Masterarbeit

SOFTWAREANFORDERUNGEN „FUNKTIONALER SICHERHEIT“ AUF ANWENDEREBENE INKLUSIVE ANALYSE EINES SOFTWAREASSISTENTEN

ausgeführt am

Fachhochschul-Masterstudiengang
Automatisierungstechnik-Wirtschaft

von
Ing. Manuel Josef Gerold, BSc.
19103222004

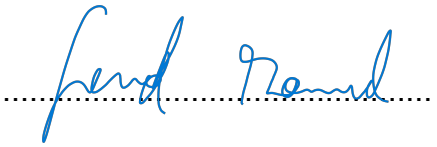
betreut und begutachtet von
Dipl.-Ing. Franz Gregor Blasge

Graz, im November 2020

.....
Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.



Unterschrift

DANKSAGUNG

Meiner Frau, meiner Familie, meinen Kolleginnen und Kollegen, sowie meinem Betreuer Herrn Dipl.-Ing. Franz Blasge möchte ich für die moralische Unterstützung und für Rat und Tat danken. Des Weiteren möchte ich meinem Arbeitgeber, der Firma Glock Ges.m.b.H., für die Unterstützung bei dieser Arbeit, aber auch für die Unterstützung beim gesamten Studium danken.

KURZFASSUNG

In den letzten Jahren werden Sicherheitsfunktionen in Maschinen vermehrt über sicherheitsgerichtete Steuerungen und die dazugehörige sicherheitsgerichtete Software realisiert. Dies hat zur Folge, dass es erhöhte Anforderungen an die Softwarequalität laut der Maschinenrichtlinie 2006/42/EG gibt. Die Anforderungen an sicherheitsgerichtete Software werden in den dazugehörigen harmonisierten Normen, wie üblich, nur sehr allgemein beschrieben und es gibt nahezu keine publizierten Beispiele dazu.

Das Ziel dieser Masterarbeit, welche in Kooperation mit der Firma Glock Ges.m.b.H. umgesetzt wird, ist es, die Anforderungen an eine sicherheitsgerichtete Software laut Maschinenrichtlinie und der harmonisierten Normenreihe EN ISO 13849 aufzuzeigen. Es sollen anhand der entsprechenden Normen Maßnahmen und Vorschläge zur praktischen Umsetzung gegeben werden. Das zweite Ziel dieser Masterarbeit ist es, das Softwaretool SOFTEMA mit Hilfe eines Applikationsbeispiels auf seine Praxistauglichkeit zu testen.

Das Ergebnis dieser Arbeit ist ein Softwarelebenszyklus mit konstruktiven und überprüfenden Tätigkeiten. Die Tätigkeiten werden mittels Umsetzungsbeispielen beschrieben und es werden Empfehlungen gegeben, welche Maßnahmen je nach erforderlichem Performance Level der Sicherheitsfunktion nötig sind. Die Anwendung des Softwaretools SOFTEMA wird mittels einer Roboterautomatisierung mit integrierter funktionaler Sicherheit getestet. Es wird ein Bericht über die Praxistauglichkeit des Softwaretools gegeben und des Weiteren wird eine Handlungsempfehlung an das Unternehmen gegeben, ob das Softwaretool in Zukunft eingesetzt werden sollte.

ABSTRACT

In recent years, safety functions in the field of mechanical engineering have been increasingly implemented with safety-related control systems and the associated safety-related software. As a result, the software quality requirements, according to the Machinery Directive 2006/42/EC, have increased significantly. The requirements are only described superficially in the associated harmonized standards and published examples are not available.

The aim of this master’s thesis, in cooperation with the company Glock Ges.m.b.H., is to demonstrate the requirements for the development of a safety related software, according to the Machinery Directive and the harmonized standards series EN ISO 13849. Another objective is to test the software tool SOFTEMA regarding its practical suitability employing an application example.

The result of this work is a process of a software life cycle with its corresponding acting and testing activities. The development activities are illustrated with implementation examples. Recommendations regarding the software developing process are made based on the corresponding standards. Furthermore, it is depicted which activities are necessary depending on the required performance level of the safety function. The test object for the practicability of the software tool SOFTEMA is a robot automation cell. Finally, a report on the usability of the software tool is created and a recommendation for the future usage of the software solution is given.

INHALTSVERZEICHNIS

1 EINLEITUNG

In den letzten Jahren werden im Bereich des Maschinenbaus Sicherheitsfunktionen vermehrt mit sicherheitsgerichteten Steuerungen umgesetzt. Durch die Forderungen der Maschinensicherheitsverordnung 2010 (nationale Umsetzung der Europäischen Maschinenrichtlinie 2006/42/EG in Österreich) ist es verpflichtend eine Risikobeurteilung durchzuführen. Wenn durch die Risikobeurteilung bestimmt wird, dass eine Gefährdung durch eine „technische Schutzmaßnahme und ergänzende Schutzmaßnahme“ vermindert werden soll, wird in vielen Fällen eine sicherheitsgerichtete Steuerung verwendet. Wenn im Bereich der Maschinensicherheitsverordnung 2010 funktionale Sicherheit eingesetzt wird, gibt es die nach der Maschinenrichtlinie harmonisierte Normenreihe EN ISO 13849 mit dem Titel „Sicherheit von Maschinen – Sicherheitsbezogene Teile von Steuerungen“. Diese Normen fordern einen strukturierten Entwicklungsprozess um diese Funktionen umzusetzen. Es wird auch für die sogenannte sicherheitsgerichtete Software im Kapitel 4.6 der EN ISO 13849-1 ein Entwicklungsprozess nach dem sogenannten V-Modell gefordert. Des Weiteren werden in der Norm auch fehlervermeidende Maßnahmen während der Entwicklung gefordert. Die geforderten Maßnahmen sind in der Norm sehr allgemein gehalten und das Nichtvorhandensein von publizierten Beispielen zur Umsetzung macht die Umsetzung für Unternehmen sehr schwierig. Diese Problematik wurde von der Deutschen Gesetzlichen Unfallversicherung (DGUV) aufgegriffen und im Rahmen eines Projektes mit der Hochschule Bonn Rhein Sieg wurde eine Matrixmethode entwickelt, welche die Unternehmen bei diesem Prozess unterstützen soll. Des Weiteren wurde für die Anwendung der Matrixmethode ein Softwareassistent mit dem Namen SOFTEMA entwickelt, um den Prozess zu vereinfachen.

Die Aufgabe dieser Masterarbeit liegt darin, aufzuzeigen welche Anforderungen es lt. Maschinensicherheitsverordnung 2010 gibt. Der Hauptteil dieser Arbeit besteht darin, aufzuzeigen welche Forderungen es laut den Normen EN ISO 13849-1 und EN ISO 13849-2 an die Softwareentwicklung einer Sicherheitsfunktion gibt. Diese sollen von der Entwicklung bis hin zur Auslieferung der fertigen Maschine dargestellt werden. Als Komponenten werden hier sicherheitsgerichtete Steuerungen mit eingeschränkter Programmiersprache und mit sicherheitsbezogenen Funktionsbaustein - Bibliotheken angenommen. Des Weiteren soll eine Vorgehensweise definiert werden, wie dieser Prozess abzulaufen hat.

Ein weiterer Punkt ist das Überprüfen des Softwareassistenten SOFTEMA auf seine Praxistauglichkeit. Die Überprüfung soll anhand eines realen Praxisbeispiels erfolgen und es soll anhand dieser Überprüfung eine Entscheidung getroffen werden, ob diese Software in Zukunft im Unternehmen zum Einsatz kommt.

2 FORDERUNGEN DER MASCHINENSICHERHEITSVERORDNUNG

Jede Maschine, die erstmals in den europäischen Wirtschaftsraum entgeltlich oder unentgeltlich in Verkehr gebracht oder Inbetrieb genommen wird, fällt in den Geltungsbereich der MRL (Maschinenrichtlinie) 2006/42/EG. Die MSV (Maschinensicherheitsverordnung) 2010 entspricht der nationalen Umsetzung der Maschinenrichtlinie 2006/42/EG in Österreich. In dieser werden grundlegende Sicherheits- und Gesundheitsschutzanforderungen gestellt.
Der Anwendungsbereich der MSV 2010 ist wie folgt:

- Maschinen
- Auswechselbare Ausrüstungen
- Sicherheitsbauteile
- Lastaufnahmemittel
- Ketten, Seile, Gurte
- Abnehmbare Gelenkwellen
- Unvollständige Maschinen [1]

Definition einer Maschine:
„Eine Maschine ist eine mit einem Antriebssystem ausgestattete oder dafür vorgesehene Gesamtheit miteinander verbundener Teile oder Vorrichtungen, von denen mindestens eine(s) beweglich ist und die für eine bestimmte Anwendung zusammengefügt sind.“[2]

Der Hersteller bzw. der Inverkehrbringer einer Maschine ist verpflichtet:

- dass die Maschine den grundlegenden Sicherheits- und Gesundheitsschutzanforderungen entspricht (MSV 2010 Anhang I)
- sicherzustellen, dass die technischen Unterlagen verfügbar sind (MSV 2010 Anhang VII Teil A)
- dass es eine Betriebsanleitung gibt
- ein zutreffendes Konformitätsbewertungsverfahren anzuwenden (MSV 2010 Artikel 12)
- dass die EG-Konformitätserklärung ausgestellt und unterfertigt wird und der Maschine beiliegt (MSV 2010 Anhang II Teil 1 Abschnitt A)
- eine CE-Kennzeichnung anzubringen (MSV 2010 Artikel 13) [3]

[1] Vgl. MSV 2010 (2020), Seite: 2

[2] Vgl. EN ISO 12100:2013, Seite: 6

[3] Vgl. MSV 2010 (2020), Seite: 6

Laut MSV 2010 Anhang I sind folgende Punkte bei dem iterativen Verfahren der Risikobeurteilung und Risikominderung zu beachten:

- Grenzen der Maschine festlegen
- Ermitteln von Gefährdungen
- Risiken abschätzen
- Risiken bewerten
- Gefährdungen auszuschalten oder zu mindern [4]

Die Risikobeurteilung kann mit Hilfe der harmonisierten Norm EN ISO 12100 durchgeführt werden, diese wird in Abschnitt 3 genauer erläutert. Für jede Maschine muss eine interne und externe technische Dokumentation erstellt werden. Die externe Dokumentation umfasst nur die „wichtigsten" Dokumente wie Betriebsanleitung usw. Hingegen sind bei der internen technischen Dokumentation folgende Dokumente zu archivieren:

- Allgemeine Beschreibung der Maschine
- Übersichtszeichnung und Schaltpläne
- Detailzeichnungen, Berechnungen, Bescheinigungen usw.
- Unterlagen der Risikobeurteilung mit angewandten Verfahren
    - Liste der grundlegenden Sicherheits- und Gesundheitsschutzanforderungen
    - Beschreibung über Abwendung ermittelter Gefahren
- Angewandte Normen
- Ergebnisse von Prüfungen
- Betriebsanleitung der Maschine
- Gegebenenfalls Einbauerklärung
- Gegebenenfalls Kopie der EG-Konformitätserklärung für verbaute Maschinen
- Kopie der EG-Konformitätserklärung [5]

Die externe Dokumentation muss dem Kunden/Betreiber zur Verfügung gestellt werden und besteht unter anderem aus der Betriebsanleitung und der Konformitätserklärung. Die Risikobeurteilung, angewandte Normen usw. sind Teil der internen Dokumentation und diese muss für Behörden nach dem Tag der Herstellung für die Dauer von 10 Jahren bei Bedarf zur Verfügung gestellt werden.[6]

[4] Vgl. MSV 2010 (2020), Seite: 14

[5] MSV 2010 (2020), Seite: 26-27

[6] Vgl. MSV 2010 (2020), Seite: 49

3 ZUSAMMENHANG ZWISCHEN RISIKOBEURTEILUNG UND SICHERHEITSBEZOGENER ANWENDUNGSSOFTWARE

In diesem Kapitel wird der grundsätzliche Aufbau einer Risikobeurteilung skizziert und der Zusammenhang zwischen der Risikobeurteilung und dem Maß an fehlervermeidenden Maßnahmen in der sicherheitsbezogenen Software erläutert.

3.1 Allgemeines zur Risikobeurteilung

Wie im Kapitel 2 aufgezeigt, wird bei Maschinen zwingend eine Risikobeurteilung z.B. nach EN ISO 12100 gefordert.

Die Risikobeurteilung soll dabei unterstützen Gefährdungen zu identifizieren und eine Hilfestellung bei der Risikoeinschätzung sowie bei der Risikobewertung geben. Dies wird für alle relevanten Phasen der Lebensdauer durchgeführt. Außerdem wird auch darauf eingegangen, wie eventuelle Gefährdungen beseitigt werden können bzw. wie das Risiko ausreichend gemindert werden kann.

Das Verfahren der Risikobeurteilung bzw. der Risikominderung lt. EN ISO 12100 ist wie in Abbildung 1 zu sehen ist, in folgende Schritte festgelegt und diese sollen iterativ durchlaufen werden:

1. Grenzen der Maschine festlegen
    a. bestimmungsgemäße Verwendung angeben
    b. vernünftigerweise vorhersehbare Fehlanwendung festlegen
2. Identifizieren von Gefährdungen und Gefährdungssituationen
3. Einschätzung des Risikos für jede Gefährdung
4. Bewerten des Risikos und treffen von Entscheidungen zur Risikominderung

→ Risikoanalyse

5. Beseitigen bzw. Vermindern der Gefährdung durch Schutzmaßnahmen [7]

→ Risikominderung

[7] Vgl. EN ISO 12100:2013, Seite: 14

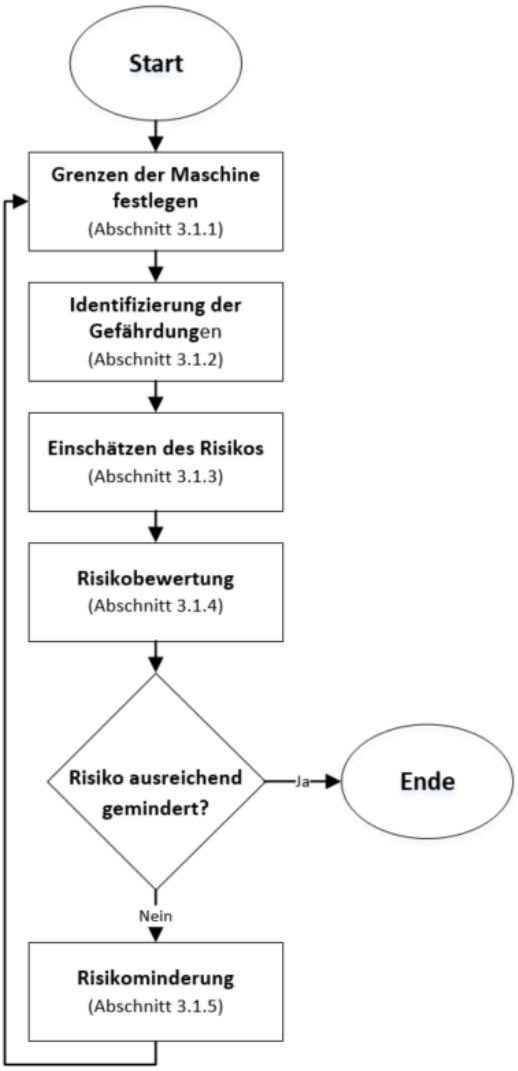


Abbildung 1 Iterativer Prozess der Risikobeurteilung, Quelle: EN ISO 13849-1, Seite: 19 (leicht modifiziert)

Dieser iterative Prozess der Risikobeurteilung, der Abbildung 1, wird in den folgenden Abschnitten genauer erläutert.

3.1.1 Grenzen der Maschine festlegen

Der erste Schritt beim Durchführen einer Risikobeurteilung ist das Festlegen der Grenzen der Maschine für alle Phasen der Lebensdauer.

Folgende Punkte sind beim Festlegen der Grenzen grundsätzlich zu betrachten:

- **Bestimmungsgemäße Verwendung**
  Angeben des spezifischen Einsatzzwecks der Maschine, das Einsatzgebiet für den diese Maschine gebaut wurde. Es soll auch eine vernünftigerweise vorhersehbare Fehlanwendung betrachtet werden.
- **Einsatzgebiet der Maschine**
  Angeben des Einsatzgebietes der Maschine, ob diese im industriellen Umfeld oder im privaten Bereich eingesetzt wird. In diesem Punkt sollen auch Merkmale von Personen berücksichtigt werden z.B. Körpergröße, Links- oder Rechtshändigkeit usw.

- **Benutzergruppe**
  Bei diesem Punkt wird die fachliche Qualifikation der verschiedenen Benutzergruppen behandelt z.B. Instandhaltungspersonal, Bedienpersonen usw.
- **Räumliche Grenzen**
  Hier wird allgemein der Platzbedarf der Maschine, der Platzbedarf von Personen und die Wechselwirkung zwischen Mensch und Maschine betrachtet.
- **Zeitliche Grenzen**
  Dieser Punkt beschäftigt sich mit der Lebensdauer der Maschine und deren Bauteilen. Angegeben werden empfohlene Wartungsintervalle, sowie die Haltbarkeit von gewissen sicherheitsrelevanten Bauteilen.
- **Weitere Grenzen**
  Unter „weitere Grenzen“ sollten Parameter wie die Umgebungstemperatur, Luftfeuchte und der erforderliche Reinlichkeitsgrad betrachtet werden.[8]

3.1.2 Gefährdungen identifizieren

Nachdem die Grenzen der Maschine festgelegt wurden, ist der nächste Schritt das systematische Identifizieren von Gefährdungen und Gefährdungssituationen während der gesamten Lebensdauer einer Maschine, d.h. von der Montage bis hin zur Demontage und Entsorgung. Es sollen während der Lebensdauer folgende Gefährdungen aufgezeigt werden:

- **Eingreifen durch Personen in die Maschine**
  - Einrichten
  - Anlauf
  - Reinigung
  - usw.
- **Mögliche Betriebszustände der Maschine**
  - Normalbetrieb
  - Versagen von vorhergesehenen Funktionen
    - Ausfall von Bauteilen
    - Störungen von außen
    - usw.
- **Vernünftigerweise vorhersehbare Fehlanwendung**
  - Verhalten des Bedieners bei Unachtsamkeit
  - Reflexartiges Verhalten bei Fehlfunktion
  - usw.[9]

[8] Vgl. EN ISO 12100:2013, Seite: 19-20

[9] Vgl. EN ISO 12100:2013, Seite: 21-22

3.1.3 Einschätzen des Risikos

Nachdem alle Gefährdungen identifiziert wurden, muss das Risiko der einzelnen Gefährdungen gewertet werden.

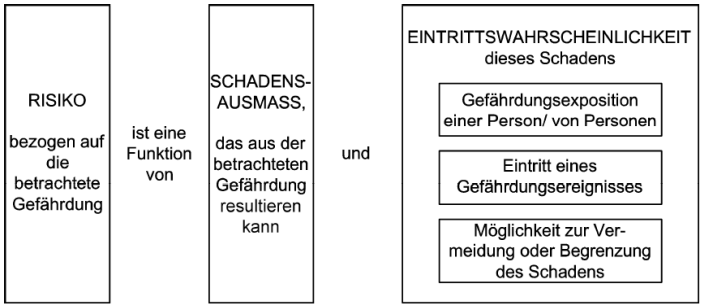


Abbildung 2 Risikoelemente, Quelle EN ISO 12100, Seite: 24

Der wichtigste Faktor bei der Bestimmung des Risikos ist der Grad des Schadensausmaßes, wie z. B. der Verlust von Händen und die Wahrscheinlichkeit des Eintritts eines Schadens. In Abbildung 2 ist dargestellt, dass die Wahrscheinlichkeit des Eintritts eines Schadens eine Funktion ist. Diese Funktion setzt sich zusammen aus der Wahrscheinlichkeit der Gefährdungsexposition einer Person, dem Eintritt des Gefährdungsereignisses und die Möglichkeit, dass der Schaden vermieden oder begrenzt wird.

Für die praktische Anwendung und deren Verfahren zur Risikobeurteilung wird in der EN ISO 1200 auf den „Technischen Report“ ISO/TR 14121-2 verwiesen. Dieser Technische Report, beinhaltet unter anderem mehrere verschiedene Instrumente zur Risikoeinschätzung wie z.B. Risikomatrix und Risikograf. In dieser Arbeit wird aber nur der in der Praxis häufig verwendete Risikograf betrachtet.

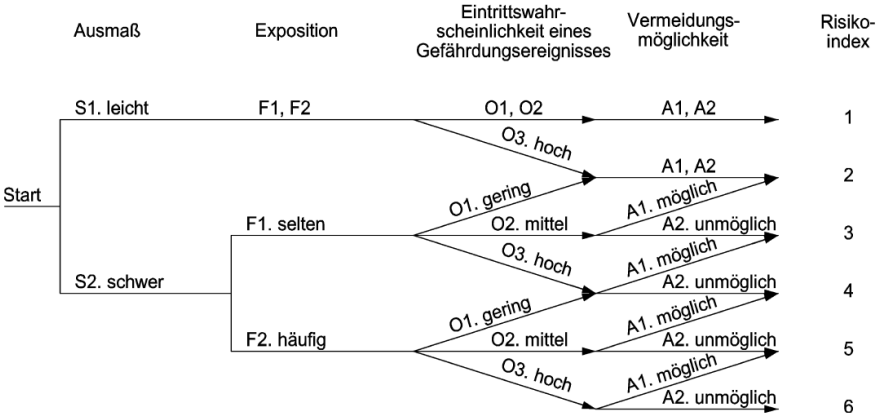


Abbildung 3 Risikograf, Quelle ISO/TR 14121-2, Seite: 19

In Abbildung 3 ist ersichtlich, wie der Prozess durchgeführt wird. Zu Beginn muss die Schwere des Schadens beurteilt werden:

- **S1**
  Leichte Verletzung z.B. Kratzer, kleine Wunden
- **S2**
  Schwere Verletzung z.B. Verlust von Gliedmaßen, einschließlich Tod

Danach wird die Aufenthaltsdauer bzw. die Dauer der Gefährdungsexposition beurteilt:

- **F1**
  Zweimal oder weniger pro Arbeitsschicht oder weniger als 15 min pro Arbeitsschicht.
- **F2**
  Mehr als zweimal pro Arbeitsschicht oder mehr als 15 min pro Arbeitsschicht.

Wenn die Aufenthaltsdauer beurteilt wurde, muss die Wahrscheinlichkeit des Eintretens der Gefährdung beurteilt werden:

- **O1**
  Anerkannte und bewährte Technik
- **O2**
  Ein technisches Versagen wurde in den letzten zwei Jahren festgestellt. Ein unangemessenes Handeln einer gut ausgebildeten Person, welche sich der Risiken bewusst ist und über mehr als sechs Monate Erfahrung an diesem Arbeitsplatz besitzt.
- **O3**
  Ein technisches Versagen tritt häufig auf, alle sechs Monate oder öfter. Ein unangemessenes Handeln einer nicht ausgebildeten Person mit weniger als sechs Monaten Erfahrung an diesem Arbeitsplatz.

Als Letztes wird noch die Möglichkeit der Abwendung bzw. die Minderung des Schadens bewertet:

- **A1**
  Abwendung ist möglich unter bestimmten Voraussetzungen z.B. wenn die Geschwindigkeit kleiner als 250 mm/s ist.
- **A2**
  Abwendung ist nicht möglich

Am Ende der Bewertung laut Risikograf ergibt sich ein Risikoindex zwischen 0 und 6. Hierbei wird 0 als geringes und 6 als hohes Risiko betrachtet. [10]

3.1.4 Treffen von Entscheidungen zur Risikominderung

Wenn das Risiko eingeschätzt wurde, muss eine Risikobewertung angewendet werden. Hierbei muss die Frage geklärt werden, welches Risiko vertretbar ist und welches Risiko vermindert werden muss.

Im Abschnitt 3.1.5 wird das sogenannte „Drei-Stufen-Verfahren“ vorgestellt, dieses ist unumgänglich, damit eine hinreichende Risikominderung möglich ist. Eine hinreichende Risikominderung kann z.B. sein, dass der Benutzer über Restrisiken ausreichend informiert wurde. Es können Risiken auch mit Risiken bei

[10] Vgl. ISO/TR 14121-2:2012, Seite: 18-19

ähnlichen Maschinen verglichen werden, jedoch ist es notwendig, dass die betrachteten Maschinen die gleichen technischen Spezifikationen usw. haben.[11]

3.1.5 Vermeidung und Beseitigung von Gefährdungen

Wenn bei der Risikobewertung entschieden wird, dass eine Gefährdung vermindert werden muss, kann diese nach dem sogenannten „Drei-Stufen-Verfahren“ durchgeführt werden. Ein „Drei-Stufen-Verfahren“ wird bereits in der MSV 2010 im Anhang I Abschnitt 1.1.2 gefordert.

- **Inhärent sichere Konstruktion**
  Eine „inhärent sichere Konstruktion“ ist immer der erste bevorzugte Schritt der Risikominderung. Eine „inhärent sichere Konstruktion“ bedeutet, dass die Maschine so konstruiert ist, dass keine Risiken durch die Maschinen selbst oder durch Wechselwirkungen mit Personen entstehen können.
  Beispiel: Verhindern von Verletzungen durch das Entgraten von Kanten
- **Technische Schutzmaßnahmen und ergänzende Schutzmaßnahmen**
  Wenn das Risiko durch eine „inhärent sichere Konstruktion“ nicht möglich ist, können weitere Maßnahmen ergriffen werden. Die technische Schutzmaßnahme kann einerseits mechanisch umgesetzt werden oder durch funktionale Sicherheit, welche in Abschnitt 3.2 erläutert wird.
  Beispiel: Schutzzaun, Berührungslos wirkende Schutzeinrichtung (z.B. Sicherheitslichtvorhang)
- **Benutzerinformation**
  Wenn durch „inhärent sichere Konstruktion“ und „technische Schutzmaßnahmen und ergänzende Schutzmaßnahmen“ das Risiko nicht zu einem vertretbaren Risiko vermindert werden konnte, ist es notwendig „Benutzerinformationen“ zu verwenden. Die „Benutzerinformation“ soll auf keinen Fall als Ersatz für eine „inhärent sichere Konstruktion“ oder eine „technische Schutzmaßnahme und ergänzende Schutzmaßnahme“ gesehen werden.
  Beispiele: Piktogramme, Anweisung für das Tragen von persönlicher Schutzausrüstung

Sobald eine der Maßnahmen getroffen wurde, muss die Gefährdung neu bewertet werden, ob das Risiko ausreichend gemindert werden konnte. Wichtig ist auch, dass betrachtet wird, ob durch die gesetzte Maßnahme nicht ein neues Risiko entsteht. Dieser aufgezeigte iterative Prozess muss solange ausgeführt werden, bis das Risiko auf ein vertretbares Maß gemindert wurde.[12]

[11] Vgl. EN ISO 12100:2013, Seite: 28

[12] Vgl. EN ISO 12100:2013, Seite: 29

3.2 Funktionale Sicherheit

Wenn bei der Risikobeurteilung, wie in Abschnitt 3.1 vorgestellt, eine Verringerung der Gefährdung durch eine technische Schutzmaßnahme mit funktionaler Sicherheit ausgewählt wird, dann gibt es verschiedenste Anforderungen lt. der EN ISO 13849-1. Funktionale Sicherheit ist sehr vielseitig und reicht vom Verwenden eines einfachen Endschalters bis hin zum Verwenden eines Lichtvorhanges in Verbindung mit einer Sicherheits - SPS.

In Abbildung 4 ist der iterative Prozess zur Gestaltung von funktionaler Sicherheit zu sehen. In den folgenden Abschnitten werden die wichtigsten Schritte in diesem Prozess erläutert.

