Trend Guide

Embedded Safety & Security

MICRO CONSULT

Training, Coaching, Consulting

Inhalt & Vorwort

Inhalt

Editorial

Sicherheit hat viele Gesichter

Wie geht es Ihnen mit dem Thema Sicherheit? Mich wundert es beispielsweise, dass Embedded-Systeme trotz der teilweise schwer vorstellbaren Komplexität überhaupt funktionieren. Allein, dass Millionen von verschalteten Transistoren in einem Mikrocontroller durch eine Software von tausenden lines of code so gesteuert werden, dass dieses System (meist) das tut, was es soll, ist erstaunlich. Es ist umso erstaunlicher, wenn man die Vielzahl von technischen, fachlichen und psychologischen Fallen kennt, die den Entwicklern solcher Systeme ständig gestellt werden. Dann sollen Produkte ja nicht nur irgendwie laufen, sondern auch betriebssicher sein. Wer glaubt, damit ist es getan, ist auf dem Holzweg. Denn die Welt des Internets und der Apps stellt hohe Herausforderungen an den Schutz vor Hackern, Viren und Würmern. Wer seine Produkthaftungsrisiken klein halten will, muss den Nachweis führen, dass die geforderten Sicherheitsmerkmale dem Stand von Wissenschaft und Technik entsprechen. Der Funktionstest am Ende des Entwicklungsprozesses alleine kann diesen Ansprüchen nicht mehr gerecht werden und ist zudem unwirtschaftlich. Wer heute Systeme rechtzeitig mit den geforderten Sicherheitsmerkmalen zu einem wettbewerbsfähigen Preis auf den Markt bringt, denkt weit über den Test hinaus. Qualitätssicherung wird nicht nur zum integralen Bestandteil des gesamten Entwicklungsprozesses, sondern auch des Personalmanagements und des Produktlebenszyklus. Sicherheit hat viele Gesichter. Die Lektüre dieses Trend Guides wird Ihnen dazu zahlreiche Anregungen geben.

Mit freundlichen Grüßen

Peter Siwon

1) STANDARDS UND QM

Alles wird besser

Sicherheit – was ist das? Sicherheit – was ist das, und von welcher Art von Sicherheit sprechen wir überhaupt? Zum Glück haben sich mittlerweile auch im deutschen Sprachraum die Begrifflichkeiten funktionale Sicherheit (Safety) und Security durchgesetzt. Laut Definition in der Norm IEC 61508 ist funktionale Sicherheit die Bezeichnung eines Zustands, der frei von unvertretbaren Risiken der Beeinträchtigung und mithin als gefahrenfrei anzusehen ist. Security umfasst den Bereich von potentiellen Angriffen auf ein System von außen, was wiederum Einfluss auf die funktionale Sicherheit haben kann. Doch wie zeigt sich, ob ein Embedded-Softwaresystem hinreichend sicher ist? Immerhin erweisen sich vermeintlich sichere Systeme leider immer wieder als unsicher oder gar gefährlich. Das Gefahrenpotenzial ist groß, im privaten wie auch im geschäftlichen, also professionellen Bereich. Zumal wenn Gefahrenquellen noch gar nicht bekannt sind und mithin nicht klar ist, in welchen Situationen ein System sich zu bewähren hat.

Forderung: Den Stand der Wissenschaft und Technik beherzigen

Weil man im Vorfeld kaum sagen kann, wo später im Betrieb eines innovativen Systems die Quellen einer Gefährdung oder Fehlfunktion liegen werden, ist es von größter Bedeutung, nach dem Stand der Wissenschaft und Technik zu entwickeln, um dem Vorwurf der Fahrlässigkeit entgegenzuwirken. Und das bedeutet nicht mehr und nicht weniger, als die notwendige Sicherheit mit den derzeit verfügbaren und vertretbaren Mitteln zu realisieren.

Immanente Verbesserung der Sicherheit von Software – 3 Regeln

Welche Möglichkeiten bestehen nun ganz generell, um die Betriebssicherheit von Software-Systemen schon in der Entwicklung zu verbessern? Im Grunde besteht die Anforderung darin, gewisse Regeln beim Programmieren einzuhalten, um Fehler (systematische) während der Entwicklung zu vermeiden. Es gibt verschiedene Möglichkeiten, um die Softwarequalität zu verbessern:

- ➔ Erstens besteht ein wichtiger Teil der Kunst darin, sich zu beschränken, was nichts anderes bedeutet, als auf gefährliche Konstrukte entweder zu verzichten oder sie nicht zuzulassen.
- ➔ Zweitens leistet auch eine Vereinfachung der Syntax einen wertvollen Beitrag. Dadurch wird das Schreiben und Lesen von Programmen vereinfacht. Das Potenzial der Syntaxvereinfachung besteht darin, Fehler durch gute Übersicht gar nicht erst entstehen zu lassen.
- ➔ Und drittens sollten zusammengehörende Dinge bei der Entwicklung zusammengefasst werden. Das sorgt für ein leichteres Verständnis und besseren Überblick.

Zusammengehöriges kann zusammengefasst werden

Die richtige Programmiersprache

Qualitativ hochwertige Softwareentwicklung fußt auf denselben Regeln, die auch sichere Systeme über Normen vorgeben. Ein konsequent umgesetzter QM-Prozess ist also bereits die halbe Miete. Objektorientierte Programmiersprachen sind bereits mit Bordmitteln ausgestattet, die eine Umsetzung der Regeln erleichtern. So unterstützt C++ beispielsweise die Umsetzung der Regel, dass zusammengehörende Dinge bei der Entwicklung zusammengefasst werden sollen. Aufzählungstypen etwa können innerhalb einer Strukturdefinition angelegt werden, ohne dass dies Speicherplatz kostet. Aber C++ ermöglicht auch auf einfache Weise, die im praktischen Einsatz zu erreichende Betriebssicherheit der Software zu verbessern. So können z.B. 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.

ISO 26262 – Normierte Sicherheit

Die Norm ISO 26262 legt Vorschriften für sicherheitsrelevante elektrische/elektronische Systeme in Kraftfahrzeugen auf Basis des heutigen Stands von Wissenschaft und Technik und der dazugehörigen Basis-Standards fest. Ziel ist immer, die funktionale Sicherheit eines Systems mit elektrischen/elektronischen Komponenten im Kraftfahrzeug zu gewährleisten.

Leider bleibt sie in der Beschreibung ihrer Anwendung recht unscharf. Daher stellt sich die Frage:

Wie umsetzen?

Wichtig ist ein geeignetes Rüstzeug für die Umsetzung der Forderungen aus der Norm und für den Nachweis, dass dem tatsächlich Genüge getan wurde. Ganz allgemein ist ein wichtiges Thema der geeignete Nachweis, dass entsprechend dem Stand der Wissenschaft und Technik gearbeitet wurde und mithin alles vertretbar technisch Machbare umgesetzt wurde. Die Einhaltung der Norm ist somit ein wichtiger Punkt, der vor dem Hintergrund von Haftungsfragen gar nicht ernst genug genommen werden kann. Dabei soll aber erwähnt werden, dass eine Zertifizierung in der Norm selbst nicht gefordert ist, wohl aber der Nachweis von qualifizierten Mitarbeitern zum Beispiel über entsprechende regelmäßige Trainings.

Abbildung:

Je weniger Funktionalität, desto mehr Sicherheit.

Je mehr beim Programmieren sinnvoll zusammengefasst wird, desto geringer ist die Fehlerquote und umso mehr Sicherheit kann erreicht werden.

