

02/2012

*Software-Sicherheit*

## **Die drei Gesichter der Sicherheit von Software-Systemen**

**Der Aspekt Sicherheit von Software-Systemen ist mehr als nur die Betriebssicherheit. Im ersten Teil dieser Serie zum Thema Software-Sicherheit schauen wir auf den letzten ESE Kongress und die dort gemachten Aussagen der Referenten zum Thema Betriebssicherheit. Dabei stellen wir Ihnen drei besonders wichtige Regeln vor.**

Elektroautos sind derzeit in aller Munde, was sich auch in der näheren Zukunft nicht ändern wird. Woran aber kaum jemand denkt: Ein Elektroauto bezieht seine Energie aus einer gezähmten Brandbombe.

Auch sonst handelt es sich um ein System mit erheblichem elektrischen Gefahrenpotenzial. Die Betriebssicherheit (Safety) wird vor diesem Hintergrund vor große und neue Herausforderungen gestellt. Das gilt im besonderen Maß für die Software der Steuergeräte dieser Fahrzeuge.

### **Die Sicherheit nach der Norm IEC 61508**

Der Aspekt Sicherheit reicht bei Zukunftstechnologien dieser Tragweite weit über die Betriebssicherheit hinaus. Neben Safety spielen Security, also der Schutz vor unbefugten Zugriffen, beispielsweise durch Hacker, und die Zukunftssicherheit eine entscheidende Rolle.

Beim Embedded Software Engineering Kongress 2011 wurden verschiedene Aspekte um diesen Themenkomplex beleuchtet. Mit diesem ersten Beitrag fassen wir in einer Rückschau wichtige Aussagen der Referenten rund um die Betriebssicherheit zusammen. Sicherheit hat viele Gesichter. Laut Definition in der Norm IEC 61508 ist Sicherheit die Bezeichnung eines Zustands, der frei von unvermeidbaren Risiken der Beeinträchtigung und mithin als gefahrenfrei anzusehen ist.

### **Vermeintlich sichere Systeme leider oft unsicher oder gefährlich**

Doch wie zeigt sich, ob beispielsweise ein technisches System, das auf Embedded Software beruht, hinreichend sicher ist? Immerhin erweisen sich vermeintlich sichere Systeme leider immer wieder als unsicher oder gar gefährlich. Die Aussage über das Elektroauto wird übrigens sinngemäß einem Chefentwickler aus der Automobilindustrie zugesprochen.

Doch das Potenzial an Gefährdungen ist schwer zu übersehen. Zumal wenn Gefahrenquellen vorab noch gar nicht bekannt sind oder wenn nicht klar ist, in welchen Situationen ein System sich zu bewähren hat. Weil man im Vorfeld kaum sagen kann, wo später im Betrieb eines Systems die Quellen einer Gefährdung oder Fehlfunktion liegen werden, ist es von größter Bedeutung, nach dem Stand der Wissenschaft und Technik zu entwickeln. Die notwendige Sicherheit ist mit den derzeit verfügbaren Mitteln zu realisieren.

### **Drei Regeln, die Softwarequalität zu verbessern**

Mit welchen Möglichkeiten lässt sich die Betriebssicherheit von Software-Systemen schon in der Entwicklung verbessern? Im Grunde besteht die Anforderung darin, gewisse Regeln umzusetzen. Drei wichtige Regeln hat Karl Nieratschker, freiberuflicher Trainer bei MicroConsult in seinem Vortrag auf dem ESE Kongress herausgehoben.

Es gibt verschiedene Möglichkeiten, um die Softwarequalität zu verbessern. Einerseits bestünde ein wichtiger Teil der Kunst darin, sich zu beschränken. Meint nichts anderes als auf gefährliche Konstrukte entweder zu verzichten oder sie nicht zuzulassen.

### **Vereinfachte Syntax als Lösung für weniger Fehler?**

Zum anderen leiste eine vereinfachte Syntax einen wertvollen Beitrag. Dadurch lassen sich Programme leichter schreiben und lesen. Ein einfaches und wirksames Mittel, dessen Wichtigkeit häufig unterschätzt wird. Das Potenzial der Syntaxvereinfachung besteht darin, Fehler durch gute Übersicht gar nicht erst entstehen zu lassen. Zu guter Letzt sollten zusammengehörende Dinge bei der Entwicklung zusammengefasst werden. Das sorgt für ein leichteres Verständnis und besseren Überblick.

