

DIN EN ISO 19014-4

ICS 53.100

Einsprüche bis 2019-08-07

Entwurf**Erdbaumaschinen –
Funktionale Sicherheit –****Teil 4: Gestaltung und Beurteilung von Software und Datenübertragung für
sicherheitsrelevante Steuerungssysteme (ISO/DIS 19014-4:2019);
Deutsche und Englische Fassung prEN ISO 19014-4:2019**

Earth-moving machinery –

Functional safety –

Part 4: Design and evaluation of software and data transmission for safety-related parts of the control system (ISO/DIS 19014-4:2019);

German and English version prEN ISO 19014-4:2019

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2019-06-07 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfs besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise online im Norm-Entwurfs-Portal von DIN unter www.din.de/go/entwuerfe bzw. für Norm-Entwürfe der DKE auch im Norm-Entwurfs-Portal der DKE unter www.entwuerfe.normenbibliothek.de, sofern dort wiedergegeben;
- oder als Datei per E-Mail an nam@din.de möglichst in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/go/stellungnahmen-norm-entwuerfe oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder in Papierform an den DIN-Normenausschuss Maschinenbau (NAM), 60498 Frankfurt am Main, Postfach 71 08 64, Lyoner Str. 18, 60528 Frankfurt am Main.

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 91 Seiten

DIN-Normenausschuss Maschinenbau (NAM)

Nationales Vorwort

Dieser Norm-Entwurf enthält sicherheitstechnische Festlegungen.

Er beinhaltet die Deutsche Fassung der vom Technischen Komitee ISO/TC 127 „Earth-moving machinery“ in Zusammenarbeit mit dem Technischen Komitee CEN/TC 151 „Bau- und Baustoffmaschinen - Sicherheit“ im Europäischen Komitee für Normung (CEN) ausgearbeiteten prEN ISO 19014-4:2019.

Um Zweifelsfälle in der Übersetzung auszuschließen, ist die englische Originalfassung beigelegt. Die Nutzungsbedingungen für den deutschen Text des Norm-Entwurfes gelten gleichermaßen auch für den englischen Text.

Die nationalen Interessen bei der Erarbeitung werden vom Arbeitsausschuss NA 060-13-01 AA „Erdbaumaschinen/Straßenbaumaschinen“ im Fachbereich „Bau- und Baustoffmaschinen“ des DIN-Normenausschusses Maschinenbau (NAM) wahrgenommen. Vertreter der Hersteller und Anwender von Erd- und Straßenbaumaschinen sowie der Berufsgenossenschaften sind an der Erarbeitung beteiligt.

Dieser Norm-Entwurf konkretisiert einschlägige Anforderungen von Anhang I der EU-Maschinenrichtlinie 2006/42/EG an erstmals im Europäischen Wirtschaftsraum (EWR) in Verkehr gebrachte Maschinen, um den Nachweis der Übereinstimmung mit diesen Anforderungen zu erleichtern.

Die im Abschnitt 2 und den Literaturhinweisen zitierten Internationalen Normen sind, sofern diese gleichzeitig als Europäische Normen übernommen worden sind, als DIN-EN-ISO-Normen mit gleicher Zählnummer veröffentlicht.

Für die zitierten Internationalen Normen und Dokumente, sofern sie nicht als DIN-ISO-Normen mit gleicher Zählnummer veröffentlicht sind, gibt es keine nationalen Entsprechungen.

**Erdbaumaschinen — Funktionale Sicherheit — Teil 4: Gestaltung und
Beurteilung von Software und Datenübertragung für sicherheitsrelevante
Steuerungssysteme (ISO/DIS 19014-4:2019)**

*Earth-moving machinery — Functional safety — Part 4: Design and evaluation of software and data transmission
for safety-related parts of the control system (ISO/DIS 19014-4:2019)*

ICS:

Deskriptoren:

Dokument–Typ: Europäische Norm
Dokument–Untertyp:
Dokumentstufe: parallele Umfrage
Dokumentsprache: D

STD Version 2.9p

Inhalt

	Seite
Europäisches Vorwort	4
Vorwort	5
Einleitung	6
1 Anwendungsbereich	7
2 Normative Verweisungen	7
3 Begriffe	7
4 Software-Entwicklung	10
4.1 Planung	10
4.2 Artefakte	12
4.3 Spezifikation der Sicherheitsanforderungen an die Software	13
4.4 Gestaltung der Software-Architektur	14
4.5 Software-Modulgestaltung und Codierung	15
4.6 Sprache, Bibliothek und Tool-Auswahl	17
4.7 Prüfen von Software-Modulen	17
4.8 Software-Modulintegration und -prüfung	19
4.9 Software-Validierung	20
5 Softwarebasierte Parametrierung	20
5.1 Allgemeines	20
5.2 Datenintegrität	21
5.3 Verifizierung softwarebasierter Parametrierung	21
6 Schutz der Übertragung sicherheitsbezogener Nachrichten in Bussystemen	21
7 Unabhängigkeit durch Software-Partitionierung	23
7.1 Mehrere Partitionen in einem einzelnen Mikrocontroller	24
7.2 Mehrere Partitionen im Rahmen eines ECU-Netzwerks	25
Anhang A (informativ) Beschreibung der Software-Methoden/Maßnahmen	26
A.1 Spezifikation der Anforderungen in natürlicher Sprache	26
A.2 Rechnergestützte Spezifikationstools	26
A.3 Informelle Methoden	26
A.4 Halbformelle Methoden	26
A.5 Formelle Methoden	26
A.6 Nachverfolgbarkeit der Sicherheitssoftware	27
A.7 Walkthrough	28
A.8 Inspektion	28
A.9 Rechnergestützte Gestaltungstools	29
A.10 Sicherheitsleistung in Echtzeit	29
A.11 Gestaltungsregeln	29
A.12 Keine dynamischen Variablen oder Objekte	30
A.13 Online-Prüfung der Erstellung dynamischer Variablen oder Objekte	31
A.14 Modularisierung	31
A.15 Strukturierte Programmierung	32
A.16 Defensive Programmierung	32
A.17 Verwendung vertrauenswürdiger/verifizierter Software-Elemente	33
A.18 Geeignete Programmiersprache	34

A.19	Unterstützung für Sprachteilmenge	34
A.20	Tools mit zunehmender Bewährtheit im Betrieb oder in der Validierung.....	34
A.21	Zertifizierte Tools und Übersetzungsprogramme	35
A.22	Grenzwertanalyse	35
A.23	Kontrollflussanalyse	36
A.24	Datenflussanalyse	36
A.25	Prüffallausführung anhand der Grenzwertanalyse	36
A.26	Funktions-/Black-Box-Prüfung.....	36
A.27	Strukturabhängige Prüfungen.....	37
A.28	Äquivalenzklassen und Eingabe-Partitionsprüfungen.....	37
A.29	Prüffallausführung aus modellbasierter Prüffallgenerierung	37
A.30	Leistungsprüfungen.....	38
A.31	Software-Modulschnittstellenprüfung	39
A.32	Direkte Vergleichsprüfung	40
Anhang B (normativ) Prüfumgebung für Softwarevalidierung.....		41
B.1	Maschinennetzwerkprüfung.....	41
B.2	Hardware-in-the-Loop-Prüfung.....	41
B.3	Prüfung der Maschinenebene	42
Anhang C (informativ) Datenintegritätssicherung		44
C.1	Berechnung der Datenintegritätssicherung.....	44
Anhang D (informativ) Methoden und Maßnahmen zum Übertragungsschutz		45
D.1	Keep-Alive-Nachrichten	45
D.2	Alive Counter.....	45
D.3	CRC	45
D.4	Sequenznummer.....	45
D.5	Nachrichtenwiederholung	45
D.6	Watchdog.....	46
D.7	Zeitgesteuerter Datenbus.....	46
D.8	Buswächter	46
D.9	Minislotting.....	46
Anhang E (informativ) Methoden und Maßnahmen für Mikrocontroller-internen Datenschutz		47
E.1	Eindeutiges bidirektionales Kommunikationsobjekt.....	47
E.2	Ausschließlich zwei unidirektionale Kommunikationsobjekte	47
E.3	IDs zur Identifizierung und Quittierung	47
E.4	Asynchrone Datenkommunikation.....	47
E.5	Streng prioritätsbasierte Planung	47
E.6	Zeitscheibenmethode	47
E.7	Speicherschutzmechanismen	47
E.8	Verifizierung sicherheitskritischer Daten	48
E.9	Statische Analyse	48
E.10	Statische Zuweisung.....	48

Europäisches Vorwort

Dieses Dokument (prEN ISO 19014-4:2019) wurde vom Technischen Komitee ISO/TC 127 „Earth-moving machinery – Safety“ in Zusammenarbeit mit dem Technischen Komitee CEN/TC 151 „Bau- und Baustoffmaschinen – Sicherheit“ erarbeitet, dessen Sekretariat von DIN gehalten wird.

