement (Project Management) bzw. Arbeitsmanagement (Work Management)
- Unterstützung (Support)
- Prozessmanagement (Process Management).

Die einzelnen Prozessbereiche werden unabhängig voneinander einer Bewertung unterzogen. Dabei werden folgende Bewertungskriterien angewendet:

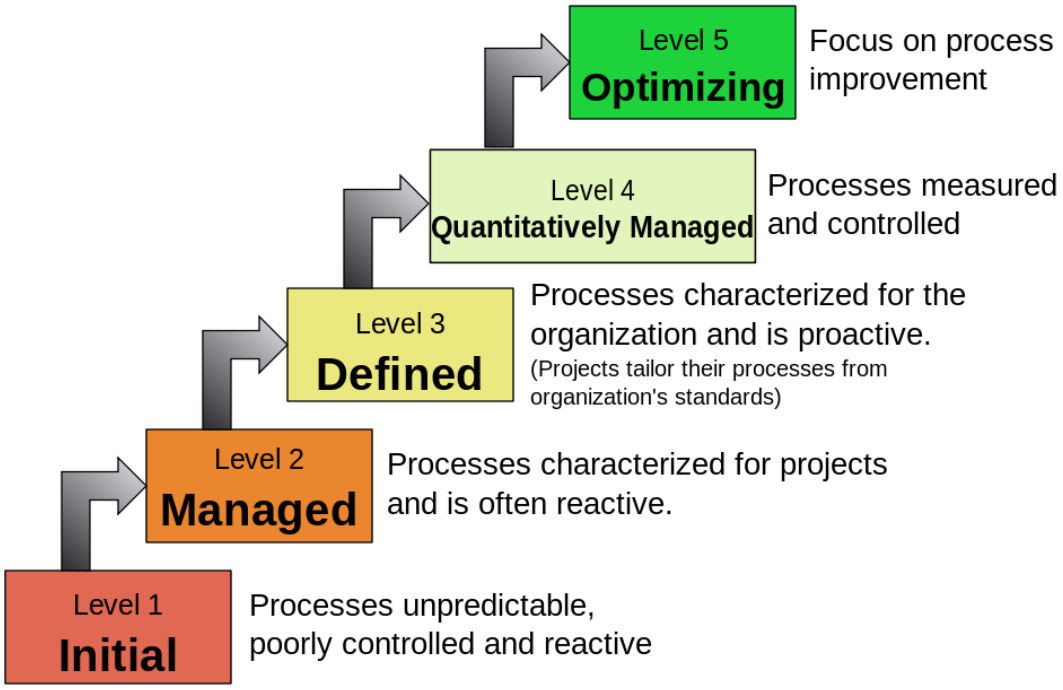

Abbildung 19: Zusammenhang der CMMI Level nach (Godfrey, 2008)

Für die einzelnen Level gelten folgende Kriterien:

Level 1 – Initial: Es gibt keine spezifischen Anforderungen, d.h. Level 1 kann für jede Organisation initial angenommen werden.

Level 2 – Managed: Die Projekteabwicklung wird gesteuert. Somit ist es möglich, vergleichbare Projekte erfolgreich zu wiederholen.

Level 3 – Defined: Es gibt einen etablierten Standardprozess, nach dem die Projekte durchgeführt werden, und der angepasst werden kann (Tailoring). Darüber hinaus gibt es eine organisationsweite kontinuierliche Prozessverbesserung.

Level 4 – Quantitatively Managed: Es gibt eine statistische Prozesskontrolle, mit der die Prozessgüte gemessen werden kann.

Level 5 – Optimizing: Auf Basis der etablierten statistischen Prozesskontrolle werden die Prozesse kontinuierlich verbessert.

ISO IEC 15504 (SPICE) UND AUTOMOTIVE-SPICE

Die ISO IEC 15504 (International Standard, 2004) bzw. SPICE (Software Process Improvement and Capability Determination) ist ein weiterer internationaler Standard für eine Prozessreifegradbewertung und Prozessverbesserung, der 2004 veröffentlicht und 2015

durch die IEC 3300x-Normenreihe (International Standard, 2015a) ersetzt wurde. Der Standard besteht aus 10 Bänden:

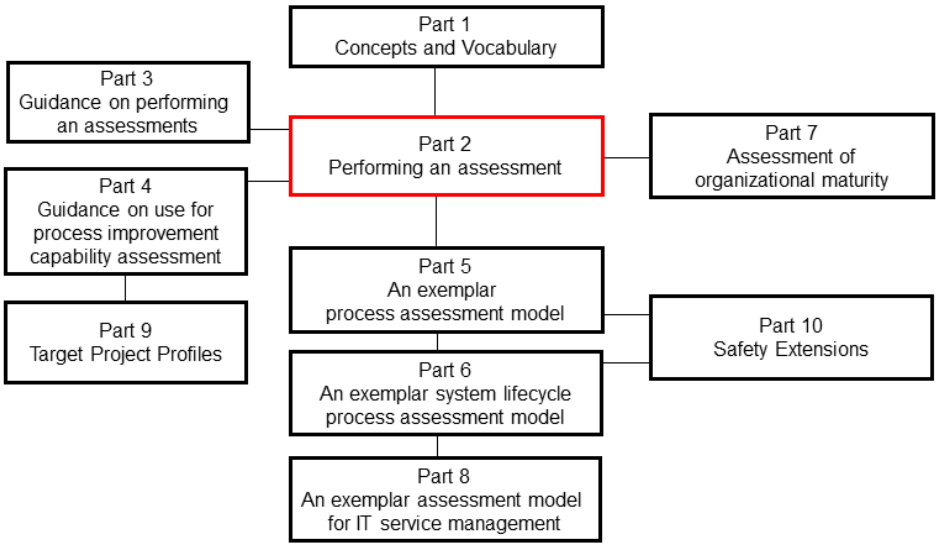

Abbildung 20: Aufbau der Normenreihe ISO IEC 15504 (International Standard, 2004)

Im Rahmen dieser Arbeit sind speziell die Bände 2 bis 5 relevant:

- ISO IEC 15504 Part 2 – „Performing an assessment“: In diesem Band werden die Anforderungen für die Durchführung eines Assessments festgelegt.
- ISO IEC 15504 Part 3 – „Guidance on performing an assessment“: Band 3 definiert die minimalen Inhalte des Teils ISO IEC 15504-2, die für die Einhaltung der Norm erreicht werden müssen. Darüber hinaus gibt es einen Vorschlag für einen Assessment-Prozess.
- ISO IEC 15504 Part 4 – „Guidance on use for process improvement and process capability determination“: Band 4 adressiert die Prozesse für die Prozessverbesserung (Process Improvement, PI) und die Ermittlung der Prozessfähigkeitsgrade (Process Capability Determination, PCD). Dafür wird ein ähnlicher levelbasierter Ansatz verwendet wie bei CMMI.
- ISO IEC 15504 Part 5 – „An exemplar software life cycle process assessment model“: Im fünften Teil wird das Prozess-Assessment-Modell definiert. Dabei wird für das Prozessreferenzmodell die Prozessdefinition der ISO IEC 12207:2008 angewendet.

Der Band 2 ist im Rahmen dieser Arbeit von zentraler Bedeutung, da hier die Anforderungen an die Durchführung von Assessments definiert werden.

Automotive-SPICE ist eine Untermenge des SPICE-Standards mit dem speziellen Fokus auf der Automobilindustrie. Die Standardisierungsaktivitäten werden durch einen Arbeitskreis des Verbandes der deutschen Automobilindustrie (VDA) koordiniert. Im Automobilumfeld kann Automotive-SPICE als de-facto-Standard für die Entwicklung komplexer elektronischer Systeme angesehen werden. Die 1. Ausgabe des „Process Assessment Model“ (PAM) wurde im Jahr 2008 vorgestellt (VDA, 2012). Die aktuelle Ausgabe ist die Version 3.0 des PAM aus dem Jahr 2015 (VDA, 2015). Dabei bezieht sich das PAM 3.0 bereits auf die IEC 3300x-Normenreihe.

Die folgende Abbildung 21 zeigt das Tailoring der zu betrachtenden Prozessumfänge, die für die Bewertung eines reinen Softwareproduktes (z.B. eines Softwareentwicklungswerkzeuges) im Fokus stehen. Deshalb können die Schnittstellen zur System- und Hardwareebene ebenso vernachlässigt werden, wie die Prozesse zum Lieferantenhandling. Im Wesentlichen stehen die Engineering-Prozesse (SWE) sowie die Unterstützungsprozesse (SUP) und das Projektmanagement (MAN) in Zentrum der Betrachtung.

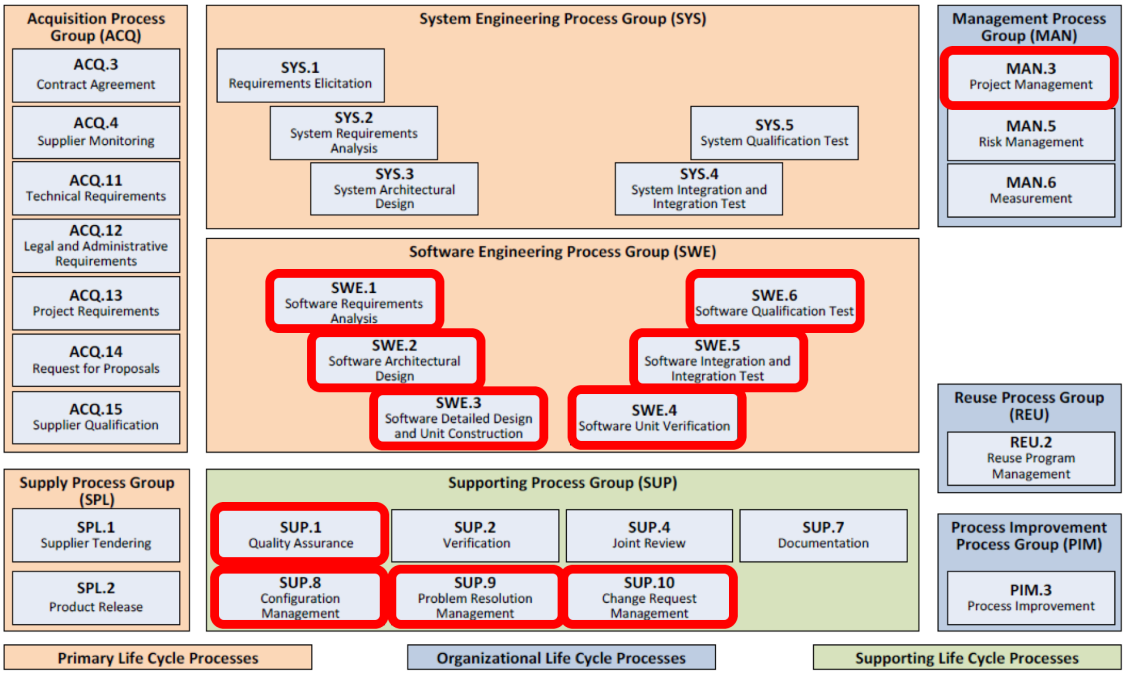

Abbildung 21: Betrachtungsumfang der Softwareanteile im Automotive-SPICE PAM 3.0 (VDA, 2015)

Dieser Umfang entspricht im Wesentlichen dem sogenannten „HIS-Scope“[29]. Damit werden die Anforderungen an Softwareprodukte als Teilmenge abgegrenzt.

ISO IEC 3300X-NORMENREIHE

Die ISO IEC 3300x-Normenreihe ist die Überarbeitung der ISO IEC 15504. Die Standards wurden im Jahr 2015 verabschiedet und gliedern sich wie folgt:

- ISO IEC 33001:2015 Concepts and terminology (International Standard, 2015a)
- ISO IEC 33002:2015 Requirements for performing process assessment (International Standard, 2015b)
- ISO IEC 33003:2015 Requirements for process measurement frameworks (International Standard, 2015c)
- ISO IEC 33004:2015 Requirements for process reference, process assessment and maturity models (International Standard, 2015d)
- ISO IEC 33014:2015 Guide for process improvement (International Standard, 2015e)
- ISO IEC 33020:2015 Process measurement framework for assessment of process capability (International Standard, 2015f)
- ISO IEC 33063:2015 Process assessment model for software testing (International Standard, 2015g)

Der Zusammenhang zwischen den einzelnen Standards der Normenreihe ist in Abbildung 22 schematisch dargestellt:

[29] HIS-Scope: „Hersteller Initiative Software“, Vereinigung der deutschen Automobilhersteller (Audi, BMW, Daimler, Porsche und Volkswagen), die Interpretationen gängiger Standards für Automotive-Anwendungen erarbeitet. Beim sogenannten HIS-Scope handelt es sich um ein Subset von Automotive-SPICE.

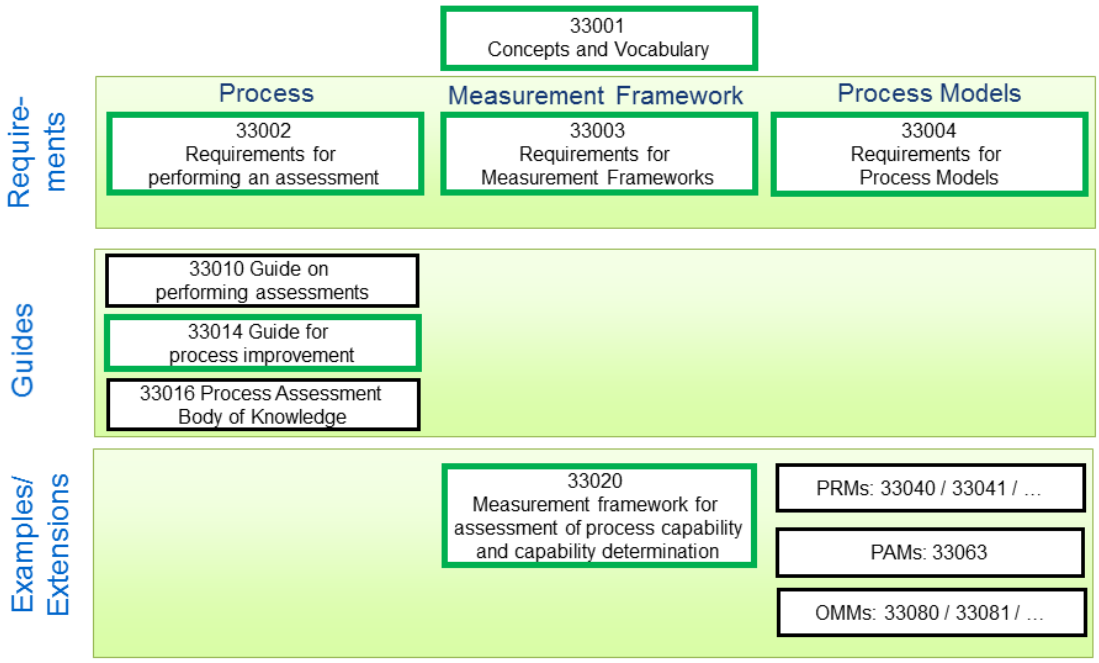

Abbildung 22: Aufbau der ISO IEC 330xx Normenreihe (Bildquelle: Automotive-SPICE PAM 3.0 (VDA, 2015)**)**

Als im Jahr 2015 der Übergang von der ISO IEC 15504 erfolgte, wurde dieser Wechsel auch im Rahmen von Automotive-SPICE beim Übergang vom Process Assessment Model 2.5 zu 3.0 vollzogen (Hindel & Sechser, 2015).

Ein wichtiger Aspekt ist auch in diesem Kontext eine Bewertung der Reifegrade der implementierten Prozesse, wobei das Vorgehen prinzipiell dem der ISO IEC 15504 ähnlich ist (Lami, Fabbrini, & Buglione, 2014).

Die IEC 3300x-Normenreihe wurde im Rahmen dieser Arbeit keiner detaillierten Betrachtung und Analyse unterzogen.

2.4.3 Agile Methoden

Im Bereich der kommerziellen Softwareentwicklung halten zunehmend agile Methoden (z.B. Scrum) Einzug. Die Herausforderung besteht darin, bei sicherheitsgerichteten Entwicklungen den agilen Ansatz mit den normativen Anforderungen zu erfüllen, die eher einen klassischen Entwicklungsansatz (z.B. auf Basis des V-Modells) verfolgen. Grundsätzlich ist ein Brückenschlag zwischen beiden Ansätzen möglich, wie z.B. in (Canditt, Rauh, & Wittmann, 2010) beschrieben. Dabei können die Phasen mit den dahinterliegenden Prozessen mittels Tailoring aus dem Ansatz des V-Modell XT abgeleitet werden. Die Umsetzung der einzelnen Phasen kann dann mit agilen Methoden erfolgen (siehe Abbildung 23).

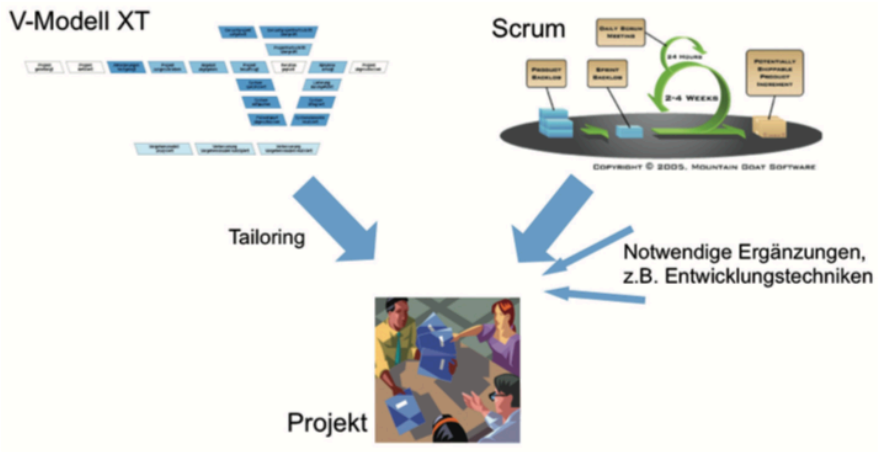

Abbildung 23: Anwendung verschiedener Vorgehensweisen im Projekt aus (Canditt et al., 2010)

Folgende grundsätzliche Vorgehensweise ist möglich: Im ersten Schritt entsteht aus den Kundenanforderungen das Lastenheft, welches zusammen mit den internen Anforderungen bei der Entwicklungsorganisation zum Pflichtenheft verfeinert wird. Die Anforderungen des Pflichtenheftes können anschließend in einen Product Backlog überführt werden. In den einzelnen Sprints werden die sogenannten Sprint Backlogs aufgebaut. Die Definition of Done beschreibt die Abnahmekritierien, mit denen der Sprint abgeschlossen bzw. die nächste Phase im V-Modell bearbeitet werden kann.

Die Integration von agilen Methoden in einen klassischen Entwicklungsansatz nach dem Vorbild des V-Modells ist im Umfeld von sicherheitsgerichteten Entwicklungen möglich, aber nicht trivial. In der Normung der Sicherheitsstandards wurde dies bisher nicht explizit berücksichtigt. Hier lassen sich zahlreiche weitere Forschungsfragen ableiten, die nicht Gegenstand dieser Arbeit sind.

2.5 Überschneidungen der Prozessanforderungen verschiedener Standards (Safety und Non-Safety)

Bei der Entwicklung von softwarebasierten eingebetteten Systemen sind vielfältige Anforderungen zu erfüllen. Es ist industrieübergreifend Stand der Technik, dass bestimmte Maßnahmen zur Qualitätssicherung und des Qualitätsmanagements implementiert werden müssen. Dafür gibt es verschiedene Standards, generische wie ISO 9001, aber auch Standards, die spezifisch den Reifegrad der Qualitätsmanagementsysteme (QMS) messen können, wie SPICE und CMMI. Üblicherweise beherrschen die Unternehmen

diesen Part sehr gut, da Qualität eine generelle Anforderung ist, unabhängig davon, ob es sich um ein sicherheitsgerichtetes System handelt.

Da es zwischen hoher Produktqualität und Sicherheit direkte Zusammenhänge gibt, werden wesentliche Anforderungen aus den QM-Standards in der Sicherheitstechnik aufgegriffen. Es gibt also signifikante Überlappungen zwischen Qualitätsanforderungen und Anforderungen an die Funktionale Sicherheit.

2.5.1 Integrale Betrachtung der Prozesslandschaft für Funktionale Sicherheit und (Automotive-)SPICE

Es ist Stand der Technik, dass bei der Entwicklung von eingebetteten Systemen die Ausprägung der Entwicklungsprozesse und deren Umsetzung auf Projektebene von besonderer Bedeutung ist. Letztlich ist das strukturierte Entwicklungsvorgehen ein wesentlicher Faktor zur Qualitätssicherung des Produktes. Mit zunehmender Komplexität der Software in Systemen verstärkt sich die Bedeutung dieses Aspektes noch zusätzlich. In Anhängigkeit von der Industriebranche gibt es verschiedene etablierte Verfahren. Die bedeutendsten sind CMMI und SPICE (siehe Kapitel 2.4.2.3). In der Automobilindustrie ist Automotive-SPICE das Standardvorgehen, ein auf die Anforderungen der Automobilindustrie zugeschnittenes Subset.

Da parallel zum Entstehen dieser Arbeit zahlreiche Veröffentlichungen zu dieser Problemstellung erschienen sind, die Vorschläge für einen integralen Ansatz der Prozessbewertung machen, wurde dies aus dem Fokus des Forschungsschwerpunktes dieser Arbeit genommen. Anstelle dessen wurden die aktuellen Voröffentlichungen aufgegriffen und bei dem in diesem Kapitel vorgestellten Ansatz angewendet. Stellvertretend sind die Arbeiten von Messnarz et al. (Messnarz et al., 2010), Dold et al. (Dold et al., 2012) und die Ergebnisse des SoQrates Projektes[30] (Messnarz et al., 2011) zu nennen.

Im folgenden Kapitel wird beschrieben, wie ein solcher Ansatz basierend auf den eben genannten Arbeiten aussehen kann. Dies wird am Beispiel einer in 2014 durchgeführten integralen Prozessbewertung in einem kommerziellen Projekt erläutert.

Auch wirtschaftlich betrachtet, ist ein integriertes Assessment sehr sinnvoll. Es gibt eine signifikante Überschneidung bei den Inhalten der Assessments (siehe Kapitel 2.5.1.1). Ein geeignetes Assessoren-Team, das die entsprechende Qualifikation besitzt, sowohl

[30] Software Quality Rates for Maturity (http://soqrates.eurospi.net)

die Aspekte der Funktionalen Sicherheit als auch die des Qualitätsmanagements kompetent zu bewerten, kann den Ablauf des Assessments in Bezug auf Zeit und Kosten deutlich effizienter gestalten.

2.5.1.1 Unterschiede und Gemeinsamkeiten

Bei der detaillierten Analyse der Anforderungen aus SPICE und den Standards der Funktionalen Sicherheit (z.B. ISO 26262) ergeben sich deutliche Überschneidungen bei den prozessualen Anforderungen (siehe Abbildung 24):

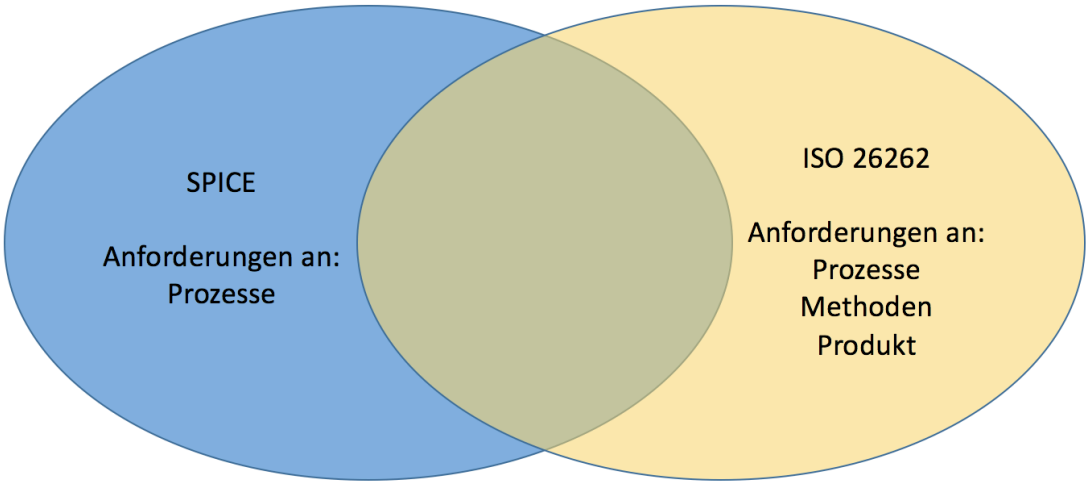

Abbildung 24: Überschneidungen der Anforderungen zwischen SPICE und der ISO 26262 in Anlehnung an (Dold et al., 2012)

Werden die Schwerpunkte von Automotive-SPICE und der ISO 26262 gegenübergestellt, ist ersichtlich, dass zu fast jedem Prozessbereich in SPICE Anforderungen in der ISO 26262 zu finden sind, die teilweise auch über die Forderungen von SPICE deutlich hinausgehen, z.B. beim Management der Funktionalen Sicherheit (FSM) (siehe Tabelle 4). Dieser Ansatz referenziert die Gliederung des PAM 2.5 von Automotive-SPICE, ist aber grundsätzlich im PAM 3.0 genauso anwendbar.

Tabelle 4: Gegenüberstellung der Schwerpunkte von Automotive-SPICE (am Beispiel des PAM 2.5) und den prozessualen Anforderungen der ISO 26262

Automotive-SPICE	ISO 26262
MAN.3 Project management	-
(ACQ.4 Supplier monitoring)	Interfaces within distributed developments
-	Overall safety management
-	Management after release
-	Item definition
-	Initiation of safety lifecycle
-	Hazard analysis & risk assessment
ENG.1 Requirements elicitation	Functional safety concept
-	Initiation of product development at system level
ENG.2 System requirements analysis	Specification of technical safety requirements
ENG.3 System architectural design	System design
-	Initiation of product development at the software level
ENG.4 Software requirements analysis	Specification of software safety requirements
ENG.5 Software design	Software architectural design
ENG.6 Software construction	Software unit design and implementation
- (included with ENG 6)	Software unit testing
ENG.7 Software integration test	Software integration and testing
ENG.8 Software testing	Verification of software safety requirements
-	Software Configuration
ENG.9 System integration test	Item Integration and testing
ENG.10 System testing	Safety Validation
-	Functional Safety Assessment
-	Release for Production
SUP.8 Configuration management	Configuration Management
SUP.10 Change request management	Change Management
-	Documentation
-	Confidence in the use of software tools
-	Qualification of software components
-	Proven in use argument
-	Qualification of hardware components

Da in SPICE nur die prozessualen Aspekte adressiert werden, sollte also SPICE als „roter Faden“ für ein integriertes Prozess-Assessment dienen. Der SPICE-typische Fragenkatalog muss dann um die Aspekte der Funktionalen Sicherheit ergänzt werden. Der Schwerpunkt liegt dabei auf dem Handling der Sicherheitsanforderungen, beginnend bei der Gefährdungsanalyse über die Sicherheitsanforderungsspezifikation bis hin zur Validierung der Funktionalen Sicherheit.

2.5.1.2 Integraler Ansatz

Der Ansatz, das Assessment zu integrieren, wird im Folgenden am Beispiel ENG.4 (Softwareanforderungsanalyse) von Automotive-SPICE (PAM 2.5) beschrieben[31].

Im ersten Schritt werden die Anforderungen an die erfolgreiche Implementierung der Prozesse nach SPICE als Basis genommen. Diese Anforderungen bilden einen Großteil der Prozessanforderungen aus der Perspektive der Funktionalen Sicherheit ebenfalls ab. Im zweiten Schritt werden diese Basisanforderungen um die spezifischen Themen der ISO 26262 ergänzt (siehe Tabelle 5).

Tabelle 5: Ergänzung der Safety-Aspekte bei den Prozessanforderungen von ENG.4 nach Automotive-SPICE (am Beispiel des PAM 2.5)

Prozess
Der Zweck des Softwareanforderungsanalyse-Prozesses besteht darin, die Softwareanforderungen der Softwareelemente des Systems zu ermitteln.

Ergänzung Safety:
Sicherheitsanforderungen, die durch Software realisiert werden, werden aus dem technischen Sicherheitskonzept abgeleitet. Das Hardware-/Software Interface (HSI) wird verfeinert.

Als Ergebnis einer erfolgreichen Umsetzung dieses Prozesses:

1) sind die Softwareanforderungen den Softwareelementen des Systems zugewiesen und ihre Schnittstellen definiert;
2) sind die Softwareanforderungen kategorisiert und auf Korrektheit und Testbarkeit untersucht;
3) ist die Auswirkung der Softwareanforderungen auf die Betriebsumgebung bewertet;
4) ist die Priorisierung für die Implementierung der Softwareanforderungen definiert;
5) sind die Softwareanforderungen nach Bedarf freigegeben und aktualisiert;
6) sind Konsistenz und Traceability in beide Richtungen zwischen den Systemanforderungen und den Softwareanforderungen hergestellt und

[31] Das Beispiel ist Bestandteil des Automotive-SPICE / ISO 26262 Prozess „Quick Scans“ der TÜV SÜD Automotive GmbH (2013) und entstand unter Federführung des Autors.

nachgewiesen; und es sind Konsistenz und Traceability in beide Richtungen zwischen der Systemarchitektur und den Softwareanforderungen hergestellt;
7) sind die Änderungen der Softwareanforderungen hinsichtlich ihrer Kosten, Zeitplan und technischen Auswirkungen bewertet und
8) sind die Softwareanforderungen als Baseline festgelegt und allen betroffenen Beteiligten mitgeteilt.

Ergänzung Safety:
1) Softwaresicherheitsanforderungen sind als solche mit dem korrekten ASIL gekennzeichnet;
2) ist das HSI verfeinert und gegenüber dem technischen Sicherheitskonzept verifiziert.

Das gleiche Vorgehen wird für den Fragenkatalog für die „Prozess-Areas“ ebenfalls angewendet (siehe Tabelle 6):

Tabelle 6: Ergänzung der Process-Areas PA 1.1 (nach PAM 2.5)

Fragen PA 1.1 Prozessdurchführung

1) Wie wurden die Softwareanforderungen identifiziert und dokumentiert?
   [Softwareanforderungsspezifikation, Tool für die Dokumentation von Anforderungen, abgeleitet aus den Systemanforderungen, Zuweisung der SW-Anforderungen auf SW-Elemente des Systems sowie den Schnittstellen des Systems]

Ergänzung Safety:
Software Safety Requirements aus Systemdesign/Technischem Sicherheitskonzept,
Berücksichtigung zusätzlicher Anforderungen aus der verfeinerten Hardware-Software-Schnittstelle (HSI),
Methodenauswahl nach ASIL zur Beschreibung der Anforderungen,
eindeutig identifizierbar als Sicherheitsanforderung,
ASIL

2) Wie wurden Softwareanforderungen kategorisiert und wie wird sichergestellt, dass sie korrekt und testbar sind?
   [Kategorisierung, Analyseergebnisse Softwareanforderungen, Verifikationskriterien]
3) Wie wurden die Auswirkungen der Softwareforderungen auf ihre Betriebsumgebung analysiert?
   [Schnittstellenspezifikation, Auswirkung auf die angrenzenden SW-Systeme und Elektronik]
4) Wie wurde die Implementierung der Softwareanforderungen priorisiert und geplant?
   [Priorität, Releasezuordnung Softwareanforderungen, Planungsdokumente]
5) Wie erfolgt die Freigabe und Aktualisierung der Softwareanforderungen?
   [Freigabeattribute für Anforderungen, SW-Anforderungsbaseline, Anforderungen an den Änderungsprozess]
6) Wie wird die Konsistenz und Verfolgbarkeit von Systemanforderungen zu SW-Anforderungen und von Systemarchitektur zu SW-Anforderungen

sichergestellt? [Verlinkung zw. Systemanforderungen und SW-Anforderungen in beide Richtungen, Verlinkung zw. Systemarchitektur und SW-Anforderungen in beide Richtungen, Verifikation bezüglich Konsistenz]

Ergänzung Safety:
Konsistenz mit der Hardware-Software-Schnittstelle (HSI),
Verifikationsmethoden: Review, Inspektion, semiformale Verifikation,
Planung der Verifikation,
Spezifikation der Verifikation

Es werden also im Wesentlichen weitere Dokumente gefordert, die spezifisch für die Funktionale Sicherheit sind. Darüber hinaus werden auch die Methoden adressiert, die in der Funktionalen Sicherheit normativ beschrieben werden, was deutlich über den Betrachtungsumfang von SPICE hinausgeht, wo ausschließlich Prozesse betrachtet werden.

2.5.1.3 Beispiel im Rahmen einer Werkzeugzertifizierung

Gerade im Rahmen einer Werkzeugzertifizierung ist die Berücksichtigung einer bereits vorhandenen oder parallel zum Assessment der Funktionalen Sicherheit durchgeführten Prozessreifegradbewertung sinnvoll. Softwarefirmen sind prozessual in der Regel sehr gut aufgestellt, so dass sich hier eine deutliche Synergie ergibt. Im Jahr 2013 wurde unter Federführung des Autors dieser Arbeit ein integriertes Prozess-Assessment auf Basis der in Kapitel 2.5.1.2 beschriebenen Methodik durchgeführt.

Die Besonderheit war ein gemischtes Assessoren-Team bestehend aus:

- 2 „intacs[32] listed Competent SPICE Assessors" (ein Assessor unabhängig, der Zweite vom Auftraggeber)
- 2 Senior Safety Experten (TÜV SÜD) für die Themen der Funktionalen Sicherheit

Das Assessment wurde wie folgt vorbereitet:

1. Strategie-Meeting der Assessoren ca. 2 Monate vor dem Termin (Januar 2013)
2. Detailplanung (Januar 2013)
3. Lieferung der ersten Dokumente seitens des Auftraggebers (Februar 2013)
4. Review der gelieferten Dokumente und Vorbereitung des Vor-Ort-Termins (Februar 2013)

[32] intacs: Ausbildungs- und Zertifizierungsschema für Assessoren für Prozess-Assessmentmodelle (basierend auf ISO IEC 15504 Teil 2).

Im März 2013 fand das fünftägige Assessment beim Kunden statt. Involviert waren auf der Kundenseite das Qualitätsmanagement, die Safety Manager, zeitweise das Entwicklungsteam und das Management. Dieses Assessment ist Bestandteil der Case Study im Kapitel 3.5.2.

Dabei war das Vorgehen wie in der Abbildung 25 dargestellt. Der „rote Faden" war das Automotive-SPICE Assessment. Nach jedem SPICE-Prozessbereich gab es analog zu dem in Kapitel 2.5.1.2 vorgestellten Vorgehen, einen Block mit zusätzlichen Fragen zum Thema Funktionale Sicherheit. Die Ergebnisse wurden dann im Assessoren-Team konsolidiert und dokumentiert.

Date	Start Time	End Time	Activity	Participants
6-Mar	8:00 AM	8:15 AM	Setup Assessment Team	Assessors
6-Mar	8:15 AM	8:30 AM	Morning Data Validation Session	Interviewees, Assessors
6-Mar	8:30 AM	10:30 AM	ENG.4 SW Requirements Analysis	Doug, Christina, Dave
6-Mar	10:30 AM	11:15 AM	Consolidation (ASPICE, 26262)	Assessors
6-Mar	11:15 AM	12:00 PM	Lunch Break (working lunch during SUP.9 interview)	
6-Mar	12:00 PM	2:00 PM	ENG.5 SW Architectural Design	Cameron, Peter
6-Mar	2:00 PM	2:45 PM	Consolidation (ASPICE, 26262)	Assessors
6-Mar	2:45 PM	3:30 PM	Preparing "Next Morning Data Validation Session"	Assessors
6-Mar	3:30 PM	4:30 PM	1. Iteration Preparing "Assessment Summary"	Assessors
7-Mar	8:00 AM	8:15 AM	Setup Assessment Team	Assessors
7-Mar	8:15 AM	8:30 AM	Morning Data Validation Session	Interviewees, Assessors
7-Mar	8:30 AM	10:15 AM	ENG.6 SW Construction (incl. Unit Testing)	Cameron, Peter
7-Mar	10:15 AM	11:45 AM	ENG.7 SW Integration Testing	Sherry, Cameron, Keith
7-Mar	11:45 AM	12:30 PM	Consolidation (ASPICE, 26262)	Assessors
7-Mar	12:30 PM	1:15 PM	Lunch Break	
7-Mar	1:15 PM	3:00 PM	ENG.8 SW Testing	Sherry, Keith
7-Mar	3:00 PM	4:30 PM	Validation	
7-Mar	4:30 PM	5:15 PM	Consolidation (ASPICE, 26262)	Assessors
7-Mar	5:15 PM	6:00 PM	Preparing "Next Morning Data Validation Session"	Assessors
7-Mar	6:00 PM	6:45 PM	2. Iteration Preparing "Assessment Summary"	
8-Mar	8:00 AM	8:15 AM	Setup Assessment Team	Assessors
8-Mar	8:15 AM	8:30 AM	Morning Data Validation Session	Interviewees, Assessors
8-Mar	8:30 AM	9:15 AM	Finalizing "Assessment Summary"	Assessors
8-Mar	9:15 AM	10:45 AM	Review Assessment Summary with Team /	
8-Mar	10:45 AM	11:45 AM	Next Steps Planning	Assessors, Nenna

Abbildung 25: Ausschnitt einer Agenda für ein integriertes Assessment - Automotive-SPICE und Funktionale Sicherheit aus (Bärwald & Bräuchle, 2013)

Das Assessment wurde erfolgreich bestanden und dem Kunden konnte ein Technischer Bericht ausgestellt werden, der beide Themenstellungen (SPICE und Funktionale Sicherheit) beinhaltet, was sowohl dem Auftraggeber als auch seinen Kunden einen deutlichen Mehrwert bietet. Zudem konnten beim Auftraggeber die internen Aufwände deutlich reduziert werden, da für beide Assessments, separat durchgeführt, die doppelte Zeit für Assessment-Meetings notwendig gewesen wäre.

2.6 Zusammenfassung und Auswahl der für diese Arbeit relevanten Standards

Ausgehend von der Forschungshypothese 1a aus Kapitel 1.2.3, wurden die für die Funktionale Sicherheit in verschiedenen Domänen relevanten Standards analysiert. Darüber hinaus gibt es zahlreiche Normen, die die Prozessanforderungen an eine moderne Softwareentwicklung (Software Engineering) definieren und es gibt Reifegradmodelle bzw. -bewertungen, die ebenfalls analysiert wurden. Die Normen zur Funktionalen Sicherheit bedienen sich der gängigen Methoden des Software-Engineerings. So ist beispielsweise der methodische Ansatz des V-Modells die Basis in allen gängigen Normen. Reifegradbewertungen sind seit Anfang der 2000er Jahre Standard im Lieferantenmanagement, z.B. in der Automobilindustrie. Es gab und gibt immer wieder Bestrebungen, die Anforderungen aus dem Bereich der Funktionalen Sicherheit (ISO 26262-6 und -8) mit den Anforderungen einer Prozessreifegradbewertung (Automotive-SPICE) zu vereinheitlichen. Normativ ist das bisher nicht geschehen. Auf der Projektebene können integrale Assessments nach (Automotive-SPICE) und ISO 26262 durchgeführt werden, wie in Kapitel 2.5.1 beispielhaft beschrieben. Hier wurden bezugnehmend auf die Forschungshypothese 1a Überschneidungen herausgearbeitet. Daraus ergeben sich weitergehende Forschungsfragestellungen zur Zusammenführung der Anforderungen und ggf. auch die Einbringung in die Standardisierung, die nicht im Fokus der Betrachtung dieser Arbeit liegen. Im weiteren Verlauf wird deshalb der Prozessunterbau einer Softwareentwicklung nur noch am Rande betrachtet.

In Kapitel 3 werden die Anforderungen an Softwareentwicklungswerkzeuge in einer sicherheitsgerichteten Entwicklung analysiert und es wird ein Verfahren zur generischen Werkzeugqualifizierung abgeleitet. Dabei wird die Umsetzung der Forschungshypothese 1a weiter detailliert. Es werden folgende Normen in die Betrachtung einbezogen: Die generische Grundnorm IEC 61508 und der Automotive-Applikationsstandard ISO 26262. Darüber hinaus wird analysiert, wie im Luftfahrtbereich das Thema Softwarewerkzeuge bereits seit langem adressiert wird. Basis dieser Betrachtung ist die DO-178B.

3 Sichere Softwareentwicklungswerkzeuge und Werkzeugketten

Der Einsatz komplexer Werkzeugketten in der Softwareentwicklung ist Stand der Technik, auch und gerade im Bereich der Sicherheitstechnik. Technologietreiber ist u.a. die zunehmende Verwendung modellbasierter Entwicklungsmethoden inklusive automatischer Code-Generierung bis hin zum automatisierten Test mit Testfall-Generierung. Dafür gibt es Ansätze, die Softwarewerkzeuge bzw. -werkzeugketten durch systematisches Testen zu validieren, wie z.B. in der Dissertation von Stürmer (Stürmer, 2006) oder der Vorgehensweise der Firma Validas AG (siehe Kapitel 3.3.3) beschrieben. Eine Schwäche solcher Ansätze ist es, dass das Werkzeug nur einer „Black-Box"-Betrachtung unterzogen wird. Im Folgenden wird ein Ansatz vorgestellt, der sowohl die Validierung als auch Betriebserfahrung (Confidence From Use) und die Bewertung des Softwareentwicklungsprozesses berücksichtigt.

Alle relevanten Sicherheitsstandards stellen Anforderungen an die Qualität von Softwareentwicklungswerkzeugen bzw. Werkzeugketten. Damit wird ein Ansatz entwickelt, der es ermöglicht, Softwareentwicklungswerkzeuge generisch zu qualifizieren und dabei den Werkzeughersteller für eine Prozessbetrachtung in den Bewertungsprozess einzubinden. Dies ist wichtig, da viele Softwarewerkzeuge eine Historie haben und häufig aus wissenschaftlichen Arbeiten hervorgegangen sind, wo eine prozesskonforme Softwareentwicklung nicht im Zentrum der Betrachtung stand. Werkzeuge wie z.B. Code-Generatoren beinhalten Kernfunktionen, die über viele Jahre unverändert im System enthalten sind, die aber u.U. nicht nach aktuell gültigen Designrichtlinien spezifiziert wurden. Unabhängig davon sind die Werkzeuge auf einem hohen Qualitätsniveau. Um diesen Nachweis auch formal zu erbringen, muss sich einer Kombination verschiedener Qualifizierungsmaßnahmen bedient werden.

Wie komplex die Entwicklung eines einfachen Werkzeuges sein kann, das für die Analyse sicherheitsgerichteter Systeme verwendet werden soll, war u.a. Gegenstand der Arbeit von Augsten im Jahr 2008 an der Hochschule München (Augsten, 2008), die vom Autor dieser Arbeit betreut wurde. In seiner Arbeit entwickelte Augsten ein Softwarewerkzeug zur Bestimmung der Ausfallwahrscheinlichkeiten sicherheitsgerichteter Hardware. Da die Entwicklung ohne Definition und Einhaltung geeigneter Entwicklungsprozesse erfolgte und auch ein Black-Box-Test nicht die vollständige Fehlerfreiheit

des Werkzeuges mit vertretbarem Aufwand zeigen konnte, wurde das Werkzeug im realen Projektgeschäft nie allein verwendet, sondern die Ergebnisse wurden stets durch Verwendung eines diversitären Werkzeuges plausibilisiert (siehe Kapitel 3.1.2.3).

Um allgemeine Anforderungen an Softwarewerkzeuge formulieren zu können, wurden im ersten Schritt folgende Sicherheitsstandards analysiert:

- DO-178B aus dem Luftfahrtbereich als bewährter Ansatz,
- IEC 61508:1998 als Basisnorm für Funktionale Sicherheit sowie
- ISO 26262 als Applikationsstandard für Funktionale Sicherheit im Automobil, der zu Beginn dieser Arbeit in einem frühen Entwurfsstand vorlag (Draft International Standard, DIS).

Ziel der Analyse der verschiedenen Sicherheitsstandards (siehe Kapitel 3.1) waren die Anforderungen an die Qualität der bei der Entwicklung von sicherheitsrelevanten Systemen eingesetzten Entwicklungswerkzeuge.

Ein erster Ansatz wurde vom Autor auf der Konferenz „safetronic 2006“ in München (Bärwald, 2006) vorgestellt. Diese Methodik wurde in der Folge in zahlreichen kommerziellen Kundenprojekten bei der TÜV SÜD Automotive GmbH kontinuierlich weiterentwickelt, die im weiteren Verlauf der Arbeit als exemplarische Fallstudien ausgeführt werden (siehe Kapitel 3.5). Das folgende Kapitel fasst die Ergebnisse der Veröffentlichungen[33] zusammen, die zu dieser Themenstellung vom Autor dieser Arbeit entstanden sind.

3.1 Analyse der Anforderungen verschiedener Standards

Werkzeugunterstützte Entwicklung spielt in der Funktionalen Sicherheit und Informationssicherheit eine entscheidende Rolle, werden doch bei höheren Sicherheitsintegritätsstufen (z.B. SIL3/4, IEC 61508) im Bereich der Funktionalen Sicherheit oder Evaluationsstufen (EAL3) im Security-Bereich die Anforderungen hinsichtlich der Verwendung von Werkzeugen in den Phasen des Lebenszyklus in den einschlägigen Normen eindeutig adressiert. Korrektheit, Integrität und Verfügbarkeit der Werkzeuge sind daher die außerordentlichen Anforderungen an die Werkzeuge und eine Herausforderung für die Anwender.

[33] (Bärwald et al., 2009), (Bärwald & Mottok, 2010), (Bärwald, Hauff, et al., 2010) und (Bärwald & Beine, 2010)

Moderne Softwarestandards fordern Entwicklungsprozesse, die sich am V-Modell (IT-Beauftragte der Bundesregierung, 2006) (siehe 2.4.2.1) orientieren, also phasenorientierte Entwicklungsprozesse, die sequentiell durchlaufen werden. Jede Phase ist dabei durch Eingabedaten, Aktivitäten und Ausgangsdaten gekennzeichnet. Jeder Entwicklungsaktivität ist eine Verifikationsaktivität gegenübergestellt. In Abbildung 26 ist die Adaption des V-Modells für die Softwareentwicklung sicherheitsgerichteter Systeme nach der IEC 61508-3 dargestellt, die ebenfalls eine strukturierte Systementwicklung fordert.

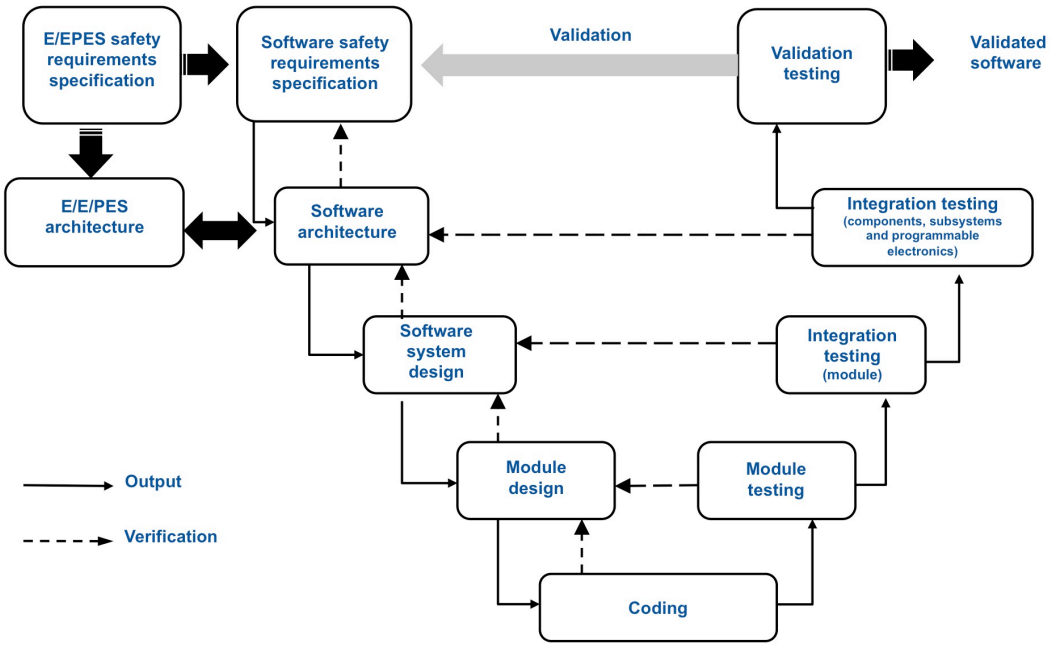

Abbildung 26: Der Software-Sicherheitsintegritäts- und -Entwicklungs-Lebenszyklus (nach IEC 61508-3, Abb. 5 (Deutsche Norm, 2010a)**)**

Besonders wichtig in der Sicherheitstechnik sind die Verifikationsaktivitäten, die sicherstellen, dass das entwickelte System die Sicherheitsanforderungsspezifikation vollständig implementiert. Das impliziert, dass nicht nur einzelne Werkzeuge benötigt werden, sondern eine komplexe Werkzeugkette, ausgehend von der Anforderungsspezifikation bis zur Verifikation und Validierung. Ein wesentlicher Treiber für die Komplexität der Werkzeugkette ist die enge Verzahnung der verschiedenen Werkzeuge in einer Entwicklung. In der Abbildung 27 ist dieser Zusammenhang am Beispiel des V-Modells dargestellt.

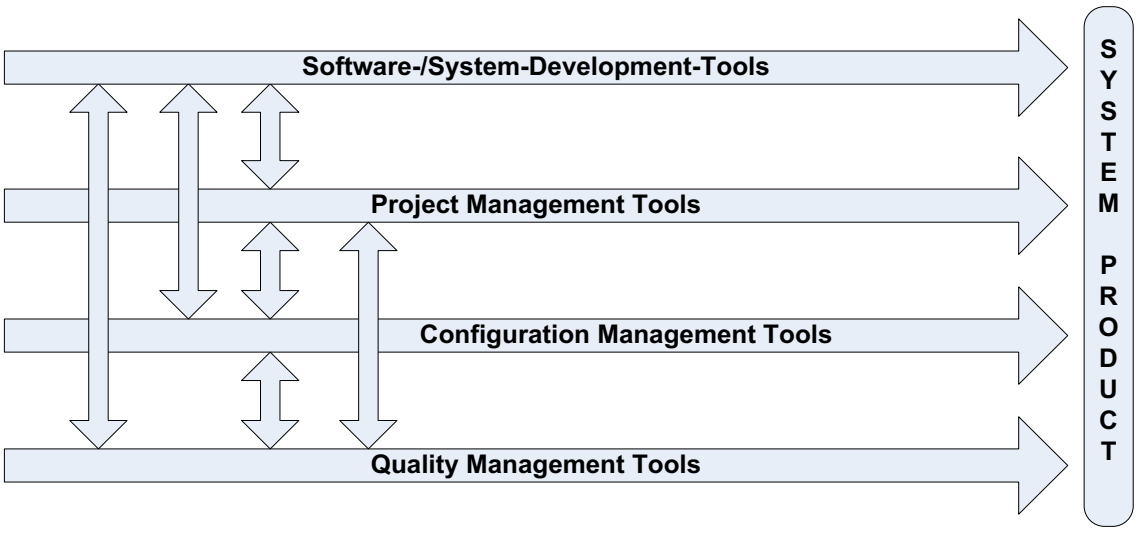

Abbildung 27: Zusammenhang verschiedener Werkzeugklassen nach V-Modell 97 und der Prozessmodule nach V-Modell XT

Das heißt, dass die verschiedenen Werkzeuge miteinander interagieren und das nicht nur sequentiell. So werden beispielsweise die Daten, die in der Softwareentwicklung entstehen, in den Konfigurationsmanagement-Werkzeugen revisionssicher abgelegt. Gleichzeitig sind diese Daten wichtiger Input für das Qualitätswesen, aber auch für das Projektmanagement.

Die Werkzeuge sind nicht nur für die Softwareentwicklung oder für die Nachweisführung der Korrektheit und Funktionalität von Bedeutung, sondern ebenso für das Projekt- und Change-Management, die Verwaltung der Versionen der Software (Configuration Management) aber auch für die Nachverfolgung der Anforderungen (Requirements Management). Zu jeder Phase des Lebenszyklus lassen sich somit Werkzeuge zuordnen, z.B. Spezifikationswerkzeuge, Generatoren für Quellcode, Compiler, Werkzeuge für die Verifikation, die Dokumentation und die Testdurchführung.

Eine in der Praxis in der modernen Elektronikentwicklung verwendete typische Werkzeugkette ist in Abbildung 28 dargestellt. Dabei werden die eingesetzten Softwarewerkzeuge den einzelnen Prozessphasen der Softwareentwicklung aus der ISO 26262-6 zugeordnet. Es ist ersichtlich, dass in jeder Phase typische Werkzeuge zum Einsatz kommen, die den Entwicklungsablauf unterstützen und/oder automatisieren.

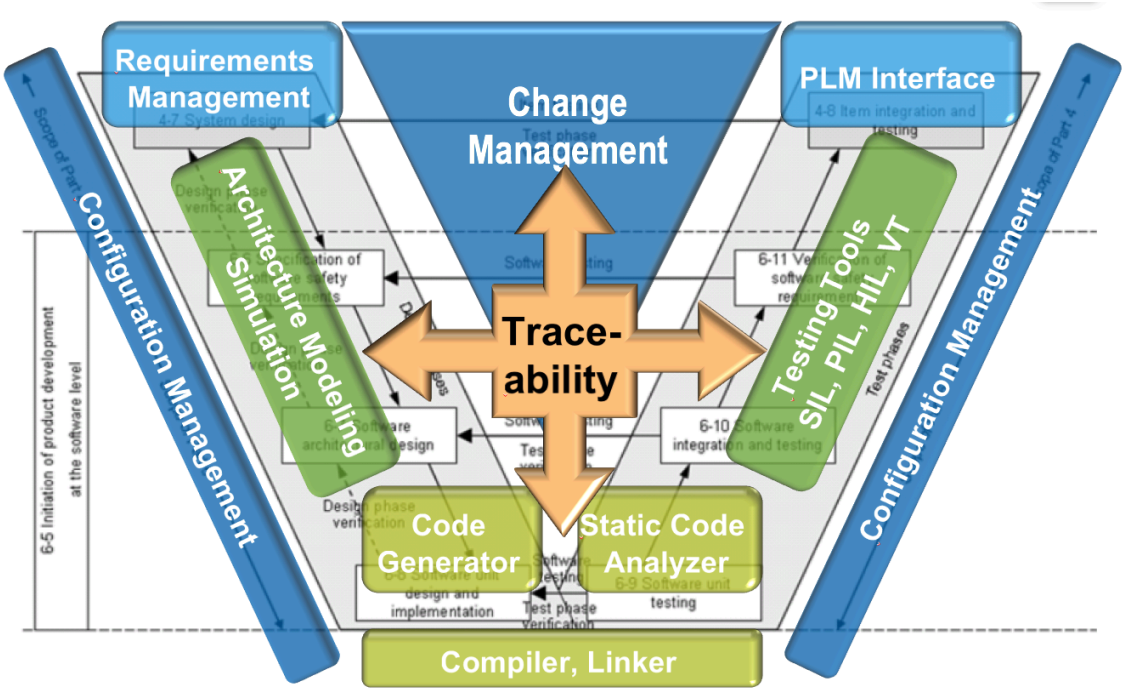

Abbildung 28: Typische Werkzeugkette in einer sicherheitsgerichteten Entwicklung (schematisch dargestellt am Beispiel der ISO 26262-6)

Da nicht von einem monolithischen Werkzeug ausgegangen werden kann, besteht die Gefahr, wenn keine vertrauenswürdige Schnittstelle zwischen den Werkzeugen existiert, dass bei dem Austausch von Daten zwischen den Phasen und den Werkzeugen Verluste von Informationen auftreten können, die die Konsistenz und auch die Information über die Semantik und den Kontext des Systems gefährden können und somit die Sicherheit des Zielsystems gefährden können.