Qualifizierung von Softwaretools

In dem Zusammenhang ist hervorzuheben, dass die Qualifizierung der verwendeten Softwaretools ein ganz wichtiger Punkt ist. Es besteht nämlich die Möglichkeit, dass durch die Tools Fehler erzeugt werden, derer sich der Programmierer/Entwickler gar nicht bewusst ist - er bemerkt sie schlicht gar nicht. Stefan Kriso (Robert Bosch GmbH) sagt dazu, es müsse „... 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." Das ist eine Aufgabe, die es in sich hat.

Herausforderung Zukunft

Jede Software wird zwangsläufig immer nur in der „aktuellen Gegenwart" erstellt. Aber die Anwendung selbst und die möglichen Probleme, die sich ergeben können, kommen ausschließlich in einer heute unbekannten Zukunft mit unbekannten Einflussgrößen zum Einsatz. Das lässt die Herausforderung deutlich werden.

Die Methoden der Failure Tree Analysis (FTA) und der Failure Mode and Effects Analysis (FMEA), beide nicht ursächlich für die Entwicklung sicherer Software ersonnen, können verwendet werden, um die Sicherheit zu steigern. Gerade weil die Systeme hochgradig vernetzt und komplex sind, und weil heute noch nicht klar ist, welchen Fehlerquellen ein System morgen gegenüberstehen kann, sind klare Regeln und Grenzen für die Softwareanalyse wichtig. Je komplexer, desto mehr Aufwand für eine verlässliche Analyse.

Wie umsetzen? Die Zweite

Die Möglichkeit, die Forderungen der Norm ISO 26262 tatsächlich in der Softwareentwicklung umsetzen zu können, besteht im Wesentlichen aus 2 Stufen: 1. Sicherheitsziele setzen. Diese ergeben sich aus der Gefahren- und Risikoanalyse und bestimmen in weiterer Folge die geforderten Methoden zur Umsetzung. 2. Softwarearchitektur gestalten und externe Maßnahmen festlegen (Systemdesign-Analysen), die dabei helfen, systematische Fehler zu vermeiden. Systematische Fehler entstehen über fehlerhafte Entwürfe, Implementierungen und Tests, während zufällige Fehler quasi von außen her auftreten; sie werden oft durch äußere Einflüsse ausgelöst. Es gilt zu bedenken, dass Softwarefehler immer auf systematische Fehler zurückzuführen sind. Hardwarefehler können hingegen sowohl systematische Ursachen haben als auch zufällig auftreten.

Was ist sie nun, die Betriebssicherheit?

Dr. Jürgen Mottok, Scientific Head of Laboratory for Secure Systems in Regensburg, erklärt: „Die Betriebssicherheit eines technischen Systems versteht sich als die Reduktion des Risikos auf ein (wirtschaftlich) vertretbares Maß. Damit verbleiben tolerierbare Restfehler in den technischen Systemen. Geeignete Fehlererkennung und -reaktionen müssen umgesetzt werden".

Weiterführende Informationen

Links:

Überblicksartikel zur Norm ISO 26262:

➔ `http://bit.ly/XoTkdL`

Bitte anklicken oder mit dem Smartphone einscannen

Die vollständige Norm ISO 26262 zum Download (kostenpflichtig): www.iso.org

Buchtipps:

➔ Zuverlässigkeit komplexer Systeme aus Hardware und Software,
Willi Fuchs, ISBN: 3-410-32889-0

➔ Software-Qualität,
Georg E. Thaller, ISBN 3-8007-2494-4

Trainings bei MicroConsult (www.microconsult.de):

➔ Software-Qualität: Erfolgsfaktor im Produktentstehungsprozess - Methoden zur erfolgreichen Projektumsetzung unter Berücksichtigung wichtiger Normen und Standards

➔ Funktionale Sicherheit (Safety) von Elektronik und deren Software: Umsetzung nach IEC 61508 und ISO 26262

Qualität vor Funktionalität

Wenn das Navigationssystem im Automobil ausfällt, ist das ärgerlich. Wenn das Bremssystem bei 200 Stundenkilometern versagt, vielleicht tödlich. Schon seit Jahren steht eine ganze Industrie im Rampenlicht der Medien: Auf den Glanz der deutschen Automobilindustrie, vom einfachen Fahrzeug bis zur Luxuskarosse, fiel der Schatten frappierender Qualitätsmängel.

Qualität ist gefordert

Laut Kraftfahrt-Bundesamt ist die Zahl der Rückrufaktionen von Automobilherstellern in den letzten Jahren immer weiter gestiegen. Die deutschen Hersteller schneiden im Vergleich gut ab, doch dieses „gut" ist schlimm genug. Immer mehr Kunden fragen sich, ob ihr Vertrauen in die Sicherheit noch gerechtfertigt ist.

Alle Bereiche betroffen

Heute ist nicht mehr nur die automobile Luxusklasse vollgestopft mit Elektronik, die an Komplexität schwer zu überbieten ist. Diese Entwicklung betrifft heute Fahrzeuge aller Klassen und Typen, vom Motorroller bis zum Sattelschlepper. Und gerade in diesem Bereich verlagert sich die Produktentwicklung verstärkt von der Hardware hin zur Software. So hängt die Qualität technischer Produkte zunehmend von Embedded-Systemen mit hoch integrativer Software ab. Und zwar nicht nur im Automotive-Bereich, sondern in praktisch jeder Branche.

Herausforderung Produktqualität

Zur Qualitätsproblematik trägt nicht nur die Komplexität der Systeme bei, sondern auch der hohe Verteilungsgrad. Es scheint, als könne das Qualitätsbewusstsein der Entwicklungsunternehmen teilweise nicht mit dem Tempo der technologischen Neuerung und wachsenden Komplexität Schritt halten. Je größer der Software-Anteil ist, desto schwieriger lässt sich hohe Produktqualität erzielen. Dies mit fatalen Folgen gerade bei kritischen Embedded-Systemen. Die Palette reicht von Reputationsverlust und hohen Haftungskosten über Umweltschäden bis hin zur Gefährdung des menschlichen Lebens.

Klare Linie finden

Doch woraus resultieren die Mängel von Embedded-Systemen konkret? Nicht zuletzt liegt die Misere in den mangelnden Ausbildungssystemen von Universitäten und Fachhochschulen begründet. Die heutigen Defizite bestehen vor allem in den Kenntnissen, Qualität umfassend zu beschreiben. Am Beginn eines Projektes fehlen häufig klare Entscheidungen, welche Eigenschaften für ein Produkt wichtig und welche verzichtbar sind. Meist beschränkt sich die Spezifikation auf funktionale Anforderungen, wobei Qualität in vielen Fällen einen ausgeprägt nichtfunktionalen Charakter besitzt. Selbst einem Auftraggeber ist oft nicht bewusst, dass er diese mit der Anforderungsspezifikation konkret einfordern muss.

Mangel an konkretem Wissen

Wenig weiß man auch, wie Qualität zu erreichen und zu kontrollieren ist. Zudem mangelt es an dem Bewusstsein, dass sie Ressourcen kostet, ihr Fehlen aber wesentlich mehr zu Buche schlägt. So kann sich ein spät entdeckter Fehler aufgrund mangelnder Qualitätssicherung immens auf Aufwand und Termine auswirken. Ein Projektleiter trifft Entscheidungen über qualitätssichernde Maßnahmen nicht selten aus dem Bauch. Selbst wenn er richtig läge, kann er weder seinen Chef noch das Team adäquat überzeugen, warum ein bestimmtes Tool oder eine neue Methode wirklich Früchte tragen soll. Auch die Rahmenbedingungen eines Projektes lassen sich mit der eigenen Intuition nur unzureichend konkretisieren.

Qualität als Mittel zum Zweck

Sinnvoll definierte Qualitätsziele helfen, die Betriebssicherheit von Embedded-Softwareprodukten zu gewährleisten. Sie sollten auf Basis der Firmenstrategie festgelegt werden und durch die Produkte und Märkte bestimmt sein, die das Unternehmen bedienen will. Daraus resultiert ein Kundenpotenzial mit spezifischen Anforderungen.