Dieses Dokument ist derzeit zur parallelen Umfrage vorgelegt.

Anerkennungsnotiz

Der Text von ISO/DIS 19014-4:2019 wurde von CEN als prEN ISO 19014-4:2019 ohne irgendeine Abänderung genehmigt.

Vorwort

ISO (die Internationale Organisation für Normung) ist eine weltweite Vereinigung nationaler Normungsorganisationen (ISO-Mitgliedsorganisationen). Die Erstellung von Internationalen Normen wird üblicherweise von Technischen Komitees von ISO durchgeführt. Jede Mitgliedsorganisation, die Interesse an einem Thema hat, für welches ein Technisches Komitee gegründet wurde, hat das Recht, in diesem Komitee vertreten zu sein. Internationale staatliche und nichtstaatliche Organisationen, die in engem Kontakt mit ISO stehen, nehmen ebenfalls an der Arbeit teil. ISO arbeitet bei allen elektrotechnischen Themen eng mit der Internationalen Elektrotechnischen Kommission (IEC) zusammen.

Die Verfahren, die bei der Entwicklung dieses Dokuments angewendet wurden und die für die weitere Pflege vorgesehen sind, werden in den ISO/IEC-Direktiven, Teil 1 beschrieben. Es sollten insbesondere die unterschiedlichen Annahmekriterien für die verschiedenen ISO-Dokumentenarten beachtet werden. Dieses Dokument wurde in Übereinstimmung mit den Gestaltungsregeln der ISO/IEC-Direktiven, Teil 2 erarbeitet (siehe www.iso.org/directives).

Es wird auf die Möglichkeit hingewiesen, dass einige Elemente dieses Dokuments Patentrechte berühren können. ISO ist nicht dafür verantwortlich, einige oder alle diesbezüglichen Patentrechte zu identifizieren. Details zu allen während der Entwicklung des Dokuments identifizierten Patentrechten finden sich in der Einleitung und/oder in der ISO-Liste der erhaltenen Patenterklärungen (siehe www.iso.org/patents).

Jeder in diesem Dokument verwendete Handelsname dient nur zur Unterrichtung der Anwender und bedeutet keine Anerkennung.

Für eine Erläuterung des freiwilligen Charakters von Normen, der Bedeutung ISO-spezifischer Begriffe und Ausdrücke in Bezug auf Konformitätsbewertungen sowie Informationen darüber, wie ISO die Grundsätze der Welthandelsorganisation (WTO, en: World Trade Organization) hinsichtlich technischer Handelshemmnisse (TBT, en: Technical Barriers to Trade) berücksichtigt, siehe www.iso.org/iso/foreword.html.

Das für dieses Dokument zuständige Komitee ist ISO/TC 127.

ISO 19014 besteht aus den folgenden Teilen:

- Earth-moving machinery — Functional Safety — Part 1: Risk assessment methodology to determine control system performance requirements
- Earth-moving machinery — Functional Safety — Part 2: Design and Evaluation of Safety-Related Machine Control Systems
- Earth-moving machinery — Functional Safety — Part 3: Environmental Testing Requirements
- Earth-moving machinery — Functional Safety — Part 4: Design and evaluation of software and data transmission for safety-related parts of the control system
- Earth-moving machinery — Functional Safety — Part 5: Table of Performance Levels

Die Normenreihe ISO 19014 ersetzt ISO 15998.

Einleitung

Diese Internationale Norm behandelt Systeme, die aus jeder möglichen Kombination elektrischer, elektronischer und programmierbarer elektronischer Bauteile [elektrische, elektronische und programmierbare elektronische Systeme (E/E/PES)], die zur Funktionssicherheit von Erdbaumaschinen verwendet werden, bestehen.

Sicherheitsnormen im Maschinenbau sind wie folgt aufgebaut:

Typ-A-Normen (grundlegende Normen) liefern Grundkonzepte, Auslegungs- und Konstruktionsgrundsätze und allgemeine Aspekte, die auf Maschinen angewendet werden können.

Typ-B-Normen (Fachgrundnormen) behandeln einen oder mehrere Sicherheitsaspekte oder eine oder mehrere Arten von Schutzeinrichtungen, die bei einer Reihe von Maschinen angewendet werden können:

- Typ-B1-Normen behandeln bestimmte Sicherheitsaspekte (z. B. Sicherheitsabstände, Oberflächentemperatur, Geräusch);
- Typ-B2-Normen behandeln Schutzeinrichtungen (z. B. Zweihandschaltungen, Verriegelungen, Kontaktmatten, trennende Schutzeinrichtungen).

Typ-C-Normen (Maschinensicherheitsnormen) behandeln detaillierte Sicherheitsanforderungen für eine bestimmte Maschine oder Gruppe von Maschinen.

Dieser Teil von ISO 19014 ist eine Typ-C-Norm entsprechend ISO 12100.

1 Anwendungsbereich

Dieser Teil von ISO 19014 legt allgemeine Grundsätze für Anforderungen an Software-Entwicklung und Signalübertragung von sicherheitsbezogenen Teilen von Maschinensteuerungssystemen (MCS, en: Machine-control Systems) in Erdbaumaschinen und deren Ausrüstung, wie in ISO 6165 definiert, fest.

Cybersicherheit liegt außerhalb des Anwendungsbereichs dieses Dokuments.

2 Normative Verweisungen

Für normative Verweisungen siehe ISO 19014-1.

3 Begriffe

Für die Anwendung dieses Dokuments gelten die Begriffe nach ISO 19014-1 und ISO 13849-1 zusammen mit den folgenden.

3.1

Bussystem

in einem elektronischen Steuerungssystem für die Übertragung sicherheitsbezogener Nachrichten verwendetes Subsystem; das Bussystem besteht aus Systemeinheit (Sources und Sinks of Information), einem Übertragungspfad/Übertragungsmedium (z. B. elektrische Leitungen, faseroptische Leitungen, HF-Übertragung) und der Schnittstelle zwischen Nachrichten-Source/Sink und Buselektronik (z. B. Protokoll-ASICs, Empfänger)

3.2

gekapseltes Bussystem

Bussystem bestehend aus einer festen Anzahl oder vorbestimmter Höchstanzahl von Busteilnehmern, die miteinander über ein Übertragungsmedium mit sorgfältig definierter und fester Leistungsstärke/Funktion verbunden sind

3.3

Ausfall der Partnerkommunikation

Kommunikationspartner ist nicht verfügbar

3.4

unbeabsichtigte Nachrichtenwiederholung

die gleiche Nachricht wird versehentlich erneut versendet

3.5

falsche Abfolge

Reihenfolge, in der die Daten gesendet wurden, hat sich während der Übertragung geändert, d. h. die Daten werden nicht in der gleichen Reihenfolge erhalten, in der sie gesendet wurden

3.6

Nachrichtenwiederholung

die gleiche Nachricht wird versehentlich erneut versendet

3.7

Nachricht

elektronische Übertragung einschließlich Nutzerdaten, Adresse und Daten zur Gewährleistung der Übertragungsintegrität

3.8

maximale Erweiterungsgröße

maximal zulässige Nummer von Sendern und Empfängern, die wie für das System definiert am Nachrichtenaustausch beteiligt sind

3.9

Ansprechzeit

Zeit von der Erkennung eines sicherheitsrelevanten Ereignisses bis zur Einleitung einer Sicherheitsmaßnahme

3.10

Nachrichtenwiederholung

Fehler aufgrund Versagens eines Busteilnehmers, wobei alte, nicht mehr aktuelle Nachrichten zu einem falschen Zeitpunkt wiederholt werden

Anmerkung 1 zum Begriff: Diese Aktivität kann eine gefährliche Störung beim Empfänger hervorrufen (z. B. Signalisieren von „Zugangstür geschlossen“, wenn sie bereits offen ist).

3.11

Nachrichtenverlust

unbeabsichtigte Löschung einer Nachricht aufgrund Versagens eines Busteilnehmers

3.12

Einfügen von Nachrichten

unbeabsichtigtes Einfügen einer Nachricht aufgrund Versagens eines Busteilnehmers

3.13

falsche Abfolge

unbeabsichtigte Änderung der Reihenfolge der Nachrichten aufgrund Versagens eines Busteilnehmers

Anmerkung 1 zum Begriff: Bussysteme können Elemente mit gespeicherten Nachrichten (FIFOs usw.) enthalten, die die richtige Abfolge abändern können.