Deshalb muss die Vertrauenswürdigkeit (Confidence) der Werkzeuge mindestens so hoch sein, wie es für die Entwicklung, Prüfung und Zertifizierung des sicherheitskritischen Systems erforderlich ist. Die Herausforderung dabei ist, diese Vertrauenswürdigkeit herzustellen und formal zu zeigen.

Vermehrt kommt die Anforderung für den modellbasierten Softwareentwicklungsansatz auch für sicherheitskritische softwarebasierte Systeme. Modellbasierte Entwicklung bietet den Vorteil, dass viele am Projekt beteiligte Personen das System auf hohem Abstraktionsniveau diskutieren können, ohne Details über die konkrete Umsetzung in Hardware und/oder Software wissen zu müssen. Als Konsequenz wird ausreichend Vertrauen in die Korrektheit der Werkzeuge und deren Ergebnisse vorausgesetzt, vor allem wenn Werkzeuge zur Automatisierung von Prozessschritten in der Entwicklung speziell bei sicherheitskritischen Systemen eingesetzt werden.

In den 1980er- und 1990er-Jahren bestand eine große Euphorie hinsichtlich formaler Spezifikation und Verifikation. Das hat die Entwicklung von formalen Spezifikationssprachen und Beweisern vorangetrieben. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) startete damals die Entwicklung des Spezifikations- und Verifikationswerkzeuges „Verification Support Environment (VSE)“, das später vom DFKI[34] in Kaiserslautern weitergepflegt wurde (Nonnengart, Rock, & Stephan, 2003).

Die formale Spezifikation und Verifikation hat in der Praxis ihre Grenzen, da nicht sichergestellt werden kann, dass das System in seiner realen Umgebung korrekt arbeitet. Durch die formale Methodik wird lediglich die Konsistenz und Korrektheit der Implementierung gegenüber der Spezifikation bewiesen, nicht aber die Korrektheit und Vollständigkeit der Anforderungen gegenüber dem realen Kontext des Systems. Ein weiteres Problem in der Praxis ist, dass Spezialisten für formale Methoden am Arbeitsmarkt praktisch nicht verfügbar sind, die aber benötigt werden, um komplexe Systeme formal spezifizieren und verifizieren zu können. Hier wird bereits die Spezifikationsphase zu einer großen Herausforderung hinsichtlich Funktionaler Sicherheit. Es besteht zudem eine Lücke bezüglich des Beweisers. Die Frage ist dabei: Wie kann sichergestellt werden, dass die Regelbasis (Axiome, Theoreme etc.) korrekt ist und keine Tautologien oder fehlerhafte Axiome oder Theoreme enthalten sind?

Somit haben vor allem Werkzeuge, die den klassischen strukturierten Entwicklungsansatz unterstützen, einen starken Einfluss auf die Funktionale Sicherheit in allen Phasen des Softwarelebenszyklus. Werkzeuge lassen sich in folgende Gebiete kategorisieren:

- Dokumentation,
- Modellierung/Spezifikation,
- Generierung,
- Kompilierung,
- Verifikation,
- Validation/Test und
- (Projekt-)Management

[34] Deutsches Forschungszentrum für Künstliche Intelligenz mit Sitz in Kaiserslautern, Saarbrücken, Bremen und Berlin

Zunehmend finden Werkzeuge zur Automatisierung der Verifikations- und Validierungstätigkeiten großes Interesse, da im Bereich der Funktionalen Sicherheit ein hoher Aufwand für die Nachweisführung der Funktionalen Sicherheit erforderlich ist. So beschränken sich die eingesetzten Werkzeugtypen nicht nur auf statische Verifikationswerkzeuge, wie Code-Checker (z.B. PCLint[35]) oder statische Verifikation (z.B. Polyspace[36]), sondern es kommen auch dynamische Tests und Simulation zum Einsatz.

Werkzeuge können generell in drei Klassen unterteilt werden:

1. **Entwicklungswerkzeuge:** Werkzeuge, die unmittelbar Fehler im Zielsystem verursachen können (z.B. Code-Generatoren, Compiler).
2. **Verifikationswerkzeuge:** Werkzeuge, die keine Fehler im Zielsystem verursachen (z.B. Dokumentations-, Verifikations- und Validationswerkzeuge), aber deren Versagen die Entdeckung von systematischen Fehlern bei der Entwicklung unterbindet.
3. **Unterstützende Werkzeuge:** Werkzeuge, die keine direkten und indirekten Auswirkungen auf den Target-Code haben können (z.B. Projektmanagement- und Ressourcenplanungswerkzeuge).

In Rahmen dieser Arbeit werden unterstützende Werkzeug nicht näher betrachtet. Es ist in diesem Zusammenhang darauf hinzuweisen, dass es auch bei unterstützenden Werkzeugen auf der Projektebene grundsätzlich einer Analyse bedarf, ob tatsächlich Seiteneffekte auszuschließen sind.

Der Zusammenhang der einzelnen Entwicklungswerkzeuge in einer typischen Systementwicklung in der Luftfahrtindustrie (basierend auf der Anwendung der DO-178B, siehe Kapitel 2.4.1.9) wurde in (Kornecki & Zalewski, 2006) untersucht und ist in Abbildung 29 dargestellt. Softwarewerkzeuge, Softwarebibliotheken und Target-Software werden bei diesem Ansatz gleichermaßen in die Betrachtung einbezogen. Die Abgrenzung der Entwicklungswerkzeuge (Box rechts oben), die eine erhöhte Kritikalität[37] für den Entwicklungsprozess haben können und somit eine besondere Behandlung

[35] PCLint ist ein Werkzeug der Firma Gimpel Software LLC für die statische Codeanalyse (http://www.gimpel.com/html/pcl.htm)

[36] Polyspace von der Firma The MathWorks Inc. ist ein Werkzeug zur statischen Verifikation (https://www.mathworks.com/products/polyspace.html)

[37] Die Kritikalität einer Betrachtungseinheit drückt aus, welche Bedeutung ihrem Fehlverhalten beigemessen wird. (nach (Dröschel & Wiemers, 2000) Seite SI-i)

erfordern, ist deutlich in der Abbildung 29 erkennbar. Der Ansatz ist in diesem Beispiel generisch. Es wird nicht berücksichtigt, wie der projektspezifische Einsatz der Werkzeugketten im Zusammenspiel mit den normativ geforderten Verifikationsmaßnahmen umgesetzt wurde.

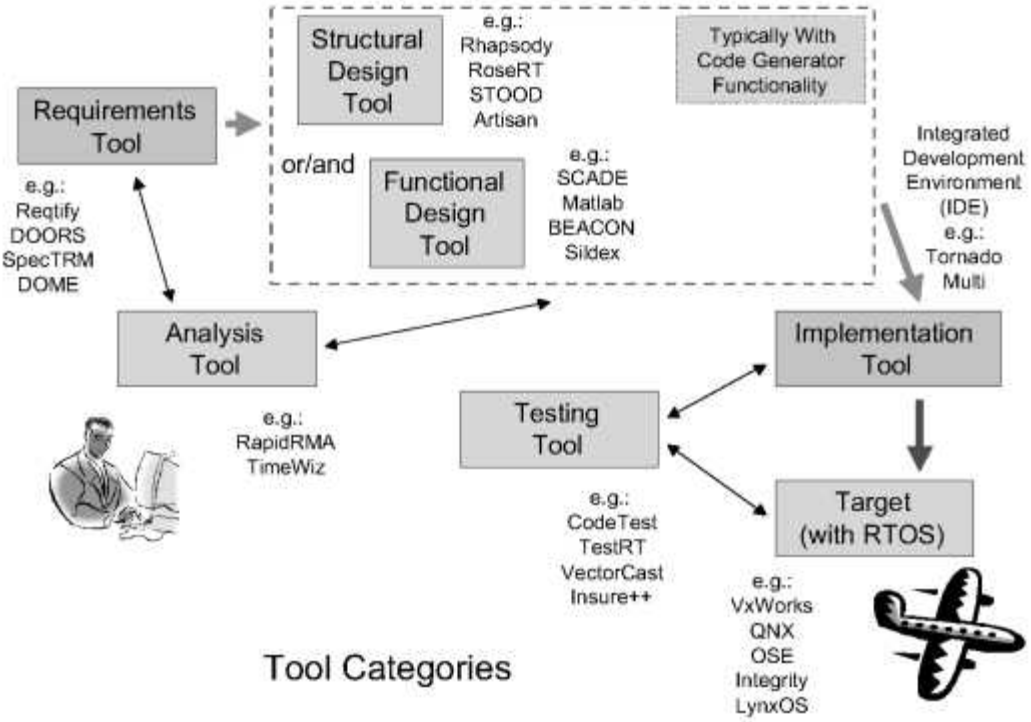

Abbildung 29: Softwarewerkzeugkategorien nach (Kornecki & Zalewski, 2006) **im Kontext der DO-178B**

Neuere Standards gehen in der Betrachtung deutlich weiter und verlangen eine projekt- bzw. organisationsspezifische Klassifizierung von Werkzeugen (siehe ISO 262626 (International Standard, 2011b) Band 8 Kapitel 11).

Die Herausforderung besteht darin, für die Werkzeugkette sicherzustellen, dass deren Ergebnissen vertraut werden kann und die Semantik erhalten bleibt. Darüber hinaus muss sichergestellt sein, dass die Dokumentation und das implementierte System konsistent sind, im Besonderen bei automatisch generierter Dokumentation.

Nachfolgend werden die normativen Anforderungen betrachtet, verschiedene Ansätze für die Validierung, Zertifizierung und Qualifizierung von Werkzeugen aufgezeigt und abschließend eine Zusammenfassung der Ergebnisse gegeben.

3.1.1 Normative Anforderungen

Alle relevanten internationalen Standards, die Anforderungen an die Softwareentwicklung für sicherheitsgerichtete Systeme beinhalten, stellen auch Anforderungen an das Vertrauensniveau für Softwareentwicklungswerkzeuge.

Die internationale generische Norm zur Funktionalen Sicherheit ist die IEC 61508. In Teil 3 dieses Standards werden die Anforderungen an die Software sicherheitsgerichteter Systeme adressiert. Darüber hinaus gibt es zahlreiche applikationsspezifische Normen für die Bereiche Medizinprodukte, Sicherheit von Maschinen, Gasmesstechnik, Bahn- und Automobilanwendungen, die von der IEC 61508 abgeleitet wurden und zum Teil deren Anforderungen präzisieren (siehe Kapitel 2.4.1).

Im Folgenden werden die Anforderungen an Softwareentwicklungswerkzeuge in den Normen IEC 61508 und ISO 26262 analysiert, da diese Normen im Automobilumfeld Anwendung finden.

3.1.1.1 IEC 61508

Nach IEC 61508-3 (Deutsche Norm, 2010a) in der Edition 1 von 1998 wird die Verwendung von betriebsbewährten oder zertifizierten Werkzeugen im Grunde unabhängig von der Risikobewertung des Zielsystems gefordert (siehe Zeile 4a & 4b in Tabelle 7).[38]

Tabelle 7: Software Design und Entwicklung: Anforderungen an Werkzeuge und Programmiersprachen nach IEC 61508-3 (1998) Tabelle A.3 (Deutsche Norm, 2010a)

Technique/Measure		Ref	SIL1	SIL2	SIL3	SIL4
1	Suitable programming language	C.4.6	HR	HR	HR	HR
2	Strongly typed programming language	C.4.1	HR	HR	HR	HR
3	Language subset	C.4.2	---	---	HR	HR
4a	Certificated tools	C.4.3	R	HR	HR	HR
4b	Tools: increased confidence from use	C.4.4	HR	HR	HR	HR
5a	Certificated translator	C.4.3	R	HR	HR	HR
5b	Translator: increased confidence from use	C.4.4	HR	HR	HR	HR
6	Library of trusted/verified software modules and components	C.4.5	R	HR	HR	HR

[38] R bedeutet „recommend“ - empfohlen. HR bedeutet „highly recommend“ - stark empfohlen. R-Maßnahmen sollten umgesetzt werden, HR-Maßnahmen müssen umgesetzt werden oder alternative Maßnahmen mit gleicher Wirksamkeit.

Dieser Nachweis ist prinzipiell für jedes verwendete Softwarewerkzeug zu erbringen. Grundsätzlich ist es für den Werkzeuganwender schwierig, den Nachweis der Betriebsbewährtheit zu erbringen, da ein einzelner Anwender im Regelfall nicht über die notwendige Fülle an Betriebserfahrungen mit einer speziellen Version in unterschiedlichem Kontext verfügt. Die Zertifizierung eines Werkzeuges ist in den meisten Fällen nur auf Initiative des Werkzeugherstellers selbst sinnvoll, da nur dieser in der Regel detailliert über das entsprechende interne Know-How verfügt. Zu Beginn dieser Arbeit gab es in der Praxis erhebliche Schwierigkeiten, diesen Passus der Norm in vollem Umfang zu erfüllen, da im Grunde keine zertifizierten Softwareentwicklungswerkzeuge am Markt verfügbar waren. Dies war eine der Hauptmotivationen, ein Zertifizierungsverfahren zu etablieren.

In der Überarbeitung der IEC 61508:2010 (Deutsche Norm, 2010a) Band 3 Kapitel 7.4.4 werden die Anforderungen an Werkzeuge präzisiert. Dabei werden Offline- und Online-Werkzeuge unterschieden. Offline-Werkzeuge sind Werkzeuge, die bei der Entwicklung von sicherheitsrelevanten Systemen verwendet werden. Online-Werkzeuge sind während des Betriebs des sicherheitsrelevanten Systems aktiv. Online-Werkzeuge können deshalb „als ein Softwareelement des sicherheitsbezogenen Systems angesehen werden“ (siehe IEC 61508 Band 3 Kapitel 7.4.4.1). Dieser Ansatz ist dem der DO-178B (International Standard, 1992) sehr ähnlich. Für die Offline-Werkzeuge wird in Abhängigkeit der Kritikalität der Werkzeuge für das Endprodukt zwischen den Werkzeugklassen T1 bis T3 unterschieden, die unterschiedlich tiefgreifende Qualifizierungsmaßnahmen erfordern (siehe IEC 61508 Band 4 Kapitel 3.2.11):

- Bei einem Werkzeug der Kategorie T1 handelt es sich um ein Werkzeug, dass keine direkte Auswirkung auf den Produktivcode des mit ihm entwickelten sicherheitsrelevanten Systems hat, z.B. unterstützende Werkzeuge.
- Ein Werkzeug der Klasse T2 hat indirekten Einfluss auf den Target-Code, z.B. sind Testwerkzeuge der Klasse T2 zuzuordnen.
- Bei Werkzeugen der Klasse T3 kann es zu direkten Auswirkungen auf das Zielsystem zur Laufzeit kommen. Typische Beispiele hierfür sind Compiler und Code-Generatoren.

Speziell für die Werkzeugklassen T2 und T3, die eine gewisse Kritikalität für das Zielsystem haben, gilt es, die im Standard beschriebenen Anforderungen zu erfüllen. Das betrifft z.B.:

- Es muss eine Spezifikation und eine Produktdokumentation geben, inkl. einer Benutzerdokumentation, dem sogenannten „Safety Manual“ (IEC 61508 Band 4 Kapitel 7.4.4.4).
- Der „Vertrauensgrad“ des Werkzeuges muss beurteilt werden (IEC 61508 Band 4 Kapitel 7.4.4.5).
- Die Werkzeugkette muss vertrauenswürdig sein (IEC 61508 Band 4 Kapitel 7.4.4.9).

Die Anforderungen an die Werkzeuge und Werkzeugketten sind im Standard nicht formal beschrieben. Im Besonderen sind weder die Qualifikationsmaßnahmen, noch die zu erzeugende Dokumentation spezifiziert. Das im weiteren Verlauf dieser Arbeit hergeleitete Vorgehen, das im Besonderen auf den Anforderungen aus der ISO 26262 (siehe Kapitel 3.1.1.3 (International Standard, 2011b)) basiert, kann auch im Rahmen des Nachweises der Vertrauenswürdigkeit von Werkzeugen im Rahmen der IEC 61508 herangezogen werden.

3.1.1.2 DO-178B

Die DO-178B (International Standard, 1992) kann als einer der ersten Standards gesehen werden, der allgemeine Anforderungen an die Verwendung von Softwareentwicklungswerkzeugen in sicherheitsgerichteten Entwicklungen gestellt hat (DO-178B Kapitel 12.2). Der Ansatz dieser Norm ist, Werkzeuge zu klassifizieren und aus dieser generellen, anwendungs- und projektunabhängigen Sicht die Werkzeugabsicherung zu betreiben. Die Softwarewerkzeuge werden dabei in folgende Klassen unterteilt:

- Software-Entwicklungswerkzeuge und
- Software-Verifikationswerkzeuge

Dabei können Entwicklungswerkzeuge direkten Einfluss auf den Target-Code nehmen. Verifikationswerkzeuge haben diese Kritikalität nicht, da sie nur zur Testausführung verwendet werden. Werkzeugfehler haben daher keinen direkten Einfluss auf den Produktiv-Code, können aber die Wirksamkeit der Teststrategie empfindlich stören.

Für Entwicklungswerkzeuge wird gefordert, dass die Entwicklungsprozesse für diese Werkzeuge dem gleichen Sicherheits-Level entsprechen, wie die Prozesse für die Entwicklung des Gesamtsystems (Flugzeug). Auch hier gilt die allgemeine Erkenntnis, dass das Sicherheits-Level der Prozesskette durch das Sicherheits-Level des schwächsten Prozessgliedes bestimmt wird. Für Verifikationswerkzeuge ist zu zeigen, dass sich das Werkzeug der Spezifikation entsprechend verhält. Dieser Zusammenhang ist in dem Entscheidungsbaum in Abbildung 30 schematisch dargestellt.

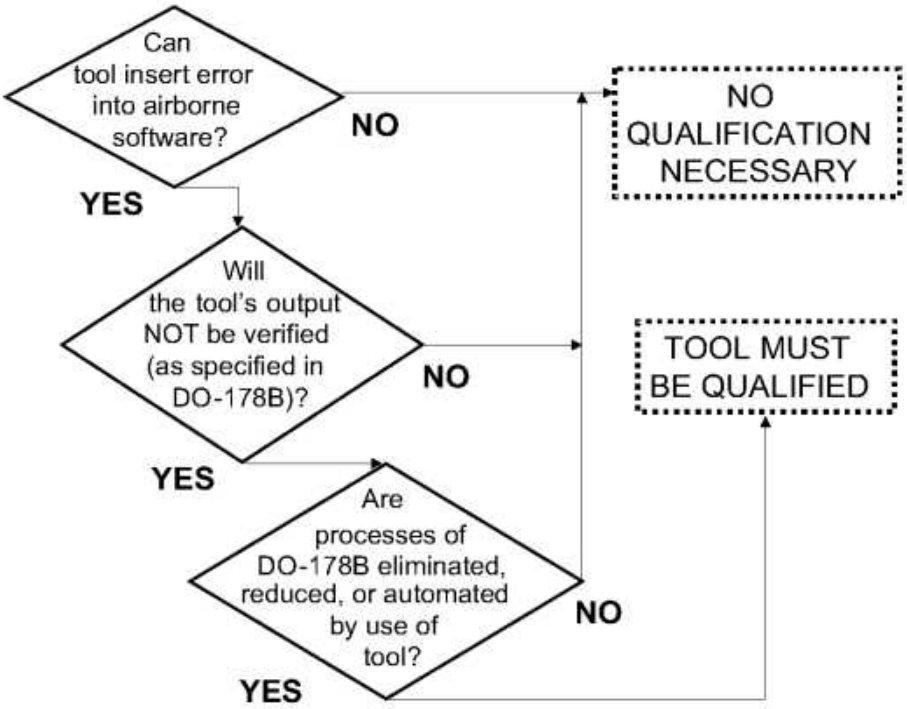

Abbildung 30: Werkzeugqualifizierungskriterien nach DO-178B (International Standard, 1992)

Darüber hinaus gibt es noch zahlreiche Anforderungen an die Dokumentation, wie z.B. Tool Qualification Data (DO-178B Kapitel 12.2.3) mit dem Tool Qualification Plan (DO-178B Kapitel 12.2.3.1) und dem Tool Qualification Agreement (DO-178B Kapitel 12.2.3.4).

3.1.1.3 ISO 26262

Nachdem frühere Entwürfe der ISO 26262 (die sogenannten Baselines) eine pauschale Klassifizierung der Softwarewerkzeuge in Entwicklungswerkzeuge, Testwerkzeuge und Begleitwerkzeuge (ähnlich der DO-178B, siehe Kapitel 3.1.1.2) vorgesehen und mit Anforderungen an die Werkzeugqualifizierung verbunden haben, sind in der finalen Version der ISO 26262 (International Standard, 2011b) alle eingesetzten Werkzeuge projektspezifisch zu bewerten und ggf. zu qualifizieren. Diese Bewertung ist

projektbezogen durch den Anwender durchzuführen. **Abbildung 31** zeigt die Methodik der Bestimmung des Vertrauensgrades eines Werkzeuges (Tool Confidence Level, TCL).

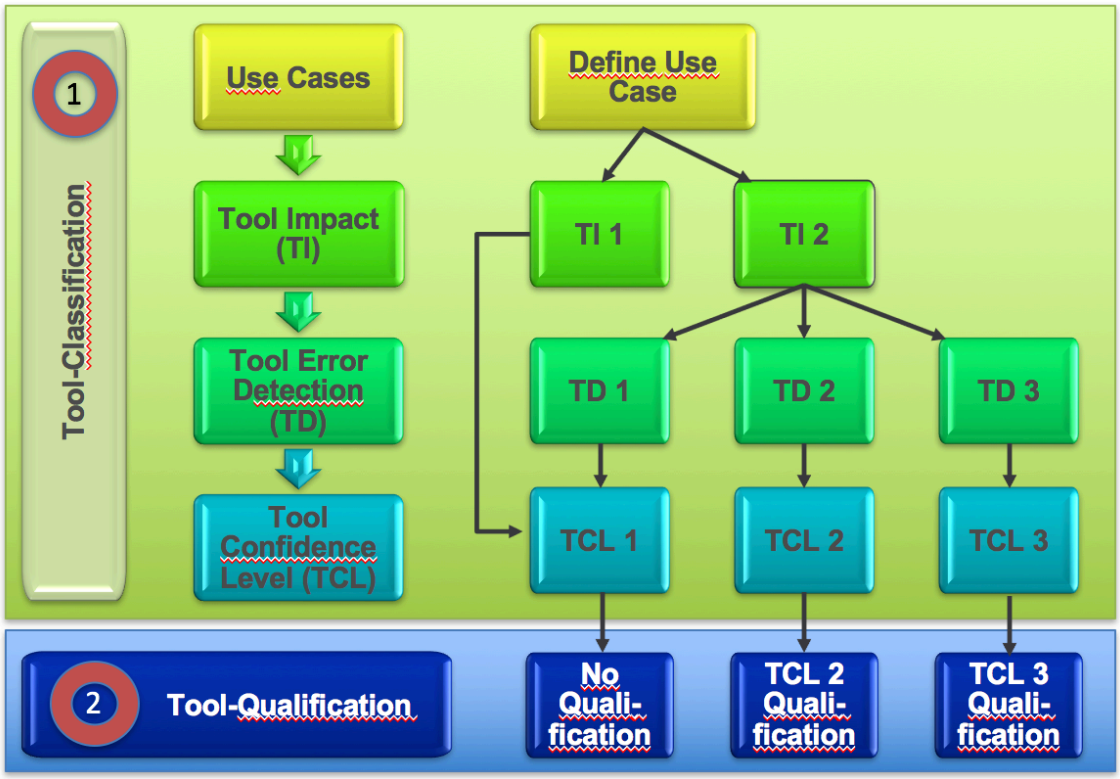

Abbildung 31: Bestimmung des Vertrauensgrades eines Werkzeuges nach ISO 26262-8

Zuerst müssen für jedes verwendete Werkzeug die im Einsatz relevanten spezifischen Use Cases definiert werden. Dann gestaltet sich das Verfahren zweistufig. Im ersten Schritt wird die Kritikalität des Werkzeuges (TCL) ermittelt. Für die Klassen TCL 2 und TCL 3 sind dann in einem zweiten Schritt spezifische Qualifizierungsmaßnahmen nötig. Der TCL ermittelt sich aus den Größen Tool Impact (TI) und Tool Error Detection (TD). In der Abbildung 32 ist der anzuwendende Graph darstellt. Dabei ist bei der Bestimmung der Bewertungsgrößen wie folgt vorzugehen:

1. Ermittlung des Tool Impact, der Auswirkung eines Werkzeugfehlers auf den Produktivcode:
    - TI 1 bedeutet, dass keine Kritikalität entstehen kann oder dass Fehler vollständig im Entwicklungsprozess erkannt werden.
    - TI 2 bedeutet, dass ein Werkzeugfehler den Werkzeug-Output sicherheitsrelevant verfälschen kann.
2. Ermittlung der Tool Error Detection, also der Bewertung, ob auftretende Fehler im Prozess gefunden werden können:

- TD 1 bedeutet, dass diese Fehler mit hoher Wahrscheinlichkeit durch Verifikationsmaßnahmen aufgedeckt werden können,
- TD 2 gewährleistet eine mittlere Aufdeckungswahrscheinlichkeit sowie
- TD 3 bedeutet nur eine zufällige Aufdeckungswahrscheinlichkeit.

Der TD ist kein quantitatives Maß, seine Ermittlung ist somit ein Stück weit subjektiv. Das Vorgehen wird im Kapitel 3.2 an einem Beispiel für die Verwendung eines komplexen Konfigurationsmanagementwerkzeuges detailliert beschrieben.

Aus dieser Einstufung ergibt sich der sogenannte Tool Confidence Level (TCL), der Maßnahmen vorschreibt, wie die Vertrauenswürdigkeit von Entwicklungswerkzeugen garantiert werden kann. Aus einem TCL 1 resultieren keine Anforderungen an Qualifizierungsmaßnahmen (wie Betriebsbewährtheit, Validierung, Bewertung des Entwicklungsprozesses oder Entwicklung nach einem Sicherheitsstandard). Für alle anderen TCL sind Qualifizierungsmaßnahmen in Abhängigkeit der Risikoeinstufung des Zielsystems vorzunehmen. Die möglichen Maßnahmen sind in den Tabellen im Band 8 Kapitel 11 der ISO 26262 in Abhängigkeit von TCL und Sicherheitsrelevanz des Zielsystems (ASIL) gelistet (siehe Abbildung 32 als Beispiel für den TCL 3).

Methods		ASIL			
		A	B	C	D
1a	Increased confidence from use in accordance with 11.4.7	++	++	+	+
1b	Evaluation of the tool development process in accordance with 11.4.8	++	++	+	+
1c	Validation of the software tool in accordance with 11.4.9	+	+	++	++
1d	Development in accordance with a safety standard[a]	+	+	++	++

[a] No safety standard is fully applicable to the development of software tools. Instead, a relevant subset of requirements of the safety standard can be selected.

EXAMPLE Development of the software tool in accordance with ISO 26262, IEC 61508 or RTCA DO-178.

Abbildung 32: Möglichkeiten zur Werkzeugqualifizierung am Beispiel der ISO 26262 Band 8 Kapitel 11 Tabelle 4 für den TCL 3

Wird der normativen Vorgabe gefolgt, so ist die Abbildung 32 so zu lesen, dass eine XOR-Methodenauswahl möglich ist. Dieses Vorgehen erscheint wenig sinnvoll, da Werkzeuge häufig eine Historie besitzen und deshalb immer eine geeignete Mischung der genannten Methoden zielführend ist. Deshalb kombiniert das in dieser Arbeit beschriebene Assessmentschema die genannten Methoden.

3.1.2 Grundsätzliche Ansätze zur Qualifizierung und Zertifizierung von Werkzeugen

Prinzipiell gibt es aus wissenschaftlicher Sicht verschiedene Qualifizierungsansätze, die Vertrauenswürdigkeit von Softwarewerkzeugen nachzuweisen:

- **„Fitness For Purpose“**, die Entwicklung des Werkzeuges nach einem bzw. in Anlehnung an einen Sicherheitsstandard (z.B. IEC 61508-1 und 3, DO-178B),
- **Formale Verifikation,** der mathematische Nachweis der Korrektheit des Werkzeuges sowie
- **Verwendung diversitärer Werkzeuge (N-out-of-M-Architektur)**

Wird ein Qualifizierungsansatz projektübergreifend und anwenderunabhängig durchgeführt, dann ist eine Zertifizierung des Werkzeuges nach dem in dieser Arbeit beschriebenen Schema prinzipiell möglich (siehe Kapitel 3.4).

Die Entwicklung eines Softwarewerkzeuges nach einem Sicherheitsstandard ist in der Regel sehr aufwendig. Dieses Vorgehen wird beispielsweise im Luftfahrtbereich angewendet (z.B. Esterel SCADE KCG[39]). Die formale Verifikation von Werkzeugen, also der mathematische Beweis der korrekten Funktionsweise des Werkzeugs in Bezug auf seine Spezifikation, wird im Umfeld der Informationssicherheit genutzt. Für den Automobilbereich sind bereits Werkzeuge mit einer „Fitness For Purpose“-Zertifizierung verfügbar (z.B. dSPACE TargetLink und PTC Integrity, siehe die Fallstudien in den Kapiteln 3.5.1 und 3.5.2), die nach dem in dieser Arbeit definierten Ansatz assessiert und zertifiziert wurden. Ein weiterer vielversprechender Ansatz ist eine „NooM“-Architektur der verwendeten Werkzeugkette. Deshalb werden die beiden letztgenannten Methoden im Folgenden im Detail betrachtet.

3.1.2.1 „Fitness For Purpose“

„Fitness For Purpose“ bedeutet, dass ein entsprechend zertifiziertes Softwarewerkzeug prinzipiell unter definierten Rahmenbedingungen, den sogenannten „conditions of use“, für den Einsatz bei der Entwicklung von sicherheitsgerichteten Systemen geeignet ist. Ziel einer solchen Zertifizierung ist, den Werkzeuganwendern einen generischen und projektunabhängigen Eignungsnachweis zur Verfügung zu stellen. Die „Fitness For

[39] SCADE Suite KCG ist ein Code-Generator der Firma Esterel Technologies SAS, der C-Code aus SCADE Modellen automatisiert erzeugt (Esterel Technologies SAS, 2015).

Purpose"-Zertifizierung wird von einem unabhängigen Prüfinstitut im Auftrag des Werkzeugherstellers durchgeführt.

Die Kriterien für eine solche Zertifizierung setzen sich aus den folgenden Aspekten zusammen:

- **Entwicklungsprozess** inkl. Modifikationsprozess: Der Prozess muss in Anlehnung an einen geeigneten Softwareentwicklungsprozess-Standard definiert werden.
- **Produktvalidierung**: Das Produkt muss hinreichend getestet sein. Dies kann z.B. durch Definition einer angemessenen Validierungssuite erfolgen.
- **Prozess für geeignetes Fehlermeldewesen** (Bug Reporting, Bug Tracking) und Umgang mit gemeldeten Werkzeugfehlern (Customer Information),
- **„Safety Manual" bzw. Referenzworkflow**: Bei komplexen Werkzeugen wie z.B. Code-Generatoren ist es sinnvoll, dem Anwender eine Verwendungsempfehlung zur Verfügung zu stellen. Dieser Referenzworkflow ist dann Bestandteil der Zertifizierung und definiert u.a. eine geeignete Teststrategie des Werkzeug-Outputs.

Die Gewichtung dieser Kriterien im Zertifizierungsprozess ist produktabhängig, das grundsätzliche Vorgehen ist in Kapitel 3.4 beschrieben.

3.1.2.2 Formale Verifikation

In den letzten 30 Jahren gab es signifikante wissenschaftliche Fortschritte auf dem Gebiet der automatisierten Theorem-Beweiser[40]. Trotzdem ist die Verifikation von komplexen Algorithmen, wie sie beispielsweise in Compilern und Code-Generatoren verwendet werden, schwer automatisierbar und erfordert deshalb eine sehr aufwendige Nutzerinteraktion mit dem Theorem-Beweiser (Frank et al., 2008). Zielsetzung einer formalen Compiler-Verifikation ist es, die vollständige mathematische Korrektheit aller Teile des Compilers zu zeigen.

Die formale Verifikation ist eine sehr aufwendige Methode. Sie ist teuer und spielt im Automotive-Umfeld keine Rolle. Im Luftfahrt- und Militärbereich wird das Verfahren seit längerem eingesetzt. Bormann beschreibt in seiner Disseration (Bormann, 2009), wie

[40] Ein Theorembeweiser ist ein System, das bei einem Kalkül (für eine gewisse Logik) und einer Formel versucht, durch wiederholte Anwendung der Inferenzregeln des Kalküls einen Beweis zu finden. (nach (Schumann, 2001) Kapitel 3.2)

die Methoden der formalen Verifikation im Halbleiterbereich eingesetzt werden können. In Rahmen dieser Arbeit wird diese Methodik nicht näher betrachtet. Hier lassen sich weitere Forschungsfragen in Bezug auf die Anwendung Formaler Verifikation für das Verhalten von Softwarewerkzeugen ableiten.

3.1.2.3 N-out-of-M-Architektur

Neben den bereits dargestellten Strategien der Toolqualifizierung gibt es alternativ den Ansatz einer N-out-of-M–Architektur. In **Abbildung 33** ist dieser Ansatz mit einer Triple Modular Redundancy (TMR) Struktur, also einer 2oo3-Architektur dargestellt.

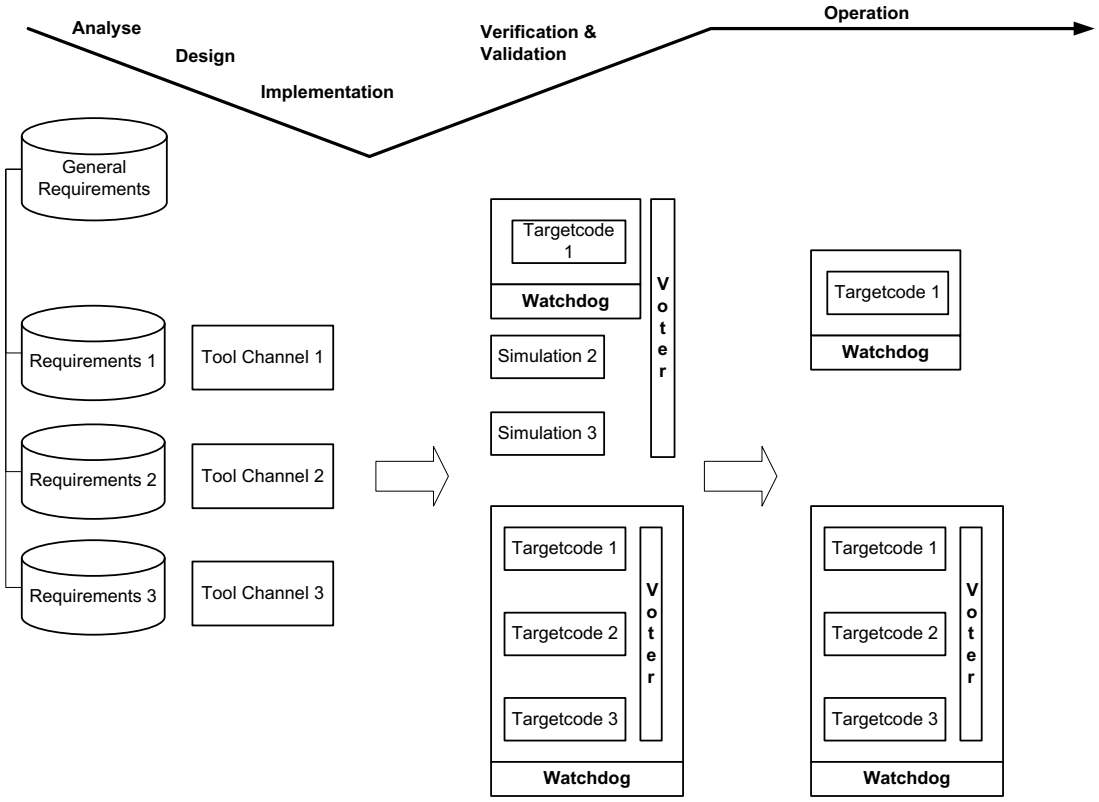

Abbildung 33: Triple Modular Redundancy (TMR) Struktur entwickelt nach der Multiversionen-Technik (N-Version Programming) am Beispiel eines 2oo3-Votings (Bärwald, Mottok, et al., 2010)

Dabei wird die Multiversionen-Technik des N-Version Programming genutzt, wobei alle realisierten Kanäle (Versionen) diversitär entwickelt werden, um die gleichen grundlegenden Anforderungen zu erfüllen. Die Beherrschung zufälliger Fehler und eine Reduktion verbleibender systematischer Fehler werden dabei möglich. Die Korrektheitsentscheidung wird durch Mehrheitsentscheid aller beteiligten Kanäle mit einem Voter getroffen. Der Ansatz kann zusätzlich mit Softwareredundanz, Informationsredundanz und Zeitredundanz kombiniert werden.

Die Diversität in den verwendeten Tools, Modellierungssprachen, Programmiersprachen und Entwicklungswerkzeugen im gesamten Entwicklungsprozess ist das Hauptargument für ein Erkennen der systematischen und zufälligen Fehler in der NooM-Architektur: In diversitär entwickelten Kanälen wird das Vorliegen gleicher systematischer Fehler (Common Cause Failure) als weniger wahrscheinlich als das Vorliegen unterschiedlicher systematischer Fehler angesehen (N-Versionen-Annahme).

Damit kann der Aufwand für eine Toolqualifizierung reduziert werden. Die diversitären Kanäle können als Referenzimplementierung des Back To Back Testing-Ansatzes (siehe Kapitel 3.5.1.4) interpretiert werden.

Neue fehlertolerante Architekturen mit diversitär-codierten Kanälen, wie zum Beispiel das Konzept der „Safely Embedded Software“ (SES) des LaS³ an der OTH Regensburg[41] (Steindl, Mottok, Meier, Schiller, & Fruechtl, 2009), nutzen Komparatoren anstelle von Votern für Fehlerkennung des Zielsystems.

Moderne Multi-Core Controller Architekturen erlauben die Verteilung der verschiedenen diversitären Kanäle der NooM-Architektur auf die verschiedenen Verarbeitungseinheiten (Cores).

Eine Variante des diskutierten Architekturansatzes nutzt die Diversität der verschiedenen Kanäle nur während der Verifikations- und Validationsphase im Lebenszyklus des sicherheitskritischen Systems. Dabei können die Kanäle aus Simulation(en) und dem Target-System gebildet werden: SIL (Software in the Loop) oder HIL (Hardware in the Loop). Das Target-System wird gegen diversitäre Simulationen getestet, wobei das simulierte System auf Basis diversitärer Spezifikationen, diversitärer Modelle, diversitärer Entwicklungsteams und diversitärer Entwicklungswerkzeuge erstellt werden kann. Problematisch kann in dieser Variante das Zeitverhalten der Simulation werden, da sich die Antwortzeiten der Simulation vom Target-System unterscheiden können. Wissen über nebenläufige Tasks, Multi-Tasking und ereignisbasierte Kommunikation auf dem Target-System wird zur Bewertung der Verifikation und Validation nötig.

[41] Laboratory for Safe and Secure Systems (LaS³) an der Ostbayerischen Technischen Hochschule Regensburg

Die N-Versionen-Annahme wurde von Leveson et al. 1990 (Brilliant, Knight, & Leveson, 1990) kritisch gewürdigt. Das DynaS³-Forschungsprojekt[42] untersucht mit maßgeblicher Mitwirkung des LaS³ die N-Versionen-Annahme im Kontext heutiger Entwicklungsprozesse. Eine kommerzielle Anwendung dieser Methodik außerhalb des Luftfahrtbereiches ist aufgrund der damit verbundenen erheblichen Kosten nicht Stand der Technik. Daher werden im Rahmen dieser Arbeit redundante Werkzeugketten nicht näher betrachtet.

3.1.3 Erste Bestandsaufnahme

Internationale Normen wie die IEC 61508 (Deutsche Norm, 2010a), ISO IEC 15504 (SPICE) (International Standard, 2004) und ISO IEC 12207 (International Standard, 2008) definieren erste Anforderungen an die Qualifikation und Zertifizierung von Werkzeugen. Domänenspezifische Standards wie die ISO 26262 (International Standard, 2011b) geben leicht unterschiedliche Argumentationen bezüglich der Qualifizierung von Werkzeugen. Wird dies berücksichtigt, ist es möglich, Strategien für die Qualifizierung und Zertifizierung von Werkzeugen zusammenzufassen:

- Zertifizierung des Entwicklungs- und Implementierungsprozesses des Werkzeuges z.B. nach IEC 61508 Teil 1 und 3,
- Formale Verifikation: Nachweis der Richtigkeit des Softwarewerkzeuges unter Verwendung mathematischer Beweisverfahren,
- „Fitness For Purpose"-Zertifizierung: Ein Werkzeug ist „fit for intended purpose", also grundsätzlich für die Verwendung in einer sicherheitsgerichteten Entwicklung geeignet,
- Bewertung des Softwareentwicklungsprozesses für die Werkzeugentwicklung (in Anlehnung an einen Sicherheitsstandard),
- Bewertung des Bug-Tracking- und Chance-Prozesses und
- Geeigneter Referenzworkflow für die Projektebene.

Die N-out-of-M-Architektur wird mit diversitären Entwicklungswerkzeugen entwickelt. Damit wird auf der einen Seite eine Reduktion der Qualifizierung- und Zertifizierungsaufwände für die Werkzeuge möglich, auf der anderen Seite erfordert die

[42] Förderprojekt: „Dynamische SW-Architekturen in Steuergeräten in Fahrzeugsystemen unter Berücksichtigung von Anforderungen zur Funktionalen Sicherheit", FKZ: 1752X07

komplexere N-out-of-M-Architektur selbst höhere Zertifizierungsaufwände im Rahmen eines Safety-Assessments.

Im weiteren Verlauf dieser Arbeit wird ein Vorgehen für einen erweiterten „Fitness For Purpose“-Ansatz entwickelt.

3.2 Klassifizierung von Softwareentwicklungswerkzeugen

Die Klassifizierung von komplexen Softwarewerkzeugen erfordert eine klare Strategie. In einem ersten Ansatz können Werkzeuge anhand ihres Einsatzzweckes klassifiziert werden, z.B. ist eine Kategorisierung des Werkzeugtyps wie folgt möglich:

- Entwicklungswerkzeuge
- Testwerkzeuge
- Begleitwerkzeuge bzw. unterstützende Werkzeuge

Diesen Kategorien werden die jeweils notwendigen Qualifizierungsmaßnahmen zugewiesen. Dieser grundlegende Ansatz, die Kritikalität eines Werkzeuges pauschal anhand des Werkzeugtyps festzulegen, wird in einigen Sicherheitsstandards (IEC 61508, DO-178B, etc.) angewandt. Diese Methode hat gerade bei komplexen Werkzeugen und Werkzeugketten Schwächen, da die Verwendung des Werkzeuges im Detail und die damit verbundene Kritikalität für das Endprodukt, das sicherheitsrelevante System, nicht berücksichtigt wird. Wird das Niveau der Qualifizierungsmaßnahmen pauschal zu hoch gesetzt, erhöhen sich signifikant die Entwicklungskosten, ohne positive Effekte auf die Sicherheit des Zielsystems. Gerade in hochsicherheitskritischen Bereichen wie dem Luftfahrt- oder Militärbereich hat dieses Vorgehen sicher seine Berechtigung, da dort die Stückkosten bei der Entwicklung eine untergeordnete Rolle spielen. Im Rahmen dieser Arbeit soll der pauschale Ansatz aber keine weitere Betrachtung erfahren. Auch in den frühen Entwurfsphasen der ISO 26262 wurde ein ähnlicher Ansatz beschrieben, bevor dann aber der im Weiteren betrachtete analytische Ansatz in den Standard aufgenommen wurde.

Der modernste Ansatz ist sicher der in Kapitel 3.1.1.3 vorgestellte aus der ISO 26262, wo jedem verwendeten Werkzeug als Ergebnis der Klassifizierung ein Tool Confidence Level zuordnet wird. Allerdings ist bei komplexen Werkzeugen (z.B. integrierten Werkzeugen für Requirements-Engineering und Konfigurationsmanagement wie PTC

Integrity) eine pauschale Aussage über die Kritikalität des Werkzeuges nur schwer möglich. Es ist eine feinere Granularität in der Betrachtung notwendig.

Deshalb wird die Werkzeugklassifizierung für komplexe Werkzeuge auf die Analyse einzelner Anwendungsfälle (Use Cases) heruntergebrochen. Dies soll mit Hilfe des Beispiels einer typischen Architektur für Werkzeuge zum Konfigurationsmanagement illustriert werden (siehe Abbildung 34):

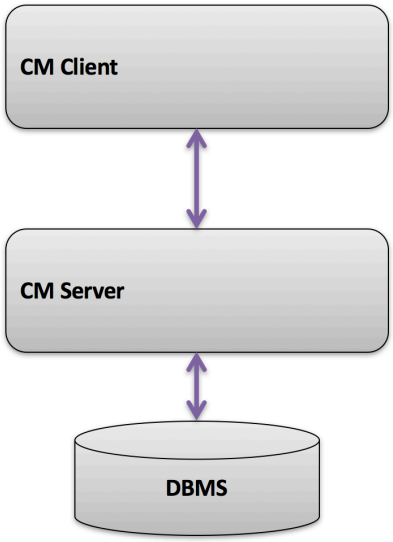

Abbildung 34: Typische Drei-Ebenen-Architektur eines Konfigurationsmanagement-Werkzeuges

Werkzeuge, mit denen die Funktionalität der Versionsverwaltung und des Konfigurationsmanagements implementiert wird, nutzen typischerweise eine Client-Server-Architektur (siehe (Tanenbaum, 2009) Seite 104ff). Dies ist erforderlich, um eine verteilte Multi-User-Anwendung (Tanenbaum & Van Steen, 2006) zu ermöglichen. Zur persistenten Speicherung der Daten wird üblicherweise auf eine etablierte Datenbanklösung (z.B. Oracle) zurückgegriffen, die über standardisierte Applikationsschnittstellen (API) wie SQL/ODBC und JDBC angesteuert werden (siehe (Bärwald, 2002) Seite 74f). So ergibt sich eine Struktur mit drei Architekturebenen, bestehend aus Datenbankmanagementsystem (DBMS), Konfigurationsmanagement-Server (CM-Server) und dem Anwender-Client (CM-Client).

3.2.1 Definition von relevanten Use Cases

Bei weniger komplexen Softwarewerkzeugen mit klar abgegrenzter Funktionalität ist es möglich, eine generische Aussage zur Zuverlässigkeit zu machen. Dieser Ansatz wird beispielsweise bei Code-Generatoren und Compilern angewendet. Allerdings ist auch hierbei eine Definition der gültigen Konfigurationen (z.B. Optionen und Switches) unabdingbar. Diese Rahmenbedingungen sind in einem Safety-Manual zu beschreiben und müssen bei der Verwendung des Werkzeuges im Entwicklungsprozess eingehalten und in das Safety-Assessment bei der Beurteilung der Funktionalen Sicherheit berücksichtigt werden. Ein Beispiel für dieses Vorgehen liefert die Fallstudie des Werkzeuges Target-Link in Kapitel 3.5.1.

Werden komplexe Werkzeuge oder Werkzeugketten betrachtet, ist eine pauschale, generische Aussage zur Werkzeugqualität bezogen auf ein spezifisches Anwendungsfeld nicht möglich. Deshalb muss bei einer generischen Werkzeugqualifizierung stets initial eine Analyse der relevanten Verwendungsfälle (Use Cases) erfolgen. Dieser Zusammenhang soll am Beispiel folgender Funktionalität eines Konfigurationsmanagementsystems analysiert werden: Ein- und Auschecken einer Revision. Schematisch ist die Prozesskette in der Abbildung 35 dargestellt:

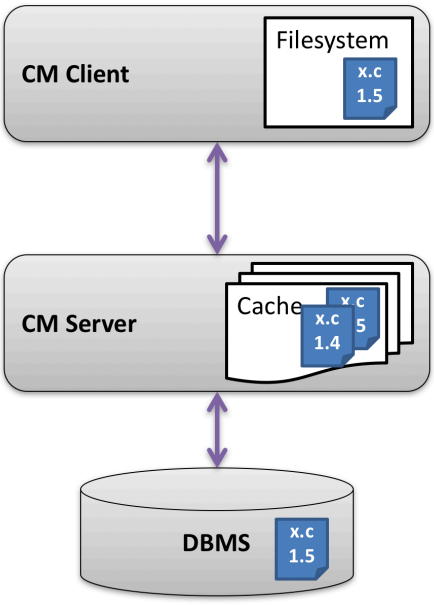

Abbildung 35: Schematische Darstellung des Vorganges "Ein- und Auschecken" einer Revision bei einem typischen Konfigurationsmanagementsystem mit Client-Server-Architektur

Folgende Verarbeitungsschritte sind zum Ablegen einer neuen Revision im Konfigurationsmanagementsystem (CM-System) notwendig:

1. Eine Quellcode-Datei mit dem Dateinamen „x.c“ wird dem Client des CM-Systems vom Benutzer übergeben, die Datei soll in Version 1.5 unter Versionskontrolle gestellt werden.
2. Die Datei wird vom CM-Client auf den CM-Server übertragen.
3. Die Datei wird auf dem CM-Server verarbeitet und für performance-optimierte Zugriffe als Version 1.5 im Cache des CM-Systems abgelegt.
4. Die Datei wird persistent im DBMS gespeichert.

Wird nun von einem Benutzer ein erneuter Zugriff auf die Datei „x.c“ in Version 1.5 angefordert, kann dies in folgender Prozesskette dargestellt werden.

5. Der Benutzer fordert über den CM-Client die Revision 1.5 der Datei „x.c“ an.
6. Der CM-Client stellt die Anfrage an den CM-Server.
7. Der CM-Server bearbeitet die Anfrage:
    a. Liegt die angeforderte Datei bereits im Cache des CM-Servers, wird die Datei aus dem Cache übertragen.
    b. Anderenfalls wird die Datei aus dem DBMS angefordert und an den CM-Server gesendet.
8. Die Datei wird vom CM-Server auf den CM-Client übertragen.

Es ist zu berücksichtigen, dass zu jedem Zeitpunkt verschiedene Versionen von ein und derselben Datei im Cache abgelegt sein können. Die daraus resultierenden Effekte und ggf. Risiken können nach ISO 26262 in einer Werkzeugklassifizierung (siehe 3.2.2) weitergehend analysiert werden. Nach der IEC 61508 könnten die Use Cases den Werkzeugklassen T1 bis T3 zugeordnet werden. Die detaillierte systematische Analyse nach ISO 26262 ist hier vorzuziehen und kann im Ergebnis auch in Rahmen der Anwendung der IEC 61508 herangezogen werden.

3.2.2 Klassifizierung eines Werkzeuges auf Basis eines definierten Use Cases nach ISO 26262

Wird die in der ISO 26262 Kapitel 3.1.1.3 (International Standard, 2011b) geforderte Methode zur Werkzeugklassifizierung bezogen auf den in Kapitel 3.2.1 eingeführten Use Case „Ein- und Auschecken einer Revision“ angewendet, ist das Vorgehen wie folgt:

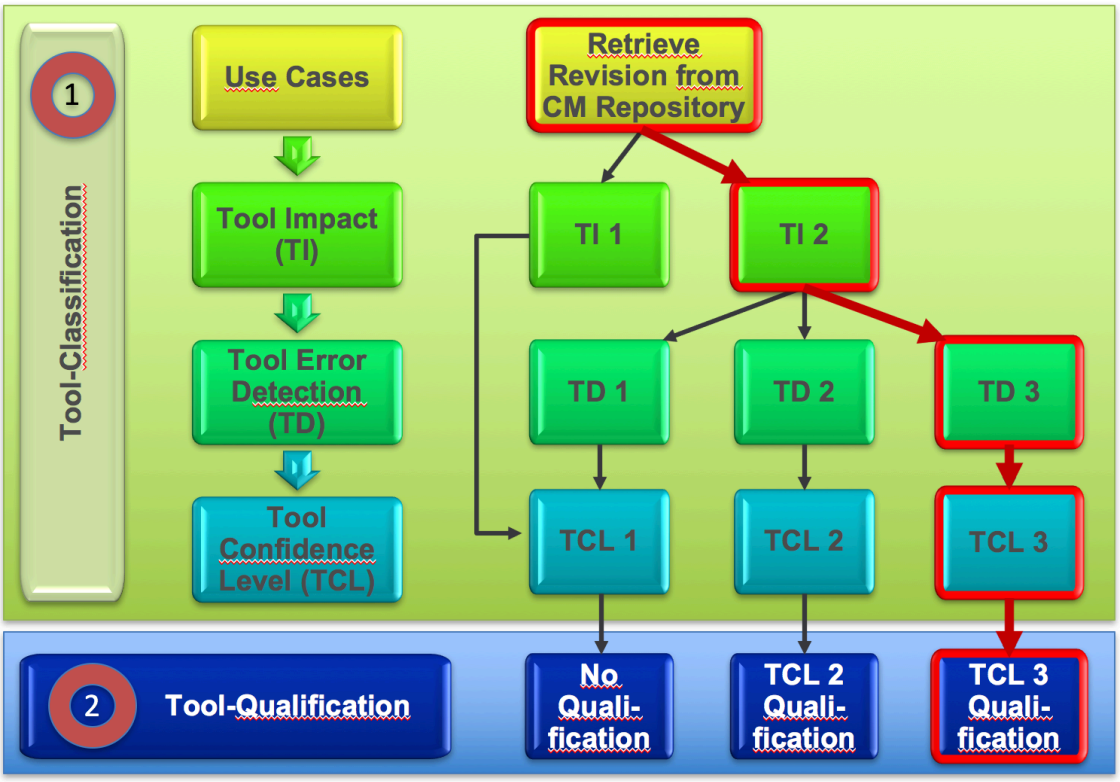

Abbildung 36: Toolklassifizierung für den Use Case "Ein- und Auschecken einer Revision" nach ISO 26262

Im ersten Schritt der Werkzeugklassifizierung wird der Einfluss eines Fehlers im Werkzeug (Tool Impact, TI) ermittelt. Im Beispiel heißt das, dass beim Vorgang des „Auscheckens“ einer Revision die Zielsoftware nicht der Spezifikation des erwarteten Revisionsstandes entsprechen kann. Wird z.B. lediglich eine falsche Version einer Quellcode-Datei zurückgeliefert, ist diese zwar syntaktisch korrekt und wird fehlerfrei vom Compiler verarbeitet. Trotzdem kann sich in der Zwischenzeit die Softwarespezifikation sowie die dazugehörige Testspezifikation geändert haben. Somit kann dies zu signifikanten Abweichungen des Systemverhaltens gegenüber der Systemspezifikation führen. Dieser Sachverhalt bedingt bei der Bestimmung der Tool Error Detection (TD) eine Bewertung von TD 3. Das heißt, solche Fehler können nicht oder nur zufällig gefunden werden. Die Einstufung in TD 3 führt somit zu einer Bewertung dieses Use Cases als TCL 3, was entsprechend dem Standard zu einer verpflichtenden Werkzeugqualifizierung mit den in Abbildung 32 aufgezeigten Optionen führt.