Sicherheit durch Qualität als Kernkompetenz

Letztendlich muss das Management Qualität als Wettbewerbsfaktor und Kernkompetenz in Zeiten begreifen, in denen sich ein Software-Haus durch Funktionalität alleine nicht auf dem Weltmarkt behaupten kann. Geschäftsprozesse beeinflussen Entwicklungsprozesse und können damit auch auf die Betriebssicherheit wirksam sein. Je größer das Qualitätsbewusstsein der Führung ist, desto mehr ist es auch in den Köpfen der Mitarbeiter verankert. Ein Verständnis von Entwicklungsabläufen ist zentral, gerade wenn ein Unternehmen in vielen Bereichen aktiv ist. Ist es nicht vorhanden, trägt das Endprodukt eher zufällige Qualitätsmerkmale und wird in heutigen Märkten langfristig scheitern. Die unternehmerische Verantwortung entbindet den einzelnen Entwickler jedoch nicht von der Eigeninitiative. Letztendlich ist er dem zukünftigen Produkt am nächsten und kann damit den Wandel zu mehr Qualität und einer besseren Betriebssicherheit selbst anregen und mitgestalten. Nicht zuletzt bedingt der immer größer werdende Anteil an Elektronik im Fahrzeug auch eine Verlagerung der Sicherheitsfunktionen in elektronische Systeme. Per se ist Qualität also kein „nice to have", sondern eine Forderung aus den Normen für funktionale Sicherheit, und jeder Entwickler trägt Verantwortung für eine entsprechende Umsetzung.

Weiterführende Informationen

Links:

Softwarequalität bei Heise.de

➔ `http://bit.ly/Z8gPuE`

Bitte anklicken oder mit dem Smartphone einscannen

Buchtipps:

➔ Basiswissen Softwaretest,
Spillner, Linz, ISBN: 3-89864-178-3

➔ Die menschliche Seite des Projekterfolgs,
Peter Siwon, ISBN: 3-898-64716-1

➔ Software automatisch testen,
Dustin, Rashka, Paul, ISBN: 3-540-67639-2

Trainings bei MicroConsult (www.microconsult.de):

➔ ISTQB® Certified Tester Foundation Level: Strukturiertes und effizientes Testen von Embedded- und IT-Systemen

➔ Funktionale Sicherheit (Safety) von Elektronik und deren Software: Umsetzung nach IEC 61508 und ISO 26262

➔ Embedded-Software-Test: Best Practices für den Unit-/Modul-/Komponenten-Test

➔ Agiles Testen und Test Driven Development von Embedded-Systemen

➔ Embedded-Linux für Tester, Support und Service

3) SECURITY ALS STRATEGIE

Schutz vor Viren, Würmern, Hackern & Co.

Safety und Security bezeichnen unterschiedliche Eigenschaften von Embedded-Softwaresystemen. Doch bei genauer Betrachtung zeigt sich, wie eng die Betriebssicherheit von softwareintensiven Embedded-Systemen mit dem Schutz vor unbefugtem Zugriff oder gezieltem Angriff verbunden ist. Die dazu notwendigen Maßnahmen stellen Software- und Hardwareentwickler gleichermaßen vor hohe Herausforderungen.

Die Herausforderung

Embedded-Systeme sind so strukturiert und in umgebende technische Systeme integriert, dass sie komplexe Steuerungs- und Datenverarbeitungsaufgaben übernehmen können. Dabei darf die Betriebs- und Datensicherheit durch äußere Einflüsse nicht gestört oder gar verhindert werden. Mithin hat die Sicherheit eine doppelte Bedeutung: Einerseits sorgt sie dafür, dass die Betriebssicherheit (safety) erhalten bleibt, andererseits versteht man darunter die Gesamtheit der Maßnahmen, um ein System nach außen zu schützen (security). Das unterstreicht sowohl die Abgrenzung dieser Begriffe, zeigt aber auch, wie eng sie zusammenhängen.

Definition Safety - Security

Im Deutschen wird in der Regel nicht wie im Englischen zwischen den beiden Themen Security („Angriffssicherheit“) und Safety („Betriebssicherheit“) unterschieden, beide Begriffe werden unter „Sicherheit“ zusammengefasst. „Safety“ bezeichnet den Schutz der Umgebung vor einem Objekt, wogegen „Security“ den Schutz des Objektes vor der Umgebung meint. Beispiel: Ein Drehkreuz erfüllt die Anforderungen der Betriebssicherheit, wenn sein ordnungsgemäßer Gebrauch zu keinen Verletzungen wie Einklemmen führt. Die Anforderungen der Angriffssicherheit erfüllt es, wenn sichergestellt ist, dass keine unbefugte Person das Drehkreuz so manipulieren kann, dass es nicht mehr betriebssicher ist bzw. seine vorgesehene Funktion nicht mehr erfüllt.

„Die Security ist der Schutz vor gezieltem und höchstwahrscheinlich auch böswilligem Handeln, mit dem Ziel, Vertraulichkeit, Integrität, Authentizität etc. zu erreichen.“

Diese pragmatische Definition des Begriffs, formuliert von Prof. Dr.-Ing. Hans-Joachim Hof (Hochschule München) zeigt den umfassenden Anspruch deutlich. Das kann sich in der Realität aber als schwierig erweisen, da zum Zeitpunkt der Projektierung unbekannt ist, welcher Art von Angriff ein System irgendwann ausgesetzt sein wird. Die einschlägigen Normen für funktionale Sicherheit sehen daher vor, potenzielle Angriffe in die Gefahren- und Risikoanalyse der funktionalen Sicherheit aufzunehmen und bei absehbarer Bedrohung der Security eine explizite Security-Analyse durchzuführen.

Würmer, Viren & Co.

Frappierendes Beispiel hierfür ist der Computerwurm Stuxnet, der 2010 für große Unruhe und Schäden verantwortlich war. Stuxnet war in einer Weise aktiv, die bis dahin als nicht möglich galt:

- ➔ Der Wurm konnte Grenzen physikalisch getrennter Netze überspringen.
- ➔ Stuxnet konnte in Embedded-Plattformen Schäden anrichten, die dem Hacker vorher nicht bekannt gewesen sein konnten. Der Wurm war so „gewitzt“, dass die mangelnde Vertrautheit mit einem System kein Hinderungsgrund war.
- ➔ Stuxnet gelang eine weitere „Unmöglichkeit“, er infiltrierte eine Atomaufbereitungsanlage. Damit drückte Stuxnet den aktuellen Möglichkeiten des technisch Machbaren seinen eigenen Stempel auf. Eine optimierte Betriebssicherheit wurde mangels Security zum Sicherheitsrisiko.

Hacker

Bedrohungen durch Hacker stellen die Entwickler von Embedded-Systemen vor zunehmende Herausforderungen. Aufgrund der zunehmenden Vernetzung von Embedded-Systemen fallen mechanische Barrieren zwischen ihnen immer mehr weg.

➔ 14.11.2012 Rund sechs Millionen Menschen sind im vergangenen Jahr Opfer von Handy-Viren geworden (Radiomeldung BR, Quelle: Chip.de).

Dennoch beruht Sicherheit häufig auf physikalischer Sicherheit und weniger auf Cyber-Security oder Informationssicherheit. Und das, obwohl es gerade in Infrastrukturnetzen (z.B. intelligente Stromzähler – Smart Meter) von großer Bedeutung ist, dass die vernetzten Komponenten autonom, ohne Benutzereingaben und vor unbefugten Zugriff geschützt miteinander kommunizieren können.