3.14

Falsifizierung von Nachrichten

unbeabsichtigte Änderung von Nachrichten aufgrund eines Fehlers eines Busteilnehmers oder eines Fehlers im Übertragungskanal

3.15

Nachrichtenverzögerung

unbeabsichtigte Verzögerung oder Verhinderung der Sicherheitsfunktion, hervorgerufen durch Überlast auf dem Übertragungspfad durch normalen Datenaustausch oder Versenden inkorrektur Nachrichten

3.16

Alive Counter

Zählelement, das mit „0“ initialisiert wird, sobald das zu überwachende Objekt erstellt oder wiederhergestellt wurde

Anmerkung 1 zum Begriff: Der Zähler erhöht sich von Zeit $t-1$ bis Zeit t , solange das Objekt aktiv ist. Zum Schluss zeigt der Alive Counter den Zeitraum an, in dem das Objekt in einem Netzwerk aktiv war.

3.17

Black-Box-Prüfung

Prüfung eines Objekts, die keine Kenntnis seiner internen Struktur oder seiner konkreten Implementierung erfordert

3.18

Partition

Ressourceneinheit, die einen Anteil eines Speichers, E/A-Geräte und CPU-Verwendung einem oder mehreren Aufträgen zuteilt

Anmerkung 1 zum Begriff: Die Partitionen können einem oder mehreren Subsystemen im Mikrocontroller-Netzwerk zugewiesen werden.

3.19

Software-Partitionierung

Methode zur Eindämmung von Software-Fehlern, die darin besteht, Ressourcen spezifischen Software-Komponenten zuzuweisen, um die Verbreitung eines Software-Fehlers auf mehrere Partitionen zu verhindern

3.20

absolute Adressierung

die ausdrückliche Identifikation eines Speicherorts oder eines Peripheriegeräts

(vgl. 3.17 relative Adressierung)

3.21

relative Adressierung

Identifikation eines Speicherorts oder eines Peripheriegeräts als Abstand von einer anderen Adresse

(vgl. 3.16 absolute Adressierung)

3.22

Software-Komponente

ein oder mehrere Software-Module

[MOD ISO 26262-1: 2011, 3.123]

3.23

Software-Modul

unabhängige Software, die unabhängig geprüft und auf eine Spezifikation zurückgeführt werden kann

Anmerkung 1 zum Begriff: Das Software-Modul ist eine unteilbare Software-Komponente.

3.24

Software-Partitionen

Laufzeitumgebung mit separat zugewiesenen Systemressourcen

3.25

Aufgabe

Laufzeiteinheiten, die innerhalb des Ressourcenbudgets der Partitionen und mit unterschiedlichen Prioritäten ausgeführt werden

3.26

Unabhängigkeit der Software

Ausschluss unbeabsichtigter Interaktionen zwischen Software-Komponenten sowie die Vermeidung von Auswirkungen auf den einwandfreien Betrieb einer Software-Komponente durch Fehler einer anderen Software-Komponente

3.27

Betriebshistorie

Betriebsdaten über eine Komponente oder ein Software-Modul während dessen Betriebszeit

3.28

Anforderungsprofil

Anwendungsbereich von Komponenten oder Software-Modulen, der ihr Verhalten während des Betriebs charakterisiert

3.29

gesamte Zykluszeit

feststehende Zeit zum Zugriff auf einen Kommunikationsbus zwischen Knoten auf Bus- oder Knotenebene

Anmerkung 1 zum Begriff: Die Anwendung eines zeitgesteuerten Protokolls sorgt dafür, dass diese Zykluszeit nicht überschritten wird.

3.30

maximale Ansprechzeit

einer Systemaktivität zugewiesene feste Zeit zum Austausch global synchronisierter Nachrichten auf einem Bus in einer zeitgesteuerten Architektur

3.31

Software-Fehler

ein falscher Schritt, Prozess oder Datendefinition in Software, die dafür sorgt, dass das System unerwartete Ergebnisse hervorbringt

3.32

Auswirkungsanalyse

Dokumentation, die das Verständnis für und die Auswirkungen einer vorgeschlagenen Änderung aufzeichnet

3.33

Konfigurationsmanagementprozess

die Aufgabe, Änderungen an den Artefakten im Entwicklungsprozess nachzuverfolgen und zu kontrollieren.

4 Software-Entwicklung

In diesem Abschnitt werden Empfehlungen für die Gestaltung von Software und den folgenden dazugehörigen Prüfungen gegeben. Die Vermeidung von Software-Fehlern muss während des gesamten Software-Entwicklungsprozesses berücksichtigt werden.

4.1 Planung

Das Hauptziel der folgenden Anforderungen besteht darin, mittels lesbarer, verständlicher, prüfbarer und wartbarer Software die Zuverlässigkeit der Software sicherzustellen.

Es muss ein Zeitplan erstellt werden, welcher die Beziehung zwischen den einzelnen Phasen der Software-Entwicklung und der zugehörigen Artefakte definiert.

Für die Software-Entwicklung müssen geeignete Methoden und Maßnahmen entsprechend dem MPLr gewählt werden.

Das MPLr des Systems kann durch paralleles Hinzufügen von zwei Systemen eines niedrigeren Leistungsniveaus erzielt werden. Bei parallelem Hinzufügen kann die Software in jedem System nach den niedrigeren MPLr-Anforderungen entwickelt werden. Dies ist nur zulässig, wenn es zwischen beiden Systemen keine gemeinsamen Ausfallursachen gibt.

Die Eignung der gewählten Methoden oder Maßnahmen für das Anwendungsgebiet muss gerechtfertigt und zu Beginn jeder Entwicklungsphase durchgeführt werden. Für eine bestimmte Anwendung ist die passende Kombination von Methoden oder Maßnahmen während der Entwicklungsplanung anzugeben. Nicht in Tabelle 1 bis Tabelle 7 angeführte Methoden oder Maßnahmen dürfen angewendet werden.

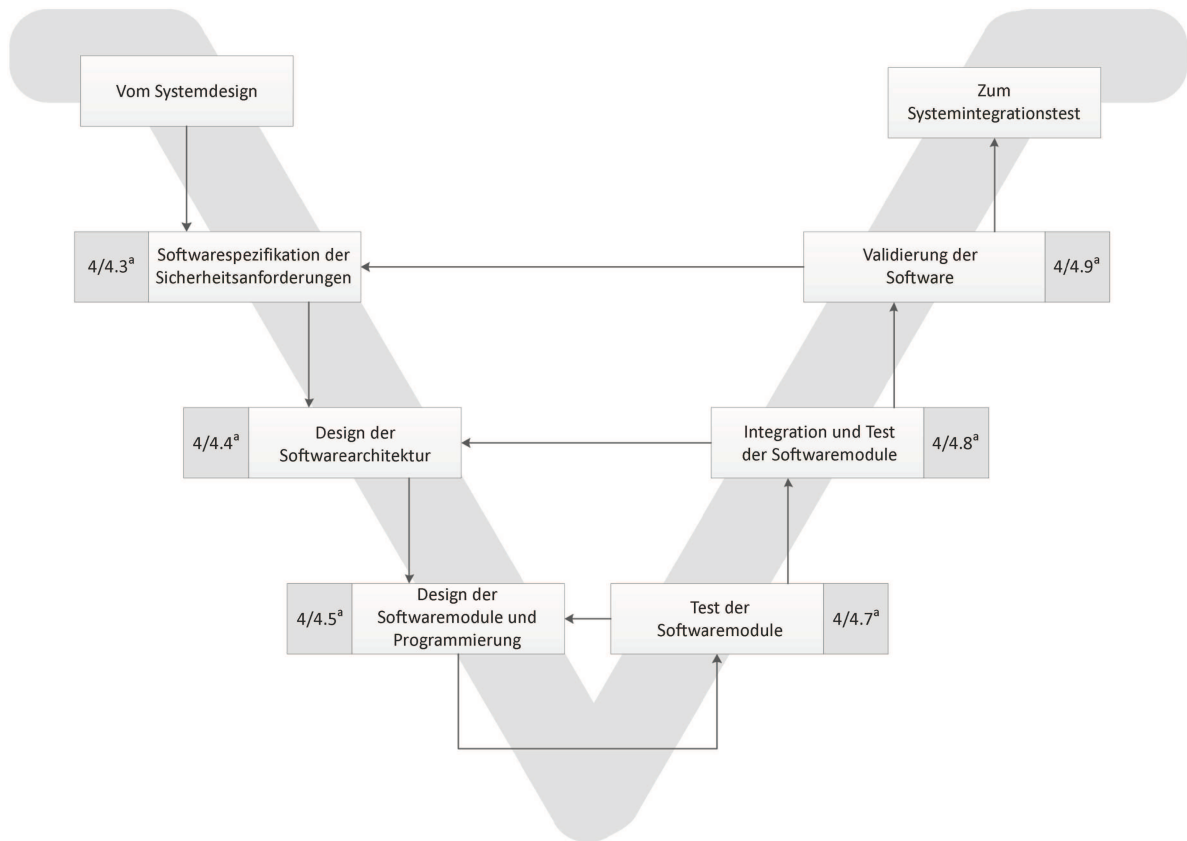


Bild 1 — Modell der Software-Entwicklung

^a Die erste Zahl in dieser und den anderen schattierten Boxen zeigt den Teil der Normenreihe ISO 19014 an, während die zweite Zahl, die von der ersten durch einen Schrägstrich getrennt ist, die Abschnittsnummer des Teils der Normenreihe anzeigt (z. B. „4/4.4“ bedeutet ISO 19014-4, Abschnitt 4,4)

