DOI 10.5162/sensoren2016/P6.1

# High-Assurance System Development with LabVIEW

*Nick Berezowski[1], Prof. Dr. Markus Haid[1]*
*[1]CCASS Hochschule Darmstadt, Birkenweg 8, 64295 Darmstadt*

## Zusammenfassung

Die Entwicklung zukünftiger sicherheitsbezogener Automatisierungssysteme hängt im Allgemeinen von der Grundnorm IEC 61508 ab. Diese stellt Anforderungen an den gesamten Sicherheitslebenszyklus. Hierbei unterscheidet man zwischen Hardware- und Softwarekonzeption, sowie dem Management eines Projekts, mit allen enthaltenen Komponenten.

Ein wesentlicher Bestandteil der Konzept- und Entwicklungsphasen ist die Softwarearchitektur für sicherheitskritische Systeme. Hierbei sollte eine Entwicklungsumgebung eingesetzt werden, die diesen Prozess angemessen unterstützt, indem sie schon vorzertifizierte Strukturen besitzt oder Leitlinien vorgibt. Durch immer komplexer werdende sicherheitskritische Systemstrukturen von zu automatisierenden Systemen, entsteht die Notwendigkeit der Weiterentwicklung solcher Entwicklungsumgebungen und Programmiersprachen zur Entwicklung modulbasierter übersichtlicher Sicherheitsfunktionalitäten.

Durch eine ausführliche Analyse der Grundnorm, Teilbereichsnormen und Programmierrichtlinien kann ein allgemeines Verständnis der einzuhaltenden Sicherheitsaspekte geschaffen werden. Anschließend bildet sich hieraus eine Beurteilung der Software LabVIEW des Unternehmens National Instruments, gegenüber den gestellten Anforderungen der Grundnorm, die einen Beweis erbringt, dass die Entwicklung von sicherheitsbezogenen Produkten in LabVIEW möglich ist. Hauptaugenmerk liegt hierbei in den Vorteilen der Programmiersprache gegenüber anderen und wie bestimmte Schwächen, durch strukturelle Vorgaben, verbessert werden können.

Indem Software-, Hardware-, Management- und Zertifizierungsvoraussetzungen, bezogen auf die sicherheitsbezogenen Kriterien und die Struktur LabVIEW´s, analysiert werden, entsteht eine Auflistung von vorhandenen und benötigten Eigenschaften. Insbesondere wird, neben dem Hauptaugenmerk auf die Softwareentwicklung und -oberfläche zur Entwicklung, Wert auf Managementtools und Hardwarestrukturen zur einfachen Zertifizierung gelegt.

**Keywords:** LabVIEW, Safety, funktionale Sicherheit, Competence Center for High-Assurance System Development, MISRA

### Grundnorm – IEC 61508

Im funktionalen sicherheitskritischen Bereich kommen die in Abbildung 1 dargestellten Normen zum Einsatz. Diese bilden jedoch nur einen kleinen Teil der existierenden Normen ab. Als Grundsatznorm wird die, in der Mitte der Abbildung 1 gezeigte, IEC 61508 genutzt. In dieser werden alle grundlegenden und weiterführenden Themenbereiche zur Herstellung eines sicherheitskritischen Produktes ausgeführt.
Durch steigende Komplexität und wachsenden Fortschritt in der heutigen Industrie ist es notwendig, Spezifikationen der einzelnen Industriesektoren zu trennen. Dies gewährleistet ein einfaches Verständnis der zu berücksichtigenden Anforderungen und klarere Definition der durchzuführenden Maßnahmen und Methoden.