„Aufgrund der Verarbeitung und Zusammenführung personenbezogener Verbrauchsdaten in Messsystemen sowie möglicher negativer Rückwirkungen auf die Energieversorgung ergeben sich hohe Anforderungen an den Datenschutz und die Datensicherheit. Bekannt gewordene Hackerangriffe auf Smart Metering Systeme in den USA und Gefährdungen wie die Schadsoftware Stuxnet haben gezeigt, dass in Deutschland ein akuter Handlungsbedarf für Rahmenbedingungen einer sicheren Lösung im Bereich Smart Metering besteht.“ (Zitat von der Website des Bundesamts für Sicherheit in der Informationstechnologie).

Was könnte die Lösung sein? Denkbar wäre eine Hardware-basierte IT-Sicherheitsarchitektur für Embedded-Systeme mit einem Security Controller oder einem Trusted-Platform-Modul im Zentrum - ein sicherer Speicher für digitale Schlüssel. Der wiederum könnte ein Zentrum für kryptografische Operationen bilden, und dieses müsste an den lokalen Computer gebunden sein und nicht an einen bestimmten Benutzer. Sodann wäre es unmöglich, das Trusted-Platform-Modul entgegen den Interessen des Eigentümers zu nutzen, sofern der Beschränkungen festgelegt hat: eine Basis für die sichere Kommunikation der Netzteilnehmer.

Um nochmals auf intelligente Stromnetze, Smart Grids, zurück zu kommen: Diese benötigen für die Ermittlung des individuellen Strombedarfs und die korrekte Abrechnung die Verwendung entsprechender Verbrauchszähler (Smart Meter). Szenarien, die böswilliges Eindringen und Betrügen ermöglichen, sind hier leicht denkbar. Die Liberalisierung des Strommarktes mit zahlreichen Klein- und Kleinst-Stromlieferanten aus Wasserkraft oder Photovoltaikanlagen kann Hackern weitere Einfallstore für mögliche Angriffe zum Zweck der kriminell motivierten Manipulation bieten. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) erarbeitet daher gemeinsam mit Industriepartnern für Smart Meter ein neuartiges Schutzprofil. Das neue Schutzprofil /3/ definiert ein anhand gemeinsamer Kriterien festgelegtes „Security Modul“ für alle kryptografischen Operationen.

Kryptologie

Bei der Entwicklung von Embedded-Systemen können durch die Anwendung kryptologischer Verfahren viele Prozesse sicher oder zumindest sicherer gemacht werden. Leider passieren nämlich immer wieder die gleichen Fehler bei der Implementierung von Kryptologie. Eindringlinge haben es dadurch oft unnötig leicht. Softwareentwickler gehen häufig davon aus, dass die von ihnen entwickelten Algorithmen niemand knacken kann, doch das Gegenteil ist der Fall. Ein gängiges Mittel ist das Schreiben unklaren oder scheinbar verworrenen Codes. Doch gerade bei der heutigen Vernetzung wird jedes Verfahren früher oder später bekannt. Zum Beispiel führt der Selbsttest eines eigenentwickelten Algorithmus möglicherweise zu dem Ergebnis, dass es nicht einmal dem Autor selbst gelingt, ihn zu knacken. Das gilt dann als besonders sicher.

Das Kerckhoffs’sche Prinzip oder Kerckhoffs’ Maxime ist ein 1883 von Auguste Kerckhoffs formulierter Grundsatz der modernen Kryptographie, welcher besagt, dass die Sicherheit eines Verschlüsselungsverfahrens auf der Geheimhaltung des Schlüssels beruht und nicht auf der Geheimhaltung des Verschlüsselungsalgorithmus. Dem Kerckhoffs’schen Prinzip wird oft die sogenannte „Security by Obscurity“ gegenübergestellt: Sicherheit durch Geheimhaltung des (Verschlüsselungs-)Algorithmus, möglicherweise zusätzlich zur Geheimhaltung des Schlüssels. (Wikipedia)

Aber: „Jeder kann einen Algorithmus entwickeln, den er selbst nicht brechen kann, aber es geht darum, dass ihn andere nicht brechen können“ (B. Schneier) – eine interessante Erkenntnis. Letztlich sieht man es einem Algorithmus nicht an, ob er nun sicher ist oder nicht. Daher gilt die Empfehlung, konservativ zu entwickeln und dabei bekannte und altbewährte Algorithmen und Methoden zu nutzen. Doch auch einer perfekt implementierten Kryptologie sind Grenzen gesetzt, wie z.B. Denial-of-Service-Attacken oder der immer Erfolg versprechende Versuch, stets das schwächste Element eines Walls von Maßnahmen für die Sicherheit zu instrumentalisieren: den User selbst. So gilt letztlich, dass gesunder Menschenverstand niemals hinter Formalismen verschwinden darf. Für die Realisierung von Sicherheit gilt immer, dass etwas Sicherheit besser ist als gar keine.

Programmierung

Die Risiken von Angriffen auf Embedded-Systeme sind real. Es ist z.B. durch Umgehen von Restriktionen möglich, eine Insulinpumpe aus der Entfernung und völlig unbemerkt zu manipulieren und die zu verabreichende Dosis zu verändern - für einen Diabetes-Patienten lebensgefährlich.

Das Bundesministerium für Wirtschaft und Technologie stellte schon 2010 eine zunehmende und alarmierende Professionalisierung von Angriffen fest. Die Medien berichten meist nur über die spektakulärsten Fälle. Die Mehrzahl der Angriffe ist zwar weniger spektakulär, doch dabei kaum weniger gefahrenvoll. Gerade überraschende Angriffsszenarien auf Embedded-Systeme zeigen, wie wichtig es ist, schon in der Softwareentwicklung spätere Angriffe zu berücksichtigen. Die sichere Softwareentwicklung, der sichere Code, spielt hier eine entscheidende Rolle. Angesichts des rasanten Fortschritts, der von zunehmendem Zeit-, Konkurrenz- und Erfolgsdruck zuverlässig begleitet wird, besteht die zentrale Herausforderung darin, neben Qualitätsforderungen auch ein angemessenes Sicherheitsniveau zu realisieren. Die Anforderungen Offenheit (Vernetzbarkeit mit anderen Systemen), Änderungsfreundlichkeit und Sicherheit sinnvoll unter einen Hut zu bringen stellt extrem hohe Ansprüche an das Software-Engineering. Dies führt beispielsweise zu folgender paradoxen Situation: Gerade bei Embedded-Software kann eine Verbesserung der Zugangssicherheit über Software-Updates erfolgen. Dies wiederum erfordert die Zugänglichkeit des Systems über Schnittstellen, z.B. zum Internet. So führt die Anforderung, die Sicherheit der Software durch Updates zu verbessern, zwangsläufig dazu, dass das System nicht vollständig gegen unbefugten Zugriff schützbar ist. Daraus ergeben sich gleichermaßen Aufgaben für die Entwicklung von Soft- und Hardware. In punkto Hardware werden dafür nämlich sichere Interfaces und Übertragungsprozeduren benötigt. Und die Software muss so aufgebaut sein, dass sie autorisiert korrigiert, aber nicht ohne Autorisierung manipuliert werden kann. Das Motto „Verbauen und Vergessen“ gehört damit der Vergangenheit an.