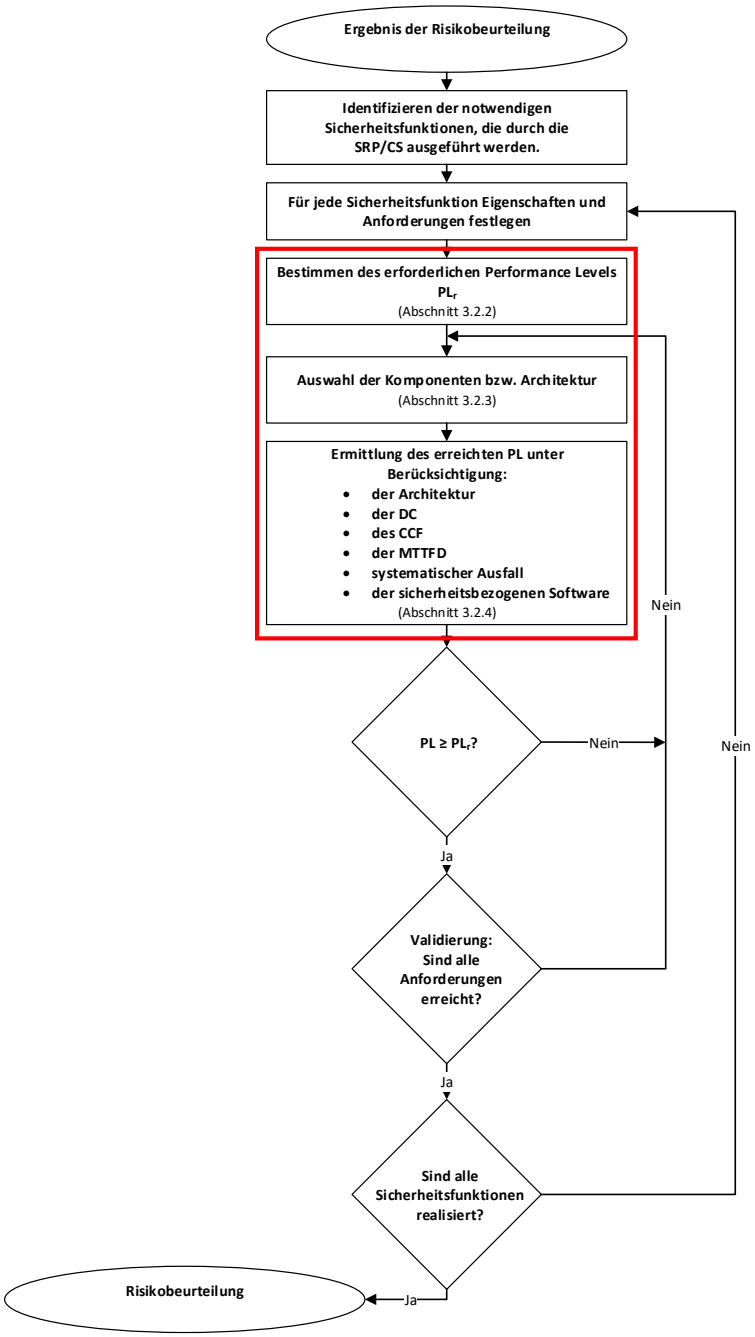


Abbildung 4 Prozess zur Gestaltung von funktionaler Sicherheit, Quelle: EN ISO 13849-1, Seite: 23 (leicht modifiziert)

3.2.1 Begriffserläuterungen

In den folgenden Abschnitten werden einige Begriffe genannt und diese werden hier erläutert.

- **CCF (Common Cause Failure)**

  Damit werden Ausfälle verschiedener Einheiten auf Grund eines einzelnen Ereignisses, wobei diese Ausfälle nicht auf gegenseitiger Ursache beruhen.

- **$MTTF_D$ (Mean Time To Dangerous Failure)**

  Wird bezeichnet als die mittlere Zeit bis zu einem gefährlichen Ausfall.

- **DC (Diagnostic Coverage)**

  Der Diagnosedeckungsgrad bezeichnet die Wirksamkeit der Diagnose. Dieser zeigt das Verhältnis von Ausfallsrate von erkannten gefährlichen Ausfällen zu allen gefährlichen Ausfällen.

- **PFH_D (Probability of a Dangerous Failure per Hour)**

  Wird als die durchschnittliche Wahrscheinlichkeit eines möglicherweise gefährlichen Ausfalls pro Stunde bezeichnet.

- **Verifikation**

  Es muss der PL (Performance Level) jeder einzelnen Sicherheitsfunktion des Systems mit dem PL_r (Performance Level Required) gegengeprüft werden. Wenn der PL_r nicht erreicht werden kann, muss der Prozess der Entwicklung bzw. des Systemdesigns wiederholt werden.[13]

- **Validierung**

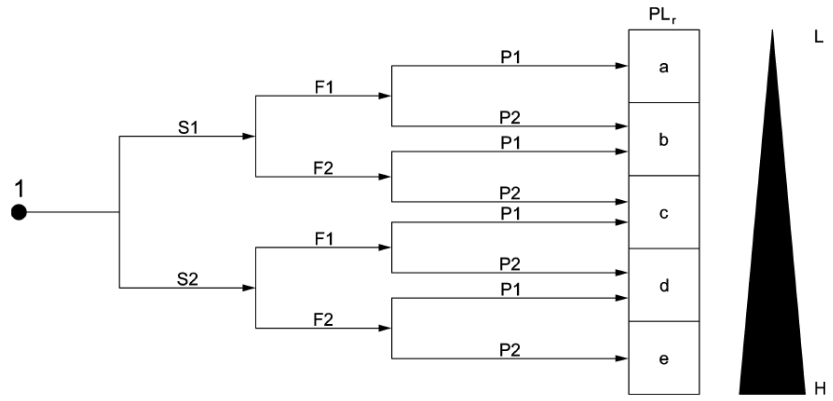
  Es muss überprüft werden, ob die Gestaltung der Sicherheitsfunktionen mit den Spezifikationen der Sicherheitsanforderungen der Maschine übereinstimmt.[14]

[13] Vgl. EN ISO 13849-1:2016, Seite: 39

[14] Vgl. EN ISO 13849-2:2013, Seite. 6

3.2.2 Bestimmen des geforderten Performance Levels

Um für die Anwendung die passenden Komponenten bzw. passende Architektur auszuwählen, ist es notwendig zuvor den benötigten PL_r zu bestimmen. In der EN ISO 13849-1 Anhang A wird als Hilfestellung ein Verfahren vorgestellt.



Legende

1	Startpunkt zur Bewertung des Sicherheitsfunktionsbeitrags zur Risikominderung
L	niedriger Beitrag zur Risikoreduzierung
H	hoher Beitrag zur Risikominderung
PL_r	erforderlicher Performance Level

Risikoparameter:

S	Schwere der Verletzung
S1	leichte (üblicherweise reversible Verletzung)
S2	ernste (üblicherweise irreversible Verletzung oder Tod)
F	Häufigkeit und/oder Dauer der Gefährdungsexposition
F1	selten bis weniger häufig und/oder die Zeit der Gefährdungsexposition ist kurz
F2	häufig bis dauernd und/oder die Zeit der Gefährdungsexposition ist lang
P	Möglichkeit zur Vermeidung der Gefährdung oder Begrenzung des Schadens
P1	möglich unter bestimmten Bedingungen
P2	kaum möglich

Abbildung 5 Graf zur Bestimmung des PL_r, Quelle: EN ISO 13849-1, Seite: 61

In Abbildung 5 ist dieses Verfahren dargestellt, welches aber nicht verbindlich ist, sondern nur generisch die höchste Eintrittswahrscheinlichkeit einer Gefährdung abbildet. Dieses Verfahren wurde im ISO/TR 22100-2:2013 vorgestellt und für die EN ISO 13849-1 übernommen.

Zu Beginn des Verfahrens muss die Schwere einer Verletzung bestimmt werden. Hier kann allgemein gesagt werden:

- **S1**
  Leichte Verletzung z.B. Kratzer
- **S2**
  Schwere Verletzung z.B. Amputation oder im schlimmsten Falle Tod

Nachdem die Schwere der Verletzung bestimmt wurde, muss die Häufigkeit der Gefährdungsexposition bestimmt werden. Für die Parameter *F1* und *F2* gibt es keine allgemein gültigen Zeiträume. Wenn eine Person dieser Gefährdung dauerhaft oder zumindest häufig ausgesetzt ist, sollte *F2* verwendet werden. Wenn keine anderen Parameter bekannt sind, sollte *F2* gewählt werden, wenn eine Person dieser Gefährdung häufiger als einmal alle 15 min ausgesetzt ist. *F1* darf gewählt werden, wenn die gesamte Expositionsdauer 1/20 der gesamten Betriebsdauer nicht überschreitet und die Häufigkeit nicht höher als einmal je 15 min ist.

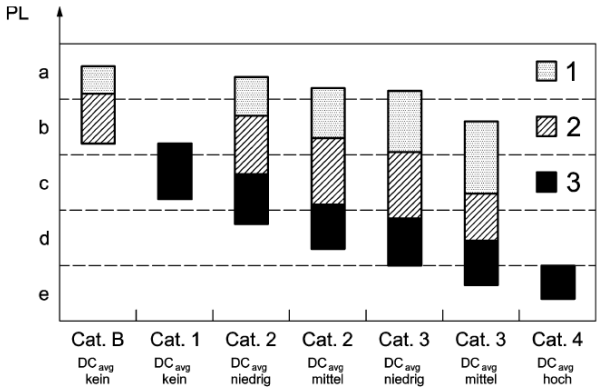
Als Letztes muss noch die Wahrscheinlichkeit der Vermeidung der Gefährdung und die Wahrscheinlichkeit des Eintritts betrachtet werden. Diese zwei Parameter sind in *P* kombiniert. Es kann allgemein gesagt werden, dass *P1* nur dann gewählt werden darf, wenn eine realistische Chance auf Nichteintreten der Gefährdung besteht bzw. dass die Gefährdung im größeren Maß vermindert werden kann. Aspekte, welche den Parameter *P* beeinflussen, sind z.B. ob die Geschwindigkeit der Gefährdung schnell oder langsam ist usw. Für die Eintrittswahrscheinlichkeit kann z.B. die Unfallgeschichte von vergleichbaren Maschinen herangezogen werden. Wenn die Eintrittswahrscheinlichkeit als sehr gering angenommen werden kann, darf der PL_r um ein Level abgesenkt werden.

Wichtig ist, dass jede Sicherheitsfunktion mit dem soeben vorgestellten Verfahren durchgeführt werden muss.[15]

3.2.3 Auswahl der Komponenten und der Architektur

Nachdem für jede Sicherheitsfunktion ein PL_r bestimmt wurde, müssen die dazugehörigen Komponenten bzw. die Architektur zur Umsetzung bestimmt werden.

In der EN ISO 13849-1 werden fünf verschiedene Kategorien benannt, mit denen, wie in Abbildung 6 ersichtlich, je nach Kategorie ein bestimmter PL erreicht werden kann. Wie die Kategorien aufgebaut sind, wird in den folgenden Ausführungen erläutert.



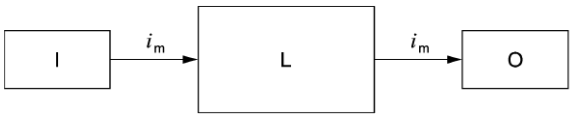
Legende
PL Performance Level
1 $MTTF_D$ jedes Kanals = niedrig
2 $MTTF_D$ jedes Kanals = mittel
3 $MTTF_D$ jedes Kanals = hoch

Abbildung 6 Erreichbarer PL nach Kategorie, Quelle: 13849-1, Seite: 31

[15] Vgl. EN ISO 13849-1:2016, Seite: 58-59

3.2.3.1 Kategorie B

Die Kategorie B ist die grundlegende Kategorie. Wie in Abbildung 7 ersichtlich, ist diese Kategorie aufgebaut aus Eingabeeinheit → Logik → Ausgabeeinheit.



Legende
i_m Verbindungsmittel
I Eingabeeinheit, z. B. Sensor
L Logik
O Ausgabeeinheit, z. B. Hauptschütz

Abbildung 7 Aufbau Kategorie B, Quelle: EN ISO 13849-1, Seite: 46

Es gibt folgende Anforderungen an die Kategorie B:

- Die Sicherheitsfunktion bzw. der Aufbau müssen lt. den zutreffenden Normen realisiert werden.
- Es müssen grundlegende Sicherheitsprinzipien eingehalten werden.

Die Kategorie B hat folgende Eigenschaften:

- Maximal erreichbarer PL = b
- Keine Maßnahmen gegen CCF bzw. nicht relevant
- $MTTF_D$ muss niedrig bis mittel sein
- Kein Diagnosedeckungsgrad

Wichtig ist, dass ein einzelner Fehler zum Ausfall der Sicherheitsfunktion führen kann.[16]

3.2.3.2 Kategorie 1

Der Aufbau der Kategorie 1 ist ident mit der in Abschnitt 3.2.3.1 vorgestellten Kategorie B, zusätzlich müssen jedoch folgende Anforderungen erfüllt werden:

- Verwendung bewährter Bauteile
- Verwendung bewährter Sicherheitsprinzipien

Die Kategorie 1 hat folgende Eigenschaften:

- Maximal erreichbarer PL = c
- Keine Maßnahmen gegen CCF bzw. nicht relevant
- $MTTF_D$ muss hoch sein
- Kein Diagnosedeckungsgrad

Wichtig ist, dass ein einzelner Fehler zum Ausfall der Sicherheitsfunktion führen kann.[17]

[16] Vgl. EN ISO 13849-1:2016, Seite: 46

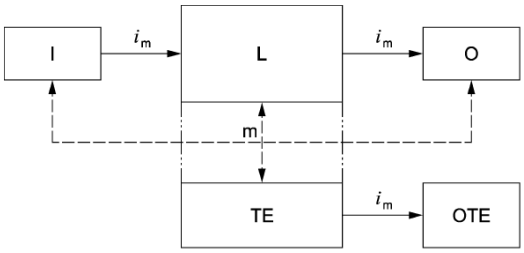
[17] Vgl. EN ISO 13849-1:2016, Seite: 46-47

3.2.3.3 Kategorie 2

Die Anforderungen sind ident mit der in Abschnitt 3.2.3.1 vorgestellten Kategorie B, zusätzlich ist Folgendes zu erfüllen:

- Test der Sicherheitsfunktionen in angemessenen Zeitabständen
    - Test bei Anlauf der Maschine
    - Test vor Einleiten einer Gefährdungssituation
- Bewährte Sicherheitsprinzipien

In Abbildung 8 ist ersichtlich, dass der in Abschnitt 3.2.3.1 vorgestellte Aufbau um eine Testung erweitert wird.



Legende

i_m	Verbindungsmittel
I	Eingabeeinheit, z. B. Sensor
L	Logik
m	Überwachung
O	Ausgabeeinheit, z. B. Hauptschütz
TE	Testeinrichtung
OTE	Ausgang der TE

Abbildung 8 Aufbau Kategorie 2, Quelle: EN ISO 13849-1, Seite: 49

Die Kategorie 2 hat folgende Eigenschaften:

- Maximal erreichbarer PL = d
- Maßnahmen gegen CCF
- $MTTF_D$ muss niedrig bis hoch sein
- Mindestens niedriger Diagnosedeckungsgrad

Wichtig ist, dass zwischen den Tests die Sicherheitsfunktion durch Auftreten eines Fehlers ausfallen kann, jedoch wird der Verlust erkannt.[18]

3.2.3.4 Kategorie 3

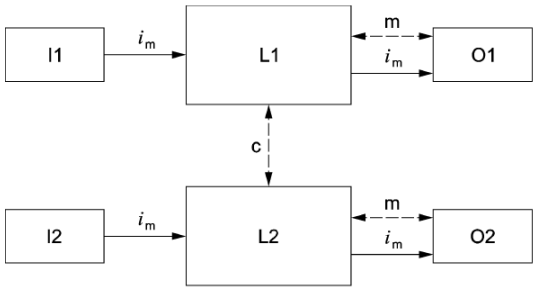
Bei der Kategorie 3 müssen die gleichen Anforderungen erfüllt werden, wie bei der in Abschnitt 3.2.3.1 vorgestellten Kategorie B, zusätzlich wird Folgendes gefordert:

- Bewährte Sicherheitsprinzipien

Die Architektur ist wie in Abbildung 9 ersichtlich, im Vergleich zu den bereits vorgestellten Kategorien zweikanalig, d.h. die Eingabeeinheiten, die Logik und die Ausgabeeinheit sind doppelt vorhanden. Das

[18] Vgl. EN ISO 13849-1:2016, Seite: 49-50

Signal wird an die Eingabeeinheit I1 und I2 angeschlossen z.B. zweikanaliger Not-Halt. Die Logik *L1* und *L2* überprüft sich durch einen Kreuzvergleich selbst. Ergänzend wird auch die Ausgabeeinheit überprüft z.B. Überprüfen durch Hilfskontakt, ob ein Schütz geschaltet hat.



Legende

i_m	Verbindungsmittel
c	Kreuzvergleich
I1, I2	Eingabeeinheiten, z. B. Sensor
L1, L2	Logik
m	Überwachung
O1, O2	Ausgabeeinheiten, z. B. Hauptschütz

Abbildung 9 Aufbau Kategorie 3, Quelle: EN ISOS 13849-1, Seite: 50

Die Kategorie 3 hat folgende Eigenschaften:

- Maximal erreichbarer PL = e
- Maßnahmen gegen das Auftreten von CCF
- $MTTF_D$ muss niedrig bis mittel sein
- Mindestens niedriger bis mittlerer Diagnosedeckungsgrad

Wichtig ist, dass durch das Auftreten eines einzelnen Fehlers kann die Sicherheitsfunktion nicht ausfallen, treten jedoch häufiger unerkannte Fehler auf, so kann die Sicherheitsfunktion verloren gehen.[19]

3.2.3.5 Kategorie 4

Bei der Kategorie 4 müssen die gleichen Anforderungen erfüllt werden wie bei der in Abschnitt 3.2.3.1 vorgestellten Kategorie B, zusätzlich wird Folgendes gefordert:

- Bewährte Sicherheitsprinzipien

Der Aufbau der Kategorie ist ident zu der in Abschnitt 3.2.3.3 vorgestellten Kategorie 3.

Die Kategorie 4 hat folgende Eigenschaften:

- Maximal erreichbarer PL = e
- Maßnahmen gegen das Auftreten von CCF
- $MTTF_D$ muss hoch sein
- Mindestens hoher Diagnosedeckungsgrad

[19] Vgl. EN ISO 13849-1:2016, Seite: 49-50

Wichtig ist, dass durch das Auftreten eines einzelnen Fehlers kann die Sicherheitsfunktion nicht ausfallen, aber wenn diese Erkennung nicht möglich ist, dann die Anhäufung von unerkannten Fehlern nicht zum Verlust der Sicherheitsfunktion führen.[20]

3.2.4 Berechnen und Überprüfen des erreichten Performance Levels

Wenn der PL_r für die Sicherheitsfunktion bestimmt ist, Komponenten und die zugehörige Architektur geplant wurden, muss der erreichte PL bestimmt werden.

Der erreichte PL ist von folgenden Faktoren abhängig:

- der Architektur
- der DC
- des CCF
- der $MTTF_D$ aller einzelnen Bauteile
- systematischen Ausfällen
- vom Verhalten der Sicherheitsfunktion bei Fehlern
- der sicherheitsbezogenen Software [21]

Um das Erreichen des PL zu berechnen, werden in der EN ISO 13849-1 verschiedene mathematische Methoden vorgestellt, diese werden jedoch in dieser Arbeit nicht näher betrachtet.

In der Industrie werden in den letzten Jahren vermehrt Softwaretools eingesetzt, um diesen Prozess zu erleichtern. Von der Deutschen Gesetzlichen Unfallversicherung wurde im Jahre 2008 das Softwaretool SISTEMA (Sicherheit von Steuerungen von Maschinen) vorgestellt. Über Eingabefenster wird nur der PL_r, die Kategorie der Steuerung, Maßnahmen gegen des CCF, der $MTTF_d$ und der mittlere DC eingegeben. Mittlerweile werden von vielen Herstellern bereits Bibliotheken zur Verfügung gestellt, in der die verschiedenen Parameter hinterlegt sind. Die Software berechnet automatisiert den erreichten PL und PFH_D und fasst die Ergebnisse in einem Report zusammen.[22]

Mit dem Softwaretool SISTEMA, aber auch mit der herkömmlichen Methode der EN ISO 13849-1 werden immer nur die Hardware und die Architektur verifiziert und validiert. Wie in Abbildung 10 zu sehen ist, wird in SISTEMA der Benutzer bei der Bearbeitung gefragt, ob für die Realisierung der Sicherheitsfunktion Software verwendet wird und ob diese nach Abschnitt 4.6 der EN ISO 13849-1 entwickelt worden ist. Auch der wichtige Punkt der Vermeidung von systematischen Fehlern wird bereits hier abgefragt.

[20] Vgl. EN ISO 13849-1:2016, Seite: 50-51

[21] Vgl. EN ISO 13849-1:2016, Seite: 25

[22] Vgl. Huelke (2007), Seite: 2-3

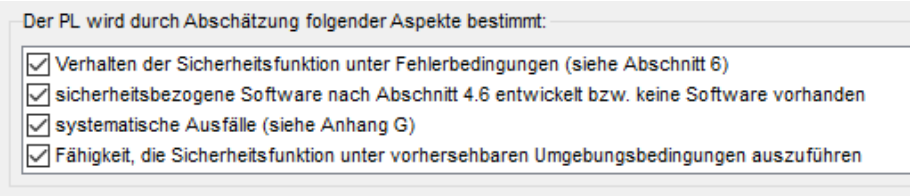


Abbildung 10 SISTEMA Systematische Ausfälle, Quelle: Eigene Darstellung

Der Teil der sicherheitsgerichteten Software wird immer umfangreicher, da die Maschinen immer größer und komplexer werden. Welche Maßnahmen zur Fehlervermeidung in der Softwareentwicklung gefordert sind und wie diese umgesetzt werden können, wird in den weiteren Ausführungen eingehend beschrieben.

4 ANFORDERUNG AN EINE SICHERHEITSBEZOGENE SOFTWARE

Im Bereich des Maschinenbaus findet hauptsächlich die Normenreihe EN ISO 13849 Anwendung. Es gibt jedoch auch andere Normen, welche ebenfalls unter der MSV 2010 harmonisiert sind z.B. EN 62061, diese beschränkt sich jedoch auf elektronische und programmierbar elektronische Systeme. Für Maschinen mit pneumatischen oder hydraulischen Steuerungsanteilen kann diese Norm gegebenenfalls nicht angewendet werden (siehe ISO/TR 23849 Leitfaden zur Anwendung von ISO 13849-1 und IEC 62061 bei der Gestaltung von sicherheitsbezogenen Steuerungen für Maschinen). Zum Thema sicherheitsgerichtete Software bzw. funktionale Sicherheit gibt es eine nicht harmonisierte Normenreihe EN 61508, welche dieses Thema sehr ausführlich beschreibt. Das Hauptanwendungsgebiet dieser Norm ist die Prozessindustrie, es können aber auch für die Gestaltung von sicherheitsbezogenen Steuerungen für Maschinen einige nützliche Informationen entnommen werden.[23]

Die MSV 2010 verlangt im Anhang I Abschnitt 1.2.1, dass es zu keiner Gefährdungssituation auf Grund des Defekts einer Hardware oder Software kommen darf. Überdies darf es auch durch einen Logikfehler nicht zu einer Gefährdungssituation kommen. Durch den Hersteller sollten auch vorhersehbare Fehlbedienungen berücksichtigt und vermieden werden, d.h. bereits in der MSV 2010 werden fehlervermeidende Maßnahmen und eine erhöhte Softwarequalität gefordert.[24]

Dieser Abschnitt beschreibt allgemein die Anforderungen und vorgeschlagenen Techniken der EN ISO 13849-1 und EN ISO 13849-2.

4.1 Grundlegende Forderungen der EN ISO 13849-1

Der Abschnitt 4.6 der EN ISO 13849-1 zeigt einen typischen Softwareentwicklungszyklus auf. Der Entwicklungszyklus, die Anforderungen an die Software selbst bzw. an die Entwicklungswerkzeuge und Entwicklungsaktivitäten werden in den folgenden Ausführungen behandelt. Es gibt, ähnlich wie bei der Hardware, mit zunehmendem PL_r erhöhte Anforderungen bzw. geforderte Wirksamkeit von Sicherheitsfunktionen.

Das Augenmerk muss während der Entwicklung einer sicherheitsgerichteten Software darauf gelegt werden, dass Tätigkeiten zur Vermeidung von systematischen Fehlern beinhaltet werden. Weitere Ziele sollen sein, eine verständliche, testbare, lesbare und wartbare Software zu entwickeln. Dies kann mit Hilfe des in Abschnitt 4.5 vorgestellten V-Modells erfolgen.[25]

[23] ISO/TR 23849:2010, Seite: 6-7

[24] Vgl. MSV 2010 (2020), Seite: 17

[25] Vgl. EN ISO 13849-1:2016, Seite: 32

4.1.1 Typen von Software

In der EN ISO 13849-1 werden zwei verschiedene Typen von Softwaresprachen kategorisiert.

4.1.1.1 LVL - Programmiersprache mit eingeschränktem Sprachumfang

LVL (Limited Variability Language) beschreibt den Typ einer Programmiersprache, welcher die Fähigkeit hat, anwendungsspezifische, vordefinierte Funktionen einer Bibliothek zu kombinieren, um somit die Anforderungen bzw. die Spezifikation der geforderten Sicherheitsfunktion zu realisieren.

Typische Beispiele sind: Kontaktplan, Funktions-Blockdiagramm. LVL wird in einer SPS (Speicher-programmierbare Steuerung) verwendet.[26]

4.1.1.2 FVL - Programmiersprache mit nicht eingeschränktem Sprachumfang

FVL (Full Variability Language) wird als Typ einer Programmiersprache bezeichnet, mit welcher ein großer Bereich von Funktionen und Anwendungen realisiert werden kann.

Typische Beispiele sind: C, C++, Assembler. FVL wird hauptsächlich in Embedded-Systems verwendet. [26]

4.1.2 Arten von Software

In der EN ISO 13849-1 werden zwei verschiedene Arten von Softwaresprachen kategorisiert.

4.1.2.1 SRASW - Sicherheitsbezogene Anwendungssoftware

SRASW (Safety Related Application Software) wird eine Software bezeichnet, welche Grenzwerte enthält und logische Sequenzen und Ausdrücke für die Steuerung der Eingänge und Ausgänge zur Verfügung stellt. Diese Software ist vom Hersteller für die Anwendung in der Maschine integriert. Weiters kann diese Berechnungen und Entscheidungen enthalten um den geforderten Anforderungen des SRP/CS (Safety-Related Parts of Control System) zu genügen.[27]

4.1.2.2 SRESW - Sicherheitsbezogene Embedded-Software

SRESW (Safety Related Embedded Software) wird eine Software bezeichnet, welche vom Anwender nicht verändert werden kann und vom Hersteller der Steuerung integriert ist. Typische Beispiele sind: Firmware, Betriebssystem usw.

Diese Arbeit behandelt nur die Anwendungssoftware, da diese im Bereich des Maschinenbaus hauptsächlich verwendet wird. Diese ist meistens in einer SPS Sprache programmiert und entspricht dem Sprachtyp LVL.[27]

[26] Vgl. EN ISO 13849-1:2016, Seite: 15

[27] Vgl. EN ISO 13849-1:2016, Seite: 33-34

4.2 Basismaßnahmen

Bei der Verwendung einer SRASW, welche in LVL geschrieben ist, ist grundsätzlich ein PL von a bis e erreichbar. Wenn eine SRASW jedoch in FVL geschrieben ist, gelten dieselben Anforderungen wie für eine SRESW und es ist auch ein PL a bis e erreichbar. Wenn Teile einer SRASW in der Hardware einen Einfluss z.B. Modifikation auf mehrere Funktionen mit nicht identen PL haben, dann muss immer der zugehörig höchste PL angenommen werden.

In den weiteren Ausarbeitungen wird nur von einer SRASW ausgegangen, welche in LVL geschrieben ist.

Folgende Maßnahmen müssen als Basismaßnahmen für PL_r a bis e angewendet werden:

- Entwicklungslebenszyklus mit Verifikation und Validierung (siehe V-Modell)
- Dokumentation der Spezifikation und Entwurf
- Modulare und strukturierte Programmierung
- Funktionale Tests
- Geeignete Entwicklungsaktivitäten nach Änderungen [28]

Sollte mit einer SRASW ein PL von c bis e erreicht werden, so müssen zusätzlich zu den Basismaßnahmen Maßnahmen mit steigender Wirksamkeit angewendet werden:

- Niedrige Wirksamkeit für PL_r bis c
- Mittlere Wirksamkeit für PL_r bis d
- Hohe Wirksamkeit für PL_r bis e

Die Wirksamkeit wird hier als Grad der Fehlervermeidung gesehen. Die weiteren Maßnahmen werden in Abschnitt 4.6 genauer erläutert.[28]

4.3 Grundlegende Forderungen der EN ISO 13849-2

Wenn die Entwicklung einer SRSW nach EN ISO 13849-1 abgeschlossen ist, muss die Software nach EN ISO 13849-2 validiert werden. Die Validierung ist durchzuführen, um eine Eignung der Software an der realen Maschine nachzuweisen. Im Anhang der EN ISO 13849-2 werden verschiedene Validierungswerkzeuge vorgestellt, welche auch bei einer SRASW zur Anwendung kommen können.