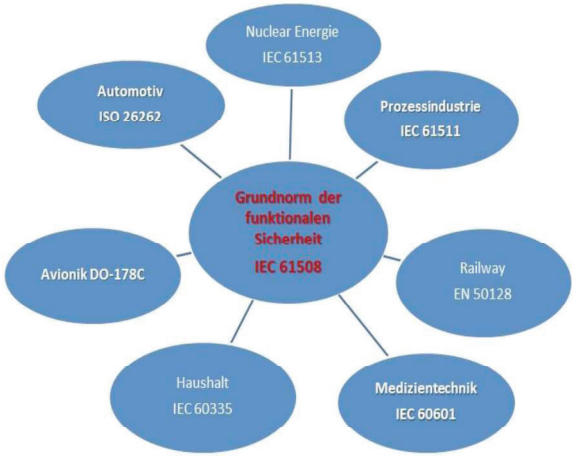


*Abb. 1: Grundaufbau sicherheitskritischer Normen in Diagrammform*

Die Norm IEC 61508 als Grundlage zur Entwicklung elektrischer, elektronischer und pro grammierbarer elektronischer Systeme in der funktionalen Sicherheit, besteht aus sieben Unternormen.
Es existieren verschiedene Arten von Anforderungen, die bei der Entwicklung beachtet werden müssen. Die technischen Anforderungen teilen sich in die verschiedenen Teilnormen auf und verweisen aufeinander. Weiterhin sind verschiedene Bestimmungen zur Dokumentation, zum Management und zur Beurteilung der funktionalen Sicherheit dargelegt. Diese werden alle im Teil 1 der Unternormen definiert.
Der zweite Teil der IEC 61508 geht spezifischer auf die Anforderungen an den Hardwareentwicklungsprozess sicherheitsbezogener E/E/PE-Systeme ein. Dabei legt sie im speziellen die Anforderungen für Tätigkeiten, die während des Entwurfs und der Herstellung anzuwenden sind, und alle notwendigen Informationen zur Ausführung der Installation, Inbetriebnahme und abschließenden Validierung, bezüglich der Sicherheit, des Systems fest.
Zu Beginn des dritten Teils der Normenreihe wird nochmals auf Anforderungen und Ziele bezüglich Dokumentation und Management von E/E/PE-Systemen eingegangen, so wie es in den vorherigen Teilen bereits beschrieben wurde. An dieser Stelle werden jedoch noch zusätzlich Strategien für die Beschaffung, Entwicklung, Integration, Verifikation, Validierung und Modifikation im Rahmen der Anforderungen des SIL der Sicherheitsfunktionen an die Software definiert.
Im Teil 4 der Normenreihe wird das Vokabular aus den restlichen Teilnormen und dessen Bedeutungen ausführlich erläutert und umschrieben.
Zunächst werden unter dieser Teilnorm Informationen zu den zugrunde liegenden Konzepten für Risikominderung und Sicherheitsintegrität erläutert und beschrieben, wie diese im Zusammenhang stehen. Hierfür werden die wichtigsten Begriffe und Methoden erläutert.
Im Teil 6 der Normenreihe wird ausführlich auf die Anwendung der bisherigen Normteile eingegangen, speziell der Teile zwei und drei. Hierzu definiert sie Richtlinien hinsichtlich der zu erfüllenden funktionalen Schritte, detaillierten Forderungen und zeigt einfache Beispielverfahren auf.
Im letzten Teil der Normenreihe sind nochmal alle bisher erwähnten Methoden und Maßnahmen ausführlich mit Ziel, Beschreibung und ggf. Literaturhinweisen erläutert. Die Gesamtstrukturierung aller Verfahren ist in Methoden zur Beherrschung von zufälligen Hardwareausfällen, Vermeidung von systematischen Ausfällen und Erreichung der Sicherheitsintegrität eingeteilt. Weiterhin gibt die Norm einen probabilistischen Ansatz zur Bestimmung der Sicherheitsintegrität von vorentwickelter Software, Maßnahmen zum Entwurf von ASIC´s, Definitionen zu den Software-Lebenszyklusphasen und eine Anleitung zur Entwicklung von sicherheitsbezogener objektorientierter Software wieder.

**Programmier- und Entwicklungsrichtlinien**

Richtlinien zur Programmierung dienen in erster Linie der Reglementierung von Code-Konstrukten, um bekannte und häufige Fehler zu vermeiden. Hierzu gehören auch Definitionen hinsichtlich gewisser Eigenschaften, die beispielsweise der Modularität, Lesbarkeit oder Wartbarkeit dienen. Des Weiteren werden meist gewisse Designregelungen aufgestellt. Diese dienen meist zur Verringerung oder Vermeidung von Abhängigkeiten zwischen verschiedenen Komponenten. Meist sind Richtlinien automatisch überprüfbar, um bei einem Übersetzungsvorgang des Compilers der gewählten Entwicklungsumgebung automatisch die Regelungen überprüfen zu können.
Die meist verbreitetste Richtlinie ist die der „Motor Industrie Software Reliability Association“, kurz MISRA. Wie der Name schon sagt wurde diese Richtlinie von Automobilherstellern und dessen Zulieferern entwickelt und sollte der Vereinheitlichung und Fehlervermeidung zur Qualitätssteigerung bei der Programmcodeerstellung dienen. Es werden verschiedene Ausführungen für die Programmiersprache C und seit einiger Zeit auch für C++ herausgegeben. Die aktuelle Ausgabe des MISRA C Standards beinhaltet 144 Regeln, wovon die meisten vorgeschrieben, einige empfohlen und ein paar veraltet sind. Diese Weiterentwicklung der Richtlinie ist notwendig, da sich Programmiertechniken und -sprachen stetig weiter entwickeln, wodurch sich auch die Standards anpassen müssen. Alle Regeln beinhalten, neben der definierten Regel, auch immer eine Erklärung und oft Beispiele um den Zusammenhang einfacher zu veranschaulichen. Durch ihre allgemein definierten Regeln beschränkt sich der Einsatz der Richtlinie mittlerweile nicht nur auf die Automobilindustrie, sondern wird auch in vielen anderen Industriezweigen angewendet.
Zusätzlich existiert die MISRA SA Richtlinie, bei der es sich um eine Erweiterung des ursprünglichen MISRA Standards handelt. Sie gibt einen tieferen Einblick und detaillierte Informationen hinsichtlich der anzuwendenden Integrität und Sicherheitsanalyse, sowie zum Sicherheitsmanagement und dem Modell des Sicherheitslebenszyklus. Dabei werden nicht, wie zuvor für die MISRA C beschrieben, durch eine Auflistung von Regeln die