Weiterführende Informationen

Links:

Bundesamt für Sicherheit in der Informationstechnik:

➔ `http://www.bsi.de`

Bitte anklicken oder mit dem Smartphone einscannen

Buchtipps:

➔ Software Inspection,
Gilb, Graham, ISBN: 0-201-63181-4

➔ Viren Würmer und Trojanische Pferde,
Andreas Winterer, ISBN-10: 3-815-82265-3

Trainings bei MicroConsult (www.microconsult.de):

➔ Security: Kryptografie richtig anwenden

➔ Requirements Engineering und Management für die Entwicklung in der Industrie

➔ Embedded-C: Effektiver Einsatz von Programmiermethoden und -tools für Embedded-Anwendungen

➔ Objektorientierte Softwareentwicklung: Spezielle Programmierprinzipien

➔ Design Patterns (nicht nur) für Embedded-Systeme

➔ Funktionale Sicherheit (Safety) von Elektronik und deren Software: Umsetzung nach IEC 61508 und ISO 26262

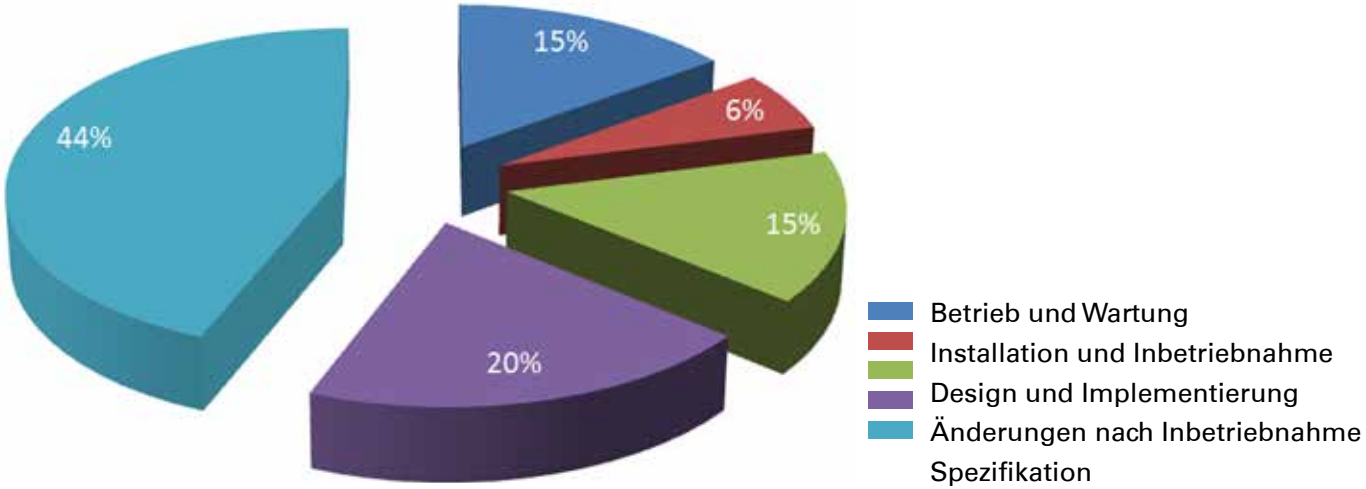
Des Übels Wurzel

Wer gut sattelt, reitet gut, sagt ein Sprichwort. Bei Projektbeginn spielt deshalb für alle Beteiligten die gemeinsame Definition der zentralen Anforderungen an Qualität und Sicherheit eine entscheidende Rolle. Standards wie die IEC 61508 liefern entscheidende Hinweise und Richtlinien für notwendige Maßnahmen bei Entwicklung und Qualitätssicherung.

Fehler im Projektlebenszyklus

Die Grafik unten zeigt anschaulich, dass die meisten Fehler ihre Wurzeln in der Spezifikationsphase haben. Dies ist umso fataler, als die Folgekosten in der Tendenz exponentiell mit der Zeitverzögerung zwischen Fehlerentstehung und seiner Beseitigung ansteigen. Wer heute noch Qualität ausschließlich über Tests am Ende des Entwicklungsprozesses realisieren will, überschreitet deshalb schnell den Kostenrahmen. Den Königsweg schlagen hier Unternehmen ein, die systematisch vorgehen und ihre Qualitätsziele von Beginn an konsequent in jeder Projektphase verfolgen.

Informationen über Maßnahmen zur Qualitätssicherung finden Sie in Kapitel 6 dieses Trend Guides.



Die Spezifikation ist die Hauptquelle aller Probleme. Sie treten meist spät zutage und sind dann wesentlich aufwändiger zu beseitigen. Quelle: MicroConsult GmbH, Trend Guide „Embedded Quality“ 2003

Merkmale auswählen

Die gezielte Auswahl von Qualitätsmerkmalen einer Software-Applikation bildet bereits eine wichtige Basis für eine bedarfsgerechte Anforderungsspezifikation. Aus den so gewonnenen Anforderungen kann dann eine Software-Architektur abgeleitet werden, die diese Qualitätsmerkmale abbildet. Diese Architektur wiederum bestimmt die Rahmenbedingungen für das Verhalten des späteren Produkts in Bezug auf das Betriebssystem und die Hardware (Design) und schafft damit die Voraussetzungen für die Implementierung.

Die wichtigsten Qualitätsmerkmale

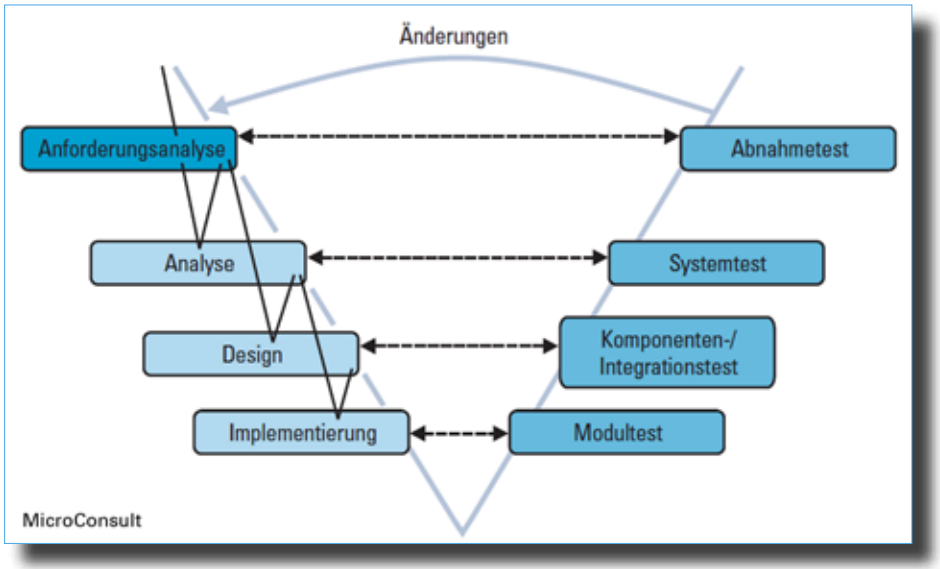
- ❑ **Zuverlässigkeit**
- ❑ **Benutzbarkeit**
- ❑ **Effizienz**

Ausführliche Informationen zu Qualitätsmerkmalen finden Sie in Kapitel 6 dieses Trend Guides.

Test

Die unterschiedlichen Testverfahren beziehen sich auf die Requirement-Analyse und sichern die dort aufgestellten Anforderungen. Im Rückschluss bedeutet dies, dass das Testen ohne eine vernünftige Anforderungsanalyse nicht oder nur durch Zufall zum Erfolg führt.

Entwickler von Embedded-Systemen stehen vor der Herausforderung, dass aus einer Anforderung nicht deterministisch die Umsetzung folgt. Dies gilt beispielsweise für zeitkritische Systeme wie Fertigungsmaschinen, die intensiv mit ihrer Umgebung interagieren. Hier gibt es keine klare Wirkungskette für das Realisieren der angestrebten Quality of Services (QoS), von der Analyse über die Architektur bis zum fertigen Code. Deshalb nähert sich das moderne Engineering über inkrementelle Vorgehensmodelle der QoS schrittweise an. Dabei werden Zwischenergebnisse nicht nur für einzelne Bereiche, sondern vor allem im Gesamtzusammenhang des Projektes betrachtet und bewertet.



