IEC 61508 konforme Testfallerstellung nach SIL 3

Eugen Sawin
esawin@stud.fh-offenburg.de

6. Juni 2009

Mit zunehmender Automatisierung von Sicherheitsprozessen wächst die Komplexität der Systeme und deren Test- und Zertifizierungsmaßnahmen. Programmierbare, elektronische Systeme werden zunehmend zur Realisierung von Sicherheitstechnik eingesetzt, dafür sind angepasste Verfahren notwendig, um die Korrektheit und Sicherheit der Komponenten zu bewerten. Der IEC 61508 Standard bietet eine Grundlage für die Entwicklung und Bewertung von sicherheitskritischer Hard- und Software. Dieser Text beschäftigt sich mit den notwendigen Maßnahmen für eine erfolgreiche Verifikation der Validierung einer Software nach Safety Integrity Level 3. Dies kann als Grundlage dienen zur Entwicklung eines standardkonformen Testphasenplans und Entscheidungshilfen bieten bei der Auswahl einer Strategie zur Validierung eines Softwaremoduls.

1 Einführung

Mit den höheren Ansprüchen an Sicherheit, steigen auch die Kosten für deren Erfüllung. Dabei haben die Personalkosten einen erheblichen Anteil daran. Durch die Automatisierung der Sicherheitsbereitstellung, werden die Prozesse wirtschaftlicher und dynamischer z.B. bezüglich des Einsatzortes oder des Umfangs. Damit wächst gleichzeitig auch die Komplexität der Systeme, die zunehmend durch programmierbare, elektronische Systeme realisiert werden. Zur Gewährleistung der Sicherheit von sicherheitstechnischen Komponenten wird u.a. der IEC[1] 61508 Standard verwendet, der die notwendigen Schritte in jeder Phase eines Projekts beschreibt, die zur Zertifizierung notwendig sind.

IEC 61508 gilt als eine allgemeine Norm für die Planung, Entwicklung, Validieren, Erweiterung, Instandhaltung und Beseitigung von E/E/PE[2] Systemen mit Sicherheitsfunktion. Er wird genau dann eingesetzt, wenn kein Sektorstandard, wie z.B. IEC 61511 für die Prozessindustrie oder IEC 50129 für Bahnanwendungen, vorliegt. Eine Ausnahme sind Subsysteme, die zwar in einem Gesamtsystem einbettet sind mit geltendem Sektorstandard, jedoch als isolierte Einzelkomponenten sektorunspezifische Funktionalität realisieren. Diese werden ebenfalls nach dem IEC 61508 realisiert, sofern sie eine sicherheitskritische Funktion darstellen.

2 Fehlerursachen

Einfach ist sicher. Steigende Anforderungen an heutige Elektronik führt zu komplexen Entwürfen und somit zu größeren potentiellen Fehlerquellen. Es gilt das Problem in geeignet proportionierte, primitive Probleme zu zerlegen und diese durch modulare Bausteine zu lösen. Sicherheit durch Einfachheit bedeutet dabei nicht nur der modulare Aufbau eines Systems durch verlässliche Komponenten, sondern ist auch ein sozialer Prozess, der die Planung des Systems, die Wahl der Technologie, die Art der Dokumentation und des Entwicklungsprozesse betrifft.

Wenn man von der Komplexität einer Software spricht, unterscheidet man unter der struktu-

[1] International Electrotechnical Commission

[2] elektrische, elektronische und programmierbar elektronische (Systeme)

rellen und kognitiven Komplexität [Sho08]. Der strukturelle Aufbau lässt sich durch Metriken wie McCabe Cylomatic Complexity oder Halstead beurteilen und anhand dessen die möglichen komplexen und somit unsicheren Stellen lokalisieren und beseitigen. Zur Beurteilung der kognitiven Komplexität hingegen stehen keine Metriken zur Verfügung, diese beziehen sich auf die Verständlichkeit einer Software unter Berücksichtigung der durchschnittlichen kognitiven Fähigkeiten eines Menschen, wie z.B. der Kurzzeitgedächtnisleistung. Denn auch ein korrektes Programm ist auf lange Sicht unsicher, wenn es nicht problemlos von verschiedenen Entwicklern gewartet und angepasst werden kann.