Zusammenhänge zur richtigen Programmiertechnik festgelegt, sondern wie in den Normen die Methoden, Bedeutungen und Beziehungen zueinander definiert. Der Unterschied zu den Normen besteht in der programmiersprachennahen Festlegung.
Als LabVIEW eigene Richtlinie dient die LabVIEW Development Guideline. Bei ihr handelt es sich nicht um eine Programmierrichtlinie, wie der MISRA C Standard. Sie erklärt im Wesentlichen den Umgang mit der Programmiersprache LabVIEW. Es sind jedoch auch gleiche Teile in Bezug auf die bisher aufgezeigten Leitlinien enthalten. Wie zum Beispiel Erläuterungen zu Lebenszyklusmodellen, ordnungsgemäße Dokumentierung von Projekten, Qualitäts- und Konfigurationsmanagement und Prototypen-Designplanung. Somit bietet sie eine sinnvolle Grundlage für eine Erweiterung im Sinne der funktionalen Sicherheit.

### LabVIEW Safety durch die Grundnorm

Der Begriff LabVIEW Safety steht für eine Initiative des CAS in Darmstadt mit dem Unternehmen National Instruments und anderen Alliance Partnern. Zukünftig soll er als prägendes Element bei der Verwendung der Entwicklungsumgebung LabVIEW zur Programmierung im High-Assurance System Development dienen.

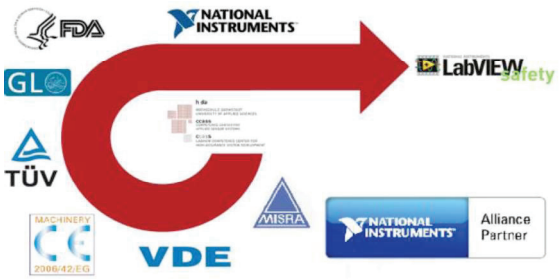


*Abb. 2: LabVIEW Safety Inititiative vom Ursprung, mit CCASS als Hauptbestandteil, bis hin zum Ziel und allen beinhaltenden Kooperationspartnern*