Das zackige V-Modell

Internationale Norm

Die alle Prozesse begleitende Norm IEC 61508 wurde ursprünglich auf das V-Modell zugeschnitten, lässt dem Entwickler jedoch prinzipiell die freie Wahl seines (zeitlichen) Vorgehens, sofern er dabei alle ursprünglichen Anforderungen weiterhin erfüllt.

Die IEC 61508 listet ganze 450 Merkpunkte für die konkrete Umsetzung auf. Die Norm ist seit 1998 international eine klare Vorgabe, wie ein Embedded-System hardware- und softwareseitig zu bauen ist. Es existieren von ihr zahlreiche anwendungsspezifische Ableitungen, beispielsweise für die Eisenbahntechnik, Medizintechnik, chemische Prozesstechnik oder Kerntechnik. In ihrem ersten Teil nennt die IEC 61508 die Anforderungen und die vier Safety Integrity Levels (SIL) für Embedded-Systeme. Im Teil drei sind die Bauvorschriften für die Konstruktion der Software festgelegt. 2010 wurde die neueste und aktuelle Version der IEC 61508, die Edition 2.0, veröffentlicht.

Hier finden Sie eine Einführung in die Entwicklung gemäß IEC 61508:

➔ `http://bit.ly/TI4Tt3`

Bitte anklicken oder mit dem Smartphone einscannen

Innovation

Kundenanforderungen, Lasten- und Pflichtenhefte, gesetzliche Vorgaben, Normen und Richtlinien usw. geben der Entwicklung von Software für Embedded-Systeme ein stabiles Rückgrat. Das klingt wenig verlockend, weil scheinbar umständlich und schwierig. Doch wer hoch hinaus will und erfolgreich Innovation betreiben will, braucht ein stabiles Rückgrat. Andernfalls besteht die Gefahr, dass das Produkt den Anforderungen der Innovation nicht gewachsen ist und buchstäblich zusammenbricht. Sicherheits- und Qualitätsstandards sind, richtig eingesetzt, keine Feinde der Innovation. Im Gegenteil, sie sind eine notwendige Voraussetzung, weil sie dem Produkt in innovativen Veränderungsphasen Struktur und den Halt geben. Sollte es allerdings im Sinne der Innovation notwendig sein, etablierte Regeln zu brechen, so haben klar formulierte Anforderungen und Standards ebenfalls einen Nutzen. Sie fordern konsequente Dokumentation und einen strukturierten Änderungsprozess und bilden die Basis für eine effiziente Handhabe von Änderungen und Innovation. Sie stellen einen Referenzpunkt dar, der die Einschätzung von Chancen und Risiken einer Innovation erleichtert und sind somit ein wichtiger Prüfstein für neue Ideen und Wege.

5) Die Zukunft im Blick

Künftigen Herausforderungen schon heute begegnen

Die Zukunftssicherheit eines Embedded-Softwaresystems sagt heute im Wesentlichen aus, dass die Anpassung an künftige Anforderungen der Betriebs- und Angriffssicherheit, der Funktionalität und Updatefähigkeit, des Designs und der Standards möglich ist.

Hellseher gefordert

Der Gedanke liegt nahe, dass es hellseherischer Fähigkeiten bedarf, um heute schon sagen zu können, welchen Gefahren, Aufgaben und Herausforderungen ein heute entwickeltes Embedded-Softwaresystem in der Zukunft gegenüberstehen wird. Doch zum Glück gibt es ganz reale und gut funktionierende Methoden und Hilfsmittel, die uns dem Ziel optimierter Sicherheit näher bringen können.

Geglückte Zukunftssicherheit

1977 starteten von Cape Canaveral 2 Trägerraketen, die mit den Voyager Raumsonden beladen waren. Voyager 1 und 2 hatten die Aufgabe verschiedene Planeten zu passieren, Messungen durchzuführen und Aufnahmen anzufertigen, über einen Zeitraum von rund 5 Jahren. Das Programm funktionierte viel besser als erwartet. Bis heute funken die Sonden ihre Signale aus den Tiefen des Weltalls, rund 30 Jahre länger als geplant.

Die Herausforderungen an Softwareentwickler für Embedded-Systeme, um Zukunftssicherheit bei ihren Produkten zu gewährleisten, sind heute recht gut bekannt. Die Entwickler können sich aus einem Topf verschiedener Möglichkeiten bedienen; dazu gehören die Einhaltung einfacher Regeln direkt bei der Programmiertätigkeit, wie auch die Inanspruchnahme externer Dienstleister, die darauf spezialisiert sind.

Generell gilt aber stets der Grundsatz: Keep it simple. Die Vermeidung undurchsichtiger Konstrukte und die Wahl eines Softwarecodex, der weitgehend selbsterklärende und überschaubare Softwareelemente fordert, ist ein guter Einstieg in diese Zukunftssicherung.

Softwareerosion – Programmiertes Scheitern

Das Smartphone-Betriebssystem Symbian wird nicht weiterentwickelt. Es leistete einige Jahre sehr gute Dienste, doch nach und nach begann es fragil und instabil zu werden. Auch bei kleinen Änderungen und Aktualisierungen dauerte es immer länger und wurde immer schwieriger, die Software anschließend wieder zu stabilisieren. Heute verschlänge die Weiterentwicklung zu große Ressourcen. Diese veränderungsbedingte Fragilität nennt man Softwareerosion. Ist sie bereits im fortgeschrittenen Stadium, ist es sehr schwierig, das Problem wieder in den Griff zu bekommen. Sie gehört damit zu den größten Bedrohungen für die schnellen Innovationszyklen in vielen Branchen.

Qualitätsbeurteilung

Jede Software kann gewissermaßen in ihrem „so sein, wie sie ist“ beurteilt werden. Dieses „so sein“ wird mit dem Begriff der Qualität bezeichnet. Qualitätsmodelle helfen dabei, spezifische Qualitätskriterien von Produkten zu beurteilen. Bei der Softwareentwicklung sind viele Personen mit unterschiedlichen Rollen und Blickwinkeln beteiligt. Die Aufgaben innerhalb der Prozesse sind hochgradig vernetzt, müssen aber auch gegeneinander abgegrenzt werden. Diese Vielfalt will erst einmal beherrscht sein. Jeder Projektbeteiligte muss einen angemessenen und realistischen Einblick in den Projektfortschritt und die Softwarequalität erhalten. Idealerweise wird diese Information auf Knopfdruck auf Basis definierter Qualitätsmodelle durch Analysetools automatisch ermittelt und visualisiert. Diesen Ansatz unterstützen z.B. Software-Analyse-Tools der Firmen Axivion, Klocwork, Squoring oder Coverity. Die große Kunst besteht nun darin, geeignete Qualitätsmodelle zu erstellen und diese auch an veränderte Bedingungen anzupassen. Bekannte und verbreitete Standards, Normen und Metriken dienen als Grundlage für solche Qualitäts- und Bewertungsmodelle. Eine Software und der damit verbundene Prozess muss z.B. anhand spezifischer Kriterien, wie Updatefähigkeit, Wirksamkeit des Change Managements oder Stabilität nach Änderungen, geprüft werden.

Änderungen nehmen zu

Die Komplexität von Embedded-Software verdoppelt sich etwa alle 10 Monate (Quelle: Vorlesung Avionik 2010, Prof. Dr. Sergio Montenegro, Universität Würzburg). Aus dem Grund sind umfangreichere Maßnahmen erforderlich, die bei der Verwendung neuester, objektorientierter Programmiersprachen oder Modellierungsmethoden beginnen und die auch neue Rollen und Aufgaben, wie z.B. die des Softwarearchitekten, sowie des Konfigurations-, Versions- und Variantenmanagements notwendig machen.