3.3 Generischer Ansatz für die Qualifizierung von Entwicklungswerkzeugen

Alle in den verschiedenen Sicherheitsstandards (siehe Kapitel 2.4.1) geforderten Maßnahmen hinsichtlich „Vertrauen“ in die Werkzeugqualität adressieren zuerst die

Projektebene, also die Entwicklung des sicherheitsrelevanten Systems. Dies ist für Eigenentwicklungen von Entwicklungswerkzeugen sicherlich sinnvoll. Da im Regelfall in den ausführenden Unternehmen nicht nur selbstentwickelte Softwarewerkzeuge zum Einsatz kommen, sondern zahlreiche Customer-Off-The-Shelf (COTS) Werkzeuge, ist es nicht immer zielführend, die Werkzeugqualifizierung auf Projektebene vollständig durchzuführen.

Es geht immer um den grundsätzlichen „Vertrauensgrad eines Werkzeuges". Dieser ist somit ein Maß für die Werkzeugqualität. In einer ersten Überlegung können die Kriterien bzw. Kategorien für den Vertrauensgrad wie folgt festgelegt werden.

Das Werkzeug ist:

a) zertifiziert (qualifiziert),
b) betriebsbewährt oder
c) getestet bzw. validiert.

Für die Kategorien a) und b) können darauf aufbauend minimale Eigenschaften definiert werden, wie in Tabelle 8 dargestellt:

Tabelle 8: Geforderte Eigenschaften betriebsbewährter und qualifizierter Werkzeuge nach (Bärwald, Inderst, Rosen, Smit-Wiesner, et al., 2011)

betriebsbewährt	**qualifiziert**
Bezug auf eine definierte Version des Werkzeuges	
Mehrere Instanzen in verschiedenen Projekten im Einsatz	Werkzeug kann eine Neuentwicklung sein
Geeignete Betriebsaufzeichnung, inkl. statistischer Daten	
Jeder bekannte Werkzeugfehler (sogenannter „Known Bug") muss an die Benutzer kommuniziert werden	
Identisches Nutzungsprofil, Analyse der verwendeten Werkzeugparameter / Konfiguration notwendig	Für definiertes Profil (Use Cases)
	Werkzeugvalidierung z.B. für Compiler-Valid-Suites

Können Werkzeuge nicht in diese beiden Kategorien einordnet werden, ist es immer möglich, Absicherungsmaßnahmen auf Projektebene vorzusehen, die die Verifikationsaufwände des Werkzeug-Outputs im Entwicklungsprozess des Zielsystems intensivieren (z.B. Back-to-Back-Tests, siehe Kapitel 3.5.1.4) oder es können diversitäre Ansätze (siehe Kapitel 3.1.2.3) gewählt werden.

Es empfiehlt sich eine generische Werkzeugbetrachtung, Use-Case-basiert (siehe Kapitel 3.2.1), in Kooperation mit dem Partner, der die Interna des Werkzeuges am besten kennt: dem Werkzeughersteller. Dabei können nach ISO 26262 die in den folgenden Kapiteln beschriebenen Methoden in Kombination angewendet werden.

3.3.1 Betriebsbewährte Werkzeuge (Proven in Use Tools)

Betriebsbewährtheit (Proven in Use) ist eine Basismethode, die in zahlreichen Sicherheitsstandards angewendet wird. Ihren Ursprung hat diese Methode zur Bewertung des Ausfallverhaltes von Hardware (z.B. ISO 26262-8:2011 Clause 14 „proven in use argument“) und wird in den verschiedenen (Sicherheits-)Standards auf ähnliche Art und Weise verwendet.

Für Hardware ist die Betriebserfahrung, vorausgesetzt es liegen repräsentative statistische Aufzeichnungen aus dem gesamten Pool an Feldgeräten vor, eine sehr wirkungsvolle Maßnahme. Im Softwarebereich hat diese Methode als alleinige Qualifizierungsmethode seine Grenzen, besonders bei Softwarewerkzeugen, wo eine deutlich höhere Release-Frequenz vorliegt, als bei sicherheitsrelevanter embedded Software.

Für eine gesamte Werkzeugekette ist das Proven-in-Use-Argument also schwierig, da Werkzeuge über einen längeren Zeitraum eingefroren werden müssten und darüber hinaus Betriebserfahrung aus zahlreichen Anwendungen mit den gleichen Use Cases dokumentiert und dem Projekt zugänglich sein müsste. Für einzelne Kernbibliotheken ist die Betriebsbewährtheit aber eine sehr wirkungsvolle Methode. So ist es z.B. nicht ungewöhnlich, dass Kernmodule eines Code-Generators mit mathematischen Basisfunktionen seit Jahrzehnten unverändert geblieben sind. In diesem Fall hat der Werkzeughersteller sehr detaillierte Erfahrungen mit seinem Werkzeug in der praktischen Anwendung. Wird die Methode Betriebsbewährtheit mit dem Testen des Werkzeuges

(siehe Kapitel 3.3.3) kombiniert, resultiert daraus ein sehr wirkungsvolles Maßnahmenpaket.

3.3.2 Assessment der Entwicklungsprozesse

Eine weitere Methode Werkzeuge nach ISO 26262 zu qualifizieren, ist das Assessment der Entwicklungsprozesse bei der Werkzeugentwicklung.

Dieses Vorgehen ist sehr effizient, da speziell im Automotive-Umfeld Bewertungen des Qualitätsmanagements und der Softwareentwicklungsprozesse und deren Einhaltung beim Hersteller seitens der OEM üblicherweise gefordert werden. Das heißt, dass Automotive-SPICE-Assessments (siehe Kapitel 2.4.2.3) de-facto-Industriestandard sind. In anderen Branchen wird CMMI (siehe Kapitel 2.4.2.3) bevorzugt, der generelle systematische Ansatz ist der gleiche.

Um diese Synergien zu nutzen, kann der in Kapitel 2.5.1.2 vorgestellte integrale Ansatz verwendet werden, der so gesehen eine Kombination mit der Maßnahme „Entwicklung nach einem Sicherheitsstandard“ (siehe Kapitel 3.3.4) ist.

Wichtig ist dabei, dass stets die Umsetzung der Entwicklungsprozesse für einen definierten Versionsstand des Werkzeuges begutachtet wird und nicht, wie häufig angenommen, eine pauschale Bewertung auf Unternehmensebene ausreichend ist. Bei neuen Releases ist dieses Assessment zu wiederholen.

3.3.3 Validierung des Werkzeuges (Black-Box-Ansatz)

Bei der Validierung eines Werkzeuges wird das Werkzeug einem Black-Box-Test unterzogen. Dieser Ansatz erfordert eine sehr detaillierte Kenntnis der Funktionalität des Werkzeuges. Werkzeughersteller bauen im Regelfall über die Jahre sehr umfangreiche und komplexe Datenbanken mit Testfallen auf, die auf Kundenfeedback und gefundenen Fehlern in der Entwicklung aber auch im Feld basieren. Typischerweise sind diese Tests dann automatisiert durchführbar und eignen sich somit auch für Regressionstests.

Des Weiteren gibt es den Ansatz, Werkzeuge bzw. Werkzeugketten projektspezifisch zu qualifizieren. Die Firma Validas AG erarbeitete im Rahmen des RECOMP-Förderprojektes[43] 2012 eine entsprechende Methodik, wo die möglichen Fehler einer Werkzeugkette in einem formalen Modell (Generic Error Model) abgebildet werden und die

[43] Reduced Certification Costs Using Trusted Multi-Core Platforms, EU-Förderprojekt von 2009 bis 2012 (ARTEMIS (Advanced Research & Technology for Embedded Intelligence and Systems), 2009)

Werkzeugkette dann im Rahmen der genutzten Funktionalitäten (Features) gegen diese Fehlermodelle getestet wird (siehe (Slotosch & Reiling, 2012) Seite 7ff). Das typische Vorgehen bei der projektspezifischen Qualifizierung kann nach (Texas Instruments & Validas AG, 2013) wie folgt zusammen gefasst werden:

Im ersten Schritt werden die Werkzeugfehler in die folgenden 3 Fehlerklassen unterteilt:

1. Potentielle Fehler durch ungenutzte Funktionalitäten,
2. Erkennbare und/oder vermeidbare Fehler sowie
3. Alle übrigen Fehler. Die Tool-Qualifizierung muss nun zeigen, dass die Fehler dieser Fehlerklasse im Einsatz keine Relevanz haben.

Ein Vorteil dabei ist, dass nur die im Projektumfeld relevanten Funktionen der Werkzeugkette in den Fokus der Betrachtung gelegt werden können.

Der sequentielle Ablauf dieses Qualifizierungsprozesses ist in der Abbildung 37 dargestellt. Dieses Vorgehen wurde u.a. beim SafeTI Compiler der Firma Texas Instruments angewendet (Texas Instruments, 2013). Das Projekt wurde von der Firma Validas AG begleitet.

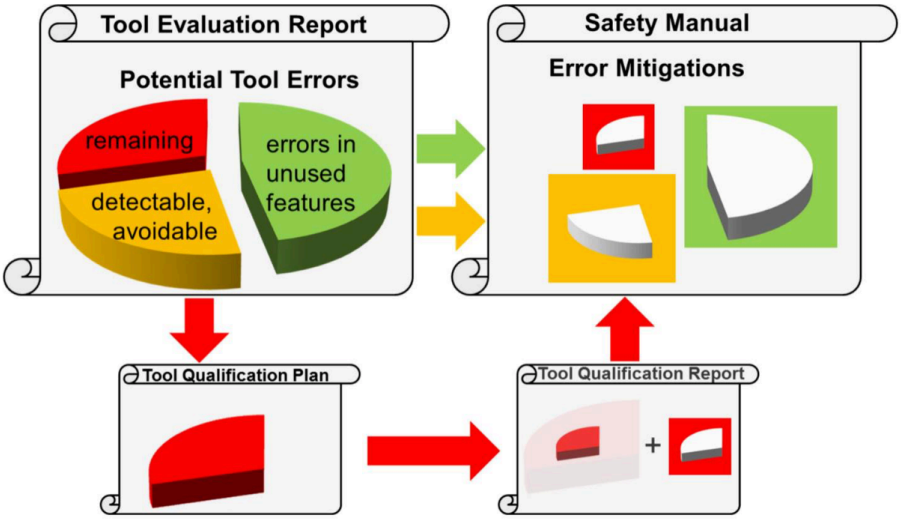

Abbildung 37: Typisches Vorgehen bei der projektspezifischen Qualifikation (Texas Instruments & Validas AG, 2013)

Wesentlicher Bestandteil ist ein Safety-Manual, welches beschreibt, wie die einzelnen Fehlerklassen behandelt werden. Für die Klasse der „übrigen Fehler" (remaining) wird im Qualification-Plan das weitere Vorgehen definiert. Darauf folgt im nächsten Schritt der Qualification-Report, der die Ergebnisse der Qualifizierungsmaßnahmen beschreibt. Für die Ausführung der Qualifikationstests werden typischerweise Testwerkzeuge wie

z.B. ACE SuperTest[44] verwendet (siehe (Beemster, 2013) Seite 11ff), die automatisiert und wiederholbar eingesetzt werden können und somit auch als Bestandteil eines Qualification-Kits dem Werkzeuganwender zur Verfügung gestellt werden können. Auf dieser Basis ist es dem Anwender dadurch möglich, die Vertrauenswürdigkeit der verwendeten Werkzeugkette projektspezifisch zu argumentieren.

Validierungssuiten bzw. Qualification-Kits werden häufig vom Werkzeughersteller zur Verfügung gestellt. Damit ist es möglich, Werkzeuge in ihrer Laufzeitumgebung beim Anwender bezüglich ihrer fehlerfreien Ausführung exemplarisch zu testen. Im Luftfahrtbereich ist die Verwendung von Qualification-Kits auf Projektebene eine seit langem bewährte Methode. Dieser Ansatz wird von einigen Werkzeugherstellern auch im Kontext der ISO 26262 verfolgt. Folgende Beispiele sind dabei zu nennen:

- SafeTI - Compiler Qualification Kit von Texas Instruments (Texas Instruments, 2013)
- Compiler Safety Package von ARM Keil (ARM, 2017)
- Wind River Diab Compiler Qualification Kit (WindRiver, 2013)
- BTC - „Validation Suite“(BTC Embedded Systems AG, 2017)

Da Werkzeugqualifizierungen, die exklusiv mit Hilfe einer Validierungs-Suite bzw. eines Qualification-Kits durchgeführt werden, immer projektspezifisch sind, wird dieser Ansatz im weiteren Verlauf dieser Arbeit nicht weiterverfolgt.

Die Methode Validierung ist essentiell für eine generische Werkzeugqualifizierung und Bestandteil des in Kapitel 3.4 vorgestellten Lösungsansatzes. Dabei werden allerdings weitere Kriterien, wie z.B. die Entwicklungsprozesse und die Qualitätssicherung bei der Entwicklung des Werkzeuges berücksichtigt, was bei der Verwendung eines Qualification-Kits keine Rolle spielt, da es sich um einen reinen Black-Box-Ansatz handelt.

3.3.4 Entwicklung nach einem Sicherheitsstandard

Die vollständige Entwicklung nach einem Sicherheitsstandard findet im Grunde nur im Luftfahrtbereich statt. Dies hängt mit der Anforderung z.B. der DO-178B zusammen, dass Level-A-Software, also Software höchster Kritikalität, mit Werkzeugen entwickelt werden muss, die nach den gleichen Level-A-Anforderungen implementiert wurden.

[44] ACE SuperTest ist ein „compiler test and validation suite“-Werkzeug der Firma SOLID SANDS B.V.

Ein generisches Werkzeug nach einem Sicherheitsstandard zu entwickeln, gestaltet sich nicht trivial. Eine signifikante Schwierigkeit ist es, generisch konkrete Sicherheitsanforderungen an das Werkzeug zu definieren, ohne die jeweiligen Zielsysteme zu kennen. Dies ist aber in der Sicherheitsanforderungsspezifikation (SRS) in den meisten relevanten Standards gefordert.

Im Automotive-Umfeld kann dieser Ansatz vernachlässigt werden und ist somit eher theoretischer Natur.

3.4 Kombination der verschiedenen Qualifizierungsansätze / TÜV SÜD Vorgehen

Ein generischer Qualifizierungsansatz kann auf Basis der Anforderungen der ISO 26262 hergeleitet werden. Der Standard ISO 26262 fordert „eine geeignete Auswahl bzw. Kombination“ der beschriebenen Maßnahmen. Theoretisch wäre die Auswahl einer Methode zulässig (XOR). Gerade die Zusammenarbeit mit dem Werkzeughersteller eröffnet die Möglichkeit, von den beim Hersteller vorliegenden umfassenden Erfahrungen zu profitieren und die Kombination mehrerer Methoden anzuwenden. Dieser Ansatz ist in der Abbildung 38 schematisch dargestellt:

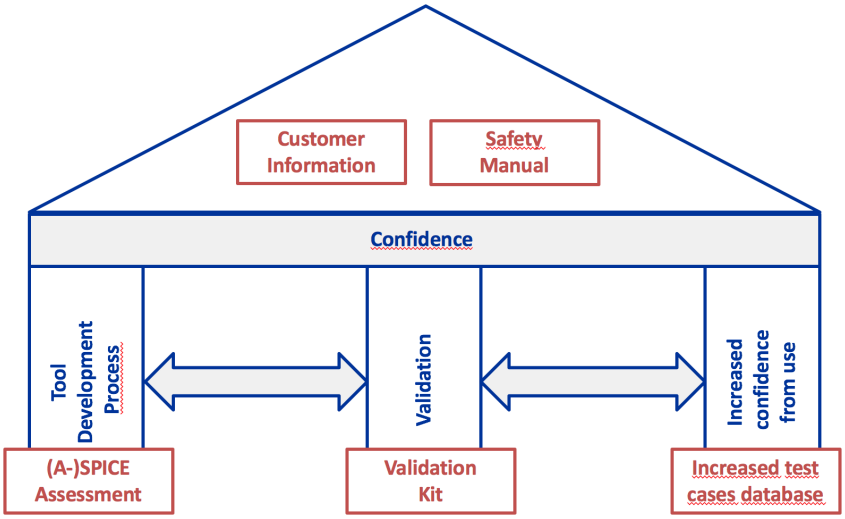

Abbildung 38: Kombination der verschiedenen Methoden zur Werkzeugqualifizierung in Anlehnung an die ISO 26262

Ziel ist es, die Vertrauenswürdigkeit (Confidence) des Werkzeuges nachzuweisen. Dieser Nachweis kann auf einer Kombination der nachfolgend genannten Methoden erfolgen:

a) Assessment des Entwicklungsprozesses (z.B. Automotive-SPICE)

b) Validierung des Werkzeuges mit Hilfe eines „Validation Kits“ durch den Werkzeughersteller und
c) Betriebserfahrung inklusive Informationsrücklauf aus dem Feld (wodurch die Werkzeughersteller umfangreiche Testfall- Datenbank aufbauen können)

Das „Mischungsverhältnis“ der verschiedenen Methoden ist dabei projektspezifisch. Bei einer kompletten Neuentwicklung werden verstärkt die Prozessaspekte in den Vordergrund gestellt. Betriebserfahrung ist dann eher von untergeordneter Bedeutung. Bei Werkzeugen mit einer langen Historie (bei Code-Generatoren sind Kernmodule mit mathematischen Basisfunktionen mit 20-jähriger Historie keine Seltenheit), wird der Fokus eher auf die Methoden Validierung und Betriebsbewährtheit gelegt.

Entwicklungsprozesse nach dem heutigen Stand der Technik können für ältere Code-Module (Legacy-Code) nicht erwartet werden, Prozesstreue bei Änderungen, die zu aktuellen Softwareständen führen, allerdings sehr wohl.

Die explizite Berücksichtigung der Entwicklungsprozesse schlägt die Brücke zur IEC 61508. Die Prozessanforderungen werden aus der ISO 26262-6 abgeleitet und entsprechen somit implizit auch den Prozessanforderungen auf der generischen Grundnorm IEC 61508-3 (Beurteilung des „Vertrauensgrades“ nach Band 4 Kapitel 7.4.4.4 und Kapitel 7.4.4.9). Abgeprüft werden die konsolidierten Anforderungen mit Hilfe einer Checkliste (siehe Kapitel 3.4.2).

In der Zukunft kann davon ausgegangen werden, dass zunehmend Open-Source-Werkzeuge wie z.B. die Eclipse[45]-Plattform für kommerzielle Softwareentwicklungsprojekte eingesetzt werden. Hier gilt es eine geeignete Qualifizierungsstrategie zu entwickeln, da eine Betrachtung der Entwicklungsprozesse in einem Open-Source-Umfeld nicht trivial ist. Ein Ansatz kann hier der Einsatz von qualifizierten bzw. zertifizierten Validierungs-Suiten (siehe Kapitel 3.3.3) sein. Die Entwicklung eines allgemeingültigen Ansatzes für die generische Qualifizierung von Open-Source-Werkzeugen ist nicht im Fokus dieser Arbeit und sollte im Rahmen weitergehender Forschung untersucht werden[46].

[45] http://www.eclipse.org

[46] Im Security-Umfeld gibt es bereits Zertifizierungen vom BSI von Open-Source-Produkten im Bereich von eCards. Allerdings stehen beim zugrunde liegenden Normenrahmen BSI TR-03124 eher technische als prozessuale Anforderungen im Vordergrund (Hühnlein, 2016).

Unabhängig vom gewählten Fokus bei der generischen Werkzeugqualifizierung gelten folgende allgemeine Anforderungen:

a) Es muss ein Safety-Manual vorliegen, das den Einsatz des Werkzeuges und die relevanten Use Cases beschreibt. Diese Anforderung wurde aus der IEC 61508 abgeleitet: In Band 4 Kapitel 7.4.4.4 wird gefordert, dass es eine Spezifikation und eine Produktdokumentation des Werkzeuges geben muss, inkl. einer Benutzerdokumentation.
b) Es muss ein Kundeninformationssystem installiert sein, was es dem Anwender jederzeit erlaubt, sich Informationen über bekannte Fehler in der verwendeten Werkzeugversion einzuholen. Dies wird heute typischerweise durch Online-Systeme realisiert, wo Release-Notes und Bug-Reports aufrufbar sind. Wichtig ist die Information betroffener Anwender, die unverzügliche und kostenfreie Fehlerbehebung ist nicht gefordert.

Diese Prozesse und Dokumente sind elementarer Bestandteil der Qualifizierungsdokumentation.

3.4.1 Generischer Qualifizierungsansatz und Zertifizierung

Softwareentwicklungswerkzeuge generisch zu qualifizieren, hat den größten Mehrwert bei Commercial-Off-The-Shelf-Werkzeugen (COTS), also bei Werkzeugen, die in zahlreichen Projekten und in vielen verschiedenen Unternehmen zum Einsatz kommen. Deshalb wird die generische Werkzeugqualifizierung im Regelfall immer im Auftrag oder zumindest mit aktiver Unterstützung des Werkzeugherstellers durchgeführt.

Basis einer Zertifizierung ist immer eine generische Qualifizierung wie in Kapitel 3.4 beschrieben. Das Vorgehen bei einer Zertifizierung gliedert sich in die Schritte Assessment und Zertifizierung.

3.4.1.1 Assessment

Das Assessment wird auf Basis der in Kapitel 3.4.2 im Detail beschriebenen Checkliste durchgeführt. Die Ergebnisse werden in einem Assessment-Bericht (Technischer Bericht) dokumentiert. Beim Assessment gilt das 4-Augen-Prinzip, d.h. alle Ergebnisse werden zwingend einem Review unterzogen. Gibt es im Assessment keine signifikanten Abweichungen, kann in einem nächsten Schritt eine Zertifizierung eingeleitet werden.

Assessment und Zertifizierung beziehen sich stets auf definierte Versionsstände, die im Assessment-Bericht eindeutig identifiziert werden müssen.

3.4.1.2 Zertifizierung

Ein Assessment ohne signifikante Abweichungen ist die Basis für eine Zertifizierung. Das Assessment muss durch eine entsprechend autorisierte Stelle erfolgen. Eine Akkreditierung als Prüflabor nach DIN EN ISO IEC 17025 (Deutsche Norm, 2005c) bzw. als Inspektionsstelle nach DIN EN ISO IEC 17020 (Deutsche Norm, 2012), ist dabei eine Voraussetzung. Das gewährleistet sowohl eine ausreichende Qualitätssicherung wie auch die Kompetenz der Assessoren.

Das Ergebnis der Zertifizierungsprozedur ist der Zertifikatsbericht und das Zertifikatsschmuckblatt. Im Zertifikatsbericht werden die Ergebnisse des Assessment-Berichtes zusammengefasst, technische und implementierungsnahe Details enthält der Bericht nicht. Es gibt einen eindeutigen Bezug zu den zertifizierten Versionen des Softwarewerkzeuges sowie einen Verweis auf die Nutzungsbedingungen (Conditions of Use), welche im Regelfall auch im Safety-Manual des Werkzeuges enthalten sind. Änderungen der Versionsstände führen im Regelfall zu einer Re-Zertifizierung. Der grundsätzliche Umgang mit Änderungen ist in Kapitel 3.4.3 beschrieben.

Beispiele für Werkzeugzertifizierungen sind in den Kapiteln 3.5.1 und 3.5.2 detailliert beschrieben. Die Abbildung 46 in Kapitel 3.5.1.2 zeigt exemplarisch ein Zertifikat, das über die Zertifizierstelle CRT3 der TÜV SÜD Product Service GmbH ausgestellt wurde.

3.4.2 Ableitung einer standardisierten Checkliste

Um ein standardisiertes, einheitliches Vorgehen bei der Werkzeugqualifizierung zu ermöglichen, sollte das Vorgehen in einen formalisierten Prozess überführt werden. Dies ist im Besonderen im Hinblick auf Nachvollziehbarkeit und Reproduzierbarkeit essentiell. Deshalb wurde eine standardisierte Checkliste[47] implementiert, die bei allen Werkzeug-Assessments und den sich daran ggf. anschließenden Zertifizierungen verwendet werden kann. Dieses Vorgehen, dass auf dem Ansatz der ISO 26262 basiert, inkludiert

[47] Die Checkliste zur generischen Tool-Qualifizierung und -Zertifizierung entstand in den Jahren 2011 bis 2015 bei TÜV SÜD Automotive unter der Leitung des Autors dieser Arbeit. An der Weiterentwicklung dieser Checkliste waren die folgenden Personen beteiligt: Doris Wild, Dr. Bernd Spanfelner, Nicole Pappler, Dr. Julian Wolf.

die Anforderungen aus der IEC 61508 und damit abgeleiteter Normen wie der IEC 62304, EN 50271 und der EN 50402, aber auch Aspekte aus Automotive-SPICE.

Die Checkliste orientiert sich an den Anforderungen an die Softwareentwicklungsprozesse aus der ISO 26262. Dabei werden sowohl die Aspekte der Qualitätssicherung aus ISO 26262 Band 2 (Functional Safety Management, FSM) als auch die Prozessanforderungen aus Band 6 adressiert. Diese werden allerdings auf die reinen Softwareumfänge reduziert, da es bei einem Softwareentwicklungswerkzeug keine Schnittstellen zu einer Systemebene bzw. Hardwareplattform gibt. Darüber hinaus sind die Anforderungen an Softwarewerkzeuge aus Band 6 eingearbeitet. Die Gliederung der Checkliste ist in Abbildung 39 dargestellt:

Abbildung 39: Gliederung der Werkzeug-Qualifizierungs-Checkliste aus dem Anhang B

Am Beginn der Checkliste gibt es eine Liste der vom Werkzeughersteller im Rahmen des Assessment zur Verfügung gestellten Dokumente. Im Folgenden soll die grundsätzliche Systematik der Checkliste exemplarisch am Beispiel des Softwarearchitekturdesigns in Kapitel 3.4 besprochen werden. Im ersten Schritt gibt es konkrete Anforderungen, die aus dem Standard direkt abgeleitet werden können.

Software architectural design/high level software design
Requirements
R11. Software architectural design/high level software design - Allocation of the high level functional requirements to the components - Specification for the main components, their functionality and their interfaces - Refined software requirements list

Abbildung 40: Abgeleitete Anforderungen an das Architekturdesign (Beispiel aus Kapitel 3.4 der Checkliste aus dem Anhang B)

Die Anforderung R11 ist ein Kondensat des Kapitels 7 („Software architectural design“) der ISO 26262 Band 6. Basierend auf diesen Top-Level-Anforderungen gibt es zu jedem Themenblock einen Fragenkatalog, der im Verlauf des Assessments abgearbeitet wird (siehe Abbildung 41).

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there an architecture management process described?	#73
OK	Is there a tool that supports the architecture description?	#74
OK	Does the tool support the (bi-directional) tracing between requirements and architecture?	#75
OK	Does the architecture process include a change management to allow propagation of changes due to technical issues?	#76
OK	Are there guidelines for using tools?	#77
OK	Are there guidelines for using methods?	#78
General TUEV SUED comments: -		

Abbildung 41: Detaillierter Fragenkatalog für das Werkzeug-Assessment (Beispiel aus Kapitel 3.4 der Checkliste aus dem Anhang B)

Der Fragenkatalog wurde auf Basis der Anforderungen an die Funktionale Sicherheit (ISO 26262) erstellt. Die Fragen wurden kontinuierlich erweitert. So haben auch Aspekte

aus anderen Standards (z.B. Automotive-SPICE) Einfluss auf die inhaltliche Gestaltung der Checkliste. Die Frage #75 adressiert beispielsweise die sogenannte „bi-direktionale Tracability“. Das bedeutet, dass Anforderungen (über die Architektur- und Design-ebene) nicht nur auf den Quellcode nachvollziehbar verlinkt werden müssen, sondern dass diese Nachvollziehbarkeit auch vom Code zu den Anforderungen gewährleistet sein muss (bi-direktional). Es muss also für jedes Quellcode-Element immer eine klare Zuordnung zu einem Design-Element geben, vom Design zur Architektur und schließ-lich von der Architektur zu den Anforderungen. Somit ist sichergestellt, dass es für jedes Code-Element auch eine Anforderung gibt. Diese Forderung kommt aus der SPICE-Welt. Die Standards der Funktionalen Sicherheit fordern im Regelfall nur die einfache Tracability von den Anforderungen zum Code, was zumindest sicherstellt, dass alle An-forderungen implementiert und getestet wurden.

Der Assessor soll mit diesem Fragenkatalog durch die Begutachtung geführt werden. Das gewährleistet Vollständigkeit in der Betrachtung, aber auch Vergleichbarkeit über mehrere Projekte. Alle Fragen haben eine eineindeutige Nummer. Die Bewertung der einzelnen Fragen kann wie folgt erfolgen:

- OK: keine Abweichungen
- POK (partiell OK): im Grunde in Ordnung, kleine Abweichungen
- NOK: grobe Abweichung

Im nächsten Schritt können offene Punkte erfasst werden, die in einem weiteren Asses-sment-Loop zur Wiedervorlage dokumentiert werden können (siehe Abbildung 42).

Work Product: Software architecture process description
TODOs: -

Abbildung 42: Erfassung offener Punkte (Beispiel aus Kapitel 3.4 der Checkliste aus dem An-hang B)

Am Ende eines jeden Kapitels gibt es eine Zusammenfassung der einzelnen Fragengrup-pen mit der finalen Bewertung („Judgement“) und Empfehlungen für Verbesserungen in der Zukunft („Obligations for the next assessment“), die aber zur aktuellen Begutach-tung keine gravierende Abweichung darstellt (siehe Abbildung 43).

Objective	Obligations for next assessment	Judgement
Work product: Architecture management process description	None	OK
Work Product: Software Architecture	None	OK
Work Product: Sample Check of Architecture	None	OK
Work Product: Architecture verification specification and report	None	OK

Abbildung 43: Zusammenfassung der Bewertungen in Kapitel 3.4 der Checkliste aus dem Anhang B

Durch die Verwendung der Checkliste ist das in dieser Arbeit beschriebene Vorgehen beim Assessment der Werkzeugentwicklung strukturiert und führt zu einem allgemeingültigen und vergleichbaren Ansatz. Das Template der vollständigen Checkliste befindet sich im Anhang B dieser Arbeit.

3.4.3 Umgang mit Änderungen an qualifizierten Softwarewerkzeugen

Änderungen am qualifizierten Werkzeug müssen über einen definierten und freigegebenen Prozess erfolgen. Dabei ist zwischen Major- und Minor-Releases sowie Hotfixes zu unterscheiden.

Ein neues Major-Release basiert immer auf neuen in das Werkzeug zu implementierenden Funktionalitäten, die mit Änderungen an den Produktanforderungen verbunden sind. Es handelt sich also um signifikante Änderungen am Produkt, so dass in den Change-Prozess der Entwicklung eingestiegen wird.

Bei Minor-Releases handelt es sich um kleinere Anpassungen am Produkt, welche aber ebenfalls mit dem vollständigen Change-Prozess umgesetzt werden. Auch hier sind ggf. Anpassungen an den Produktanforderungen in minimalem Umfang nötig.

Bei einem Hotfix handelt sich um eine Ad-hoc-Änderung am Produkt, z.B. aufgrund eines im Feld aufgetretenen Problems. Hotfixes müssen nicht zwingend an alle Anwender ausgeliefert werden. Sie können auch kundenspezifisch erstellt werden.

Besonders der Hotfix-Prozess muss zyklisch überwacht werden. TÜV SÜD verwendet hierzu ein speziell für Softwarewerkzeuge angepasstes Verfahren im Rahmen der

jährlich wiederkehrenden „Fertigungsstätten-Inspektion“[48], die in der Prüf- und Zertifizier-Ordnung der TÜV SÜD Gruppe (siehe (TÜV SÜD AG, 2016) Seite 7 Abschnitt A-1.11) verpflichtend ist. Hierbei wird in einem Audit beim Werkzeughersteller in besonderem Maße der Hotfix-Prozess und das Qualitäts- bzw. Release-Management begutachtet.

3.5 Fallstudien

Eine in vielen technischen Bereichen, vor allem in den Bereichen der ITK-Technologien übliche Praxis ist es, anhand der Untersuchung von Praxisbeispielen in Form von Fallbeispielen (Case Studies) zielführende und verwertbare Ergebnisse zu erreichen. Die Tauglichkeit der in den vorherigen Kapiteln hergeleiteten Methodik soll deshalb nach diesem Verfahren untersucht werden. Eine Fallstudie gliedert sich typischerweise in 5 Phasen (Runeson & Höst, 2009), die wie in Tabelle 9 in dieser Arbeit umgesetzt wurde.

Tabelle 9: Umsetzung der Case-Study-Methode im Rahmen dieser Arbeit

Phase	Umsetzung im Rahmen dieser Arbeit
1. Definition der Ziele	• Qualitative Nachweise, dass die erarbeitete Methodik praxistauglich ist (Kapitel 3.5)
2. Vorbereitung der Datenerhebung	• Auswahl der Fallbeispiele (Kapitel 3.5)
3. Datenerhebung	• Beschreibung der Projektumsetzung (siehe 3.5.1.1ff und 3.5.2.1ff)
4. Analyse	• Bewertung der Tauglichkeit (siehe 3.5.1.5 und 3.5.2.4)
5. Ergebnisbericht	• Zusammenfassung (siehe 3.5.3)

Im Bereich des Software-Engineerings ist es aufgrund der schnellen technologischen Entwicklung üblich, die Wirksamkeit einer neuen Methode oder Technologie anhand von Fallstudien zu untersuchen. Dabei können einzelne Fallbeispiele auf Basis festgelegter qualitativer Kriterien untersucht werden. Das ist besonders dann sinnvoll, wenn die

[48] Das Verfahren basiert auf der Checkliste des „permanenten Dokumentes“ „CIG 023 – Factory Inspektion Report“ der EEPCA (EEPCA - the European Electrical Products Certification Association, 2014), das bei allen renommierten Prüforganisationen im Rahmen von Produktzertifizierungen verwendet wird.

neue Methode oder Technologie erst am Anfang einer größeren Verbreitung steht und statistische Erhebungen oder auch Expertenbefragungen keine repräsentativen Ergebnisse liefern.

Die in den vorangegangenen Kapiteln vorgestellte Methodik wurde ab 2009 erfolgreich in vielen verschiedenen Zertifizierungsprojekten bei der TÜV SÜD Automotive GmbH unter Federführung des Autors angewendet. Zu nennen sind u.a. folgende kommerziellen Produkte:

- dSPACE TargetLink (2009)
- BTC Embedded Tester (2009)
- ETAS ASCET (2010)
- MKS / PTC Integrity (2011)
- PRQA QAC (2011)
- IBM Rational Doors (2012)
- GAIO Coverage Master (2012)

In den folgenden beiden Unterkapiteln wird die praktische Anwendung an zwei Beispielen aufgezeigt. Dafür wurden die Werkzeuge TargetLink von der Firma dSPACE, das als Pilotprojekt fungiert hat, und Integrity von der Firma PTC ausgewählt, wo die Methodik hin zur Use-Case-basierten Werkzeugqualifizierung weiterentwickelt wurde. In Anlehnung an (Yin, 2003) handelt es sich beim gewählten Ansatz um zwei getrennte holistische Falluntersuchungen (siehe Abbildung 44), mit dem jeweiligen Bezug auf das untersuchte Softwareentwicklungswerkzeug.

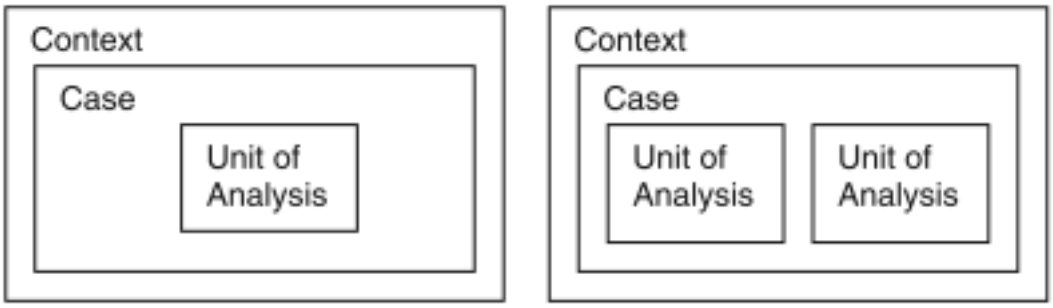

Abbildung 44: Klassifizierung von Fallstudien: Holistische (links) und eingebettete Fallstudie (rechts) aus (Runeson & Höst, 2009) nach (Yin, 2003)

Die Bewertung wird qualitativ vorgenommen. Das qualitative Maß der Untersuchung ist die Fragestellung: Können Softwareentwicklungswerkzeuge nach dem beschriebenen Methodenmix generisch qualifiziert werden?

3.5.1 Fallstudie dSPACE TargetLink[49]

TargetLink von der Firma dSPACE aus Paderborn ist ein Serien-Code-Generator, der zur automatischen Hochsprachen-Code-Generierung aus MATLAB/Simulink/Stateflow Modellen verwendet wird. TargetLink hat eine hohe Verbreitung im Automotive-Bereich.

Um die zunehmende Komplexität bei der Systementwicklung beherrschen zu können, verändert sich das Entwicklungsvorgehen zunehmend. Einst wurden Hochsprachen wie C eingeführt, da die Komplexität der Applikationen in Assembler nicht mehr darstellbar war. Heute werden die Hochsprachen durch den Einsatz einer zusätzlichen Abstraktionsebene (Modellebene) ersetzt oder zumindest ergänzt. Daher wird in der modernen Softwareentwicklung zunehmend der modellbasierte Ansatz (siehe Abbildung 45) verwendet.

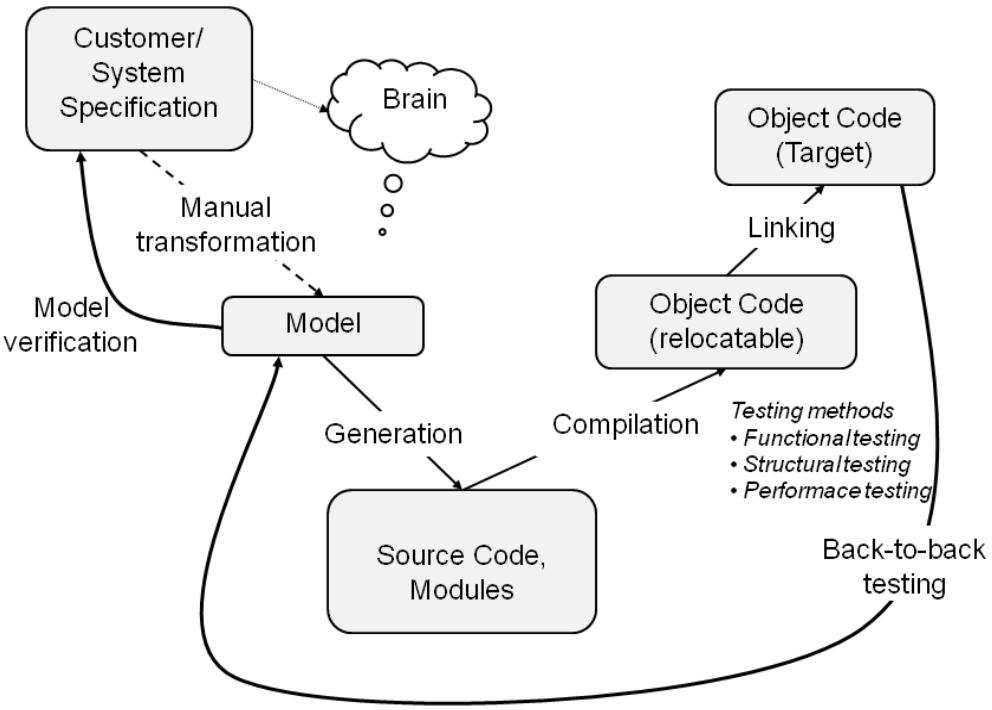

Abbildung 45: Workflow bei einer modellbasierten Softwareentwicklung mit Seriencodegenerierung (Bärwald & Beine, 2010)

Um modellbasierte Entwicklung effizient durchzuführen, bedarf es einer geeigneten Auswahl von Softwareentwicklungswerkzeugen, die die spezifischen

[49] Dieses Kapitel wurde in wesentlichen Teilen unter dem Titel „Sichere Codegenerierung“ als gemeinsame Veröffentlichung mit Michael Beine im Jahr 2010 in der Fachzeitschrift „Hanser Automotive“ publiziert (Bärwald & Beine, 2010).

Entwicklungsphasen möglichst optimal unterstützen. Die eingesetzten Werkzeuge müssen an den Schnittstellen harmonieren und ein definiertes Vertrauensniveau bzgl. Zuverlässigkeit, Qualität und somit auch in Bezug auf die Sicherheit haben. Trotz allem ist eine Absicherung (Verifikation) des generierten Codes erforderlich.

Alle relevanten internationalen Standards zur Funktionalen Sicherheit fordern einen solchen Vertrauensnachweis (siehe Kapitel 3.1.1ff). Eine effiziente Möglichkeit, die geforderte Vertrauenswürdigkeit eines Softwareentwicklungswerkzeugs nachzuweisen, ist eine generische Qualifizierung für bewährte, weit verbreitete Nutzungsprofile durch eine unabhängige und neutrale Prüfstelle. Dieser Ansatz wurde vom Werkzeughersteller dSPACE und der TÜV SÜD Automotive GmbH bei der Zertifizierung des Seriencode-Generators TargetLink gewählt.


ZERTIFIKAT ◆ CERTIFICATE ◆ 認證 ◆ СЕРТИФИКАТ ◆ CERTIFICADO ◆ CERTIFICAT

CERTIFICATE

No. Z10 09 08 71483 001

Holder of Certificate: dSPACE GmbH
Technologiepark 25
33100 Paderborn
GERMANY

Factory(ies): 71483

Certification Mark:

Product: Software Tool for Safety Related Development

Model(s): TargetLink

Parameters: The code generator is fit for purpose to develop safety related software according to IEC 61508 and/or ISO DIS 26262.

The report no. DP74761C is a mandatory part of this certificate.

Tested according to: IEC 61508-3:1998 (Fitness for purpose)
ISO DIS 26262:2009 (Fitness for purpose)

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.: DP74761C

Date, 2009-08-20

Page 1 of 1

TÜV SÜD Product Service GmbH · Zertifizierstelle · Ridlerstrasse 65 · 80339 München · Germany

Abbildung 46: TÜV SÜD Zertifikat für das Werkzeug TargetLink der Firma dSPACE

3.5.1.1 Werkzeugklassifizierung am Beispiel TargetLink

Nach ISO 26262-8 (siehe Kapitel 3.1.1.3) muss der Vertrauensgrad (Tool Confidence Level, TCL) eines jeden verwendeten Werkzeuges ermittelt werden. Für Seriencode-Generatoren gilt, dass Werkzeugfehler Fehler in der Target-Software verursachen können. Das heißt, es besteht Fehlerpotential (Tool Impact TI von 2). Die Aufdeckungswahrscheinlichkeit von Werkzeugfehlern im Projekt wird maßgeblich bestimmt durch die Qualität der Verifikationsmaßnahmen, die auf den Werkzeug-Output angewandt werden. Um eine hohe Fehleraufdeckungswahrscheinlichkeit (Tool Error Detection TD von

1) erreichen zu können, wurde von der Firma dSPACE ein Referenzworkflow für die Verwendung des Werkzeuges TargetLink in sicherheitsgerichteten Entwicklungen (Beine, 2009) erstellt. Das führt zu einem Werkzeugvertrauensgrad (TCL) von 1. Somit sind bei Umsetzung des Referenzworkflows, der Bestandteil des Zertifizierungsumfangs ist, keine zusätzlichen Qualifizierungsmaßnahmen auf Projektebene erforderlich.

3.5.1.2 Anwendung der Qualifizierungsmethode „Fitness For Purpose“

Im Rahmen der Zertifizierung des Werkzeugs TargetLink wurde für die Werkzeugqualifizierung die Methode „Fitness For Purpose“ angewendet. Das bedeutet, dass ein entsprechend zertifiziertes Softwarewerkzeug prinzipiell unter definierten Rahmenbedingungen, den sogenannten „Conditions of Use“, für die Entwicklung sicherheitsgerichteter Systeme geeignet ist (siehe Kapitel 3.1.2.1). Ziel einer solchen Zertifizierung ist, den Werkzeuganwendern einen generischen und projektunabhängigen Eignungsnachweis zur Verfügung zu stellen. Eine oftmals aufwendige Werkzeugqualifizierung durch anwenderseitige Betriebserfahrungen (Proven in Use) oder umfangreiche und aufwendige Testverfahren (z.B. „Black-Box-Validierung“) durch den Werkzeuganwender ist nicht mehr notwendig. Der Anwender muss die Eignung im Hinblick auf das Zusammenspiel der eingesetzten Werkzeuge für seinen Verwendungszweck analysieren und dokumentieren.

3.5.1.3 Die TargetLink-Zertifizierung

Die Einhaltung der im TargetLink-Referenzworkflow beschriebenen Verifikations- und Validierungsmaßnahmen erlaubt es, potentielle Werkzeugfehler mit hoher Wahrscheinlichkeit aufzudecken. Die Anforderungen für eine generische Werkzeugzertifizierung gehen aber deutlich über eine Bewertung des Referenzworkflows hinaus (siehe Kapitel 3.4). Deshalb wurde im Rahmen der Zertifizierung der gesamte TargetLink-Entwicklungsprozess begutachtet.

Insgesamt wurden folgende Aspekte betrachtet:

- TargetLink-Entwicklungsprozess mit Schwerpunkt auf dem Produkttest,
- Fehlermeldewesen und Umgang mit gemeldeten Werkzeugfehlern sowie
- Referenzworkflow.

3.5.1.4 Der TargetLink-Referenzworkflow

Der Referenzworkflow für die Entwicklung sicherheitsrelevanter Software unter Verwendung der etablierten Werkzeugkette MATLAB/Simulink/Stateflow und TargetLink (Beine, 2009) gibt Orientierung und Hilfestellung bei der Erfüllung von Anforderungen der Sicherheitsstandards ISO 26262 (International Standard, 2011b) und IEC 61508 (Deutsche Norm, 2010a) beim Einsatz modellbasierter Methoden und Werkzeuge. Er basiert auf Best Practices und Erfahrungen aus erfolgreich abgeschlossenen, sicherheitsrelevanten Serienprojekten. Anwender von modellbasierten Methoden können sich direkt an dem Workflow orientieren und zeigen, wie die verschiedenen Methoden und Maßnahmen im konkreten Projekt umgesetzt werden. Dabei sind Abweichungen von den vorgeschlagenen Methoden möglich. Diese müssen jedoch entsprechend begründet und dokumentiert werden.

Bestandteil des Workflowdokuments ist eine detaillierte, tabellarische Übersicht über sämtliche für die modellbasierte Softwareentwicklung relevanten Anforderungen aus Teil 6 der ISO 26262. In zusätzlichen Spalten ist hier jeweils aufgeführt, welche Methoden und Maßnahmen auf Modell- bzw. auf Codeebene zur Verfügung stehen, um diese zu adressieren. Diese Referenztabellen können als Template für eine Art Checkliste im Projekt verwendet werden und sind hilfreich bei der Abstimmung mit den für die Funktionale Sicherheit zuständigen Mitarbeitern, deren natürlicher Zugang und Blick auf den Prozess üblicherweise aus Sicht der Norm erfolgt.

Betrachtet werden die Phasen „Design“ und „Implementierung“, von den textuellen Anforderungen über das Modell als ausführbare Spezifikation und Input für die Code-Generierung bis hin zum Objekt-Code, sowie die Phasen „Verifikation des Designs“ und „Verifikation der Implementierung“. Weitere Kerninhalte sind die Nachverfolgbarkeit von Anforderungen, Anforderungen zum Aufbau von Modellen unter Architekturaspekten inkl. Modellierungs- und Codierungsrichtlinien, Wartbarkeit und Testbarkeit sowie der normgerechten Durchführung von Software-Unit- und Integrationstests.

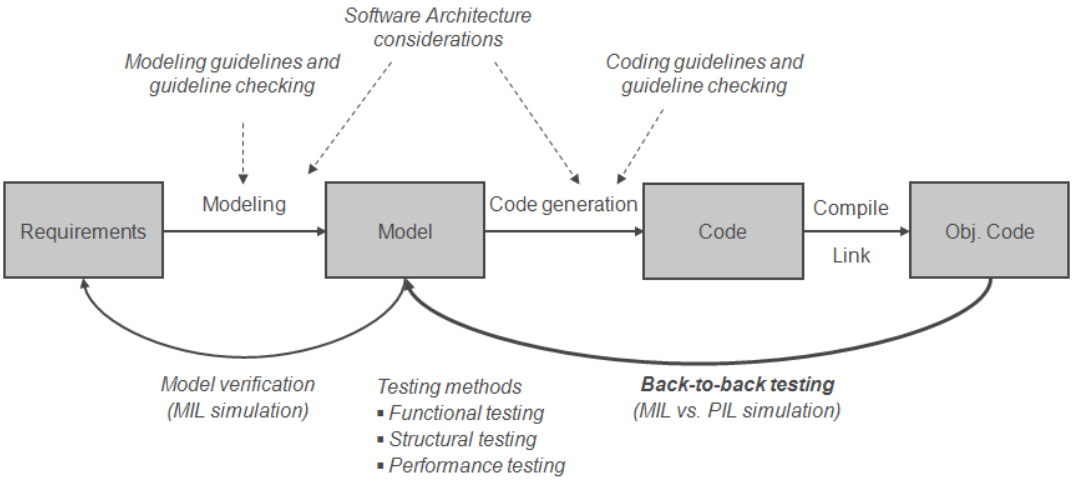

Abbildung 47: Elemente des Referenzworkflows. Die „Back-to-Back" Tests zwischen Objekt-Code und Modell sind die zentrale Absicherungsmethode für den Code (Bärwald & Beine, 2010)

Die Nachverfolgbarkeit der Anforderungen über alle Phasen der Produktentwicklung hinweg, also von den Anforderungen in textueller Form über das modellbasierte Design bis hin zur Implementierung auf Code-Ebene sowie für das anforderungsbasierte Testen, ist ein zentraler Aspekt bei der Entwicklung sicherheitsrelevanter Software. Im Hinblick auf die modellbasierte Entwicklung ist die Möglichkeit, Anforderungen in das Modell und vom Modell aus nachverfolgen zu können, von großer Bedeutung. Entsprechende prozessunterstützende Methoden und Maßnahmen sind schon vor Beginn der eigentlichen Softwareentwicklung einzuplanen.

Ebenfalls vor Beginn der eigentlichen Softwareentwicklung erfolgt die Auswahl und Festlegung von Modellierungs- und Codierungsrichtlinien. ISO 26262 empfiehlt die Verwendung und Einhaltung von Design- und Codierungsrichtlinien sowohl für die Modellierung als auch für den Code. Der Referenzworkflow stellt etablierte Richtliniendokumente wie die MISRA-Modellierungsrichtlinien für TargetLink (Motor Industry Software Reliability Association, 2007) vor und befasst sich damit, wie die Einhaltung von Richtlinien sichergestellt werden kann.

Ein weiterer Schwerpunkt des Referenzworkflows liegt in der Beschreibung eines normgerechten Vorgehens zur Absicherung der Software anhand von Software-Unit- und Integrationstests. Ein wesentlicher Aspekt besteht dabei in der Absicherung der automatischen Übersetzung des Modells in Steuergeräte-Code.

Die zentrale Methode nachzuweisen, dass der Code das validierte Modell korrekt und fehlerfrei umsetzt und das Verhalten des generierten Codes auf der Zielhardware dem

Verhalten des Modells entspricht, ist die Durchführung von Vergleichstests zwischen Modell und erzeugtem Code, sogenannten „Back-to-Back"-Tests. „Back-to-Back"-Tests werden von der ISO 26262 explizit für alle Integritätslevel (ASIL) empfohlen. Durch zusätzliche Analyse der Code-Abdeckung wird gleichzeitig gezeigt, dass der Code keine unspezifizierten Bestandteile enthält.

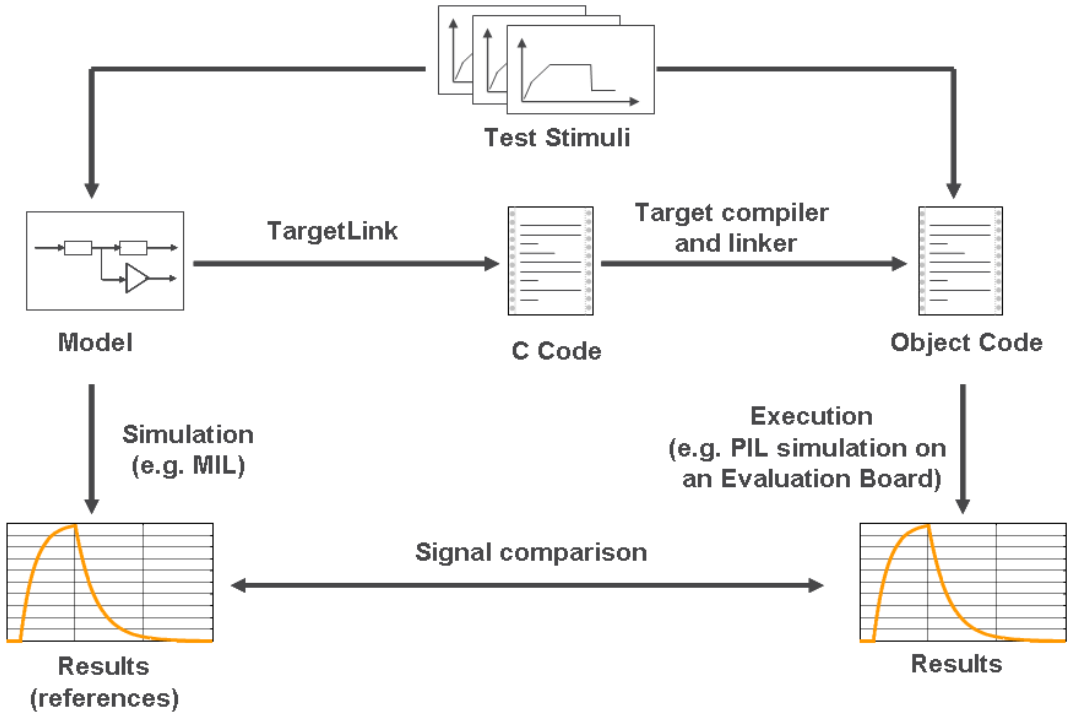

Abbildung 48: „Back-to-Back"-Teststrategie. Die Ergebnisse der sogenannten Model In The Loop (MIL)-Simulationen dienen als Referenzwerte und werden verglichen mit den Ergebnissen der Simulation des erzeugten Codes, wofür Processor In The Loop (PIL)-Simulationen eine ideale Plattform darstellen (Bärwald & Beine, 2010)

Der „Back-to-Back"-Test ist ein essentielles Mittel, um das Verhalten von Modell und Zielsoftware vergleichen zu können. Dabei müssen Testspezifikation, Testfälle und die Testergebnisse der verschiedenen Abstraktionsebenen (Model In The Loop (MIL), Software In The Loop (SIL) und Processor In The Loop (PIL)) verwaltet werden. Dies ist nur durch die Verwendung von Softwarewerkzeugen wie z.B. dem Embedded Tester der Firma BTC Embedded System AG effizient möglich (siehe Abbildung 49).