An dieser Stelle soll darauf eingegangen werden, was zu beachten ist, um funktionale Sicherheit zu gewährleisten. Dabei zeigen sich zusätzliche Eignungen für Weiterentwicklungen ins IoT, die durch andere Programmiersprachen nur sehr schwer zu realisieren sind. Wie im Abschnitt zur Grundnorm beschrieben, sind hierfür eine Menge an Regeln und Verfahren einzuhalten. Einige hiervon gehen auf das Grundverständnis, hinsichtlich Projektmanagement und Qualitätssicherung, ein, andere hingegen definieren Methoden mit klar gesetzten Grenzwerten. Alle diese Regelungen lassen sich dabei drei Untergruppen zuordnen, dem Management, der Software und der Hardware. Managementregelungen dienen meist der Qualitätssicherung, Projektübersicht und Nachvollziehbarkeit hinsichtlich Tests und Sicherheitsmerkmalen von Systemen und Projektkonstellationen. Softwareregelungen, hierzu gehören auch Regelungen für Programmiersprachen, definieren den korrekten Umgang und das Einhalten von Richtlinienwerten, um lediglich sicheren Source-Code zu erstellen oder damit ein Fehler in der Software nur ungefährliche Folgen haben kann. Hardwareregelungen geben, neben verschiedenen Verfahren zur Bestimmung der Sicherheitsintegrität und Risikoanalyse von Bauteilgruppen, auch maximal zulässige Grenzwerte für entwickelte Systeme an. Je nach Anwendungsgebiet benötigt ein solches System verschiedene Grenzwerte und Problemansätze, die jedoch alle denselben Grundentwürfen folgen. Dies ist der Grund für die Norm- und Richtlinienentwicklung verschiedener Industriezweige nach ihren eigenen Anforderungen und Bestimmungen.
Wie bereits erwähnt, existiert eine Vielzahl an Normen und Richtlinien, die jeweils unregelmäßig aktualisiert oder neu verfasst werden. Die Notwendigkeit erschließt sich dabei zum einen aus der stetigen Weiterentwicklung von Technologien und Verfahren, zum anderen jedoch auch aus Neuentwicklungen, die neue Denkweisen und Problemstellungen ansprechen. Nicht alle diese Richtlinien und Normen beschreiben anwendungsbezogene Regelungen, da sie sich zwischen Entwicklungsrichtlinien, Anwendungsrichtlinien und rechtlichen Richtlinien differenzieren lassen. Entwicklungs- und Anwendungsrichtlinien verfügen über keine direkten rechtlich anwendbaren Bestimmungen. Sie fungieren vielmehr als Grundlagen für Entwickler, als für rechtliche Zwecke, können zum Nachweis der sicheren Entwicklung des Systems jedoch dafür angewendet werden. Es gibt allerdings auch solche Richtlinien, die einzig und allein die rechtlichen Einzelheiten regeln. Speziell in der Medizintechnik sind diese für die Betriebssicherheit notwendig. Grundsätzlich behandeln sie die gleichen Themenstellungen und verweisen aufeinander.
Um auf den Grundgedanken der LabVIEW Safety Konzept Initiative zurückzukommen, empfiehlt sich somit die Erstellung einer eigenen Guideline. Einem Leitfaden, der die richtige Verwendung von LabVIEW aufzeigt, um alle Kriterien der funktionalen Sicherheit einzuhalten. Hierfür werden in den nächsten Abschnitten die erforderlichen Aspekte, bezogen auf die Normungen, definiert. Einige notwendige Bestandteile zur Sicherheitskonformität sind bereits in LabVIEW definiert, andere benötigen eine Überarbeitung. Das Grundproblem liegt außerdem weniger in der Machbarkeit, sondern in der Regel bei der

Referenzierbarkeit. Um sicherheitsrelevante Software zu erstellen, muss vor der Markteinführung des Produkts sichergestellt sein, dass alle relevanten Richtlinien und Normen ordnungsgemäß angewendet wurden. Durch Überlappungen der Inhalte und der hohen Anzahl darauf zugeschnittener Anwendungsrichtlinien, ist dies in etablierten Systementwürfen, wie beispielsweise für speicherprogrammierbare Steuerungen, recht einfach möglich. Alle in der Software zur Programmierung von SPS-Systemen enthaltenen Elemente wurden durch die verschiedenen Prüfeinrichtungen der einzelnen Länder überprüft. Dadurch ist jeder Entwickler eines solchen Systems, hinsichtlich der Entwicklungsumgebung, abgesichert. Dies ist somit ein wesentlicher Bestandteil der Gesamtanalyse.
Nachdem die Entwicklungs- und Funktionsabläufe von LabVIEW durch eine Prüfeinrichtung verifiziert und validiert sind, steht, bezogen auf die Softwarevoraussetzungen, der funktionalen sicheren Programmierung nichts mehr im Wege. Bezogen auf Hardwarevoraussetzungen ist ein ähnliches Prozedere einzuhalten. Für die Einhaltung der Managementvoraussetzungen benötigt ein Projekt in erster Linie keine Automatisierung. Alle einzuhaltenden Aspekte sind durch reine Dokumentation und analoges Projektmanagement anwendbar. Jedoch ist diese Handhabung heutzutage nicht mehr zeitgemäß. Die Anwendung von Managementwerkzeugen erleichtert nicht nur die Nachvollziehbarkeit der Dokumentation, sondern auch die Anwendung von Requirement- und Diagnosetests oder das Handling von Mehrfachzugriffen auf Dateien.