Offene Codes

Im Bereich der Web Content Management Systeme (WCMS) gibt es jede Menge hervorragender Lösungen, die als Open Source Produkte veröffentlicht wurden und werden. „Drupal“ ist aktuell eines der besten. Rund 900 Programmierer entwickeln es kontinuierlich weiter. Die Webseite drupal.org weist rund 300.000 registrierte Nutzer auf - täglich gehen eine Vielzahl von Ideen ein, die das System mitunter um große Schritte voranbringen. Die Vielzahl der Entwickler, die daran arbeiten, es weiterentwickeln und voranbringen, sowie die mittlerweile große Verbreitung sorgen für die Sicherstellung der Aktualität und hohen Qualität des Systems. Projekte und Produkte aus dem Open Source Umfeld weisen nämlich eine Besonderheit auf: Jeder kann ein vorhandenes System verändern und verbessern. Die dafür erforderlichen Informationen und Daten stehen öffentlich zur Verfügung. Bei Nutzung von Open Source Software ist zu prüfen, inwieweit sie den Bestimmungen der sogenannten GNU General Public License (GPL) und dem „Copyleft“ der GPL unterliegen (Alle Änderungen, die vorgenommen werden, müssen anschließend ebenfalls veröffentlicht werden, um ggf. erneut weiterentwickelt oder verbessert werden zu können). Was hier einer möglichst kontinuierlichen Verbesserung eines Produktes dient, kann aber den Interessen eines Unternehmens, sein geistiges Eigentum zu schützen, entgegenstehen. Hinsichtlich der Erlangung und Erhaltung von Sicherheit mag die GPL sehr sinnvoll sein, hinsichtlich der Erhaltung betrieblicher Alleinstellungsmerkmale ist sie das eher nicht. Dr. Carsten Emde, OSADL, räumt in diesem Zusammenhang dem Begriff der Ableitung eine ganz zentrale Bedeutung ein. Die Frage, ob eine Software von einer anderen (die vielleicht unter der GPL entstand) abgeleitet wurde, ist entscheidend dafür, wem die Software letztlich tatsächlich gehört. Die Antwort lässt sich im Einzelfall nicht immer einfach ermitteln.

Zukunftssicherheit durch Kompetenz

Die Zukunftssicherheit der Software spiegelt sich in besonderem Maße in der Zukunftsfähigkeit ihrer Produktmanager, Architekten und Entwickler wider. Wer heute Software entwickelt, sollte seinen Blick in die Zukunft richten. Dabei lohnt sich der Blick über den Embedded-Tellerrand in die große, weite Welt der IT. Welche Methoden und Tools werden dort heute eingesetzt und welche Bedeutung könnten sie für die Embedded-Welt von Morgen haben? Hellseherei ist dabei gar nicht nötig, es genügt meist, vor die Tür zu treten und die Augen zu öffnen. In jedem Falle ist es empfehlenswert, sich nach Technologien und Darstellungsmöglichkeiten umzusehen, die von Programmiersprachen unabhängig sind. Solche Ansätze erleichtern es, die Architektur, Ideen und Prinzipien einer Software in die Zukunft mitzunehmen.

Weiterführende Informationen

Links:

➔ `http://bit.ly/11AxOHz`

Bitte anklicken oder mit dem Smartphone einscannen

***Trendaussage**: Prof. Dr. Christian Siemers: „Wir werden - abgesehen von Multicore-Systemen - zu dualen Systemen kommen: Ein Knoten verrichtet die Arbeit (Controller), der andere, unmittelbar dazugehörend, überwacht (Observer). Dieser zweite Knoten sammelt Fehler aus dem ersten und dem Umfeld. Auf Basis dieser Informationen konfiguriert er dann entsprechend die weitere Betriebssituation. Dies wird Self-X, speziell Self-Awareness und Self-Configuration genannt. Das hat erheblichen Input auf die Softwareentwicklung, denn nun müssen auch verschiedene Konfigurationen vorgehalten werden.*

Lesenswert:
http://de.wikipedia.org/wiki/Organic_Computing

Weiterführende Informationen zu diesem Thema: siehe Anhang auf Seite 19.

Weniger ist mehr

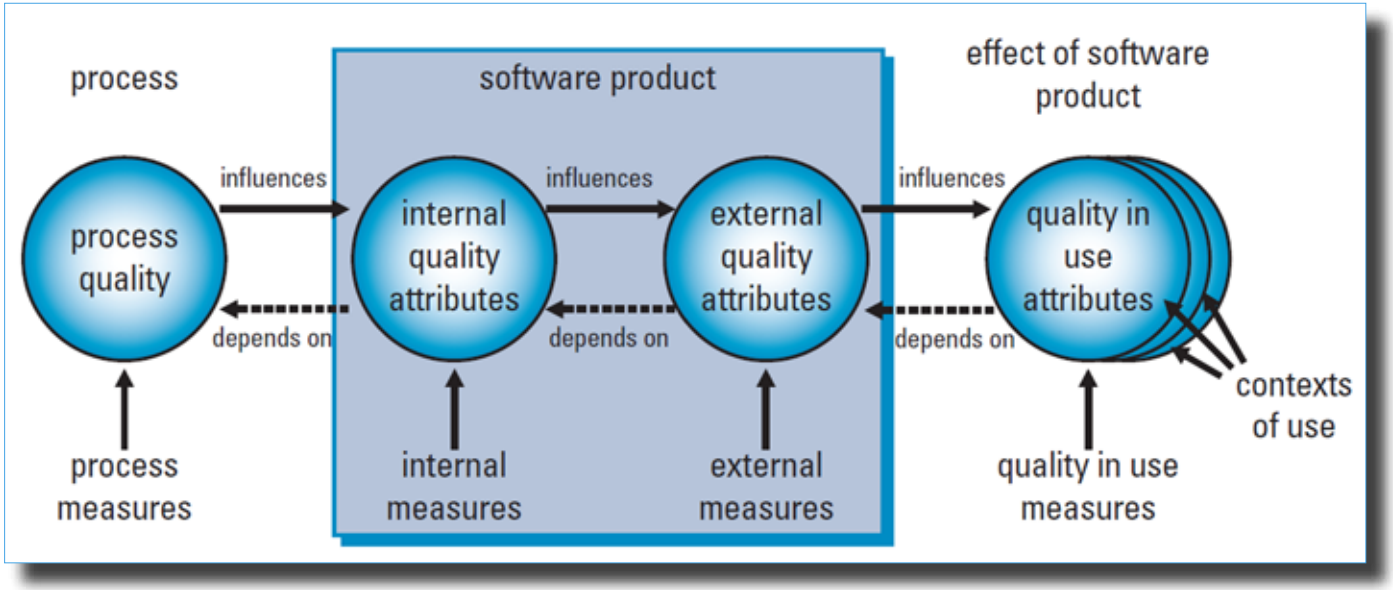
Wer Entwickler nach ihrem Verständnis zur Qualität befragt, wird wahrscheinlich höchst unterschiedliche Antworten erhalten. Missverständnisse bestehen nicht nur bezüglich des Begriffes an sich, sondern betreffen auch die Bewertung und Gewichtung ihrer Merkmale. Dieser Umstand kann ein Unternehmen schnell teuer zu stehen kommen.

Begriffsdefinition

Der Qualitätsbegriff beinhaltet heute zwei wesentliche Aspekte:

- Qualität des Prozesses: Wie gut sind die Entwicklungsabläufe, mit denen Embedded-Software erstellt wird?
- Qualität des Produktes als Resultat der Prozesse: Sie wird vor allem von der Eignung für den späteren Einsatz bestimmt.