Der Vortrag trug den Titel "Mehr Sicherheit und Komfort in C-Anwendungen". Vorgenannte Regeln gelten ganz allgemein in der Softwareentwicklung, die objektorientierten Programmiersprachen sind bereits mit Bordmitteln ausgestattet. So unterstützt C++ beispielsweise die Umsetzung der Regel, dass zusammengehörende Dinge bei der Entwicklung zusammengefasst werden sollen, Aufzählungstypen können beispielsweise innerhalb einer Strukturdefinition angelegt werden, ohne dass dies Speicherplatz kostet.

Aber C++ bietet ganz pragmatische Verbesserungen in Bezug auf die Betriebssicherheit der Software im praktischen Einsatz. So können Pointer, die häufig Fehler in der Codeerstellung mit sich bringen, oft durch Referenzen ersetzt werden, was zur Syntaxvereinfachung beiträgt und sogar Sicherheitsabfragen überflüssig machen kann.

### **Die ISO 26262 ist nur unscharf definiert**

Mit der im November 2011 veröffentlichten Norm ISO 26262 wurden die Vorschriften für sicherheitsrelevante elektrische/elektronische Systeme in Kraftfahrzeugen auf Basis des heutigen Stands der Technik und der dazugehörigen Standards festgeschrieben. Die funktionale Sicherheit eines Systems soll immer mit elektrischen/elektronischen Komponenten im Kraftfahrzeug gewährleistet werden.

Die Norm bezieht sich speziell auf Fahrzeuge mit einem Gesamtgewicht bis 3,5 t, denn hier sind Embedded-Anwendungen sehr stark verbreitet. Leider bleibt die Norm in der Beschreibung ihrer Anwendung recht unscharf. Wichtig ist daher, dass ein geeignetes Handwerkszeug für die Umsetzung der Forderungen aus der Norm und für den Nachweis existiert und den Anforderungen tatsächlich genüge getan wurde.

### **Alles technisch Machbare sollte umgesetzt werden**

Ein zentraler Punkt der Norm ist der Nachweis, alles technisch Machbare umzusetzen. Ein Punkt, der vor dem Hintergrund von Haftungsfragen gar nicht ernst genug genommen werden kann. Stefan Kriso von Robert Bosch erklärte in dem Zusammenhang, dass die Qualifizierung der verwendeten Softwaretools ein ganz wichtiger Punkt sei. Es bestehe die Möglichkeit, dass durch die Tools Fehler erzeugt werden, derer sich der Programmierer oder Softwareentwickler gar nicht bewusst ist – er bemerkt diese Fehler schlicht gar nicht. Dazu müsse nach den Worten von Kriso "(...) geklärt werden, mit welcher Wahrscheinlichkeit in einem späteren Schritt im Entwicklungsprozess fehlerhafte Inhalte, die vom Werkzeug ausgegeben werden, zum Beispiel durch Reviews oder Tests entdeckt werden können."

### **Klare Regeln und Grenzen für die Softwareanalyse**

Die Tatsache, dass alle Software und alles andere auch immer nur in der "aktuellen Gegenwart" erstellt wird, die Anwendung selbst und die möglichen Probleme, die sich ergeben können, aber ausschließlich in einer heute unbekannt Zukunft mit unbekannt Einflussgrößen zum Einsatz kommen, lässt die Herausforderung klar werden.

In ihrem Vortrag stellten Dr. Jens Lisner und Dr. Thomas Wenzel vom TÜV Nord heraus, dass die Methoden der Fehlerbaumanalyse (FTA) und der Failure Mode and Effects Analysis (FMEA), die beide nicht ausdrücklich für die Entwicklung sicherer Software ersonnen wurden, alternativ verwendet werden können, um die Qualität in puncto Sicherheit zu steigern.

Weil die Systeme hochgradig vernetzt und komplex sind und künftige Fehlerquellen nicht bekannt sind, müssen klare Regeln und Grenzen für die Softwareanalyse aufgestellt werden. Je komplexer die Systeme sind, desto mehr steigt der Aufwand für eine verlässliche Analyse.

### **Zwei Wege, Sicherheitsnormen umzusetzen**

Ulrich Becker von Method Park Software hat in seinem Vortrag "Sicherheitsnorm gelesen – und dann?" erläutert, wie die Forderungen der Norm ISO 26262 tatsächlich in der Softwareentwicklung umgesetzt werden können. Der Weg besteht aus zwei Stufen: Zunächst sollen die Sicherheitsziele definiert und daraus werden die Anforderungen an die Betriebssicherheit abgeleitet.

Diese Anforderungen wiederum bestimmen einerseits die Gestaltung der Softwarearchitektur und legen zum anderen externe Maßnahmen, die System Design Analysen, fest. Diese helfen dabei, systemische und zufällige Fehler zu vermeiden. Systemische Fehler sind in einem System bereits vorhanden und treten unter bestimmten Umständen auf, zufällige Fehler dagegen treten von außen her auf und zeigen sich meist erst im Betrieb, sie werden oft durch äußere Einflüsse ausgelöst.