[28] EN ISO 13849-1:2016, Seite: 32-35

Grundlegende Sicherheitsprinzipien	Bemerkung
Vermeidung undefinierter Zustände	Undefinierte Zustände sollten vermieden werden. Die Steuerung sollte so realisiert sein, dass bei allen Betriebsbedingungen der Zustand z.B. des Ausgangs vorher bestimmt werden kann.
Zustandsausrichtung bei Ausfällen	Wenn möglich sollten alle Komponenten bei einem Ausfall in einen sicheren Zustand übergehen bzw. zu sicheren Bedingungen.
Verringerung der Fehlermöglichkeit	Sicherheitsbezogene Funktionen sollen von anderen Funktionen getrennt werden.
Gleichgewicht zwischen Komplexität und Vereinfachung	Zwischen der Komplexität der Komponenten sollte ein Ausgleich hergestellt werden, um einerseits eine bessere Steuerung zu realisieren und andererseits eine Vereinfachung der Einrichtungen zu bekommen. Damit kann die Zuverlässigkeit des Systems erhöht werden.

Tabelle 1 Bewährte Sicherheitsprinzipien, Quelle: EN ISO 13849-2, Seite: 51-53

In Tabelle 1 werden einige bewährte Sicherheitsprinzipien vorgestellt, welche für die Entwicklung einer SRASW angewendet werden können. Für bestimmte Fehler kann unter gewissen Voraussetzungen ein sogenannter Fehlerausschluss gemacht werden, wie im Beispiel der Tabelle 2 dargestellt ist.

Fehler	Fehlerausschluss	Bemerkung
Kurzschluss zwischen einem beliebigen Leiter und einem ungeschützten leitenden Teil oder der Erde oder einer Schutzleiterverbindung	Kurzschlüsse zwischen Leiter und jedem ungeschützten leitenden Teil innerhalb eines Einbauraumes.	Als Voraussetzung gilt, dass Leitungen und der Einbauraum lt. EN 60204-1 ausgeführt sind.

Tabelle 2 Fehlerausschluss, EN ISO 13849-2, Seite: 56

Dieser Fehlerausschluss ist für verschiedenste im Anhang D.2 der EN ISO 13849-2 aufgelistete Fehler gemacht worden. Für die Anwendung, bei der ein integrierter Schaltkreis z.B. SPS verbaut ist, ist kein Fehlerausschluss möglich (siehe Tabelle D.21 der EN ISO 13849-2).[29]

Weitere Anforderungen an die Validierung werden im Abschnitt 4.6.7 genauer erläutert.

[29] Vgl. EN ISO 13849-2:2013

4.4 Das V-Modell

Das V-Modell (Vorgehens-Modell) wurde im Jahre 1979 von Barry Boehms vorgestellt. Es basiert auf dem sogenannten Wasserfallmodell und wurde am Anfang im Bereich der Softwaretechnik eingeführt.[30] Die Idee hinter dem Modell in Abbildung 11 ist, dass die gesamte Softwareentwicklung in Phasen aufgeteilt wird. Das V-Modell unterscheidet zwischen den Entwicklungsphasen und den Testphasen zur Qualitätssicherung. Die linke Seite des Modells wird allgemein als konstruktive Maßnahme und die rechte Seite als integrative Seite bezeichnet. Nachdem die Anforderungen definiert wurden, folgt das Grobdesign, welches unter anderem die Systemarchitektur beinhaltet. Der technische Systementwurf ist eine Phase, in der definiert werden soll, auf welchen Systemen die Software lauffähig sein soll. Die Komponenten-Spezifikation ist ein Prozess des Feindesigns, danach findet erst die eigentliche Programmierung statt. Nachdem eine Entwicklungsphase abgeschlossen ist, sollen die Ergebnisse des vorangegangenen Prozesses geprüft und gegebenenfalls angepasst werden. Der Komponententest beinhaltet das Testen einzelner Software-Bausteine und es wird überprüft, ob das Ergebnis mit der Komponenten-Spezifikation übereinstimmt. Der Integrationstest beinhaltet das Überprüfen der Software und Hardware auf Lauffähigkeit gegenüber dem technischen Systementwurf. Beim Systemtest wird beim kompletten System getestet, ob die Anforderungen des Kunden erfüllt wurden. Als Letztes wird der Abnahmetest durchgeführt, welcher sich vom Systemtest nur insofern unterscheidet, dass dieser meistens gemeinsam mit dem Kunden durchgeführt wird.[31]

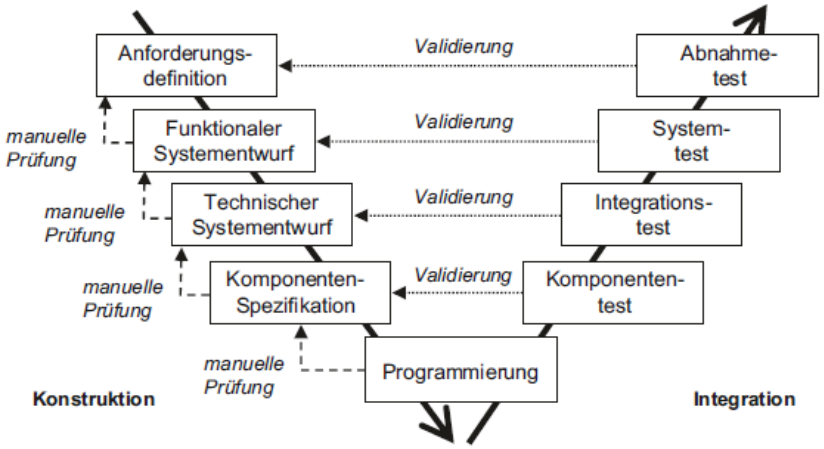


Abbildung 11 Allgemeines V-Modell, Quelle: Grundkurs Software-Engineering mit UML, Seite: 34

Das V-Modell wird mittlerweile nicht mehr nur für Softwareentwicklung verwendet, sondern für verschiedenste Bereiche der Entwicklung. Vom VDI (Verband Deutscher Ingenieure) wurde eine Richtlinie VDI 2206 „Entwicklungsmethodik mechatronischer Systeme“ entwickelt. Diese beinhaltet, wie in Abbildung 12 ersichtlich, ein V-Modell, das elektrische, mechanische und informationstechnische

[30] Vgl. Boehm (1979)

[31] Vgl. Kleuker (2008), Seite: 33-34

Komponenten miteinander verbindet, da diese im mechatronischen System immer öfter gemeinsam vorhanden sind.[32]

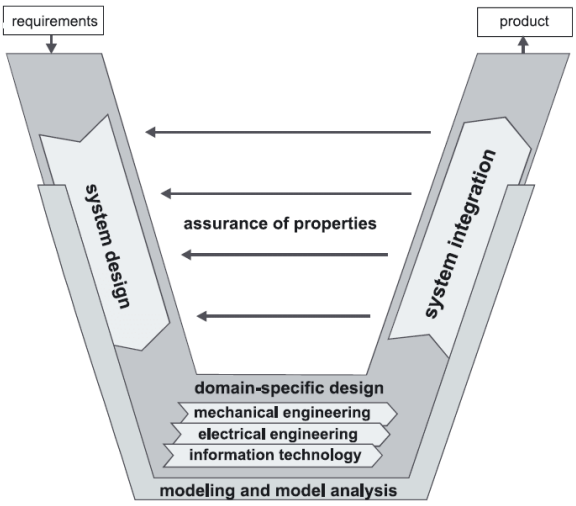


Abbildung 12 V-Modell VDI 2206, Quelle: NEW GUIDELINE VDI 2206 – A FLEXIBLE PROCEDURE MODEL FOR THE DESIGN OF MECHATRONIC SYSTEMS, Seite: 5

Die Idee des V-Modells wurde auch für den Bereich der funktionalen Sicherheit, im Speziellen der Entwicklung von sicherheitsgerichteter Software, übernommen. Verschiedenste Normen haben dieses Modell, wie in Abbildung 13 zu sehen ist, übernommen. In der detailliertesten Form wird dieses in der Normenreihe EN 61508 beschriebenen. Der hohe Detailierungsgrad ist hier notwendig, da das Anwendungsgebiet sehr weitläufig ist, es werden z.B. Gebiete der Industrieautomatisierung für Prozessindustrie bis hin zu einfacheren Steuerungsaufgaben abgedeckt.

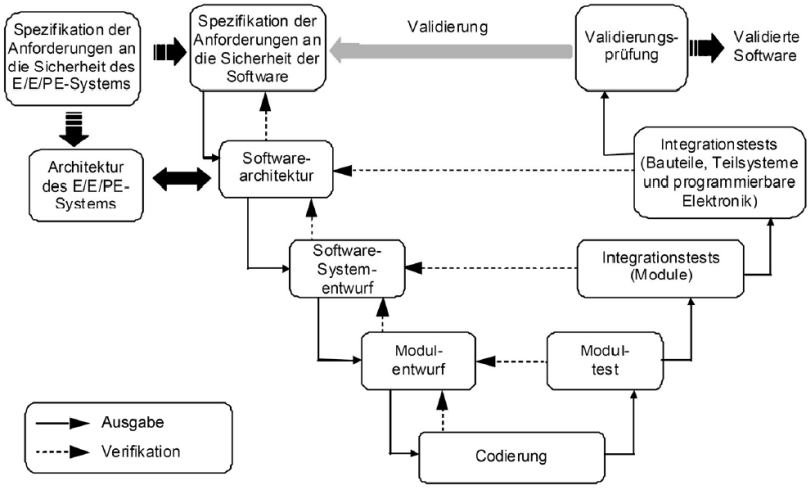


Abbildung 13 V-Modell, Quelle: EN 61508-3:2011, Seite 17.

[32] Vgl. J. Gausemeier (2003), Seite: 3-7

4.5 Vereinfachtes V-Modell des Software-Sicherheitslebenszyklus

Ausgehend vom V-Modell der Normenreihe EN 61508 wurde für den Anwendungsbereich der MSV 2010 mit der harmonisierten Norm EN 13849-1 das Modell der Abbildung 13 übernommen und vereinfacht. Eine Vereinfachung ist möglich, da im Bereich der Maschinen häufig eine Sicherheits - SPS verwendet wird. Die Phase der Softwarearchitektur entfällt hier, da die Architektur der Sicherheits - SPS vom Hersteller nach der EN 61131-3 entwickelt und zertifiziert wurde. Auch die Integrationstests der Hardware sind hier hinfällig, da die Software wie die Hardware direkt vom Hersteller der Sicherheits - SPS getestet wurden. Die Softwarearchitektur kann somit vom Anwender nicht beeinflusst werden und es wird auch ein geeignetes Programmiertool vom Hersteller zur Verfügung gestellt.

In Abbildung 14 ist das V-Modell der EN ISO 13849-1 zu sehen, welches so aufgebaut ist, dass es immer konstruktive Vorgänge gibt (linke Seite des Modells) und diesen gegenübergestellt immer überprüfende Vorgänge (rechte Seite des Modells). Das Ergebnis eines Vorgangs ist, was in diesem Vorgang erstellt wurde, dies kann z.B. Softwarecode sein. Die Ergebnisse der Vorgänge sind immer der Eingang für den nächsten Vorgang. Die Verifikation ist das Überprüfen eines Ergebnisses mit Vorgaben des Vorgangs. Am Ende wird die gesamte Software validiert. Damit wird überprüft, ob die gesamten Anforderungen der Softwarespezifikation umgesetzt wurden.

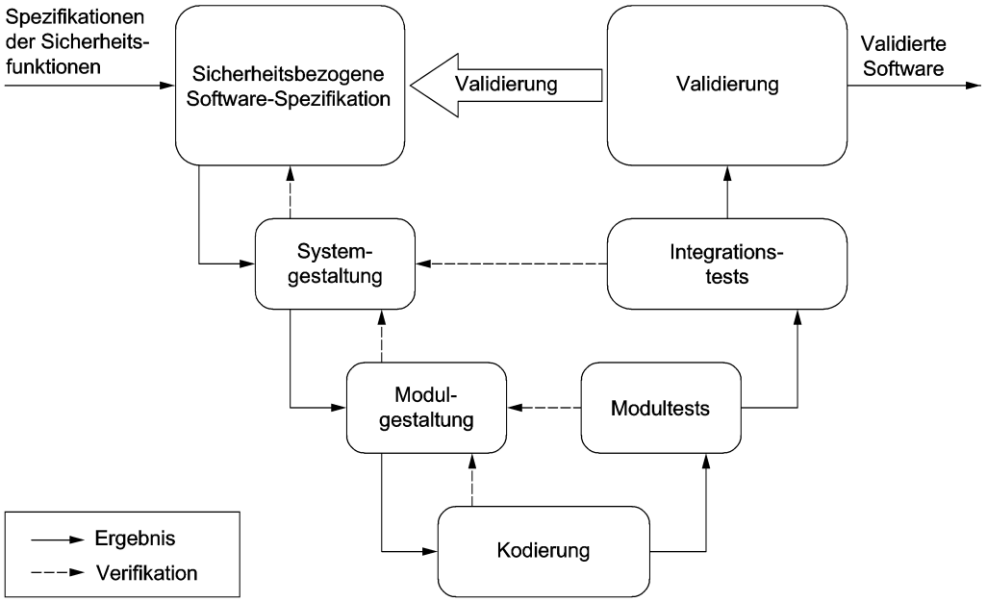


Abbildung 14 Vereinfachtes V-Modell, Quelle: EN ISO 13949-1: 2016, Seite 33.

Als Softwarespezifikation wird ein Dokument bezeichnet, in dem die Anforderungen an die Software beschrieben sind z.B. PL_r, Reaktionszeiten usw. Bei der Systemgestaltung sollten der Ablauf und der Aufbau und die verwendeten validierten Funktionsbausteine der Software grafisch dargestellt werden um die Software für Dritte verständlich zu machen. Die Modulgestaltung bezeichnet das Entwickeln von eigenen Funktionsbausteinen, hierbei müssen Schnittstellen und Testfälle für den Modultest festgelegt werden. Der letzte Punkt der konstruktiven Vorgänge, die Kodierung, ist die eigentliche Programmierung der SRASW. Zu beachten sind hier unter anderem Programmierrichtlinien (siehe Abschnitt 4.6.6) usw. Der Modultest umfasst das Testen der selbstentwickelten Funktionsbausteine auf die festgelegte

Modulgestaltung. Beim Integrationstest wird der korrekte Ablauf von Hardware und Software untersucht und überprüft, ob die Software nach der Systemgestaltung realisiert ist. Als Letztes wird die gesamte SRASW validiert, ob diese mit den Anforderungen der Software Spezifikation übereinstimmt.[33]

4.6 Weitere Maßnahmen

Um eine SRASW für Hardware mit einem PL_r von c bis e verwenden zu können, ist es notwendig zu den im Abschnitt 4.1 vorgestellten Basismaßnahmen weitere Maßnahmen mit steigender Wirksamkeit zu setzen. Im Abschnitt 4.6.3 der EN ISO 13849-1 werden diese für eine SRASW aufgezählt. In den folgenden Abschnitten werden diese aufgegriffen, vorgestellt und ergänzt.

4.6.1 Spezifikation der sicherheitsbezogenen Software

Als Erstes muss die Spezifikation der Sicherheitsfunktion überprüft werden. Alle Personen, die am Lebenszyklus der sicherheitsbezogenen Software beteiligt sind, müssen eine Beschreibung der Sicherheitsfunktionen mit den Betriebsarten und den dazugehörigen PL erhalten. Weiters sollten sie Auskunft erhalten über Leistungskriterien wie z.B. Reaktionszeiten, Hardwarearchitektur mit externen Signalschnittstellen und das Erkennen und die Beherrschung externer Ausfälle.[34]

Der Anhang J.3 der EN ISO 13849-1 gibt weitere informative Hinweise, wie die Softwarespezifikation verifiziert werden kann. Es ist nachzuweisen, dass in der Spezifikation alle sensiblen Punkte aufgelistet sind. Des Weiteren sollen Fälle vermieden werden, wo es möglicherweise zu falschen Interpretationen der Spezifikation kommen könnte. Die Spezifikation sollte so gestaltet werden, dass es zu keinen Lücken kommen kann. Es sollen die Bedingungen genannt werden, wie Funktionen aktiviert oder deaktiviert werden können. Es sollen Konsistenzprüfungen gemacht werden und es soll eine Garantie abgegeben werden, dass alle möglichen auftretenden Fälle betrachtet wurden. Es soll auch genau definiert werden, was geschehen soll, wenn es zu einem Ausfall kommt.[35]

4.6.2 Wahl der Sprachen und Bibliotheken

Es wird empfohlen, für die Programmierung eine modulare anerkannte LVL Sprache lt. EN 61131-3 zu verwenden. Besonders empfohlen werden hier z.B. Funktionsplan und Kontaktplan. Wenn möglich sollen nur validierte Funktionsbaustein - Bibliotheken verwendet werden. Es ist empfohlen, besonders bei einem PL_r = e diese aus einer Hersteller - Funktionsbaustein - Bibliothek zu verwenden. Als Alternative können

[33] Vgl. Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)(DGUV) (Hrsg.) (2017), Seite: 65-67

[34] Vgl. EN ISO 13849-1:2016, Seite 35

[35] Vgl. EN ISO 13849-1:2016, Seite: 68

validierte Funktionsbausteine, welche in Übereinstimmung mit der EN ISO 13849-1 entwickelt wurden, verwendet werden.[36]

4.6.3 Geeignete Werkzeuge mit Betriebsbewährung

Wenn ein PL_r = e gefordert wird, ist besonders darauf zu achten, dass das Programmiertool einer geeigneten Sicherheitsnorm entspricht. Bei der Verwendung von zwei redundanten Komponenten, welche ein verschiedenes Programmiertool verwenden, kann Betriebsbewährung ausreichend sein. Die Fähigkeiten eines Programmiertools sollen sein, dass Bedingungen erkannt werden, die zu systematischen Fehlern führen können z.B. Rekursion, Datentyp - Unverträglichkeit usw. Die Überprüfung auf Fehler sollte vor dem Laden in die Hardware geschehen, es sollte also bei der Kompilierung auf Fehler überprüft werden. Programmiertools sollten den Programmierer durch den Entwicklungsprozess leiten und bestimmte Programmierrichtlinien erzwingen.[36]

4.6.4 Merkmale des Softwareentwurfs

Die Codelänge von Funktionsblöcken sollte immer minimal sein, die Programmierung sollte strukturiert und modular sein. Eine solche Realisierung kann mit validierten sicherheitsbezogenen Funktionsbaustein-Bibliotheken umgesetzt werden. Der Code sollte innerhalb eines Funktionsblocks immer so ablaufen, dass es einen Eingangssprung und einen Ausgangssprung gibt. Die Architektur des Modells sollte in drei Stufen erfolgen: Eingänge → Verarbeitung → Ausgänge wie in Abbildung 15 zu sehen ist.[37]

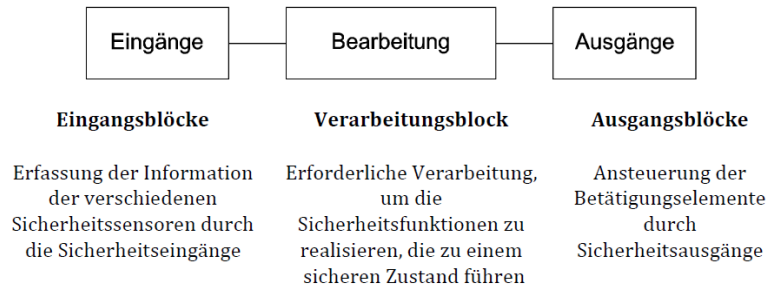


Abbildung 15 Architekturmodell der Software, Quelle: EN ISO 13849-1:2016, Seite 36.

Das oberste Ziel bei der Programmierung ist die Modularisierung und Strukturierung der Programmierung. Die EN ISO 13849-1 gibt über die genauere Bedeutung keine Auskunft, in der EN 61508-7 werden diese zwei Begriffe genauer erläutert.

Modularisierung:

Das Ziel der Modularisierung ist es, die Vermeidung von gefährlichen Ausfällen und die Reduzierung der Komplexität, in Bezug auf Schnittstellen zwischen den Teilsystemen. Es soll die Größe des Teilsystems beschränkt werden und im Entwurf soll jedes Teilsystem klar definiert sein. Zwischen den Teilsystemen

[36] Vgl. EN ISO 13849-1:2016, Seite: 35

[37] Vgl. EN ISO 13849-1:2016, Seite: 35-36

sollen die Schnittstellen so einfach wie möglich realisiert werden. Der Austausch von Daten sollte minimiert werden. Ein weiterer wichtiger Punkt ist, dass die Komplexität der einzelnen Teilfunktionen beschränkt werden soll.[38]

Strukturierung:

Das Ziel der Strukturierung ist, durch hierarchische Strukturen die Komplexität zu reduzieren. Dies soll dazu führen, dass Schnittstellen zwischen Anforderungen nicht ausfallen. Es soll zwischen den Teilanforderungen eine möglichst einfache Beziehung herrschen. Die Funktionen sollen so weit zerlegt werden, dass man kleine eindeutige Teilanforderungen unterscheiden kann. Das Ziel der Zerlegung soll sein, dass eine hierarchische Struktur von Teilanforderungen entsteht. Mit dieser Methode können wirksam Schnittstellenausfälle vermieden werden.[39]

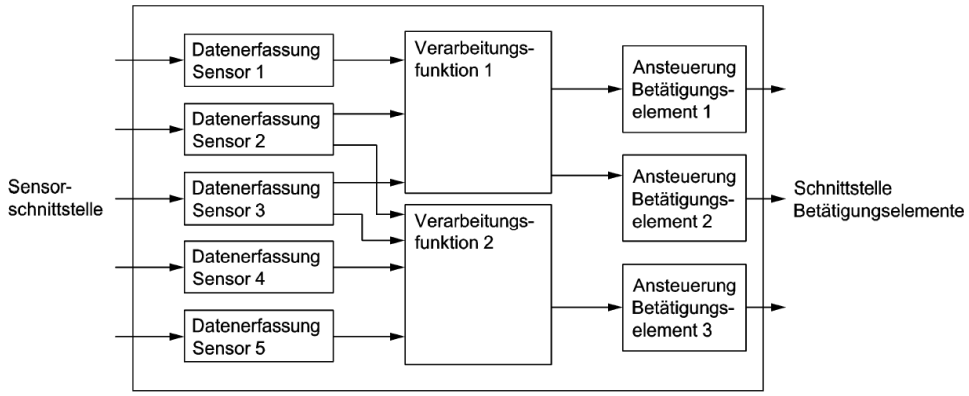


Abbildung 16 Entwurf eines Softwarebeispiels, Quelle: EN ISO 13849:2016, Seite: 97

In Abbildung 16 ist ein Beispiel dieser Modularisierung zu sehen. An der linken Seite werden die Signale im Eingangsblock erfasst, in der Mitte werden die erfassten Signale im Verarbeitungsblock verarbeitet und an der rechten Seite wird das verarbeitete Signal im Ausgangsblock wieder ausgegeben.

4.6.5 Trennung von sicherheitsbezogener und nicht sicherheitsbezogener Software

Bei den meisten Anwendungen laufen auf einer Hardware eine SRASW und nicht SRASW zugleich. Es ist daher eine strikte Trennung erforderlich, da die nicht SRASW eventuell öfters geändert bzw. modifiziert wird und dies sollte ohne Einfluss auf die SRASW geschehen. In den meisten Sicherheitssteuerungen namhafter Hersteller sind diese getrennt.

Die SRASW sollte nur über definierte Datenschnittstellen mit der nicht SRASW kommunizieren, ergänzend sollen beide in unterschiedlichen Funktionsblöcken programmiert werden. Es darf auch keine Verknüpfung zwischen sicherheitsbezogenen und nicht sicherheitsbezogenen Daten geben, welche die Integrität von

[38] Vgl. EN 61508-7:2011, Seite 45

[39] Vgl. EN 61508-7:2011, Seite 37-38

sicherheitsbezogenen Signalen beeinflussen könnte z.B. ein sicherheitsbezogener Ausgang wird durch eine logische *ODER* Verknüpfung eines nicht sicherheitsbezogenen und eines sicherheitsbezogenen Signals gesteuert.[40]

4.6.6 Programmierregeln

Um einen verständlichen bzw. einen zu einem späteren Zeitpunkt modifizierbaren Softwarecode zu gewährleisten, ist es wichtig Programmierregeln aufzustellen. Diese Regeln werden allgemein im Anhang J.4 der EN ISO 13849-1 dargestellt, durch den Anwender können diese ergänzt oder geändert werden.

Grundsätzlich sollte jedes Programm durch den Autor, das Datum des letzten Ladens, die Version und den letzten Zugriff identifizierbar sein.

4.6.6.1 Programmierregeln auf Ebene der Programmstruktur

Für öfters verwendete Programm- und Funktionsblöcke sollen Vorlagen verwendet werden, des Weiteren soll das Programm aufgeteilt werden und die verschiedenen Bereiche sollen gekennzeichnet werden mit „Eingaben“, „Verarbeitung“ und „Ausgabe“. Jeder Abschnitt der Software soll kommentiert werden, damit im Falle einer Modifizierung die Aktualisierung erleichtert wird. Bei dem Aufruf eines Funktionsblocks soll beschrieben werden, welche Aufgabe dieser hat. Der Speicherbereich soll nur durch einen eindeutig gekennzeichneten einzelnen Datentyp verwendet werden. Der Programmablauf soll unabhängig von Sprungadressen und Variablen sein, welche während der Laufzeit berechnet werden. Erlaubt sind nur bedingungsunabhängige Sprünge.[41]

4.6.6.2 Programmierregeln bezüglich der Verwendung von Variablen

Globale Variablen, Output und Input sollen einen eindeutigen Namen erhalten, welcher auf seine Funktionalität hinweist. Das Ansteuern der Ausgänge sollte pro Zyklus nur einmal erfolgen z.B. am Ende des Softwarecodes. Die Strukturierung des Programms sollte so sein, dass die Operationen zum Aktualisieren einer Variable zentral abgearbeitet werden.[41]

4.6.6.3 Programmierregeln auf der Ebene des Funktionsblocks

Es sollten vorzugsweise nur Funktionsblöcke verwendet werden, die durch den Hersteller der SRP/CS getestet wurden. Für die Größe des programmierten Softwarecodes sollten folgende Richtwerte eingehalten werden:

- Parameter sollen maximal acht digitale, zwei Integer Eingänge und einen Ausgang besitzen.
- funktionaler Code soll aus maximal zehn lokalen Variablen und maximal 20 booleschen Operationen bestehen.

[40] Vgl. EN ISO 13849-1:2016, Seite 36

[41] Vgl. EN ISO 13849-1:2016, Seite: 99

Globale Variablen sollten nicht von Funktionsblöcken verändert werden. Um den Gültigkeitsbereich sicherzustellen, sollen digitale Werte zuvor von vorgegebenen Vergleichstests kontrolliert werden. Jeder Funktionsblock sollte versuchen bei Variablen, Widersprüchlichkeiten aufzudecken. Um eine Unterscheidung von Fehlern zu gewährleisten, sollte jeder Funktionsblock offen bzw. zugänglich sein. Außerdem sollten die Fehlercodes und der Zustand nach Auftreten des Fehlers im Kommentar vermerkt sein. Letzteres soll das Rücksetzen des Funktionsblocks zum Ursprung bzw. Ausgangszustand im Kommentar beschreiben.[42]

4.6.7 Test der Software

Es wird in der Normenreihe EN ISO 13849 zwischen dem funktionalen Test, welcher in der EN ISO 13849-1 gefordert wird, und dem in der EN ISO 13849-2 geforderten erweiterten Funktionstest unterschieden.[43]

In den folgenden Ausführungen wird davon ausgegangen, dass SRASW umfangreicher und komplexer ist, da es bei kleineren Programmen ausreichend sein kann eine Programmanalyse zu machen. Diese wird durch Analyse des Kontrollflusses und Nachprüfung der Prozeduren durchgeführt. Zum Überprüfen werden der Softwarecode von Funktionsblöcken, I/O Zuweisungen und das Kontrollflussdiagramm verwendet.[43]

Zu Beginn der Prüfung sollte immer die Dokumentation überprüft werden, welche die Spezifikation und die Gestaltung der SRASW enthalten muss. Beim Überprüfen der Dokumentation sollte darauf geachtet werden, dass eine fehlerhafte Auslegung vermieden werden kann, es keine Widersprüche gibt und die Dokumentation vollständig vorhanden ist.[43] Die Dokumentation wird ausführlich im Abschnitt 4.6.8 beschrieben.