Laut [Sho08] entstehen ca. 44% aller Unfälle in Verbindung mit Softwarefehler durch ungenügende Spezifizierung. Dabei kann der Begriff Fehler irreführen, denn in den meisten Fällen funktioniert die Software fehlerfrei, die Ursache liegt bereits im falschen Design. Aufgrund von unzureichendem Verständnis für die Einsatzumgebung und den Zweck der Software entstehen falsche Annahmen und Anforderungen, die bei Funktionstests nicht als Fehler erkannt werden.

2.1 Techniken zur Vermeidung von Fehlern

[IEC05] gibt zur Vermeidung von Fehlern und falschen Entscheidungen eine Reihe von Techniken vor. Für SIL[3] 3 gibt es folgende Richtlinien für das Vermeiden von Fehlentwicklungen während der

Anforderungsspezifikation:

1. Ein korrekt ausgeführtes Projektmanagement.
2. Sorgfältige Dokumentation.
3. Zur Reduzierung der Komplexität bei der Risikobewertung sollen sicherheitstechnische von nicht sicherheitsrelevanten Funktionen isoliert betrachtet werden. Letztere werden nicht vom Standard[4] betrachtet und benötigen keine Zertifizierung.
4. Die strukturierten Spezifikationsdokumente müssen zur Abnahme inspiziert werden. Hierbei reichen semi-formale Methoden aus zur Evaluierung der Korrektheit der Spezifikation.
5. Zur weiteren Einschränkung möglicher Fehlerquellen während der Anforderungsspezifikation stehen noch die Methoden der Checklistenführung für ausstehende/abgearbeitete Aufgaben, computergestützte Spezifizierungswerkzeuge und formale Beweise zur Wahl. Hierbei ist das Anwenden einer Methode ausreichend.

[3] Safety Integrity Level
[4] IEC 61508

3 Allgemeine Anforderungen

3.1 Dokumentenführung

[IEC05] beschreibt eine Reihe von Anforderungen an jede Phase eines Produktlebenszyklus. Die Wahl eines passenden Prozessmodells ist entscheidend, um später den dokumentarischen Nachweis für den erfolgreichen Abschluss einer Phase zu erbringen. Ein geeignetes Prozessmodell ist das V-Modell (Abb. 1), welches die Spezifikation klar von der Realisierung trennt und entsprechend dokumentiert.

Die Dokumente müssen versioniert sein und in einer erkennbaren Dokumentenstruktur untergebracht sein. Bei komplexen Systemen kann die Dokumentenstruktur nach Zielgruppen orientiert sein. Zu jedem Zeitpunkt in einem Projekt, muss gewährleistet werden, dass jede Zielgruppe Zugang zu entsprechenden Dokumenten erhält.

3.2 Middleware und Werkzeuge

Beim Einsatz von Middleware jeglicher Art, ist der Einsatz stets zu rechtfertigen. Sobald Module von dritter Hand direkt an einer Sicherheitsfunktion beteiligt sind, sind diese ebenfalls der Zertifizierung unterworfen und müssen somit bei der Verifizierung berücksichtigt werden.

Bei eingesetzten Compilern muss ein Nachweis der Konformität zu dem entsprechenden Standard erstellt werden. Dieser wird für die meisten Programmiersprachen durch sog. Validierung Suites, wie z.B. Plum-Hall Validierung Suite für C und C++, erbracht [Mor07].

Wird modellgetriebene Entwicklung praktiziert, müssen entsprechend eingesetzte Code-Generatoren zertifiziert sein.