### **Sicherheit durch umgesetzte Architekturmerkmale und eingehaltene Regeln**

Die technischen Systeme, um die es heute beispielsweise in Fahrzeugen geht, sind hochkomplexe Gebilde, gewöhnlich bestehend aus Soft- und Hardware sowie mechanischen und elektromechanischen Komponenten. Jedes davon in sich selbst hochgradig kompliziert und dabei miteinander nahezu untrennbar verzahnt, so dass ein planvolles Vorgehen bei der Produktentwicklung von Anfang an unbedingt erforderlich ist – ansonsten wird eine angemessene Sicherheit zum Glücksspiel.

Hier helfen gut geschulte Leute mit einem hohen Bewusstsein für ihre Verantwortung. Peter Siwon, Business Development Manager beim Schulungs- und Consulting-Spezialisten MicroConsult, führt dazu aus: "Projektteams müssen frühzeitig dafür sensibilisiert werden, dass das Schreiben von Code im Grunde das Allerwenigste ist. Sicherheit kommt dadurch zustande, dass Architekturmerkmale umgesetzt und Regeln eingehalten werden, die diese Sicherheit unterstützen."

Eine Aussage, die gestützt wird von Frank Listing, Dozent für Softwareengineering im gleichen Unternehmen. Er sieht Schwachstellen in der Ausbildung als großes Problem und bemerkt dazu: Seiner Meinung sind Ingenieure und Elektrotechniker, Maschinenbauer oder Physiker fachlich gut ausgebildet, aber bei der Software-Entwicklung gibt es oft Schwachstellen.

### **Softwarearchitektur stets aktuell dokumentieren**

Von entscheidender Bedeutung ist die Planung einer geeigneten Softwarearchitektur, wie Stephan Drozniak von B. Braun Melsungen und Thomas Eisenbarth von Axivion erläutern. Eine gut und stets aktuell dokumentierte Softwarearchitektur gebe dem Projektteam die Möglichkeit, während des gesamten Entwicklungsprojekts den Überblick zu behalten.

Außerdem ermögliche sie einem zunächst projektfremden Entwickler, in kurzer Zeit sowohl Ziele als auch Inhalte eines Projektes oder Produktes zu erlernen und zu verstehen. Die Softwarearchitektur gibt ein Bild über den inneren Aufbau und die Funktionsweise eines Softwareproduktes, ohne dabei zu tief ins Detail zu gehen. Alle Komponenten der Software sind hierbei hierarchisch oder strukturiert dargestellt, inklusive der Beziehungen der einzelnen Komponenten zueinander.

**Gut durchdacht und gut dokumentiert zu mehr Softwaresicherheit**

Insofern ist die Aussage gut nachvollziehbar, dass ein großer Teil des Weges hin zu sicherer Software in einer durchdachten und gut dokumentierten Softwarearchitektur liegt. Wenn dieses Prinzip einmal verinnerlicht ist und angewendet wird, ist man ein gutes Stück vorangekommen.

Nach den Worten von Dr. Jürgen Mottok, Scientific Head of Laboratory for Secure Systems in Regensburg, versteht sich die Betriebssicherheit eines technischen Systems als die Reduktion des Risikos auf ein wirtschaftlich vertretbares Maß. Damit verbleiben tolerierbare Restfehler in den technischen Systemen. Eine geeignete Fehlerkennung und -reaktion müssen umgesetzt werden.

**Softwarequalität ist erlernbar**

Gut ausgebildete Entwickler müssen für die Thematik Sicherheit angemessen sensibilisiert werden. Qualität lässt sich nicht in ein Softwaresystem hinein-testen.

Programmierregeln, Qualitätsstandards, Forderungen aus Normen und ihre Umsetzung, Wahl und Anwendung der Programmiersprache und der Softwarearchitektur – alles samt erlernbare Dinge, die sich jeder in einem Lernprozess aneignen müssen. Entscheidend ist, wie sich der Lernprozess gestalten lässt, damit er schnell und mit geringem Risiko zum Ziel führt.

**Autor:**

Peter Siwon ist Business Development Manager beim Münchner Unternehmen MicroConsult, Trainingsspezialist für Embedded Software Engineering.

Alexander Sedlak ist als freier Autor tätig.

**MicroConsult GmbH:**

Training, Coaching und Beratung für Software- und Hardwareentwicklung in der Industrie.

[www.microconsult.de](http://www.microconsult.de)