4.6.7.1 Funktionaler Test

Das zu verwendende Verfahren soll der sogenannte Black Box Test sein, mit welchem die Leistungskriterien z.B. zeitliches Leistungsverhalten und das funktionale Verhalten getestet werden können. Für die Testplanung sollten fixe Testfälle mit Abschlussbedingungen fixiert werden. Mit dem I/O Test soll sichergestellt werden, dass alle sicherheitsbezogenen Signale der SRASW in richtiger Art und Weise verwendet werden. Wenn ein PL d oder e gefordert ist, müssen auch Testfälle auf Basis von Grenzwertanalysen, wie in Abschnitt 4.6.7.2 beschrieben, ausgeführt werden. [44]

[42] Vgl. EN ISO 13849-1:2016, Seite: 36

[43] Vgl. EN ISO 13849-2:2013, Seite 20

[44] Vgl. EN ISO 13849-1:2016, Seite: 37

4.6.7.2 Erweiterter Funktionstest

Das Ziel dieser Tests soll sein, Fehler während der Entwicklung aufzudecken. Hierbei wird das Verhalten getestet bei selten oder nicht festgelegten Eingaben.

Beim erweiterten Funktionstest wird das funktionale Verhalten des Systems auf selten auftretendes Eingangsverhalten getestet z.B. Ausfall der Steuerung oder Peripherie. Eine weitere Funktionsprüfung ist das Verhalten auf nicht in der Spezifikation vermerkten Eingabebedingungen z.B. nicht ordnungsgemäßer Betrieb. Für Bedingungen, die nur sporadisch auftreten, wird die Spezifikation mit dem Verhalten des Systems gegenübergestellt. Die Sicherheit einer Funktion muss erneut überprüft werden, wenn ein Verhalten im System nicht charakterisiert ist.[45]

Analoge Signale sollten mit Testfällen nach einer Grenzwertanalyse durchgeführt werden. Diese soll Softwarefehler entdecken, wenn Parameter oder Eingänge bestimmte Grenzen erreichen. Beispiele dafür sind:

- Division durch Null
- volles Array
- leeres Listenelement
- Überlauf eines Datentyps

Es sollten auch Testfälle gemacht werden, bei denen der Ausgang über die Grenzwerte hinaus erzwungen wird.[46]

4.6.7.3 Fehlersimulation und Vorgehensweise bei validierten Bausteinen

Es sollten Prüffälle erstellt werden, bei der ein Fehlerfall simuliert und das vorausgesagte Ergebnis, mit dem wirklichen Ergebnis verglichen wird. Damit kann die Eignung der Software bzw. können die Maßnahmen der Fehlerbeherrschung quantifiziert werden.[47]

Es ist keine erneute Validierung für bereits validierte Softwarefunktionen oder Funktionsbausteine nötig. Wenn diese jedoch mit anderen Funktionen verschaltet oder kombiniert werden, muss die sich ergebende Sicherheitsfunktion validiert werden.[47]

4.6.7.4 Testabdeckung

Im unternehmerischen Umfeld werden Tests bzw. der Testaufwand in Abhängigkeit des PL durchgeführt. Die Höhe der Testabdeckung wird in der EN ISO 13849-1 nicht genauer spezifiziert. Als Hilfestellung wird hier die EN 61508-3 gewählt.

---

[45] Vgl. EN 61508-7:2011, Seite: 61

[46] Vgl. EN 61508-7:2011, Seite: 99

[47] Vgl. EN ISO 13849-2:2013, Seite: 21

Wenn Bausteine aus einer Funktionsbaustein - Bibliothek des Herstellers verwendet werden, müssen diese nicht getestet werden. Es ist hier nur die Verschaltung untereinander und die Parametrisierung zu testen.

Testabdeckung	SIL1 / PL a, b, c	SIL2 / PL d	SIL3 / PL e
100% der Eingänge	erforderlich	erforderlich	erforderlich
100% der Anweisungen	nicht erforderlich	erforderlich	erforderlich
100% der Verzweigungen	nicht erforderlich	nicht erforderlich	erforderlich

Tabelle 3 Testabdeckung, Quelle: EN 61508-3:2011, Seite: 60 (leicht modifiziert)

Wie in Tabelle 3 ersichtlich, hängt die Testabdeckung mit dem PL zusammen. Wenn ein PL von a bis e erreicht werden soll, müssen alle sicherheitsrelevanten Eingänge bzw. alle Schutzeinrichtungen getestet werden. Um einen maximalen PL von d zu erreichen, ist es zusätzlich notwendig, alle Programmanweisungen zu testen, d.h. jede Sicherheitsfunktion bzw. jeder Funktionsbaustein soll getestet werden. Wenn ein PL von e erreicht werden soll, ist es zusätzlich zu den zuvor erwähnten Tests notwendig, alle Verzweigungen zu testen. Dies bedeutet, dass durch Fehlersimulation die Diagnosefunktion getestet werden soll, zusätzlich soll es auch einen Test des Wiederanlaufens geben.

4.6.8 Dokumentation von sicherheitsgerichteter Software

Die Software sowie auch die gesamte Hardwareplanung müssen dokumentiert werden. Ein wichtiger Punkt ist auch, dass Änderungen vermerkt werden. Zur Qualität der Dokumentation ist zu sagen, dass diese vollständig, allgemein verständlich, lesbar und verfügbar sein sollte. Die Softwaredokumentation sollte in Modulköpfe gegliedert werden. In diesen sollte, die Funktion, eine I/O Beschreibung, der Programmierer, die Version der Funktionsbaustein - Bibliothek sowie eine verständliche Kommentierung der Anweisungen vermerkt werden.[48]

Das Thema Dokumentation wird in der EN ISO 13849-1 nur sehr knapp beschrieben, genauere Informationen zum Thema Dokumentation gibt es in der EN 61508-1, diese werden hier ergänzt.

Es steht dem Hersteller frei, wie bzw. in welcher Form die Dokumentation vorhanden ist z.B. Papier oder Datenmedium. Die Dokumentation muss Informationen beinhalten über die Ausführung der Beurteilung der funktionalen Sicherheit, damit mit den Informationen und Ergebnissen aus jeder Beurteilung die funktionale Sicherheit abgeleitet werden kann. Die Dokumentation sollte einen Titel haben, welche auf den Inhalt hinweist und weiters soll eine Art von Registerteilung vorhanden sein, welche auf die zutreffenden Normen verweist. Die Struktur ist frei zu wählen und kann nach den Gegebenheiten des Unternehmens bzw. nach den firmeninternen Vorgaben aufgebaut werden. Ein wichtiger Punkt ist, dass die Dokumentation immer mit einer Versionsnummer versehen wird, damit die verschiedenen Versionen unterschieden werden können. Überdies ist es ratsam, die Dokumentation mit einem angemessenem System der Dokumentenlenkung zu versehen, wie sie im Abschnitt 4.7 beschrieben wird.[49]

[48] Vgl. EN ISO 13849-1:2016, Seite: 37

[49] Vgl. EN 61508-1:2011, Seite: 59

4.6.9 Konfigurationsmanagement und Änderungen

Für SRASW ist es wichtig Verfahren zur Datensicherung zu verwenden, damit Softwaremodule, Ergebnisse der Verifikation bzw. Validierung, Dokumente und das Programmiertool identifiziert bzw. archiviert werden können. Wenn Änderungen der SRASW vorgenommen werden, muss sichergestellt werden, dass die Spezifikation der Sicherheitsfunktionen noch immer erfüllt wird. Um missbräuchliche Änderungen zu vermeiden, ist es ratsam Zugriffsrechte zu vergeben z.B. Programmschlüssel. Ergänzend müssen alle Änderungen dokumentiert werden und wieder mit dem Dokumentenlenkungssystem verwaltet werden.[50]

Die EN ISO 13849-1 beschreibt die erwähnten notwendigen Schritte nur sehr allgemein, in der EN 61508-3 werden diese genauer beschrieben und es gibt Vorschläge, wie diese in der Praxis umgesetzt werden könnten.

Folgende Konfigurationsmerkmale sind notwendig um die Integrität des SRP/CS zu gewährleisten:

- Sicherheitsanalyse und Sicherheitsanforderungen
- Spezifikation der Software und Entwurfsdokumente
- Softwaremodule im Source Code
- Testpläne und Testergebnisse
- Verifikationsdokumente
- verwendete Bausteine aus der Funktionsbaustein - Bibliothek
- verwendetes Programmiertool [51]

Es sollen Verfahren angewendet werden, um Änderungen zu kontrollieren:

- Nicht autorisierte Modifikation verhindern z.B. Programmschlüssel
- Mindestanforderung an die Konfiguration in der Softwareentwicklung festlegen und Integrationstests der Anforderungen dokumentieren
- Einzelheiten der Änderungen und die Bevollmächtigung dokumentieren
- Aufbau und Zusammenstellung um Mindestanforderung der Software gewährleisten zu können [51]

Alle Änderungen bzw. die Konfiguration müssen angemessen dokumentiert werden, um bei einem Audit der funktionalen Sicherheit vorgelegt werden zu können. Folgende Punkte müssen unbedingt dokumentiert sein:

- Begründung der Einflussnahme
- Genehmigung der Modifikation
- Status der Konfiguration
- Version
- alle Einzelheiten der Änderungen [51]

[50] Vgl. EN ISO 13849-1:2016, Seite: 37

[51] Vgl. EN 61508-3:2011, Seite: 13-14

4.7 Management funktionaler Sicherheit

Das Ziel des Managements funktionaler Sicherheit ist es, Verantwortlichkeiten und Tätigkeiten festzulegen, welche beim Management funktionaler Sicherheit auszuführen sind. Die Organisation bzw. der Hersteller muss eine Person bestimmen, welche die Koordination sicherheitsbezogener Tätigkeiten innehat. Des Weiteren hat diese Person Verantwortung für das System und seine Lebenszyklen. Die Person soll als Schnittstelle zwischen eventuell von anderen Unternehmen ausgeführten Phasen dienen. Es ist zu empfehlen, dass die Gesamtkoordination von nur einer Person durchgeführt wird. Die Verantwortung von einzelnen Tätigkeiten während des Sicherheitslebenszyklus kann auf andere mit angemessener Qualifizierung delegiert werden. Während der Beurteilung funktionaler Sicherheit ist der Verantwortliche dafür zuständig, dass die Planung, Integration, Kommunikation, Dokumentation und Empfehlungen gemäß den Normen, umgesetzt werden. Der Verantwortliche hat außerdem dafür zu sorgen, dass die Dokumentation, wie in Abschnitt 4.6.8 beschrieben, entsprechend verwaltet wird.[52]

Für die Dokumentenlenkung wird ein Qualitätsmanagementsystem z.B. ISO 9001 empfohlen.

4.8 Unabhängigkeitsgrade bei der Überprüfung

Im Prozess der Entwicklung der SRASW gibt es laufend überprüfende Tätigkeiten, wie die Validierung und die Verifikation. Diese haben den Sinn, dass z.B. Fehler des Programmcodes rechtzeitig erkannt werden und in weiterer Folge behoben werden können. Die Definition der Person, welche die Validierung durchführen sollte, ist in der EN ISO 13849-2 beschrieben.

Die Validierung sollte nicht von derselben Person durchgeführt werden, welche für die Entwicklung der SRP/CS zuständig ist. Sofern gesetzlich nicht anderes geregelt, kann die Prüfung in der Organisation selbst vorgenommen werden und muss nicht von einer externen Prüfstelle gemacht werden.[53]

In der EN ISO 61508-1 wird der Unabhängigkeitsgrad genau in Abhängigkeit des PL definiert, wie „unabhängig“ die prüfende Person sein muss. Jedoch kann diese Vorgabe nicht 1:1 übernommen werden, da die EN 61508-1 typischerweise für komplexere Sicherheitsfunktionen angewendet wird z.B. in der Prozessindustrie.

Unabhängigkeitsgrad	PL a, b und c	PL d	PL e
Unabhängige Person	empfohlen	nicht ausreichend	nicht ausreichend
Unabhängige Abteilung	nicht notwendig	empfohlen	nicht ausreichend
Unabhängige Organisation	nicht notwendig	nicht notwendig	empfohlen

Tabelle 4 Unabhängigkeitsgrade, Quelle: EN 61508-1, Seite: 58 (leicht modifiziert)

[52] Vgl. EN 61508-1:2011, Seite: 12-15

[53] Vgl. EN ISO 13849-2:2013, Seite: 7

Definitionen der Unabhängigkeitsgrade:

- **Unabhängige Person:**
  wird definiert als eine Person, die nicht aktiv eingebunden ist in die überprüfende Tätigkeit und sie trägt auch keine Verantwortung für ihr Handeln z.B. Hardware Entwickler.
- **Unabhängige Abteilung:**
  wird definiert als eine Abteilung, die nicht aktiv an der Entwicklung der SRASW beteiligt ist z.B. Qualitätssicherung.
- **Unabhängige Organisation:**
  wird definiert als eine Organisation, die in keiner Weise in Verbindung mit der Entwicklungsorganisation steht z.B. Prüfstelle.[54]

In Tabelle 4 ist ersichtlich, dass es für einen PL a, b und c es ausreichend ist, wenn eine unabhängige Person die SRASW überprüft. Da die Komplexität und das Schadensausmaß bei dem PL d höher ist, wird hier eine Prüfung von einer unabhängigen Abteilung, die nicht beim Projekt beteiligt ist, gefordert. Bei dem höchsten PL e, ist eine Prüfung von einer unabhängigen Organisation gefordert.

[54] Vgl. EN 61508-4: 2001, Seite: 34

5 DIE MATRIXMETHODE DES INSTITUTS FÜR ARBEITSSICHERHEIT

Im folgenden Kapitel wird die sogenannte Matrixmethode des IFA (Institut für Arbeitssicherheit) der DGUV (Deutsche gesetzliche Unfallversicherung) vorgestellt. Von 2011 bis 2013 wurde das Projekt FF-FP0319 durchgeführt, Ziel dieses Projekts war es, die normgerechte Entwicklung und Dokumentation von sicherheitsgerichteter Anwendungssoftware darzustellen. Als Ergebnis dieses Forschungsprojektes wurde die sogenannte Matrixmethode vorgestellt.[55]

5.1 Allgemeines

Um die Matrixmethode anwenden zu können, muss die SRASW laut EN ISO 13849-1 folgende Softwarestruktur aufweisen:

- Eingangsblock
- Verarbeitungsblock
- Ausgangsblock

Für den Eingangsblock und den Ausgangsblock werden meist vom Hersteller gelieferte Funktionsbausteine einer validierten Funktionsbaustein - Bibliothek verwendet, d.h. es ist lediglich auf die Ansteuerung des Funktionsbausteins des Verarbeitungsblocks zu achten. Da die Struktur hier vorgegeben ist, wird kein extra Vorgang der Systemgestaltung und des Integrationstests benötigt, dadurch kann das V-Modell der EN ISO 13849-1 weiter vereinfacht werden. Die Vereinfachung sieht so aus, dass der Vorgang der Spezifikation und der Systemgestaltung zum Vorgang des Softwareentwurfs zusammengefasst werden kann. Wenn bei der Entwicklung einer SRASW Funktionen benötigt werden, welche nicht direkt vom Hersteller in der Funktionsbaustein - Bibliothek vorhanden sind, dann ist es notwendig, diese selbst zu erstellen und zu validieren. Dies kann in einem eigenen separaten V-Modell erfolgen.[56]

[55] Vgl. Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)(DGUV) (Hrsg.) (2016), Seite: 9

[56] Vgl. Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)(DGUV) (Hrsg.) (2016), Seite: 21-23

Im Abschnitt 4.5 wurde das vereinfachte V-Modell der EN ISO 13849-1 vorgestellt. Ausgehend von diesem Modell kann dies noch wie folgt vereinfacht werden, sodass sich zwei kleinere V-Modelle ergeben:

- **V-Modell zur Realisierung einer Sicherheitsfunktion**

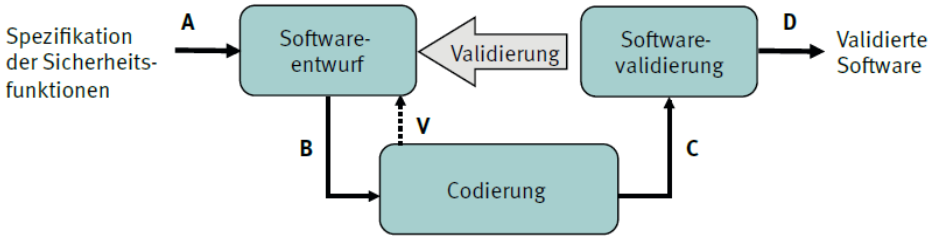


Abbildung 17 V-Modell zur Realisierung von SF, Quelle: IFA Report 2/2016, Seite: 23

Dieses Modell wird in den weiteren Ausführungen behandelt und wird auch im Abschnitt 6 mit Hilfe des Softwareassistenten SOFTEMA praktisch umgesetzt.

- **V-Modell zur Entwicklung eines wiederverwendbaren Funktionsbausteins**

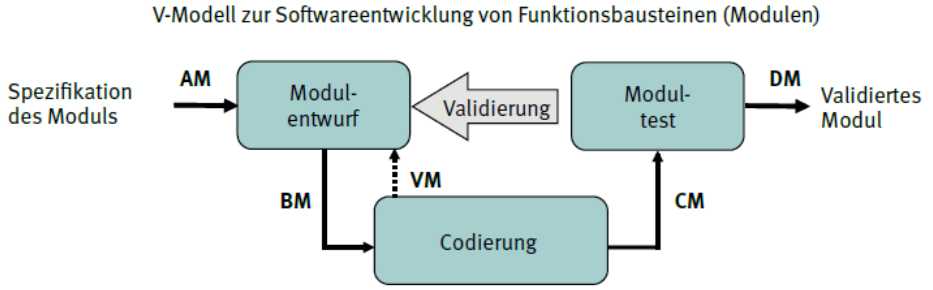


Abbildung 18 V-Modell zur Entwicklung eines FB, Quelle: IFA Report 2/2016, Seite: 23

Die Entwicklung und Überprüfung von eigenen Funktionsbausteinen wird in Abschnitt 6.4.1 genauer erläutert.

5.2 Funktionsweise der Matrixmethode

Die Matrixmethode ist eine auf Excel Tabellen aufbauende Methode, um Software zu spezifizieren und zu dokumentieren. Um diese Methode anwenden zu können, ist es wichtig, die Softwarestruktur, wie in Abschnitt 4.6.4 beschrieben, aufzubauen. Diese Dreiteilung der Software, wie in Abbildung 19 ist, essenziell, da der Eingangsblock und der Ausgangsblock meistens mit Bausteinen einer Hersteller - Funktionsbaustein - Bibliothek umgesetzt werden können. Vom Anwender ist lediglich noch der Verarbeitungsblock mit Hilfe der sogenannten C+E Matrix (Cause & Effect Matrix) zu spezifizieren.[57]

[57] Vgl. Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)(DGUV) (Hrsg.) (2016), Seite: 33

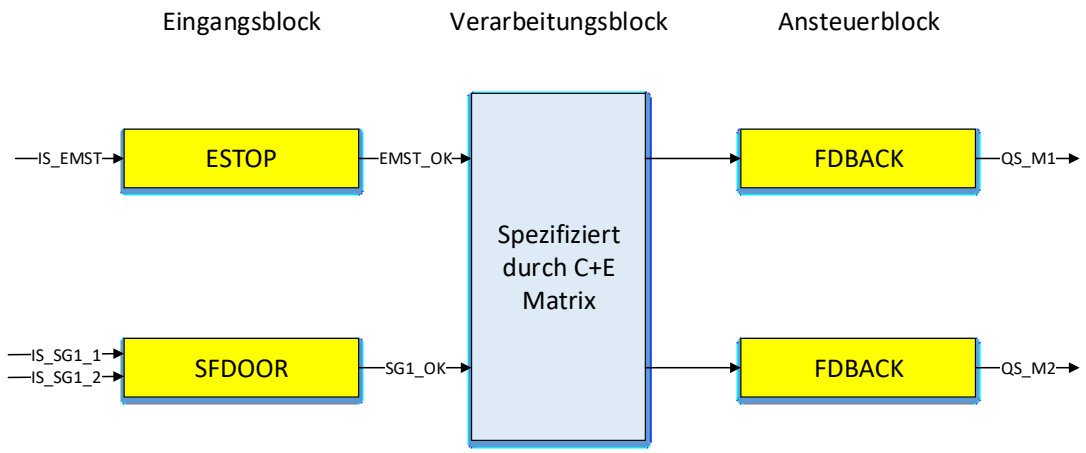


Abbildung 19 Dreiteilung der Software, Quelle: IFA Report 2/2016, Seite: 33 (leicht modifiziert)

Überdies ist es wichtig bei der Namensgebung von Variablen strukturiert vorzugehen, sodass das Programm übersichtlich ist und zusammengehörende Module immer erkenntlich sind.

Die Matrixmethode ist aus folgenden Schritten bzw. Dokumenten zusammengesetzt:

- Definition aller einzelnen Sicherheitsfunktionen
- Dokumentation der Adressen und Variablenbezeichnungen
- Treffen einer Auswahl an fehlervermeidenden Maßnahmen
- Festlegung der normativen Anforderungen
- Dokumentation der Architektur des sicherheitsgerichteten Programms
- Dokumentation der Modularchitektur
- Erstellen der C+E Matrix (Cause & Effect Matrix)
- Durchführen des Codereviews
- Durchführen der Softwarevalidierung [58]

Der Kern der Matrixmethode ist das Erstellen der C+E Matrix, diese zeigt anschaulich das Schaltverhalten aller Sicherheitsfunktionen. Die Matrix kann bereits während der Spezifikationsphase aufgestellt werden um bei der späteren Programmierung nur mehr die erstellte Ansteuerung in Softwarecode umsetzen zu müssen. Ergänzend dient diese Matrix mit dem eindeutigen Schaltverhalten der abschließenden Softwarevalidierung.

[58] Vgl. Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)(DGUV) (Hrsg.) (2016), Seite: 33-34

			Cause									Effect				D1		
			Involvierte Eingänge								Sicherheitsfunktionen	Ausgänge						
Betriebsart	Vorgängerzustand bei Test	Zustand	IS_EMST (I8.4)	IS_SG1_1 (I8.2)	IS_SG1_2 (I9.6)	IS_SG2_1 (I8.1)	IS_SG2_2 (I9.5)	IS_SG3_1 (I8.0)	IS_SG3_2 (I9.4)	IS_SL_SG2 (I8.5)		Q5_M1 (Q24.0)	Q5_M2 (Q24.1)	Q5_M3 (Q24.2)	Quittierung I_ACK (I4.0)	Geprüft (ok/n.ok)	Name	Datum
											C1: Software entspricht der Matrixdokumentation							
		1	1	1	1	1	1	1	1	1	ALL_OK	EIN	EIN	EIN				
	1	2	0	1	1	1	1	1	1	1	SF1: Not-Halt betätigt	EMST_OK AUS	EMST_OK AUS	EMST_OK AUS	EIN			
	1	3	1	0	0	1	1	1	1	1	SF2: SG1 offen	SG1_OK AUS	NOP	NOP	EIN			
	1	4	1	1	1	0	0	1	1	1	SF3: SG2 offen	NOP	SG2_OK AUS	NOP	EIN			
	1	5	1	1	1	0	0	0	0	1	SF4: SG2 und SG3 offen	SG2_OK v SG3_OK AUS	NOP	NOP	EIN			
	1	6	1	1	1	0	0	1	1	0	SF5: Sicherheitsleiste betätigt	NOP	NOP	IS_SL_SG2 AUS	EIN			
V1	Verifikation durchgeführt (ok/n. ok):													Softwaresignatur:				
	Datum:																	
	Name:																	

Abbildung 20 C+E Matrix, Quelle: IFA Report 2/2016, Seite: 42

In Abbildung 20 ist die C+E Matrix ersichtlich, auf der linken Seite unter der Bezeichnung „Cause“ werden alle involvierten Eingänge aufgezählt und es wird ihr auslösendes Ereignis dargestellt. Auf der rechten Seite unter der Bezeichnung „Effect“ sieht man die Auswirkung bei den Aktoren durch dieses Ereignis. Die gesamten Sicherheitsfunktionen werden je in einer Zeile unter der Bezeichnung „Sicherheitsfunktionen“ aufgelistet. Die oberste Zeile „ALL_OK“ beschreibt den Zustand, wenn die Maschine gestartet wird. Von diesem Zustand aus können die sicherheitsgerichteten Eingänge bei der dazugehörigen Sicherheitsfunktion geschaltet werden. Zusätzlich wird an der linken Seite noch die Betriebsart vermerkt, da nicht immer alle Sicherheitsfunktionen in jeder Betriebsart verwendet werden. Unter den Aktoren sind in blauer Schriftfarbe die Variablen zu sehen, welche relevant für diese Sicherheitsfunktion sind.[59]

Bildung der Softwarespezifikation:

Bei jeder Sicherheitsfunktion, die einen Schaltvorgang bei einem Aktor in Bezug auf den Vorgängerzustand (ALL_OK) auslöst, soll eine nachvollziehbare Verknüpfung der Eingangsgrößen der Ansteuerung in die Zelle eingetragen werden. Der jeweilige Schaltvorgang wird als „EIN“ bzw. „AUS“ angegeben. Wenn eine Sicherheitsfunktion keinen Schaltvorgang bei einem Aktor auslöst, wird dies als „NOP“ vermerkt.[60]

[59] Vgl. Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)(DGUV) (Hrsg.) (2016), Seite. 41-43

[60] Vgl. Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)(DGUV) (Hrsg.) (2016), Seite: 42

5.3 Das Softwaretool SOFTEMA

Damit die vorgestellte Matrixmethode des IFA in der Praxis anwenderfreundlich und auch qualitätssichernd eingesetzt werden kann, wurde das Softwaretool SOFTEMA (Software von Steuerungen an Maschinen) entwickelt. Die Software ist kostenlos und zurzeit ist die Beta Version 0.8.0 verfügbar, eine Vorstellung am Markt der Version 1 ist nach der Testphase angedacht. Durch die Anwendung von SOFTEMA sollen Eingabefehler verringert und aufgedeckt werden. Die Software basiert auf den entwickelten Excel Arbeitsblättern der IFA Matrixmethode. Es gibt ein Benutzerverwaltungskonzept, damit jeder Benutzer nur diese Funktionen zur Verfügung hat, welche er für seine tatsächliche Arbeit benötigt. Standardmäßig sind folgende Benutzer angelegt: Projektleiten, Projektieren, Inbetriebnehmen, Prüfen 1, Prüfen 2, Validieren und Administrieren.

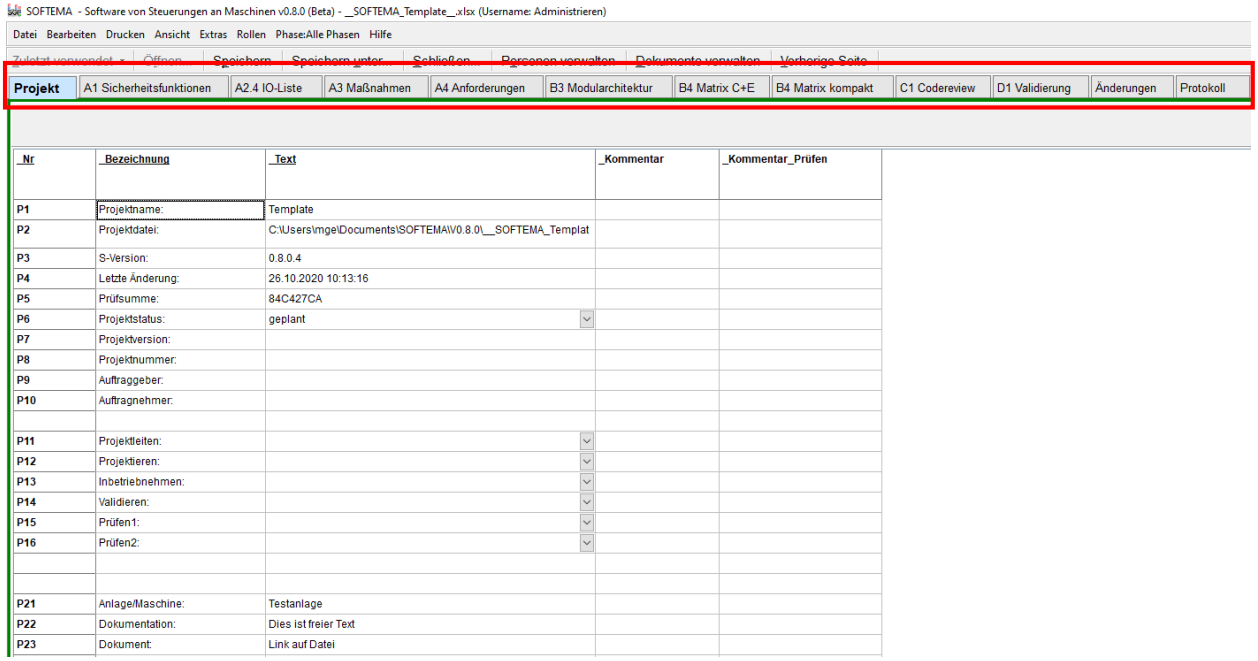


Abbildung 21 SOFTEMA Programmübersicht, Quelle: Eigene Darstellung

In Abbildung 21 ist die Programmoberfläche zu sehen, es gibt verschiedene Registerkarten z.B. A1 Sicherheitsfunktionen, in denen die jeweiligen Daten spezifiziert bzw. dokumentiert werden können.[61]

Der Aufbau der Registerkarten ist dem Vorgehen nach dem V-Modell der EN ISO 13849-1 (siehe Abschnitt 4.5) nachempfunden, d.h. es werden alle geforderten konstruierenden bzw. überprüfenden Tätigkeiten abgebildet.