**Safety durch MISRA in neuer Guideline**

Der Begriff einer Richtlinie ist sehr allgemein und weit umfassend, allerdings verfolgen alle Richtlinien den gleichen Zweck, dem Entwickler zu ermöglichen, seine Projekte und entwickelten Produkte einfach gesetzes- und sicherheitskonform zu gestalten, sowie zu verifizieren.
Bisher existieren schon viele vereinzelte Richtlinien für die richtige Vorgehensweise in LabVIEW. Allerdings handelt es sich zumeist um Programmierrichtlinien, die lediglich auf einen Punkt des Programmdesigns ausgelegt sind. Außerdem ist keine von ihnen für sicherheitskritische Anwendungen ausgelegt. Somit stellt sich in diesem Abschnitt die Frage, welche Regelungen benötigt werden, um sicherheitsbezogene Software zu designen.
MISRA-C und -C++ Richtlinien bestehen aus einer Auflistung von Regeln zum sicheren und einheitlichen Programmieren in den Programmiersprachen. Der Zweck besteht in der einfachen Verifizierbarkeit gegenüber Zertifizierungsstellen. Nach dieser Richtlinie erstellte Projekte erfüllen somit alle rechtlichen Regelungen.
Ein großes Problem für LabVIEW bildet noch immer die Akzeptanzbildung. Viele Unternehmen und Ingenieure vertrauen einer so neuen, grafisch basierenden Sprache nicht. Trotz des hohen Einsatzes in Prüf- und Testapplikationen wird ihr sicherheitskritische Implementierung nicht zugetraut. Dies liegt sicher auch an dem einfachen intuitiven Umgang mit der Software. Dies erhöht in den Augen vieler das Gefahrenpotenzial falscher Anwendung.
Nimmt man beispielsweise ein VI in LabVIEW. Es kann in jedem Fall eigenständig agieren, solange keine Abhängigkeiten bestehen. Somit lassen sich in diesem einen VI Tausende an SubVI´s implementieren, dessen Verbindungen quer hindurch übereinander gelegt werden können. Ab einer gewissen Komplexität lässt sich der Programmcode keineswegs mehr lesen oder von einem fremden Entwickler verstehen. Solche unlesbaren Konstrukte sind in anderen Sprachen auch möglich, indem wahllos und unkommentiert von Funktion zu Funktion mit Pointern und Aufrufen gesprungen wird, ohne dies zu begründen. Um solche durch unstrukturierte Programmentwürfe entstehenden Probleme zu unterbinden existieren Regelungen gegen unübersichtliche LabVIEW Programme, genauso wie MISRA es für C oder C++ darstellt.
Die LabVIEW Development Guideline stellt das Pendant zur MISRA. Ein direkter Vergleich beider ist jedoch nur schwer möglich, da der Aufbau dieser weniger auf aufgegliederten Regeln basiert, sondern auf einem Leitfaden zum Entwurf und zur Realisierung eines Projektes in LabVIEW. Es bestehen somit zwei Möglichkeiten der Aufbereitung. Die erste Methode wäre ein komplettes Neudesign der LabVIEW Guideline, um dem Format einer MISRA Richtlinie zu entsprechen, wodurch eine einfachere Überprüfung der gesetzten Bestimmungen möglich wird. Allerdings beinhaltet dies einen enormen Aufwand und infolge der großen Unterschiede grafischer und textbasierter Programmiersprachen sind einige notwendigen Regelungen schwer zu beschreiben oder würden eventuell, aufgrund der entstehenden Größe, die Guideline schlechter lesbar machen. Die andere Möglichkeit der Aufbereitung wäre eine Ergänzung der fehlenden Reglementierungen. Dadurch bliebe der bestehende LabVIEW Charakter erhalten und es müssten lediglich Erweiterungen und Eingrenzungen gesetzt werden.
Beide Methoden zur Erstellung einer LabVIEW Guideline, die die Softwarequalität in der Entwicklung steigert und dem etablierten Niveau entspricht, benötigen zunächst eine Analyse der aktuellen Regeln. Hierfür wurde eine Analyse durchgeführt, in der alle, in

MISRA-C enthaltenen Regeln, aufgelistet und gegenüber der Konformität in LabVIEW überprüft wurden.
Die erstellte Tabelle teilt sich in vier Spalten. Die ersten zwei zeigen die einzelnen Regeln und ob sie erforderlich sind oder nur angeraten werden. In Spalte drei und vier ermittelt sich die Konformität mit LabVIEW. Ein Farbcode erstellt eine erste Eingliederung in Grün, Regelung in LabVIEW bereits vorhanden, Gelb, Regelung muss angepasst oder realisiert werden, und Rot, die Regel ist in LabVIEW nicht nötig.