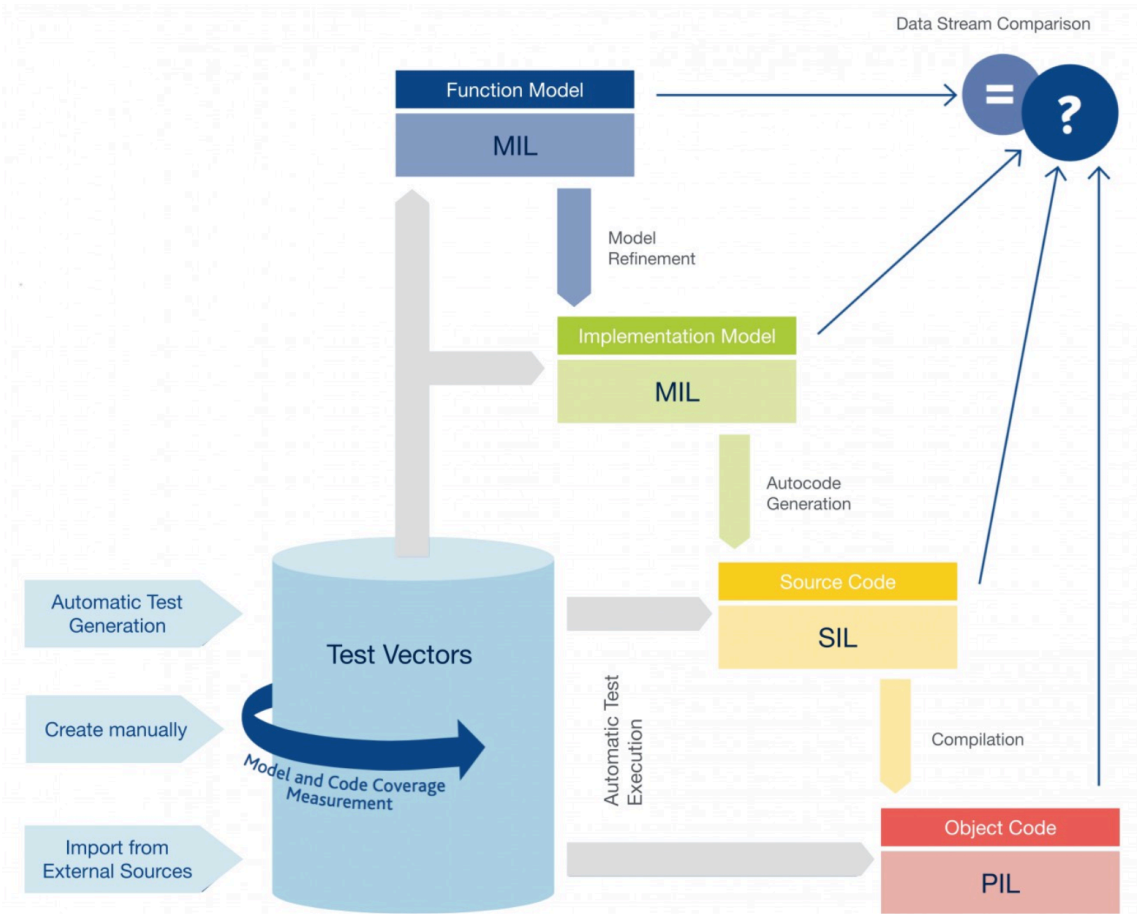

Abbildung 49: Werkzeugunterstützung beim „Back-to-Back"-Test (Bildquelle: BTC Embedded Systems AG)

Auch für die „Back-to-Back"-Testwerkzeuge gilt, dass sie ggf. qualifiziert werden müssen, denn ein Versagen eines Testwerkzeuges kann zu kritischen Zuständen des Zielsystems führen, wenn Fehler in der zu testenden Software nicht erkannt werden.

3.5.1.5 Zusammenfassung der Fallstudie TargetLink

Modellbasierte Entwicklung ist ein geeigneter und in der Praxis bereits vielfach verwendeter Ansatz für die Entwicklung sicherheitsgerichteter Systeme. Durch die Zertifizierung des automatischen Code-Generators TargetLink wird dessen grundsätzliche Verwendbarkeit in sicherheitsgerichteten Entwicklungen nach IEC 61508 und ISO 26262 bestätigt. Durch die Verwendung von zertifizierten Softwarewerkzeugen entsteht ein erheblicher Mehrwert für den Anwender, da zusätzliche und aufwendige Qualifizierungsmaßnahmen entfallen. Die Bereitstellung des Referenzworkflows zur normgerechten Verwendung von TargetLink erleichtert darüber hinaus den Nachweis, dass das Entwicklungsvorgehen für die Entwicklung sicherheitsrelevanter Software geeignet ist.

Eine Orientierung am Referenzworkflow ist aus dem Blickwinkel der Qualitätssicherung auch für nicht sicherheitsrelevante Anwendungen eine adäquate Maßnahme.

3.5.2 Fallstudie MKS/PTC Integrity

Integrity der Firma PTC[50] ist ein sogenanntes Application Lifecycle Management Werkzeug (ALM), also ein Werkzeug, das den gesamten Softwareentwicklungsprozess unterstützt.

„PTC Integrity is an application lifecycle management solution that manages all global software development processes and connects all software engineering artifacts, including requirements, models, code, and testing, to ensure comprehensive lifecycle traceability."[51]

Den Kern der Integrity-Suite bilden folgende Funktionalitäten:

1. Konfigurationsmanagement (Versionsverwaltung) mit Anbindung an einen Prozessworkflow und
2. Anforderungsmanagement.

Beide Aspekte sind in das Werkzeug voll integriert. Das heißt, dass sich alle Entwicklungsdokumente entsprechend des vom Anwender implementierten Prozessworkflows abbilden und ablegen lassen und vollständig auf die Anforderungen verlinkt werden können (Tracability).
Die Firma PTC verwendet zur Entwicklung des Werkzeuges selbst eine Instanz der eigenen Entwicklungsumgebung. Dadurch werden eindrucksvoll die Funktionalität und vor allem die Eignung für einen agilen Entwicklungsansatz gezeigt.

3.5.2.1 Use-Case-basierte Werkzeugqualifizierung

Für PTC Integrity wurden mehrere Use Cases definiert, die den Einsatz des Werkzeuges in sicherheitsgerichteten Entwicklungen analysieren. Für diese Use Cases werden Anforderungen an die Funktionalität des Werkzeuges definiert. Gegen diese (funktionalen) Anforderungen muss jedes Release vollständig getestet werden. Dies ist elementarer Bestandteil jeder (Re-)Qualifizierung von neuen Werkzeugversionen.

[50] Vormals MKS Integrity, MKS wurde 2011 von PTC akquiriert.
[51] Zitat aus (PTC, 2013)

Am Beispiel des Use Case „Ein- und Auschecken einer Revision“ aus Kapitel 3.2ff, wird klar ersichtlich, dass das Werkzeug für diesen Anwendungsfall qualifiziert werden muss. In einer ersten Überlegung kann das Werkzeug bzgl. dieses Szenarios z.B. durch Testen (Methode Validierung) qualifiziert werden. Eine alternative Möglichkeit ist, eine fehlererkennende Maßnahme (Risk Mitigation Measure) zu implementieren. Das ist eine zusätzliche Softwarefunktion, die die Fehleraufdeckung des im Use Case beschriebenen potentiellen Fehlverhaltens des Werkzeuges erkennt und darauf reagiert. Soll erkannt werden, ob eine Revision korrekt ins System eingecheckt und anschließend wieder korrekt ausgecheckt wurde, kann eine Absicherung per Hash bzw. Checksumme (z.B. CRC) implementiert werden (siehe Abbildung 50). Diese Checksumme muss separat von den eigentlichen Daten gespeichert werden.

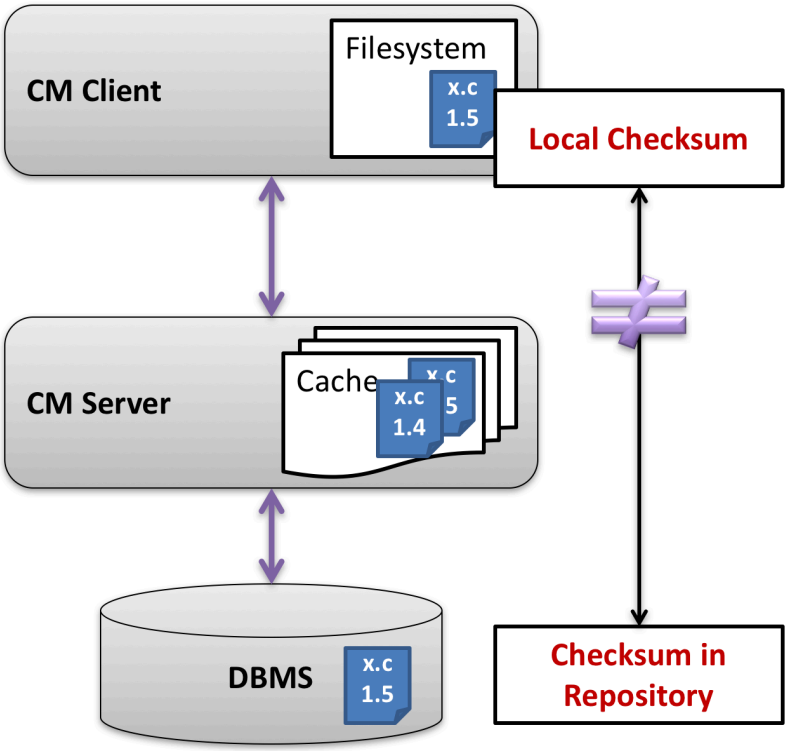

Abbildung 50: Erweiterung des Werkzeuges PTC Integrity um eine integrierte Absicherungsmaßnahme für den Use Case „Ein- und Auschecken einer Revision“

Mit dieser technischen Maßnahme ist es möglich, dieses Fehlerszenario zu eliminieren, da alle verfälschten bzw. fehlerhaften Check-Outs vom System detektiert werden. Wird in der Folge eine Methode zur Toolqualifizierung angewendet, ist es hinreichend, wenn nicht das gesamte Werkzeug hinsichtlich des Fehlerszenarios qualifiziert wird, sondern der Nachweis erbracht wird, dass die Fehlerbeherrschungsmaßnahme wirksam ist. Das

heißt, es werden die „Risk Mitigation Measures“ qualifiziert, also die integrierten Absicherungsmaßnahmen.

Die Wirksamkeit einer Absicherungsmaßnahme kann mit Hilfe einer Maßnahmenkarte strukturiert und analysiert werden. In der Abbildung 51 ist diese Methodik am eben eingeführten Beispiel exemplarisch gezeigt.

UC 8 – Check Out a Revision from CM Tool

Description
When synchronizing the local workspace a specific revision is requested from the CM tool. The content of this revision is transferred from the database via the CM server to the local client where it is written to the local filesystem.
Possible tool errors can result in corruption of the file content or the wrong revision being transferred to the local workspace.

Tool Impact (TI)			
TI Value	TI 2		
TI Rationale	When the wrong revision is compiled, defects or unintended behavior can be introduced to the binary code.		
Tool Error Detection (TD)			
Without Risk Mitigations		***With Applied Risk Mitigations***	
TD Value	TD 3	***TD Value***	**TD 1**
TD Rationale	Software often still compiles successfully. Test coverage uncertain if wrong revision is retrieved.	***TD Rationale***	**Checksum functionality automatically discovers any inconsistencies of the local files with the originals in the repository**
Tool Confidence Level (TCL)			
Without RM	TCL 3	***Effective TCL***	**TCL 1**
TCL Rationale	Correct functionality of check out mechanism in CM Tool is critical	***TCL Rationale***	**Reliable automatic error detection is in place.**

Abbildung 51: Maßnahmenkarte für den Use Case „Ein- und Auschecken einer Revision“

Diesem Ansatz folgend, sinkt der TCL des Gesamtwerkzeuges von TCL 3 auf TCL 1. Das heißt, dass für das Gesamtwerkzeug keine Qualifizierungsmaßnahme normativ gefordert ist. Allerdings muss die Absicherungsmaßnahme, die Implementierung des CRC mit den Anforderungen an einen TCL 3 qualifiziert werden. Der TCL der Absicherungsmaßnahme wird gewissermaßen vom ursprünglichen Use Case ohne Absicherung vererbt.

3.5.2.2 Die PTC Integrity Zertifizierung

Der generische Qualifizierungsansatz beim Werkzeug PTC Integrity basiert zum einen auf einer umfassenden Bewertung der Entwicklungsprozesse (siehe Kapitel 3.5.2.3) und zum anderen auf einem nachgewiesenen 100%-Test für alle definierten „Risk Mitigation Measures“ (siehe Kapitel 3.5.2.1). Für diese bewerteten Use Cases gibt es eine definierte Testspezifikation, denn diese Tests sind bei jedem neuen Release vollständig zu wiederholen. Dies steht bei jedem Assessment eines neuen Software-Releases auf der Agenda.

Im Wesentlichen wird das dadurch ermöglicht, dass die in der Entwicklung verwendete Test-Suite regressionstestfähig ist und somit automatisiert ausgeführt werden kann.
Zum Umfang der Zertifizierung gehört darüber hinaus ein umfangreiches Safety-Manual, das den Anwender über den Zertifizierungsumfang und die zu verwendenden Konfigurationen informiert. Des Weiteren gibt es ein Fehlermeldewesen, das die Anwender web-basiert über den Umgang mit gemeldeten Werkzeugfehlern in Kenntnis setzt.
Die in Kapitel 3.4.2 vorgestellte Checkliste findet auch bei der Zertifizierung von PTC Integrity seine Anwendung.

3.5.2.3 Integriertes Assessment Automotive-SPICE und Funktionale Sicherheit unter Einbeziehung eines agilen Entwicklungsvorgehens

Eine Besonderheit beim Werkzeug PTC Integrity ist, dass das Vor-Ort-Assessment der Funktionalen Sicherheit zusammen mit einem Automotive-SPICE-Assessment durchgeführt wird.

„Automotive development organizations are managing complex requirements for embedded software in their products, … Continuing to certify PTC Integrity with automotive safety industry standards like A-SPICE and ISO 26262 will help our customers comply with functional safety standards to ensure safe software development processes.“[52]

Das prinzipielle Vorgehen bei diesem Ansatz ist in Kapitel 2.5.1.3 detailliert beschrieben. Die dabei entstandene Ergebnismatrix des Automotive-SPICE-Assessment (siehe beispielhaft Abbildung 52) ist Bestandteil des Assessment-Berichtes für die Funktionale Sicherheit und ist somit den Werkzeuganwendern zugänglich.

[52] Zitat aus (PTC, 2013)

ID	Process name	PA 1.1	PA 2.1	PA 2.2	Capability Level
MAN.3	Project management	F	F	F	2
ENG.4	Software requirements analysis	F	F	F	2
ENG.5	Software design	L	F	L	1
ENG.6	Software construction	F	F	F	2
ENG.7	Software integration	F	F	F	2
ENG.8	Software testing	F	L	F	2
SUP.1	Quality assurance	F	L	F	2
SUP.8	Configuration management	F	F	F	2
SUP.9	Problem resolution management	F	F	F	2
SUP.10	Change request management	F	F	F	2

Abbildung 52: Exemplarische Bewertungsmatrix eines Automotive-SPICE-Assessment aus (Bärwald & Bräuchle, 2013)

Bei der Entwicklung von PTC Integrity wird ein agiler Entwicklungsansatz verfolgt. Das bedeutet, dass mehrere SCRUM-Teams an verschiedenen Standorten in die Entwicklung der Software involviert sind. Der Entwicklungsprozess ist in diesem Fall nicht an das V-Modell angelehnt, sondern sehr iterativ (siehe Abbildung 53). Die Anforderungen eines neuen Releases werden in den verschiedenen Backlogs[53] gesammelt. Die Implementierung wird in sogenannte Iterationen heruntergebrochen, die aus mehreren Sprints bestehen, in denen in den verschiedenen Teams die neuen Funktionen umgesetzt werden. Am Ende einer jeden Iteration gibt es eine Integration und einen Regressionstest.

Die Verwendung agiler Methoden ist im Umfeld der Funktionalen Sicherheit problemlos möglich und findet zunehmend in Praxis Anwendung (siehe Kapitel 2.4.3). Ein starres Festhalten am V-Modell wäre sicher nicht zeitgemäß. Wichtig ist aber, dass die in den Standards der Funktionalen Sicherheit (z.B. ISO 26262) geforderte Dokumentation (Work Products) korrekt erstellt wird, die Teststrategie passend definiert ist und ein Qualitätsmanagement (bzw. Functional Safety Management) installiert ist.

[53] Ein Backlog ist in der agilen Entwicklung die Sammlung von Produktanforderungen, die zur Implementierung anstehen. Dabei kann in verschiedene Arten von Backlogs geclustert werden: z.B. Technical Backlog (für Basisfunktionen), Maintainance Backlog (für Produktpflege) und New Product Backlog (für neue Produktfunktionalitäten).

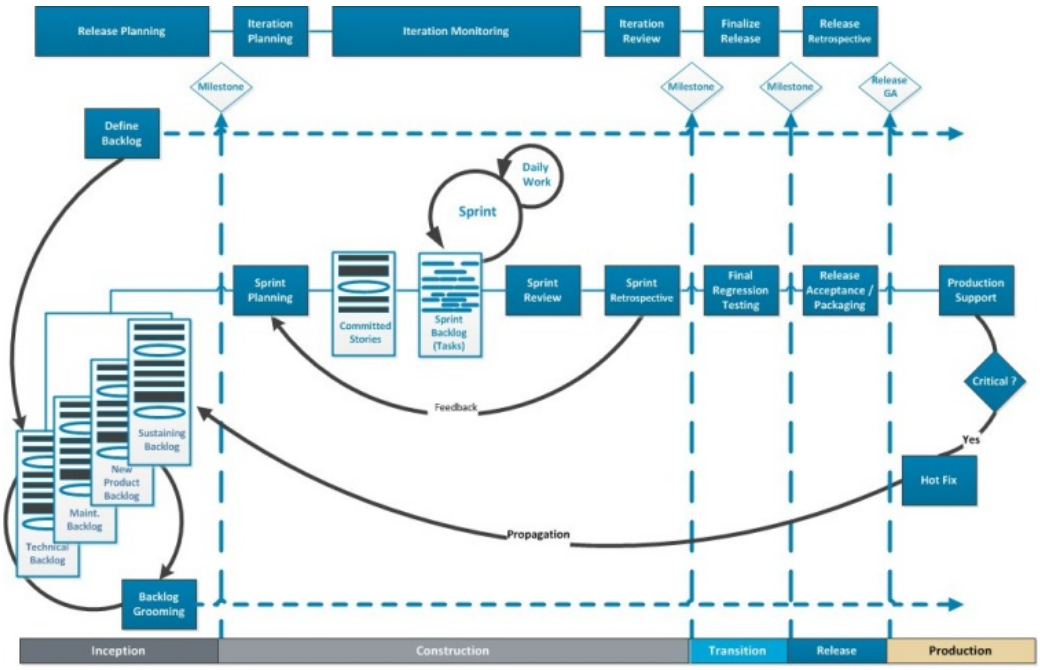

Abbildung 53: Agile Methodik bei der Entwicklung von PTC Integrity aus (Bärwald & Bräuchle, 2013)

Ein interessantes Detail ist, dass PTC Integrity selbst für die eigene Entwicklung verwendet wird. Dadurch wird eindrucksvoll die Funktionalität des Werkzeuges gezeigt. Besonders fällt dabei die Eignung für agile Softwareentwicklungsprojekte auf.

3.5.2.4 Zusammenfassung der Fallstudie PTC Integrity

Die Erfahrungen bei der Zertifizierung von PTC Integrity haben gezeigt, dass die Use-Case-basierte Werkzeugbetrachtung erheblich den Qualifizierungsaufwand reduziert. Die Herausforderung liegt dabei darin, alle relevanten Use Cases zu bestimmen und zu analysieren und die daraus abgeleiteten „Risk-Migation-Measures“ in die Teststrategie einzubetten und deren Ausführung bei jedem Release sicherzustellen.

Ein weiterer Aspekt ist das integrierte Assessment bzgl. Funktionaler Sicherheit und Automotive-SPICE, das maßgebliche Synergien hebt und den Assessment-Aufwand reduziert. Hierbei wird die in dieser Arbeit vorgestellte Checkliste (siehe Kapitel 3.4.2) aus Anhang B verwendet.

3.5.3 Vergleich und Bewertung der Fallstudien

In beiden Fallstudien ist das Assessment der Entwicklungsprozesse die zentrale Maßnahme. Dadurch wird grundsätzlich die Qualität der Software sichergestellt. Im Fall von TargetLink kann das Werkzeug im Einsatz zusätzlich durch einen Back-To-Back-Test abgesichert werden. Dieser ist im sogenannten Referenzworkflow vom Werkzeug-

hersteller gefordert und beschrieben. Im Fall von PTC Integrity liegt der Schwerpunkt auf der Analyse der relevanten Use Cases und der daraus abgeleiteten sicherheitsrelevanten Funktionalitäten, deren möglichst fehlerfreie Umsetzung auch über ein Prozess-Assessment sichergestellt wird.

Bei anderen Werkzeugzertifizierungen, z.B. BTC Embedded Tester, liegt der Schwerpunkt auf einer Validierungssuite, mit der das Werkzeug in der produktiven Installation beim Anwender mit einem Black-Box-Ansatz validiert werden kann.

Mit dem in diesem Kapitel vorgestellten Qualifizierungsvorgehen können sowohl Entwicklungswerkzeuge als auch Testwerkzeuge qualifiziert werden, unabhängig von deren Kategorisierung. Die Maßnahmen entsprechen immer den Anforderungen an den höchsten Kritikalitätsgrad TCL 3 nach ISO 26262-8.

In beiden Fallstudien konnte gezeigt werden, dass die jeweils angewendete Qualifizierungsstrategie erfolgreich und eine generische, projekt- und anwenderunabhängige Werkzeugqualifizierung möglich ist. Damit kann auch die Brücke zu den Forschungshypothesen 1a-c geschlagen werden.

3.6 Einsatz zertifizierter Softwarewerkzeuge in Praxis

Der Einsatz von zertifizierten Softwareentwicklungswerkzeugen bei der Entwicklung sicherheitsgerichteter Systeme entwickelt sich zwischenzeitlich zum Stand der Technik. In der Automobilindustrie werden von allen OEMs Sicherheitsnachweise bzw. ein Assessment der Funktionalen Sicherheit gefordert, wenn es sich um sicherheitsrelevante Systeme handelt. Bei der Entwicklung der Software, die im Regelfall durch einen Zulieferer implementiert wird, ist es somit wichtig, die Anforderungen der geforderten Standards zu erfüllen. Das schließt Automotive-SPICE aber auch die ISO 26262 ein. Somit ist die Vertrauenswürdigkeit der Werkzeugkette ein wichtiger Aspekt. Dies soll an folgendem Beispiel aus der Praxis illustriert werden.

Das Werkzeug ETAS ASCET wird bei der Entwicklung von Leistungselektronik und Elektromaschinen für Elektro- und Hybridantriebe des Produktbereichs GS-EH (Electric Vehicle and Hybrid Systems) innerhalb des Geschäftsbereichs Gasoline Systems der Robert Bosch GmbH eingesetzt (Bulach, Martin, Raichle, & Dr. Lauff, 2013). Die so entwickelten Systeme sind mittlerweile erfolgreich bei den Fahrzeugherstellern VW, Porsche und Peugeot im Serieneinsatz. Dabei werden die in ASCET modellierten Funktionen für

Motor-Generatorsteuerungen automatisiert in C-Code überführt. Dies wird mittels des zertifizierten Code-Generators ETAS ASCET durchgeführt (siehe Abbildung 54).

„Für die Entwicklung der neuen, dritten Leistungselektronikgeneration wird eine ASIL D-Zertifizierung konform zur ISO 26262-Norm für die Entwicklung von sicherheitsrelevanten Systemen für Pkw angestrebt. Mit dem Code-Generator, dessen Einsatztauglichkeit für die Entwicklung von ASIL D-Systemen vom TÜV SÜD zertifiziert wurde, und der zusätzlichen Integration einer Model-Coverage-Analyse lässt sich die ISO 26262-konforme Softwareentwicklung in idealer Weise mit Hilfe von ASCET abbilden.“[54]

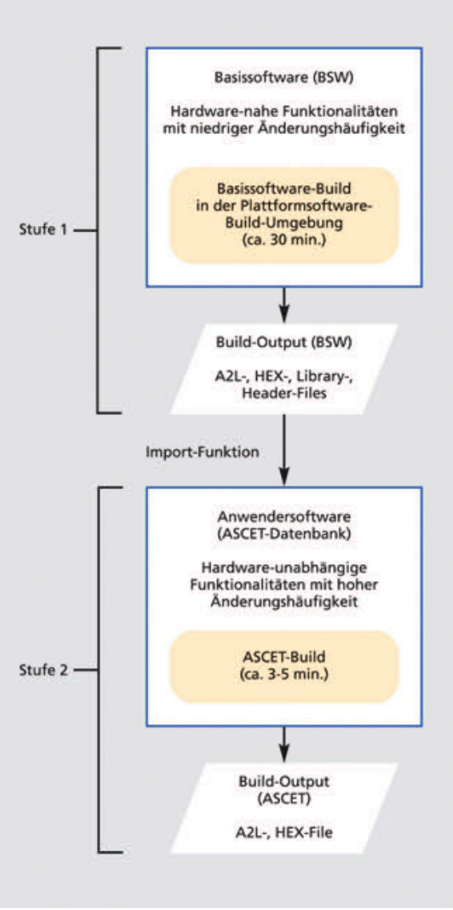

Abbildung 54: Einsatz von ETAS ASCET im zweistufigen Build-Prozess für die Software der Leistungselektronikplattform des Geschäftsbereichs Gasoline Systems der Robert Bosch GmbH nach (Bulach et al., 2013)

[54] Zitat aus (Bulach et al., 2013)

Die Firma DENSO in Japan führte im Jahr 2011 einen unternehmensweit gültigen Prozess für Funktionale Sicherheit ein. In diesem Rahmen erfolgte eine Analyse der Anforderungen an eine geeignete Werkzeugkette, um Projekte mit Bezug zur Funktionalen Sicherheit erfolgreich und effizient implementieren zu können. Die Zielsetzung dabei war, bei zunehmend komplexeren Projekten weiterhin die Maxime „Quality First" unter Beachtung der Standards für die Funktionale Sicherheit umzusetzen. Im Rahmen dieses Prozesses fiel die Auswahl im Bereich der ALM-Werkzeuge auf PTC Integrity (Fukuda, 2012), das mit dem in dieser Arbeit entwickelten Ansatz generisch qualifiziert und zertifiziert wurde.

Diese Beispiele zeigen: Der Einsatz von generisch qualifizierten und zertifizierten Softwarewerkzeugen ist in der Praxis akzeptiert und nimmt stetig an Verbreitung zu.

3.7 Zusammenfassung und Ausblick für den Umgang mit komplexen Werkzeugketten

In der Praxis kann die Bewertung von ggf. hunderten eingesetzten Werkzeugen in großen Konzernen zu hohen Aufwänden und daraus resultierend erheblichen Kosten führen. Deshalb ist es von besonderer Bedeutung, die spezifische Verwendung und die Einsatzumgebung unternehmensspezifisch zu analysieren, z.B. durch den in Kapitel 3.3.3. vorgestellten Ansatz der Werkzeugqualifizierung der Firma Validas AG. Die Aufwände können allerdings nochmals erheblich reduziert werden, wenn generisch vorqualifizierte Softwareentwicklungswerkzeuge Verwendung finden, die nach dem in diesem Kapitel vorgestellten Ansatz qualifiziert werden.

Eine essentielle Rolle bei der Bewertung von Werkzeugqualität und damit der Zuverlässigkeit der eingesetzten Werkzeuge ist die Bewertung der Entwicklungsprozesse beim Werkzeughersteller und deren Einhaltung. Besonders im Umfeld der Automobilindustrie hat sich dafür die Prozessreife-Bewertung nach dem Automotive-SPICE-Schema als Stand der Technik etabliert. Bei näherer Betrachtung der Kriterien bei einem SPICE-Assessment ist eine hohe Überschneidung mit den Anforderungen an die Entwicklungsprozesse in den einschlägigen Standards zur Funktionalen Sicherheit (ISO 26262, IEC 61508, etc.) erkennbar. Vor diesem Hintergrund erscheint es sinnvoll, die Bewertung der Prozessanforderungen und SPICE-Assessments zu integrieren. Ein entsprechender Ansatz wurde im Kapitel 2.5.1 vorgestellt.

Es wurde gezeigt, dass es möglich ist, Werkzeuge generisch zu bewerten. Dies kann unabhängig von konkreten Projekten erfolgen. Damit wurde die Forschungshypothese 1b bestätigt. Dass die generische Werkzeugqualifizierung keinem starren Schema folgt, sondern eine geeignete Mischung von verschiedenen Methoden ist, stützt die Forschungshypothese 1c.

3.7.1 Zukünftige integrierte Assessments: Funktionale Sicherheit und Automotive-SPICE

In der Zukunft wird die Betrachtung allgemeiner Qualitätsstandards und das Assessment der Funktionalen Sicherheit zunehmend verschmelzen. Für jegliche Softwareentwicklungen im Automotive-Bereich ist Automotive-SPICE de-facto-Standard. Das heißt, jeder Zulieferer muss sich mit dem Thema SPICE in irgendeiner Form auseinandersetzen. Eine separate Entwicklungsmethodik für sicherheitsgerichtete Systeme und Standardsysteme erscheint wenig wirtschaftlich. Hierzu werden auch Hinweise des VDA QMC[55] in der Zukunft erwartet. Als wesentlich effizienter erweist es sich, einen vollständigen Safety-Prozess durch geeignetes Tailoring für Projekte mit geringerer Sicherheitskritikalität und Standardprojekte nutzbar zu machen.

Ein weiterer Technologietreiber wird die zunehmende Verwendung agiler Methoden sein, die auch vermehrt für die Entwicklung sicherheitsrelevanter Systeme angewendet werden (Reinelt, Mishr, & Breault, 2017). In diesem Zusammenhang lassen sich weitere Fragestellungen ableiten, die zu weiteren Forschungsarbeiten führen sollten. Speziell das Mapping von agilen Entwicklungsprozessen und den dabei entstandenen Entwicklungsdokumenten mit den Anforderungen aus den einschlägigen Sicherheitsstandards, die eher ein V-Modell als Entwicklungsvorgehen implizieren, ist ein umfassendes Forschungsfeld.

[55] VDA QMC: Qualitäts-Management-Center beim Verband der Automobilindustrie e. V. (www.vda-qmc.de)

3.7.2 Zukünftige Bewertung von Toolketten

In der zukünftigen Bewertung von Werkzeugketten sind folgende Trends absehbar:

1. **Use-Case-basierte Bewertung von Werkzeugen und Werkzeugketten:** Aufgrund der immer weiter zunehmenden Komplexität von Werkzeugen und Werkzeugketten wird es zukünftig nicht mehr oder nur schwer möglich sein, generelle Aussagen über das korrekte Verhalten derer zu treffen. Die Bewertung bzw. die Werkzeugqualifikation wird sich auf eng festgelegte Anwendungsfälle beziehen. Es ist davon auszugehen, dass diese Methodik in alle relevanten Standards zur Funktionalen Sicherheit Einzug halten wird.
2. **Schnittstellen zwischen den Tools**: Bei Werkzeugketten sollte eine ganzheitliche Betrachtung erfolgen. Das bedeutet, dass die Schnittstellen zwischen den einzelnen Werkzeugen eine zunehmend gewichtigere Rolle einnehmen.
3. **Modellbasierte Entwicklung:** Die Verwendung von modellbasierten Entwicklungsmethoden, die ihre volle Stärke durch den Einsatz von automatischer Code-Generierung entfaltet, bedingt eine komplexe integrierte Werkzeugkette.
4. **Agile Methoden:** Ähnlich wie der Einsatz modellbasierter Entwicklung wird auch die agile Entwicklungsmethodik die Toollandschaft weiterentwickeln und neue Werkzeuge werden bei der Entwicklung sicherheitsgerichteter Software Einzug halten.
5. **Verwendung von komplexen Werkzeugen für System- und Sicherheitsanalysen:** Werden Softwarewerkzeuge verwendet, um beispielsweise das Zeitverhalten und die Abhängigkeit bestimmter Softwarepartitionen auf Multicore-Systemen zu modellieren und zu simulieren, ist die Qualität und die Fehlerfreiheit bei diesen sogenannten „Dependability“-Betrachtungen verwendeten Werkzeuge essentiell (siehe (Bärwald, Deubzer, et al., 2010) und (Bärwald, Deubzer, et al., 2012)). Das heißt, dass die verwendeten Simulations- und Optimierungswerkzeuge einer Werkzeugklassifizierung und im Regelfall daran anschließend auch einer Werkzeugqualifizierung unterzogen werden müssen.

6. **Assessmentumgebung vs. Entwicklungsumgebung:** Stand der Technik ist heute, dass Assessments der Funktionalen Sicherheit typischerweise nicht direkt in den Entwicklungsumgebungen der Systementwickler stattfinden. Dieser Bruch ist ineffizient und macht den Prozess besonders bei Systemänderungen langsam und fehleranfällig. Deshalb bietet es sich an, den Assessmentprozess in die Entwicklungs-Toolkette zu integrieren.

3.7.3 Schlussfolgerung aus der Verfügbarkeit von qualifizierten Toolketten

Dem Trend Nr. 6 aus dem vorherigen Kapitel folgend, nach dem es sinnvoll ist, Brüche zwischen den Werkzeugumgebungen bei Entwicklung und Assessment zu vermeiden, sollte ein Konzept entwickelt werden, das es ermöglicht, eben diese Integration herbeizuführen. Ein Ansatz, wie ein solches Konzept umgesetzt werden kann, wird in Kapitel 4 dieser Arbeit detailliert ausgeführt. Bis jetzt ist keine kommerzielle Lösung am Markt verfügbar, die diese Funktionalität vollständig abbildet.

4 Werkzeuggestütztes Assessment sicherheitsgerichteter Systeme

Die bei der Entwicklung von sicherheitsrelevanten Systemen typischerweise verwendete Werkzeugkette deckt nahezu alle Entwicklungsphasen und die daraus resultierenden Arbeitsergebnisse ab (siehe Abbildung 28). Durch eine durchgängige Anwendung der in Kapitel 3 dieser Arbeit beschriebenen Methodik zur Klassifizierung und daraus resultierend der generischen Qualifizierung aller Werkzeuge in einer komplexen Werkzeugkette kann unterstellt werden, dass der sicherheitskritische Einfluss durch Werkzeugfehler auf das Endprodukt hinreichend gering ist. Als Voraussetzung für dieses Kapitel wird also unterstellt, dass alle verwendeten Werkzeuge hinreichend qualifiziert sind und fehlerfrei arbeiten.

Diese Betrachtung beschränkt sich allerdings in der Praxis ausschließlich auf die Produktentwicklung. Das heißt, dass alle entwicklungsrelevanten Daten, wie z.B. Anforderungsspezifikationen, Sicherheitsanalysen, Architektur- und Designdokumente, der Quellcode, aber auch alle Ergebnisse der Verifikations- und Validierungsschritte und die Prozessanforderungen in einer idealerweise homogenen Toolkette gespeichert, verlinkt und konsistent gehalten werden. Beim Assessment, der sicherheitstechnischen Bewertung dieser Systeme, erfolgt allerdings ein Bruch dieser Systematik, da die Dokumente dem Assessor meist in Exportformaten (z.B. PDF) aus den Werkzeugen übergeben werden und die Assessment-Ergebnisse in einer separaten Werkzeuglandschaft dokumentiert werden. Das folgende Kapitel zeigt einen Lösungsansatz, wie es gelingen kann, den Bruch der Werkzeugumgebung zwischen Systementwicklung und Assessment zu vermeiden.

4.1 Bestandsaufnahme und Zielsetzung

Im folgenden Kapitel wird die derzeitig typische Herangehensweise in der Praxis beim Umgang mit Arbeitsergebnissen bzw. Dokumenten, die in komplexen Werkzeugketten verwaltet werden, bei der Begutachtung analysiert. Daraus werden Lösungsansätze abgeleitet, die im weiteren Verlauf der Arbeit in einem konkreten Implementierungsvorschlag ausgearbeitet werden.

4.1.1 Normative Anforderungen

Die in dieser Arbeit betrachteten Sicherheitsstandards (siehe Kapitel 2.4) machen keine Vorgaben, ob und wie Werkzeugketten bei der Begutachtung (Assessment) anzuwenden sind. Das heißt, dass aus der Normenwelt keine Empfehlung abzuleiten ist. Es gibt lediglich Anforderungen an die Art von Reviews, deren Vollständigkeit, die Dokumentation der Reviewergebnisse sowie der Unabhängigkeit der in die Begutachtung involvierten Personen.

4.1.2 Vorgehen einer Zertifizierungsorganisation

Typischerweise sind bei einer Zertifizierung zwei Schritte grundsätzlich zu unterscheiden. Im ersten Schritt erfolgt ein Assessment auf Basis einer etablierten Prüfgrundlage (z.B. nationaler oder internationaler Standard). Die Ergebnisse des Assessments werden in einem Assessment Report (Technischer Bericht) dokumentiert. Dieser Bericht beinhaltet alle technischen Aspekte und Ergebnisse der Begutachtung. Hier gilt, wie im Kapitel 3.4.1.1 ausgeführt, das 4-Augen-Prinzip. Das heißt, ein Bericht hat mindestens einen Verfasser und einen Reviewer.

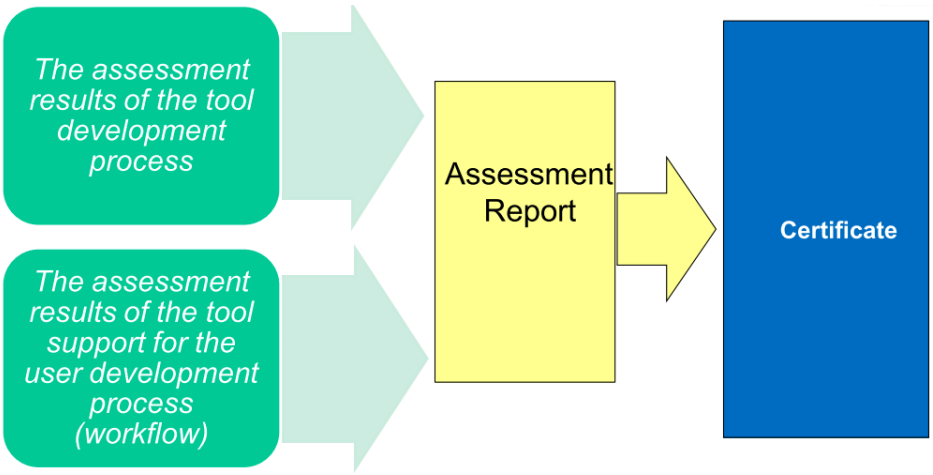

Abbildung 55: Grundsätzlicher Ablauf einer Zertifizierung am Beispiel eines Softwarewerkzeuges

Ist das Ergebnis des Assessments Konformität zur Prüfgrundlage ohne signifikante Abweichungen, kann im zweiten Schritt in die Zertifizierungsprozedur eingestiegen werden. Dabei gelten die Regularien der involvierten Zertifizierstelle (siehe z.B. Prüf- und Zertifizierordnung von TÜV SÜD (TÜV SÜD AG, 2016)). Bei einer Produktzertifizierung bezieht sich das Zertifikat immer auf definierte Versionsstände des Produktes. Änderungen sind der Zertifizierstelle anzuzeigen. Das Zertifikat besteht aus einem

Schmuckblatt und dem Bericht zum Zertifikat, der den Zertifizierungsumfang, die angewandte Prüfgrundlage und die Rahmenbedingungen enthält. Unterzeichnet werden Zertifikat und Zertifikatsbericht durch einen Vertreter der Zertifizierstelle, während der Assessment Report im Prüflabor erstellt wird. Bei TÜV SÜD werden Zertifikate z.B. durch sogenannte Fachzertifizierer freigegeben und unterzeichnet.

4.1.3 Typische Toolkette am Beispiel eines ALM-Werkzeuges

Es ist Stand der Technik, dass in der Softwareentwicklung für sicherheitsgerichtete Systeme sogenannte ALM-Werkzeuge (Application Lifecycle Management) verwendet werden. Ein typischer Vertreter dieser Werkzeugklasse ist PTC Integrity (siehe Kapitel 3.5.2). PTC Integrity implementiert die Funktionalitäten des Anforderungs- und Konfigurationsmanagements. Das heißt, es können alle für die Softwareentwicklung notwendigen Artefakte in der PTC Integrity Plattform abgelegt und mit den modellierten Anforderungen verlinkt werden (siehe Abbildung 56). Auch die Einbindung externer Daten ist über entsprechende Schnittstellen (API) möglich.

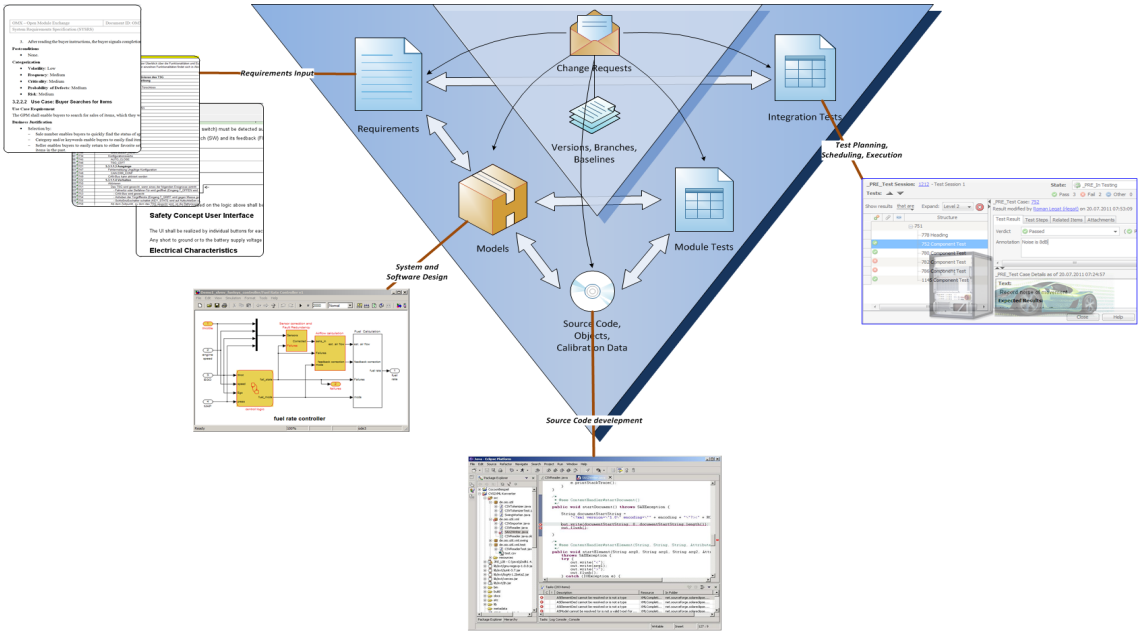

Abbildung 56: Zusammenhang zwischen verwendeter Werkzeugkette und Arbeitsprodukten am Beispiel eines ALM-Werkzeuges (Bärwald et al., 2015a)

Werden alle normativen Anforderungen ebenfalls im Requirements-Engineering-Werkzeug abgebildet, kann zumindest deren Implementierung im Entwicklungsprozess über die entstehenden Entwicklungsdokumente verlinkt werden (Traceability). Diesen Ansatz verfolgt beispielsweise PTC mit der sogenannten PTC Integrity Automotive Solution (PTC, 2015) (siehe Abbildung 57).

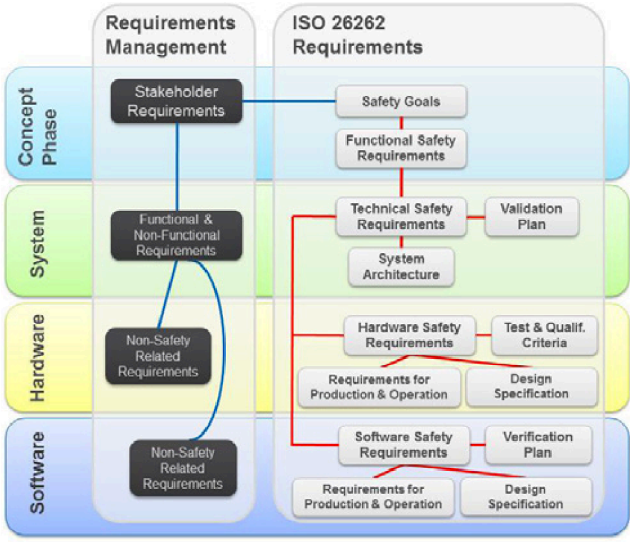

Abbildung 57: ISO 26262 Traceability-Model der PTC Integrity Automotive Solution (PTC, 2015)

Bei dieser Lösung werden die Anforderungen der ISO 26262 in einem Framework zur Installation in der Anwenderumgebung ausgeliefert. Die Automotive Solution stellt die Klauseln der Norm zur Verfügung und liefert Templates der geforderten Dokumente (Work Products) wie z.B. Sicherheitsziele (Safety Goals), Funktionales und Technisches Sicherheitskonzept, Architektur bis zur Verifikation und Validierung. Im Projekt können die entstehenden Dokumente sowohl mit den zugrundeliegenden normativen Anforderungen als auch mit weiteren (externen) Dokumenten verlinkt werden. Eine Liste der in der ISO 26262 geforderten Work Products befindet sich im Anhang A.

Gleichwertige Lösungen sind auf ähnlichen Plattformen wie z.B. IBM DOORS oder SIEMENS Polarion ebenfalls umsetzbar.

Dieser Ansatz ist sicher ein erheblicher Effizienzgewinn für die Entwicklung, adressiert aber die Aufgaben des Assessors nicht.

4.1.4 Status quo und Analyse von existierenden Lösungsansätzen

Die entscheidende Schwachstelle bei den aktuellen Assessment-Ansätzen ist im Grunde ein doppelter Bruch in der Werkzeugkette auf Seiten der Gutachter. Ganz klassisch werden die Ergebnisse bei den Systementwicklern aus der Werkzeugkette in einem Dateiformat wie z.B. PDF exportiert. Dieser Dokumentensatz wird dann vom Gutachter einem Review unterzogen. Dabei entstehen diverse Dokumente wie Checklisten und Reviewberichte. Im nächsten Schritt werden die finalen Dokumente für die Prüfstelle erstellt. Dieser Schritt ist nach dem Stand der Technik im Regelfall nicht in die

Entwicklungswerkzeugkette integriert, was einen weiteren Bruch bedeutet (siehe Kapitel 4.1.4.1). Deshalb wurde bei TÜV SÜD versucht, eine auf Microsoft Access basierte datenbankgestützte Lösung zu implementieren (siehe Kapitel 4.1.4.2).

4.1.4.1 Checklisten und Berichte in gängigen Office-Umgebungen

Der eben beschriebene typische Ablauf der Dokumentation eines Assessments eines sicherheitsgerichteten Systems ist in Abbildung 58 schematisch dargestellt. Von besonderer Bedeutung sind die beiden Brüche in der Werkzeugkette, die eine direkte Verlinkung der Entwicklungsdokumentation auf die Assessment-Berichte explizit nicht ermöglichen.

Die vom Entwickler gelieferte Dokumentation muss in einer Dokumentenliste händisch erfasst werden. Gibt es Änderungen an den Dokumenten, müssen die geänderten Versionsstände ebenfalls händisch erfasst und nachgezogen werden. Eine direkte Verbindung zu den Assessment-Checklisten und -Berichten gibt es nicht.

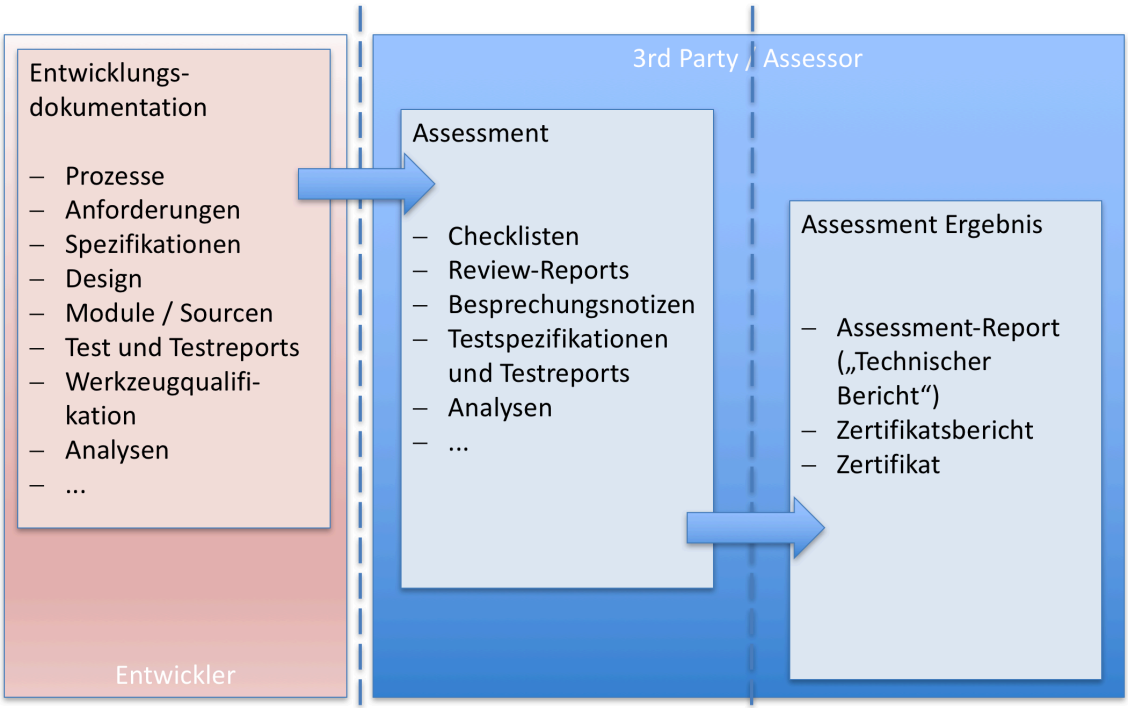

Abbildung 58: Typischer Workflow für Assessments ohne Werkzeugunterstützung

Auf der Assessment-Seite kommen für Checklisten und Berichte typischerweise die Programme der Microsoft Office Suite, also Excel und Word, zur Anwendung. Die Umsetzung mittels Microsoft Office Word ist in Anhang C dargestellt.

4.1.4.2 Datenbankgestützter Assessment-Tool-Prototyp bei TÜV SÜD Automotive

Um die Schwäche der Nichtdurchgängigkeit auf der Assessment-Seite in den Griff zu bekommen, wurde bei TÜV SÜD Automotive eine datenbankgestützte Lösung für die Dokumentation der Assessment-Ergebnisse und eine daran anschließende Generierung von Assessment-Berichten implementiert. Wesentliche Teile der Berichte können durch Textbausteine, die den datenbankgestützten Checklisten zugeordnet sind, automatisch generiert werden. Über entsprechende Kommentarfelder in den Checklisten können zusätzliche Texte für die Abschlussberichte hinterlegt werden. Dieser Ansatz führt zu dem in Abbildung 59 dargestellten Workflow, bei dem zumindest der Bruch auf der Assessment-Seite eliminiert wird.

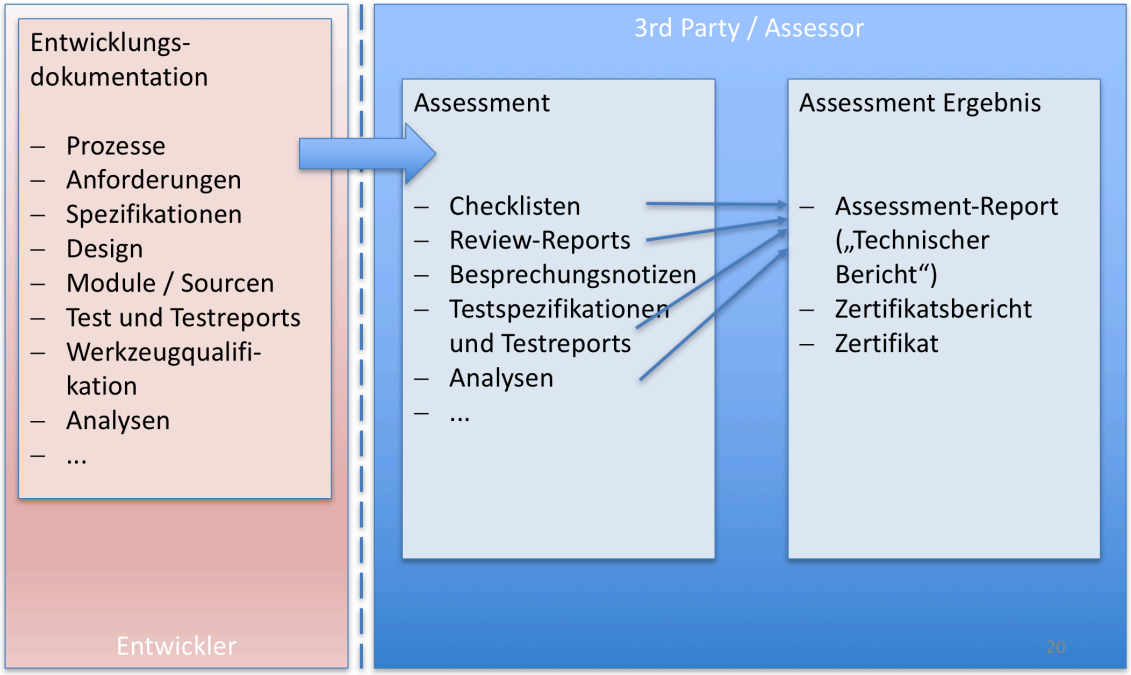

Abbildung 59: Workflow bei der Verwendung einer datenbankgestützten Lösung

Die Lösung wurde basierend auf Microsoft Access umgesetzt. Das in Abbildung 60 dargestellte ERM in Martin-Notation[56] (siehe (Balzert, 2009) Seite 202) zeigt das Datenmodell, das der Assessment-Datenbank zugrunde liegt.