[61] Vgl. Michael Huelke (2015), Seite: 76-79

Es wird bei jedem Öffnen des Projektes überprüft, ob die Excel Arbeitsmappe in der Zwischenzeit von jemandem geändert wurde und in Abbildung 22 ist das Pop Up Fenster ersichtlich, welches den Bediener darauf hinweist.

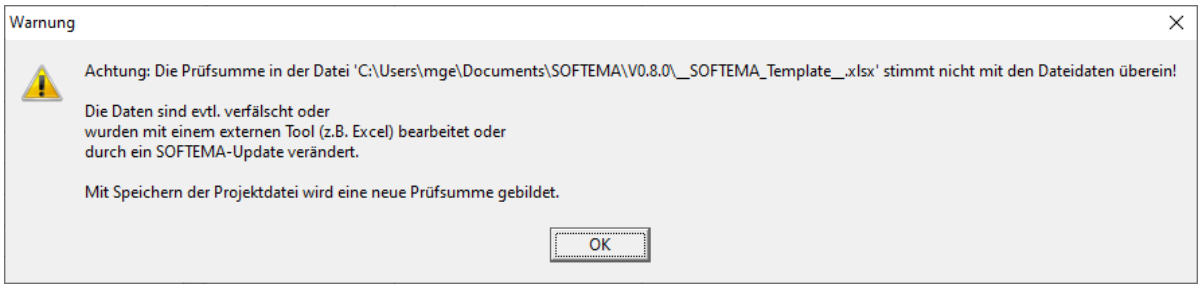


Abbildung 22 SOFTEMA Warnung Daten bearbeitet, Quelle: Eigene Darstellung

Beim Prozess der Softwareentwicklung gibt es viele verschiedene Verifizierungs- und Validierungsprozesse, welche von den verschiedenen Benutzern durchgeführt werden müssen. In Tabelle 5 ist ersichtlich in welchem Dokument welche Art der Prüfung erforderlich ist.

Art der Prüfung	**Dokument**	**Einträge**
Validierung	Sicherheitsfunktionen	
Validierung	IO-Liste	IO-Test
Validierung	Normanforderungen	
Validierung	C+E Matrix	
Validierung	C+E Matrix kompakt	
Validierung	Softwarevalidierung	
Verifikation	IO-Liste	Software - Verifikation
Verifikation	IO-Liste	Diagnose - Test
Verifikation	Fehlervermeidende Maßnahmen	
Verifikation	Modularchitektur	
Verifikation	C+E Matrix	
Verifikation	C+E Matrix kompakt	
Verifikation	Codereview	

Tabelle 5 SOFTEMA Validierung und Verifizierung, Quelle: Eigene Darstellung

In Abbildung 23 ist als Beispiel die Validierung einer Sicherheitsfunktion zu sehen. Wenn die Einträge mit den Vorgaben geprüft wurden, muss der prüfende Benutzer die jeweilige Zeile auf „OK“ ändern und am Ende der Einträge unten seinen Namen und das Datum zur Dokumentation auswählen.

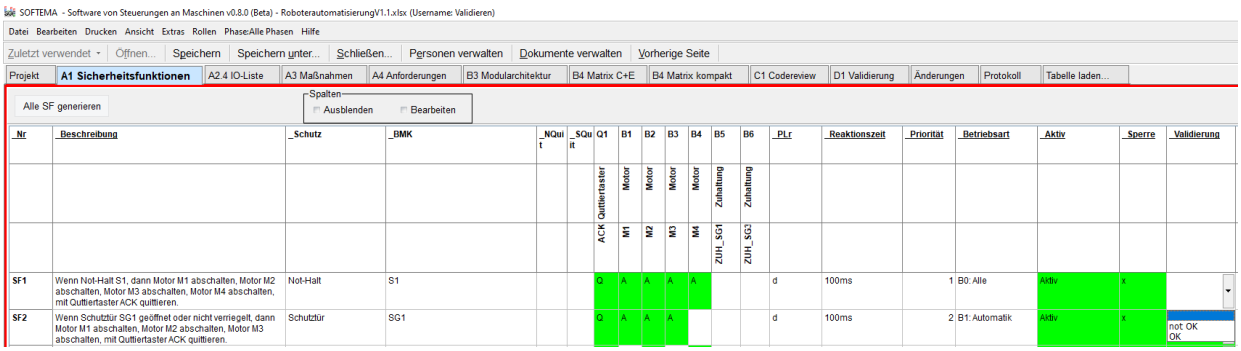


Abbildung 23 SOFTEMA Validierung, Quelle: Eigene Darstellung

6 PRAKTISCHE UMSETZUNG

Im folgenden Abschnitt wird das Softwaretool SOFTEMA mittels eines realen Beispiels auf seine Praxistauglichkeit getestet. Der Projektablauf bei der Umsetzung mit den dazugehörigen Kapiteln wird in Abbildung 24 gezeigt. Die folgenden Abschnitte werden jeden dieser Schritte beschreiben und Praxisempfehlungen geben.

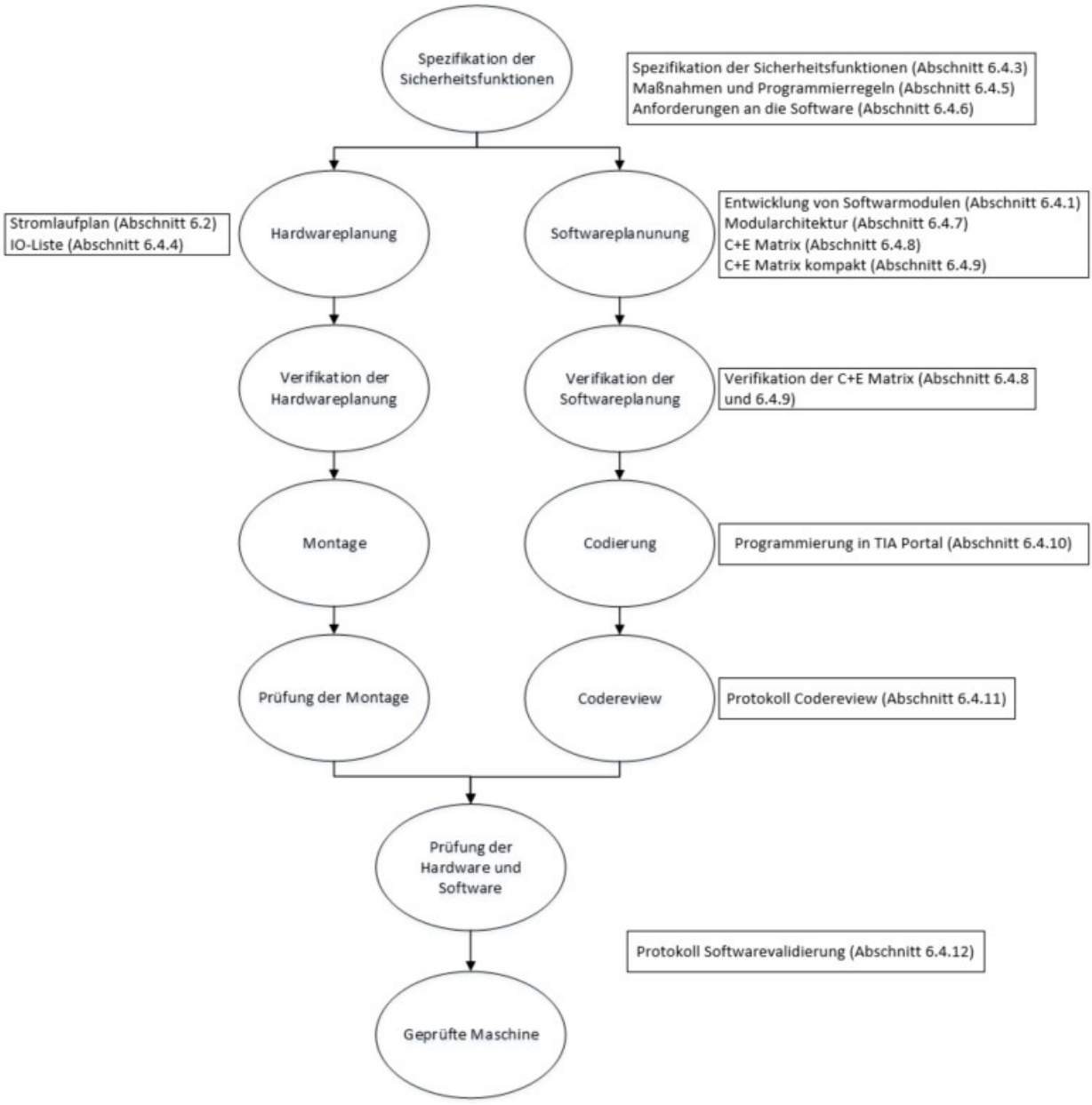


Abbildung 24 Typischer Projektablauf, Quelle: IFA Report 2/2016, Seite: 20 (leicht modifiziert)

6.1 Anlagenlayout

Als Beispielanlage wurde eine Roboterzelle, wie in Abbildung 25 ersichtlich, ausgewählt. Diese Anlage dient zur automatisierten Be- und Entladung eines CNC-Bearbeitungszentrums.

Die Funktion der Anlage ist wie folgt:

- Der Roboter (M1) belädt das CNC-Bearbeitungszentrum mit Bauteilen aus der Palette.
- Die Palette mit den Bauteilen wird über die Beladeklappe SG3 und das Hubtor SG2 in den Arbeitsraum des Roboters gebracht.
- Die Teile Zu– und Abführung haben zwei Ebenen (Palette oben und Palette unten). Die Paletten oben und unten haben jeweils einen Motor M3 & M4 als Antrieb verbaut.
- Das Hubtor SG2 trennt den Belade- vom Arbeitsraum.
- Die Schutztür SG1 wird als Wartungszugang für Instandhaltungstätigkeiten verwendet.
- Die Schutztür SG1 darf nur geöffnet werden, wenn die Motoren M1 und M3 stillstehen und die Betriebsart Leerfahren ausgewählt ist.
- Die Schutztür SG3 darf nur geöffnet werden, wenn die Motoren M2, M4 stillstehen und das Hubtor SG2 geschlossen ist.

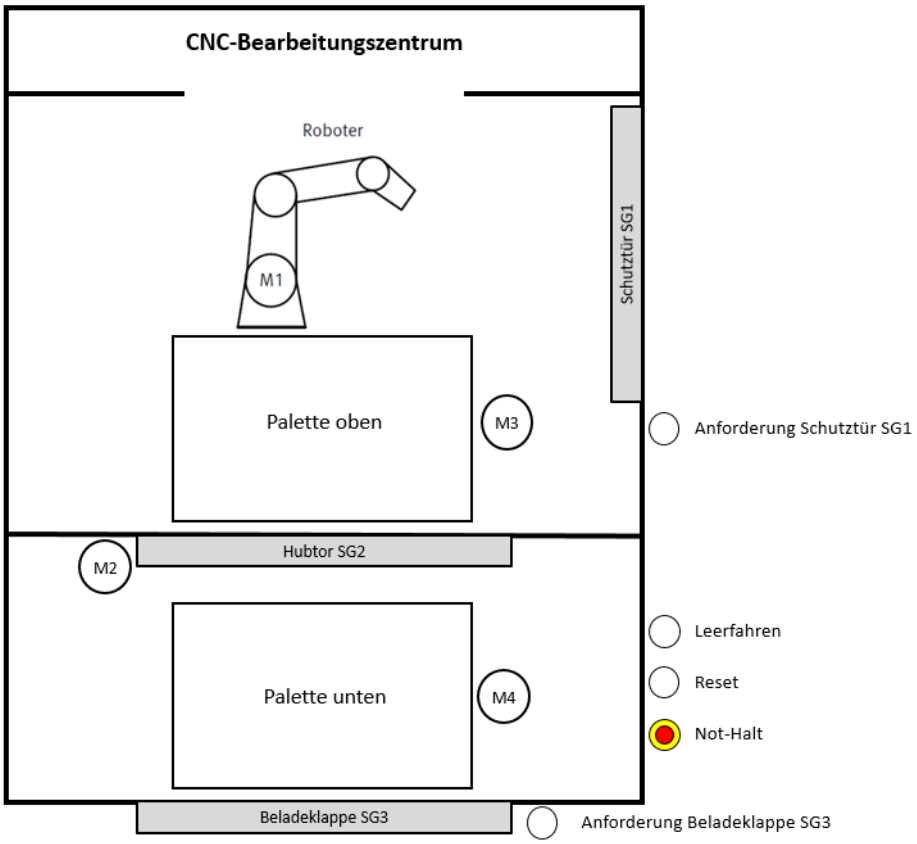


Abbildung 25 Roboterzelle, Quelle: Eigene Darstellung

6.2 Verwendete Hardwarekomponenten und Stromlaufplan

Zur Realisierung der Roboterzelle wurde eine Sicherheits – SPS mit dazugehöriger dezentraler Peripherie verwendet. In Tabelle 6 ist ein Auszug aus den wichtigsten verwendeten Komponenten zu sehen.

Stück	Komponente	Bezeichnung	Hersteller
1x	SPS	S7-155F-1 PN	Siemens
1x	Dezentrale Peripherie	IM 155-6 PN ST	Siemens
5x	Dezentrale Peripherie	F-DI 8x24VDC HF_1	Siemens
1x	Dezentrale Peripherie	F-DQ 8x24VDC/0.5A PP HF	Siemens
1x	Not-Halt	NOT-HALT-Pilzdrucktaste 3SU1150 r	Siemens
2x	Zuhaltung	Sicherheitszuhaltung AZM 170	Schmersal
6x	Positionsschalter	Positionsschalter PS215	Schmersal
8x	Schütz	Leistungsschütz 3RT2015	Siemens
4x	Hilfsschalter	Hilfsschalter 3RH2911	Siemens
1x	Roboter	M-10iA/10M	Fanuc

Tabelle 6 Auszug aus der Hardware Stückliste, Quelle: Eigene Darstellung

In Abbildung 26 ist der projektierte Stromlaufplan zu sehen, dieser wird in den folgenden Abschnitten in der Software umgesetzt.

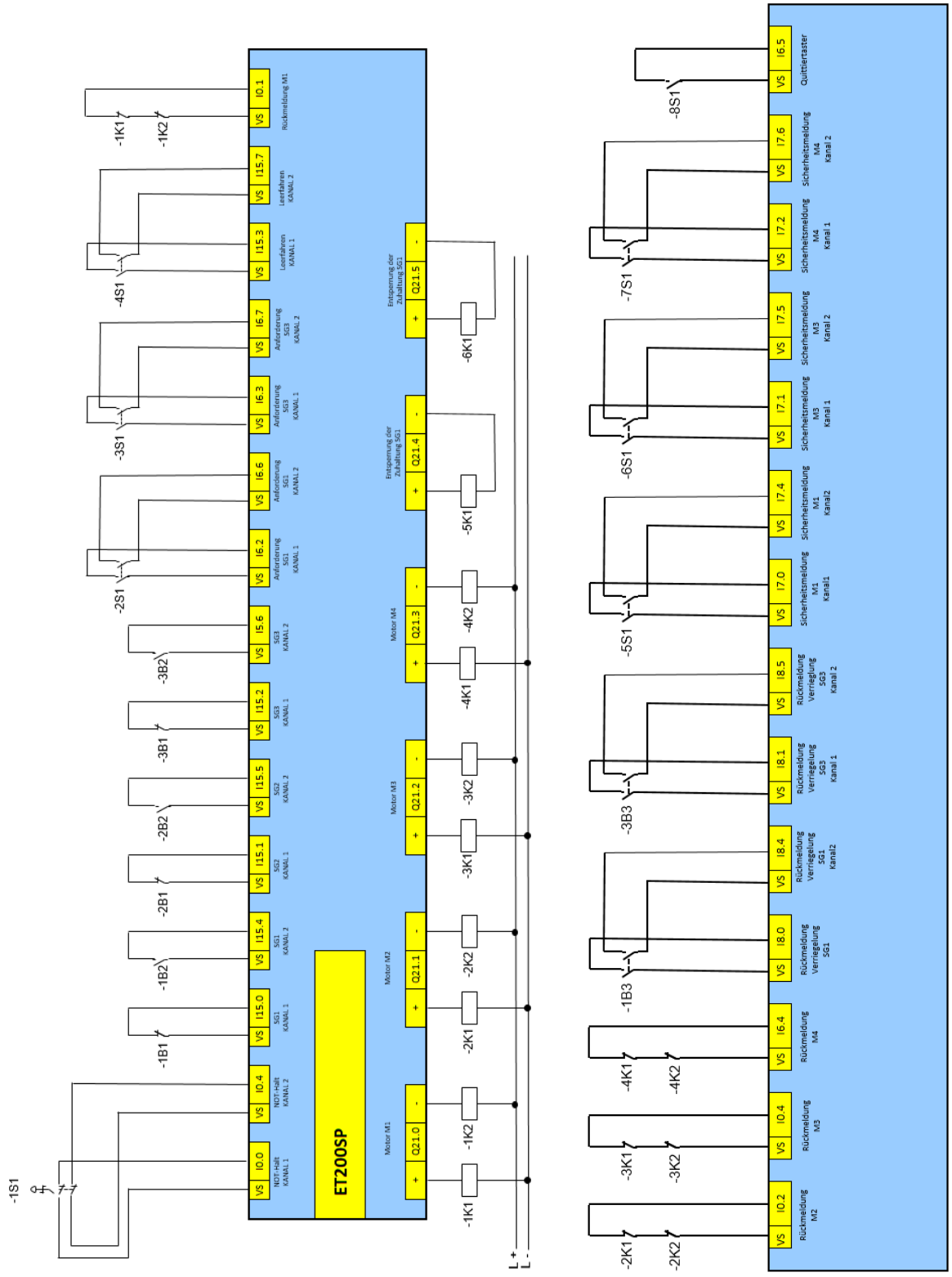


Abbildung 26 Stromlaufplan, Quelle: Eigene Darstellung

6.3 Sicherheitsfunktionen und geforderter Performance Level

Zu der in Abschnitt 6.1 vorgestellten Maschine wurde eine Risikobeurteilung nach EN ISO 12100 durchgeführt. Die Risikobeurteilung ergab, dass bestimmte Gefährdungen nicht durch „inhärent sichere Konstruktion“ beseitigt werden können, deswegen sind ergänzende Schutzmaßnahmen in Form von funktionaler Sicherheit erforderlich.

Folgende Sicherheitsfunktionen sind zu realisieren:

Nummer	Beschreibung	PL_r	Reaktionszeit	Priorität
SF1	Wenn Not-Halt S1, dann Motor M1 abschalten, Motor M2 abschalten, Motor M3 abschalten, Motor M4 abschalten, mit Quittiertaster ACK quittieren.	d	100ms	1
SF2	Wenn Schutztür SG1 geöffnet oder nicht verriegelt, dann Motor M1 abschalten, Motor M2 abschalten, Motor M3 abschalten, mit Quittiertaster ACK quittieren.	d	100ms	2
SF3	Wenn Schutztür SG3 geöffnet oder nicht verriegelt, dann Motor M2 abschalten, Motor M4 abschalten, mit Quittiertaster ACK quittieren.	d	100ms	2
SF4	Wenn Anforderung SG1 & IS_LEER & Stillstand & T#5s, dann Zuhaltung ZUH_SG1 abschalten, mit Quittiertaster ACK quittieren.	d	100ms	2
SF5	Wenn Anforderung SG3 & IS_LEER & Stillstand & T#5s, dann Zuhaltung ZUH_SG3 abschalten, mit Quittiertaster ACK quittieren.	d	100ms	2
SF6	Wenn Anforderung Leerfahren, dann Motor M1 abschalten, Motor M2 abschalten, Motor M3 abschalten, Motor M4 abschalten (ohne Quittierung).	d	100ms	2
SF7	Wenn Schutztür SG2&SG3 geöffnet, dann Motor M1 abschalten, Motor M3 abschalten, Motor M4 abschalten, mit Quittiertaster ACK quittieren.	d	100ms	2

Tabelle 7 Auflistung der Sicherheitsfunktionen, Quelle: Eigene Darstellung

Die Ermittlung des PL_r wurde anhand des in Abschnitt 3.2.2 vorgestellten Verfahrens mit Hilfe der Software SISTEMA vorgestellt.

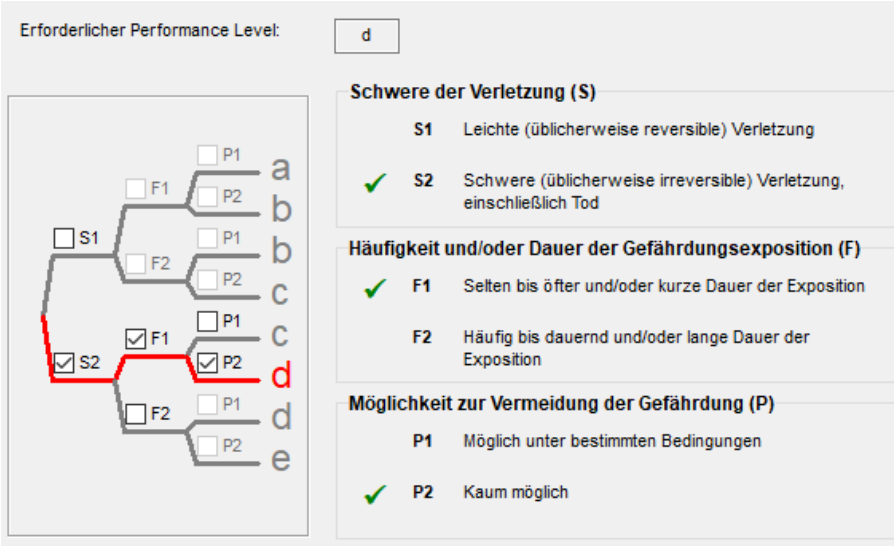


Abbildung 27 Ermittlung des PL_r, Quelle: Screenshot SISTEMA

In Abbildung 27 ist eine mögliche Vorgehensweise zu sehen, wie der erforderliche Performance Level direkt in SISTEMA bestimmt werden kann, in diesem Beispiel ist der erforderliche Performance Level PL_r = d. Die Bestimmung des PL_r sollte grundsätzlich immer im Rahmen der Risikobeurteilung bestimmt werden, dies wird im Abschnitt 3.2.2 genau erläutert. Dies deckt sich ebenso mit der C-Norm für Industrieroboter EN ISO 10218-2, in denen immer ein PL_r von d bzw. ein Aufbau nach Kategorie 3 gefordert wird.

Die aufgebaute Verschaltung der Kategorie 3 wurde in SISTEMA eingepflegt und auf seine Wirksamkeit überprüft:

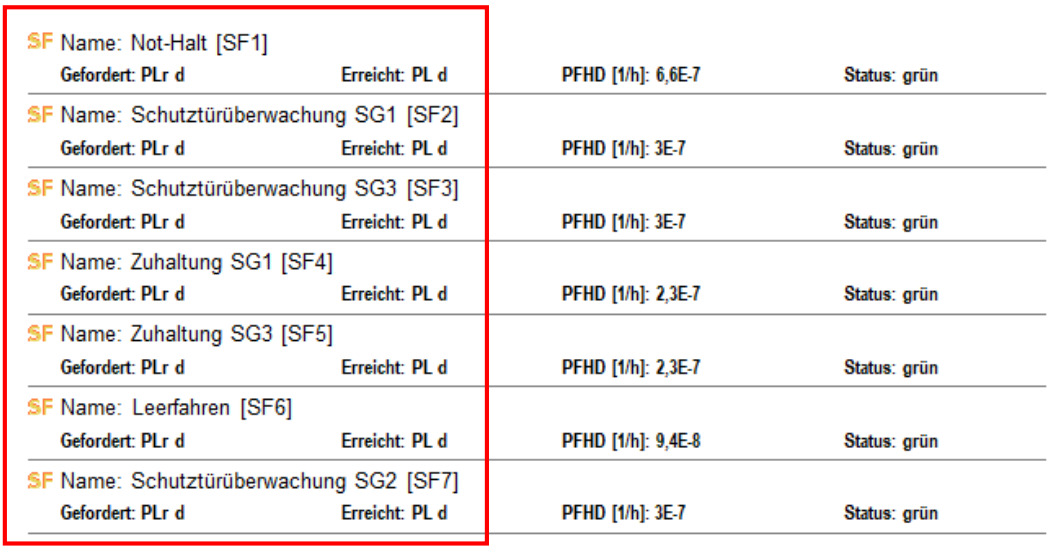


Abbildung 28 SISTEMA Bericht, Quelle: Eigene Darstellung

In Abbildung 28 ist das Ergebnis der SISTEMA Beurteilung zu sehen, alle Sicherheitsfunktionen erreichen den geforderte PL_r.

6.4 Entwicklung der Software

Im folgenden Abschnitt wird das Entwickeln von Softwaremodulen vorgestellt, das Softwaretool SOFTEMA mit Hilfe des Praxisbeispiels getestet und die Software in TIA (Totally Integrated Automation) Portal V16 Update 1 programmiert.

6.4.1 Entwicklung von eigenen Softwaremodulen

Da es in der Standard - Bibliothek von TIA Portal V16 nur wenige zertifizierte Bausteine gibt, mussten im Rahmen dieser Arbeit eigene Funktionsbausteine programmiert bzw. verifiziert und validiert werden. Von TIA Portal werden standardmäßig folgende Funktionsbausteine zur Verfügung gestellt:

- ESTOP1
  Dient zur Realisierung von NOT-Halt Abschaltung mit Quittierung der Kategorie 0 und 1.
- TWO_HAND
  Dient zur Realisierung einer Zweihandüberwachung.
- MUT_P
  Dient zur Realisierung von parallelem Muting von zwei oder vier Mutingsensoren.
- EV1oo2DI
  Dient zur Realisierung einer „Eins aus Zwei Auswertung“ von zwei einkanaligen Gebern mit einer Diskrepanzüberwachung.
- FDBACK
  Dient zur Realisierung einer Ansteuerung eines Aktors z.B. Schütz mit Rückführkreisüberwachung.
- SFDOOR
  Dient zur Realisierung einer zweikanaligen Schutztürüberwachung mit Quittierung.
- ACK_GL
  Dient zur Depassivierung bzw. Wiedereingliederung der F-Peripherie nach Kanal- oder Kommunikationsfehlern.[62]

Zur Realisierung der geforderten Sicherheitsfunktionen, welche im Abschnitt 6.3 vorgestellt wurden, sind drei verschiedene Softwaremodule zu entwickeln.

Folgende Softwaremodule werden benötigt:

- **Softwaremodul Lock**
  Dieses Modul ist dafür zuständig, dass die Verriegelung der Schutztür überwacht wird. Sobald diese nicht mehr verriegelt ist, muss ein Ausgang auf *False* gesetzt werden. Wenn die Schutztür wieder verriegelt ist und der Fehler quittiert ist, dann wird der Ausgang wieder auf *True* gesetzt.
- **Softwaremodul OP_Door**
  Dieses Modul ist dafür zuständig, dass eine Schutztür sicher nach einer Verzögerungszeit entsperrt wird. Voraussetzungen für das Öffnen der Schutztür sind, dass die Betriebsart Leerfahren

[62] Vgl. SIEMENS AG (Hrsg.) (2019), Seite: 461-522

ausgewählt ist, die Schutztür geschlossen ist und nach der Anforderung die Verzögerungszeit abgelaufen ist. Der Ausgang wird bei Freigabe auf *True* geschalten.

- **Softwaremodul Leerfahren**
  Dieses Modul ist dafür zuständig, dass die Betriebsart Automatik überwacht wird. Wenn die Betriebsart Leerfahren ausgewählt wurde, wird der Ausgang auf *False* gesetzt. Ist der Automatikbetrieb wieder ausgewählt, so wird nach erfolgter Quittierung der Ausgang wieder auf *True* gesetzt.