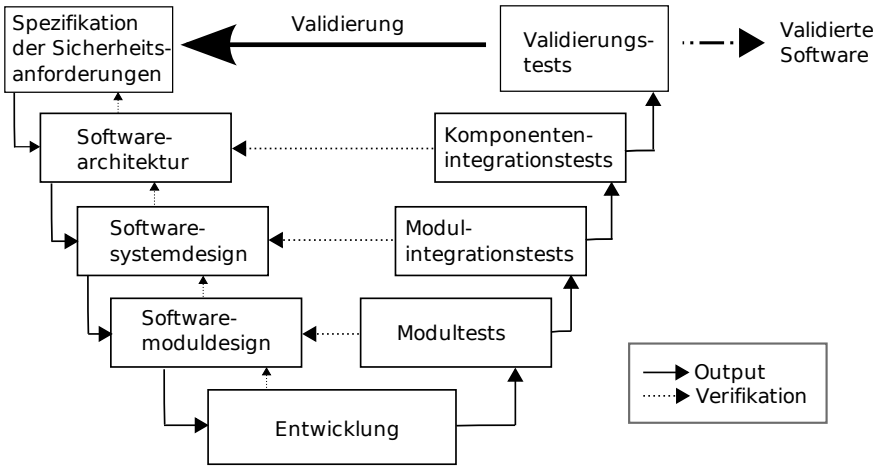

Abbildung 1: Softwaresicherheitsintegrität und die Entwicklungsphasen nach dem V-Modell (Quelle: [IEC05] Teil 3 S. 27)

3.3 Personalkompetenz

Involvierte Person in einer oder mehreren Aktivitäten im Entwicklungsprozess müssen entsprechendes Fachwissen, Erfahrung und Qualifikation im relevanten Tätigkeitsfeld vorweisen. Sowohl Managementfunktionen als auch technische Leistungen müssen mit entsprechend geeignetem Personal besetzt sein.

Die Ansprüche an das Personal steigen mit der Rigorosität des Systems und der Einstufung des Safety Integrity Levels. Mit SIL 3 ist somit wichtig Personen mit einschlägiger Erfahrung zu wählen, die sie neben hoher Kompetenz in der Softwareentwicklung mitbringen. Sie haben auch entsprechende Kenntnisse im Umgang mit den eingesetzten Werkzeugen, in Safety Engineering und auch dem regulatorischen Framework vorzuweisen.

[May02] beschreibt ein Kompentenzmodell zur Evaluierung möglicher Kandidaten. Man allokiert für jede Sicherheitsfunktion bis zu fünfzehn zielorientierte Kompetenzen, die am wichtigsten sind für eine erfolgreiche Ausführung der sicherheitsrelevanten Funktion. Die Einteilung wird erleichtert, indem man die Kompetenzen in aufgabenspezifische und funktionsabhängige teilt. Aufgabenspezifische Fähigkeiten sind meist technischer Natur, während funktionale Kompetenzen sich auf das Verständnis und Verhalten beziehen [May02].

4 Analyse

4.1 Gefahrenanalyse (Hazard Analysis)

Die Gefahrenanalyse soll - laut Standard - die Funktionen identifizieren, die zur Sicherung aller möglichen gefährlichen Ereignisse geeignet sind. Solch eine Funktion wird auch als Sicherheitsfunktion bezeichnet. Dabei ist das vorsorgliche Vermeiden des gefahrbringenden Ereignisses, durch eine Umgestaltung des Systems, keine Instanz von funktionaler Sicherheit, ist aber ebenfalls als Auflösung eines Hazards geeignet.

4.1.1 Quantitative Determination des SIL

[HMA07], [IEC05] beschreiben eine Möglichkeit der Klassifizierung von Risiken durch den Einsatz einer Risikomatrix (Abb. 2). Die Gegenüberstellung von Schadensausmaß und Eintrittswahrscheinlichkeit eines Hazards bildet die Entscheidungsgrundlage für Maßnahmen zur Beseitigung oder Entschärfung einer Gefahr. Besteht Todesgefahr oder Gefahr für Schwerverletzte bei relativ hohen und mittleren Eintrittswahrscheinlichkeiten, so ist die Reduzierung der Gefahr durch eine Sicherheitsfunktion nach SIL 3 möglich [HMA07]. Zuvor ist zu evaluieren, ob eine Beseitigung der Gefahr durch Verfahrens- oder Designänderung nicht tragbar ist. Die Definitionsbereiche der Einstufungen sind sektorspezi-

Frequency	Consequence			
	Catastrophic	Critical	Marginal	Negligible
Frequent	I	I	I	II
Probable	I	I	II	III
Occasional	I	III	III	III
Remote	III	III	III	IV
Improbable	III	III	IV	IV
Incredible	IV	IV	IV	IV

Abbildung 2: Beispiel für Risikoklassifizierung von Unfällen (Quelle: [IEC05] Teil 5 S. 35)

fisch.