ANMERKUNG Diese Zahl ist eine Darstellung einer möglichen Gestaltungsmethoden (V-Modell). Jeder organisierte, bewährte Entwicklungsprozess, der die Anforderungen in ISO 19014 erfüllt, kann zur Software-Entwicklung verwendet werden

Bei der Auswahl der Methoden und Maßnahmen kann neben der manuellen Codierung eine modellbasierte Entwicklung durchgeführt werden, bei der der Quellcode automatisch aus Modellen generiert wird.

Bei jeder Methode oder jeder Maßnahme in den Tabellen gibt es eine unterschiedliche Empfehlungsstufe für jedes Leistungsniveau. Zu diesen Empfehlungen gehören:

+	Die Methode oder Maßnahme muss für dieses MPLr angewendet werden. Wenn diese Methode oder Maßnahme nicht angewendet wird, muss die Begründung dafür während der Sicherheitsplanungsphase dokumentiert werden.
o	Die Methode oder Maßnahme kann für dieses MPLr angewendet werden.
–	Die Methode oder Maßnahme ist nicht für die Einhaltung dieses MPLr geeignet.

Es müssen die dem jeweiligen MPLr entsprechenden Methoden und Maßnahmen gewählt werden. Alternative oder gleichwertige Methoden und Maßnahmen sind durch Buchstaben nach der Zahl gekennzeichnet. Es muss mindestens eine der alternativen oder gleichwertigen Methoden und Maßnahmen gewählt werden, die mit einem „+“ gekennzeichnet sind. In einem solchen Fall ist es nicht erforderlich, eine Begründung anzugeben.

BEISPIEL

Methode/Maßnahme		MPLr = a	MPLr = b, c	MPLr = d	MPLr = e
1.a	Maßnahme 1	+	+	–	–
1.b	Maßnahme 2	+	+	+	+
1.c	Maßnahme 3	+	+	+	+

In diesem Fall gilt:

- eine Maßnahme von Maßnahme 1, Maßnahme 2 oder Maßnahme 3 muss für MPLr=a, b, c erfüllt werden;
- eine Maßnahme von Maßnahme 2 oder Maßnahme 3 muss für MPLr=d, e erfüllt werden;
- andernfalls muss eine Begründung für die nicht spezifizierte alternative Methode/Maßnahme angegeben werden, um die Anforderung der Norm für die spezifische MPLr zu erfüllen.

Wird eine Methode oder Maßnahme nicht in den Tabellen aufgeführt, kann sie verwendet werden, benötigt jedoch eine Begründung oder Rechtfertigung.

Wenn eine Software-Komponente sich auf verschiedene Sicherheitsfunktionen mit anderer MPLr auswirkt, dann gelten die Anforderungen im Zusammenhang mit der höchsten MPLr.

Wenn die Software Komponenten mit unterschiedlicher MPLa oder sicherheitsbezogene und nicht sicherheitsbezogene Komponenten enthält, dann muss die gesamtintegrierte Software-MPLa auf die Software-Komponente mit der niedrigsten MPLa begrenzt werden, sofern nicht eine ausreichende Unabhängigkeit zwischen den Software-Komponenten in Übereinstimmung mit Abschnitt 7 nachgewiesen werden kann.

Wenn eine bestehende Software-Komponente nach einer früheren Norm entwickelt und durch Anwendung, Verwendung und Validierung nachgewiesen wurde, dass sie das Risiko so gering wie praktisch möglich hält, gibt es keine Anforderung an die Aktualisierung der Dokumentation Software-Lebenszyklus auf Modulebene. Wenn die zuvor eingesetzte Software-Komponente modifiziert wird, muss eine Auswirkungsanalyse durchgeführt werden. Ein Maßnahmenplan muss erstellt und für den Software-Gesamtlebenszyklus umgesetzt werden, basierend auf den Ergebnissen der Auswirkungsanalyse, um die Einhaltung der Sicherheitsziele zu gewährleisten.

4.2 Artefakte

Mit Blick auf das Ausmaß und die Komplexität des Projekts dürfen alle Artefakte in den einzelnen in Bild 1 dargestellten Phasen modifiziert werden. Sobald die einzelnen Software-Entwicklungsphasen festgelegt worden sind, müssen die Artefakte für jede durchzuführende Phase definiert werden. Weitere Phasen und dazugehörige Artefakte dürfen durch Aufteilung der Aktivitäten und Aufgaben hinzugefügt werden.

ANMERKUNG Es ist üblich, einzelne Phasen zu kombinieren, wenn die verwendete Methode/Maßnahme eine klare Unterscheidung zwischen den Phasen erschwert. So können beispielsweise die Gestaltung der Software-Architektur und die Software-Implementierung nacheinander mit demselben rechnergestützten Entwicklungstools generiert werden, wie dies im modellbasierten Entwicklungsprozess der Fall ist.

Im Rahmen des Software-Entwicklungsprozesses müssen die Artefakte:

- a) entsprechend der von den geplanten Phasen erwarteten Ergebnisse dokumentiert werden;
- b) als Folge einer Auswirkungsanalyse modifiziert werden, und nur die beeinträchtigte Software muss regressionsgeprüft werden;
- c) einem Konfigurationsmanagementprozess unterworfen sein.

Das erste Artefakt, das für diesen Prozess angewendet wird, ist der Software-Entwicklungsplan. Die folgenden im Plan festgelegten Artefakte müssen enthalten:

- Gestaltungsspezifikation und dazugehörigen Verifizierungsbericht für jede Software-Gestaltungsphase (absteigender Ast des V-Modells in Bild 1);
- Prüfspezifikation und dazugehörigen Prüfbericht für jede Software-Designphase (aufsteigender Ast des V-Modells in Bild 1);
- ausführbare Software.

4.3 Spezifikation der Sicherheitsanforderungen an die Software

Die Spezifikation der Software-Sicherheitsanforderungen muss, falls zutreffend, die Sicherheitsanforderungen an die Software für Folgendes beschreiben:

- Funktionen, die es dem System ermöglichen, einen sicheren Zustand zu erreichen oder aufrechtzuerhalten;
- Funktionen zur Erkennung, Anzeige und Handhabung von Fehlern in den sicherheitsbezogenen Teilen des Steuersystems;
- Funktionen zur Erkennung, Anzeige und Handhabung von Fehlern in der Software;
- Funktionen für die Online- und Offline-Prüfungen der Sicherheitsfunktionen;

ANMERKUNG 1 Eine Online-Prüfung wird ausgeführt, während das geprüfte System in Betrieb ist. Eine Offline-Prüfung wird ausgeführt, während das geprüfte System nicht in Betrieb ist.

ANMERKUNG 2 Ein Beispiel für eine Online-Prüfung wäre die Fehlersuche im Steuersystem während des Fahrens der Maschine. Ein Beispiel für eine Offline-Prüfung wäre die Fehlersuche im Steuersystem, bevor die Maschine zur Bewegung freigegeben wird.

- Funktionen, die eine sichere Durchführung von Software-Modifikationen ermöglichen;
- Schnittstellen mit nicht sicherheitsbezogenen Funktionen;
- Leistung und Ansprechzeit;
- Schnittstellen zwischen Software und Hardware des elektronischen Steuergeräts.

Geeignete Methoden oder Maßnahmen müssen aus Tabelle 1 ausgewählt werden, um das angegebenen MPLr einzuhalten.