Um diese auch entsprechend den Forderungen der EN ISO 13849-1 zu entwerfen bzw. zu prüfen, wurde im Abschnitt 5.1 das V-Modell der IFA für selbstentwickelte Funktionsbausteine vorgestellt. Es wurde ähnlich wie zur Realisierung von Sicherheitsfunktionen eine Matrixmethode entwickelt, um auch diese einsetzen zu können. Die Matrixmethode ist hier gleich wie bei zertifizierten Hersteller - Softwaremodulen auf Basis von Excel aufgebaut. Jedoch ist es nicht möglich diese Excel Arbeitsmappe mit Hilfe des Softwaretools SOFTEMA zu verwenden. Die Anwendung dieser Methode für selbstentwickelte Softwaremodule wird hier anhand des Softwaremoduls OP_Door gezeigt.

6.4.1.1 Schnittstellendefinition

Der erste Schritt, wenn ein eigenes Softwaremodul entwickelt werden sollte, ist alle Schnittstellen zur Ansteuerung dieses Moduls zu definieren. Dafür wird in der zur Verfügung gestellten Excel Mappe die Tabelle AM1 Schnittstellendefinition verwendet.

Dokument: **AM1 - Schnittstellendefinition und Funktionsbeschreibung**

Funktionsbeschreibung		
Der Funktionsblock OP_Door dient dem sicheren Entsperren einer Schutztür nach einer Verzögerungszeit. Die Freigabe erfolgt mit einer logischen 1. Die Freigabe wird erteilt, wenn Leerfahren angewählt ist (BAWS=1), die Schutztür geschlossen ist (SG=1) und nach der Anforderung zum Öffnen die Entriegelungszeit abgelaufen ist.		
Eingänge		
Variable	**Datentyp**	**Bemerkung**
BAWS	Safebool	Betriebsartenwahlschalter
OP_SG	Safebool	Anforderung Schutztür öffnen
SG	Safebool	Kontakt Schutztür geschlossen
Ueb_Zeit	S5TIME	Entriegelungszeit
Ausgänge		
Unlock_SG	Safebool	Freigabe Schutztür öffnen (Initialwert FALSE)
Interne Variablen		
IM	Safebool	Zwischenvariable zur Ansteuerung der Verzögerung

Abbildung 29 Matrixmethode Schnittstellendefinition, Quelle: Eigene Darstellung

In Abbildung 29 wird das Dokument mit den erforderlichen Einträgen zu den Eingängen und Ausgängen gezeigt. Wichtig ist es hier, dass die Funktionsbeschreibung sehr detailliert gemacht wird um keinen Platz für mögliche Interpretationen zu geben.

6.4.1.2 Maßnahmen und Programmierregeln

Als Nächstes ist es wichtig die fehlervermeidenden Maßnahmen und Programmierregeln zu definieren. Die Maßnahmen und Regeln decken sich hier großteils mit den geforderten Maßnahmen für das Entwickeln von Sicherheitsfunktionen aus Funktionsbaustein-Bibliotheken, welche in Abschnitt 6.3 gezeigt werden.

Dokument: AM2 - Katalog fehlervermeidender Maßnahmen, Tools und Programmierregeln

allgemeine Maßnahmen

	CM1	
	Kürzel	ok/n.ok
Namenssystematik für Variablen		
Die Variablennamen sollen möglichst selbsterklärend sein.	R1	ok
Kommentare:		
Jedes Netzwerk wird mit einer Überschrift und einem Kommentar versehen.	R2	ok
Jede Variable wird im Deklarationsteil mit einem Kommentar versehen.	R3	ok
Entwicklung eigener Funktionsbausteine:		
Eigene wiederverwendbare Funktionsbausteine werden separat nach V-Modell entwickelt und dokumentiert.	R4	ok
Der Test des Bausteins geschieht per Simulation.	R5	ok
Für eigene Funktionsbausteine muss ein Bibliotheksmagagement gepflegt werden.	R6	ok
Es werden keine globalen Variablen verwendet. Der Baustein kann nur über die Schnittstellen kommunizieren.	R7	ok
Aktivitäten nach Änderungen:		
Alle Änderungen müssen in der Änderungshistorie dokumentiert werden.	R8	ok
Nach Änderungen muss die Validierung der geänderten Software wiederholt werden.	R9	ok

Herstellerspezifische Maßnahmen

		CM1	
		Kürzel	ok/n.ok
Programmeditor / Programmiersprache			
Genutzter Programmeditor	Siemens Step 7 Safety V16	R14	ok
Programmiersprache	Funktionsbausteinsprache (FUP)	R15	ok
Softwarebibliothek	Distributed Safety (V1)	R16	ok
		Datum:	15.10.2020
		Prüfer:	Programmierer 2

Abbildung 30 Matrixmethode Maßnahmen und Programmierregeln, Quelle: Eigene Darstellung

In Abbildung 30 ist ein Ausschnitt aus den geforderten Maßnahmen zu sehen. Maßnahmen sind unter anderem, dass Variablen möglichst selbsterklärende Namen haben sollten oder dass im Kommentar einer Variable immer kurz eine Beschreibung der Funktion gegeben wird. Wichtig ist auch, dass definiert wird, in welchem Programmeditor bzw. mit welcher Programmiersprache die Software umgesetzt wird.

6.4.1.3 Modulspezifikation und Testplan

Der Kern dieser Methode ist das Aufstellen der sogenannten C+E Matrix. Diese ist ähnlich der in Abschnitt 5.2 vorgestellten Methode aufgebaut. Mithilfe dieser soll der Aufbau der Software definiert werden, sodass dieser Aufbau nur mehr im Programmiertool umzusetzen ist. Des Weiteren soll mit Hilfe dieser Tabelle nach erfolgter Programmierung die Funktionalität überprüft werden.

Dokument: BM1 - Modulspezifikation und Testplan

Funktionsbeschreibung

Der Zwischenwert IM wird als **speichernder** Wert ausgeführt.
IM: Gesetzt wird das RS-Flip-Flop vom Betriebsartenwahlschalter und der Anforderung Schutztür öffnen. Zurückgesetzt wird es, sobald der Schutztürkontakt nicht mehr aktiv ist.
Unlock_SG: Sobald IM aktiv ist, läuft die Entsperrzeit. Nach Ablauf dieser Zeit wird der Ausgang aktiv und bleibt 1, solange IM aktiv

			Cause					Effect				
			Zustände der Eingänge,					Interne Variable	Ausgang	DM1		
Verhalten Flip-Flop	Vorgängerzustand bei	Zustand	BAWS	OP_SG	SG	IM	Bezeichnung	IM	Unlock_SG	Geprüft	Name	Datum
							CM1: Software entspricht der Matrixdokumentation					
		1	1	0	1	0	Schutztür geschlossen, Leerfahren angewählt, keine Anforderung zum Öffnen	AUS	AUS			
Setzen	1	2	1	1	1	1	Schutztür geschlossen, Leerfahren angewählt, Anforderung zum Öffnen	BAWS & OP_SG EIN	AUS			
Rücksetzen	2	3	1	0	0	1	Schutztür wird geöffnet, keine Anforderung zum Öffnen	/SG AUS	AUS			
	2	4	1	0	1	1	Die Entsperrzeit ist abgelaufen nach Aktivierung von IM	EIN	IM EIN			
VM1	Verifikation durchgeführt (ok/n.ok): ok								Softwaresignatur:			
	Datum:	15.10.2020										
	Prüfer:	Programmierer 2										

Hinweise: Farben rot=Variable besitzt den Zustand FALSE
blau=Variable besitzt den Zustand TRUE

Zeichen / =Zustand FALSE (entspricht roter Farbe)
v = oder
& = und
> = positive Flanke

Die gelbe Spalte markiert den Grundzustand vor der Aktivierung der Ausgänge.
Der Test der Abschaltung der Signale erfolgt immer vom Zustand EIN aus.

Abbildung 31 Matrixmethode Modulspezifikation und Testplan, Quelle: Eigene Darstellung

In Abbildung 31 ist in der C+E Matrix das Schaltverhalten der Eingänge zu den Ausgängen zu sehen. Wenn also z.B. der Betriebsartenwahlschalter auf Leerfahren geschaltet wird und der Taster für die Anforderung zum Öffnen gedrückt ist, muss die interne Variable IM auf *True* geschalten sein und die Variable muss so lange gedrückt werden, bis die Türe aufgemacht wird. Nachdem IM auf den Wert *True* geschaltet wird, soll es nach einer einstellbaren Verzögerungszeit möglich sein, die Türe Unlock_SG zu öffnen.

Die spezifizierten Anforderungen müssen nun im nächsten Schritt in der Zeile VM1 verifiziert werden. Wenn diese abgeschlossen ist, kann mit dem eigentlichen Umsetzen der Spezifikation im Programmiertool begonnen werden.

6.4.1.4 Programmierung des Softwaremoduls

Die Umsetzung des Softwaremoduls wurde im Programmiertool TIA Portal V16 realisiert. Gemäß den Spezifikationen der Abschnitte 6.4.1.1 bis 6.4.1.3 wurde dies umgesetzt.

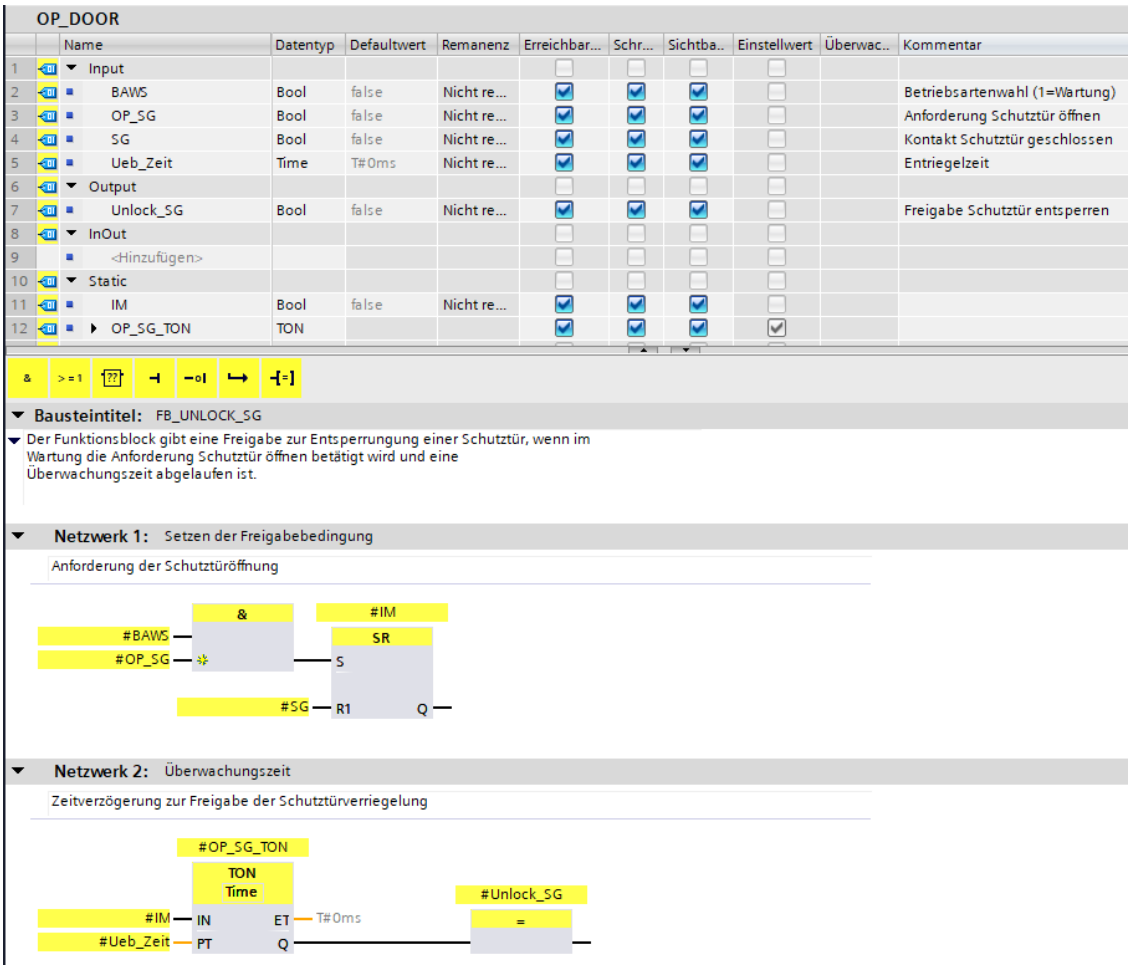


Abbildung 32 TIA Portal FB OP_DOOR, Quelle: Eigene Darstellung

In Abbildung 32 ist der Funktionsbaustein in TIA Portal zu sehen. Am oberen Ende der Grafik sieht man die Eingänge und die Ausgänge bzw. die interne Variable lt. Spezifikation von Abschnitt 6.4.1.1. Damit der Zustand *True* gespeichert werden kann, wenn die Betriebsart Leerfahren ausgewählt und der Taster für die Anforderung der Schutztür gedrückt ist, muss ein SR-Flipflop verwendet werden. Das Flipflop wird, wenn die Türe geöffnet wurde (SG=*False*), wieder zurückgesetzt. In den Spezifikationen wurde gefordert, dass die Türe erst nach einer einstellbaren Verzögerungszeit Ueb_Zeit geöffnet werden kann. Die Verzögerung wurde mit einer Einschaltverzögerung realisiert.

6.4.1.5 Simulieren des Softwaremoduls

Nachdem das Softwaremodul im Programmiertool umgesetzt ist, muss dieses auch auf seine Funktionsfähigkeit gegenüber den Spezifikationen getestet werden. Um die Software zu testen, wurde der Baustein mit der Software S7-PLCSIM V16 in Verbindung mit TIA Portal V16 getestet. Es wurden Testfälle generiert, mit denen alle Zustände abgebildet werden können. Hier werden plakativ zwei verschiedene Testfälle dargestellt:

- **Testfall 1**

			Zustände der Eingänge,					Interne Variable	Ausgang	DM1		
Verhalten Flip-Flop	Vorgängerzustand bei	Zustand	BAWS	OP_SG	SG	IM	Bezeichnung	IM	Unlock_SG	Geprüft	Name	Datum
Setzen	1	2	1	1	1	1	Schutztür geschlossen, Leerfahren angewählt, Anforderung zum Öffnen	BAWS & OP_SG EIN	AUS	ok	Programmierer 2	15.10.2020

Abbildung 33 Softwaremodul Testfall 1, Quelle: Eigene Darstellung

In Abbildung 33 sind die Anforderungen lt. Matrixmethode an die Software zu sehen. Wenn alle Eingänge den Zustand *True* haben, muss die interne Variable des SR Flipflops *True* sein. Wenn die Entsperrzeit noch nicht abgelaufen ist, muss der Ausgang des Moduls Unlock_SG *False* sein.

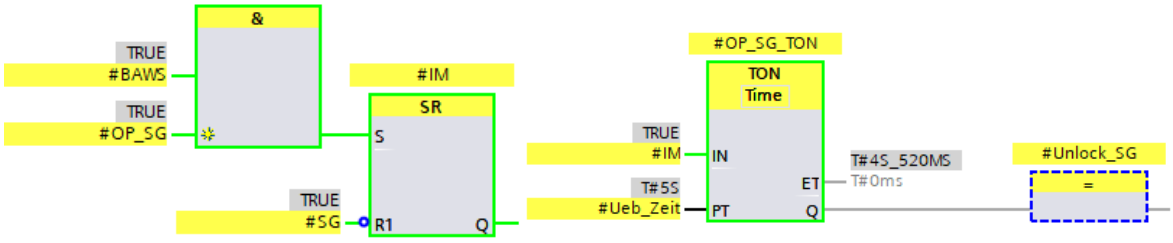


Abbildung 34 Softwaremodul Testfall 1 TIA Portal, Quelle: Eigene Darstellung

In Abbildung 34 ist die Simulation in TIA Portal bzw. PLCSIM ersichtlich. Alle Eingänge haben den Zustand *True* und die interne Variable IM hat den Zustand *True*. Die Verzögerungszeit ist 4,52s d.h. die Verzögerungszeit von 5s wurde noch nicht erreicht, deswegen hat der Ausgang Unlock_SG den Zustand *False*.

Da die Anforderungen erfüllt wurden, muss dies in der Matrix in der Spalte DM1 als OK mit Namen und Datum vermerkt werden.

- **Testfall 2**

			Zustände der Eingänge,					Interne Variable	Ausgang	DM1		
Verhalten Flip-Flop	Vorgängerzustand bei	Zustand	BAWS	OP_SG	SG	IM	Bezeichnung	IM	Unlock_SG	Geprüft	Name	Datum
	2	4	1	0	1	1	Die Entsperrzeit ist abgelaufen nach Aktivierung von IM	EIN	IM EIN			
VM1	Verifikation durchgeführt (ok/n.ok): ok								Softwaresignatur:			
	Datum:						15.10.2020					
	Prüfer:						Programmierer 2					

Abbildung 35 Softwaremodul Testfall 2, Quelle: Eigene Darstellung

In Abbildung 35 sind die Anforderungen lt. Matrixmethode an die Software zu sehen. Wenn alle Eingänge außer der Anforderung der Schutztür den Zustand *True* haben, muss die interne Variable des SR Flipflops *True* sein. Wenn die Entsperrzeit abgelaufen ist, muss der Ausgang des Moduls Unlock_SG *True* sein.

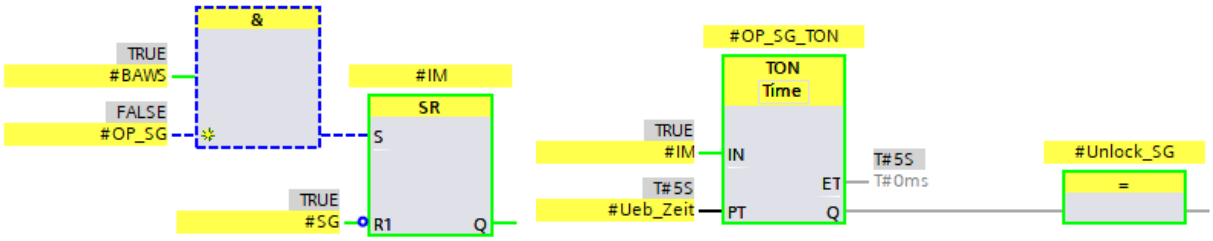


Abbildung 36 Softwaremodul Testfall 2 TIA Portal, Quelle: Eigene Darstellung

In Abbildung 36 ist die Simulation in TIA Portal bzw. PLCSIM ersichtlich. Alle Eingänge außer der Anforderung der Schutztür haben den Zustand *True* und die interne Variable IM hat den Zustand *True*. Die Verzögerungszeit von 5s ist abgelaufen, deswegen hat der Ausgang Unlock_SG den Zustand *True*.

Da die Anforderungen erfüllt wurden, muss dies in der Matrix in der Spalte DM1 als OK mit Namen und Datum vermerkt werden.

6.4.1.6 Programmskizze

Damit die Dokumentation in der Excel Mappe übersichtlich ist, gibt es die Möglichkeit, direkt in der Mappe eine Übersicht der Software hinzuzufügen.

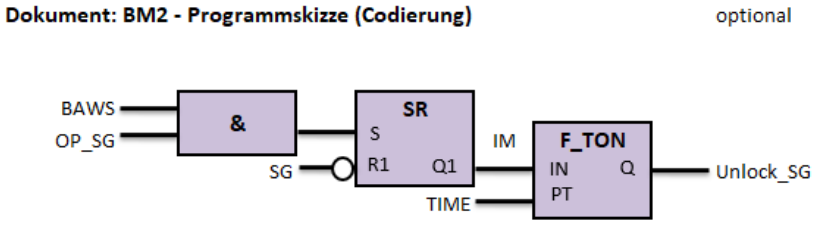


Abbildung 37 Matrixmethode Programmskizze, Quelle: Eigene Darstellung

In Abbildung 37 ist der Aufbau des Programms zu sehen, die Eingänge Betriebsart und Anforderung der Schutztür müssen zugleich geschaltet werden, damit das Flipflop geschaltet wird. Die Rücksetzung des Flipflops wird über den Kontakt der Türe realisiert, d.h. wenn die Türe aufgemacht wird, wird das Flipflop zurückgesetzt. Der Ausgang des Softwaremoduls wird nach Ablauf einer einstellbaren Zeit geschaltet.

6.4.1.7 Codereview

Nachdem alle anderen Schritte abgeschlossen sind, muss noch der sogenannte Codereview durchgeführt werden. Dieser Codereview umfasst, wie in Abbildung 38 zu sehen ist, z.B. ob die Maßnahmen und Programmierregeln eingehalten wurden, d.h. es muss nach dem Vier-Augen-Prinzip von einem zweiten Programmierer die Software gegenüber den Anforderungen in der Matrix geprüft werden.

Dokument: **CM1 - Protokoll Codereview**

Prüfdetail	Ref.	ok/n.ok
1. Sind die vereinbarten Programmierregeln eingehalten worden?	AM2	ok
2. Sind die vereinbarten Tools benutzt worden?	AM2	ok
3. Ist die Spezifikation der Software aus der Matrix umgesetzt?	BM1	ok
4. Ist die Umsetzung der gesamten Funktionalität plausibel?	AM1-BM1	ok
Datum:	15.10.2020	
Prüfer:	Programmierer 2	
Softwaresignatur:	E8003E50	

Abbildung 38 Matrixmethode Codereview, Quelle: Eigene Darstellung

Als letzter Punkt muss noch die Softwaresignatur vermerkt werden, um eine Rückverfolgung zu haben, ob ein ausgeliefertes Softwaremodul nachträglich verändert wurde.

Das getestete Modul kann nun in die Bibliothek von SOFTEMA übernommen werden (siehe Abschnitt 6.4.7) und kann somit für die Umsetzung einer Sicherheitsfunktion eingesetzt und immer wieder verwendet werden.

6.4.2 Projektdaten editieren

Vom IFA wird ein Excel Template zur Verfügung gestellt, welches die erforderliche Formatierung besitzt, um es mit SOFTEMA zu bearbeiten. Dies sollte für jedes neue Projekt am Anfang verwendet und für die eigenen Bedürfnisse angepasst werden. Zu Beginn sind in der Projektansicht die wichtigsten Daten, wie in Abbildung 39 zu sehen, einzugeben. Dazu gehört z.B. Auftraggeber, Seriennummer der Anlage, Projektteam usw. In dieser Ansicht wird auch die sogenannte Prüfsumme angezeigt, welche sich nach jeder Speicherung neu generiert, d.h. mit diesem Code kann überprüft werden, ob die Software-dokumentation geändert wurde.

Nr	Bezeichnung	Text
P1	Projektname:	Roboterautomatisierung
P2	Projektdatei:	C:\Users\mge\Documents\SOFTEMA\V0.8.0\Roboterautomatisieru
P3	S-Version:	0.8.0.4
P4	Letzte Änderung:	25.10.2020 13:40:51
P5	Prüfsumme:	45E2437C
P6	Projektstatus:	geprüft
P7	Projektversion:	V1.0
P8	Projektnummer:	2020-01
P9	Auftraggeber:	Glock GesmbH
P10	Auftragnehmer:	Glock GesmbH
P11	Projektleiten:	Projektleiter
P12	Projektieren:	Programmierer
P13	Inbetriebnehmen:	Inbetriebnehmer
P14	Validieren:	Programmierer 2
P15	Prüfen1:	Prüfer
P16	Prüfen2:	Prüfer 2
P21	Anlage/Maschine:	Roboterautomatisierung
P22	Dokumentation:	
P23	Dokument:	Link auf Datei
€€€€		

Abbildung 39 SOFTEMA Projektdaten, Quelle: Eigene Darstellung

6.4.3 Sicherheitsfunktionen

Der nächste Schritt ist das Auflisten der benötigten Sicherheitsfunktionen. Die Ansicht soll dabei helfen alle Sicherheitsfunktionen transparent darzustellen und außerdem bereits wichtige Informationen für die spätere Programmierung zu erstellen.

Nr	_SFK	_Beschreibung	_Schutz	_BMK	_NQuit	_SQuit	Q1	B1	B2	B3	B4	B5	B6	_PLr	_Reaktionszeit	_Priorität	_Betriebsart	_Aktiv	_Sperre	_Validierung
							Quittiertaster	Motor	Motor	Motor	Motor	Zuhaltung	Zuhaltung							
							ACK	M1	M2	M3	M4	ZUH_SG1	ZUH_SG3							
SF1	NOT-Halt	Wenn Not-Halt S1, dann Motor M1 abschalten, Motor M2 abschalten, Motor M3 abschalten, Motor M4 abschalten, mit Quttiertaster ACK quittieren.	Not-Halt	S1			Q	A	A	A	A			d	100ms	1	B0: Alle	Aktiv	o	
SF2	SG1	Wenn Schutztür SG1 geöffnet oder nicht verriegelt, dann Motor M1 abschalten, Motor M2 abschalten, Motor M3 abschalten, mit Quttiertaster ACK quittieren.	Schutztür	SG1			Q	A	A	A				d	100ms	2	B1: Automatik	Aktiv	x	

Abbildung 40 SOFTEMA Auflistung der Sicherheitsfunktionen, Quelle: Eigene Darstellung

In Abbildung 40 ist ein Ausschnitt aller Sicherheitsfunktionen zu sehen. Alle Sicherheitsfunktionen sind untereinander aufgelistet und jede hat ihre spezifische Nummer SF1 - S7 und die Betriebsmittelkennzeichnung bzw. die Verknüpfung der Betriebsmittel. Es ist anzugeben, welcher Ausgang von der Sicherheitsfunktion beeinflusst wird und ob eine manuelle Quittierung der Sicherheitsfunktion nach dem Auslösen erforderlich ist. Weitere wichtige Angaben sind der PL_r, die Reaktionszeit, die Priorität und in welchen Betriebsarten diese Sicherheitsfunktion wirkt. Es gibt die Möglichkeit, die Eingaben mit der Funktion „Formale Checks“ zu prüfen, damit können die Eingaben auf Vollständigkeit, Zulässigkeit der Quittierung usw. überprüft werden. Es ist nicht nötig, die Beschreibung der Sicherheitsfunktion auszufüllen, denn wenn der Button „Alle SF generieren“ gedrückt wird, wird diese automatisch generiert. Nachdem alle Sicherheitsfunktionen und Angaben eingegeben wurden, kann die jeweilige Zeile gesperrt werden, sodass diese nachträglich nicht mehr geändert werden kann.

Beim Vorgang der Softwarevalidierung (siehe Abschnitt 6.4.12) müssen die Angaben zu den Sicherheitsfunktionen mit den Vorgaben der Sicherheitsfunktionen der Maschine übereinstimmen. Wenn dies zutreffend ist, ist das Feld Validierung zu bestätigen, das Datum der Prüfung und der Namen anzugeben.

6.4.4 IO-Liste

Die IO-Liste dient dazu einen Überblick aller sicherheitsrelevanten Ein- und Ausgänge zu geben. Die IO-Liste wird aus dem Stromlaufplan aus Abschnitt 6.2 erstellt.

Nr	Beschreibung	Symbol	Adresse	Datentyp	Aktiv	Sperre	SW-Verif.	IO-Test	DIAG-Test
	Eingänge								
I1	Not-Halt, zweikanalig (NC) (1S1)	IS_EMST	I0.0	Bool	Aktiv	o			
I2	Kontakt1 Schutztür Roboterzelle SG1 (NC) (1B1)	IS_SG1_1	I15.0	Bool	Aktiv	x			
I3	Kontakt2 Schutztür Roboterzelle SG1 (NC) (1B2)	IS_SG1_2	I15.4	Bool	Aktiv	x			
I4	Kontakt1 Hubtor SG2 (NC) (2B1)	IS_SG2_1	I15.1	Bool	Aktiv	x			
I5	Kontakt2 Hubtor SG2 (NC) (2B2)	IS_SG2_2	I15.5	Bool	Aktiv	x			
I6	Kontakt1 Beladeklappe SG3 (NC) (3B1)	IS_SG3_1	I15.2	Bool	Aktiv	x			
I7	Kontakt2 Beladeklappe SG3 (NC) (3B2)	IS_SG3_2	I15.6	Bool	Aktiv	x			

Abbildung 41 SOFTEMA IO-Liste, Quelle: Eigene Darstellung

In Abbildung 41 ist ein Ausschnitt der IO-Liste zu sehen, jedem Ein- und Ausgang wird jeweils die spezifische Bezeichnung, die Adresse in der Steuerung und der Datentyp zugeordnet. Die Ein- und Ausgänge werden dem Stromlaufplan der Abbildung 26 entnommen. Mit dem Button „Formale Checks“ wird die spezifizierte IO-Liste auf Duplikate und fehlende Variablennamen überprüft. Nachdem alle Eingänge bzw. Ausgänge definiert worden sind, kann die jeweilige Zeile gesperrt werden.