Die Risikomatrix soll nun mit Risikoklassen belegt werden, die als Vorgabe für eine passende Lösungsstrategie dienen. Eine mögliche Konzeption der Klassen zeigt Abb. 3. So sieht nach [HMA07] eine beispielhafte Risikoklasseneinteilung, in etwas vereinfachter Form, aus:

Klasse 1 Nicht tolerierbares Risiko. Verfahrens- oder Entwurfsänderung notwendig.

Klasse 2 Nicht tolerierbares Risiko. Verfahrens- oder Entwurfsänderung notwendig, oder eine Schutzeinrichtung nach SIL 3.

Klasse 3 Mittleres Risiko. Verfahrens- oder Entwurfsänderung notwendig oder eine Schutzeinrichtung nach SIL 2.

Klasse 4 Geringes Risiko. Überwachungseinrichtung, dokumentierte Prüfung und organisatorische Maßnahmen von guter Qualität ausreichend.

Klasse 5 Tolerierbares Risiko. Keine Maßnahmen notwendig.

Nach der Bewertung der Gefährdungen auf dem EUC, ermittelt man die notwendigen Risikoreduktionen um ein tragbares Risiko zu erreichen. Die Maßnahmen zur Reduktion der Risiken werden im folgenden Schritt in Sicherheitsfunktionen allokiert.

4.1.2 Qualitative Determination des SIL

[IEC05] Teil 5 beschreibt zwei Methoden zur qualitativen Bestimmung des SIL. Die Maßnahmen sind insbesondere dann anzuwenden, wenn die Ereignisfrequenz einer Gefährdung nicht quantifizierbar ist.

4.2 Risikoeinstufung (Risk Assessment)

Der Grad für tragbares Risiko wird durch den Standard mit Hilfe von statistischen Tabellen in vier Kategorien klassifiziert. SIL 3 bildet dabei die zweithöchste Einstufung. Vor der Spezifizierung des SIL, wird iterativ eine Risikoreduktion über das Gesamtsystem durchgeführt, unter Beachtung der Sicherheitsintegritätsallokationen der Sicherheitsfunktionen, die auf das System wirken und von äußeren Faktoren.

Safety Integrity Level	Low demand mode of operation (Average probability of failure to perform its design function on demand)
4	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-2}$ to $< 10^{-1}$

Abbildung 4: Safety Integrity Levels: Zielausfallrate for Sicherheitsfunktion in niederfrequenten Bedarfsmodus (Quelle: [IEC05] Teil 1 S. 65)

Bei der Zuweisung eines Sicherheitsintegritätsgrades wird statistisch zwischen zwei Typen von Sicherheitsbereitstellung unterschieden. Der sog. Low demand mode of operation (Abb. 4) bietet eine statistische Verteilung für Sicherheitsfunktionen, die selten und bei Bedarf aktiviert werden. Die Verteilung bezieht sich dabei auf die durchschnittliche Wahrscheinlichkeit einer Fehlfunktion bei Ausführung, für SIL 3 liegt diese bei 10^{-4} bis 10^{-3}. Für Funktionen im Hochverfügbarkeitsszenario beschreibt eine weitere Tabelle die statistische Verteilung für die Wahrscheinlichkeit eines gefährlichen Ausfalls

Risk class	Interpretation
Class I	Intolerable risk
Class II	Undesirable risk, and tolerabel only if risk reduction is impracticable or if the costs are grossly disproporionate to the improvement gained
Class III	Tolerable risk if the cost of risk reduction would exceed the improvement gained
Class IV	Negligible risk

Abbildung 3: Beispiel für Risikoklasseninterpretation (Quelle: [IEC05] Teil 5 S. 35)

pro Stunde, für SIL 3 liegt das bei 10^{-8} bis 10^{-7} (Abb. 5). Ein Verfahren zur einheitlichen Spezifizierung des SIL wird in [JVB08] mit Hilfe einer zeitabhängigen Funktion vorgeschlagen.