[56] Die sogenannte Martin-Notation ist in der Literatur auch unter der Beziehung „Krähenfuss-Notation“ (Crow’s Foot Notation) zu finden.

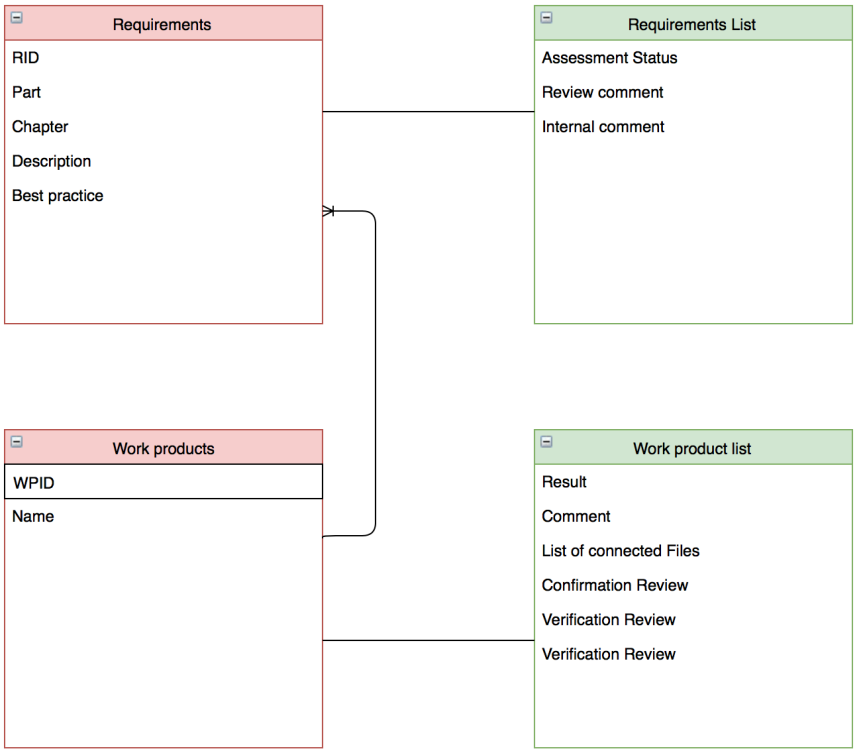

Abbildung 60: ERM Datenmodell der Assessment-Datenbank in Martin-Notation

Dabei sind die Inhalte auf der linken Seite des Diagramms vordefiniert. Sie ergeben sich aus der Norm und bilden somit den Standard ISO 26262 ab. In der Tabelle „Requirements“ sind alle Klauseln der Norm erfasst. Darüber hinaus gibt es zu jedem Requirement eine Hilfestellung für den Assessor. Im Attribut „Best Practice“ werden Stichworte typischer Inhalte vorgehalten, an denen sich der Assessor orientieren kann. In der Tabelle „Work Products“ sind alle in der Norm geforderten Dokumente (Work Products) gelistet. Jedem „Work Product“ sind mehrere „Requirements“ zugeordnet (1:n Beziehung).

In der „Work Product List“ werden die einzelnen, projektspezifischen Work Products erfasst und bewertet. Es können mehrere Dateien zugeordnet werden. In der Tabelle „Requirements List“ wird jedem Eintrag in der Tabelle „Requirements“ eine projektspezifische Instanz entgegengestellt. Auf dieser Basis wird die Erfüllung jeder Klausel der Norm bewertet. Aus dem Assessment-Status („o.k.“, „nicht o.k.“, „o.k. mit Anpassungen“ und „nicht anwendbar“) und Freitextkommentaren kann ein Assessment-Report weitgehend als Microsoft Word Datei generiert werden. Screenshots der implementierten und lauffähigen Lösung sind im Anhang D zu finden.

Aufgrund der Verwendung von Microsoft Access gibt es nur eine eingeschränkte Multi-User-Fähigkeit, was für den Roll-out einer großen Sachverständigenorganisation und die Verwendung in großen Projekten mit mittlerer bis langer Projektlaufzeit eine erhebliche Schwachstelle darstellt.

Insgesamt kann beim Einsatz in der Praxis festgestellt werden, dass der Aufwand reduziert und das Assessment effizienter wird. Allerdings wird der Bruch zur Werkzeugumgebung auf der Entwicklerseite mit diesem Ansatz nicht überwunden.

4.1.4.3 Schwächen der etablierten Methoden

Wesentliche Schwachstellen sind die Brüche in der Werkzeugkette, die es bei beiden vorgestellten Lösungen nicht ermöglichen, direkte Verbindung (Traces) zu den Entwicklungsdokumenten herzustellen. Dieser Umstand reduziert in erheblichem Maß die Effizienz bei der Begutachtung. Besonders signifikant ist dieser Umstand, wenn im Rahmen einer Produktänderung ein Re-Assessment notwendig ist. Hier können mit einer geeigneten Lösung erheblich Aufwände und somit Kosten reduziert werden.

4.1.4.4 Lösungsansätze

Basierend auf der Analyse der aktuell verfügbaren Methoden und deren Schwachstellen lassen sich folgende Anforderungen für eine werkzeugunterstützte Komplettlösung formulieren:

1. Die Brüche in der Werkzeugkette müssen beseitigt werden. Entwicklung und Assessment müssen in der gleichen Werkzeugumgebung stattfinden.
2. Zur Effizienzsteigerung muss eine Automatisierung einzelner Schritte möglich sein, z.B. bei der Generierung von Berichten.
3. Die Lösung muss sich grundsätzlich auf verschiedene Sicherheitsstandards anwenden lassen.
4. Speziell für entwicklungsbegleitende Begutachtungen[57] ist es essentiell, dass die Werkzeuge für eine Multi-User-Fähigkeit ausgelegt sind.

[57] Die Erfahrung zeigt, dass bei ca. 95% aller sicherheitsrelevanten Projekte, deren Assessment von einer unabhängigen Prüforganisation durchgeführt wird, die Assessment-Aktivitäten entwicklungsbegleitend sind.

Damit ist eine in ein ALM-Werkzeug eingebettete Lösung naheliegend, ein sogenanntes Assessment-Modell. Dieser Ansatz wird in Kapitel 4.3 hergeleitet und exemplarisch entwickelt.

Zuvor wird im folgenden Kapitel die Fragestellung untersucht, inwieweit es möglich ist, Sicherheitssysteme generisch und unabhängig von der Zieldomäne zu entwickeln und diese Ergebnisse formalisiert abzulegen. Dieser Aspekt wurde u.a. im Rahmen des Forschungsvorhabens SPES 2020 untersucht, dessen Ergebnis das Konzept eines sogenannten Safety-Plug-In ist.

4.2 Der Safety-Plug-In im SPES 2020 Forschungsprojekt[58]

Im Rahmen des Förderprojektes SPES 2020 (TU München et al., 2012) wurde im Teilprojekt ZP-AP4 untersucht, inwieweit es Überschneidungen zwischen den Anforderungen verschiedener Sicherheitsstandards gibt, um daraus eine effiziente Vorgehensweise zur Bewertung, Freigabe und Zertifizierung sicherheitsgerichteter Systeme abzuleiten.

Aus dem Teilprojekt heraus entstanden eine Analyse der Anforderungen und ein theoretischer Designvorschlag. Die Ergebnisse werden im folgenden Kapitel ausgeführt. Auf Basis der Erfahrungen aus dem SPES 2020 Projekt wurde ein Ansatz zum werkzeugbasierten Assessment hergeleitet. Auf dieser Grundlage wurde anschließend ein Demonstrator exemplarisch implementiert (siehe im Kapitel 4.3).

Ziel im SPES 2020 Projekt war, die normativen Anforderungen in einem generischen Safety-Datenmodell abzubilden – dem „Safety-Plugin“ (siehe Abbildung 61). Damit können auf Projektebene als Zertifizierungs-Suite die für die Zertifizierung relevanten Daten extrahiert und dem Assessor bzw. Zertifizierer in derselben Werkzeugumgebung zur Verfügung gestellt werden. Die Erstellung der geforderten Prüfdokumentation (Assessment-Bericht) sollte weitgehend automatisiert ablaufen.

[58] Dieses Kapitel fasst die Arbeiten des Autors im Forschungsprojekt SPES 2020 zusammen (Bärwald, Inderst, Rosen, Stelzig, et al., 2011) (Bärwald, Inderst, Rosen, Smit-Wiesner, et al., 2011) (Bärwald, Inderst, et al., 2012).

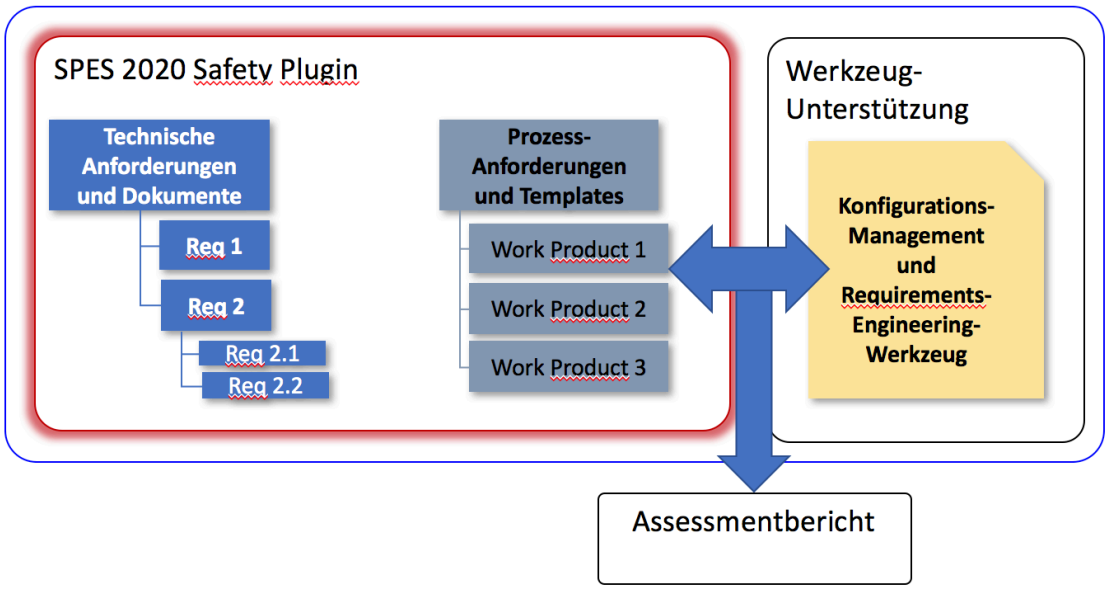

Abbildung 61: Struktur des Safety-Plugin im SPES 2020 Forschungsprojekt

Basierend auf dieser Systematik können Ableitungen für verschiedene Sicherheitsstandards implementiert werden. Dabei wird das generische Datenmodell durch die spezifischen Anforderungen eines Sicherheitsstandards überladen.

Im ersten Schritt erfolgt eine Analyse der grundsätzlichen Anforderungen an eine Entwicklungsdokumentation am Beispiel der ISO 262626.

4.2.1 Geforderte Dokumente

Die ISO 26262 fordert für die software-relevanten Teile 43 technische Dokumente, welche im Anhang A aufgelistet sind. Neben den 16 spezifischen Dokumenten, die für Support- und Querschnittsaktivitäten, wie Konfigurations-Management, Qualitätssicherungs- und Projektplanung aus Part 6 und 8 des Standards gefordert werden, gibt es 27 projektspezifische Dokumente bei der höchsten Sicherheitskategorie (ASIL D).

Zusätzlich könnten weitere Dokumente für die einzelnen Life-Cycle-Phasen generiert werden. Diese können wie folgt gegliedert werden:

1. das eigentliche Phasen-Dokument (z.B. System-Design-Dokument)
2. das dazugehörige Guideline-Dokument
3. das phasenbezogene Safety-Analyse-Dokument
4. die phasenbezogene Checkliste zur Verifikation

Darüber hinaus fordern alle Sicherheitsstandards klar definierte Entwicklungsprozesse. Das schließt eine strukturierte Dokumentation ein. Deshalb muss es für alle Dokumente geeignete Vorlagen (Templates) geben. Daraus ergeben sich am Beispiel der ISO 26262

bei vollständiger Umsetzung dieses Ansatzes jeweils 7 zusätzliche Dokumente in den 10 Phasen des Softwarelebenszyklus, also insgesamt 70 weitere prozessbezogene Dokumente.

Generell erfordern Sicherheitsnormen einen erheblichen Dokumentationsaufwand mit komplexen Strukturen und Abhängigkeiten zwischen den einzelnen Dokumenten. Die wesentliche Herausforderung ist, die Widerspruchsfreiheit, Redundanzfreiheit und die Vollständigkeit des Dokumentensatzes fortlaufend sicherzustellen.

4.2.2 Ableitungen für Entwicklungsdokumentation am Beispiel der ISO 26262

Die Problemstellungen, die sich aus dieser Vielzahl von Dokumenten für die Dokumentationserstellung und damit auch für deren Bewertung (Assessment) ergeben, wird im folgenden Kapitel analysiert.

4.2.2.1 Vollständigkeit

Wenn es darum geht, einen vollständigen Satz von Dokumenten nach ISO 26262 für den gesamten Life Cycle inklusive der Querschnittsaktivitäten zu erstellen, dann ist eine systematische Liste (siehe Anhang A) sehr hilfreich, die alle im Projekt geplanten Dokumente umfasst. Diese Liste an Dokumenten stellt den Safety-Case dar.

Es sind aber außer der formalen Vollständigkeit der Dokumente, also deren Existenz, auch deren inhaltliche Vollständigkeit und technische Eignung gefordert. Um sich in dieser Hinsicht abzusichern, sollte für alle Dokumente des Lebenszyklus projektunabhängig ein generisches Template erzeugen werden, das projektspezifisch nach einer entsprechenden Tailoring-Vorschrift angepasst und bearbeitet werden kann. Die Vollständigkeit und Widerspruchsfreiheit bleibt dann noch ein subjektives Kriterium für das Assessment, solange die Inhalte nicht wenigstens semiformal spezifiziert sind.

4.2.2.2 Widerspruchsfreiheit

Ist ein Dokument unvollständig oder überdefiniert, z.B. in der Form, dass die Anforderungen und deren Ableitungen in unterschiedlichen Kapiteln mit einem unterschiedlichen Detaillierungsgrad definiert wurden, kann es in den untereinander abhängigen, verschiedenen Dokumenten zu einer widersprüchlichen Beschreibung der Anforderungen oder deren Auslegung kommen.

Bei der Anwendung eines Entwicklungsprozesses, der dem Grundgedanken des V-Modells (siehe Kapitel 2.4.2.1) folgt, ist jedes Dokument im absteigenden, linken Ast

(Softwareentwicklung) und im aufsteigenden, rechten Ast (Verifikation und Validierung - V&V) voneinander abhängig, entweder als vertikale (Top-Down-) Traceability (Refinement) oder als horizontale Traceability (Verification).

Wenn es kein formales Schema dieser Zusammenhänge gibt und die entsprechende Werkzeugunterstützung fehlt, können widersprüchliche, überlappende oder unvollständige Anforderungen, Design- und Implementierungsentscheidungen und Tests definiert werden, deren Entdeckung und Korrektur nur sehr schwer bis zufällig erfolgt (Turban, Kucera, Tsakpinis, & Wolff, 2009). Dies hat auch Einfluss auf das Assessment, da die Vollständigkeitsprüfung und Prüfung auf Widerspruchsfreiheit ggf. nicht mehr systematisch durchgeführt werden kann.

4.2.2.3 Redundanzfreiheit

Die in den vorherigen Kapiteln analysierten Defizite bzgl. Vollständigkeit und Widerspruchsfreiheit sind im Wesentlichen der fehlenden semiformalen Spezifikation der Inhalte der von der ISO 26262 geforderten Dokumente geschuldet, was unweigerlich zu der Gefahr von redundanten Informationen führt. Das bedeutet nicht zwingend, dass z.B. Anforderungen inhaltlich falsch sein müssen, aber durchaus mehrfach oder überdefiniert sein können, auch über die weiteren Entwicklungsphasen hinweg.

Das kann zu erheblichen Problemen bei Änderungen führen, da ggf. an verschiedenen Stellen im Dokumentationspfad geändert werden muss, was zu unübersichtlichen Änderungsprozeduren führt und letztlich sehr fehleranfällig ist.

An die oben diskutierten Fragestellungen schließt sich unmittelbar das Problem der Wartbarkeit bzw. Änderungsfreundlichkeit an.

4.2.2.4 Wartbarkeit von Dokumenten

Alle in den vorangegangenen Kapiteln betrachteten Aspekte führen direkt oder indirekt zu Implikationen für die Wartbarkeit von Dokumenten.

In der Vergangenheit gab es in Industrieprojekten häufig das Problem, der nicht eindeutigen Identifizierung von Anforderungen auf allen Entwurfsebenen. Werden beispielsweise zur Indizierung die Kapitelnummern der Gliederung eines Dokuments verwendet, besteht z.B. beim Einfügen neuer Inhalte die Gefahr, dass vorhandene Platzhalter nicht mehr ausreichen und dann Verweise und Bezüge nicht mehr korrekt sind oder

aber sukzessive geändert werden müssen mit der Konsequenz, dass alle Verweise in andere Dokumente aktualisiert werden müssen.
Wird dieser Prozess manuell umgesetzt, besteht darüber hinaus das akute Risiko, dass jeder Schreibfehler beim Einfügen neuer Kapitelnummern schwer feststellbar ist und somit auch zu fehlerhaften Dokumenten führt.

4.2.2.5 Auswirkungen für Assessments

Alle diskutierten Aspekte bzgl. Vollständigkeit, Widerspruchsfreiheit, Redundanzfreiheit und Wartbarkeit von Dokumenten können zu stark erhöhten Aufwänden und schlimmstenfalls zu Inkonsistenzen beim Assessment führen. Es skalieren sich hier die aufgezeigten Probleme zumindest linear auf den zu erwartenden Aufwand und damit auf die Dauer und die Kosten des Assessments.

4.2.2.6 Zusammenfassung und Schlussfolgerungen für Verbesserungen

Zusammengefasst kann festgestellt werden, dass die Erstellung der von der ISO 26262 geforderten Dokumente aufwendig ist und auch nicht zwingend die notwendige Präzision besitzt, um eine eindeutige Assessment-Aussage bzgl. der angewendeten Prozesse und der erstellten Dokumente für die geforderte Sicherheitskategorie (ASIL) zu formulieren. Die Inhalte und die Struktur der von der ISO 26262 geforderten Dokumente sind nicht zwingend formal (bzw. semiformal) spezifiziert, so dass eine systematische Lösung benötigt wird.
Verschärft wird diese identifizierte Schwachstelle der mangelnden Formalisierung von Dokumenten und Dokumentationsvorgaben durch eine Vielzahl von manuellen Bearbeitungsschritten in der Praxis. Abhilfe kann durch die geeignete Verwendung eines Softwarewerkzeuges für das Konfigurations- sowie das Requirement-Management, wie z.B. IBM DOORS, PTC Integrity oder Siemens Polarion, geschaffen werden. Während der Forschungstätigkeit für diese Arbeit ist die Verwendung dieser Werkzeuge global zum Industriestandard geworden. Die in den vorherigen Kapiteln beschriebenen Problemstellungen lassen sich nur durch den konsequenten Einsatz solcher Werkzeuge beherrschen. Deshalb wird die Verwendung solcher Toolketten für den Lösungsansatz im Folgenden als gegeben vorausgesetzt. Basierend auf dieser Hypothese wird in den nachfolgenden Kapiteln ein Vorschlag zur effizienten Nutzung bei Assessments erarbeitet.

4.2.3 Darstellung der Lösung - SPES 2020 Safety-Plug-In

Die effiziente Erstellung von Zertifizierungsunterlagen und der daraus resultierende geringere Aufwand im Assessment ist das Ziel einer sogenannten Zertifizierungs-Suite, die sowohl die Entwickler als auch den Assessor eines sicherheitskritischen Produkts unterstützen soll. Der zentrale Aspekt sind dabei die sogenannten Safety-Soll-Daten, die Daten und Metadaten, die von einem Sicherheitsstandard gefordert werden.

Die für ein Assessment relevanten Daten können abstrakt in den folgenden 3 Datenstrukturen[59] abgebildet werden:

1. Daten der phasenbezogenen „Requirements Traceability List and Test Coverage Matrix",
2. Daten der phasenbezogenen Safety Analysen in Form von „Hazard Logs" und
3. Daten der phasenbezogenen Verifikationsaktivitäten und deren Ergebnisse.

4.2.3.1 Festlegung der gemeinsamen Datenbasis vom Hersteller und Nutzer einer Zertifizierungs-Suite

Die Datenbasis der Zertifizierungs-Suite besteht aus einem Satz von ADT (Abstact Data Types)[60] zur Realisierung der Safety-Soll-Daten und den entsprechenden Operationen. Zentraler Bestandteil der Zertifizierungs-Suite ist ein „Safety Server", der als Vermittler bzw. Übersetzer zwischen der IT-Infrastrukur auf der Entwicklerseite und der Assessor-Seite fungiert (siehe Abbildung 62).

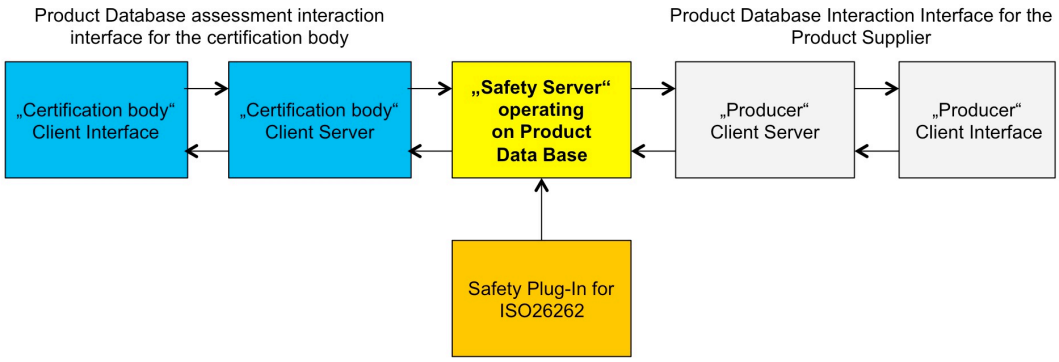

Abbildung 62: Realisierungsvorschlag für eine Zertifizierungs-Suite aus (Bärwald, Inderst, et al., 2012) **Seite 19**

Ziel ist, die Vollständigkeit und die Korrektheit der Entwicklungsdokumentation basierend auf obige ADT zu prüfen und zu beurteilen. Der Assessor muss auf alle relevanten

[59] Querschnittsaktivitäten, wie CM, QA, Projektplanung sind bei dieser Betrachtung ausgeschlossen und sind in der herkömmlichen Art und Weise als Dokumente entsprechend den Standardvorgaben zu erstellen.

[60] Siehe (Puntambekar, 2008) Seite 2-2 und (Sane & Deshpande, 2006) Seite 1-5ff und 2-3f

Daten zugreifen können, um die Korrektheit und Vollständigkeit der selbigen zu überprüfen.

4.2.3.2 Datenstrukturen

Die Datenstrukturen für die Safety-Soll-Daten sind im Safety-Plug-In vorgegeben und bestehen aus:

Datenstrukturen zu den Traceability-Listen für die Development- und Test-Phasen

Um die Durchgängigkeit von Anforderungen bis in die Implementierung und den Test gewährleisten zu können, muss der Zusammenhang zwischen Anforderungen auf verschiedenen Designebenen und der zugehörigen Testabdeckung modelliert werden. Mit unterschiedlichen Designebenen ist in diesem Zusammenhang gemeint, dass auch das Architektur-, Grob- und Feindesign in der Praxis häufig als weitere Detailierungsebenen abgebildet werden. Zu deren Dokumentation wird im Regelfall ebenfalls das Requirements-Engineering-Werkzeug verwendet.

Die Abbildung 63 illustriert in EXPRESS-G-Notation[61] den Zusammenhang, dass es zwischen den Designebenen eine Verlinkung geben muss. Gleichzeitig müssen für alle Einträge Traces zu den Verifikationsmaßnahmen, also der Testspezifikation (Test Procedure), den Testdokumentation (Test Report) und zum Testergebnis (Test Result) existieren.

[61] EXPRESS-G ist die graphische Darstellung der EXPRESS-Notation zur Datenmodellierung siehe (Schenck & Wilson, 1994) Kapitel 2.4 Seite 17ff

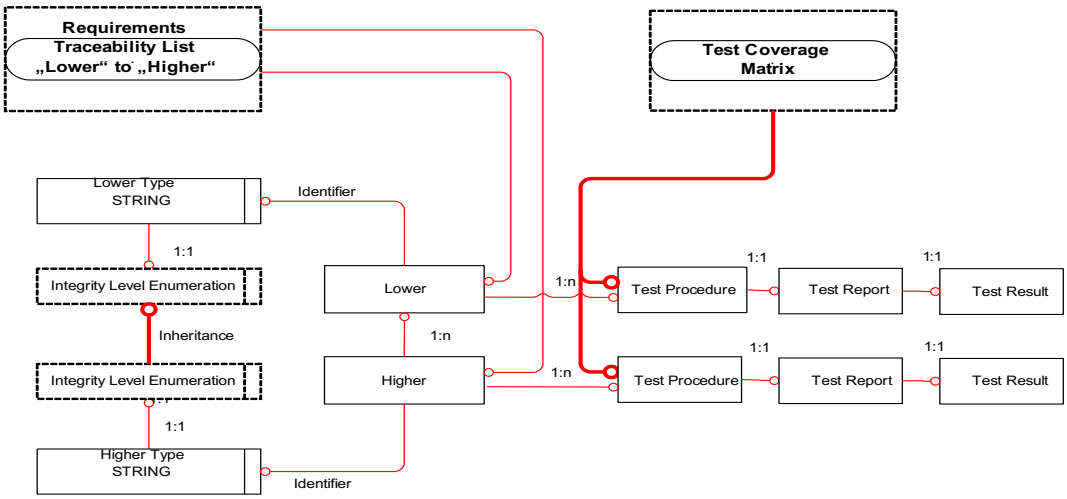

Abbildung 63: Zusammenhang zwischen verschiedenen Anforderungsebenen und der Testabdeckung aus (Bärwald, Inderst, et al., 2012) **Seite 24**

Datenstrukturen für die Ergebnisse der Safety Analysen (sogenannter Hazard Log)

Im Hazard Log werden die auf der Systemebene aus den Gefährdungen und Risiken analysierten und klassifizierten Sicherheitsziele (z.B. Safety Goal mit einem definierten ASIL nach ISO 26262) auf die Implementierungsebene heruntergebrochen (siehe Abbildung 64).

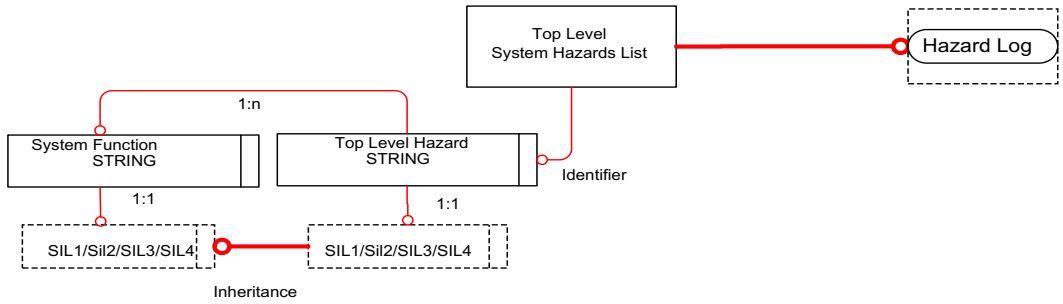

Abbildung 64: Zusammenhang zwischen Safety Goal und Implementierung im Hazard Log aus (Bärwald, Inderst, et al., 2012) **Seite 29**

Es gilt also sicherzustellen, dass jede Funktion mit der geforderten Sicherheitsintegrität implementiert wird. Hierbei können Sicherheitsziele auch auf mehrere Funktionen heruntergebrochen werden (ASIL Dekomposition). Auf der Dokumentenebene muss dieser Zusammenhang vollständig und nachvollziehbar dargestellt werden. Das generische Entity „Hazard Log“ in Abbildung 64 ist als Container für die Ergebnisse aus der Top Level System Hazard Analysis konzipiert, der die abgeleiteten Sicherheitsziele und Sicherheitsfunktionen zugeordnet werden können.

Eine allgemeingültige (generische) Struktur (siehe Abbildung 65) soll sich demnach anderer generischer Komponenten bedienen (u.a. der Traceability-Information von Requirements aus der generischen Komponente „Requirements Traceability List and Test Coverage Matrix“).

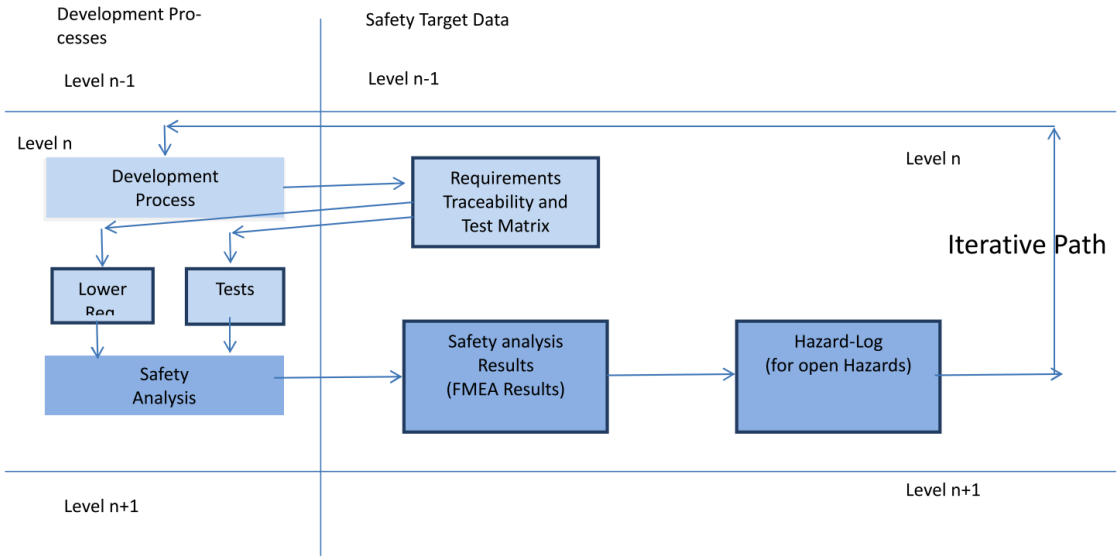

Abbildung 65: Zusammenhang von Entwicklungsdokumentation und Safety-Soll-Daten einer bestimmten Designebene aus (Bärwald, Inderst, et al., 2012) **Seite 32**

Die Darstellung in Abbildung 65 bezieht sich auf eine Hierarchieebene und muss für alle n Ebenen instanziiert werden.

Datenstruktur für die Verifikations-Listen

In Abbildung 66 ist der Aufbau der Datenstruktur für die Verifikationsschritte dargestellt. Es handelt sich um die generische Definition einer typischen Checkliste.

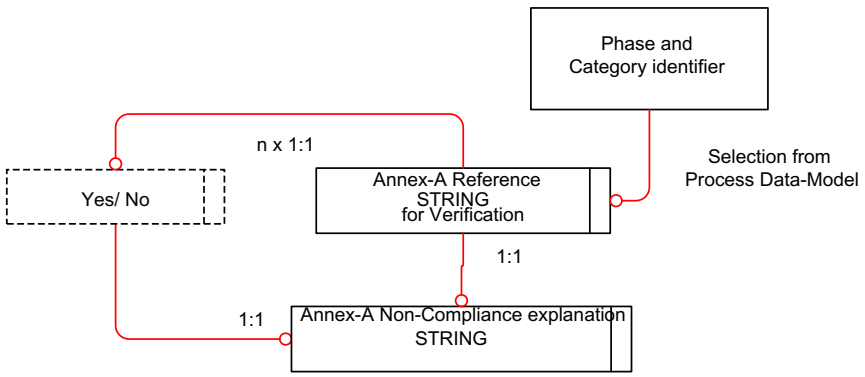

Abbildung 66: Datenstruktur für die Bewertung in einem Verifikationsschritt aus (Bärwald, Inderst, et al., 2012) **Seite 25**

Jeder normativen Anforderung (Phase) ist eine textuelle Beschreibung zugeordnet. Es gibt eine Bewertung (im Beispiel „Yes/No“), was in einer Praxislösung auf zusätzliche Attribute wie „nicht anwendbar“ und „teilweise ok“ erweitert werden muss. Darüber hinaus gibt es ein Kommentarfeld (Explanation) für den Assessor.

Für jede Instanz eines jeden spezifischen Standards muss für die Zertifizierungs-Suite eine Abbildung der Anforderungen der Norm hinterlegt werden. Dies geschieht typischerweise in Form einer Checkliste, die alle Klauseln der Norm abbildet.

4.2.3.3 Realisierungsvorschlag

Der im Folgenden vorgestellte Realisierungsvorschlag für eine Zertifizierungs-Suite basiert auf der Funktionalität eines Produkt-Datenmanagement-Systems (PDM)[62]. Ein PDM im eigentlichen Sinne umfasst das Datenhandling eines kompletten Produktlebenszyklus inklusive aller Entwicklungsdokumente, Produktionsdaten und Felddaten. Das Requirements-Management und das Konfigurations-Management sind Teilfunktionalitäten eines PDM (siehe Abbildung 67).

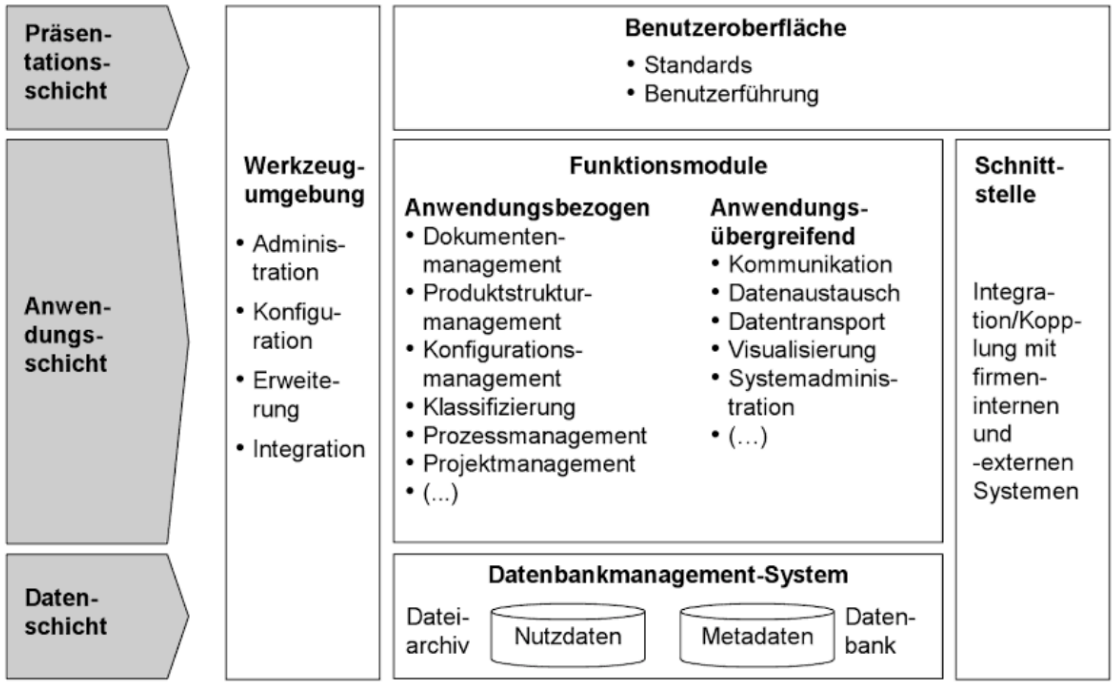

Abbildung 67: Aufbau eines PDM Systems aus (Krause, 2008) **Seite 16**

[62] „Das Produktdatenmanagement (PDM) ist das entscheidende informationstechnische Werkzeug... Die Hauptziele von Produkt-Datenmanagement-Systemen liegen in der Verwaltung sämtlicher Produktdaten über alle Phasen des Produktlebenszyklus...“ aus (Schilke, 2010) Seite 24

Grundidee bei der Verwendung eines PDM ist, dass entsprechende Werkzeuge bereits eine Großzahl von Strukturierungsfunktionen für Daten und entsprechende Zugriffsoperationen inklusive einer angemessenen Benutzeroberfläche (UI) zur Verfügung stellen und die Realisierung der Safety-Soll-Daten-Strukturen und der Nutzeroperationen als eine Erweiterung der bereits vorhandenen Strukturen und Operationen darstellt werden kann.

So ist im Nachfolgenden die Struktur einer Zertifizierungs-Suite zunächst in Form eines Schichtenmodells dargestellt (siehe Abbildung 68), wobei die unterste Schicht die Funktionalität des benutzten Datenbank-Servers beinhaltet, welcher mit den Safety-Soll-Daten des Safety-Plug-In instanziiert werden muss. Im folgenden Beispiel werden die Anforderungen des Automotive-Standards ISO 26262 abgebildet.

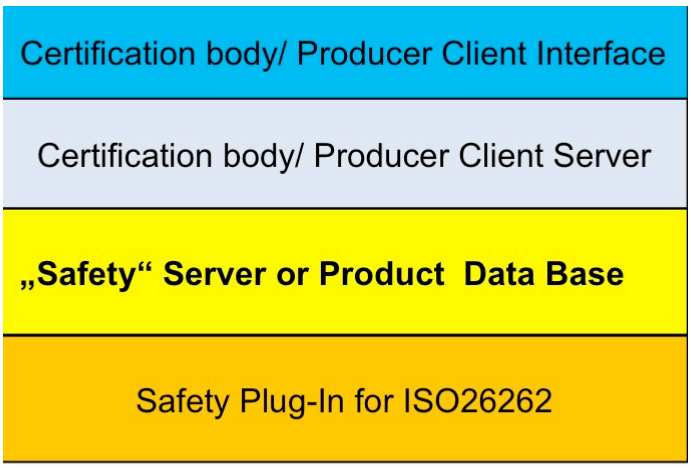

Abbildung 68: Vorgeschlagenes Schichtenmodell der Zertifizierungs-Suite basierend auf der Architektur aus Abbildung 62 aus (Bärwald, Inderst, et al., 2012) **Seite 18**

Der Safety-Server ist das zentrale Element in dieser Architektur. Er beinhaltet entweder die Datenbasis selbst oder implementiert eine Schnittstelle zur Datenbank, in der die Entwicklungsdaten gehalten werden. Daran gekoppelt ist die Zertifizierungsschnittstelle, die als Client-Server-Lösung dargestellt ist. Der Zertifizierungs-Server kann die entsprechenden Projektdaten zur Verfügung stellen, die vom Assessor im Zertifizierungs-Client benötigt werden.

Dieser Ansatz trennt die Entwicklungsdaten von den Assessment-Dokumenten. Vorteil ist es, dass nicht der komplette Datenbestand durch den Assessor einsehbar ist, sondern nur die für die Zertifizierung relevanten Daten. Dies hat speziell im Automotive-Umfeld

seine Stärken, wo strenge Auflagen bezüglich Geheimhaltung und IP[63]-Schutz bestehen. Nachteil dieser Lösung ist, dass ein zusätzliches Tool in die Werkzeugkette integriert werden muss, welches die Funktionalität des Datentreuhänders (Data Handler) implementiert.

Das Datenmodell der in Kapitel 4.2.3.1 eingeführten Safety-Soll-Daten des Safety-Plug-Ins muss für jede Architekturebene die definierten Datenstrukturen abbilden (siehe Abbildung 69).

[63] IP = intellectual property, geistiges Eigentum

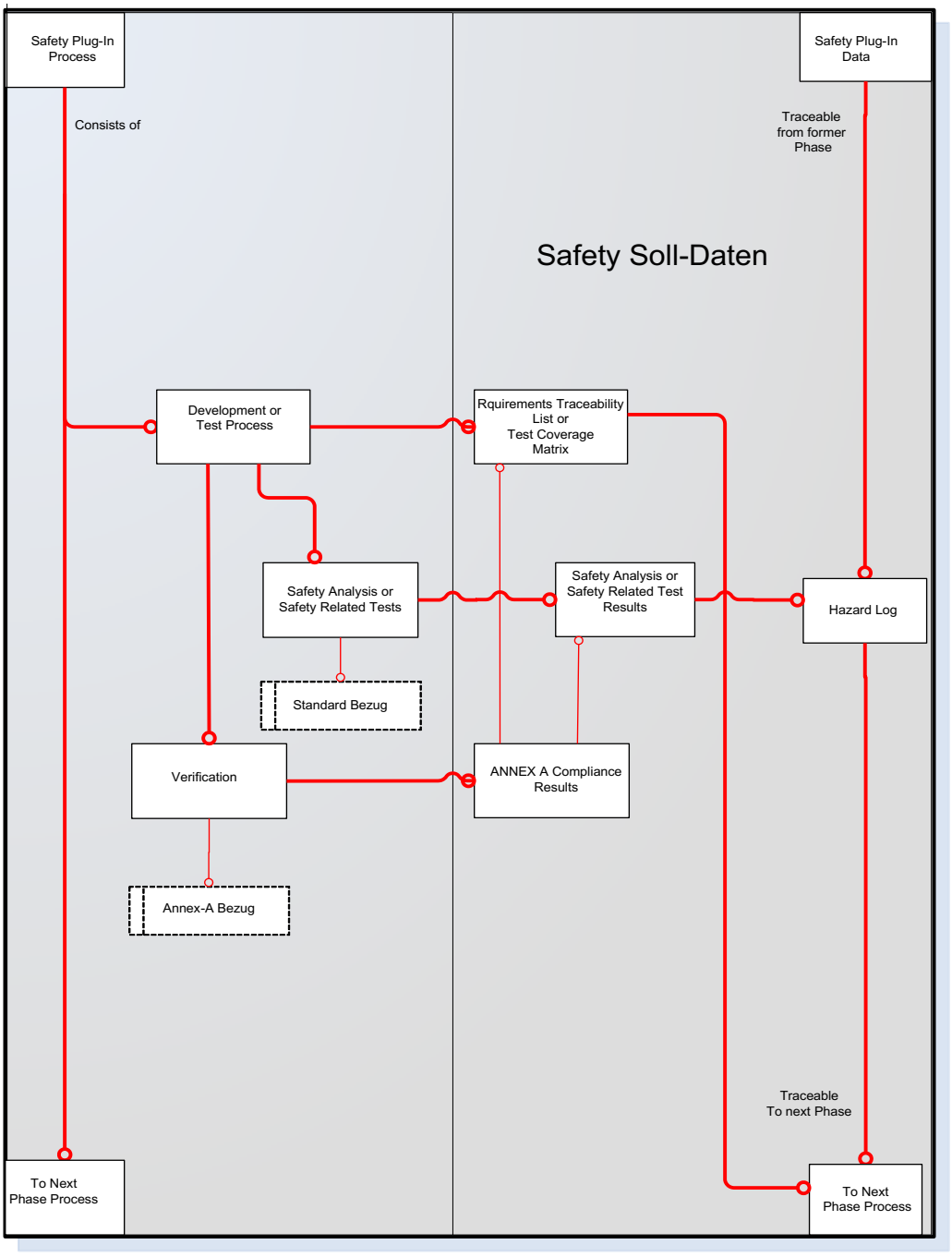

Abbildung 69: Generisches Datenmodell für eine Phase bzw. Design-Ebene im V-Modell aus (Bärwald, Inderst, et al., 2012) **Seite 22**

Das Datenmodell beinhaltet Prozess- und Entwicklungsdokumente, Verifikationsergebnisse, Safety-Analysen und deren aller Traceability. Die Darstellung der Abbildung 69 entspricht einer Architektur-Hierarchie-Ebene (z.B. Anforderungen, Architektur, Grob- oder Feindesign). Das muss nun in den gesamten Entwicklungsprozess eingebunden und kaskadiert werden. Am Beispiel des V-Modells, wie es im Kapitel 2.4.2.1 in seiner als Spezifikation des V-Modells XT dargestellt wurde und wie es für die Abwicklung öffentlicher Aufträge von der Bundesstelle für Informationstechnik verfügbar ist, wird

in Abbildung 70 diese Einbindung und Kaskadierung in den Entwicklungsprozess verdeutlicht.

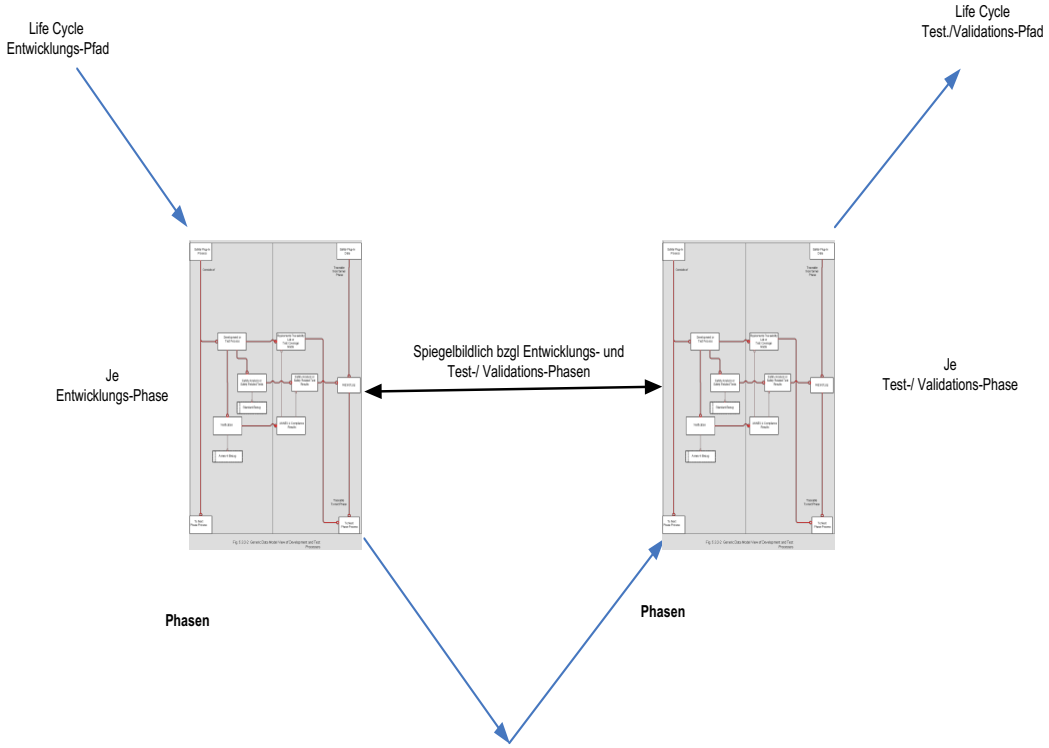

Abbildung 70: Instanziierung des generischen Metadatenmodells in den Safety Life Cycle aus (Bärwald, Inderst, et al., 2012) **Seite 23**

4.2.4 Zusammenfassung der Ergebnisse von SPES 2020 in Bezug auf diese Arbeit

Im Forschungsprojekt SPES 2020 wurde gezeigt, dass es basierend auf einer etablierten Werkzeugkette grundsätzlich möglich ist, Anforderungen an sicherheitsgerichtete Systeme zu formulieren und deren Umsetzung werkzeugunterstützt nachzuweisen und zu dokumentieren.

Aufgrund der hohen Komplexität und der weitgehend theoretischen Betrachtungen im Forschungsprojekt wurde im Rahmen dieser Arbeit beschlossen, weder den SPES 2020 Ansatz für einen Demonstrator zu verwenden, noch das Vorgehen auf verschiedene andere Sicherheitsnormen zu erweitern.

Stattdessen wird auf Basis einer bestehenden Werkzeugkette durch Erweiterungen deren Funktionalität eine exemplarische Implementierung für einen ausgewählten Sicherheitsstandard umgesetzt. Sicher wäre es gerade bei generischen Softwareprodukten hilfreich, die domänenübergreifend in elektronischen Systemen zum Einsatz kommen, wie z.B. embedded Betriebssystemen (OS), die Anforderungen für verschiedene Standards

parallel zu erfassen, zu bearbeiten und danach in ein allgemeingültiges Framework zu überführen. Dies ist eine Themenstellung für weitergehende Forschung.

4.3 Das Assessment-Modell

Nachdem im vorherigen Kapitel die theoretischen Grundlagen eines werkzeugunterstützten Assessments beschrieben wurden, wird im Folgenden eine exemplarische Implementierung eines Assessment-Modells am Beispiel der ISO 26262 hergeleitet. Für dieses Beispiel werden die Anforderungen des Bandes 3 der Norm (Concept Phase) umgesetzt, da dieser Band aufgrund seines kompakten Umfanges und seiner geringeren Komplexität besonders gut für einen Demonstrator geeignet ist.

Das in Kapitel 4.2.3 hergeleitete Vorgehen für ein werkzeugunterstütztes Assessment basiert auf der Verwendung eines PDM-Werkzeuges, also einer Werkzeugumgebung, die die kompletten Entwicklungs- und Produktionsdaten verwaltet. Die im Kapitel 4.2 im Detail beschriebene Zertifizierungs-Suite erfasst, verwaltet und speichert alle für das Assessment relevanten Daten. In Abhängigkeit von der normativen Grundlage wird der Dokumentensatz (Safety Case) für die Sicherheitsfreigabe weitgehend automatisiert erstellt. Dabei muss das Werkzeug auf alle relevanten Entwicklungsdokumente Zugriff haben.

Um einen pragmatischen Ansatz zu wählen, ist es naheliegend, auf eine bereits existierende Plattform aufzusetzen. Die geforderte Funktionalität eines Assessment-Werkzeuges wird deshalb exemplarisch auf Basis des kommerziell verfügbaren ALM-Werkzeuges PTC Integrity mit Automotive Solution (siehe Kapitel 4.1.3) und entsprechender Erweiterungen implementiert.

Ziel ist es, sowohl den Bruch in der Werkzeugkette (siehe Kapitel 4.1.4.3) zu überwinden, als auch die Entwicklungsdokumente in derselben Werkzeugumgebung zu verwalten. Dieser Zusammenhang ist in Abbildung 71 dargestellt.

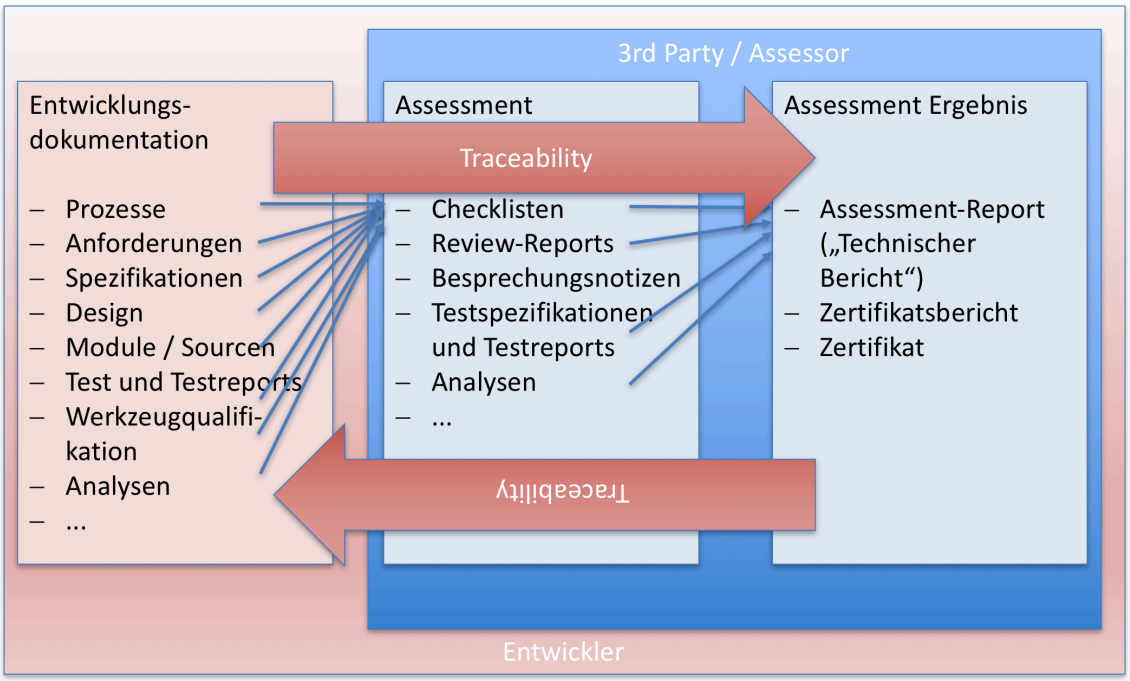

Abbildung 71: Idealer Workflow bei der Verwendung einer integrierten Assessment-Lösung

Der Vorteil dieses Ansatzes ist, dass eine direkte Nachverfolgbarkeit (Traceability) von den Entwicklungsdokumenten über das Assessment als solches bis hin zum Assessment-Ergebnis sichergestellt werden kann. Das erhöht die Effizienz in der Entwicklung und der Systemfreigabe. Darüber hinaus kann leichter die Vollständigkeit der Dokumente und deren Bewertung sichergestellt werden.

4.3.1 Technischer Ansatz

Das Assessment-Modell muss alle technischen, methodischen und inhaltlich formalen Anforderungen nach dem zu entwickelnden Standard und deren Prüfbarkeit abbilden. Es besteht zum einen aus den Anforderungen des Standards direkt, zum anderen aber werden die bereits beim traditionellen Ansatz (siehe Kapitel 4.1.4.1) verwendeten Checklisten vollständig digitalisiert und in das ALM-Werkzeug integriert. Das gewährleistet, dass alle Entwicklungsdokumente, die im ALM-Werkzeug ohnehin konsistent gehalten werden, nun mit den normativen Anforderungen und einzelnen Unterpunkten in den Checklisten für das Assessment im ALM-Werkzeug direkt miteinander verlinkt werden können. Dadurch wird die Durchgängigkeit von der Entwicklungsdokumentation bis hin zum Assessment-Bericht gewährleistet. Der Bruch in der Werkzeuglandschaft wird vermieden.