Tabelle 1 — Spezifikation der Software-Sicherheitsanforderungen

Methode/Maßnahme		Referenz	MPLr = a	MPLr = b, c	MPLr = d	MPLr = e
1.	Spezifikation der Anforderungen in natürlicher Sprache	A.1	+	+	+	+
2.	Rechnergestützte Spezifikationstools	A.2	o	o	o	+
3.a	Informelle Methoden	A.3	+	+	+	–
3.b	Halbformelle Methoden ^a	A.4	+	+	+	+

Methode/Maßnahme		Referenz	MPLr = a	MPLr = b, c	MPLr = d	MPLr = e
3.c	Formelle Methoden	A.5	+	+	+	+
4.	Nachverfolgbarkeit nach vorn zwischen den Systemsicherheitsanforderungen und den Software-Sicherheitsanforderungen	A.6	o	o	o	+
5.	Rückverfolgbarkeit zwischen den Software-Sicherheitsanforderung und den Systemsicherheitsanforderungen		o	o	o	+
6.a	Walkthrough der Software-Sicherheitsanforderungen	A.7	+	+	+	-
6.b	Inspektion der Software-Sicherheitsanforderungen	A.8	+	+	+	+
ANMERKUNG Die detaillierte Beschreibung dieser Methoden/Maßnahmen findet sich in Anhang A						

4.4 Gestaltung der Software-Architektur

Es muss eine Software-Architektur entwickelt werden, die basierend auf den Software-Sicherheitsanforderungen die hierarchische Struktur für alle sicherheitsbezogenen Software-Komponenten jedes SCS beschreibt. Geeignete Methoden oder Maßnahmen müssen aus Tabelle 2 ausgewählt werden, um das angegebene MPLr einzuhalten.

Tabelle 2 — Gestaltung der Software-Architektur

Methode/Maßnahme		Referenz	MPLr = a	MPLr = b, c	MPLr = d	MPLr = e
1.a	Informelle Methoden	A.3	+	+	-	-
1.b	Halbformelle Methoden	A.4	+	+	+	+
1.c	Formelle Methoden	A.5	+	+	+	+
2.	Rechnergestützte Gestaltungstools	A.9	o	o	o	+
3.a	Zyklisches Verhalten, mit garantierter maximaler Zykluszeit	A.10	o	o	+	+
3.b	Zeitgesteuerte Architektur		o	o	+	+
3.c	Ereignisgesteuert, mit garantierter maximaler Ansprechzeit		o	o	+	+
4.	Nachverfolgbarkeit nach vorn zwischen den Software-Sicherheitsanforderungen und der Software-Architektur	A.6	o	o	o	+
5.	Rückverfolgbarkeit zwischen der Software-Architektur und der Spezifikation der Software-Sicherheitsanforderungen		o	o	o	+
6.a	Walkthrough der Software-Architektur	A.7	+	+	+	-
6.b	Inspektion der Software-Architektur	A.8	+	+	+	+
ANMERKUNG Die detaillierte Beschreibung dieser Methoden/Maßnahme findet sich in Anhang A.						

4.5 Software-Modulgestaltung und Codierung

Die Ziele dieser Phase der Software-Entwicklung lauten:

- detaillierte Spezifizierung des Verhaltens der sicherheitsbezogenen Software-Module, die von der Software-Architektur vorgeschrieben sind;
- Erstellung lesbarer, prüfbarer und wartbarer Module (z. B. manueller Code, Modell usw.);
- Verifizierung, dass die Software-Architektur vollständig und richtig implementiert wurde.

Geeignete Methoden oder Maßnahmen müssen aus Tabelle 3 ausgewählt werden, um das angegebene MPLr einzuhalten. Es ist nicht erforderlich, automatisch erzeugten Code zu überprüfen;

Tabelle 3 — Software-Gestaltung und Codierung

Methode/Maßnahme		Referenz	MPLr = a	MPLr = b, c	MPLr = d	MPLr = e
1.a	Informelle Methoden	A.3	+	+	-	-
1.b	Halbformelle Methoden	A.4	+	+	+	+
1.c	Formelle Methoden	A.5	+	+	+	+
2.	Rechnergestütztes Gestaltungstool	A.9	o	o	o	+
3.	Verwendung von Gestaltungs- und Codierungsnormen	A.11	o	+	+	+
4.	Kein unstrukturierter Steuerungsablauf in Programmen in übergeordneten Sprachen ^b		o	o	+	+
5.	Begrenzte automatische Typkonvertierung ^b		o	o	+	+
6.	Eingeschränkte Verwendung von Interrupts ^b		o	o	o	+
7.	Festgelegte Verwendung von Zeigern ^b		o	o	o	+
8.	Eingeschränkte Verwendung von Rekursionen		o	o	o	+
9.a	Keine dynamischen Variablen oder Objekte ^b	A.12	o	o	+	+
9.b	Online-Prüfung der Erstellung dynamischer Variablen	A.13	o	o	+	+
10.	Größenbeschränkung von Software-Modulen	A.14	+	+	+	+
11.	Ein Eintritts-/Austrittspunkt in Unterprogrammen und Funktionen ^b		o	+	+	+
12.	Vollständig festgelegte Schnittstelle		o	+	+	+
13.	Verbergen/Kapseln von Informationen		o	o	+	+
14.	Software-Komplexitätskontrolle		o	o	o	+
15.	Strukturierte Gestaltung oder Codierung	A.15	o	+	+	+
16.	Defensive Gestaltung oder Codierung	A.16	o	o	o	+
17.	Verwendung vertrauenswürdiger/verifizierter Software-Elemente ^a	A.17	o	o	o	o
18.	Nachverfolgbarkeit nach vorn zwischen den Software-Sicherheitsanforderungen und der Software-Gestaltung	A.6	o	o	o	+
19.a	Walkthrough der Software-Gestaltung, des Quellcodes oder von beidem	A.7	+	+	+	-
19.b	Inspektion der Software-Gestaltung, des Quellcodes oder von beidem	A.8	+	+	+	+
ANMERKUNG Die detaillierte Beschreibung dieser Methoden/Maßnahmen findet sich in Anhang A. ^a Wenn vertrauenswürdige und verifizierte Software-Elemente verfügbar sind, wird dies dringend empfohlen. ^b Diese Methoden oder Maßnahmen sind nicht immer für in der modellbasierten Entwicklung verwendete grafische Modellierungsnotationen anwendbar.						

4.6 Sprache, Bibliothek und Tool-Auswahl

Die Sicherheitsintegrität der zu entwickelnden Software kann direkt von der ausgewählten Programmiersprache, dem für Entwicklung und Prüfung verwendeten Tool und der Verwendung bestehender, vertrauenswürdiger, verifizierter Software-Module beeinflusst werden. Geeignete Methoden oder Maßnahmen müssen aus Tabelle 4 ausgewählt werden, um das angegebene MPLr einzuhalten.

Tabelle 4 — Tools und Programmiersprache

Methode/Maßnahme		Referenz	MPLr = a	MPLr = b, c	MPLr = d	MPLr = e
1.	Geeignete Programmiersprache	A.18	+	+	+	+
2.	Unterstützung für Sprachteilmenge	A.19	0	o	o	+
3.a	Tools und Übersetzungsprogramme mit zunehmender Bewährtheit im Betrieb oder in der Validierung	A.20	0	+	+	+
3.b	Zertifizierte Tools und Übersetzungsprogramme	A.21	0	+	+	+
ANMERKUNG Die detaillierte Beschreibung dieser Methoden/Maßnahmen findet sich in Anhang A.						

4.7 Prüfen von Software-Modulen

Software-Modulprüfungen haben zum Ziel, zu verifizieren, dass die entworfenen und implementierten Software-Module die Software-Sicherheitsanforderungen erfüllen. In dieser Phase muss ein Verfahren für das Prüfen der Software-Module gegen ihre Anforderungen erstellt und die Prüfungen entsprechend diesem Verfahren durchgeführt werden.

Für eine systematische Herangehensweise an die Prüfung der Software hilft die Verwendung geeigneter Tools für die Prüfungsverwaltung und Prüfungsautomatik die arbeitsintensiven und fehleranfälligen Aufgaben beim Prüfen der Software. Da Unterstützung verfügbar ist, bietet sich eine umfassendere Herangehensweise sowohl an normale als auch Regressionsprüfungen an.

Für eine einfachere Verifizierung, Validierung, Bewertung und Wartung sollten alle Daten, Entscheidungen und Begründungen im Software-Projekt dokumentiert werden. Die Dokumentation der Software-Module sollte Folgendes beinhalten:

- durchgeführte Prüfungen;
- Entscheidungen und deren Begründung;
- Probleme und deren Lösungen.

ANMERKUNG Die Aufzeichnung von Daten ist wichtig für die Wartung von Computersystemen, da die Begründung für bestimmte während des Entwicklungsprojekts getroffene Entscheidungen den Wartungsingenieuren nicht immer bekannt ist.