Safety Integrity Level	High demand or continuous mode of operation (Probability of a dangerous failure per hour)
4	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-6}$ to $< 10^{-5}$

Abbildung 5: Safety Integrity Levels: Zielausfallrate for Sicherheitsfunktion in Hochverfügbarkeitsmodus (Quelle: [IEC05] Teil 1 S. 65)

5 Spezifikation

Ziel dieser Voraussetzung ist der Entwurf eines Plans zum Bewerkstelligen der Validierung eines sicherheitsbezogenen Systems. Inhalte des Plans setzen fest den Zeitpunkt der Validierung, die Verantwortlichen der Durchführung, die Operationsmodi des EUC[5] im Zusammenhang mit dem sicherheitsbezogenen System, die Spezifikation der Sicherheitsfunktionen für jeden Operationsmodus des Zielsystems, die Validierungsstrategie, die Prozeduren zur Verifikation der Konformität der Sicherheitsfunktionen mit den Spezifikationen der Risikoanalyse und der Risikoeinstufung, die Anforderungen an die Testumgebung, die Erfolgs- und Misserfolgskriterien und die Prozeduren der Resultatsbewertungen.

Resultierende Dokumente daraus können sein der Testplan, die Spezifikation des Testdesigns, der Testfälle und Testprozeduren [KWMH07].

[5]Equipment Under Control

- Der Testplan hält die organisatorischen Faktoren fest, wie den Zeitplan, die Testumgebung und die Testaktivitäten.
- In der Spezifikation des Testdesigns werden die Sicherheitsfunktionen festgelegt, die zu validieren sind. Die Anforderungen bieten die Grundlage der Spezifizierung welche Eigenschaften einer Funktion zu testen sind und wie ein erfolgreicher Testabschluss dafür definiert ist. Zusätzlich sollen Verfahrensweisen zur Handhabung von unzutreffenden Testresultaten vereinbart werden.
- Die Testfallspezifikation dokumentiert die Prozeduren jedes einzelnen Testdurchlaufs. Die Einstellungen des EUC, der Zustand des EUC und die Ein- und Ausgabeparameter der Sicherheitsfunktion sind in diesem Dokument erfasst. Bei automatischer Testfallgenerierung besteht die Möglichkeit dieses Dokument auch automatisiert zu erstellen.
- Durch die Spezifikation der Testprozedur wird der Testverlauf und die eingesetzten Werkzeuge festgelegt. Im Falle einer manuellen Testfallabarbeitung muss dieses Dokument die Schrittabfolge definieren, die die ausführende Person zu tätigen hat. Ähnlich ist es bei einem automatischen Testverlauf, hierbei muss jedoch auch das zum Einsatz kommende Werkzeug und dessen Handhabung genau spezifiziert werden.

6 Validierung der Softwaremodule

Zur Verifikation eines Gesamtsystems oder eines Softwaremoduls nach [IEC05], ist die konforme Validierung aller Softwaremodultests vorausgesetzt. Der Standard bietet dabei Vorschläge

zum Bewerkstelligen der Validierung der Softwaremodule unter Abhängigkeit des Sicherheitsintegritätslevels an. In den folgenden Kapiteln werden die Verfahren vorgestellt, die für SIL 3 als *Recommended* bzw. *Highly Recommended* eingestuft sind.

Zur Laufzeitreduzierung des Testvorgangs und der Testfallgenerierung werden verschiedene Strategien ergriffen, oft auch in Kombination miteinander. IEC 61508-3 bietet die Bildung von Äquivalenzklassen, strukturbasierte Testfälle, die Randwertanalyse und Kontrollflussanalyse als mögliche Szenarien für eine Testfallbildung an.

6.1 Probalistisches Testen

Stochastische Ansätze wie z.B. die Monte-Carlo-Simulation sind Möglichkeiten von konformen Validierungsmaßnahmen. Diese sind geeignet für das Testen von zustandsbasierten Systemen mit großen variablen Zustandsentwicklungen- und abhängigkeiten. Dabei bedient man sich numerischer Methoden zur Beurteilung der Testergebnisse in Abhängigkeit von der Zusammenstellung der Testfälle. Solche Methoden zeigen ihre höchste Effizienz durch das Mitwirken von Zufallszahlengeneratoren, die eine statistische Gleichverteilung ermöglichen und von Heuristiken, die durch semantikbewusste Testfallfilterung unwahrscheinliche Szenarien ausschließen.