Diese Durchgängigkeit ermöglicht eine signifikante Effizienzsteigerung in der Entwicklung von sicherheitsrelevanten Systemen. Es kann zu jederzeit der Bearbeitungsfortschritt des Assessments nachvollzogen werden. Bei Änderungen gibt es zusätzlich zu den bereits genutzten Traces von Anforderungen zu Testfällen die Möglichkeit, Re-Assessments zu planen und dabei nur die betroffenen Dokumente einem Review zu unterziehen. Alle Abhängigkeiten von Dokumenten bleiben auch auf der Assessment-Ebene erhalten. Daraus abgeleitet, können auch auf einfache Art und Weise Management-Dashboards (siehe Abbildung 100 im Anhang E) erstellt werden, die in Echtzeit den Entwicklungsfortschritt inklusive der Freigabeaktivitäten komprimiert darstellen. So können Kenngrößen wie Requirements-Coverage[64] nicht nur auf der Entwicklungsseite, sondern auch auf der Assessment-Seite gemessen werden. Im Kontext des Assessments bedeutet das, dass alle normativen Anforderungen mit einem Link in die Entwicklungsdokumentation nachvollziehbar implementiert werden müssen und in der Assessment-Checkliste deren Umsetzung vom Assessor abgefragt und bewertet werden muss.

4.3.2 Das Datenmodell

Die in Abschnitt 4.2.3.2 erarbeiteten theoretischen Ansätze können mit einem ALM-Werkzeug umgesetzt werden. Im Folgenden wird die exemplarische Implementierung der Konzepte beschrieben:

- Traceability über die verschiedenen Artefakte,
- Integration der Safety Analysen (die in der ISO 26262 integraler Bestandteil sind) und
- Verifikation / Checklisten.

Als Basis für die Implementierung dient die klassische Funktionalität eines ALM-Werkzeuges, dass es erlaubt, alle für die Softwareentwicklung relevanten Dokumente und Artefakte zu verwalten und zu verlinken. Die Daten werden im ALM-Repository abgelegt. Dabei handelt es sich sowohl um Entwicklungsdokumente wie Spezifikationen und Analysen, als auch um Quell- und Target-Code. Darüber hinaus können über API-

[64] Die Requirements-Coverage ist der Grad der Abdeckung der vollständig implementierten Anforderungen. Es wird vorausgesetzt, dass jede Anforderung durch eine hinreichende Zahl von Testfällen abgeprüft wird. Anforderungen können technische, funktionale, nicht funktionale, aber auch methodische und formale, normative oder regulative Anforderungen sein.

Schnittstellen auch externe Applikationen angebunden werden (siehe Abbildung 72), z.B. mittels Open Services for Lifecycle Collaboration (OSLC)[65].

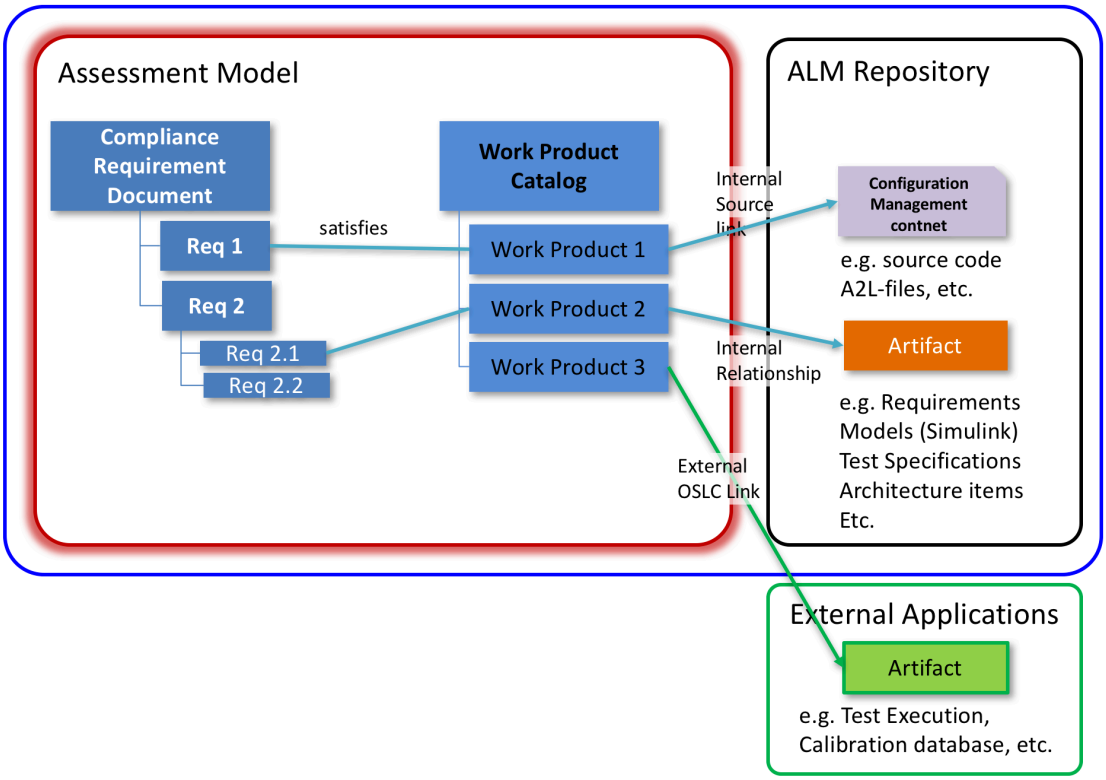

Abbildung 72: Das Datenmodell des Assessment-Modells in der Beispielimplementierung auf Basis von PTC Integrity

Die so erfassten sowie konsistent und persistent gespeicherten Dokumente bilden den Entwicklungsprozess hinreichend ab, allerdings ist der Freigabeprozess durch einen externen Assessor zuerst nicht im Fokus. Deshalb wird das Datenmodell im ALM-Werkzeug um das Assessment-Modell erweitert, das ebenfalls im ALM-Werkzeug an das ALM-Repository angebunden und verlinkt wird.

Dabei bildet das Assessment-Modell alle normativen Anforderungen in den Compliance-Requirements ab, deren Erfüllung vom Assessor direkt im Assessment-Modell dokumentiert werden kann. Die Compliance-Requirements sind eine Checkliste, die im Assessment-Modell hinterlegt ist und den Assessor bei seiner Arbeit durch das Projekt führt.

Grundlage für das Assessment sind stets die Informationen, die im Repository des ALM-Werkzeuges hinterlegt sind. Diese können dann auf die normativ geforderten

[65] OLSC ist eine offene Initiative, die die Einbindung von Softwareentwicklungswerkzeugen definiert (Li, Liu, & Yi, 2015) Seite 212f.

Arbeitsprodukte (Work Products in der ISO 26262) verlinkt werden. Die finale Bewertung wird somit durch die Verlinkung zwischen den Work Products und den Compliance-Requirements umgesetzt.

Da im Besonderen bei modellbasierter Entwicklung Softwaremodell und sonstige Dokumente mit anderen Werkzeugen (z.B. Enterprise Architect, Simulink, etc.) erzeugt werden, muss eine Einbindung dieser Daten ebenfalls ermöglicht werden. Das wird durch interne Links umgesetzt. Des Weiteren gibt es auch Werkzeuge, die z.B. Testdaten oder Kalibrierungsdaten verwalten. Durch die externen Links können auch die Informationen im Assessment-Modell zugänglich gemacht, für die Bewertung herangezogen und verlinkt werden.

Darüber hinaus ist es sinnvoll, Sicherheitsanalysen im Werkzeug abzubilden, deren Vorgehen durch den Standard vorgegeben ist. Da die ISO 26262 z.B. für die Gefährdungs- und Risikoanalyse ein genaues Vorgehen definiert (siehe Kapitel 2.2), kann dieses als Template in das Assessment-Modell integriert werden.

4.3.3 Beispielimplementierung auf Basis von PTC Integrity

Der im Rahmen dieser Arbeit entwickelte Demonstrator des Assessment-Modells wurde in PTC Integrity auf Basis des in Kapitel 4.1.4.2 beschriebenen Datenmodells implementiert. Das heißt, die normativen Anforderungen und die Inhalte der Checklisten konnten übernommen werden. Zusätzlich wurde ein Template für eine normgerechte Gefährdungs- und Risikoanalyse umgesetzt (siehe Abbildung 73). Dadurch wird ermöglicht, dass auch die vom Standard geforderten Safety-Analysen in derselben Werkzeugumgebung gehalten werden.

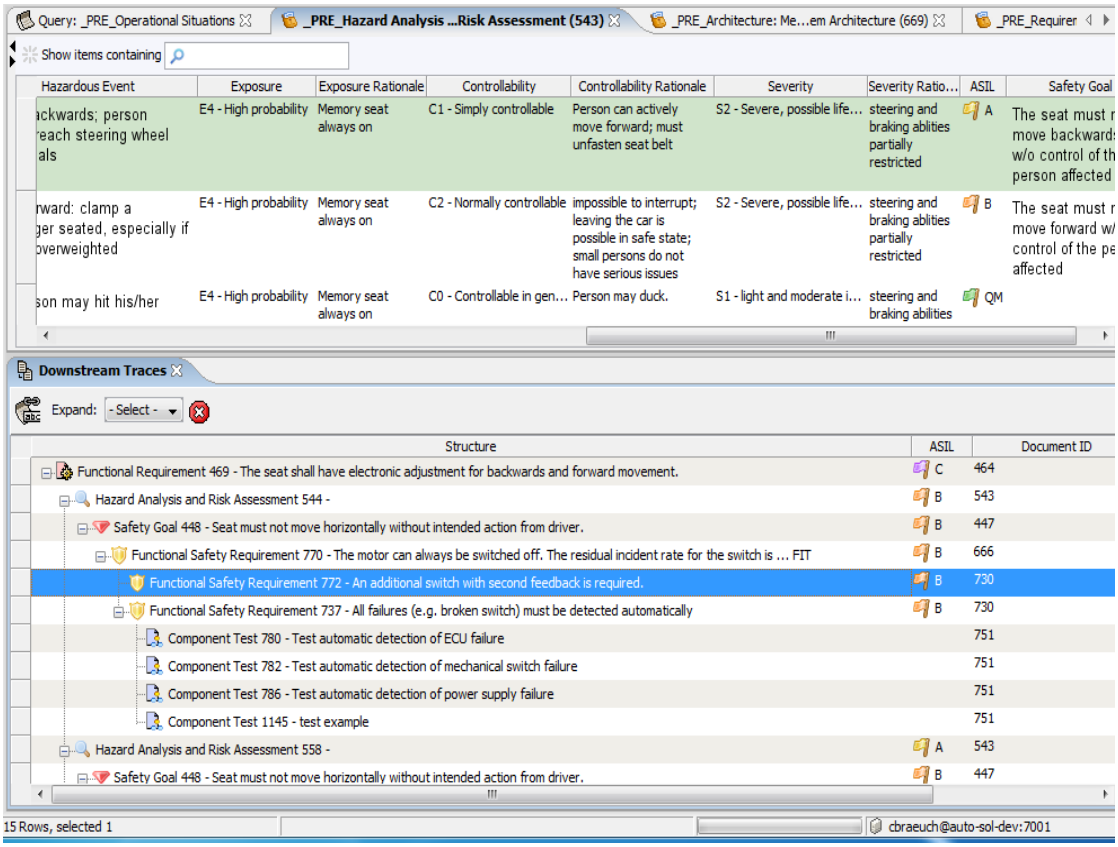

Abbildung 73: Umsetzung eines Templates für die Gefährdungs- und Risikoanalyse nach ISO 26262 auf Basis von PTC Integrity

Die von der ISO 26262 im Band 3 geforderten Inhalte sind in den Compliance-Requirements modelliert (siehe Abbildung 95 in Anhang E), die entsprechend der Norm auf die zu erstellenden Dokumente (Work Products) verlinkt werden können (siehe Abbildung 74).

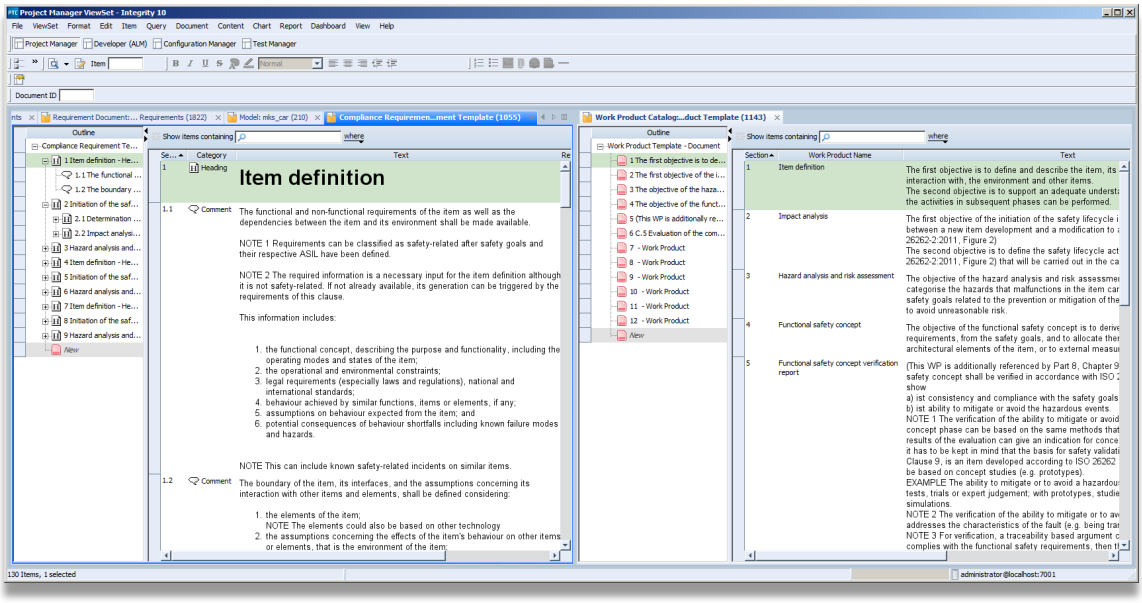

Abbildung 74: Zusammenspiel von Compliance-Requirements und den dazugehörigen Work Products nach ISO 26262

Den verschiedenen geforderten Work Products können die im ALM-Werkzeug hinterlegten Inhalte zugeordnet werden (siehe Abbildung 93 in Anhang E). Für Testdokumente gilt dieselbe Methodik (siehe Abbildung 94 in Anhang E). Die Work Products sind gewissermaßen ein virtueller View auf ein Subset der sicherheitsrelevanten, vom Standard geforderten Entwicklungsdokumente.

Jedem Work Product kann für das Assessment als Parameter die geforderte Review-Art (Verification oder Confirmation Review) zugewiesen werden, dies in Abhängigkeit von der Sicherheitskritikalität (ASIL-Einstufung) (siehe Abbildung 92 in Anhang E).

Die Compliance-Requirements bilden die Klauseln der Norm ab und werden mit Bewertungsparametern für den Assessor erweitert (siehe Abbildung 96 in Anhang E). Es gibt ein Auswahlfeld für das Assessment-Ergebnis und ein Freitextfeld für Kommentare, die später in den Assessment-Bericht automatisiert übernommen werden können. Jedes Compliance-Requirement befindet sich jeweils in einem Umsetzungszustand (unspecified, active, in review oder completed), wodurch der Bearbeitungsfortschritt festgehalten wird (siehe Abbildung 97 in Anhang E).

Für jedes einzelne Compliance-Requirement muss der Assessor eine Bewertung durchführen. Auch diese Dokumentation erfolgt direkt im ALM-Werkzeug. Folgende Bewertungsklassen sind dafür definiert: OK, NOT OK, PARTLY OK, NOT RELEVANT (siehe Abbildung 75).

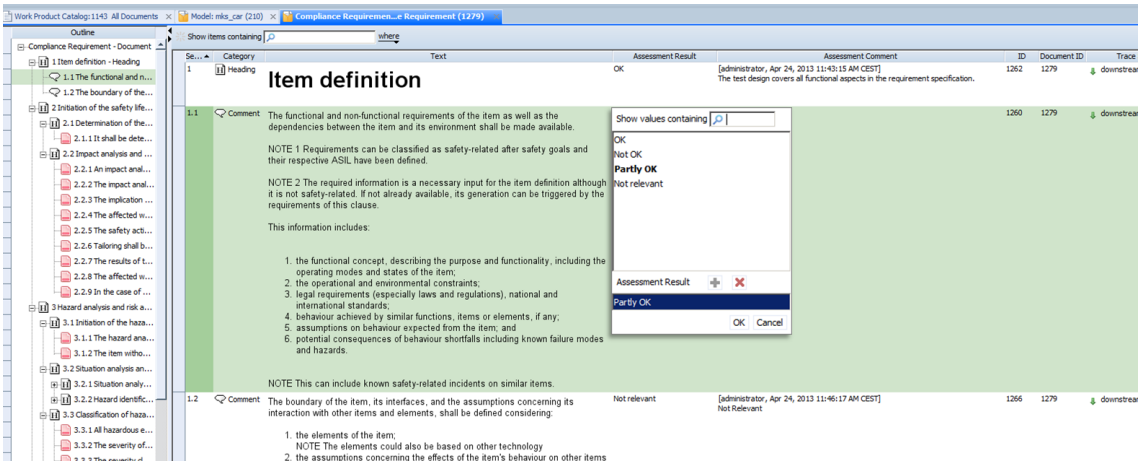

Abbildung 75: Bewertung der Erfüllung der Norm durch den Assessor

Für die Autorisierung von Benutzern wurde das bereits in PTC Integrity vorhandene Benutzer- und Rollenmanagement verwendet. Es musste lediglich die zusätzliche Rolle „Assessor“ hinzugefügt werden. Der Assessor hat Leseberechtigung bei allen sicherheitsrelevanten Dokumenten und hat Vollzugriff auf alle Masken, die dem Assessment-Modell zugeordnet sind.

Die Erfassung der Assessment-Ergebnisse direkt im Werkzeug bringt folgende Vorteile und Effizienzsteigerungen mit sich:

1. Offene Punkte sind in einer Prioritätenliste nahezu in Echtzeit sichtbar und können durch die Verlinkung auf die Entwicklungsdokumentation direkt bearbeitet werden (siehe Abbildung 99 in Anhang E). Dabei sind auch Seiteneffekte auf andere Dokumente sichtbar.
2. Diverse statistische Auswertungen über den Projekt- und Assessment-Fortschritt können grafisch dargestellt werden. Dies verbessert die Steuerungsmöglichkeiten der verantwortlichen Projektmanager sowohl für das Entwicklungsprojekt als auch für das Assessment erheblich (siehe Management-Dashboard in Abbildung 76).

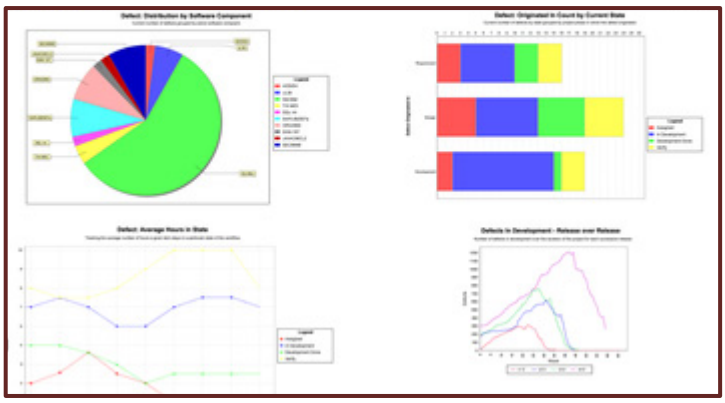

Abbildung 76: Management-Dashboard-Beispielimplementierung

Im Anhang E sind alle Bilder der Beispiel-Implementierung im Detail zu finden.

4.4 Fazit und Ausblick

Es konnte gezeigt werden, dass es möglich ist, auf Basis eines ALM-Werkzeuges eine Plattform für ein werkzeuggestütztes Assessment aufzusetzen. Damit wird die Forschungshypothese 2b gestützt. Um auf die Ergebnisse des Assessment-Werkzeuges vertrauen zu können, ist eine Qualifizierung der eingesetzten Werkzeugkette nötig. Im gezeigten Beispiel wurde auf eine bereits generisch qualifizierte ALM-Suite aufgesetzt, so dass hier keine Zusatzaufwände für die Qualifizierung entstehen. In der Forschungshypothese 2a wurde formuliert, dass genau dies möglich sein muss.

4.4.1 Bewertung des wirtschaftlichen Nutzens

Nach exemplarischer Verwendung dieser Methode in Beispielprojekten kann davon ausgegangen werden, dass die Assessment-Kosten (interne Aufwände und Kosten der unabhängigen Prüforganisation) bei Verwendung einer geeigneten Werkzeugkette deutlich reduziert werden können. Typische Assessment-Projekte eines Independent Safety Assessors (ISA) gliedert sich in 3 Phasen (im Folgendem am Beispiel der ISO 26262, siehe

Tabelle 10 bis

Tabelle 12):

Tabelle 10: Phase 1: Functional Safety Management und Konzeptbewertung

Activities

FSM 1:
Audit and review of the general and project specific safety organisation and the functional safety management. Tailoring of the safety lifecycle.
Agenda:
• Overall Safety management (Part 2)
  o 2-6 Safety management during the concept phase and the product development
  o 2-7 Safety management after the item´s release for production
• ISO 26262 : Part 8 Supporting Processes
  o 8-5 Interfaces within distributed developments
  o 8-6 Specification and management of safety requirements
  o 8-7 Configuration management
  o 8-8 Change management
  o 8-10 Documentation
  o 8-14 Proven in use argument
• Quality Management System including safety

Review of documents: • Safety Plan • Development interface agreement (DIA) • Configuration management plan • Change management plan • Proven in use argumentation (if applicable)
Concept und System 1: Assessment of the following topics according to ISO 26262-3,4, 5 und 6: • Safety Goals • Functional safety concept including the specification of the functional safety requirements • Technical safety concept incl. the system design specification with allocated technical safety requirements • Safety Analysis on system level (system FMEA and FTA) • HW architecture • SW architecture • HW/SW Interface HSI • Item integration and test plan (Confirmation Review) • Validation Plan (Confirmation Review) • Analysis of dependent failures
Review of documents: • Functional safety concept (with regard to content) • Technical safety concept (with regard to content) • Item integration and test plan (confirmation review) • Validation Plan (confirmation review) • Safety analysis on system level (confirmation review)

Tabelle 11: Phase 2: Detailbewertung

Activities
HW: Assessment and audit of the HW development phase according to ISO 26262-5: • Specification of HW safety requirements • Hardware Design: architecture and detailed design • Determination of HW architectural metrics - FMEDA • Evaluation of possible violations of safety goals according to random HW failures • Hardware integration and tests • Demonstration selected „*Fault Insertion Tests*“ • Qualification of HW components
Review of documents: • HW safety requirements (with regard to content) • HW design specification (with regard to content) • Safety analysis – FMEDA and FTA (confirmation review) • HW component qualification plan and report (with regard to content)
SW: Assessment and audit of the HW development phase according to ISO 26262-6: o SW verification plan o Specification of SW safety o Software Design: SW architecture, unit design and implementation o SW Analyses on architecture level (Confirmation Review) o Modelling and coding guidelines; guidelines for using SW-tools o Software unit tests o Software integration and integration tests o Verification of SW safety requirements o Demonstration of selected SW tests o Qualification of SW components
Review of documents: • SW safety requirements (with regard to content) • SW architecture (with regard to content) • SW safety analysis (confirmation review) • SW verification plan and specification(s) (random examination) • SW verification report(s) (random examination) • SW component qualification plan and report (with regard to content)

Tabelle 12: Phase 3: Projektabschluss

Activities

System 2:
Audit of the test and validation activities according to ISO 26262-4 und 7:
• Item integration und test
• Safety validation (*Validation report/plan*)
• Evidence of executed assessments and audits at the suppliers'
• Evidence of environmental testing
  • EMC testing
  • Tests for electrical safety
  • Environmental tests (temperature, shock&vibrations, etc.)

Review of documents: • Test reports (random examination) • Validation report (random examination)
FSM 2: Audit with additional spot checks of the general and project specific FSM for project conclusion: • List of open points • Safety plan • Safety case
Review of final documents: • Safety plan (final confirmation review) • Safety case (confirmation review)
Creation of the technical report
Project coordination and communication

Es werden grundsätzlich alle vom Standard geforderten Arbeitsergebnisse (Work Products, siehe Anhang A) in die Betrachtung einbezogen. Die Dokumente müssen auf formale Nachvollziehbarkeit (Traceablility), Konsistenz und Versionierung sowie technisch und inhaltlich gegen die Anforderungen der Prüfgrundlage geprüft werden. Ohne Werkzeugunterstützung beim Assessment müssen Dokumentenversionen aufwendig konsistent gehalten, Checklisten bearbeitet und am Ende die relevanten Assessment-Berichte geschrieben werden.

Im Umfeld des Autors wurden 63 Assessment-Projekte aus den Jahren 2011 bis 2014 bzgl. deren Gesamtaufwand untersucht, deren Aufwände sich auf die drei Projektphasen Konzept-, Detailphase und Projektabschluss wie in Abbildung 77 dargestellt verteilen.

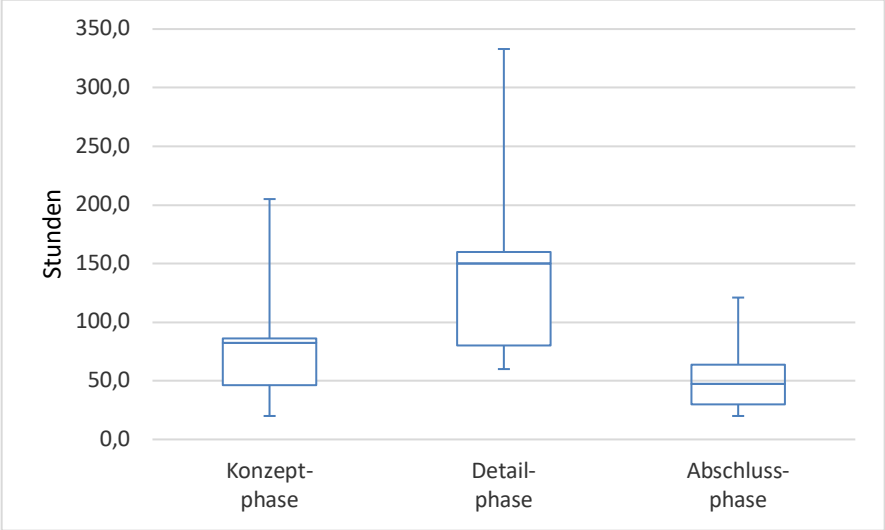

Abbildung 77: Verteilung der Aufwände auf die einzelnen Projektphasen in den untersuchten Projekten

Dabei verteilen sich die Arbeitsaufwände auf Projektarbeit wie Dokumentenstudium, Meetings, Telefonkonferenzen, Reisezeiten u. Ä. im Verhältnis zu Dokumentationsaufwänden (Bearbeiten von Checklisten, Erfassen der eingereichten Dokumente, Reviewprotokolle und -berichte, Besprechungsnotizen, Erstellung von Technischen Berichten und Zertifikatsberichten) wie in Abbildung 78 dargestellt.

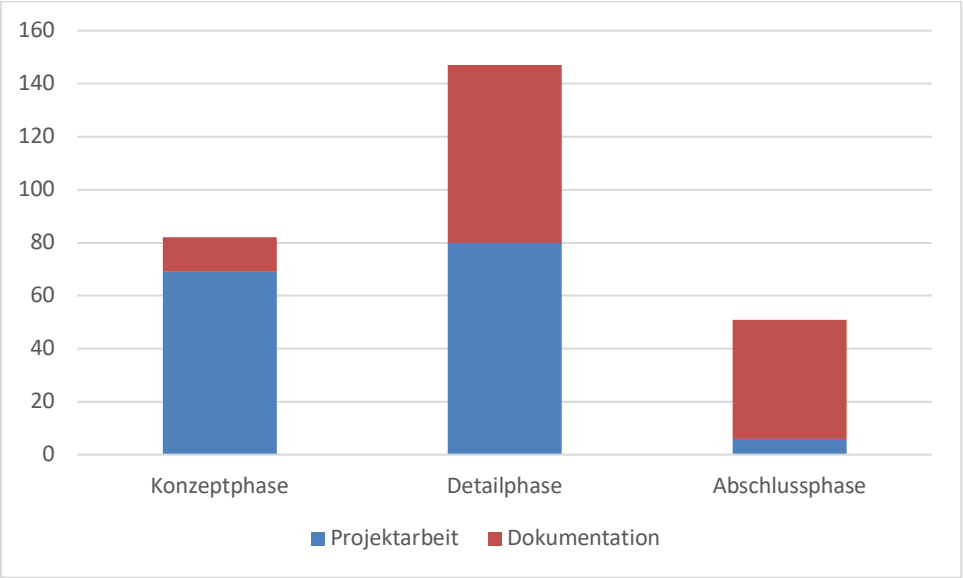

Abbildung 78: Anteil von Dokumentationsaufwänden im Projekt

Daraus ergibt sich ein durchschnittlicher Dokumentationsaufwand von 44,6% der Arbeitszeit pro Projekt. Es wurde in einem weiteren Schritt zusammen mit einem Team von 3 erfahrenen Assessoren das Einsparpotential bei der Verwendung eines Assessment-Werkzeuges abgeschätzt (siehe **Tabelle 13**).

Tabelle 13: Auswertung des Aufwands in 63 exemplarischen Projekten zur Funktionalen Sicherheit in Stunden

	Gesamtaufwand in h	Dokumentations-aufwand in h	Einsparpotential in h (geschätzt)
Konzeptphase	82	13	3
Detailphase	147	67	41
Projektabschluss	51	45	30
Gesamt	**280**	**125**	**74**

Der Effekt in der Konzeptphase ist noch nicht signifikant, da ein Projekt in dieser Phase grundsätzlich aufgesetzt werden muss; auch mit Werkzeugunterstützung sind hier Aufwände nötig. Der Effizienzgewinn wird in der Detailprüfung und im Projektabschluss deutlich, wo von Verlinkungen und der Berichtsgenerierung Kredit genommen werden kann. Somit ist eine durchschnittliche Zeiteinsparung von 74 h pro Assessment-Projekt möglich), was einer durchschnittlichen Einsparung von 26 % entspricht. Die Verteilung des Einsparpotentials auf die einzelnen Projektphasen ist in der Abbildung 79 detailliert dargestellt. Dabei entsprechen die „zusätzlichen manuellen Aufwände“ den Aufwänden, die ohne Werkzeugunterstützung nach dem herkömmlichen Vorgehen zusätzlich nötig sind und mit Werkzeugunterstützung somit eingespart werden können.

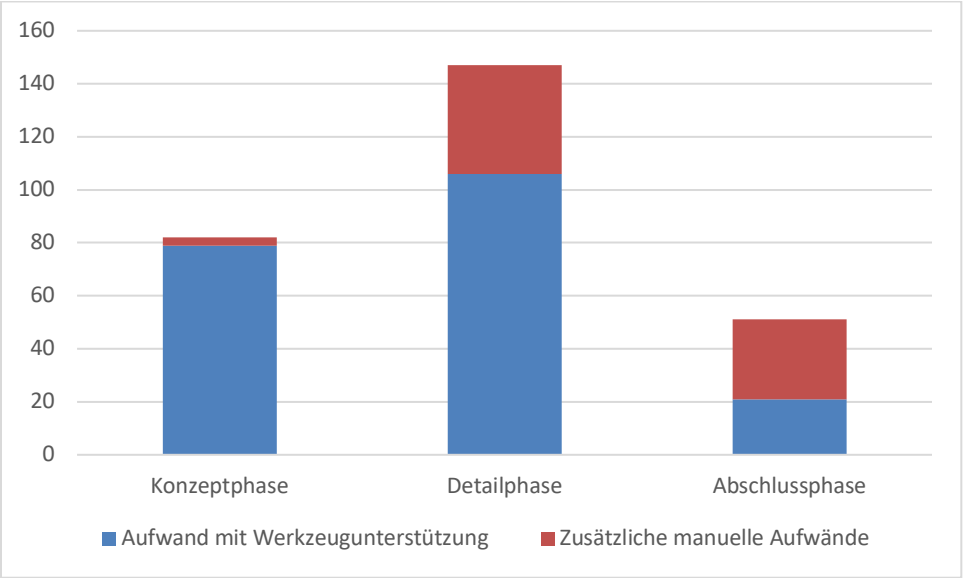

Abbildung 79: Einsparpotential durch die Verwendung eines integrierten Assessment-Werkzeuges

Prüforganisationen können somit mit dem gleichen Personalbestand einen höheren Projektdurchsatz realisieren, was bei der aktuellen Arbeitsmarktsituation und dem Fachkräftemangel einen erheblichen Wettbewerbsvorteil darstellt. Weitere Vorteile sind eine deutlich verbesserte Transparenz und die signifikante Beschleunigung bei Änderungen an sicherheitsrelevanten Systemen. Damit ist das Ziel der Forschungshypothese 2c, die deutliche Reduktion der Aufwände bei gestiegener Qualität und Nachvollziehbarkeit, erreicht.

4.4.2 Resultierende weiterführende wissenschaftliche Fragestellungen

Eine wesentliche noch zu lösende Aufgabe ist der von den Industrieunternehmen vorausgesetzte Grad an Informationssicherheit. Der Assessor benötigt Zugriff auf deren Core-IP. Das Assessment-Modell berücksichtigt diesen Aspekt momentan nicht. Hier muss gemeinsam mit der Industrie eine akzeptierte Methode etabliert werden. Der Ansatz aus Kapitel 4.2.3 würde zumindest die Vertraulichkeit der Entwicklungsdaten garantieren, da die Zertifizierungs-Suite in diesem Ansatz die Brücke zwischen den Entwicklungsdokumenten auf der einen Seite und den Zertifizierungsaktivitäten auf der anderen Seite schlägt. Dieser Ansatz zur Datenkapselung sollte in weiteren Forschungsarbeiten weiter eruiert werden. Die Herausforderung besteht darin, die Technologie für bereits bestehende ALM-Lösungen zu adaptieren, da es nicht sinnvoll erscheint, eine vollkommen neue und eigenständige kommerzielle Lösung zu entwickeln. Die Gründe dafür sind die Wirtschaftlichkeit und die Kundenakzeptanz, da in der Industrie bereits etablierte Entwicklungs-Tool-Ketten verwendet werden, die sonst mit erheblichem Aufwand verändert werden müssten.

Der nächste Schritt bei der Implementierung einer kommerziellen Lösung ist die vollständige Abbildung aller Bände der ISO 26262 im Assessment-Modell. Hierzu müssen lediglich die Anforderungen der Norm und die bereits existierenden Checklisten in das System übertragen werden.

In einer weiteren Ausbaustufe können weitere Standards im Assessment-Modell abgebildet werden. Dabei ist die Methodik nicht auf die Abbildung von Safety-Standards begrenzt, sondern grundsätzlich auch für jede Art von Standards und Normen geeignet. Die Systematik ist dabei stets dieselbe – es werden die Anforderungen selbst und die Kriterien zur Bewertung benötigt. Wird dann im ALM-System die

Entwicklungsdokumentation hinterlegt, kann das Assessment direkt auf den Produktentwicklungsdaten erfolgen.

Die Methodik des Assessment-Modells ist nicht von der Verwendung eines speziellen ALM-Werkzeuges abhängig, sie kann grundsätzlich für jedes ALM-System adaptiert werden. Grundlage hierfür ist immer das in die dieser Arbeit hergeleitete Datenmodell.

5 Zusammenfassung und Ausblick

Ausgehend von den wissenschaftlichen Zielen und den daraus abgeleiteten Forschungshypothesen wurden im Rahmen dieser Arbeit ein Vorgehen für eine generische Werkzeugqualifizierung und ein Ansatz für eine werkzeugunterstützte Sicherheitsbewertung entwickelt. Die erreichten Ergebnisse sind in Tabelle 14 und Tabelle 15 zusammengefasst.

Tabelle 14: Forschungsschwerpunkt 1 „Sichere Werkzeuge und Werkzeugketten“ mit den abgeleiteten Forschungshypothesen und den Ergebnissen dieser Arbeit

Sichere Werkzeuge und Werkzeugketten: **Entwicklung eines generischen Ansatzes zur Bewertung der Vertrauenswürdigkeit von Werkzeugen und Werkzeugketten**	
1a	Verschiedene internationale Standards stellen Anforderungen an Softwareentwicklungsprozesse. Sicherheitsnormen verlangen den vertrauenswürdigen Umgang mit den in der Systementwicklung eingesetzten Softwareentwicklungswerkzeugen. Hier gibt es Überschneidungen, die für einen generischen Ansatz genutzt werden können.
	Die Analyse der im Umfeld der Funktionalen Sicherheit relevanten Standards hat gezeigt, dass die Anforderungen der einzelnen betrachteten Standards vergleichbar sind. Im Wesentlichen führt die Umsetzung der Anforderungen zur Implementierung eines strukturierten Softwareentwicklungsprozesses, der letztendlich zu einer hohen Softwarequalität führt.
1b	Die Qualifizierung von Softwareentwicklungswerkzeugen muss unabhängig von konkreten Projekten, Domänen und Standards erfolgen. Ziel ist ein generischer Ansatz für die Werkzeugqualifizierung.
	In der Arbeit wurde eine generische Checkliste hergeleitet, die auf den Anforderungen der betrachteten Standards beruht und zusätzlich Best Practices einschließt. Dadurch ist es möglich, die Werkzeugqualifizierung generisch und somit unabhängig vom Projektkontext und der Industriedomäne durchzuführen.

1c	Zahlreiche Softwareprodukte haben eine langjährige Historie. Bei der Qualifizierung von Softwareentwicklungswerkzeugen muss ein geeigneter Methodenmix angewendet werden, der sich am individuellen Produkt ausrichtet.
	Der in dieser Arbeit entwickelte Qualifizierungsansatz schließt die Kombinationen mehrerer Qualifikationsmethoden ein. Somit ist es möglich, für Bestandsmodule einen Validierungsansatz zu verwenden und für neu entwickelte Teile moderne Softwareentwicklungsprozesse inkl. Änderungsmanagement vorzuhalten.

Es wurde gezeigt, dass es möglich ist, Werkzeuge und Werkzeugketten generisch zu qualifizieren. Dabei gibt es keinen Anwendungs- bzw. Projektbezug. Die Qualifizierungsergebnisse sind domänenübergreifend nutzbar.

In der Zukunft wird die Betrachtung allgemeiner Qualitätsstandards und das Assessment der Funktionalen Sicherheit zunehmend verschmelzen. Im Automotive-Umfeld sollten zukünftig die Themen Softwarequalität (Automotive SPICE) und Funktionale Sicherheit integral betrachtet werden. Es empfehlen sich einheitliche Softwareentwicklungsprozesse, die durch geeignetes Tailoring sowohl für Projekte mit Sicherheitskritikalität als auch für Standardprojekte genutzt werden können.

Tabelle 15: Forschungsschwerpunkt 2 „Werkzeugunterstützte Sicherheitsnachweise“ mit den dazugehörigen Forschungshypothesen und den Ergebnissen dieser Arbeit

Werkzeugunterstützte Sicherheitsnachweise: **Herleitung eines integralen Ansatzes, um Safety-Assessments effizienter und präziser innerhalb eines ALM-Werkzeuges durchführen zu können**	
2a	Qualifizierte Softwarewerkzeuge und Werkzeugketten, die in der Entwicklung sicherheitsrelevanter Systeme eingesetzt werden, sollen durch eine geeignete Methodik auch für die Beurteilung der Funktionalen Sicherheit (Assessment) und die Freigabe solcher Systeme verwendet werden.
	Es konnte gezeigt werden, dass es möglich ist, auf Basis eines qualifizierten ALM-Werkzeuges eine Plattform (Assessment-Modell) für ein werkzeuggestütztes Assessment aufzusetzen.

2b	Moderne Konfigurationsmanagement-Systeme mit integriertem Anforderungsmanagement bilden die Klammer um alle Dokumente im Softwareentwicklungsprozess und können somit durch eine geeignete Erweiterung der Funktionalität auch für das Assessment verwendet werden.
	Im Assessment-Modell wird auf Basis der realen Entwicklungsdaten, die im ALM-Werkzeug hinterlegt sind, die Bewertung der Funktionalen Sicherheit vorgenommen. Damit ist eine vollständige Traceability von den Systemanforderungen bis hin zum Assessment-Bericht möglich.
2c	Werkzeugunterstütztes Assessment reduziert die Aufwände erheblich und verbessert die Qualität und Vollständigkeit von Assessments.
	Im Rahmen der Bewertung des wirtschaftlichen Nutzens konnte gezeigt werden, dass eine Reduktion des Aufwandes um bis zu 26% bei der Begutachtung eines sicherheitsrelevanten Systems möglich ist.

Im Rahmen dieser Arbeit wurde eine Methodik für ein werkzeugunterstütztes Assessment entwickelt. Die Anwendung dieser Methodik hat ein bedeutendes betriebswirtschaftliches Potential. In einer Beispielbetrachtung konnte demonstriert werden, dass die Aufwände für das Assessment sicherheitsrelevanter Systeme signifikant reduziert werden können. Dabei sind Einsparungen von mehr als einem Viertel des Assessment-Aufwandes möglich. Durch die Beispielimplementierung konnte gezeigt werden, dass die Umsetzung auch praktisch möglich ist. In einem nächsten Schritt kann auf dieser Basis eine kommerzielle Lösung entwickelt werden, die die Anforderungen eines beliebigen Sicherheitsstandards vollständig implementiert.

Eine Herausforderung in der Zukunft wird die Integration von Anforderungen an die Informationssicherheit (Information Security) sein. Dabei gibt es ebenfalls technische und prozessuale Anforderungen. Wichtig ist diese Betrachtung deshalb, da durch eine Kompromittierung des Systems aus einem Security-Incident ein Safety-Issue werden kann, Security also Safety beeinflusst. Die Standards zur Funktionalen Sicherheit decken dieses Szenario aktuell nur unzureichend ab oder verweisen auf die einschlägigen Security-Normen. Ein integraler Ansatz auf normativer Seite fehlt.

Ein weiterer wesentlicher Trend, der in der nahen Zukunft zunehmend an Bedeutung gewinnen wird, ist die stark steigende Verwendung von Free- und Open-Source-Software. Grund dafür ist eine erhebliche Beschleunigung der Entwicklung durch die Verwendung bereits existierender Komponenten (siehe Abbildung 80) und darüber hinaus eine signifikante Einsparung bei den Lizenzkosten.

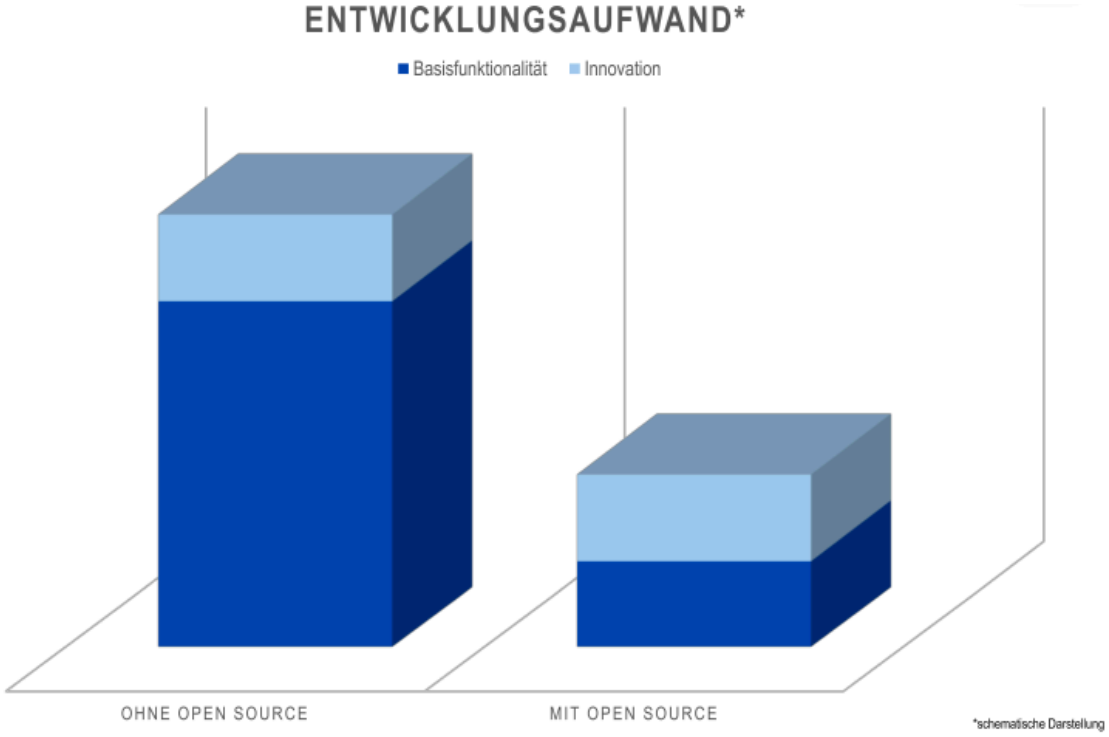

Abbildung 80: Entwicklungsaufwand mit und ohne Verwendung von Open-Source-Komponenten (schematische Darstellung) nach (Ainöder et al., 2017)

Der Annahme folgend, dass in Zukunft bei Innovationen aus der Perspektive Entwicklungskosten und Time to Market der Einsatz von Open-Source-Komponenten aus Gründen der Wirtschaftlichkeit unverzichtbar ist, müssen Themen wie Qualitätssicherung, Lizenz-Management und das Assessment solcher Softwarekomponenten weiter erforscht und vor allem formalisiert werden. Bei IT-Applikationen ist der Einsatz von Open Source bereits Stand der Technik. Alle großen Player im IT-Markt, wie z.B. Amazon, Facebook, Google, IBM, Microsoft oder SAP, setzen bereits seit Ende der 1990er Jahre großflächig erfolgreich Open-Source-Software ein (Bärwald et al., 2018). Für embedded-Anwendungen ist eine vergleichbare Dynamik bei der Verwendung von Open Source (in Hardware und Software) in den kommenden Jahren zu erwarten. Erste

industrietaugliche Lösungen auf Raspberry-Pi-Basis mit einem Linux-Betriebssystem sind bereits verfügbar[66].

Da Open-Source-Projekte nicht nach den in den Sicherheitsstandards geforderten stringenten Prozessmodellen gesteuert werden, sondern einem „Community-Ansatz" folgen, sind bei der Beurteilung der Open-Source-Komponenten völlig neue Ansätze nötig. Hierzu sind weitere Forschungsarbeiten nötig, wie z.B. aktuell im Rahmen des Projektes „Safety Critical Linux" (OSADL eG, 2018) des Open Source Automation Development Lab OSADL (OSADL eG, 2017).

[66] z.B. die Lösung Revolution Pi der Firma KUNBUS GmbH (https://revolution.kunbus.de/)

Abkürzungsverzeichnis

ACQ	Aquisition Process Group
ADAS	Advanced Driver Assistant System
ADT	Abstact Data Types
ALM	Application Lifecycle Management
ANSI	American National Standards Institute
API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
(A)SPICE	(Automotive-) Software Process Improvement and Capability Determination
ATEX	Atmosphères Explosibles
BIT	Bundesamt für Informationstechnik
CENELEC	Comité Européen de Normalisation Électrotechnique
CMMI	Capability Maturity Model Integration
CM	Configuration Management (System)
COTS	Commercial Off-The-Shelf
CRC	Cyclic Redundancy Check
DAL	Design Assurance Level
DBMS	Datenbankmanagementsystem
DC	Diagnostic Coverage
DFKI	Deutsches Forschungszentrum für Künstliche Intelligenz
DIA	Development Interface Agreement
DIS	Draft International Standard
DIN	Deutsches Institut für Normung
DKE	Deutsche Kommission Elektrotechnik Elektronik Informations-technik
EAL	Evaluation Assurance Level (Common Criteria)
EASA	European Aviation Safety Agency EBA Eisenbahn-Bundesamt
EBO	Eisenbahn-Bau- und Betriebsordnung
EN	Europäische Norm
EPB	Electric Parking Brake
EPS	Electric Power Steering

ERM	Entity Relationsship Model
ESP	Electronic Stability Control
EUROCAE	European Organization for Civil Aviation Equipment
FAA	Federal Aviation Administration
FME(D)A	Failure Model and Effects (and Diagnostics) Analysis
FTA	Fault Tree Analysis
FSM	Functional Safety Management
HIL	Hardware In The Loop
HIS	Hersteller-Initiative Software
HSI	Hardware Software Interface
HW	Hardware
IABG	Industrieanlagen-Betriebsgesellschaft
IEC	International Electrotechnical Commission
IIC	Industrial Internet Consortium
IIRA	Industrial Internet Reference Architecture
IISF	Industrial Internet Security Framework
IoT	Internet of Things
IP	Intellectual Property
ISA	Independent Safety Assessor
ISO	International Organization for Standardization
IT	Informationstechnik
JDBC	Java Database Connectivity
MAN	Management (Process Group)
MIL	Model in the Loop
NooM	N out of M
ODBC	Open Database Connectivity
OEM	Original Equipment Manufacturer
OO	Objekt-orientiert (Methode)
OS	Operating System
OSLC	Open Services for Lifecycle Collaboration
PAM	Process Assessment Model (Automotive-SPICE)
PCD	Process Capability Determination

PDM	Produkt-Datenmanagement-System
PFD	Probability of Failure on Demand
PFH	Probability of Failure per Hour
PIL	Processor In The Loop
PLM	Product Lifecycle Management
QA	Quality Assurance
QM	Qualitätsmanagement
QMA	Qualitätsmanagementsystem
QMC	Qualitäts-Management-Center
REU	Reuse (Process Group)
RTCA	Radio Technical Commission for Aeronautics
SaaS	Software as a Service
SEI	Software Engineering Institute
SFF	Safe Failure Fraction
SIL	Safety Integrity Level
SIL	Software In The Loop
SPICE	Software Process Improvement and Capability Determination
SPL	Supply (Process Group)
SQL	Structured Query Language
SRS	Safety Requirements Specification
SUP	Support Untersützungsprozess; Supporting (Process Group)
SW	Software
SWE	Software Engineering (Process Group)
SYS	System Engineering (Process Group)
TC	Technical Commission
TCL	Tool Confidence Level
TD	Tool Error Detection
TI	Tool Impact
TMR	Triple Modular Redundancy
UI	User Interface
UML	Unified Modeling Language
VDA	Verband der Deutschen Automobilindustrie

VDE	Verband der Elektrotechnik, Elektronik und Informationstechnik
VSE	Verification Support Environment
VwV	Verwaltungsvorschrift
WG	Working Groupe
XOR	Exclusive OR

Literaturverzeichnis

Ainöder, A., Bärwald, A., & Pappler, N. (2017). Free and Open Source Software (FOSS) sicher in IoX-Projekten verwenden. In *Internet of Things - vom Sensor bis zur Cloud 2017*. München: Weka Fachmedien GmbH.

ARM. (2017). Compiler Safety Package. Retrieved February 18, 2017, from http://www2.keil.com/mdk5/safety

ARTEMIS (Advanced Research & Technology for Embedded Intelligence and Systems). (2009). Reduced Certification Costs for Trusted Multi-core Platforms (RECOMP). *Business*. Retrieved from https://artemis-ia.eu/project/poster/download/771

Asplund, F., El-khoury, J., & Törngren, M. (2012). Qualifying Software Tools, a Systems Approach. In F. Ortmeier & P. Daniel (Eds.), *SAFECOMP 2012, LNCS 7612* (pp. 340–351). Berlin, Heidelberg: Springer Verlag.

Augsten, J. (2008). *Entwicklung eines Softwaretools zur Bestimmung der Ausfallwahrscheinlichkeit sicherheitsgerichteter Hardware*. Hochschule München.

Bährle, K. R. (2016). *Marktanalyse zu sicherheitskritischer Software in der Automobilbranche*. Karlsruher Institut für Technolgie.

Balzert, H. (2009). *Lehrbuch der Softwaretechik: Basiskonzepte und Requirements Engineering* (3. Auflage). Heidelberg: Spektrum Verlag.

Bärwald, A. (2002). *Untersuchung neuer Funktionalitäten ausgewählter Datenbankbetriebssysteme - Diplomarbeit*. Hochschule für Technik und Wirtschaft Dresden.

Bärwald, A. (2004). MISRA C und IEC61508 - Zertifizierung einer IEC 61508-konformen Software-Entwicklung mit MISRA C. In *MISRA DAY 2004*. Ludwigsburg: QA Systems GmbH.

Bärwald, A. (2005). *IEC 61508 and MISRA C - The Benefits of Utilising IEC 61508 and MISRA C for Automotive Applications in The 1st IEE Automotive Electronics Conference, p7-14*. London: The Institution of Engineering and Technology.

Bärwald, A. (2006). Werkzeuge in einem sicherheitsgerichteten Entwicklungsprozess. In *safetronic'06*. München: Hanser Verlag.

Bärwald, A. (2010a). Certification of safety relevant systems - Benefits of using pre-certified components. In *Wind River Industrial Event Paris 2010*. Paris: Wind River Systems.

Bärwald, A. (2010b). ISO 26262 - Der neue Automotive Standard für Funktionale Sicherheit. In *Bayerisches IT-Sicherheitscluster: Automotive Forum AFS3*. Regensburg: Bayerisches IT-Sicherheitscluster e.V.

Bärwald, A. (2013). Functional Safety in the Automotive Industry - Status quo & upcoming trends from the point of view of a certification authorit. In *The 2nd International Symposium on Road Vehicles - Functional Safety Standards and its Application*. Shanghai: CATARC.

Bärwald, A. (2014a). Functional Safety in the Automotive Industry - ISO 26262 - Status quo of the ISO 26262: A snapshot from the point of view of an assessment and certification authority. In *TÜV SÜD safe future now conference 2014*. Tokyo: TÜV SÜD Japan Ltd.

Bärwald, A. (2014b). Functional Safety in the automotive industry - Status quo & upcoming trends from the point of view a certification authority. In *Automotive Embedded Multi-Core Systems*. Sindelfingen: IQPC Gesellschaft für Management Konferenzen mbH.

Bärwald, A. (2016). Software Tool Qualification and Certification for Functional Safety. In *IIC Meeting in Tokyo, June 7th, 2016*. Tokyo: Industrial Internet Consortium.

Bärwald, A., & Barry, R. (2009). Creating Safety Relevant Systems by using Generic Software Components (for example an RTOS). In *Embedded Systems Conference ESC UK 2009*. Farnborough.

Bärwald, A., & Beine, M. (2009). Einsatz zertifizierter Codegenerierungswerkzeuge in sicherheitsgerichteten Entwicklungen. In *safetronic'09*. München: Hanser Verlag.

Bärwald, A., & Beine, M. (2010). Sichere Codegenerierung. *HANSER Automotive*, *1–2/2010*, 30–33.