Geeignete Methoden oder Maßnahmen müssen aus **Software-Einheiten** ausgewählt werden. Die Prüfung darf in verschiedenen Umgebungen durchgeführt werden, zum Beispiel:

- Model-in-the-Loop-Prüfungen,
- Software-in-the-Loop-Prüfungen,
- Processor-in-the-Loop-Prüfungen,
- Hardware-in-the-Loop-Prüfungen.

Tabelle 5 zur Einhaltung des festgelegten MPLr.

Die Prüfung von Software-Einheiten darf in verschiedenen Umgebungen durchgeführt werden, zum Beispiel:

- Model-in-the-Loop-Prüfungen,
- Software-in-the-Loop-Prüfungen,
- Processor-in-the-Loop-Prüfungen,
- Hardware-in-the-Loop-Prüfungen.

Tabelle 5 — Prüfung von Software-Modulen

Methode/Maßnahme		Referenz	MPL r = a	MPLr = b, c	MPL r = d	MPLr = e
1.	Grenzwertanalyse	A.22	o	o	+	+
2.	Kontrollflussanalyse	A.23	o	o	+	+
3.	Datenflussanalyse	A.24	o	o	+	+
4.	Prüffallausführung anhand der Grenzwertanalyse	A.25	o	o	o	+
5.	Funktions-/Black-Box-Prüfung [einschließlich Fehlersimulationsprüfung (FI, en: Fault Insertion)]	A.26	+	+	+	+
6.a	Umfang der Strukturprüfung (Eintrittspunkte)	A.27	o	+	–	–
6.b	Umfang der Strukturprüfung (Anweisungen)		o	+	+	–
6.c	Umfang der Strukturprüfung (Verzweigungen)		o	+	+	+
7.	Äquivalenzklassen und Eingabe-Partitionsprüfungen ^a	A.28	o	o	+	+
8.	Prüffallausführung aus modellbasierter Prüffallgenerierung	A.29	o	o	o	+
9.	Ansprechzeiten und Speicherbeschränkungen ^a	A.30	o	+	+	+
10.	Leistungsanforderungsprüfung ^a		o	+	+	+
11.	Belastungs-/Stressprüfung ^a		o	o	o	+
12.	Software-Modulschnittstellenprüfung	A.31	o	o	o	+
13.	Direkte Vergleichsprüfung	A.32	o	o	+	+
14.	Nachverfolgbarkeit nach vorn zwischen der Software-Modulgestaltung und den Modul-Prüfspezifikationen	A.6	o	o	o	+
ANMERKUNG Die detaillierte Beschreibung dieser Methoden/Maßnahmen findet sich in Anhang A. ^a Der Prüfer kann sich dafür entscheiden, dieses Prüfverfahren nicht durchzuführen, muss es aber auf Integrationsebene durchführen, wenn es obligatorisch erforderlich ist.						

4.8 Software-Modulintegration und -prüfung

Die Ziele dieser Phase der Software-Entwicklung lauten:

- Integration der Software-Module in Software-Komponenten über die gesamte systemintegrierte Software des SCS;
- Verifizierung, dass die Anforderungen an die Software von der systemintegrierten Software korrekt umgesetzt werden.

In dieser Phase werden die einzelnen Integrationsebenen auf die Software-Sicherheitsanforderungen geprüft. Die Schnittstellen zwischen den Software-Modulen und zwischen Software-Modulen und Software-Komponenten werden ebenfalls geprüft. Die Integrationsschritte und Prüfungen der Software-Komponenten müssen direkt der hierarchischen Architektur der Software entsprechen.

Geeignete Methoden/Maßnahmen müssen aus Tabelle 6 ausgewählt werden, um das angegebene MPLr einzuhalten. Der Prüfer kann sich jedoch dafür entscheiden, ein Prüfverfahren oder eine Maßnahme nicht durchzuführen, muss sie aber auf Modulebene durchführen, wenn es obligatorisch erforderlich ist.

Eine Software-Integrationsprüfung darf in verschiedenen Umgebungen durchgeführt werden, zum Beispiel:

- Model-in-the-Loop-Prüfungen,
- Software-in-the-Loop-Prüfungen,
- Processor-in-the-Loop-Prüfungen,
- Hardware-in-the-Loop-Prüfungen.

Tabelle 6 — Software-Modulintegration und -prüfung

Methode/Maßnahme		Referenz	MPLr = a	MPLr = b, c	MPLr = d	MPLr = e
1.	Funktions-/Black-Box-Prüfung [einschließlich Fehlersimulationsprüfung (FI)]	A.26	+	+	+	+
2.	Äquivalenzklassen und Eingabe-Partitionsprüfungen	A.28	0	o	+	+
3.	Ansprechzeiten und Speicherbeschränkungen	A.30	0	+	+	+
4.	Leistungsanforderungsprüfung		0	+	+	+
5.	Belastungs-/Stressprüfung		0	o	o	+
6.	Direkte Vergleichsprüfung	A.32	0	o	+	+
7.	Nachverfolgbarkeit nach vorn zwischen der Software-Architekturgestaltung und den Prüfspezifikationen für Integration	A.6	0	o	o	+
ANMERKUNG Die detaillierte Beschreibung dieser Methoden/Maßnahmen findet sich in Anhang A.						

4.9 Software-Validierung

Das Ziel dieser Phase der Software-Entwicklung ist der Nachweis, dass die Software-Sicherheitsanforderungen von der systemintegrierten Software korrekt umgesetzt sind.

Diese Prüfung ist die wichtigste Software-Verifizierungsmethode. Animation und Modellierung dürfen als Ergänzung der Verifizierungsaktivitäten verwendet werden.

Die Software muss durch Simulation von Folgendem ausgeführt werden:

- im Normalbetrieb vorhandene Eingabesignale;
- erwartete Vorkommnisse;
- unerwünschte Umstände, die eine Systemaktion erfordern.

Die Wirksamkeit der angewendeten Prüfverfahren und aller anderen verwendeten Maßnahmen muss anhand des Sicherheitskonzepts nach Abschluss des Verifizierungsverfahrens ausgewertet werden, um die Software durch Prüfung im Hinblick auf die Sicherheitsanforderungen zu validieren.

Geeignete Methoden/Maßnahmen müssen aus Tabelle 7 ausgewählt werden, um das angegebene MPLr einzuhalten.

Tabelle 7 — Software-Validierung

Methode/Maßnahme		Referenz	MPLr = a	MPLr = b, c	MPLr = d	MPLr = e
1.a	Maschinennetzwerkprüfung	B.1	+	+	+	+
1.b	Hardware-in-the-Loop-Prüfung	B.2	+	+	+	+
1.c	Prüfung der Maschinenebene	B.3	+	+	+	+
2.	Nachverfolgbarkeit nach vorn zwischen den Software-Sicherheitsanforderungen und dem Software-Verifizierungsplan (einschließlich Datenverifizierung)	A.6	o	o	o	+
3.	Rückverfolgbarkeit zwischen dem Software-Verifizierungsplan (einschließlich Datenverifizierung) und den Software-Sicherheitsanforderungen		o	o	o	+
ANMERKUNG Die detaillierte Beschreibung dieser Methoden/Maßnahmen findet sich in Anhang B.						

5 Softwarebasierte Parametrierung

5.1 Allgemeines

Die softwarebasierte Parametrierung bezieht sich auf die Möglichkeit, das Software-System nach Abschluss der Entwicklung an unterschiedliche Anforderungen anzupassen, indem Parameter zur Modifizierung der Funktionalität der Software geändert werden. Die softwarebasierte Parametrierung sicherheitsbezogener Parameter sollte als Teil des SCS betrachtet werden, und muss in der Spezifikation der Software-Sicherheitsanforderungen beschrieben werden.

Softwarebasierte Parameter umfassen:

- Variantencodierung (z. B. Ländercode, Links-/Rechtslenkung usw.);
- Parameter (z. B. Wert für niedrige Leerlaufdrehzahl, Motorkennliniendiagramme usw.);
- Kalibrierungsdaten (z. B. fahrzeugspezifisch, Endanschlag für Gaseinstellung usw.).

5.2 Datenintegrität

Die Integrität der für die Parametrierung verwendeten Daten muss erhalten bleiben und Modifizierungen durch Unbefugte müssen verhindert werden. Dies muss durch Anwendung von Methoden oder Maßnahmen erreicht werden, um Folgendes zu überwachen:

- den Bereich gültiger Eingaben;
- Datenkorruption vor und nach der Übertragung;
- Fehler aus dem Parameterübertragungsprozess;
- Auswirkungen einer unvollständigen Parameterübertragung und
- Fehler- und Ausfallwirkungen der Hardware und Software des für die Parametrierung verwendeten Tools.