Probalistisches Testen wird beim Testen von Softwaremodulen eingesetzt, wenn keine Einsicht in den Quellcode besteht.

Eine erfolgreiche Abnahme nach [IEC05] erfordert den Nachweis der Vorbereitung der Testfälle. Hierbei ist zu zeigen, dass die probalistische Entwicklung der Testfälle eine vollständige Abdeckung aller möglichen Fehlerereignisse darstellt, nicht aber notwendigerweise die Abdeckung aller möglichen Zustände des Moduls. Die Validierungsprozedur muss vollständig dokumentiert sein, das bedeutet Testspezifikation, Testausführungsprotokoll, Testresultat und Testresultatbeurteilung. Die Reproduzierbarkeit der Validierung setzt eine deterministische Zufallszahlenerzeugung voraus.

6.2 Dynamische Analyse

Die dynamische Analyse wird beim Testen von Softwaremodulen mit Einsicht auf Quellcode eingesetzt.

6.2.1 Randwertanalyse

Bei der Randwertanalyse beschränken sich die Testfälle auf Grenzwerte, die durch die Hard- und Softwarearchitektur, die Implementierung und das Interface bestehen. Dieses Analyseverfahren ist effektiv um Fehlverhalten an den Grenzen des Gültigkeitsbereichs festzustellen. Durch die signifikante Häufigkeit solcher Fehler in Softwaremodulen, wird die Randwertanalyse obligatorisch in Verbindung mit anderen Testverfahren angewendet.

6.2.2 Error Seeding

Ein Verfahren, dass eine probalistischen Grad des Fehlerbefalls entwickelt, ist das sog. Error Seeding. Es besteht darin explizit künstliches Fehlverhalten in das, zu testende, Modul zu platzieren um nach der Testphase einen Wahrscheinlichkeitswert über die noch vorhandenen, nicht künstlichen, Fehler zu berechnen. Die Kalkulation bedient sich der Stochastik und bildet mit den Variablen F_U für die Anzahl an unentdeckten Fehlern, F_G für die Anzahl an nicht-künstlichen entdeckten Fehlern, F_E für die Anzahl an platzierten künstlichen Fehlern und F_{EG} für die Anzahl an entdeckten platzierten künstlichen Fehlern folgende Gleichung: $F_U = F_G * \frac{F_E}{F_{EG}}$. Darüber hinaus gibt das Verfahren Aufschluss über die Qualität der Testprozeduren, indem man das Verhältnis von künstlichen Fehlern und entdeckten künstlichen Fehlern betrachtet.

6.2.3 Äquivalenzklassen

Durch die Permutation aller Eingabeparameter einer Sicherheitsfunktion entstehen im Regelfall Redundanzen. Reduktion der Eingangskombinationen kann durch Äquivalenzklassenbildung erfolgen. Dabei entwickelt man die Testfälle nach disjunkten Gruppen von Zustandsübergängen, die bei Vereinigung eine vollständige Abdeckung des Testszenarios bilden. Bei der Abstraktion des funktionalen Verhaltens eines Softwaremoduls können mathematische Annahmen über Wertebereiche und Wertemengen nicht immer mit den Systemeigenschaften übereinstimmen. Zur Vermeidung von Fehlschlüssen oder schlechten Fehlerabdeckungsquoten, fließt die Randwertanalyse mit in die Bildung der Äquivalenzklassen ein.

6.3 Funktionales und Black-Box Testen

Black-Box Testen bedeutet das Abarbeiten von Testszenarien unter vollständiger Abstraktion der Implementierungsschicht. Dabei werden Input- und Outputwerte, Fehlerbehandlungen und Performancemessungen als Faktoren der Validierung analysiert. Die Testfälle können aus Konsequenzdiagrammen, Prototypen, der Randwertanalyse und der den Äquivalenzklassen hergeleitet werden [IEC05].

Funktionales Testen wird beim Testen von Softwaremodulen ohne Einsicht auf Quellcode angewandt und beim Integrationstesten. Zur erfolgreichen Zertifizierung wird ebenfalls die vollständige Dokumentation der Vorbereitung der Tests, der Testprozeduren, der Testresultate und der Analyse der Resultate erwartet.