Bärwald, A., & Bräuchle, C. (2013). Reproduzierbare und effiziente Toolqualifizierung und TCL-Avoidance. In *safe.tech 2013*. München: TÜV SÜD Akademie GmbH.

Bärwald, A., Bräuchle, C., & Mottok, J. (2015a). Tool based assessments. In *IQPC Embedded Conference*. Berlin: IQPC Gesellschaft für Management Konferenzen mbH.

Bärwald, A., Bräuchle, C., & Mottok, J. (2015b). Toolunterstützte Assessments. In *safe.tech 2015*. München: TÜV SÜD Akademie GmbH.

Bärwald, A., & Bruce-Boye, C. (2008). Softwarebus für sicherheitsgerichtete Anwendungen. In *Konferenz Sicherheitsgerichtete Systeme 2008*. Stuttgart: Deutsche Kongress.

Bärwald, A., & Buchwieser, A. (2008). Hypervisor - Virtualisierungsplattform zur Trennung von Sicherheitsfunktionen verschiedener Safety Integrity Level. In *safetronic'08*. München: Hanser Verlag.

Bärwald, A., & Buchwieser, A. (2009). Hypervisor - Virtualisierungsplattform zur Trennung von Sicherheitsfunktionen verschiedener Safety Integrity Level. In *SIL Conference 2009*. Basel: TÜV SÜD Akademie GmbH.

Bärwald, A., Deubzer, M., Hobelsberger, M., & Mottok, J. (2012). Toolunterstützung für die Entwicklung von eingebetteten Multicore-Systemen mit Safety- und Echtzeitanforderungen. In *safe.tech 2012*. München: TÜV SÜD Akademie GmbH.

Bärwald, A., Deubzer, M., & Mottok, J. (2010). Dependability-Betrachtung von Multicore-Scheduling. *HANSER Automotive*, *11/2010*, 24–27.

Bärwald, A., & Duygun, S. (2008). Implementierung eines sicherheitsgerichteten Software-Entwicklungsprozesses am Beispiel eines Kfz-Zulieferers. In *safetronic'08*. München: Hanser Verlag.

Bärwald, A., Hauff, H., & Mottok, J. (2009). Qualification and Certification of Development Tools for Safety-Critical Applications. In *dependability@Siemens'2009*. München: SIEMENS AG.

Bärwald, A., Hauff, H., & Mottok, J. (2010). Qualifizierung und Zertifizierung von Software-Entwicklungswerkzeugen. *HANSER Automotive*, *1–2/2010*, 34–39.

Bärwald, A., Inderst, M., Rosen, R., Smit-Wiesner, E., Stelzig, P. E., Wehrstedt, J. C., & Wild, D. (2011). Simulation im Zertifizierungsprozess von Produkten : Anforderungen an die Simulationsmodelle und die Simulationswerkzeuge (Certification of Embedded Software System : State of the Art and Future Requirements). *Software Plattform Embedded Systems (SPES) 2020*.

Bärwald, A., Inderst, M., Rosen, R., Stelzig, P. E., Wehrstedt, J. C., & Wild, D. (2011). Zertifizierung von Automatisierungssoftware : Ist-Analyse und Anforderungen. *Software Plattform Embedded Systems (SPES) 2020*.

Bärwald, A., Inderst, M., & Soyer, D. (2012). SPES 2020 "Konzept einer PDM-Tool gestützten Erstellung von Software-Systemen und einer Zertifizierungs-Suite". *Software Plattform Embedded Systems (SPES) 2020*.

Bärwald, A., & Lange, B. (2012). Konsequent - Interview: ISO 26262 fordert Aktivitäten im kompletten Entwicklungszyklus. *IX Extra*, *2/2012*, VI ff.

Bärwald, A., Maracke, D. C., Takanishi, K., & Pappler, N. (2018). Open Source Licensing and Governance. In *sate.tech 2018*. Munich: TÜV SÜD Akademie GmbH.

Bärwald, A., & Mottok, J. (2010). Zertifizierung von Werkzeugen und Werkzeugketten. In *Software im Automobil 2010*. Stuttgart: Euroforum.

Bärwald, A., Mottok, J., & Hauff, H. (2010). Qualification and Certification of Development Tools for Safety-Critical Applications. In (Hrsg.), H. B. Keller, E. Plödereder, P. Dencker, & H. Klenk (Eds.), *Automotive - Safety & Security 2010 - Sicherheit und Zuverlässigkeit für automobile Informationstechnik* (pp. 135–149). Stuttgart: Shaker Verlag GmbH.

Bärwald, A., & Pfisterer, P. (2017). IEC 61850 Interoperability & Integration Integrity Checks. In *DISTRIBUTECH 2017*. San Diego: PennWell Corporation Tulsa.

Bärwald, A., & Spanfelner, B. (2011). Softwarewerkzeuge für sicherheitsgerichtete Entwicklungen Anforderungen und Lösungsansätze. In *Funktionale Sicherheit - Fachkonferenz*. Düsseldorf: Parametric Technology GmbH.

Bärwald, A., & Wild, D. (2011). Softwarewerkzeuge für sicherheitsgerichtete Entwicklungen Anforderungen und Lösungsansätze. In *2.FuSi Forum Osnabrück*. Osnabrück: UB Dietz.

Becker, B. (1995). *Gefährungshaftung und unternehmerischen Entscheidung*. Wiesbaden: Deutscher Universitäts Verlag.

Beemster, M. (2013). ISO 26262 Qualification of Compilers using SuperTest. Retrieved from http://www.ukintpress-conferences.com/uploads/SPKTXEU14/Day2_1_Marcel_Beemster.pdf

Beine, M. (2009). *Model Based Software Development for Safety-Critical Systems – TargetLink Reference Workflow*. Paderborn: dSPACE GmbH.

Beschaffungsamt des Bundesministeriums des Innern. (2015). *UfAB VI - Unterlage für Ausschreibung und Bewertung von IT-Leistungen*. Bonn: Bundesministerium des Innern.

Birch, J., Rivett, R., Habli, I., Bradshaw, B., Botham, J., Higham, D., … Palin, R. (2013). Safety cases and their role in ISO 26262 functional safety assessment. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8153 LNCS*, 154–165. http://doi.org/10.1007/978-3-642-40793-2_15

Bormann, J. (2009). *Vollständige funktionale Verifikation*. Universität Kaiserslautern. Retrieved from http://kluedo.ub.uni-kl.de/volltexte/2009/2356/

Brilliant, S. S., Knight, J. C., & Leveson, N. G. (1990). Analysis of Faults in an N-Version Software Experiment. *IEEE Transaction on Software Engineering, Vol.16, No.2*.

Bröhl, A.-P. (Hrsg. ., & Dröschel, W. (Hrsg. . (1995). *Das V-Modell: Der Standard für die Softwareentwicklung mit Leitfaden* (2. Auf.). München-Wien: Oldenbourg Verlag.

Brown, D. (1998). *Solving the Software Safety Paradox*. embedded.com, 1998. Retrieved from http://www.embedded.com/98/9812/9812feat2.htm, ftp://ftp.embedded.com/pub/1998/cpu_test.txt

BTC Embedded Systems AG. (2017). Embedder Tester. Retrieved February 18, 2017, from https://www.btc-es.de/de/produkte/btc-embeddedtester/iso-26262.html

Bulach, D., Martin, E., Raichle, D., & Dr. Lauff, U. (2013). Entwicklung von Motor-Generatorsteuerungen mit ETAS ASCET. *Hanser Automotive - OEM SUPPLIER*, 44–46.

Bundesamt für Strahlenschutz. (2011). *Rechnerbasierte Sicherheitsleittechnik für den Einsatz in der höchsten Sicherheitskategorie in deutschen Kernkraftwerken - RSK-Arbeitsgruppe EINSATZ RECHNERBASIERTE LEITTECHNIK (ERL)*. Salzgitter: Bundesamt für Strahlenschutz.

Bundesgerichtshof. (2008). *Urteil VI ZR 107/08*. Retrieved from http://juris.bundesgerichtshof.de/cgi-bin/rechtsprechung/document.py?Gericht=bgh&Art=en&Datum=Aktuell&Sort=12288&nr=48599&pos=20&anz=565&Blank=1.pdf

Bundesministerium für Umwelt für Naturschutz und Reaktorsicherheit. (2004). Risikomangagment im Rahmen der Störfall-Verordnung. *Bericht SFK-GS-41*, 1–106. Retrieved from papers3://publication/uuid/3469D739-BE8A-4632-984E-9BCA0D087EF9

Burton, S., Likkei, J., Vembar, P., & Wolf, M. (2015). *Automotive Functional Safety = Safety + Security*. York, Stuttgart, München: ETAS GmbH, Robert Bosch GmbH, ESCRYPT GmbH.

Canditt, S., Rauh, D., & Wittmann, M. (2010). Brückenschlag, Das V-Modell XT mit Scrum Inside. *OBJEKTspektrum*, *5*, 55–62.

CMMI Product Team. (2010). CMMI for Development, Version 1.3. *Software Engineering Institute*. Retrieved from http://repository.cmu.edu/sei/387

Deutsche Norm. (2001). DIN EN 50128 - Bahnanwendungen - Telekommunikationstechnik, Signaltechnik und Datenverarbeitungssysteme Kurzbeschreibung: Software für Eisenbahnsteuerungs- und Überwachungssysteme, *50128*(831004).

Deutsche Norm. (2005a). DIN EN 61511-1 (VDE 0810-1) Funktionale Sicherheit – Sicherheitstechnische Systeme für die Prozessindustrie – Teil 1 : Allgemeines , Begriffe , Anforderungen an Systeme , Software und Hardware (IEC 61511-1 : 2003 + Corrigendum 2004); Deutsche Fassung, *1*(810000).

Deutsche Norm. (2005b). DIN EN 62061:2005-10 (VDE 0113-50:2005-10) Sicherheit von Maschinen - Funktionale Sicherheit sicherheitsbezogener elektrischer, elektronischer und programmierbarer elektronischer Steuerungssysteme.

Deutsche Norm. (2005c). DIN EN ISO/IEC 17025:2005 Allgemeine Anforderungen an die Kompetenz von Prüf- und Kalibrierlaboratorien.

Deutsche Norm. (2007a). DIN EN 62304 (VDE 0750-101) Medizingeräte-Software – Software-Lebenszyklus-Prozesse Deutsche Fassung EN 62304 : 2006 Zusammenhang mit Europäischen und Internationalen Normen, *62304*(750124).

Deutsche Norm. (2007b). DIN IEC 60880 (VDE 0491-3-2) Kernkraftwerke – Leittechnik für Systeme mit sicherheitstechnischer Bedeutung – Softwareaspekte für rechnerbasierte Systeme zur Realisierung von Funktionen der Kategorie A, *60880*(491011).

Deutsche Norm. (2008a). DIN EN ISO 13849-1:2008-12 Titel (deutsch): Sicherheit von Maschinen - Sicherheitsbezogene Teile von Steuerungen - Teil 1: Allgemeine Gestaltungsleitsätze (ISO 13849-1:2006);

Deutsche Norm. (2008b). DIN EN ISO 9001 Qualitätsmanagementsysteme – Anforderungen.

Deutsche Norm. (2009). ISO/TS 16949:2009 Qualitätsmanagementsysteme. Besondere Anforderungen bei Anwendung von ISO 9001:2008 für die Serien- und Ersatzteilproduktion in der Automobilindustrie; *VDA*.

Deutsche Norm. (2010a). DIN EN/IEC 61508: Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme.

Deutsche Norm. (2010b). DIN EN 61226 VDE 0491-1:2010-08 Kernkraftwerke – Leittechnische Systeme mit sicherheitstechnischer Bedeutung.

Deutsche Norm. (2010c). DIN EN 62138 VDE 0491-3-3:2010-03 Kernkraftwerke – Leittechnik für Systeme mit sicherheitstechnischer Bedeutung Softwareaspekte für rechnerbasierte Systeme zur Realisierung von Funktionen der Kategorien B oder C.

Deutsche Norm. (2010d). DIN EN ISO 12100:2010 Sicherheit von Maschinen.

Deutsche Norm. (2011). DIN IEC 61513 (VDE 0491-2:2013-09) Nuclear power plants - Instrumentation and control important to safety - General requirements for systems.

Deutsche Norm. (2012). DIN EN ISO/IEC 17020:2012-07 Konformitätsbewertung - Anforderungen an den Betrieb verschiedener Typen von Stellen, die Inspektionen durchführen.

Diesterer, G., Fels, F., & Hausotter, A. (2003). *Taschenbuch der Wirtschaftsinformatik* (2. Aufl.). Fachbuchverlag Leipzig.

Dold, A., Frank, H., Gantner, T., & Pohl, M. (2012). Effiziente Bewertung der

Entwicklungsprozesse nach ISO 26262 und Automotive SPICE. In *safetronic 2012*. München: Hanser Verlag.

Dröschel, W. (Hrsg. ., & Wiemers, M. (Hrsg. . (2000). *Das V-Modell 97 - Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz*. München Wien: Oldenbourg Verlag.

EEPCA - the European Electrical Products Certification Association. (2014). *PERMANENT DOCUMENT CIG 023 Factory Inspection Report*. Brussels: ETICS European Testing Inspection Certification System.

Eisenbahn-Bundesamt. (2018). Abnahme nach § 32 EBO. Retrieved from https://www.eba.bund.de/DE/Themen/Fahrzeugzulassung/Abnahme/abnahme_node.html

Ekert, D., & Hagenmeyer, P. (2009). Development of a concept for integrating various Quality Standards. In *EuroSPI 2009* (pp. 13–24).

Esterel Technologies SAS. (2015). *SCADE Suite 17.0*. Retrieved from http://www.esterel-technologies.com/wp-content/uploads/2014/02/SCADE-Suite-Technical-Datasheet.pdf

Europäische Kommission. (2015). Amtsblatt der Europäischen Union C 054 vom 13. Februar 2015, *2009*(768).

Federal Aviation Administration. (2005). Certification Authorities Software Team (CAST) Position Paper CONSIDERATIONS WHEN USING A QUALIFIABLE DEVELOPMENT ENVIRONMENT (QDE) IN CERTIFICATION PROJECTS. *North*.

Fey, I., & Stürmer, I. (2008). Code Generation for Safety-Critical Systems –Open Questions and Possible Solutions. *SAE Int. Journal Passenger Cars - Electronic Systems 1(1)*, 150–155.

Frank, S., Grabmüller, M., Hofstedt, P., Kleeblatt, D., Pepper, P., Mai, P., & Schneider, S. (2008). Safety of Compilers and Translation Techniques - Status quo of Technology and Science. In (Hrsg.), H. B. Keller, E. Plödereder, P. Dencker, & H. Klenk (Eds.), *Automotive Safety & Security 2008*. Stuttgart: Shaker Verlag GmbH.

Friedrich, J., Kuhrmann, M., Sihlin, M., & Hammerschall, U. (2009). *Das V-Modell XT für Projektleiter und GS-Verantwortliche*. Berlin, Heidelberg: Springer Verlag.

Fukuda, J. (2012). *Comprehensive Management of the Software Development Lifecycle with PTC Integrity in Compliance with Functional Safety Standards , Improvement in Software quality* . (DENSO CORPORATION, Ed.). Tokyo: PTC Japan Inc.

Godfrey, S. (2008). *What Is CMMi?* Washington, D.C.: NASA. Retrieved from https://web.archive.org/web/20090404065835/http://software.gsfc.nasa.gov/docs/What is CMMI.ppt

Hedtke, R. (1984). *Mikroprozessorsysteme*. Berlin, Heidelberg: Springer Verlag.

Helmig, E. (2013). Produkthaftung braucht Reform. *QZ*, *01/2013*, 22–24.

Henning, K. (2016). The Digitale Transformation of Human Machine Interaction. In *5. Augsburger Technologietransfer-Kongress*. Augsburg: RWTH Aachen.

Hindel, B., & Sechser, B. (2015). Qualitätssicherung für die Software im Auto mit Automotive SPICE 3 . 0 Was bleibt anders? Richtlinien zur einheitlichen Verwendung englischer Begriffe. Retrieved February 16, 2017, from http://heise.de/-3029717

Hölscher, H., & Rader, J. (1984). *Mikrocontroller in der Sicherheitstechnik (Microcontroller in Safety)*. Verlag TÜV Rheinland.

Hühnlein, D. (2016). *BSI zertifiziert erstmals Open Source eID-Client – Zertifikat gemäß BSI TR-03124 für Open eCard App*. Michelau: Bundesamt für Sicherheit in der Informationstechnik.

Industrial Internet Consortium. (2016a). Industrial Internet Reference Architecture (IIRA) V1.8. Needham: Industrial Internet Consortium.

Industrial Internet Consortium. (2016b). Industrial Internet Security Framework (IISF) V0.17.

Needham: Industrial Internet Consortium.

International Standard. (1985). DO-178A, Software Considerations in Airborne Systems and Equipment Certification. *RTCA, EUROCAE*.

International Standard. (1987). *DIN VDE 31000 Teil 2:1987-12: Allgemeine Leitsätze für das sicherheitsgerechte Gestalten technischer Erzeugnisse – Begriffe der Sicherheitstechnik – Grundbegriffe.*

International Standard. (1992). DO-178B, Software Considerations in Airborne Systems and Equipment Certification. *RTCA, EUROCAR*.

International Standard. (2004). ISO/IEC 15504 Information technology – Process assessment, also termed Software Process Improvement and Capability Determination (SPICE).

International Standard. (2008). *ISO/IEC 12207 Systems and software engineering — Software life cycle processes*.

International Standard. (2011a). DO-330, Software Tool Qualification Considerations. RTCA, EUROCAE.

International Standard. (2011b). ISO 26262:2011 Road vehicles - Functional safety.

International Standard. (2012). DO-178C, Software Considerations in Airborne Systems and Equipment Certification. *RTCA, EUROCAE*.

International Standard. (2015a). ISO/IEC 33001:2015 Information technology -- Process assessment -- Concepts and terminologyNo Title.

International Standard. (2015b). ISO/IEC 33002:2015 Information technology -- Process assessment -- Requirements for performing process assessment.

International Standard. (2015c). ISO/IEC 33003:2015 Information technology -- Process assessment -- Requirements for process measurement frameworks.

International Standard. (2015d). ISO/IEC 33004:2015 Information technology -- Process assessment -- Requirements for process reference, process assessment and maturity models.

International Standard. (2015e). ISO/IEC 33014:2015 Information technology -- Process assessment -- Guide for process improvement.

International Standard. (2015f). ISO/IEC 33020:2015 Information technology -- Process assessment -- Process measurement framework for assessment of process capability.

International Standard. (2015g). ISO/IEC 33063:2015 Information technology -- Process assessment -- Process assessment model for software testing.

IT-Beauftragte der Bundesregierung. (2006). *V-Modell ® XT V1.3*. Retrieved from http://www.v-modell-xt.de/

Klindt, T. (2012). Rechtlicher Überblick : Überblick Produkthaftung , Produktsicherheit und Produktrückrufe. In *safe.tech 2012*. München: TÜV SÜD Akademie GmbH.

König, F., Pappler, N., & Wild, D. (2013). Kombinierte SPICE / Safety Begutachtung, ein Erfahrungsbericht. München: TÜV SÜD Akademie GmbH.

Kornecki, A., & Zalewski, J. (2006). The Qualification of Software Development Tools From the DO-178B Certification Perspective. *The Journal of Defense Software Engineering*.

Krause, L. (2008). *Methode zur Implementierung von integriertem Produktdatenmanagement (PDM)* (2. durchge). Berlin: GITO-Verlag. Retrieved from https://books.google.de/books?id=kzq9V1A3s1cC&printsec=frontcover&hl=de#v=onepage&q&f=false

Lami, G., Fabbrini, F., & Buglione, L. (2014). An ISO/IEC 33000-compliant Measurement Framework for Software Process Sustainability Assessment. In *24°International Workshop on Software Measurement (IWSM) and 9th International Conference on Software Process and Product Measurement (MENSURA)*.

Larus, J. (1997). *SPIM S20: A MIPS R2000 Simulator*. Madison: Computer Sciences Department University of Wisconsin-Madison.

Larus, J. (2018). SPIM: A MIPS32 Simulator. Retrieved from http://spimsimulator.sourceforge.net

Li, X., Liu, Z., & Yi, W. (2015). *Dependable Software Engineering: Theories, Tools, and Applications*. Cham, Heidelberg,New York, Dordrecht, London: Springer International Publishing Switzerland.

Messnarz, R., Habel, S., & Ross, H.-L. (2010). Integrated Approach for Automotive SPICE and ISO 26262 Assessments. In *EuroSPI 2009 Conference, Wiley SPIP Journal*.

Messnarz, R., Sokic, I., Habel, S., König, F., & Bachmann, O. (2011). Extending Automotive SPICE to cover functional safety requirements and a safety architecture. *Communications in Computer and Information Science*, *172*, 298–307. http://doi.org/10.1007/978-3-642-22206-1

Miller, B., & Lin, S.-W. (2016). Interoperability and the IIRA. Retrieved from https://www.researchgate.net/publication/281971976_Interoperability_and_the_IIRA

Motor Industry Software Reliability Association. (2007). *MISRA AC TL: Modelling style guidelines for the application of TargetLink in the context of automatic code generation*. Warwickshire: HORIBA MIRA Ltd.

Nonnengart, A., Rock, G., & Stephan, W. (2003). VSE - Verification Support Environment. Saarbrücken: German Research Center for Artificial Intelligence.

Object Management Group. (2015). OMG Unified Modeling Language TM (OMG UML), Superstructure v.2.3. Needham: Object Management Group. http://doi.org/10.1007/s002870050092

OSADL eG. (2017). OSADL Project: Safety Critical Linux. Retrieved October 22, 2017, from https://www.osadl.org/Safety-Critical-Linux.safety-critical-linux.0.html

OSADL eG. (2018). Open Source Automation Development Lab eG. Retrieved February 24, 2018, from https://www.osadl.org

Pickert, K., & Kohlschein, R. (2005). Leitfaden zur Anwendung der EN 50128 auf Schienenfahrzeugen. Bonn: Eisenbahn-Bundesamt.

Pothon, F., Pomies, L., Comar, C., & Brosgol, B. (2013). DO-330 / ED-215 Benefits of the New Tool Qualification Document.

PTC. (2013). PTC Continues Commitment to Automotive Industry with Enhanced Safety Standard Certifications. Retrieved February 20, 2017, from http://investor.ptc.com/releasedetail.cfm?releaseid=754900

PTC. (2015). PTC Integrity ™ Automotive Software - Data Sheet. Needham: PTC Inc.

Puntambekar, A. A. (2008). *Data Structures and Files*. Pune: Technical Publications Pune.

Reinelt, W., Mishr, A., & Breault, R. (2017). Sicher agil – agil sicher! In *safe.tech 2017*. München: TÜV SÜD Akademie GmbH.

Riedl, M. (2002). *Proseminar Software Desaster ARIANE 5 Absturz des Flugs 501*. Retrieved from http://www4.in.tum.de/lehre/seminare/ps/WS0203/desaster/Riedl-Arianne5-Ausarbeitung-27-11-02.pdf

Ross, H.-L. (2014). *Funktionale Sicherheit im Automobil - ISO 26262, Systemengineering auf Basis eines Sicherheitslebenszyklus und bewährten Managementsystemen*. München Wien: Carl Hanser Fachbuchverlag.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131–164. http://doi.org/10.1007/s10664-008-9102-8

Sane, S. S., & Deshpande, N. A. (2006). *Data Structures and Algorithms*. Pune: Technical Publications Pune.

Schenck, D., & Wilson, P. (1994). *Information Modelling the EXPRESS way*. New York, Oxford: Oxford University Press, Inc.

Schilke, M. (2010). *Einsatz von Produktdatenmanagement-Systemen im Sondermaschinenbau für die Automobilindustrie*. Saarbrücken: UNIVERSITÄT DES SAARLANDES SCHRIFTENREIHE.

Schneider, S., & Mai, P. R. (2009). The Validation Suite Approach to Safety Qualification of Tools. SAE. http://doi.org/10.4271/2009-01-0746

Schneider, U., & Werner, D. (2001). *Taschenbuch der Informatik* (4.Aufl.). Fachbuchverlag Leipzig.

Schumann, J. (2001). *Automated Theorem Proving in Software Engineering*. Berlin, Heidelberg: Springer Verlag.

Siemens AG. (2016). Safety - Nachhaltigkeit - Siemens Global Website. Retrieved January 5, 2017, from http://www.siemens.com/global/de/home/unternehmen/nachhaltigkeit/safety.html

Slotosch, O. (2012). Model-based tool qualification the roadmap of eclipse towards tool qualification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 7991 LNCS, pp. 215–228). Berlin, Heidelberg: Springer Verlag. http://doi.org/10.1007/978-3-642-54338-8_18

Slotosch, O., & Reiling, M. (2012). Applying ISO26262 to Tool Qualification Content : Tool Chain Analysis. In *Embedded World 2012* (pp. 1–18). Nürnberg: Validas AG.

Steindl, M., Mottok, J., Meier, H., Schiller, F., & Fruechtl, M. (2009). Migration of SES to FPGA Based Architectural Concepts. *Softwaretechnik Trends, Gesellschaft Für Informatik*.

Stürmer, I. (2006). *Systematic Testing of Code Generation Tools*. Pro Business GmbH, Berlin.

Tanenbaum, A. S. (2009). *Moderne Betriebssysteme* (3. Auflage). Hallbergmoos: Pearson Deutschland GmbH.

Tanenbaum, A. S., & Van Steen, M. (2006). *Distributed Systems: Principles and Paradigms* (Ed. 2). London: Pearson Education Inc.

Texas Instruments. (2013). SafeTI - Compiler Qualification Kit. Retrieved February 18, 2017, from http://processors.wiki.ti.com/index.php/Compiler_Qualification_Kit

Texas Instruments, & Validas AG. (2013). TI Compiler Qualification Kit. Retrieved February 18, 2017, from http://processors.wiki.ti.com/index.php/Compiler_Qualification_Kit

TU München et al. (2012). SPES 2020 - Software Plattform Embedded Systems. Retrieved January 14, 2017, from http://spes2020.informatik.tu-muenchen.de/partnerTUMRes.html

Turban, B., Kucera, M., Tsakpinis, A., & Wolff, C. (2009). Bridging the requirements to design traceability gap. In *Lecture Notes in Electrical Engineering* (Vol. 38, pp. 275–288). http://doi.org/10.1007/978-1-4020-9823-9_20

TÜV SÜD AG. (2015). *Geschäftsbericht 2015*. Retrieved from https://www.tuev-sued.de/uploads/images/1460453018884122621074/2015-tuev-sued-gb-einzelseiten.pdf

TÜV SÜD AG. (2016). Prüf- und Zertifizierordnung TÜV SÜD Gruppe. München: TÜV SÜD AG. Retrieved from https://www.tuev-sued.de/uploads/images/1452669366751122120712/pzo-konzern-deutsch-ab1.pdf

VDA. (2012). Quality Management in the Automotive Industry - Automotive SPICE ®: Process Assessment Model. Berlin: Verband der Automobilindustrie.

VDA. (2015). Automotive Spice - Process Reference Model / Process Assessment Model 3.0. Berlin: Verband der Automobilindustrie. Retrieved from http://vda-qmc.de/software-

prozesse/automotive-spice/

WindRiver. (2013). WIND RIVER DIAB COMPILER ISO 26262 QUALIFICATION KIT. Retrieved from http://www.embedded-tools.de/sites/default/files/Wind_Rvier_Diab_Compiler_ISO_26262_Qualification_Kit.pdf

World Nuclear Association. (2015). Safety Classification for I&C Systems in Nuclear Power Plants -Current Status & Difficulties Title: Safety Classification for I&C Systems in Nuclear Power Plants - Current Status & Difficulties. London: World Nuclear Association.

Yin, R. K. (2003). *Case study research. Design and methods*. Thousand Oaks: SAGE Publications Inc.

Anhang A: Liste der Arbeitsprodukte (Work Products) nach ISO 26262

	Document / work product	**Created in ISO 26262**	**Influenced by or refined in**
FSM1	Organization-specific rules and processes for functional safety	Part2-5	
FSM2	Evidence of competence	Part2-5	
FSM3	Evidence of quality management	Part2-5	
FSM4	Safety plan	Part2-6	Part6-5 Part6-7 Part6-Annex C Part8-12
FSM5	Project plan (refined)	not safety specific	Part2-6
FSM6	Safety case	Part2-6	
FSM7	Functional safety assessment plan	Part2-6	
FSM8	Confirmation measure reports	Part2-6	
FSM9	Evidence of field monitoring	Part2-7	
SW1	Software verification plan	Part6-5	Part6-6 Part6-9 Part6-10 Part6-11 Part6-Annex C Part8-9
SW2	Design and coding guidelines for modelling and programming languages	Part6-5	
SW3	Tool application guidelines	Part6-5	
SW4	Software safety requirements specification	Part6-6	Part6-7 Part8-6
SW5	Hardware-software interface specification (refined)		Part6-6

	Document / work product	**Created in ISO 26262**	**Influenced by or refined in**
SW6	Software verification report	Part6-6	Part6-7 Part6-8 Part6-9 Part6-10 Part6-11 Part6-Annex C Part8-9
SW7	Software architectural design specification	Part6-7	Part9-5
SW8	Safety analysis report	Part6-7	Part9-8
SW9	Dependent failures analysis report	Part6-7	Part9-6 Part9-7
SW10	Software unit design specification	Part6-8	
SW11	Software unit implementation	Part6-8	
SW12	Software verification specification	Part6-9	Part6-10 Part6-11 Part6-Annex C Part8-9
SW13	Embedded software	Part6-10	
SW14	Configuration data specification	Part6-Annex C	
SW15	Calibration data specification	Part6-Annex C	
SW16	Configuration data	Part6-Annex C	
SW17	Calibration data	Part6-Annex C	
SP5-1	Supplier selection report	Part8-5	
SP5-2	Development interface agreement (DIA)	Part8-5	
SP5-3	Supplier's project plan	Part8-5	
SP5-4	Supplier's safety plan	Part8-5	
SP5-5	Safety assessment report	Part8-5	
SP5-6	Supply agreement	Part8-5	
SP7-1	Configuration management plan	Part8-7	
SP8-1	Change management plan	Part8-8	
SP8-2	Change request	Part8-8	
SP8-3	Impact analysis and change request plan	Part8-8	
SP8-4	Change report	Part8-8	

	Document / work product	**Created in ISO 26262**	**Influenced by or refined in**
SP10-1	Document management plan	Part8-10	
SP10-2	Documentation guideline requirements	Part8-10	
SP11-1	Software tool criteria evaluation report	Part8-11	
SP11-2	Software tool qualification report	Part8-11	
SP12-1	Software component documentation	Part8-12	
SP12-2	Software component qualification report	Part8-12	

Anhang B: Tool-Assessment-Checkliste

Certification of Software Tools - Examined Process Areas

Summary

This document lists all requirements that are the basis for a successful assessment of the development process including all supporting processes of a SW-Tool development, the validation, and the customer information.

Contents

	Author	Version	Template Creator/Review	Template Version
			ab	V 2.0
	Applied Project	File Name	Project Number	Page
				1

1 Introduction

One qualification method derived from ISO 26262 – part 8, clause 11 "Confidence in the use of software tools", is the evaluation of the development process for the software tool. This qualification method requires that the development process applied for the development of the software tool shall comply with an appropriate national or international standard (ISO 26262 – part 8, 11.4.8.2). The evaluation of this development process applied for the development of the software tool shall be provided by an assessment and the proper application of the assessed development process shall be demonstrated (ISO 26262 – part 8, 11.4.8.2). This checklist provides an assessment model based on the ISO 26262 itself.
The next table shows the basic structure for the requirements. The list of requirements defines a conjunctive set that needs to be fulfilled.

Project management (ISO 26262 – part 2)
Requirements
R1. Competence management - People responsible for the development of the tool should have a sufficient level of skills, competences and qualifications corresponding to their responsibilities. …

To check the requirements special questions are contained in each section. These questions serve as a checklist and must be answered. The following table gives an example and demonstrated how the checklist is used.

Name(s) of document(s): Put here the names of documents that serve as the work product	
Judgement	***Topic:***
[ok|pok|nok]	Here the question is asked
	Descriptions and Comment: Note here information and descriptions given by the customer and additional comments.
General comments: Any additional information that helps understanding the process but does not fit to none of the questions can be written here	

The checklist has areas to put information and explanations given by the customer. Finally, a judgement must be given. The customer's activities, methods and means can either be

- sufficient: put "OK" in the judgement,
- partially sufficient: put "POK" in the judgement and describe the deviation in the comment
- not ok: put "nok" in the judgement and give the rationale in the comment

Open issues shall be collected as ToDo's in the respective sections to track their progress. The final judgement at the end of each part gives the result at the end of the assessment process and describes possible obligations for the next (re-)certification.

Within the next paragraphs the specific requirements will be presented. They were derived from the applicable requirements of the ISO 26262 standard. This does not mean that the development process applied for the development of the software tool must be stringently compliant with the V model which is demanded by the ISO 26262 standard. Any other sophisticated process model, as e.g. the agile SCRUM Process model can be applied, if the requirements stated in this section are fulfilled.

The applicable requirements, as taken from the ISO 26262 standard, are adapted to the characteristics of the development of a software tool. The source of the particular requirement is stated in the respective subtitles.

2 Documents, provided by the vendor

The following documents were delivered in advance to the on-site assessment and after the meetings to address findings.

No.	Title/Description	Rev.	Date
General Documents			
[D1]	Product Engineering Process (PEP)		
Project Management Artifacts			
[D2]	Project Card		
[D3]	Project Contract Addendum		
[D4]	Contract		
[D5]	Configuration Management Plan		
[D6]	Project Management Plan		
[D7]	Release Plan		
[D8]	Backlog (containing Risks and Impediments) (Sample)		
[D9]	Definition of Done		
[D10]	Baseline Audits (QA2)		
[D11]	Quality Goals & Release Criteria		
[D12]	Project Reporting (Sample)		
[D13]	Orientation Plan of new employee		
[D14]	Release Meeting Email		
[D15]	Quality Assessment Log		
Artifacts of Development Process			
[D16]	Safety Case		
[D17]	Hazard&Risk Analysis		
[D18]	Safety_Advise		
[D19]	System Test Strategy		
[D20]	System Test Results		
[D21]	System Test Progress Results		
[D22]	Verification Plan		
[D23]	Low Level and Product Integration Test Strategy		
[D24]	Developer Test Result		
[D25]	Static code analysis report		
[D26]	Beta Test Strategy		

Product Related Documents and Artifacts			
[D27]	Release		

	Author	Version	Template Creator/Review	Template Version
			ab	V 2.0
	Applied Project	File Name	Project Number	Page
				6

3 Process areas to cover (ISO 26262 – part 8, 11.4.8)

3.1 Project management

3.1.1 Requirements

The following table lists the general requirements for this part. In section "plan for the assessment" more specific questions address various aspects and guide through the assessment. Any additional information that is useful for understanding the customer's approach can be recorded in the "General comments" field below the assessment questions.

Project management (ISO 26262 – part 2)
Requirements

R1. Competence management
- People responsible for the development of the tool should have a sufficient level of skills, competences and qualifications corresponding to their responsibilities.

R2. Roles and responsibilities
- A project manager shall be appointed

R3. Planning and coordination/project plan
- A project plan (corresponding to "Safety Plan" in ISO26262) should be set up and maintained for every release version
- Contents of the project plan:
  - Identification of the product release
  - Scope of the product release
  - Planning of the development activities including methods and tools
  - Planning of the supporting processes
  - Planning of the verification and validation activities (test plan, reviews etc.)
  - Planning of the required resources
- The project plan should be available for every involved person
- The project plan should be reviewed
- The project manager (or an instructed person) should plan, coordinate and monitor all activities listed in the project plan (→ project status)

R4. Suitable tailoring of the development activities
- The activities may be tailored (according to the applied process model)

R5. Software development handbook (or the like)
- Methods, tools and their application guidelines

R6. Release planning
- Numbering, classification (intern, for customer for testing, official release, …), and schedule
- Functional content of every release
- Definition of criteria for the product release

3.1.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
	Project Plan
	Development Handbook / Process description
	Project Card
	Samples for Monthly Project Reporting
	CM Plan
	Orientation Plan of new employee
	Project Contract
	Quality Assessment Log (QA3)

3.1.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it "ok". If it is not plausible, judge it "nok" and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture their names in the documents list above. Record any additional information that is useful for understanding the customer's approach in the field "**General comments"**

3.1.3.1 Work Product: Project plan / safety plan / related plans (PP)

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there a template for the project plan (put name of documents in documents list)?	#01
OK	Are people responsible for the development listed in the PP?	#02
OK	Is there evidence for sufficient level of skills, competences and qualifications corresponding to their responsibilities?	#03
OK	A project manager (Name!)	#04
OK	Contents of the project plan: - Identification of the product release - Scope of the product release - Planning of the development activities including methods and tools - Planning of the supporting processes - Planning of the verification and validation activities (test plan, reviews etc.)	#05

	- Planning of the required resources	
OK	Is the plan available for every involved person?	#06
OK	Is the project plan reviewed?	#07

General comments:
-

3.1.3.2 Work Product: Project state monitoring

Name(s) of document(s):

Judgement	***Topic***	***Reference***
OK	Are there regular meetings to retrieve the project state?	#08
OK	Is the project state recorded (put name of document in documents list)?	#09
OK	What are the actions if the state deviates from the plan? Are there corrective actions?	#10
OK	Are there means to rate the performance of the project? Here especially performance of safety related activities (reviews, tests)	#11

General comments:
-

3.1.3.3 Work Product: Project Handbook (PH)

Name(s) of document(s):

Judgement	***Topic***	***Reference***
OK	Does the PH describe the relevant phases (requirements engineering, design, coding, testing)	#12
OK	Does the PH describe methods for each of the respective phases?	#13
OK	Do the selected methods match i.e. are they seamless?	#14
OK	Are tools listed that are used during development?	#15
OK	Are there guidelines for using tools?	#16

General comments:
-

3.1.3.4 Work Product: Release planning

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there a numbering scheme uniquely identifying the release?	#17
OK	Is there a classification of revisions (internal, for customer for testing, official release, …) and their schedule?	#18
OK	Is the content (scope) of each release defined (which functionality Ids are contained in a release)?	#19
OK	Is there a definition of criteria for the product release (definition of complete)?	#20
OK	Is the release plan approved?	#21
OK	Is the release approved?	#22

General comments:
-

3.1.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: Project plan
TODOs: -

Work Product: Development Handbook
TODOs: -

Work Product: Project state monitoring
TODOs: -

Work Product: Release Planning
TODOs: -

3.1.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

	Obligations for next assessment	Judgement
Work Product: Project plan	**None**	**OK**
Work Product: Development Handbook	**None**	**OK**
Work Product: Project state monitoring	**None**	**OK**
Work product: Release planning	**None**	**OK**

3.2 Quality Management (ISO 26262 – Part 2)

3.2.1 Requirements

The following table lists the general requirements for this part. In section "plan for the assessment" more specific questions address various aspects and guide through the assessment. Any additional information that is useful for understanding the customer's approach can be recorded in the "General comments" field below the assessment questions.

Quality management
Requirements
R7. Quality assurance - Quality assurance strategy: Planning of all quality assurance measures, responsibilities, required resources - Escalation process - Independency There should be something like ISO 9001/CMMI or equivalent. At least the tool manufacturer should aim for it. Note: In this document the required quality assurance activities are listed in several sections.

3.2.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
	Quality Goals & Release Criteria
	Quality Assessment Log
	Project Card
	Samples for Monthly Project Reporting
	Baseline Audits

3.2.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it "ok". If it is not plausible, judge it "nok" and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture their names in the documents list above. Record any additional information that is useful for understanding the customer's approach in the field "**General comments"**

3.2.3.1 Work Product: Quality assurance strategy

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there a plan for the QM (can be part of the PP)?	#23
OK	Does the plan at least consider one process application audit? At which phase, how often if multiple?	#24
OK	Is the QA included in the PP?	#25
OK	Does the QM plan consider a retrospective (lessons learned)?	#26
OK	Are the responsibilities for QA assigned?	#27
OK	Is there an escalation process?	#28
OK	Is the QA independent (at least QA must not be executed by the person that executed the activity under consideration)?	#29
OK	Is there a QM certificate (ISO 9001/CMMI) (Not mandatory!)?	#30
OK	Is the QA executed (Who, when, what)?	#31
OK	Is the result of the QA recorded, are deviations tracked and removed?	#32
General comments: -		

3.2.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: QM strategy
TODOs: -

3.2.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

	Obligations for next assessment	Judgement
Work product: Quality assurance strategy	**None**	**OK**

3.3 Requirements management and release planning (ISO 26262 – part 8)

3.3.1 Requirements

The following table lists the general requirements for this part. In section "plan for the assessment" more specific questions address various aspects and guide through the assessment. Any additional information that is useful for understanding the customer's approach can be recorded in the "General comments" field below the assessment questions.

Requirements management and release planning
Requirements
R8. High level system description/high level functional requirements list - purpose and functionality - system boundaries and interfaces - any frame-conditions for operation **R9.** Specification of configurations (in case only) **R10.** Specification and management of requirements - A requirements management process should be installed and should be supported by a adequate tool - Requirements should have the following characteristics: o Hierarchical structure and traceability between the hierarchical levels o Consistency o Unambiguous and comprehensible o Verifiable (→ relation to a test case) o Unique ID, status - Dependencies to/from other requirements

3.3.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
	Process description
	Project Contract

	Consistency Check

3.3.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it "ok". If it is not plausible, judge it "nok" and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture their names in the documents list above. Record any additional information that is useful for understanding the customer's approach in the field "**General comments"**

3.3.3.1 Work Product: Requirements management process description

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there a requirements management process described?	#33
OK	Is there a definition of a requirement?	#34
OK	If applicable: Is there a definition of a feature (use case, function, user story) in contrast to a requirement?	#35
OK	Is there a tool that supports the requirements management?	#36
OK	Does the tool support the (bi-directional) tracing between requirements?	#37
OK	Is there a guideline or template for phrasing requirements (e.g. IEEE 1233-1998)?	#38
OK	Does the requirements process include a change management to allow propagation of changes due to technical issues?	#39
OK	Are there guidelines for using tools?	#40
OK	Are there guidelines for using methods (if applicable and different from question about phrasing requirements)?	#41
General comments: -		

3.3.3.2 Work Product: High level system description (HLSD)

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Does the HLSD describe the purpose of the functions?	#42

OK	Does the HLSD describe the limitations of the function (conditions of use)?	#43
OK	Is the function uniquely identified?	#44
OK	Are there a number of test cases identified with each of the functions?	#45
OK	Is there a relation to the functions described in the safety manual?	#46
OK	Is there an agreed set of functions for a release?	#47
OK	Is the range of functions baselined?	#48
General comments: -		

3.3.3.3 Work Product: Requirements specification

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Does every requirement have a source (higher level requirement or HLSD) i.e. are requirements derived from HLSD?	#49
OK	Does every requirement have a child (lower level requirement or element of architecture?)	#50
OK	Is every requirement associated with a (set of) test(s) (i.e. are requirements testable)?	#51
OK	Are all requirements uniquely identified?	#52
OK	Are possible dependencies to other requirements recorded (needed e.g. for integration or sprint planning)?	#53
OK	Is the linking between requirements established accordingly?	#54
OK	Is there an agreed set of requirements derived from the functional descriptions?	#55
OK	Are safety related requirements (relate to risk analysis in safety manual) categorized as such? If not yet, is it possible to establish the like?	#56

OK	Are changes that have impact on HLSD propagated and communicated?	#57
OK	Are requirements baselined?	#58
OK	Are there test cases identified with each of the requirements?	#59

General comments:
-

3.3.3.4 Work Product: Sample checks of requirements

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Are requirements understandable?	#60
OK	Are requirements comprehensive?	#61
OK	Are requirements unambiguous?	#62
OK	Is the traceability of requirements verified in a review process?	#63
OK	Are the requirements complete? Check for contents of its children or the combination of its siblings in order to completely cover the parent.	#64

General comments:
-

3.3.3.5 Work Product: Requirement verification specification and report

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Was the review planned and specified?	#65
OK	Does the requirement review cover all requirements (at least the new ones for a release)? How is this checked?	#66
OK	Does the review address the unambiguousness of the requirements?	#67
OK	Does the review address the comprehensibility of the requirements?	#68
OK	Does the review address the completeness of the requirements?	#69
OK	Does the review address the linking between requirements?	#70

OK	Does the review address the availability of adequate tests?	#71
OK	*Has the verification been performed on an already stable version of the requirements?*	#72
General comments: -		

3.3.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: Requirements management process description
TODOs: -

Work Product: High level system specification
TODOs: -

Work Product: Requirements specification
TODOs: -

Work Product: Requirements verification (+ report)
TODOs: -

3.3.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

	Obligations for next assessment	Judgement
Work Product: RM process description	**None**	**OK**
Work Product: High level system description	**None**	**OK**
Work Product: Requirements specification	**None**	**OK**
Work Product: Sample check of requirements	**None**	**OK**
Work product: Requirements verification specification and report	**None**	**OK**

3.4 Software architectural design/high level software design (ISO 26262 – part 6)

3.4.1 Requirements

The following table lists the general requirements for this part. In section "plan for the assessment" more specific questions address various aspect and guide through the assessment. Any additional information that is useful for understanding the customer's approach can be recorded in the "General comments" field below the assessment questions.

Software architectural design/high level software design
Requirements
R11. Software architectural design/high level software design - Allocation of the high level functional requirements to the components - Specification for the main components, their functionality and their interfaces - Refined software requirements list

3.4.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
	Architecture Document

3.4.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it "ok". If it is not plausible, judge it "nok" and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture their names in the documents list above. Record any additional information that is useful for understanding the customer's approach in the field "**General comments"**

3.4.3.1 Work Product: Architecture management process description

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there an architecture management process described?	#73
OK	Is there a tool that supports the architecture description?	#74
OK	Does the tool support the (bi-directional) tracing between requirements and architecture?	#75
OK	Does the architecture process include a change management to allow propagation of changes due to technical issues?	#76

OK	Are there guidelines for using tools?	#77
OK	Are there guidelines for using methods?	#78

General comments:
-

3.4.3.2 Work Product: Software architecture

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Is there a high-level SW architecture description?	#79
OK	How is the architecture derived from the requirements (expert judgement)?	#80
OK	Are requirements allocated to elements of the architecture (does this enable an impact analysis)?	#81
OK	Does the architecture describe the static structure of the SW?	#82
OK	Are interfaces of the elements of the architecture defined (names, types, ranges etc.)?	#83
OK	Does the architecture describe the dynamic aspects of the elements?	#84
OK	Does the architecture refine the design to an atomic level (unit, module, function, class etc.?)? Are those related to code level elements?	#85
OK	Is it possible to identify the atomic level elements (unit, module, function, class etc.) that need to be changed to implement a new requirement?	#86
OK	Are there test cases identified which address the interaction between components?	#87

General comments:
-

3.4.3.3 Work Product: Sample checks of architecture

Name(s) of document(s):

Judgement	*Topic*	*Reference*

OK	Are descriptions of dynamic aspects of the architectural elements understandable?	#88
OK	Is the static structure comprehensible?	#89
OK	Are interfaces of architecture elements well defined?	#90
OK	Is it possible to identify related requirements?	#91
OK	Does the architectural element (possibly by involving others) satisfy one super-ordinate requirement as an example?	#92

General comments:
-

3.4.3.4 Work Product: Architecture verification specification and report

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Are verification criteria for the architecture defined?	#93
OK	Are dynamic aspects of the architecture checked to satisfy their requirements?	#94
OK	Are all requirements checked to be implemented in the architecture?	#95
OK	Are interfaces checked for consistency?	#96
OK	Does the review address the linking between requirements and architecture?	#97
OK	Does the review address the linking between architecture and atomic level elements?	#98
OK	Does the review address the availability of adequate tests?	#99

General comments:
-

3.4.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible

to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: Software architecture process description
TODOs: -

Work Product: Software architecture
TODOs: -

Work Product: Verification specification and report for architecture
TODOs: -

3.4.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

Objective	Obligations for next assessment	Judgement
Work product: Architecture management process description	**None**	**OK**
Work Product: Software Architecture	**None**	**OK**
Work Product: Sample Check of Architecture	**None**	**OK**
Work Product: Architecture verification specification and report	**None**	**OK**

3.5 Implementation

3.5.1 Requirements

The following table lists the general requirements for this part. In section “plan for the assessment” more specific questions address various aspects and guide through the assessment. Any additional information that is useful for understanding the customer’s approach can be recorded in the “General comments” field below the assessment questions.

Implementation
Requirements
In general, Software unit design and implementation from the ISO26262 Part 6 section 8 shall be covered

3.5.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
	Configuration Management Plan

3.5.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it "ok". If it is not plausible, judge it "nok" and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture their names in the documents list above. Record any additional information that is useful for understanding the customer's approach in the field "**General comments"**

3.5.3.1 Work Product: Coding guidelines

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is a subset of the programming language used (not mandatory)?	#100
OK	Is there a naming convention for identifiers?	#101
OK	Are design patterns defined (not mandatory)?	#102
OK	Are there guidelines for undesired constructions?	#103
OK	Is there a guideline for code documentation?	#104
General comments: -		

3.5.3.2 Work Product: Source code

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there an identification of source code level modules with the architecture?	#105
OK	Is the source code under version control?	#106
OK	Is the version control of the source code integrated with the configuration management of other documents?	#107
General comments: -		

3.5.3.3 Work Product: Source code verification specification and report

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Does the code review address the functional correctness of the code?	#108

OK	Does the review address the interfaces of the atomic level elements?	#109
OK	Does the review address coding and code documentation guidelines?	#110
OK	Are code metrics measured (not mandatory for a tool)?	#111
General comments: -		

3.5.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: Coding guidelines
TODOs: -

Work Product: Source code
TODOs: -

Work Product: Source code verification specification and report
TODOs: -

3.5.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

Objective	Obligations for next assessment	Judgement
Work Product: Coding guidelines	**None**	**OK**
Work Product: Source code	**None**	**OK**
Work Product: Source code verification specification and report	**None**	**OK**

3.6 Testing (ISO 26262 – part 6)

3.6.1 Requirements

The following table lists the general requirements for this part. In section “plan for the assessment” more specific questions address various aspects and guide through the assessment. Any additional information that is useful for understanding the customer’s approach can be recorded in the “General comments” field below the assessment questions.

Testing
Requirements
In general, Software unit design and implementation from the ISO26262 Part 6 section 9 and 10 shall be covered **R12.** Integration and testing plan **R13.** Test specification **R14.** Test reports **R15.** Regression test **R16.** Release for production report

3.6.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
	Process description
	Developer Test Results
	System Test Results

3.6.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it "ok". If it is not plausible, judge it "nok" and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture their names in the documents list above. Record any additional information that is useful for understanding the customer's approach in the field "**General comments"**

3.6.3.1 Work Product: Module testing specification and report

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there a test strategy that describes the test strategy and classes of tests?	#112
OK	Is there a test specification that unambiguously describes the pass and fail criteria for each test?	#113
OK	Are Test cases identified with units?	#114
OK	Is there a systematic way to derive module tests? At least for every function in a module a test shall exist.	#115
OK	Are there positive and negative tests?	#116
OK	Is there a notion of coverage for tests?	#117

OK	Is there a systematic way (tool) to record the test results?	#118
OK	Are the test results evaluated?	#119
OK	Is there a strategy if tests fail?	#120
OK	Are test specifications reviewed to be appropriate?	#121
OK	Is there a test strategy that describes the test strategy and classes of tests?	#122

General comments:
-

3.6.3.2 Work Product: Integration plan

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Is the integration plan derived from the architecture?	#123
OK	Does the integration plan consider different stages of integration (not mandatory)?	#124

General comments:
-

3.6.3.3 Work Product: Integration testing specification and report

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Is there a test specification that describes the test strategy and classes of tests?	#125
OK	Is there a test specification that unambiguously describes the pass and fail criteria for each test?	#126
OK	Is there a systematic way to derive integration tests?	#127
OK	Are there tests that focus on the interfaces between modules or components?	#128
OK	Are test identified with higher level components?	#129
OK	Is there a systematic way (tool) to record the test results?	#130

OK	Are the test results evaluated?	#131
OK	Is there a strategy if tests fail?	#132
OK	Are test specifications reviewed to be appropriate?	#133
OK	Are there guidelines on how to derive tests?	#134

General comments:

3.6.3.4 Work Product: Regression test strategy

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Are regression tests planned to assure the healthiness of "old" functionality in a new release?	#135
OK	Are regression tests planned to assure the healthiness of "new" (already implemented) functionality after changes during new releases?	#136
OK	Is there a comprehensive measure for the completeness of regression tests?	#137
OK	Are all regression tests executed before release?	#138

General comments:
-

3.6.3.5 Work Product: Release for production report

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Is there a report that definitively states the products fitness for release?	#139
OK	Are test reports evaluated?	#140
OK	Is the release report approved and signed by a product manager?	#141

General comments:

3.6.3.6 Work Product: Validation of functionality described in the safety manual

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Are functions described in the safety manual identified in the software specification?	#142
OK	Are tests identified that address the functions of the safety manual?	#143
OK	Do the tests cover the main functionality?	#144
OK	Do the tests cover exceptional conditions?	#145
OK	Do the tests comprise all conditions that are listed in the safety manual?	#146
OK	Is the operational environment controlled (are common combinations considered and systematically tested)?	#147
OK	Are all operational environments described in the safety manual considered?	#148
General comments:		

3.6.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: Module test specification and report
TODOs: -

Work Product: Integration plan
TODOs: -

Work Product: Integration testing specification and report
TODOs: -

Work Product: Regression test strategy
TODOs: -

Work Product: Release for production report
TODOs: -

Work Product: Validation of functionality described in the safety manual
TODOs: -

3.6.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

Objective	Obligations for next assessment	Judgement
Work Product: Module testing specification and report	None	OK
Work Product: Integration plan	None	OK
Work Product: Integration testing specification and report	None	OK
Work Product: Regression test strategy	None	OK
Work Product: Release for production report	None	OK
Work Product: Validation of functionality described in the safety manual	None	OK

3.7 Bidirectional traceability

3.7.1 Requirements

The following table lists the general requirements for this part. In section “plan for the assessment” more specific questions address various aspects and guide through the assessment. Any additional information that is useful for understanding the customer’s approach can be recorded in the “General comments” field below the assessment questions.

Bidirectional traceability
Requirements
R17. Bidirectional traceability

3.7.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
-	No specific documents.

3.7.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it “ok”. If it is not plausible, judge it “nok” and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture their names in the documents list above. Record any additional information that is useful for understanding the customer’s approach in the field “**General comments”**

3.7.3.1 Work Product: Traceability Matrix

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Pick an example from the safety manual and follow the tracing from the feature down to code level. Is downward tracing seamless possible? Is upward tracing seamless possible?	#149
OK	For each level of abstraction (feature, requirement, architecture, module, code) follow the trace to the test cases. Are they bidirectional traceable?	#150
OK	Describe here which method has been used to establish the traceability. Is the method appropriate (i.e. does it scale, is it maintainable)?	#151
General comments: -		

3.7.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: Traceability Matrix
TODOs: -

3.7.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

Objective	Obligations for next assessment	Judgement
Work Product: Traceability matrix	**None**	**OK**

3.8 Safety/Risk Management

3.8.1 Requirements

The following table lists the general requirements for this part. In section “plan for the assessment” more specific questions address various aspects and guide through the assessment. Any additional information that is useful for understanding the customer’s approach can be recorded in the “General comments” field below the assessment questions.