*Tab. 1: Gegenüberstellung MISRA vs. LabVIEW*

| Regeln aus der MISRA-C:2004 | | Anwendbarkeit auf LabVIEW | Erg. |
|---|---|---|---|
| R | All usage of implementation-defined behaviour shall be documented. | Ausführlich in der bisherigen Guideline beschrieben. | |
| R | The character set and the corresponding encoding shall be documented. | Ausführlich in der bisherigen Guideline beschrieben. | |
| A | The implementation of integer division in the chosen compiler should be determined, documented and taken into account. | Der LabVIEW Compiler wurde noch nicht im Rahmen der funktionalen Sicherheit eindeutig überprüft | |
| R | All uses of the #pragma directive shall be documented and explained. | Nicht textbasiert. Kann in LabVIEW nicht genutzt werden. | |

Warum Regeln in Rot nicht definiert werden müssen, kann viele Gründe habe. Meistens jedoch begründet es sich in den Unterschieden zu grafischen Sprachen, da dort falsche Schreibweisen von Befehlen schlicht nicht möglich sein können. Eine weitere Begründung liegt in dem nicht vorhanden sein gewisser Aufruftypen und Operationen. Die Begründung grün markierter Regeln kann auch an verschiedenen Stellen liegen. Einige Regelungen sind schon in der LabVIEW Development Guideline enthalten und andere benötigen keine Definition, da in LabVIEW aufgrund des Bausteinprinzips keine andere Möglichkeit existiert, als konform zu handeln. Alle Gelb markierten Regeln sind in einer neuen Richtlinie mit aufzunehmen und einzuarbeiten.

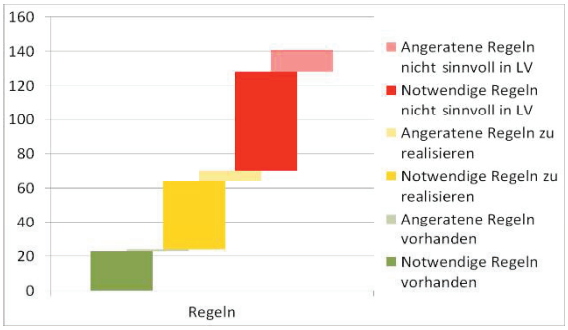


*Abb. 3: Regeln der MISRA Guideline im Vergleich zu LabVIEW und seiner Development Guideline*

Zur Auswertung der analysierten Ergebnisse wurde ein Diagramm erstellt, einzusehen in Abbildung 3, um den Sachverhalt zu veranschaulichen. Insgesamt definiert die MISRA-C 141 Regeln, wovon lediglich 121 erforderlich sind. Bei den restlichen 20 handelt es sich um Regeln, dessen Anwendung empfohlen wird. Diese Regeln sind zwar im Diagramm mit dargestellt, dienen jedoch nur der Veranschaulichung und benötigen nicht unbedingt eine Definition in der neuen Guideline. Dies gilt generell auch für alle grün und rot markierten Regeln. Hierbei liegt der Grund allerdings bei der Verfügbarkeit, wie zuvor schon beschrieben. Einige sind schon in der LabVIEW Guideline beschrieben, andere können nicht angewendet werden. Mittels des Farbcodes lassen sich die neu zu implementierenden Regeln somit auf knapp ein Drittel aller verringern.
Bei den vierzig erforderlichen, noch zu realisierenden Regeln, handelt es sich grundsätzlich um die reine Definition der richtigen Anwendung LabVIEW´s. Dass LabVIEW alle diese Regelungen einhalten kann, steht außer Frage, da die Funktionalitäten die gleichen wie in Hochsprachen sind. Durch den speziellen Aufbau der Programmiersprache lassen sich sogar einige gelb markierte Regeln verifizieren,
ohne in der Guideline definiert zu werden. Beispielsweise reglementiert die MISRA Compilerkonfigurationen, die in LabVIEW evtl. nicht nötig sind, da nur die eigenen Compiler verwendet werden können und diese bereits durch die Etablierung bei Zertifizierungsstellen verifiziert sein könnten. Um jegliches Fehlerpotenzial und Komplikationen zu verhindern, ist eine zusätzliche Definition allerdings sehr sinnvoll.
Ein nächster Schritt für die neue Guideline würde somit darin bestehen, die notwendigen Regeln zu verwenden, umzuformulieren und in die LabVIEW Guideline einzuarbeiten. Hierbei ist jedoch darauf zu achten, alle Regeln für eine grafische Programmiersprache zu formulieren. Die MISRA beschreibt die meisten Regeln mit textbasierten Beispielen. Für LabVIEW

bedeutet dies eine grafisch basierte Anleitung zur richtigen Verwendung, eine Leitlinie.