5.3 Verifizierung softwarebasierter Parametrierung

Folgende Verifizierungsaktivitäten müssen für die softwarebasierte Parametrierung durchgeführt werden:

- Verifizierung der korrekten Einstellung jedes sicherheitsbezogenen Parameters (Minimum, Maximum und repräsentative Werte);
- Verifizierung, dass die sicherheitsbezogenen Parameter durch Verwendung ungültiger in die Software eingeschriebener Werte während der Konfigurationsphase zur Verifizierung deren Verhaltens auf Plausibilität überprüft wurden;
- Verifizierung, dass eine unbefugte Modifizierung sicherheitsbezogener Parameter verhindert wird;
- Verifizierung, dass die Daten/Signale für die Parametrierung so generiert und verarbeitet werden, dass Fehler nicht zu einem Verlust der sicherheitsbezogenen Funktionen führen können.

6 Schutz der Übertragung sicherheitsbezogener Nachrichten in Bussystemen

In diesem Abschnitt werden Empfehlungen für die Übertragung in SCS und in der Kommunikation verwendeten sicherheitsbezogenen Nachrichten gegeben, die zwischen verschiedenen Komponenten (z. B. Mikrocontroller, intelligente Sensoren, intelligente Aktoren) eines SCS stattfinden kann.

ANMERKUNG 1 Zu diesem Zeitpunkt werden nur diese gekapselten Bussysteme berücksichtigt, bei denen der Hersteller Anzahl und Typ der Busteilnehmer (d.h. an den Bus angeschlossene Einheiten) festgelegt hat. Eine Erweiterung dieses Systems auf Datenübertragung über lange Distanzen wird hier nicht berücksichtigt. Interne Daten- und Adressenbusse zählen nicht zum Anwendungsbereich.

Steuerungsübertragungsfehler, -methoden und -maßnahmen müssen nach IEC 61508-2:2010, Anhang A, Tabelle A-1 implementiert werden, um festzulegen, welche Teile erforderlich sind, um sicherzustellen, dass keine der anwendbaren Fehler die korrekte Durchführung der sicherheitsbezogenen Funktionen behindert:

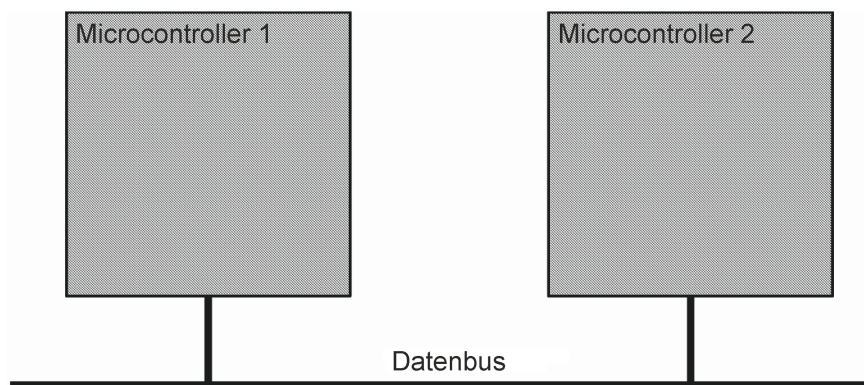


Bild 2 — Mehrere Partitionen innerhalb eines ECU mit mehreren Prozessoren

In Tabelle 8 werden die Methoden und Maßnahmen zum Schutz vor den oben gelisteten Übertragungsfehlern definiert, indem bestimmte Fehlerwirkungen ausgeschlossen werden, die in Verbindung mit der Kommunikation oder dem Schutz der sicherheitsbezogenen Daten im SCS stehen.

Tabelle 8 — Methoden und Maßnahmen innerhalb eines Mikrocontroller-Netzwerks

			Übertragungsfehler siehe § 6								
			Ausfall der Partnerkommunikation	Nachrichtenviederholung	Nachrichtenvverlust	Einfügung von Nachrichten	Falsche Abfolge	Falsifizierung von Nachrichten	Nachrichtenzögerung	Zugangssperre zum Datenbus	Ständige Übertragung von Nachrichten
Methode/ Maßnahme	D.1	Keep-Alive-Nachrichten	JA	NEIN	NEIN	NEIN	NEIN	JA	NEIN	NEIN	NEIN
	D.2	Alive Counter	NEIN	JA	JA	JA	NEIN	NEIN	NEIN	NEIN	JA
	D.3	CRC	NEIN	NEIN	NEIN	NEIN	NEIN	JA	NEIN	NEIN	NEIN
	D.4	Sequenznummer	NEIN	JA	JA	JA	JA	NEIN	NEIN	NEIN	JA
	D.5	Nachrichtenviederholung	NEIN	NEIN	JA	NEIN	JA	JA	NEIN	NEIN	NEIN
	D.6	Watchdog	JA	NEIN	JA	JA	NEIN	NEIN	JA	JA	JA
	D.7	Zeitgesteuerter Datenbus	JA	NEIN	NEIN	JA	NEIN	NEIN	JA	NEIN	NEIN
	D.8	Buswächter	NEIN	NEIN	NEIN	NEIN	NEIN	JA	NEIN	NEIN	JA
	D.9	Mini-slotting	NEIN	NEIN	NEIN	NEIN	NEIN	NEIN	JA	NEIN	JA

ANMERKUNG Begriffe definieren

- Ausfall der Partnerkommunikation, d. h. Kommunikationspartner ist nicht verfügbar;
- Unbeabsichtigte Nachrichtenwiederholung, d. h. die gleiche Nachricht wird versehentlich erneut versendet;
- Nachrichtenverlust, d. h. die Nachricht geht während der Übertragung verloren;
- Unbeabsichtigte Einfügung von Nachrichten, d. h. der Empfänger erhält versehentlich eine zusätzliche Nachricht, die als mit einer korrekten Quelle und Zieladresse versehen angesehen wird.
- Falsche Abfolge, d. h. Reihenfolge, in der Daten versendet wurden, änderte sich während der Übertragung, d. h. die Daten werden nicht in derselben Reihenfolge empfangen, in der sie versendet wurden;
- Falsifizierung von Nachrichten, d. h. ein oder mehrere Datenbits werden während der Übertragung verändert;
- Nachrichtenverzögerung, d. h. die Nachricht wird zwar korrekt, aber nicht rechtzeitig empfangen;
- Zugangssperre zum Datenbus, d. h. ein fehlerhafter Knoten hält sich nicht an die erwarteten Nutzungsmuster und fordert übermäßig Dienste an; damit verringert er die für die anderen Knoten zur Verfügung stehenden Dienste, z. B. während sie fälschlicherweise auf nicht existierende Daten warten;
- Ständige Übertragung von Nachrichten, auch bekannt als „Babbling Idiot“, d. h. ein fehlerhafter Knoten überträgt ständig und beeinträchtigt dadurch den Betrieb des gesamten Bus.

Die Einhaltung des MPLr durch in der Netzwerkkommunikation umgesetzte Maßnahmen kann durch die Berechnung der Datenintegritätssicherung überprüft werden, wie in Anhang C.1 angegeben.

7 Unabhängigkeit durch Software-Partitionierung

Die Software-Partitionierung ist dafür vorgesehen, den Planer durch den Nachweis der Unabhängigkeit von Software-Modulen zu unterstützen. Eine ausreichende Unabhängigkeit von Software-Komponenten wird dadurch garantiert, dass bestimmte Softwarefehlerwirkungen, die diese Unabhängigkeit verletzen, ausgeschlossen werden.

Hierzu müssen Methoden und Maßnahmen mit zumindest mittlerer Effektivität umgesetzt werden:

- zur Beherrschung von Gefährdungen, die in Teilsystemen auftreten können, sodass sie andere Teilsysteme nicht negativ beeinflussen können;
- zur Gewährleistung ausreichender Unabhängigkeit von Software-Komponenten durch Software-Partitionierung.

Zum Erreichen einer ausreichenden Unabhängigkeit von Software-Komponenten sollten die Systemressourcen unabhängigen Partitionen, von denen jede eine bestimmte Laufzeitumgebung darstellt, zugewiesen werden. Der Einsatz der Software-Partitionierung ist nicht auf die Koexistenz von Software mit unterschiedlichem MPLa in derselben Laufzeitumgebung beschränkt. Sie kann auch Folgendes unterstützen:

- a) Änderungen einer Partition ohne Reverifizierung der unveränderten Software-Partitionen;
- b) Koexistenz von Software einer anderen Art (innerbetrieblich, von Dritten).

Für Software-Partitionen mit einem MPLa = c, d, e ist Unabhängigkeit erforderlich.