Beim Vorgang des Codereviews bzw. bei der Softwarevalidierung (siehe Abschnitt 6.4.11 und 6.4.12) müssen zu einem späteren Zeitpunkt die Eintragungen mit der Software bzw. Hardware überprüft werden.

- SW-Verif zur Verifikation der Verschaltung der Signale in der Software
- IO-Test zur Validierung der Verschaltung direkt an der Hardware
- Diag-Test zur Verifikation der Diagnose bzw. Testfunktionen der Signale

Wenn dies zutreffend ist, ist dies in den zutreffenden Feldern zu bestätigen und das Datum der Prüfung mit dem Namen anzugeben.

6.4.5 Maßnahmen

Diese Checkliste dient dazu den Programmierer darauf hinzuweisen, dass bestimmte Maßnahmen wie eine einheitliche Bezeichnung der Variablen einzuhalten sind.

Nr	Maßnahmen	Aktiv	Verifikation
	allgemeine Maßnahmen		
	Namenssystematik für Variablen		
	Eingänge:		
R1	- Nicht sicherheitsrelevant: I_...	Aktiv	
R2	- Sicherheitsrelevant: IS_...	Aktiv	
	Ausgänge:		
R3	- Nicht sicherheitsrelevant: Q_...	Aktiv	
R4	- Sicherheitsrelevant: QS_...	Aktiv	
	Interne Variablen:		
R5	- Interne Variablen mit einem Bezug zu Eingängen beginnen möglichst mit einem Bezug zu den verarbeiteten Eingängen (z.B. Eingang Schutztür IS_SG1 ergibt Instanzvariable (Ausgang Schutztürüberwachungsbaustein) beginnend mit SG1_)	Aktiv	
R6	Die Variablennamen sollen möglichst selbsterklärend sein mit Bezug zu Bezeichnungen des Stromlaufplans.	Aktiv	
	Kommentare:		
R7	Jedes Netzwerk wird mit einer Überschrift und einem Kommentar versehen.	Aktiv	
R8	Jede Variable wird im Deklarationsteil mit einem Kommentar versehen.	Aktiv	
	Signalverarbeitung:		
R9	Im Programm werden Eingänge möglichst mit (Bibliotheks-)Bausteinen überwacht, um daraus die Abschaltfunktionen für die sicherheitsrelevanten Ausgänge über die Matrixnotation zu erzeugen.	Aktiv	
R10	Die Fehlersignale der I/O-Karten und die F-Softwarebausteine müssen wegen Wiederanlauf quittierpflichtig sein.	Aktiv	
R11	Alle zweikanaligen Taster werden in den I/O-Karten auf Diskrepanz überwacht (Diskrepanzzeit 50ms).	Aktiv	
R12	Alle Not-Halt Taster werden mit Not-Halt Überwachungsbausteinen überwacht.	Aktiv	
R13	Alle Schutztüren werden mit Schutztürüberwachungsbausteinen überwacht.	Aktiv	
R14	Schutztüren sind quittierpflichtig, ausgenommen sind Automatiktüren.	Aktiv	

Abbildung 42 SOFTEMA Maßnahmen zur Fehlervermeidung, Quelle: Eigene Darstellung

In Abbildung 42 ist ein Ausschnitt der Maßnahmen zu sehen, welche bei der Entwicklung der Software zu berücksichtigen sind. Wenn eine Maßnahme für diese Software nichtzutreffend ist, kann diese auf nicht aktiv geschaltet werden.

Im Rahmen des Codereviews (siehe Abschnitt 6.4.11) müssen die Maßnahmen gegen die Software geprüft werden, wenn eine Maßnahme eingehalten wird, dann ist dies in den zutreffenden Feldern zu bestätigen und das Datum der Prüfung mit dem Namen anzugeben.

6.4.6 Anforderungen an die Softwareentwicklung

Diese Checkliste dient dazu, den Programmierer auf die Forderungen des Abschnitts 4.6.3 der EN ISO 13849-1 hinzuweisen.

Nr	Anforderungen	PLr	Erfüllung	Aktiv	Validierung
	Basismaßnahmen:				
A1	Entwicklungslebenszyklus mit Verifikation und Validierung	a,b,c,d,e		Aktiv	
A2	Dokumentation der Spezifikation und Entwurf	a,b,c,d,e		Aktiv	
A3	modulare und strukturierte Programmierung	a,b,c,d,e		Aktiv	
A4	funktionale Tests	a,b,c,d,e		Aktiv	
A5	geeignete Entwicklungsaktivitäten nach Änderungen	a,b,c,d,e		Aktiv	
	Zusätzliche Maßnahmen mit steigender Wirksamkeit (niedrig bei PLr c, höher bei PLr e)				
	a) Spezifikation muss überprüft werden, jeder beteiligten Person verfügbar sein, Mindestbeschreibungen enthalten:				
A6	1) Sicherheitsfunktionen mit erforderlichem PL und zugehörigen Betriebsarten	c,d,e		Aktiv	
A7	2) Leistungskriterien, z. B. Reaktionszeiten,	c,d,e		Aktiv	
A8	3) Hardwarearchitektur mit externen Signalschnittstellen	c,d,e		Aktiv	
A9	4) Erkennung und Beherrschung externer Ausfälle	c,d,e		Aktiv	
	b) Auswahl der Werkzeuge, Bibliotheken, Sprachen:				
A10	1) Geeignete Werkzeuge mit Betriebsbewährung: Für PL = e, der mit einer Komponente und deren Werkzeug erreicht wird, muss das Werkzeug mit einer geeigneten Sicherheitsnorm übereinstimmen (weiteres in Norm)	c,d,e		Aktiv	
A11	2) Wann immer angemessen durchführbar, sollten nach Abschnitt 6.4 validierte Funktionsblock-Bibliotheken (FB) verwendet werden, bei PL = e sind vom Werkzeughersteller gelieferte FB-Bibliotheken besonders empfohlen	c,d,e		Aktiv	
A12	3) Eine begründete LVL-Teilmenge für ein modulares Verfahren sollte verwendet werden. Grafische Sprachen (z. B. Funktionsbaustein-Sprache, Kontaktplan) sind besonders empfohlen	c,d,e		Aktiv	

Abbildung 43 SOFTEMA Anforderungen an die Softwareentwicklung, Quelle: Eigene Darstellung

Die Anforderungen, der Abbildung 43, sind aufgeteilt in Basismaßnahmen und weitere Maßnahmen, welche im Abschnitt 4.2 und 4.6 genau beschrieben werden. Wenn ein PL_r von a oder b gefordert wird, sind die weiteren Maßnahmen nicht erforderlich und diese können auf nicht aktiv geschaltet werden. Beim Vorgang der Softwarevalidierung (siehe Abschnitt 6.4.12) müssen die Anforderungen mit den Vorgaben Sicherheitsfunktionen der Maschine übereinstimmen. Wenn dies zutreffend ist, ist das Feld Validierung zu bestätigen und das Datum der Prüfung mit dem Namen anzugeben.

6.4.7 Modularchitektur

Die Modularchitektur soll einen Überblick darüber geben, welche Funktionsbausteine für den Eingangsblock und den Ausgangsblock in der Software verwendet werden. Es sind nur möglich Bausteine zu verwenden, welche in der Bibliothek vermerkt sind. Die Bibliothek ist als .ini Datei im Projektordner abgelegt und kann vom Benutzer beliebig bearbeitet bzw. erweitert werden.

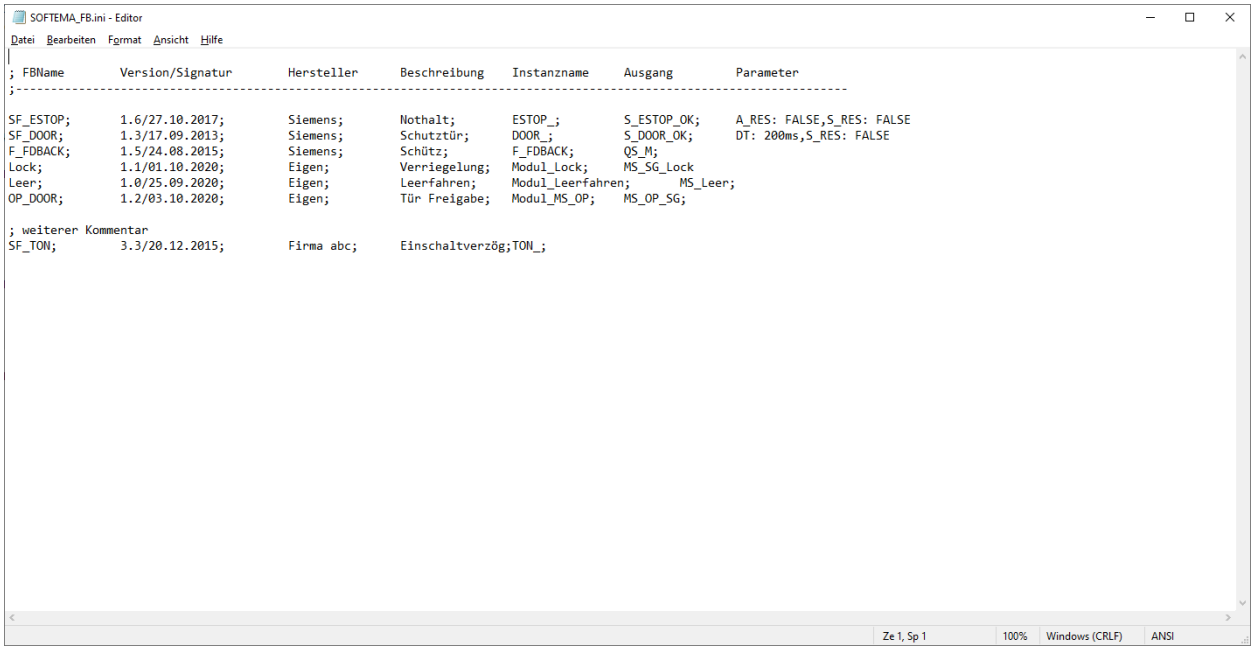


Abbildung 44 SOFTEMA Modul Bibliothek, Quelle: Eigene Darstellung

In Abbildung 44 ist der Aufbau der Bibliothek zu sehen, Standard F - Funktionsbausteine wie „SF_ESTOP“ usw. sind bereits in der Bibliothek angelegt, es ist jedoch nötig, die aktuelle Versionsnummer und das Versionsdatum zu ändern. Die selbstentwickelten Softwaremodule (siehe Abschnitt 6.4.1) „Lock“, „Leer“ und „OP_DOOR“ wurden mit Versionsnummer und Datum zu dieser Bibliothek hinzugefügt, damit diese in SOFTEMA verwendet werden können.

Nr	FB-Name	Instanznam	_Eingänge	_Ausgänge	_Beschreibung	_Parameter	_Hersteller	_Version/Signatur	Aktiv	Sperre	Verifikation
	Eingangsmodule										
IM1	SF_ESTOP	ESTOP_	IS_EMST	S_ESTOP_OK	0: Nothalt betätigt		Siemens	1.6/27.10.2017	Aktiv	o	
IM2	SF_DOOR	Schutztür SG1	IS_SG1_1 IS_SG1_2	SG1_OK	0: Schutztür geöffnet		Siemens	1.3/17.09.2013	Aktiv	x	
IM3	SF_DOOR	Schutztür SG2	IS_SG2_1 IS_SG2_2	SG2_OK	0: Schutztür geöffnet		Siemens	1.3/17.09.2013	Aktiv	x	

Abbildung 45 SOFTEMA Modularchitektur, Quelle: Eigene Darstellung

In Abbildung 45 ist ein Ausschnitt der Modularchitektur zu sehen, z.B. der Funktionsbaustein „SF_ESTOP“. Der Funktionsbaustein hat in der Software den Instanznamen „ESTOP_“, dieser besitzt den Eingang „IS_EMST“ und den Ausgang „S_ESTOP_OK“. Wenn der Not-Halt gedrückt ist, hat der Ausgang den Zustand *False*. Weitere wichtige Angaben sind die Herstellerbezeichnung und die Version des Bausteins.

Im Rahmen des Codereviews (siehe Abschnitt 6.4.11) müssen die Module gegen die Software geprüft werden. Wenn die angegebenen Module verwendet wurden, dann ist dies in den zutreffenden Feldern zu bestätigen und das Datum der Prüfung mit dem Namen anzugeben.

6.4.8 C+E Matrix

Die C+E Matrix dient dazu die Logik für die Ansteuerung des Ausgangsblocks zu erstellen. Bei dieser Methode werden alle Sicherheitsfunktionen, alle Eingänge und alle Ausgänge in der Ansicht automatisch dargestellt, d.h. es müssen für jede Sicherheitsfunktion die dazugehörigen Eingänge geschaltet werden und für jeden Ausgang muss das dazugehörige Modul ausgewählt bzw. verschalten werden, um das geforderte Verhalten lt. Sicherheitsfunktion abzubilden.

Nr	Betriebsart	Test	SF-Nr	SFK	Prio	SF-Name	O1	O2	O3	O4	O5	O6	Sperre	Verifikation	Validierung
							QS_M1 [Q21.0]	QS_M2 [Q21.1]	QS_M3 [Q21.2]	QS_M4 [Q21.3]	QS_ZUH_SG3 [Q21.4]	QS_ZUH_SG1 [Q21.5]			
C0						ALLOK	ON	ON	ON	ON	OFF	OFF	x		
C1	B0: Alle	C0	SF1	NOT-Halt	1	Wenn Not-Halt S1, dann Motor M1 abschalten, Motor M2 abschalten, Motor M3 abschalten, Motor M4 abschalten, mit Quttiertaster ACK quittieren.	OFF (*IM1*) ESTOP_.S_ESTOP_OK	OFF (*IM1*) ESTOP_.S_ESTOP_OK	OFF (*IM1*) ESTOP_.S_ESTOP_OK	OFF (*IM1*) ESTOP_.S_ESTOP_OK	NOP	NOP	x		

Abbildung 46 SOFTEMA C+E Matrix, Quelle: Eigene Darstellung

In Abbildung 46 sieht man einen Ausschnitt der sogenannten C+E Matrix. Die Sicherheitsfunktion 1 löst in allen Betriebsarten dann aus, wenn der Eingang IS_EMST den Zustand *False* aufweist. Welches Softwaremodul diese Sicherheitsfunktion umsetzt, wird im sogenannten Logikeditor der Abbildung 47 bestimmt. Die Ausführung der Sicherheitsfunktion wird über das Softwaremodul ESTOP_S realisiert, der Ausgang des Softwaremoduls S_ESTOP_OK beeinflusst direkt die Motoren M1, M2, M3 und M4.

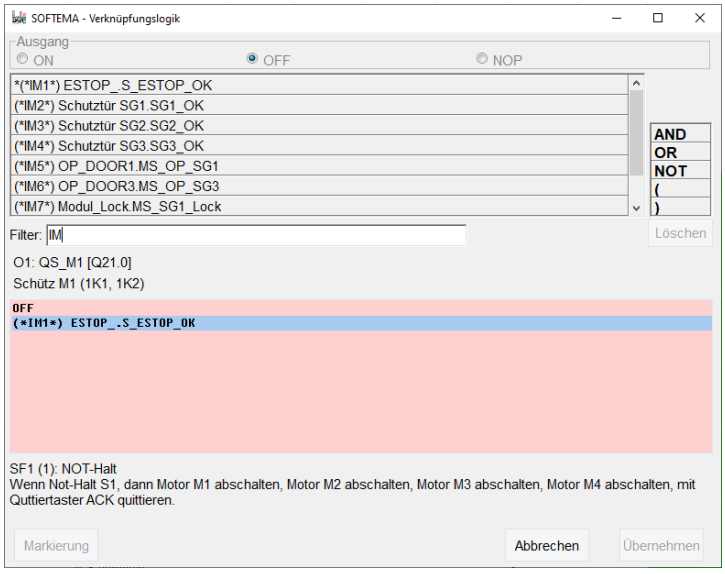


Abbildung 47 SOFTEMA Logikeditor Sicherheitsfunktion 1, Quelle: Eigene Darstellung

Der Logikeditor arbeitet nach der Negativlogik, d.h. soll z.B. eine Sicherheitsfunktion wie in Abbildung 48 ersichtlich, dann auslösen, wenn eine Sicherheitstür offen ODER nicht zugehalten ist, so muss im Logikeditor UND ausgewählt werden.

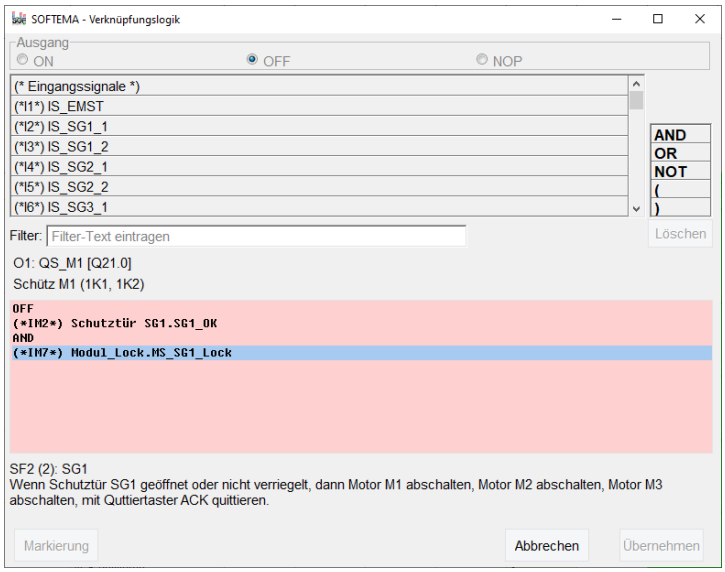


Abbildung 48 SOFTEMA Logikeditor Sicherheitsfunktion 2, Quelle: Eigene Darstellung

Die Verschaltung der Eingänge bzw. der Softwaremodule muss in diesem Schritt verifiziert werden, um bei der anschließenden Programmierung nur geprüfte Angaben einfließen zu lassen.

Beim Vorgang der Softwarevalidierung (siehe Abschnitt 6.4.12) müssen zu einem späteren Zeitpunkt die Eintragungen mit der Software bzw. Hardware überprüft werden. Wenn ein Eintrag als OK geprüft ist, ist dies in den zutreffenden Feldern zu bestätigen und das Datum der Prüfung mit dem Namen anzugeben.

6.4.9 C+E Matrix kompakt

Die Darstellung der C+E Matrix kann auch vereinfacht für jeden Aktor einzeln dargestellt werden. Diese Ansicht soll dazu dienen, dem Programmierer die Verschaltung der Verarbeitungsebene vorzugeben, sodass diese nur mehr im Programmiertool umgesetzt werden muss.

Nr	Ausgang	Beschreibung	B0: Alle	B1: Automatik	SF (Prio)	Verifikation	Validierung
E1	O1: QS_M1 [Q21.0]	Schütz M1 (1K1, 1K2)	OFF: (*IM1*) ESTOP_S_ESTOP_OK	OFF: (*IM2*) Schutztür SG1.SG1_OK AND (*IM7*) Modul_Lock.MS_SG1_Lock OFF: (*IM9*) .MS_Leer OFF: (*IM2*) Schutztür SG1.SG1_OK AND (*IM3*) Schutztür SG2.SG2_OK	B0:SF1 (1), B1:SF2 (2),SF6 (2),SF7 (2),	OK	

Abbildung 49 SOFTEMA C+E Matrix kompakt, Quelle: Eigene Darstellung

In Abbildung 49 sieht man das der Ausgang QS_M1 von folgenden Instanzen geschalten wird:

- ESTOP_SP.ESTOP_OK
- Schutztür SG1.SG1_OK AND Modul_Lock.MS_SG1_OK
- Schutztür SG1.SG1_OK AND Schutztür SG2.SG2_OK
- MS_Leer

Alle Betriebsarten

Betriebsart Automatik

Die Ansteuerung der Ausgänge muss in diesem Schritt verifiziert werden, um bei der anschließenden Programmierung nur geprüfte Angaben einfließen zu lassen.

Beim Vorgang der Softwarevalidierung (siehe Abschnitt 6.4.12) müssen zu einem späteren Zeitpunkt die Eintragungen mit der Software bzw. Hardware überprüft werden. Wenn ein Eintrag als OK geprüft ist, ist dies in den zutreffenden Feldern zu bestätigen und das Datum der Prüfung mit dem Namen anzugeben.

6.4.10 Programmierung

Die folgenden Ausführungen beschreiben die Umsetzung der spezifizierten Software im Programmiertool TIA Portal V16.

6.4.10.1 Hardwarekonfiguration

In den folgenden Ausführungen wird die Hardwarekonfiguration der Komponenten erläutert.

Abbildung 50 TIA Portal Hardwareaufbau, Quelle: Eigene Darstellung

In Abbildung 50 sind die projektierten Komponenten in der Software dargestellt, welche in Abschnitt 6.2 spezifiziert wurden.

	Name	Typ	Adresse	Variablentabelle	Kommentar
	IS_EMST_Ch1	Bool	%I0.0	Standard-Variablentabelle	Not-Halt, zweikanalig (NC) 1S1
	IS_SM1	Bool	%I0.1	Standard-Variablentabelle	Rückmeldung Schütz M1 (NC) (1K1, 1K2)
	IS_SM2	Bool	%I0.2	Standard-Variablentabelle	Rückmeldung Schütz M2 (NC) (2K1, 2K2)
	IS_SM3	Bool	%I0.3	Standard-Variablentabelle	Rückmeldung Schütz M3 (NC) (3K1, 3K2)
	IS_EMST_Ch2	Bool	%I0.4	Standard-Variablentabelle	Not-Halt, zweikanalig (NC) 1S1

Abbildung 51 TIA Portal Variablentabelle, Quelle: Eigene Darstellung

Die IO-Variablentabelle in Abbildung 51 wurde wie in Abschnitt 6.4.4 spezifiziert, mit den Maßnahmen an die Software laut Abschnitt 6.4.5 Variablenbezeichnungen beginnend mit IS_... und den dazugehörigen Kommentaren aufgebaut.

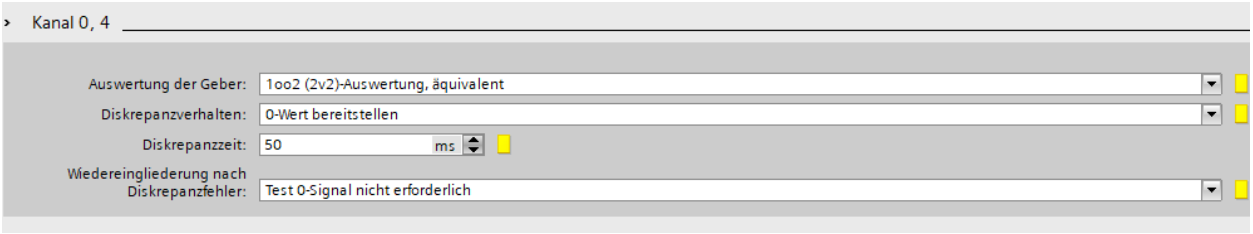


Abbildung 52 TIA Portal Diskrepanzüberwachung, Quelle: Eigene Darstellung

Da hier ein Aufbau nach Kategorie 3 lt. Abschnitt 6.3 gefordert wird, ist es z.B. notwendig, für das Signal des Not-Halt, wie in Abbildung 52 ersichtlich, einen Diskrepanzüberwachung durchzuführen. Dies bedeutet, dass eine Prüfung stattfindet, ob der Kanal 0 und der Kanal 4 denselben booleschen Zustand aufweisen.

Sollte dies nicht der Fall sein, wird automatisch ein 0-Wert bereitgestellt. Aufgrund der Mechanik in einem Not-Halt Schaltgerät muss auch eine Diskrepanzzeit angegeben werden, d.h. wenn ein Kanal den Wert ändert, hat der zweite Kanal 50ms Zeit um denselben Wert anzunehmen, ansonsten wird automatisch der 0-Wert bereitgestellt.

6.4.10.2 Softwareprogrammierung

In den folgenden Ausführungen wird auf die Umsetzung der Software näher eingegangen. Die gesamte Software wurde in FUP (Funktionsplan) realisiert.

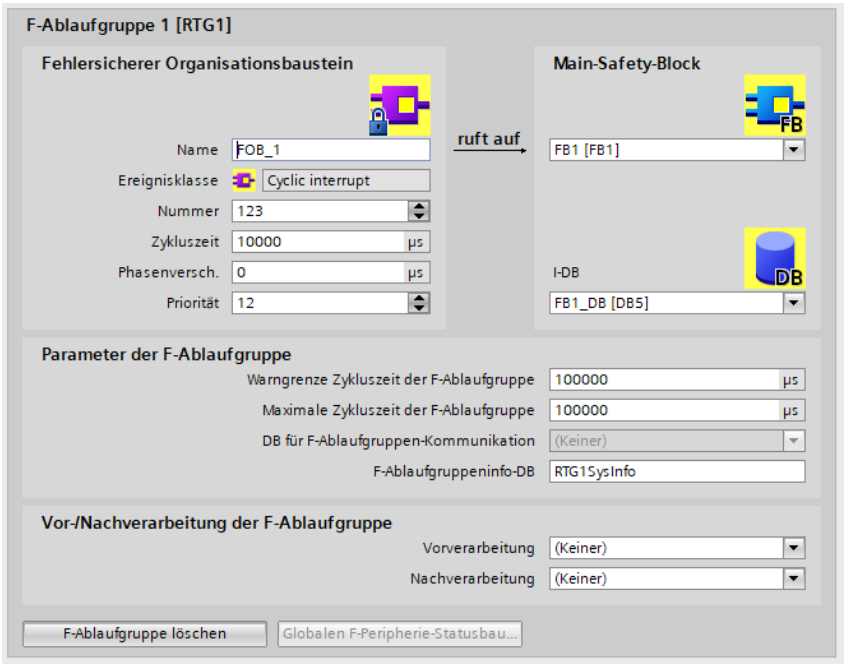


Abbildung 53 TIA Portal F-Parameter, Quelle: Eigene Darstellung

In Abbildung 53 sind die Einstellungen der F-Ablaufgruppe ersichtlich. Als Main Baustein dient der Funktionsbaustein FB1 und die Zykluszeit ist 10ms.

Der Softwareaufbau wurde wie in Abschnitt 4.6.4 gefordert in Blöcke realisiert.

Eingangsblock:

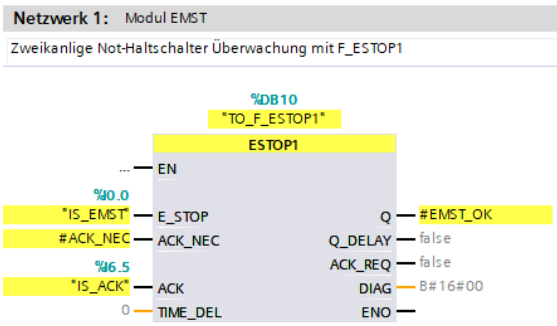


Abbildung 54 TIA Portal Eingangsblock ESTOP, Quelle: Eigene Darstellung

In Abbildung 54 sieht man den Aufbau des Eingangsblocks anhand des Beispiels des Not-Halt Schaltgerät. Die Bezeichnung der Eingänge und Ausgänge und die Bezeichnung des Moduls wurden lt. Abschnitt 6.4.5

realisiert. Der Eingang ACK_NEC wird dauerhaft mittels einer statischen Variable auf *True* gesetzt, d.h. ein „Reset“ ist nach Auslösen des Not-Halt immer erforderlich.

Verarbeitungsblock:

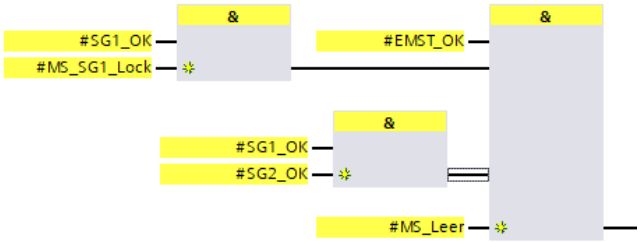


Abbildung 55 TIA Portal Verarbeitungsblock M1, Quelle: Eigene Darstellung

In Abbildung 55 ist der Aufbau des Verarbeitungsblocks zu sehen, dieser wurde laut den Vorgaben der C+E Matrix kompakt des Abschnitts 6.4.9 aufgebaut.