**Aufwand und Fazit**

In der Entwicklungsumgebung von NI LabVIEW können bereits viele Grundfunktionalitäten zur sicherheitsbezogenen Programmierung von E/E/PE-Systemen nachgewiesen werden. Die Programmiersprache besitzt eine strukturierte und leicht zu analysierende Benutzeroberfläche, die durch ihre grafische Ausführung, ähnlich einiger SPS-Sprachen, besonders geeignet ist.
Grundlegend teilt sich der Gesamtaufwand zur Erstellung sicherheitsbezogener Software in zwei Unterkategorien, die voneinander abhängen. Hierbei handelt es sich zum Einen um die Erstellung von Richtlinien durch National Instruments und unterstützender Unternehmen, die später dazu beitragen, zum Anderen eine einfache Zertifizierung mit LabVIEW erstellter sicherheitsbezogener Software zu garantieren. Grundvoraussetzung hierzu ist zunächst die Zertifizierung der Entwicklungsumgebung, um nachzuweisen, dass alle nötigen Vorgaben eingehalten werden. Hierzu wurde die bereits bewährte MISRA Guideline mit der von National Instruments bereitgestellten LabVIEW Development Guideline verglichen.
Um auf gesetzlicher Ebene vorauszusetzen, dass alle Vorgaben in der bisherigen Entwicklungsumgebung eingehalten werden, bietet sich eine Vorabzertifizierung bestehender Funktionalitäten an. Hierbei ist besonderer Wert auf die GrundVI-Ebene zu legen, die alle Bestandteile komplexerer Funktionalitäten enthalten. Es empfiehlt sich, eine Darstellung derer gegenüber Zertifizierungsstellen, zur konkreten Analyse.
Der in LabVIEW enthaltene Compiler kann in seiner Grundstruktur beibehalten werden. Hierbei liegt das Hauptproblem mehr in der Vertrauensherstellung. Compiler textbasierter Sprachen besitzen eine höhere Betriebsbewährtheit als die grafischer. Außerdem gewähren sie eine genauere Einsicht bei der Umwandlung in Maschinencode. Um hierbei eine Zertifizierung LabVIEW´s zu erlangen, bietet sich ein Vergleich mit Compilern von C-Code an. Auch diese besitzt Algorithmen zum Optimieren des Programmcodes, die zur sicherheitskritischen Implementierung abgeschaltet werden müssen, um genaue Laufzeitbedingungen festlegen zu können. Mittels Anpassung solcher, teilweise zufälliger, Algorithmenauswahlkriterien in LabVIEW kann der Compiler auch für Echtzeitprogrammier- und in FPGA´s alle nötigen Kriterien besitzen.
Im strukturellen Aufbau von sicherheitskritischen Systemen sind viele Kriterien zu beachten. Die Programmierung erfolgt in zwei verschiedenen und klar getrennten Teilen. In dem Einen sind die funktionalen Anforderungen des Systems implementiert und in dem Anderen die sicherheitsbezogenen Sicherheitsfunktionalitäten, die das System überwachen. Unsichere Codekonstrukte, die nicht zu eindeutigen Ergebnissen führen, wie Zufallsalgorithmen, dürfen im sicherheitsbezogenen Teil der Programmierung nicht verwendet werden. Es bietet sich somit eine reduzierte VI-Bibliothek für diesen Teil an, um solche grundsätzlichen Fehler zu vermeiden.
Auch an die Programmiertechnik sind hohe Maßstäbe an Übersichtlichkeit und Nachvollziehbarkeit gesetzt. Diese werden bereits durch die bisherige LabVIEW Development Guideline sehr detailliert eingehalten. Es fehlen jedoch einige Einschränkungen für Interrupts und Rekursionen, Ausschlusskriterien zur Verwendung dynamischer Variablen und statische Verifikationsmethoden, die im sicherheitsbezogenen Teil der Programmierung benötigt werden.
Eine neue Richtlinie sollte die, in den letzten zwei Abschnitten beschriebenen, Anforderungen genau darlegen. Außerdem ist es anzuraten, dass National Instruments oder ein unterstützendes Unternehmen vordefinierte Sicherheitsfunktionalitäten erstellt, diese in einer VI-Bibliothek zusammenfasst, zertifizieren lässt und in die neue Richtlinie einarbeitet. Dies bietet den Vorteil einer einfachen Verwendung und Zertifizierung damit erstellter Systeme. Jede VI-Sicherheitsfunktion benötigt hierfür ein maximal erreichbares Sicherheitsintegritätslevel.
Abschließend muss eine Verfahrensweise angegeben werden, die darlegt, wie in einen sicherheitsbezogenen Projekt vorgegangen werden soll, um alle Sicherheitsbestimmungen normgerecht einzuhalten und abschließend eine einfache Zertifizierung erstellter Systeme zu erhalten. Es bietet sich der Aufbau von einer Schrittkettenfunktionalität, bspw. mit Zustandsdiagrammstrukturen, an, die detailliert den Ablauf und die Bedingungen regeln.
Durch Darlegung und Einschränkung in einer neuen LabVIEW Richtlinie und dessen Absegnen durch eine Zertifizierungsstelle, kann sich National Instruments, bezogen auf die Softwarebestimmungen in den Normen, rechtlich absichern und bildet zusätzlich einen Anreiz für Entwickler sicherheitskritischer Systeme zur Wahl von LabVIEW als Entwicklungsumgebung.