7 Funktionale Sicherheitsbeurteilung (Functional Safety Assessment)

Nach jeder Phase des Produktlebenszyklus findet die Sicherheitsbeurteilung statt. Diese hat zum Ziel die Evaluierung der tatsächlich erreichten Sicherheitsintegrität einer Sicherheitsfunktion. Erst wenn der Bewertungsplan sowohl von den ausführenden Personen als auch von dem Management bewilligt ist, findet die Bewertung statt. Der Plan setzt die Verantwortlichen für die Bewertung und deren Kompetenzen, die resultierende Dokumente, die involvierten Sicherheitsobjekte und notwendigen Ressourcen fest [IEC05].

Weiterhin beschreibt der Plan auch den Unabhängigkeitsgrad des Bewertungspersonals. Die notwendige Unabhängigkeit wird durch den Standard reguliert. Für SIL 3 verlangt dieser – in Abhängigkeit von einigen Faktoren – eine unabhängige Abteilung oder eine unabhängige Gesellschaft. Ist eine strikte Trennung der Safety-Abteilung von der Entwicklungsabteilung gewährleistet, ist das Personal geeignet geschult und bringt die nötige Erfahrung in Risikobewertung und sicherheitsabhängigen Systemen mit, so kann die Safety-Abteilung die Beurteilung übernehmen. Falls das nicht gegeben ist, muss ein externer Dienstleister, der die nötigen Qualitäten mitbringt, mit der Risikobewertung beauftragt werden.

Zur Bestimmung des SIL einer vorhandenen Software stellt [IEC05] Normen auf. Durch statistisches Testen kann man mit Hilfe der Tabellen den SIL der Software bestimmen. Es findet wieder eine Unterscheidung zwischen *Low demand mode of operation* und *High demand mode of operation* statt. Abb. 6 zeigt die Verteilung für die erste Funktionsvariante.

SIL	Low demand mode of operation	Number of treated demands	
	Probability of failure to perform its design function on demand	$1-\alpha = 0{,}99$	$1-\alpha = 0{,}95$
4	$\geq 10^{-5}$ to $< 10^{-4}$	$4{,}6 \times 10^5$	3×10^5
3	$\geq 10^{-4}$ to $< 10^{-3}$	$4{,}6 \times 10^4$	3×10^4
2	$\geq 10^{-3}$ to $< 10^{-2}$	$4{,}6 \times 10^3$	3×10^3
1	$\geq 10^{-2}$ to $< 10^{-1}$	$4{,}6 \times 10^2$	3×10^2

Abbildung 6: Statistische Verteilung zur Bestimmung des SIL für Low demand mode of operation (Quelle: [IEC05] Teil 7 Annex D S. 209)

8 Modellbasiertes Testen

Stetig wächst das Interesse nach Vereinfachung und Beschleunigung der Softwareentwicklung aus ökonomischen Gründen. Modellgetriebene Softwareentwicklung bietet eine Abstraktion von der Implementierung und Modularität an.

Robuste, verlässliche und sichere Module können die Bausteine von großen Softwareprojekten sein. Das erleichtern nicht nur die Entwicklung, sondern insbesondere auch die Verifikation der Sicherheitsintegrität. Modelle sind einfacher strukturiert und dadurch verständlicher für den Menschen. Verständlichkeit und Einfachheit sind wichtig bei der Ermittlung der Korrektheit des Designs und machen die Funktionen anpassungsfähiger, wenn die Notwendigkeit der Entwurfsänderung eintritt, z.B. zur Reduktion eines nicht-tolerierbaren Risikos.

Wichtig für die Akzeptanz von solchen Entwicklungsverfahren ist neben der Wirtschaftlichkeit auch die Qualität der Werkzeuge, wie z.B. dem Code-Generator. [Glo08] beschreibt den Zertifizierungsprozess des ASCET Code Generators nach [IEC05].