Ausgangsblock:

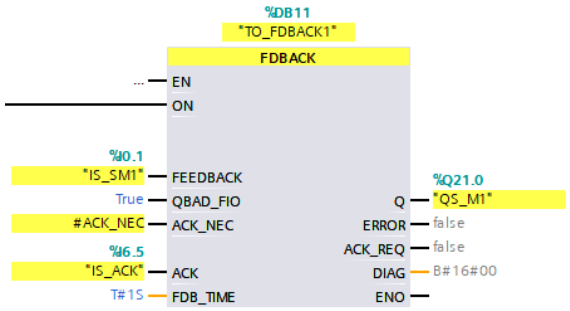


Abbildung 56 TIA Portal Ausgangsblock M1, Quelle: Eigene Darstellung

In Abbildung 56 sieht man den Aufbau des Ausgangsblocks hier anhand des Beispiels des Motor 1 bzw. der Schützsteuerung. Die Bezeichnung der Eingänge und Ausgänge und die Bezeichnung des Moduls wurden lt. Abschnitt 6.4.5 realisiert. Der Eingang ACK_NEC wird dauerhaft mittels einer statischen Variable auf *True* gesetzt, d.h. ein „Reset“ ist nach Auslösen des Not-Halt immer erforderlich. Überdies wird als Rücklesekontakt der Eingang IS_M1 eingelesen, auf Grund der Mechanik in einem Schütz muss hier eine Rücklesezeit angegeben werden, die dieser Kontakt maximal für eine Wertänderung benötigen darf.

6.4.11 Codereview

Wenn die Programmierung der Software durchgeführt wurde, muss ein sogenannter Codereview durchgeführt werden.

Nr	Beschreibung	Referenzblatt	Verifikation
R1	Sind die vereinbarten fehlervermeidenden Maßnahmen und Tools und Programmierregeln eingehalten worden?	A3 Maßnahmen	OK
R2	Ist der Systemaufbau der Hardware umgesetzt worden?	A2.3 Systemaufbau	OK
R3	Ist die Verschaltung der I/O-Signale korrekt umgesetzt?	A2.4 IO-Liste	OK
R4	Ist die Architektur des Sicherheitsprogramms eingehalten worden?	B1 Architektur Sicherheitspr.	OK
R5	Ist die Modularchitektur eingehalten worden?	B3 Modularchitektur	OK
R6	Ist die Spezifikation der Software aus der Matrix umgesetzt?	B4 Matrix C+E	OK
€€€			
		Summe	OK
		Datum	11.10.2020
		Name	Programm

Abbildung 57 SOFTEMA Codereview, Quelle: Eigene Darstellung

In Abbildung 57 ist die Übersicht der Softwarevalidierung abgebildet. Es müssen alle zuvor definierten Spezifikationen und Eintragungen gegen die Software verifiziert werden. Die Verifizierung muss in der jeweiligen Ansicht durchgeführt werden. Wenn die Verifizierung im jeweiligen Dokument durchgeführt wurde, wird dies hier als OK dargestellt. In dieser Ansicht ist anzugeben, ob der Systemaufbau in der Hardware und ob die Modularchitektur in der Software umgesetzt wurden.

Beispiel für die Korrektheit der Verschaltung der IO-Signale:

Nr	Beschreibung	Symbol	Adresse	Datentyp	Aktiv	Sperre
	Eingänge					
I1	Not-Halt, zweikanalig (NC) (1S1)	IS_EMST	I0.0	Bool	Aktiv	x

Abbildung 58 SOFTEMA Codereview IO-Liste, Quelle: Eigene Darstellung

	Name	Typ	Adresse	Variablentabelle	Kommentar
	IS_EMST	Bool	%I0.0	Standard-Variablentabelle	Not-Halt, zweikanalig (NC) 1S1

Abbildung 59 TIA Portal Codereview IO-Liste, Quelle: Eigene Darstellung

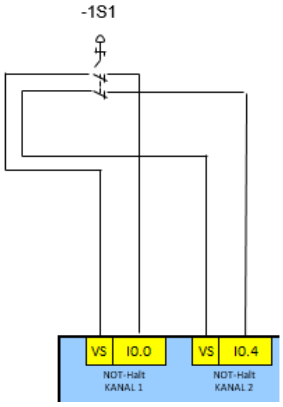


Abbildung 60 Schaltplan Codereview IO-Liste, Quelle: Eigene Darstellung

Zu prüfen sind die Bezeichnung, die Beschreibung, die Adresse und der Datentyp. In den Abbildungen 58, 59 und 60 ist ersichtlich, dass der projektierte Not-Halt sich in allen IO-Listen deckt und somit als OK markiert werden kann.

6.4.12 Softwarevalidierung

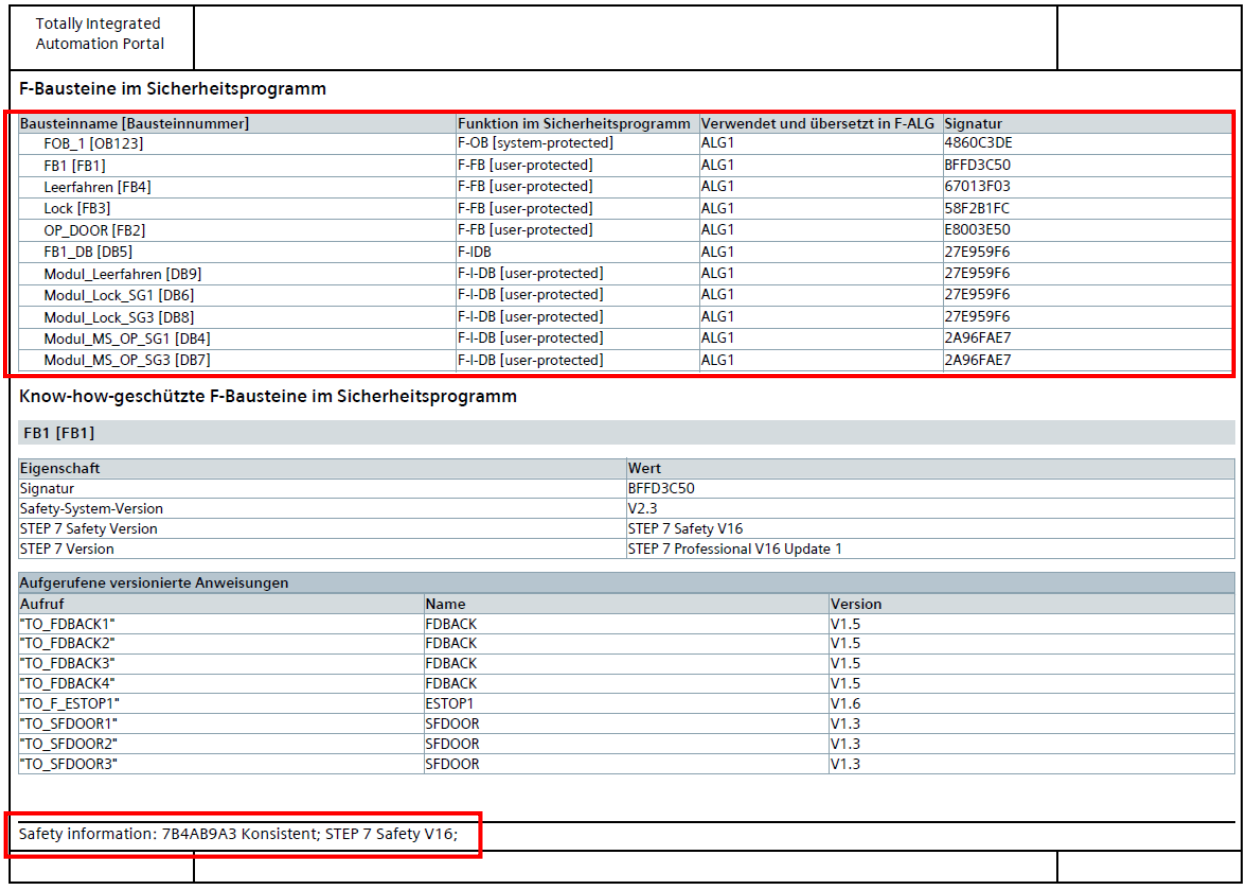
Nachdem der Codereview durchgeführt wurde, muss die Software überprüft werden, ob sie mit den Spezifikationen der Sicherheitsanforderungen der Maschine übereinstimmt.

Nr	Beschreibung	Referenzblatt	Validierung
	Wurden die Aktivitäten durchgeführt?		
V1	Validierung Sicherheitsfunktionen (D1)	A1 Sicherheitsfunktionen	OK
V2	Validierung I/O-Check (D1)	A2.4 IO-Liste	OK
V3	Validierung normativer Anforderungen (D1)	A4 Anforderungen	OK
V4	Verifikation der Modularchitektur (V1)	B3 Modularchitektur	OK
V5	Verifikation der Matrix (V1)	B4 Matrix C+E	OK
V6	Validierung Matrix (D1)	B4 Matrix C+E	OK
V7	Verifikation Codereview	C1 Codereview	OK
V8	Prüfung der Peripheriegeräte		OK
V9	Prüfung der Sensoren		OK
	Ist die Dokumentation komplett?		
D1	Dokumente des V-Modells aus diesem Excel-Dokument		OK
D2	PDF-Ausdruck aller sicherheitsrelevanten Software inkl. Checksumme		OK
D3	PDF-Ausdruck der Hardwarekonfiguration (mit allen Einstellungen) inkl. Checksumme		OK
D4	Archivierung der Handbücher aller Systemkomponenten		OK
D5	PDF-Ausdruck der Konfiguration von Peripheriegeräten inkl. Checksummen		OK
D6	Abnahmevorschriften der Hersteller (z.B. Parametrierung von Sicherheitskomponenten)		OK
D7	Einzuhaltende Vorgaben aus C-Normen		OK
D8	Einzuhaltende Vorgaben aus B-Normen		OK

Abbildung 61 SOFTEMA Softwarevalidierung, Quelle: Eigene Darstellung

In Abbildung 61 ist die Übersicht der Softwarevalidierung zu sehen. Es müssen alle zuvor definierten Spezifikationen und Eintragungen validiert werden. Die Validierung muss in der jeweiligen Ansicht durchgeführt werden. Wenn die Validierung im jeweiligen Dokument durchgeführt wurde, wird dies hier als OK dargestellt. Es sind außerdem Angaben zu treffen, ob Peripheriegeräte und Sensoren geprüft worden sind, weiters sind Angaben zur Dokumentation zu treffen.

Ein wichtiger Punkt hierbei ist der Ausdruck des Sicherheitsprogramms, in Abbildung 62 ist ein Ausschnitt des Ausdrucks zu sehen. Es sind alle Softwaresignaturen dokumentiert, die Bausteine sind allesamt Know-how geschützt und die Software ist konsistent.



Totally Integrated Automation Portal

F-Bausteine im Sicherheitsprogramm

Bausteinname [Bausteinnummer]	Funktion im Sicherheitsprogramm	Verwendet und übersetzt in F-ALG	Signatur
FOB_1 [OB123]	F-OB [system-protected]	ALG1	4860C3DE
FB1 [FB1]	F-FB [user-protected]	ALG1	BFFD3C50
Leerfahren [FB4]	F-FB [user-protected]	ALG1	67013F03
Lock [FB3]	F-FB [user-protected]	ALG1	58F2B1FC
OP_DOOR [FB2]	F-FB [user-protected]	ALG1	E8003E50
FB1_DB [DB5]	F-IDB	ALG1	27E959F6
Modul_Leerfahren [DB9]	F-I-DB [user-protected]	ALG1	27E959F6
Modul_Lock_SG1 [DB6]	F-I-DB [user-protected]	ALG1	27E959F6
Modul_Lock_SG3 [DB8]	F-I-DB [user-protected]	ALG1	27E959F6
Modul_MS_OP_SG1 [DB4]	F-I-DB [user-protected]	ALG1	2A96FAE7
Modul_MS_OP_SG3 [DB7]	F-I-DB [user-protected]	ALG1	2A96FAE7

Know-how-geschützte F-Bausteine im Sicherheitsprogramm

FB1 [FB1]

Eigenschaft	Wert
Signatur	BFFD3C50
Safety-System-Version	V2.3
STEP 7 Safety Version	STEP 7 Safety V16
STEP 7 Version	STEP 7 Professional V16 Update 1

Aufgerufene versionierte Anweisungen

Aufruf	Name	Version
"TO_FDBACK1"	FDBACK	V1.5
"TO_FDBACK2"	FDBACK	V1.5
"TO_FDBACK3"	FDBACK	V1.5
"TO_FDBACK4"	FDBACK	V1.5
"TO_F_ESTOP1"	ESTOP1	V1.6
"TO_SFDOOR1"	SFDOOR	V1.3
"TO_SFDOOR2"	SFDOOR	V1.3
"TO_SFDOOR3"	SFDOOR	V1.3

Safety information: 7B4AB9A3 Konsistent; STEP 7 Safety V16;

Abbildung 62 TIA Portal Softwaredokumentation, Quelle: Eigene Darstellung

Der Sicherheitsausdruck umfasst auch die gesamte Hardwarekonfiguration, eine IO-Liste, den Modulaufbau und die Verschaltung der einzelnen Module.

6.4.13 Protokoll

Nachdem alle Punkte abgearbeitet wurden, ist es möglich die gesammelte Spezifikation bzw. die Dokumentation als Zusammenfassung auszudrucken.

Übersicht der Tabellen

Sicherheitsfunktionen:	Anzahl: 7		100 % validiert
IO-Signale:	Anzahl: 26	100 % verifiziert	100 % validiert
Maßnahmen:	Anzahl: 44	100 % verifiziert	
Anforderungen:	Anzahl: 36		100 % validiert
Module:	Anzahl: 12	100 % verifiziert	
Zeilen CE-Matrix:	Anzahl: 8	100 % verifiziert	100 % validiert
Zeilen CE-kompakt:	Anzahl: 6	100 % verifiziert	100 % validiert
Fragen Codereviews:	Anzahl: 6	100 % verifiziert	
Fragen Validierung:	Anzahl: 17		100 % validiert
Personen:	Anzahl: 13		
Dokumente:	Anzahl: 13		
Änderungen:	Anzahl: 18		

Abbildung 63 SOFTEMA Zusammenfassung, Quelle: Eigene Darstellung

In Abbildung 63 sind alle bearbeiteten Tabellen ersichtlich und ob diese Tabellen verifiziert und validiert wurden.

Liste der CE-Matrix-Kompakt

Anzahl Zeilen: 6	Anzahl verifiziert: 6 (100 %)	Anzahl validiert: 6 (100 %)
Verifizieren Datum/Name: 11.10.2020 / Programmierer		
Veri.Prüfen1 Datum/Name: 11.10.2020 / Prüfer 1		
Veri.Prüfen2 Datum/Name: <leer> / <leer>		
Veri.Signatur Programm: <leer>		
Validieren Datum/Name: 11.10.2020 / Programmierer 2		
Vali.Prüfen1 Datum/Name: 11.10.2020 / Prüfer 1		
Vali.Prüfen2 Datum/Name: <leer> / <leer>		
Vali.Signatur Programm: <leer>		
CEC E1: O1: QS_M1 [Q21.0] = Schütz M1 (1K1, 1K2)		
Nr (Prio): B0:SF1 (1), B1:SF2 (2),SF6 (2),SF7 (2),	Verifikation: OK	Validierung: OK
CEC E2: O2: QS_M2 [Q21.1] = Schütz M2 (2K1, 2K2)		
Nr (Prio): B0:SF1 (1), B1:SF2 (2),SF3 (2),SF6 (2),	Verifikation: OK	Validierung: OK
CEC E3: O3: QS_M3 [Q21.2] = Schütz M3 (3K1, 3K2)		
Nr (Prio): B0:SF1 (1), B1:SF2 (2),SF6 (2),SF7 (2),	Verifikation: OK	Validierung: OK
CEC E4: O4: QS_M4 [Q21.3] = Schütz M4 (4K1, 4K2)		
Nr (Prio): B0:SF1 (1), B1:SF3 (2),SF6 (2),SF7 (2),	Verifikation: OK	Validierung: OK
CEC E5: O5: QS_ZUH_SG3 [Q21.4] = Entriegelung Zuhaltung Schutztür 3 (5K1)		
Nr (Prio): B0: B1:SF5 (2),	Verifikation: OK	Validierung: OK
CEC E6: O6: QS_ZUH_SG1 [Q21.5] = Entriegelung Zuhaltung Schutztür 1 (6K1)		
Nr (Prio): B0: B1:SF4 (2),	Verifikation: OK	Validierung: OK

Ende Liste der CE-Matrix-Kompakt

Abbildung 64 SOFTEMA C+E Matrix kompakt Protokoll, Quelle: Eigene Darstellung

In dem Ausschnitt der Dokumentation der Abbildung 64 sieht man die Ansteuerung der einzelnen Ausgänge über die verschiedenen Softwaremodule. Das Protokoll der bearbeiteten Punkte bzw. Tabellen, ist mitsamt der anderen Dokumentation, z.B. Sicherheitsausdruck der Software als interne Dokumentation vom Hersteller zu archivieren.

7 ZUSAMMENFASSUNG, ERGEBNISSE UND AUSBLICK

Im folgenden Abschnitt wird ein Resümee über die Arbeit und die gewonnenen Ergebnisse gezogen, des Weiteren wird ein Ausblick gegeben, ob und in welcher Form die Software in Zukunft im Unternehmen eingesetzt werden soll.

7.1 Zusammenfassung und Ergebnisse

Der immer größer werdende Teil an Realisierung von Sicherheitsfunktionen mit sicherheitsgerichteter Anwendersoftware hat hohe Anforderungen an die Fehlervermeidung und an die Softwarequalität. Die MSV 2010 erwähnt bereits in den Ausführungen, dass durch die Programmierung keine Gefährdung ausgehen darf. Wenn bei der Risikobeurteilung im „Drei-Stufen-Verfahren“ zur Risikominderung ausgewählt wird, dass eine „technische Schutzmaßnahme und ergänzender Sicherheit“ umzusetzen ist, fällt sehr oft die Wahl auf funktionale Sicherheit. Die praktische Umsetzung der Hardware nach EN ISO 13849-1 ist bereits seit Jahren im Maschinenbau bekannt und sehr gut publiziert. Die Umsetzung der Software hingegen weist so gut wie keine publizierten praktischen Beispiele auf. Die eingesetzten Softwaretools zur Bestimmung des Performancelevels, z.B. SISTEMA, weisen den Benutzer darauf hin, dass die Software nach Abschnitt 4.6 der EN ISO 13849-1 umzusetzen ist. Im Abschnitt 4.6 der EN ISO 13849-1 wird ein V-Modell vorgestellt, wie ein Softwarelebenszyklus auszusehen hat. Wenn für die Umsetzung der Software ein Standard Programmiertool, z.B. TIA Portal verwendet wird, werden im Abschnitt 4.6.3 der EN ISO 13849-1 sogenannte Basismaßnahmen vorgestellt, welche immer einzuhalten sind. Wenn jedoch ein höherer Performance Level als b gefordert wird, sind weitere Maßnahmen erforderlich. Die Beschreibung dieser Maßnahmen ist teilweise sehr allgemein gehalten, deswegen wird in mehreren Bereichen auf die Normenreihe EN 61508 verwiesen, diese gibt in vielen Bereichen sehr konkrete Beispiele, wie Maßnahmen umgesetzt werden müssen.

Anhand des Beispiels konnte der Umgang bzw. die Anwendung der Matrixmethode des IFA bzw. die Umsetzung in der Software SOFTEMA getestet werden. Die Software SOFTEMA kann auf jeden Fall als Unterstützung in der Projektierung und Überprüfung eingesetzt werden, denn die Vorteile überwiegen gegenüber den Nachteilen. In SOFTEMA werden alle Forderungen der EN ISO 13849-1 abgebildet und in weiterer Folge auch überprüft, ob diese eingehalten worden sind. Die Spezifikation der Sicherheitsfunktionen ist einer der wichtigsten Angaben, wenn diese nicht vollständig sind, gibt es immer Interpretationsspielräume. Die zur Verfügung gestellten Excel Arbeitsblätter um eigene Softwaremodule zu entwickeln, geben dem Programmierer die Sicherheit, dass auf keinen der vorgegebenen Schritte vergessen werden kann. Die Ansicht der Matrix hat für den Test des Softwaremoduls große Vorteile, da das Soll-Schaltverhalten auf einen Blick zu sehen ist. Die Ansichten der Maßnahmen und Anforderungen sind für den Programmierer eine sehr gute Stütze bzw. ein Nachschlagewerk, wenn bei der Programmierung selbst Fragen auftreten, wie etwas lt. Norm umgesetzt werden muss. Die matrixbasierte Spezifizierung der Verarbeitungsblocks nimmt dem Programmierer eigentlich den Teil der Programmierung vorweg, denn die entwickelte Matrix ist lediglich nur mehr in Software umzusetzen. Bei der Programmierung selbst, hat die IO-Liste große Vorteile gegenüber anderen üblichen Aufzeichnungen, da die Adressen, die genauen Bezeichnungen usw. in einer Liste zusammengefasst sind. Nachdem die Programmierung abgeschlossen ist, gibt der Codereview einen Leitfaden vor, wie und welche Punkte verifiziert werden

müssen. Die Softwarevalidierung am Ende gibt dem Programmierer die Sicherheit, dass die umgesetzte Software in Verbindung mit der Hardware die Anforderungen an die Sicherheit der Maschine erfüllt. Die Zusammenfassung bzw. das Protokoll hilft dabei alle einzelnen Ansichten nochmals transparent darzustellen und dient als Teil der Softwaredokumentation.

In dieser Beta Version gibt es noch keine Schnittstellen zu anderen Softwareprodukten. Es wäre von Vorteil, wenn die Auflistung der Sicherheitsfunktionen z.B. direkt aus der Software SISTEMA entnommen werden könnte. Der Aufwand an Dokumentation ist bei größeren Projekten auf keinen Fall zu unterschätzen, da jede einzelne Sicherheitsfunktion, jeder Eingang und Ausgang per Hand eingefügt und verarbeitet werden muss. Wegen der Fülle an Eingängen und Sicherheitsfunktionen wird die C+E Matrix schnell sehr unübersichtlich. Ein weiterer wichtiger Punkt ist, dass ein Teil der Dokumentation doppelt vorhanden ist. Die Definition der Sicherheitsfunktionen ist in der Software zur Berechnung des erreichten Performance Levels z.B. SISTEMA abgebildet. Informationen zur Software, wie die Modularchitektur, Verschaltung der Module, die IO-Liste mit eingestellten Parametern (z.B. Diskrepanzüberwachung), werden bereits mit dem Sicherheitsausdruck der Software dokumentiert. Die Verwaltung bzw. die Ansicht für die verschiedenen Benutzer der Software ist etwas unübersichtlich, da jeder Benutzer z.B. als Administrator alle Einträge bearbeiten könnte. Eine mögliche Umsetzung wäre hier eine Benutzerverwaltung mit Passwort oder über den Windows User.

7.2 Ausblick

Ein Ziel dieser Arbeit war es eine Entscheidung zu treffen, ob diese Software in Zukunft im Unternehmen eingesetzt wird. Die Software wird auf jeden Fall als Unterstützung der Programmierer bei der täglichen Arbeit eingesetzt. Die nächsten Schritte sind es, den Softwareassistenten zur Unterstützung bei umfangreicheren Projekten zu verwenden. Ein weiterer Punkt ist das Testen des Softwareassistenten für den Fall einer Softwareänderung bei einer Maschine. Durch diese Arbeit ist außerdem ein neues Projekt ins Leben gerufen worden, nämlich eine konzernweite Standardisierung im Bereich Programmierung. Diese umfasst einheitliche Programmierrichtlinien, definierte Tests, wie zum Beispiel das Durchführen von einheitlichen Tests bei allen Anlagen und das Einführen von definierten Projektteams, um das Vier-Augen-Prinzip konsequent umzusetzen.

8 LITERATURVERZEICHNIS

Gedruckte Werke (4)

Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)(DGUV), Deutsche (Hrsg.) (2016): *Sicherheitsbezogene Anwendungssoftware von Maschinen – Die Matrixmethode des IFA*, Berlin

Institut für Arbeitsschutz der Deutschen Gesetzlichen Unfallversicherung (IFA)(DGUV), Deutsche (Hrsg.) (2017): *Funktionale Sicherheit von Maschinensteuerungen*, Berlin

SIEMENS AG (Hrsg.) (2019): *Industrie Software SIMATIC Safety - Projektieren und Programmieren*, SIEMENS AG, Nürnberg

Kleuker, Stephan (2008): *Der pragmatische Weg zu erfolgreichen Softwareprojekten*, 4 Auflage, Springer Vieweg, Osnabrück

Konferenzbeiträge (4)

Boehm, B. (1979): *Guidelines for Verifying and Validating Software Requirements and Design Specifications. In P. A. Samet (ed.)*, in: Samet, P. (Hrsg.): *EURO IFIP 79: Proceedings of the European Conference on Applied Information Technology of the International Federation for Information Processing*, London, S. 711-719

Huelke, Michael (2007): *SISTEMA: Ein Tool zur einfachen Anwendung der Steuerungsnorm ENN ISO 13849-1*, in: Unfallversicherungsanstalt, Deutsche (Hrsg.): *SPS/IPC DRIVES Elektrische Automatisierung - Systeme und Komponenten*, Nürnberg

J. Gausemeier, S. (2003): *NEW GUIDELINE VDI 2206 – A FLEXIBLE PROCEDURE MODEL FOR THE DESIGN OF MECHATRONIC SYSTEMS*, in: Heinz Nixdorf Institut (Hrsg.): *INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN*, Universität Paderborn, Paderborn

Michael Huelke, Norbert (2015): *IFA Matrix Method for development of safety related application software*, in: (DGUV), Deutsche (Hrsg.): *8th International Conference Safety of Industrial Automated Systems – SIAS 2015* , Königswinter, S. 76-79

Normen (10)

Austrian Standards (Hrsg.)*EN 61508-1:2011: Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme Teil 1: Allgemeine Anforderungen*

Austrian Standards (Hrsg.)*EN 61508-3:2011: Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme Teil 3: Anforderungen an Software*

Austrian Standards (Hrsg.)*EN 61508-4: 2001: Funktionale Sicherheit elektrischer/elektronischer/programmierbar elektronischer sicherheitsbezogener Systeme - Teil 4: Begriffe und Abkürzungen*

Austrian Standards (Hrsg.)*EN 61508-7:2011: Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme Teil 7: Überblick über Verfahren und Maßnahmen*

Austrian Standards (Hrsg.)*EN ISO 12100:2013: Sicherheit von Maschinen - Allgemeine Gestaltungsleitsätze - Risikobeurteilung und Risikominderung*

Austrian Standards (Hrsg.)*EN ISO 13849-1:2016: Sicherheit von Maschinen — Sicherheitsbezogene Teile von Steuerungen Teil 1: Allgemeine Gestaltungsleitsätze*

Austrian Standards (Hrsg.)*EN ISO 13849-2:2013: Sicherheit von Maschinen — Sicherheitsbezogene Teile von Steuerungen Teil 2: Validierung*

Austrian Standards (Hrsg.)*ISO/TR 14121-2:2012: Sicherheit von Maschinen – Risikobeurteilung – Teil 2: Praktischer Leitfaden und Verfahrensbeispiele*

Austrian Standards (Hrsg.)*ISO/TR 23849:2010: Leitfaden zur Anwendung von ISO 13849-1 und IEC 62061 bei der Gestaltung von sicherheitsbezogenen Steuerungen für Maschinen*

Bundesgesetz Republik Österreich (Hrsg.) (2020): *MSV 2010: Maschinen-Sicherheitsverordnung 2010*

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGSVERZEICHNIS

C+E	Cause & Effect
CCF	Common Cause Failure
CE	Conformité Européenne
DC	Diagnostic Coverage
DGUV	Deutsche gesetzliche Unfallversicherung
EG	Europäische Gemeinschaft
EN	Europäische Norm
FVL	Full Variability Language
IFA	Institut für Arbeitssicherheit
ISO	Internationale Organisation für Normung
lt.	laut
LVL	Limited Variability Language
MRL	Maschinenrichtlinie
MSV	Maschinensicherheitsverordnung
$MTTF_D$	Mean Time To Dangerous Failure
PFH_D	Probability of a Dangerous Failure per Hour
PL	Performance Level
PL_r	Performance Level Required
SISTEMA	Sichere Steuerungen von Maschine
SOFTEMA	Software von Steuerungen an Maschinen
SPS	Speicher Programmierbare Steuerung
SRASW	Safety Related Application Software
SRESW	Safety Related Embedded Software
SRP/CS	Safety-Related Parts of Control System
TIA	Totally Integrated Automation
VDI	Verband Deutscher Ingenieure
V-Modell	Vorgehens-Modell
z.B.	zum Beispiel