**Zukunftsperspektiven**

Da sich diese Analyse aus Gründen der Vielfalt und Komplexität sehr allgemein mit dem Thema der Erstellung von sicherheitsbezogenen Systemen mit LabVIEW befasst, sind weiterführende Erarbeitungen, hinsichtlich der Soft- und Hardware, nötig. An dieser Stelle

handelt es sich lediglich um eine erste Anleitung, auf welche Faktoren, Bedingungen und Anforderungen besonderer Wert gelegt werden muss, damit eine einfache Zertifizierung aller Komponenten möglich wird.
Im Softwarebereich können, mittels der in dieser Analyse aufgezeigten Vorgehensweisen, sicherheitsbezogene Systeme implementiert werden, auch ohne Änderung und Anpassung der Entwicklungsumgebung. Dies ist jedoch wenig reizvoll für Unternehmen, da alle Zertifizierungen zur Zulassung für den Markt eines Produkts somit nur schwer zu erhalten sind. Eine Weiterentwicklung der in dieser Analyse aufgezeigten Konzeption, Anpassung der Programmierumgebung und Vorzertifizierung würde dies wesentlich vereinfachen und zusätzlichen Anreiz zur Nutzung schaffen.
Durch Entwicklung einer neuen sicherheitsbezogenen LabVIEW Richtlinie, die genaue Vorgehensweisen zur Entwicklung vorgibt, kann das Fehlerpotenzial, gegenüber textbasierten Sprachen, erheblich gesenkt werden, da im modularen bausteinbasierten Aufbau keine Schreibfehler oder nicht ersichtliche Verbindungen möglich sind, ähnlich der grafischen SPS-Sprachen.
Relevant bei einer Etablierung ist die Verteilung sicherheitsbezogener Systeme in der Industrie. Nachdem sich ein solches LabVIEW-basierendes System auf dem Markt etabliert und die Absätze steigen, sinken auch entsprechende Stückpreise für LabVIEW-Systementwicklungen. Somit steigt auch die Attraktivität zum Einsatz in einfachen sicherheitsbezogenen Anwendungsbereichen.
Zukünftig kann die Mitgestaltung solcher Systeme nicht nur von einem kleinen Entwicklerteam abhängen. Durch das grafische Design LabVIEW´s und die Verwendung als Programmiersprache der 4. Generation im Bereich des IoT kann mehr Teammitgliedern bis zu einem gewissen Punkt Einblick in die Softwaregestaltung geben werden, um gewisse Funktionalitäten und den Gesamtzusammenhang zu verstehen. Somit kann durch wesentlich einfachere Mittel als in anderen Programmiersprachen ein sehr hohes Knowhow zur Fehlererkennung, -vermeidung und Lösungsfindung erzeugt werden. Dies bedeutet im Wesentlichen keine Vereinfachung der Strukturen, da diese in LabVIEW genauso hoch sind, wie in anderen Sprachen, sondern die Schaffung einer Plattform für das Internet der Dinge.

## Literaturnachweis

[1] Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme – Teil 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 (IEC 61508:2010)
[2] http://www.misra-cpp.com/, 07.07.2015
[3] http://www.ni.com/pdf/manuals/321393d.pdf, 08.07.2015
[4] http://www.misra.org.uk/Activities/MISRASafetyAnalysis/tabid/92/Default.aspx, 08.07.2015
[5] https://en.wikipedia.org/wiki/LabVIEW, 02.08.2015
[6] http://www.tuev-sued-stiftung.de/uploads/images/1339742016358384280324/funktionalesicherheit-61511.pdf, 04.09.2015