Safety/Risk management
Requirements
R18. Safety culture - People responsible for the development of the tool must be aware of the safety-critical errors. - The used process model should provide activities in the appropriate phases to mitigate the risk to introduce safety-critical malfunctions and there should be evidence that these activities were carried out properly.

- Communication of errors: There should be a process for reporting safety-critical errors.

R19. Considerations regarding possible malfunctions of the software tool that lead to erroneous outputs of the developed product
- Based on a “Hazard analysis & risk assessment” (in case only)

R20. Impact analysis regarding possible new malfunctions (in the case of modification)
R21. Concept for avoidance or mitigation of these malfunctions
R22. Safety analysis report regarding safety-critical errors
- (In case only) Performing a “Safety Analysis” SW architectural level

3.8.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
[D1]	Product Engineering Process (PEP)

3.8.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it “ok”. If it is not plausible, judge it “nok” and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture their names in the documents list above. Record any additional information that is useful for understanding the customer’s approach in the field “**General comments”**

3.8.3.1 Work Product: List of possible Hazards

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Have the risks been analyzed?	#152
OK	Has the completeness of the risk analysis been checked (systematic approach or review?)	#153
OK	Are risks identified with countermeasures in the safety manual or explained in the safety manual to allow users to define their own countermeasures?	#154
General comments: -		

3.8.3.2 Work Product: Impact analysis process description

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there a method defined to judge a new features’ possible impact on existing safety relevant features?	#155
OK	Is the method to judge the safety criticality of new features applied?	#156

OK	Is there a method to judge the criticality (safety relevance) of bugs?	#157
OK	Are bugs with safety impact rated and treated with high priority?	#158
OK	Are modifications due to bugs checked for their impact?	#159
OK	Are modifications due to new features checked for their impact?	#160

General comments:
-

3.8.3.3 Work Product: Escalation/Communication process description

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Is there an escalation process defined?	#161
OK	Are issues and their escalation documented?	#162

General comments:
-

3.8.3.4 Work Product: RASI Table

Name(s) of document(s):

Judgement	*Topic*	*Reference*
OK	Is there a role defined for budged planning (budged for safety related activities)?	#163
OK	Is there a role that takes responsibility for safety related activities (including approval of bug ratings)?	#164
OK	Is there a responsible role for signing of releases/ acceptance & refusal of features and bugs?	#165

General comments:
-

3.8.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: List of possible hazards
TODOs: -

Work Product: Impact analysis process description
TODOs: -

Work Product: Escalation/Communication process description
TODOs: -

Work Product: RASI Table
TODOs: -

3.8.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

Objective	Obligations for next assessment	Judgement
Work Product: List of possible hazards	**None**	**OK**
Work Product: Impact analysis process description	**None**	**OK**
Work Product: Escalation / Communication process description	**None**	**OK**
Work Product: RASI Table	**None**	**OK**

3.9 Configuration management

3.9.1 Requirements

The following table lists the general requirements for this part. In section “plan for the assessment” more specific questions address various aspects and guide through the assessment. Any additional information that is useful for understanding the customer’s approach can be recorded in the “General comments” field below the assessment questions.

Configuration management and Documentation
Requirements
R23. Configuration management - The installed configuration management process should allow the unique identification for each work product and the reproduction of each work product at each point in time - Relations and differences between current and earlier versions should be traceable - Documentation of the principles and general conditions of the creation of the work products **R24.** Documentation - available, understandable and maintainable for each involved person - entry point: list of all valid documents (or the like) should be available - formal requirements: o author o title, scope, content

- unique identification/versioning
- status (draft, reviewed, released, …)

- change history

3.9.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
	Configuration Plan

3.9.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it "ok". If it is not plausible, judge it "nok" and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture their names in the documents list above. Record any additional information that is useful for understanding the customer's approach in the field "**General comments"**

3.9.3.1 Work Product: Configuration Management Plan

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Are all development WP under configuration management?	#166
OK	Are all documents under CM identified?	#167
OK	Does the CM-Plan describe each document under CM?	#168
OK	Does the CM-Plan list the latest valid versions of each document?	#169
OK	Does the CM describe each change and its responsible person?	#170
OK	Does the CM describe rights? (Who is allowed to edit/approve documents)?	#171
OK	Is the CM supported by using a tool? If yes, which tool?	#172
OK	Does the CM-Plan describe Baselines that relate compatible versions of WPs?	#173
OK	Are there user permissions defined in the Configuration management? Are there different levels of user groups? Is it traceable who made changes?	#174
OK	If there are documents handled outside the config management tool: How are they handled, how is versioning, history etc ensured?	#175

OK	Does the CM-Plan describe the states of documents and the conditions for state changes? Is the document lifecycle described for the different classes of documents?	#176

General comments:
-

3.9.3.2 Work Product: General documentation strategy for paper (pdf etc. documents)

Name(s) of document(s):

Judgement	***Topic***	***Reference***
OK	Is there a unique entry point for the documentation (documentation plan?)	#177
OK	Does every document have a title and author (responsible person)	#178
OK	Can every document be uniquely identified?	#179
OK	Has every document a state? Are approved documents signed (at least project plan, release plan and release)?	#180
OK	Has every document a history?	#181
OK	Are there templates for recurring documents?	#182
OK	Are there templates or guidelines for creating new documents?	#183

General comments:

3.9.3.3 Work Product: General documentation strategy for tool(s) supporting document maintenance

Name(s) of tools(s):

Judgement	***Topic***	***Reference***
OK	Does the tool enforce a user and rights management (in accordance to the description in the ConfMgmt-Plan)?	#184
OK	Are users uniquely identified?	#185
OK	*Does the tool allow the identification of a change (person)?*	#186
OK	Does the tool support the recording of the history of changes?	#187

OK	Does the tool support base lining?	#188
OK	Is there an approach to enforce approval and signing in the tool?	#189
OK	Is there an approach to uniquely identify exports from the tool?	#190
General comments:		

3.9.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: CM-Plan
TODOs: -

Work Product: General documentation strategy for paper (pdf etc. documents)
TODOs: -

Work Product: General documentation strategy for tool(s) supporting document maintenance
TODOs: -

3.9.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

Objective	Obligations for next assessment	Judgement
Work Product: CM-Plan	**None**	**OK**
Work Product: General documentation strategy for paper (pdf etc. documents)	**None**	**OK**
Work Product: General documentation strategy for tool(s) supporting document maintenance	**None**	**OK**

3.10 Customer Information, Hot fix procedure and Change management

3.10.1 Requirements

The following table lists the general requirements for this part. In section “plan for the assessment” more specific questions address various aspects and guide through the assessment. Any additional information

that is useful for understanding the customer's approach can be recorded in the "General comments" field below the assessment questions.

Customer information, Hot-fix procedure and change management
Requirements
R25. Customer information - "Field monitoring" - Reporting and correcting actions (also relation to: change management and hot-fix) **R26.** Hot-fix process - Decision and planning - Documentation of the decision (before the change request will be operated): o Analysis of the change request o Identification of affected work products o Schedule for the realization and verification o The decision shall be made by authorized persons (project manager, quality assurance manager, involved developers, …) - Documentation and carrying out of the change: o Exact description o Reason o Responsible person o List of changed work products o Details of the carried-out change - Planned date for deployment/release **R27.** Change management - Systematic planning, control, monitoring, implementation and documentation of changes - Documentation of the decision (before the change request will be operated): o Analysis of the change request o Identification of affected work products o Schedule for the realization and verification o The decision shall be made by authorized persons (project manager, quality assurance manager, involved developers, …) only - Documentation and carrying out of the change: o Exact description o Reason o Responsible person o List of changed work products o Details of the carried out change - Planned date for deployment/release

3.10.2 Specific documents for topic

The following table shows a list of relevant documents for this section in relation to mandatory or optional work products.

Reference(s)	Work Product
	Product Engineering Process (PEP)

3.10.3 Plan for the assessment

Ask the following questions. All topics shall be answered. If any document or information is missing, the reason needs to be recorded. If the reason is plausible, make a note and judge it "ok". If it is not plausible, judge it "nok" and make it an open issue (e.g. in the ToDo-List). Record any additional information that helps understanding the judgement. If additional documents are shown during the assessment, capture

their names in the documents list above. Record any additional information that is useful for understanding the customer's approach in the field "**General comments"**

3.10.3.1 Work Product: Customer Information

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there a communication path defined to inform customers about newly identified bugs (active or passively?)	#191
OK	Is a role defined that is responsible for customer information?	#192
OK	Is a process defined that feeds new bugs into the hot-fix process?	#193
OK	Are there methods to track new bugs and to inform customers about availability of resolutions of bugs/availability of a hot fix (even those customers that are not involved in discovering the bug)?	#194
OK	How are bugs traceable from customer support to the development internal change management system?	#195
General comments: --		

3.10.3.2 Work Product: Hot Fix

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is there a role that is responsible after SOP for deciding about hot fixes?	#196
OK	Is a hot fix treated like a change (see above)? If not, answer the questions from section 3.12 for the hot fix process.	#197
OK	Are there sufficient regression tests before the release of a HF?	#198
OK	Are hot fixes prioritized?	#199
OK	How is a defect decided to be hot fixed or implemented in a later release	#200
OK	Is the implementation of the hot fix planned?	#201
OK	Is there a development completion criterion?	#202
General comments:		

3.10.3.3 Work Product: Change Management Plan

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Is the change management process defined (only applicable if different from normal development)?	#203
OK	Is the change management process consistent to the development (are relevant activities executed, e.g. reviews, approvals, verification etc.)?	#204
OK	Are sources of CRs defined? Which are the sources?	#205
OK	Does the change management process consider the analysis of the change request (possible safety impact, relevance, and criticality)?	#206
OK	Does the change management process consider defined roles for the decision about changes? Is the decision recorded with a rationale?	#207
OK	Are the states and progression of changes well defined?	#208
OK	Does the CR Plan consider scheduling of the CRs?	#209
OK	Is there a special process for late changes?	#210

General comments:
-

3.10.3.4 Work Product: CR Management

Name(s) of document(s):		
Judgement	***Topic***	***Reference***
OK	Has every CR a state (at least accepted, in progress, ready for verification, closed)?	#211
OK	Has every CR an owner?	#212
OK	Has every CR state an assigned responsible person?	#213
OK	Has every CR acceptance criteria?	#214
OK	Has every CR a priority?	#215

OK	Is the priority of safety relevant CRs high?	#216
OK	Are changed WP recorded with the CR (possibly with relation to CM)?	#217
OK	Has every CR a due date or a planned release?	#218
OK	Is there always (in every state) somebody visibly assigned to take action/be responsible for the change request?	#219
OK	Is there a timeline defined for every change? Who keeps track of the progress?	#220

General comments:
-

3.10.4 Open issues from the assessment and the reviews

If there are open issues from the onsite-assessment or the reviews other than those already captured in the review protocols, put them below. To combine earlier comments in the review protocols it is possible to move open issues of the review protocols to this list. In that case, make a statement in the review protocol, that the topic has been moved to the process checklist.

Work Product: Customer information
TODOs: -

Work Product: Hot Fix
TODOs: -

Work Product: Change Management Plan
TODOs: -

Work Product: CR Management
TODOs: -

3.10.5 Overall Judgement

Here the final overall judgement is recorded. The records shall be given for the phase as a whole and the main documents. In addition, any obligations for a re-certification shall be recorded.

Objective	Obligations for next assessment	Judgement
Work Product: Customer Information	**None**	**OK**
Work Product: Hot Fix	**None**	**OK**

Work Product: Change Management Plan	**None**	**OK**
Work Product: CR management	**None**	**OK**

Anhang C: Klassisches Assessment

Typischerweise werden Assessments in Textdokumenten (Microsoft Word) erstellt. Im folgenden Beispiel ist ein Technischer Bericht dargestellt.

Abbildung 81: Deckblatt eines Technischen Berichtes in Microsoft Word

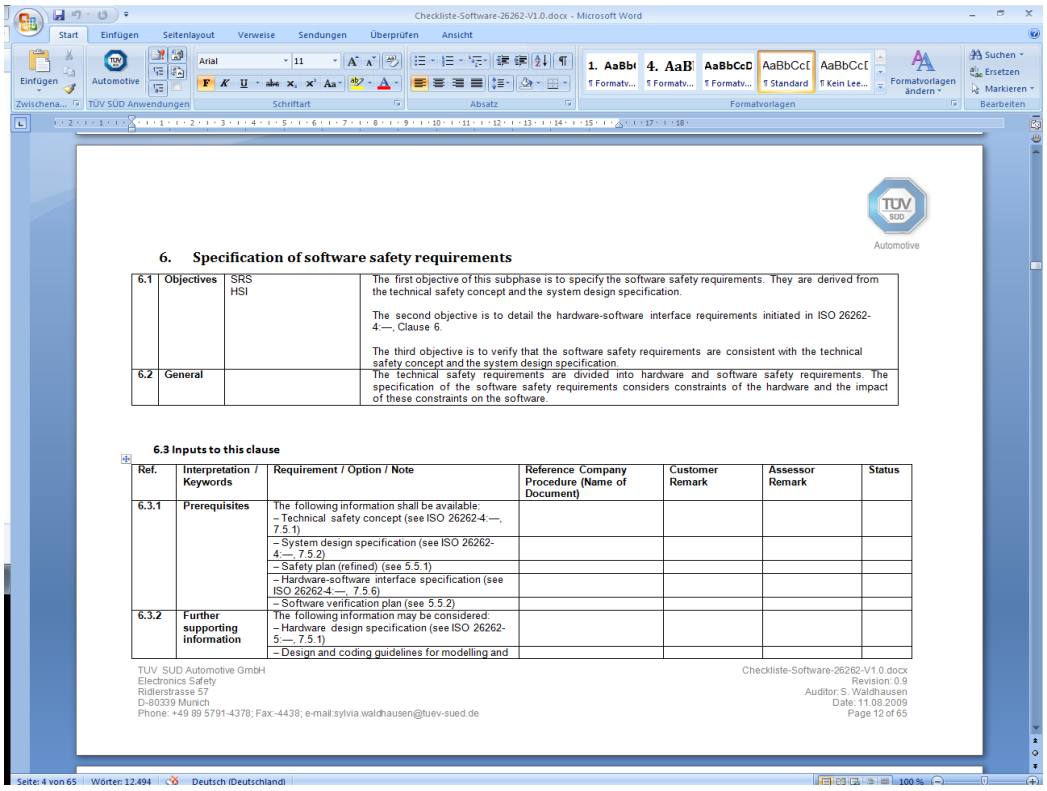

Abbildung 82: Ausschnitt aus der Checkliste zur Prüfung der Norm ISO 26262 (im Bild Band 4 Kapitel 6)

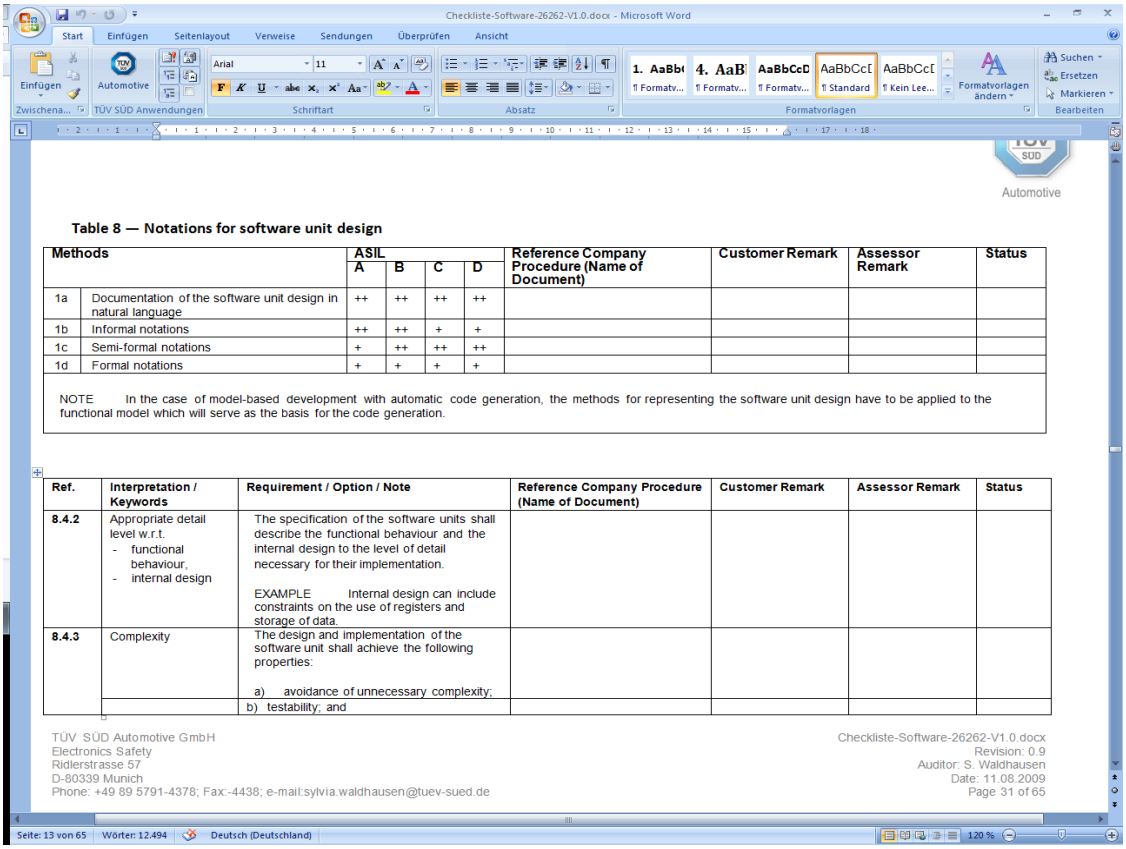

Table 8 — Notations for software unit design

Methods		ASIL A	ASIL B	ASIL C	ASIL D	Reference Company Procedure (Name of Document)	Customer Remark	Assessor Remark	Status
1a	Documentation of the software unit design in natural language	++	++	++	++				
1b	Informal notations	++	++	+	+				
1c	Semi-formal notations	+	++	++	++				
1d	Formal notations	+	+	+	+				

NOTE In the case of model-based development with automatic code generation, the methods for representing the software unit design have to be applied to the functional model which will serve as the basis for the code generation.

Ref.	Interpretation / Keywords	Requirement / Option / Note	Reference Company Procedure (Name of Document)	Customer Remark	Assessor Remark	Status
8.4.2	Appropriate detail level w.r.t. - functional behaviour, - internal design	The specification of the software units shall describe the functional behaviour and the internal design to the level of detail necessary for their implementation. EXAMPLE Internal design can include constraints on the use of registers and storage of data.				
8.4.3	Complexity	The design and implementation of the software unit shall achieve the following properties: a) avoidance of unnecessary complexity;				
		b) testability; and				

TÜV SÜD Automotive GmbH
Electronics Safety
Ridlerstrasse 57
D-80339 Munich
Phone: +49 89 5791-4378; Fax:-4438; e-mail:sylvia.waldhausen@tuev-sued.de

Abbildung 83: Ausschnitt aus der Checkliste zur Prüfung der Norm ISO 26262 (im Bild Band 6 Kapitel 8)

Anhang D: Microsoft-Access-basiertes Assessment-Tool von TÜV SÜD Automotive

Im Folgenden sind Screenshots der in Kapitel 4.1.4.2 vorgestellten Mircosoft-Access-basierten Lösung zur Assessment-Unterstützung zu finden.

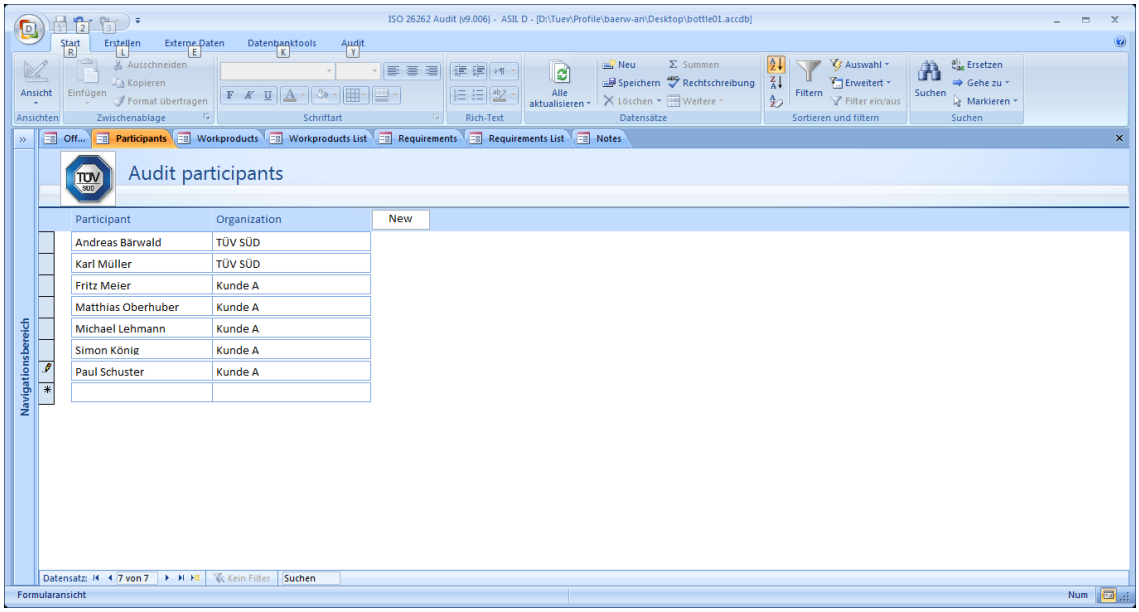

Abbildung 84: Erfassung der am Assessment beteiligten Sachverständigen

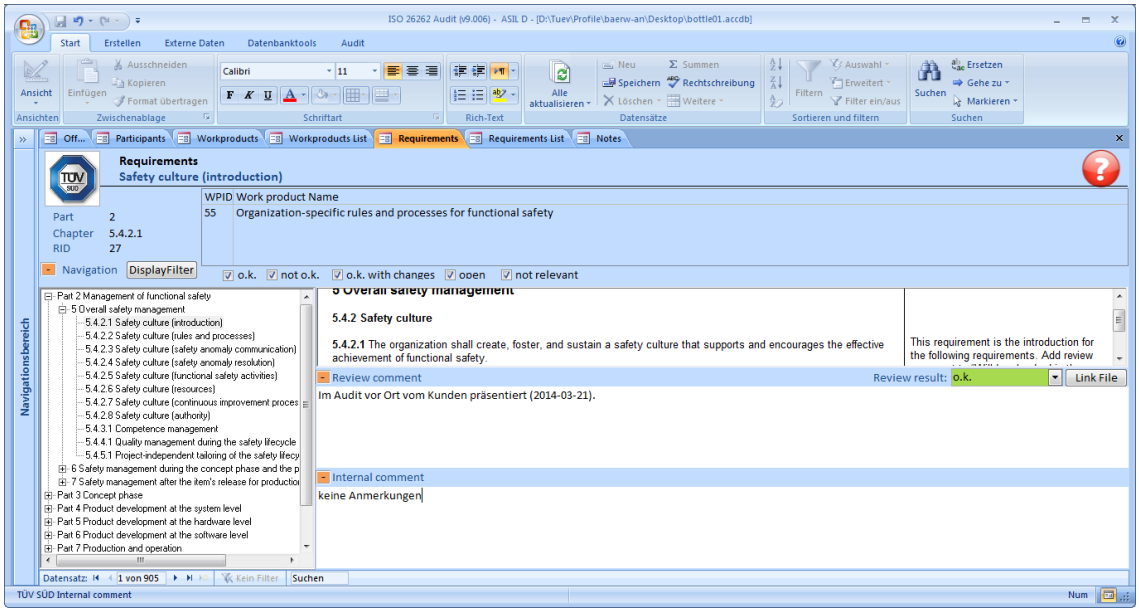

Abbildung 85: Liste aller zugrundeliegenden Anforderungen (Abbildung eines Standards am Beispiel der ISO 26262)

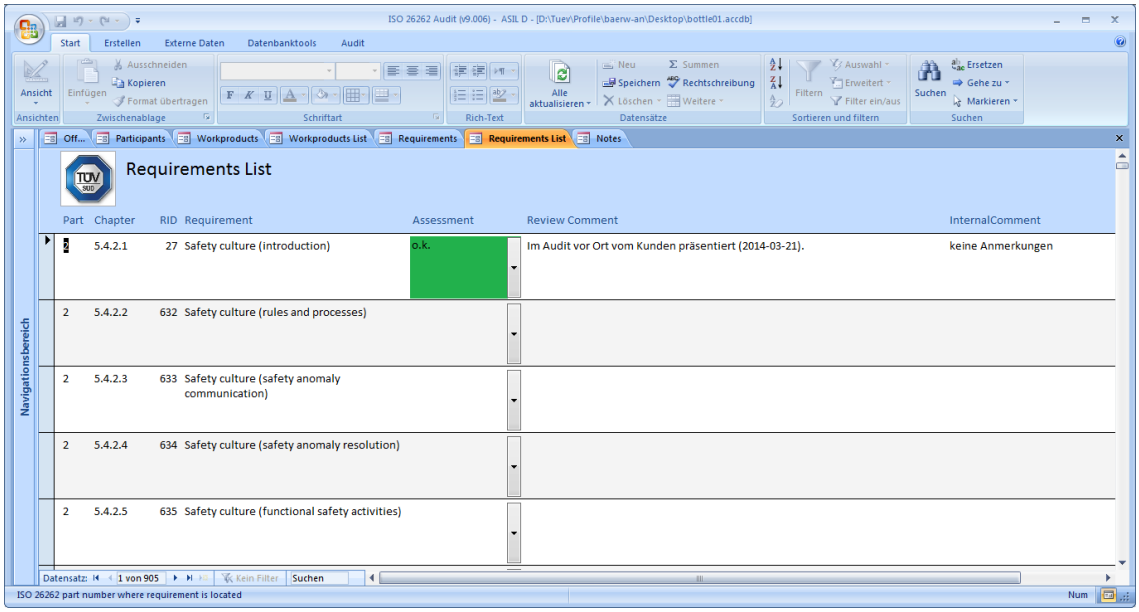

Abbildung 86: Liste der Bewertungen

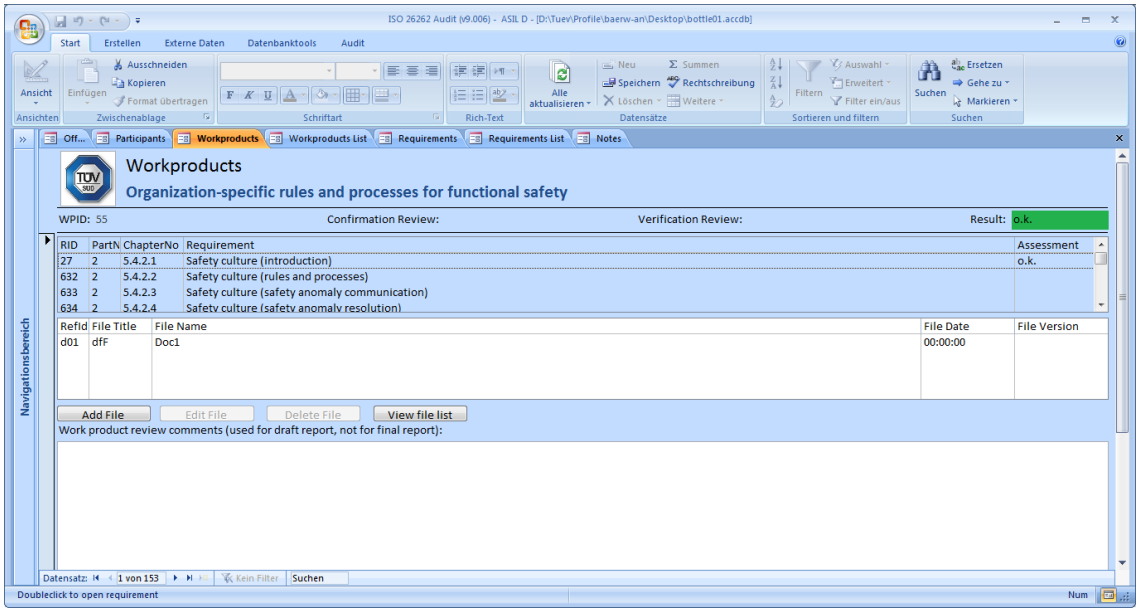

Abbildung 87: Auflistung aller vom Standard geforderten Dokumente (Work Products)

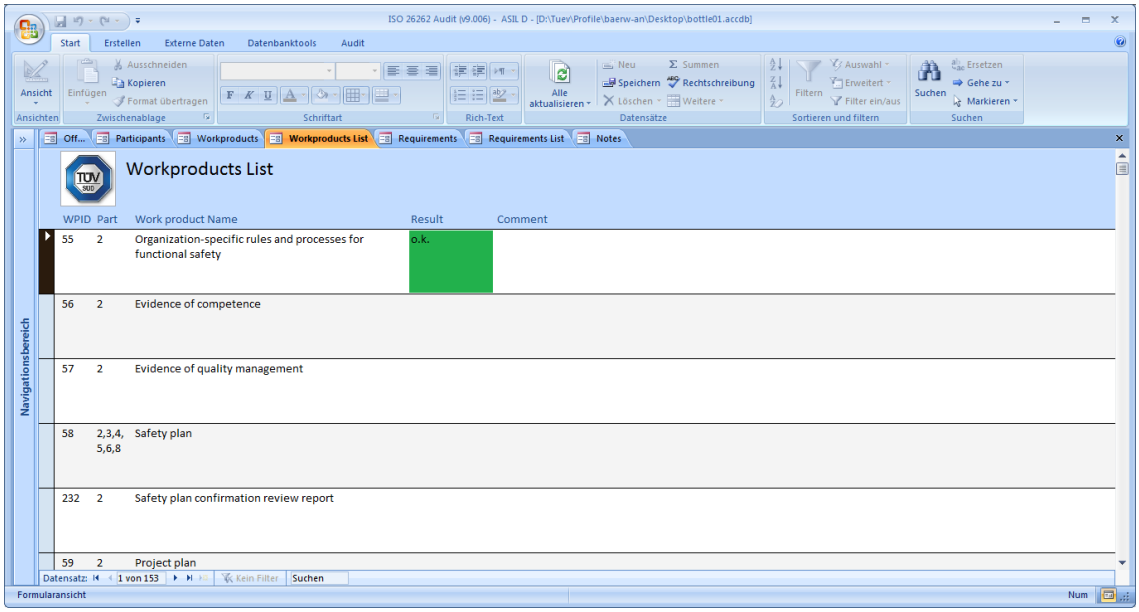

Abbildung 88: Bewertung der projektspezifischen Dokumente (Work Products)

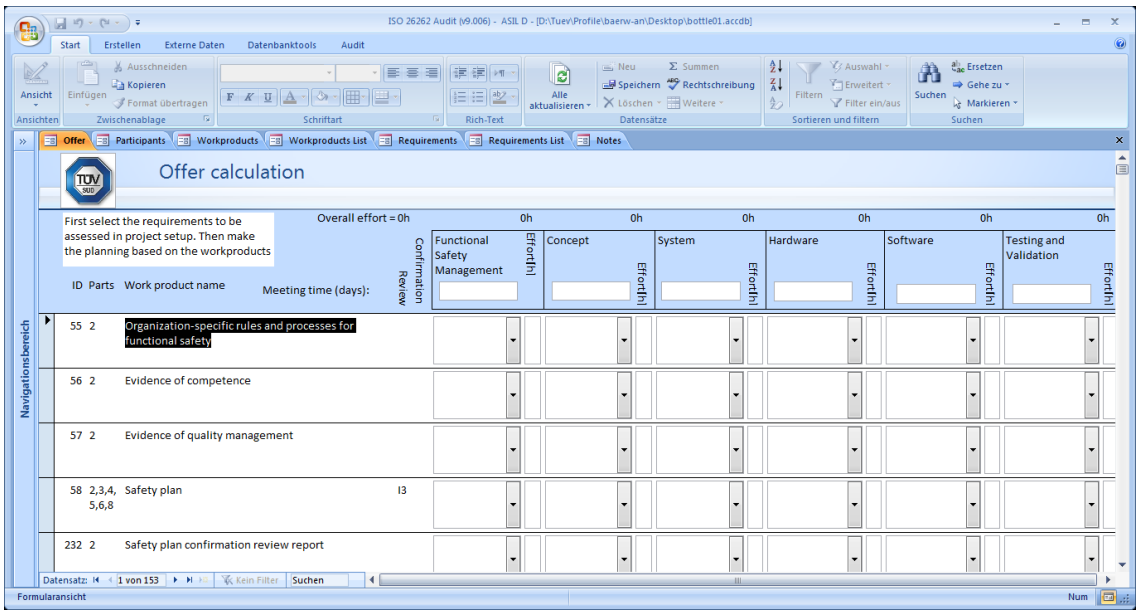

Abbildung 89: Kalkulation der Projektaufwände auf Basis der vom Standard geforderten Dokumente

Anhang E: PTC Integrity Assessment-Model-Plug-In-Demonstrator

In diesem Teil des Anhangs wird der auf PTC Integrity basierende Demonstrator (siehe Kapitel 4.3.3) anhand von Screenshots der Lösung vorgestellt.

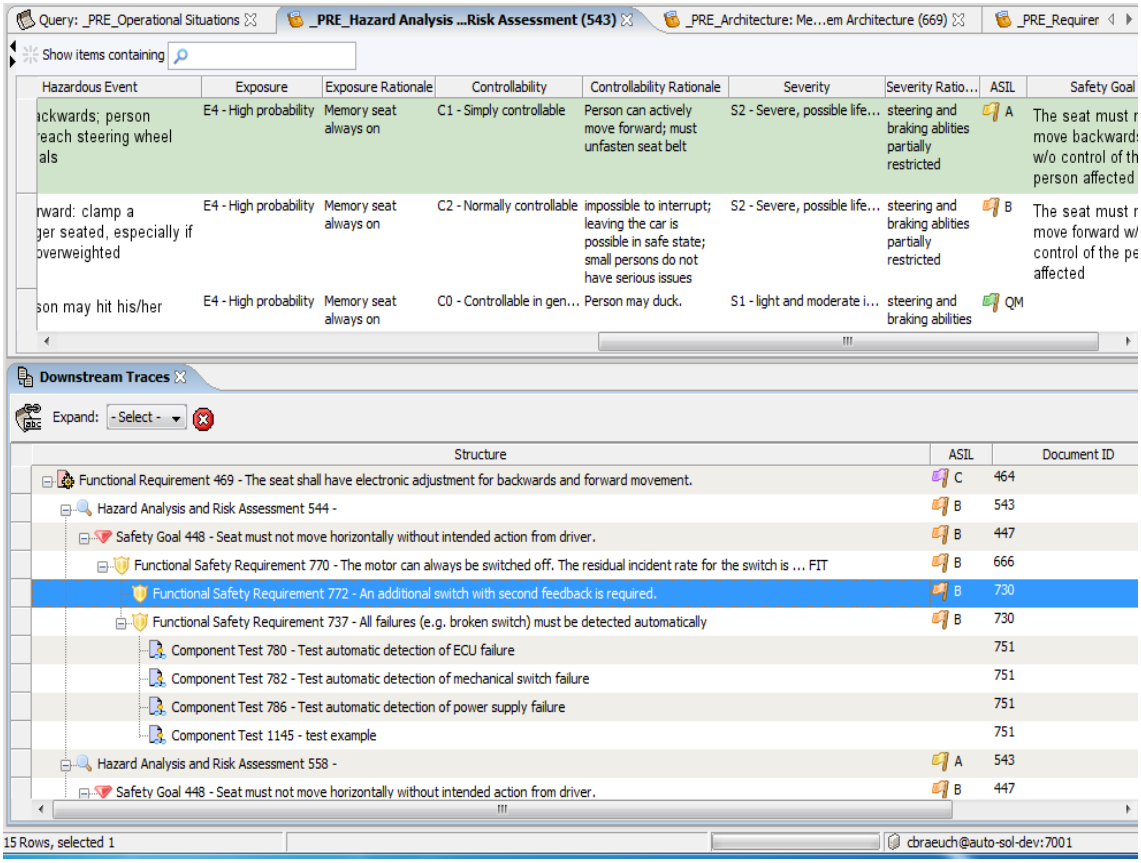

Abbildung 90: Umsetzung eines Templates für die Gefährdungs- und Risikoanalyse nach ISO 26262 auf Basis von PTC Integrity

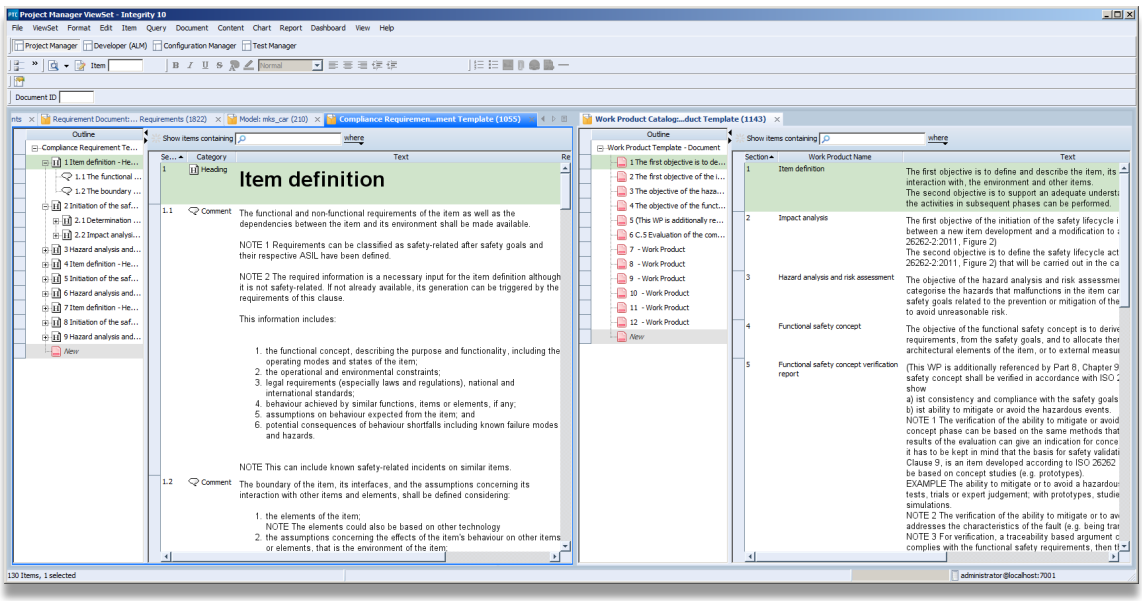

Abbildung 91: Zusammenspiel von Compliance-Requirements und den dazugehörigen Work Products nach ISO 26262

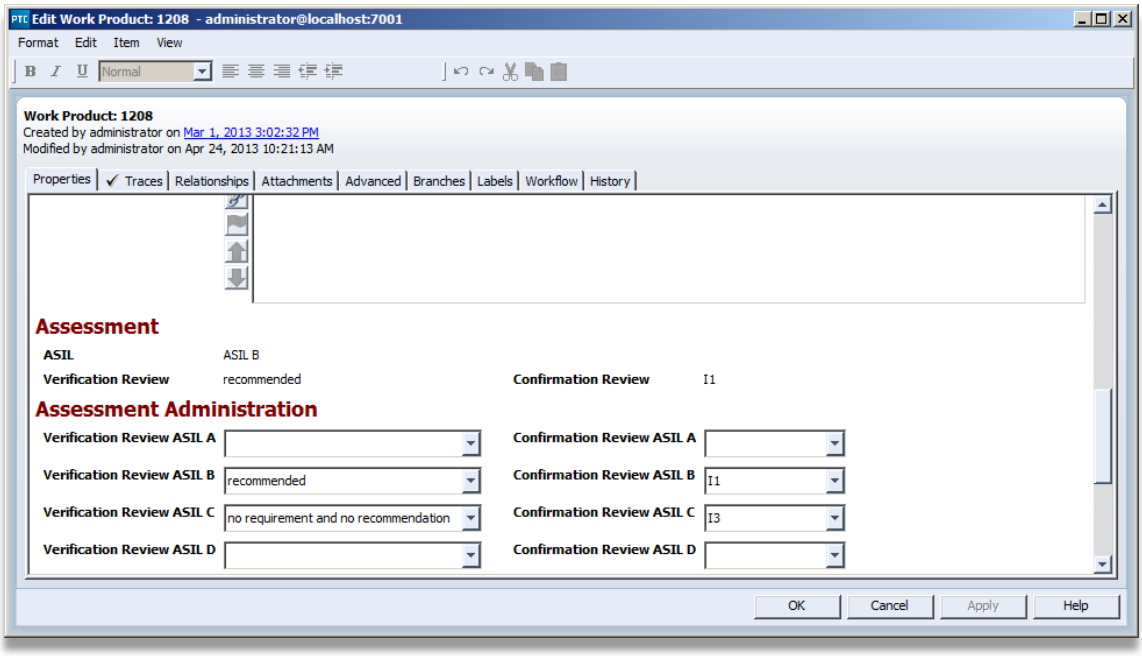

Abbildung 92: Assessment-Kriterien für Work Products

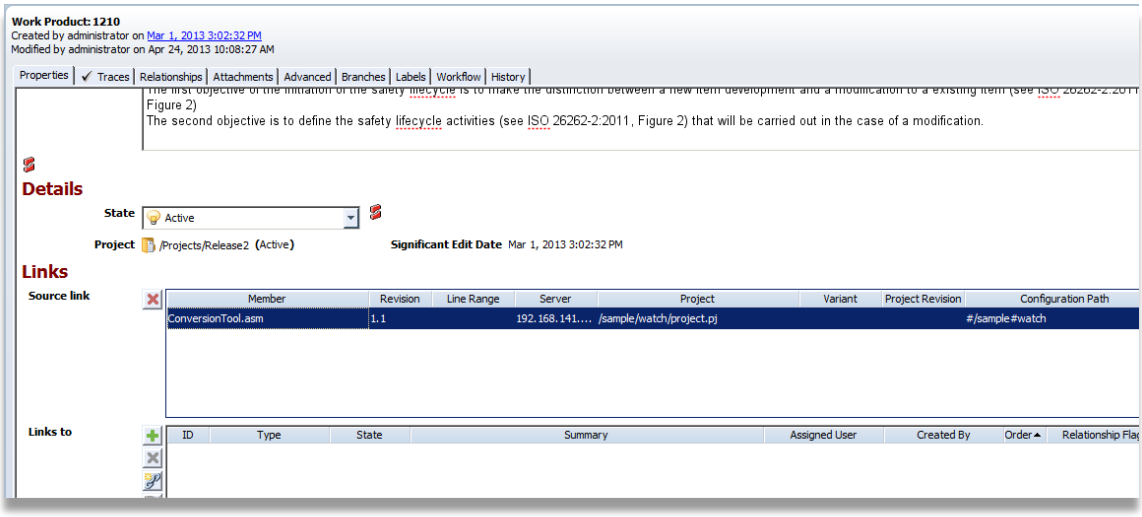

Abbildung 93: Verlinkung von Work Products zu den Entwicklungsdokumenten im ALM-Werkzeug

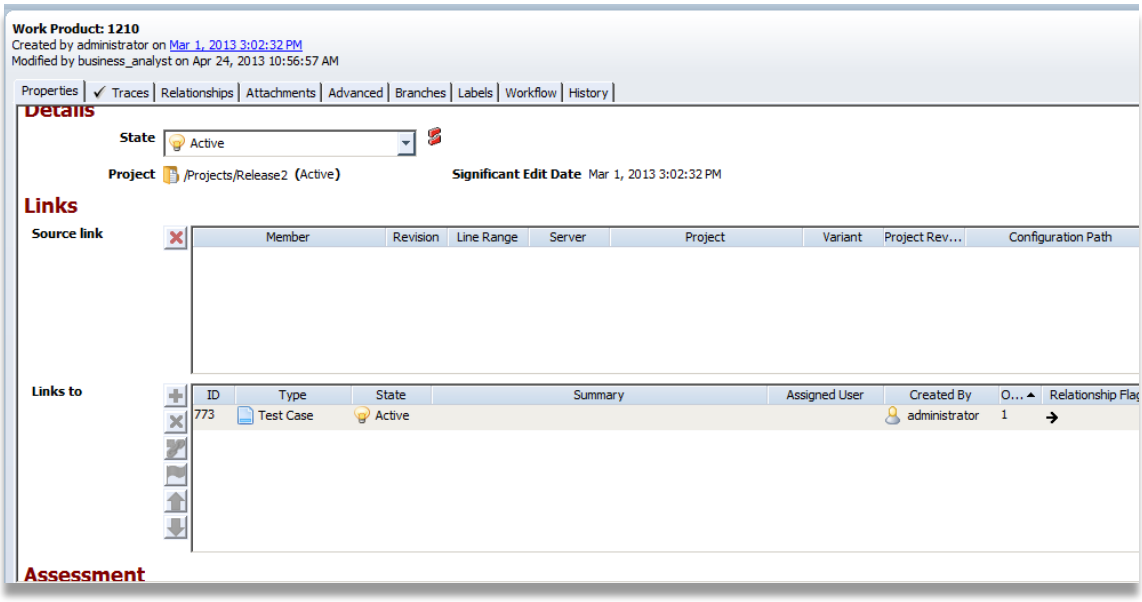

Abbildung 94: Einbindung von Verifikationsdokumenten

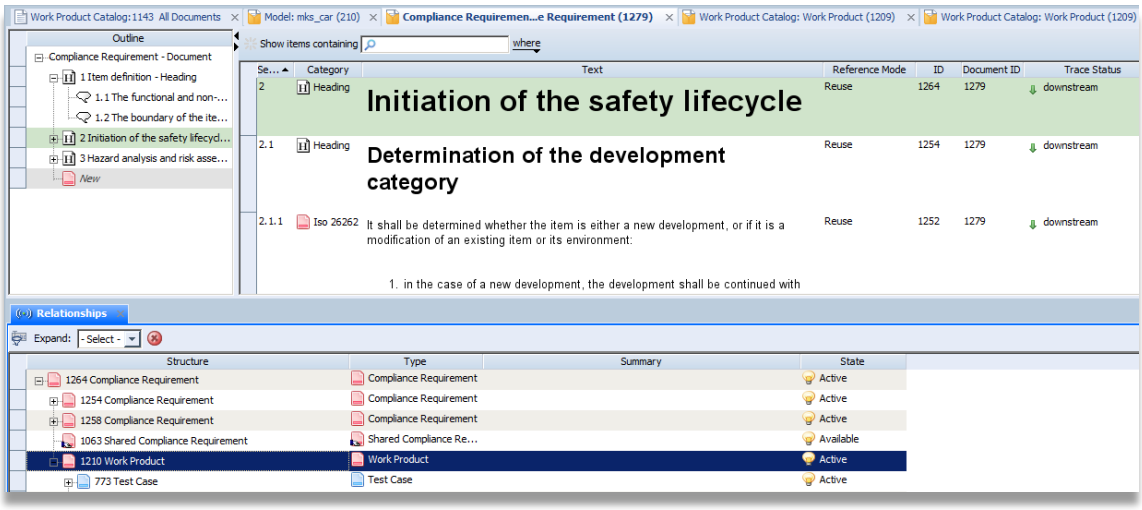

Abbildung 95: Umsetzung der normativen Anforderungen in der Liste der Compliance-Requirements

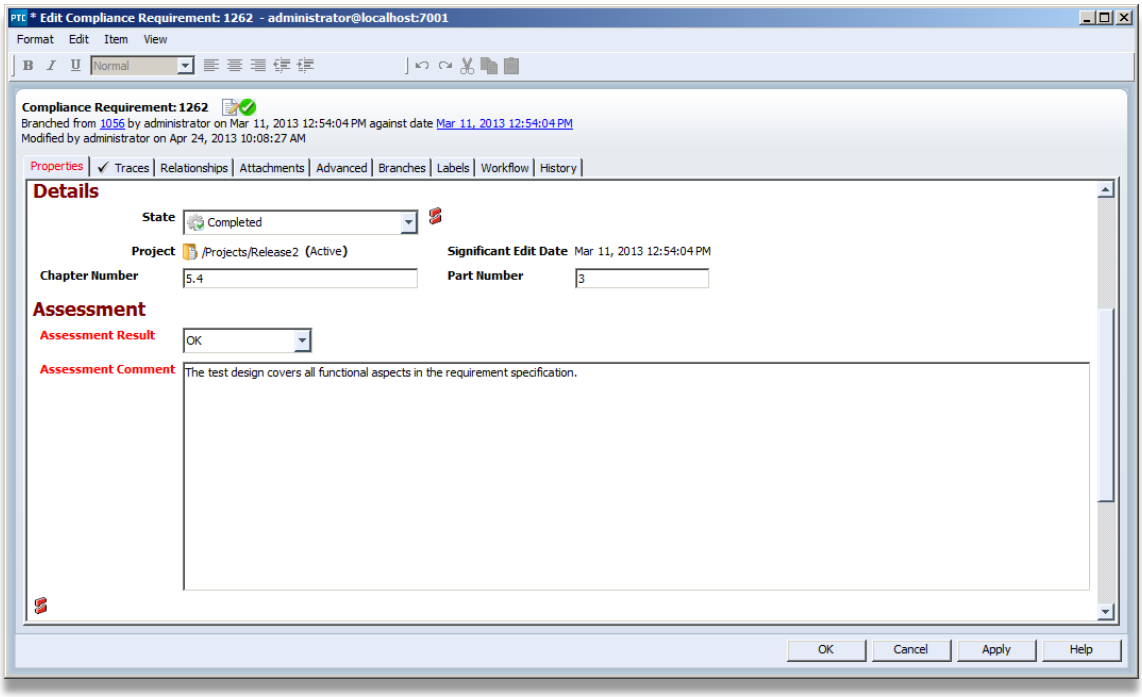

Abbildung 96: Bewertungsparameter für die Umsetzung einer normativen Klausel (Compliance-Requirement) im Assessment-Modell

Abbildung 97: Zustandsmodell für die Umsetzung der Compliance-Requirements

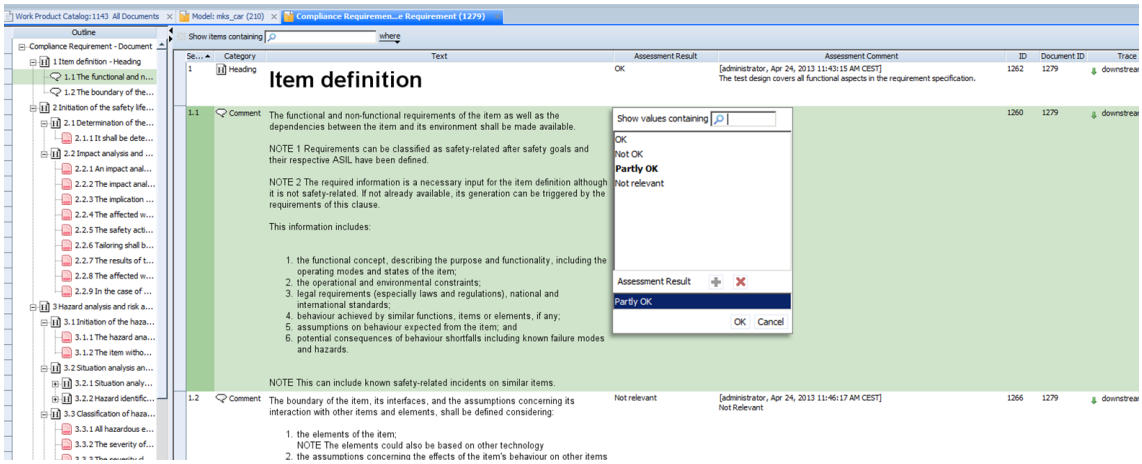

Abbildung 98: Bewertung der Erfüllung der Norm durch den Assessor

Item ID (Web Link)	Summary	State	Assigned User	P
Priority:				
838	Support Disabling Extra Rinse while Cycle running	Closed	administrator	/Rele
837	Revise Defaults for Delicate Mode	In Progress	administrator	/Rele
841	Increase Drum Material Strength to Over 9000	In Progress	administrator	/Rele
890	Authorizing change to Top Load Washer Drum Assembly Requirements	In Progress	student	/Rele
903	Create properties file for configurable parameters and localization	In Progress	student	/Rele
905	Add default delay value to washing machine	In Progress	student	/Rele
907	Add ending to washing machine cycle	In Progress	student2	/Rele
909	Add second rinse to washing machine cycle	In Progress	student	/Rele
911	Add stop command to power function in washing machine	In Progress	student	/Rele
839	Light Indicator Falls out of Sync with Light Status	Planned	administrator	/Rele
840	Add Clean Cycle Reminder Light to System	Triage		/Rele
				Priority Iter
Priority: Low				
431	Add signal buzzer for end-of-cycle.	Draft		/Rele
433	Add water level indicators to inside of drum.	Draft		/Rele
438	Set High as default Spin cycle for Power Wash wash cycle.	Planned		/Rele
				Priority Low Ite
Priority: Medium				
430	Change color of start button to BLUE.	In Progress	administrator	/Rele
432	Change door lock indicator color to RED.	Planned		/Rele
437	Set Warm/Warm as default temperature for Normal wash cycle.	Planned		/Rele
				Priority Medium Ite

Abbildung 99: Prioritätenliste zur Abarbeitung offener Punkte im Assessment

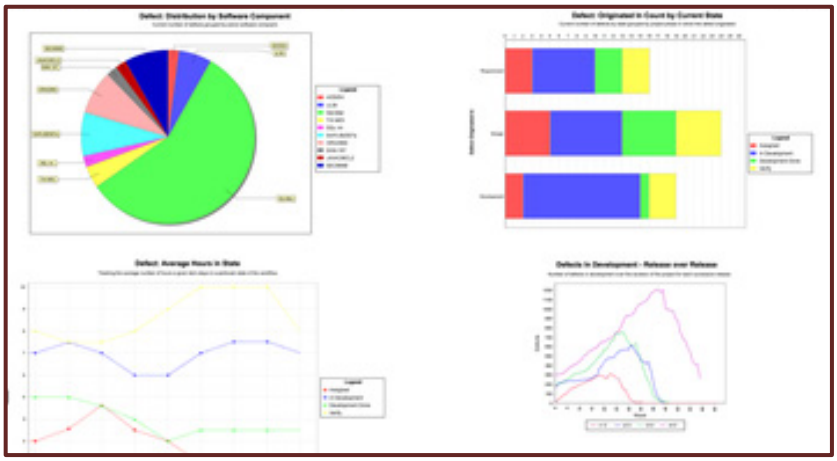

Abbildung 100: Management-Dashboard-Beispielimplementierung