Zur Validierung eines Softwaremoduls werden bei der statischen Analyse nur noch die Model-

le betrachtet, ohne sich mit den Gegebenheiten des entstandenen Quellcodes auseinander zu setzen. Das Entfallen von Implementierungsfehlern ermöglicht eine effizientere Validierungsprozedur und erhöht die Wahrscheinlichkeit der Identifikation von Fehlerquellen bei gleichem Aufwand.

9 Fazit

Softwaresicherheit, Softwaretests und automatisierte Testfallerstellung sind weiterhin offene Probleme. Der Anstieg an Softwarelösungen für Sicherheitsfunktionen ist durch die wachsende Komplexität der Systeme unausweichlich. Methoden zur Vermeidung, Aufspüren und Beseitigung von Fehlern ist ein aktives Forschungsgebiet, welches durch die Notwendigkeit einer verlässlichen und allgemeingültigen Lösung getrieben wird.

Trotz Kritik an IEC 61508, es sei vielmehr ein Standard für Qualitätssicherung als für funktionale Sicherheit, biete keine genauen Richtlinien zur Berücksichtigung der Gesamtgröße eines Systems [Sho08] und definiert keine Möglichkeit zur einheitlichen Ermittlung des SIL für SIS[6] mit variablen Bedarfsanforderungen [Sat07],[JVB08], findet dieser in immer mehr Sektoren Einsatz und dient für viele andere Standards als Basis.

Dieser Text soll einen Einstieg in die Problematik bieten und notwendige Vorgehensweisen zur Zertifizierung nach SIL 3 erläutern. Testfallerstellung kann in Anbetracht von Verifizierungsmaßnahmen nicht isoliert betrachtet werden, sondern als interdisziplinäres Gesamtkonzept verstanden und gelöst werden.

Der Einsatz von modellgetriebener Entwicklung unter Verwendung von zertifizierten Code-Generatoren oder der formale Beweis der Fehlerfreiheit, wie er zur Zeit in einer experimentellen Studie für den Microsoft Hypervisor erstellt wird, können in Zukunft die Basis zur Validierung von Sicherheitssystemen bilden. Der IEC 61508 ist jedoch weiterhin auf die Kompetenz und Erfahrung der Ingenieure angewiesen, gleiches gilt für die Normen, die auf dem IEC 61508 basieren. Die wesentlichen Maßnahmen zur Unterstützung der Personalarbeit bei der Verifikation der Sicherheitsintegrität sind gründliche Dokumentenführung, geeignete Prozessmodelle und Werkzeugunterstützung in allen Entwicklungsphasen.

[6] Safety Instrumented System

Literatur

[Glo08] Tilman Gloetzner. *IEC 61508 Certification of a Code Generator.* 3rd IET International conference on System Safety, 2008.

[Hal04] Wolfgang A. Halang. *Automated Control Systems for the Safety Integrity Levels 3 and 4.* 9th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, 2004.

[HMA07] Dirk Hablawetz, Norbert Matalla, and Gerhard Adam. *IEC 61511 in der Praxis - Erfahrungen eines Anlagebetreibers.* atp, 2007.

[IEC05] IEC. *IEC 61508 Functional safety of electrical/electronic/programmable electronic safety-related systems.* International Electrotechnical Commission, 1998/2005.

[JVB08] PhD Julia V. Bukowski. *A Unified Model for Evaluating the Safety Integrity Level of Safety Instrumented Systems.* Reliability and Maintainability Symposium, 2008.

[KWMH07] Pawel Kwasnowsk, Grzegorz Wrobel, Zbigniew Mikos, and Grzegorz Hayduk. *Blackbox Testing Methodology for SafetyLON according to the IEC 61508 Standard.* 5th IEEE International Conference on Industrial Informatics, 2007.

[May02] Rodney May. *Personal Competencies and the Requirements of IEC 61508.* Programmable Electronics and Safety Systems: Issues, Standards and Practical Aspects, 2002.

[Mor07] Olwen Morgan. *Certified Testing of C Compilers for Embedded Systems.* 3rd Institution of Engineering and Technology Conference on Automotive Electronics, 2007.

[Sat07] Yoshinobu Sato. *Throwing a Bridge between Risk Assessment and Functional Safety.* SICE Annual Conference, 2007.

[Sho08] R.C. Short. *Towards a Calculus for Software SIL.* 3rd IET International Conference on System Safety, 2008.