ANMERKUNG Partitionen können in einem einzelnen Mikrocontroller oder mehreren Mikrocontrollern in einem Mikrocontroller-Netzwerk zugewiesen werden.

Je nach gewählter Architektur können zwei Ansätze verfolgt werden:

- a) mehrere Partitionen in einem einzelnen Mikrocontroller;
- b) mehrere Partitionen im Rahmen eines ECU-Netzwerks (elektronisches Steuerungssystem).

Das Konzept der Software-Partitionierung und zugehöriger Methoden und Maßnahmen für die Unabhängigkeit von Software-Modulen muss bei der Spezifikation der Software-Architektur berücksichtigt werden.

Derjenige Teil der Software, der die Unterstützung zur Partitionierungsimplementierung implementiert, muss dasselbe oder ein höheres MPLa als das mit den Software-Partitionen verbundene höchste MPLa aufweisen.

7.1 Mehrere Partitionen in einem einzelnen Mikrocontroller

Um eine angemessene Unabhängigkeit der Software-Komponenten innerhalb eines einzelnen Mikrocontrollers zu erreichen, muss die korrekte Ausführung der sicherheitsbezogenen Funktionen vor jeder der folgenden Fehlerwirkungen geschützt werden:

- Speicherverfälschung (unbeabsichtigtes Schreiben in den Speicher einer anderen Partition);
ANMERKUNG Speicherverfälschung kann bei Memory Mapped E/A, Speicherregistern usw. auftreten;
- Blockieren von Partitionen (aufgrund von Kommunikationsblockaden);
- falsche Zuweisung der Prozessorausführungszeit;
- falscher Kommunikationspartner (der Absender sendet Nachrichten an den falschen Empfänger oder missbraucht die Identifikation eines anderen Absenders);
- Verfälschung der E/A-Schnittstelle durch unbeabsichtigtes Schreiben an eine E/A-Schnittstelle einer anderen Partition.

ANMERKUNG Verfälschung der E/A-Schnittstelle bezieht sich nur auf externe Geräte.

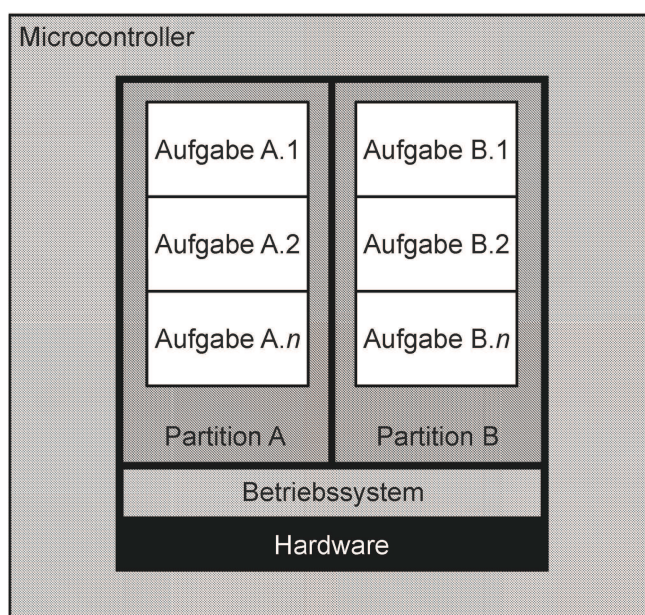


Bild 3 — Mehrere Partitionen in einem einzelnen Mikrocontroller

Die folgende Tabelle 9 definiert die Maßnahmen so, dass alle in § 7.1 angeführten entsprechenden Fehlerwirkungen mindestens mit mittlerer Wirksamkeit behandelt werden können, um eine angemessene Unabhängigkeit der Software-Komponenten sicherzustellen.

Tabelle 9 — Methoden und Maßnahmen innerhalb des Mikrocontrollers

			Fehlerwirkung siehe § 7.1				
			Speicher- ver- fälschung	Blockieren von Partitionen	Falsche Zuweisung der Prozessor- ausführungs- zeit	Falscher Kommu- nikations- partner	Verfälschung der E/A- Schnittstelle
Me- thode/ Maß- nahme	E.1	Eindeutiges bidirektionales Kommunikationsobjekt	NEIN	NEIN	NEIN	JA	NEIN
	E.2	Ausschließlich zwei unidirektionale Kommunikations- objekte	NEIN	NEIN	NEIN	JA	NEIN
	E.3	IDs zur Identifizierung, Quittierung oder beidem.	NEIN	NEIN	NEIN	JA	NEIN
	E.4	Asynchrone Datenkommunikation	NEIN	JA	NEIN	NEIN	NEIN
	E.5	Streng prioritätsbasierte Planung	NEIN	NEIN	JA	NEIN	NEIN
	E.6	Zeitscheibenmethode	NEIN	NEIN	JA	NEIN	NEIN
	E.7	Speicherschutz- mechanismen	JA	NEIN	NEIN	NEIN	JA
	E.8	Verifizierung sicherheitskritischer Daten	JA	NEIN	NEIN	NEIN	NEIN
	E.9	Statische Analyse	JA	NEIN	NEIN	NEIN	NEIN
	E.10	Statische Zuweisung	JA	JA	JA	JA	JA

7.2 Mehrere Partitionen im Rahmen eines ECU-Netzwerks

Um eine passende Netzwerkkommunikation zu garantieren, wenn die Unabhängigkeit der Software-Komponenten mittels eines Mikrocontroller-Netzwerks umgesetzt wird, siehe Abschnitt 6.

Anhang A (informativ)

Beschreibung der Software-Methoden/Maßnahmen

A.1 Spezifikation der Anforderungen in natürlicher Sprache

Die Spezifikation der Software-Sicherheitsanforderung sollte eine Beschreibung des Problems in natürlicher Sprache sowie, falls erforderlich, weitere informelle Methoden, z. B. Bilder und Diagramme, umfassen.

A.2 Rechnergestützte Spezifikationstools

Diese Tools sollten angewendet werden, um das automatische Erkennen von Mehrdeutigkeiten und Vollständigkeit in halbformellen und formellen Spezifikationsmethoden zu erleichtern. Dieses Tool sollte Spezifikationen in Form einer Datenbank erstellen, die zur Bewertung der Konsistenz und Vollständigkeit automatisch untersucht werden kann. Im Allgemeinen unterstützt diese Methode nicht nur die Erstellung der Spezifikation, sondern auch des Entwurfs und anderer Lebenszyklusphasen des Projekts.

A.3 Informelle Methoden

Informelle Methoden liefern ein Mittel, um eine Beschreibung des Systems auf irgendeiner Stufe seiner Entwicklung zu erstellen, d. h. Spezifikation, Gestaltung oder Codierung mit Hilfe von natürlicher Sprache, Diagrammen, Bildern usw.

A.4 Halbformelle Methoden

Halbformelle Gestaltungsmethoden müssen ein Konzept, eine Spezifikation oder eine Gestaltung eindeutig und konsequent darstellen, sodass einige Fehler und Auslassungen erkannt werden können.

In manchen Fällen sollte die Beschreibung maschinell analysiert oder zur Anzeige verschiedener Aspekte des Systemverhaltens animiert werden. Die Animation kann das zusätzliche Vertrauen schaffen, dass das System die Anforderungen auch erfüllt.

Beispiele halbformeller Methoden beinhalten, sind aber nicht begrenzt auf: Datenflussdiagramme, Pseudo-Codierung, Modellierungstools, endlicher Automat/Zustandsübergangsdiagramme.

A.5 Formelle Methoden

Formelle Methoden bieten eine Möglichkeit, eine Systembeschreibung in strikter Notation zu entwickeln, die einer mathematischen Analyse unterzogen werden sollte, um verschiedene Klassen von Inkonsistenzen oder Fehlern zu erkennen. Darüber hinaus sollte die Beschreibung in einigen Fällen maschinell mit einer Genauigkeit analysiert werden, die der Syntaxprüfung eines Quellprogramm-Compilers ähnlich ist, oder animiert werden, um verschiedene Aspekte des Verhaltens des beschriebenen Systems darzustellen. Die Animation kann zusätzliches Vertrauen schaffen, dass das System die tatsächliche Anforderung sowie die formal spezifizierte Anforderung erfüllt, weil sie das Erkennen des spezifizierten Verhaltens durch den Menschen erleichtert.

Eine formelle Methode bietet allgemein eine Notation (für gewöhnlich wird eine Art diskreter Mathematik verwendet), eine Methode zur Herleitung einer Beschreibung in dieser Notation sowie mehrere Analyseformen zur Prüfung einer Beschreibung auf verschiedene Korrektheitseigenschaften.