Je komplexer Systeme sind, desto mehr wird sich eine hohe Prozessqualität auf die **Produktqualität** auswirken. Die sinnvolle Gestaltung der Prozesse sorgt für Termin- sowie Budgettreue, und sie bestimmt letztendlich die Kundenzufriedenheit. **Prozessqualität** bewährt sich auch in der Akquise-Politik: Wer seinem Interessenten den Weg zum Ziel verständlich machen und klare Zwischenziele stecken kann, schafft Vertrauen. Gerade bei kritischen Embedded-Systemen (Reputationsverluste, Umweltschäden, Verletzte oder Tote) rückt dagegen die Produktqualität verstärkt in den Vordergrund.



Axiom des Software Engineering. Quelle: Fenton, Norman E.; Software Metrics, 1991. Fenton konstatiert, dass ein direkter Zusammenhang zwischen internen Strukturen und externen Qualitätsmerkmalen besteht.

Prozess vs. Produkt

Aus den Anforderungen an Prozesse und das Produkt lassen sich interne und externe Qualitätsmerkmale ableiten. Die internen werden direkt am System gemessen, beispielsweise ob der Leiterbahnabstand auf einer Platine ausreichend groß ist. Externe Merkmale, wie beispielsweise Zuverlässigkeit oder Funktionalität, sind dagegen vom Zusammenwirken mit der äußeren Umgebung bestimmt.

Die scheinbare Trennung von interner und externer Qualität relativiert das „Axiom des Software Engineering" von Norman E. Fenton. Danach bildet eine gute interne Qualität die Voraussetzung für eine gelungene externe. Dies umso mehr, je länger die Lebensdauer einer Software ist, die mit wechselnden Anforderungen einem ständigen Wandel unterliegt.

Die Normenreihe ISO/IEC 250xx von 2005 legt folgende interne und externe Qualitätsmerkmale fest:

- Funktionalität: Angemessenheit, Richtigkeit, Interoperabilität, Ordnungsmäßigkeit, IT-Sicherheit
- Zuverlässigkeit: Reife, Fehlertoleranz, Wiederherstellbarkeit
- Benutzbarkeit: Verständlichkeit, Erlernbarkeit, Bedienbarkeit
- Effizienz: Zeitverhalten, Verbrauchsverhalten
- Änderbarkeit: Analysierbarkeit, Modifizierbarkeit, Stabilität, Prüfbarkeit
- Übertragbarkeit: Anpassbarkeit, Installierbarkeit, Konformität, Austauschbarkeit

In der Realität

Unternehmen müssen bei der Umsetzung von Qualität auf ein optimales Kosten-Nutzen-Verhältnis achten: Wie kann ich dem Kunden für das Geld, das er zu zahlen bereit oder fähig ist, das Produkt anbieten, das seinen Wünschen am nächsten kommt?

Der Schlüssel liegt nach wie vor in der Gewichtung und Beschränkung der Qualitätsmerkmale. Oder einmal ganz plastisch: Würde ein Automobilhersteller einen Formel 1-Wagen mit einem Verbrauch von drei Litern produzieren, der Container transportieren soll?

Die richtige Gewichtung

Fokussiert sich ein Projektteam auf maximal vier bis sechs Merkmale, gelingt ihm erfahrungsgemäß der Spagat zwischen Kundenanforderungen und Entwicklungskosten bravourös. Gegen einen Rundumschlag in Sachen Qualität spricht auch, dass sich einige Merkmale gegenseitig ausschließen oder zumindest teilweise widersprechen, beispielsweise die Zuverlässigkeit und harte Zeitanforderungen an ein System. Bei ersterer zählen gute Strukturen, bei letzteren sind diese weitgehend zu ignorieren.

Gewichtung von Merkmalen

Qualität kostet - Merkmale gewichten

Da Qualität kostenintensiv ist, müssen sich Entwicklungsunternehmen auf bestimmte Merkmale fokussieren. Einige Beispiele verdeutlichen die zentrale Bedeutung der Gewichtung von Merkmalen konkret:

Die wichtigsten Qualitätsmerkmale

- **Zuverlässigkeit**: Sie gibt an, wie verlässlich eine dem Produkt oder System zugewiesene Funktion in einem Zeitintervall erfüllt wird. Sie ist nicht direkt messbar und muss statistisch bestimmt werden.
- **Benutzbarkeit**: Dieses Qualitätsmerkmal ist dann wenig ausgeprägt, wenn ein Anwender lange braucht, um ein System in den Griff zu bekommen. Eine schlechte Ausprägung dieses Merkmals könnte jedoch sinnvoll sein, wenn ein Produkt nur selten bedient wird.
- **Effizienz**: Sie wird heute in vielen Projekten unterbewertet, beispielsweise im Hinblick auf Speicher- oder Prozessorauslastung. Daraus resultieren häufig fehlende Ressourcen für die Integration mehrerer Systemkomponenten in einem Gesamtsystem. Ist beispielsweise ein gutes Zeitverhalten wirklich unumgänglich, muss diese Anforderung in der Spezifikation bewusst formuliert sein.

Neue Ansätze

Zur Erlangung qualitativ hochwertiger und sicherer Software wird immer mehr zwischen der „inneren“ und der „äußeren“ Softwarequalität unterschieden, die sich allerdings gegenseitig beeinflussen. Die innere Qualität wird vor allem durch Software-Engineeringmethoden beeinflusst, die Lesbarkeit, Testbarkeit, Änderungsfreundlichkeit und Erweiterbarkeit des Codes verbessern. Die äußere Qualität bezieht sich auf Merkmale, die die praktische Anwendbarkeit des Systems aus Sicht des Nutzers bestimmen, z.B. Betriebssicherheit. Prof. Dr. Rainer Koschke von der Universität Bremen bemerkt dazu, dass das Testen allerdings nur die äußere Qualität prüfen kann, also nur die Mängel aufdeckt, die bereits geschehen sind. Besser ist es aber, Fehler und Mängel zu vermeiden, wobei eine gute innere Softwarequalität hilft.

*„Bei einem für seine Produktqualität bekannten Unternehmen wie Miele spielen daher auch diese Qualitätsaspekte (innere und äußere Qualität, Zuverlässigkeit, Wartbarkeit, Testbarkeit, Red.) eine wichtige Rolle. Diese zu gewährleisten, ist Aufgabe eines (iterativen) Qualitätssicherungsprozesses, bestehend aus (praktikablen!) konstruktiven (Entwicklungsprozess-Vorgaben wie FMEA, Coding-Convention *vor* Entwicklungsbeginn, Architektur-Prinzipien), statischen (toolgestützte Codeanalyse/ Metriken, Peer-Review) und dynamischen Maßnahmen (Simulation, Unit-Testing, Kontinuierliche Integration, Integrations- und Systemtests), die den Entwicklungsfortschritt flankierend begleiten.“* (Dr. Christian Scheering, Miele)

Steigender Kostendruck in der Produktentwicklung, sowohl bei den Geräteherstellern als auch bei den Programmierern, erschweren es immer mehr oder machen es mitunter unmöglich, für eine vernünftige Entwicklung Geld und Zeit aufbringen zu können. *„Eine vernünftige Software-Architektur mit hoher Modularität und guter Wartbarkeit, konsequentes Design-for-Test sowie eine starke Testautomatisierung sind hier die Mittel der Wahl, welche wir bei Zühlke all unseren Kunden empfehlen bzw. für sie implementieren. Bei größeren Projekten setzen wir grundsätzlich immer auf Testautomatisierung und Continuous Integration. Aber all diese Maßnahmen sind erst bei einem längeren Produktlebenszyklus lohnenswert. Bei Low-Cost-Geräten werden wir mit „You get what you pay for" (Du bekommst das, wofür du bezahlst) leben müssen.*" (Dr. Eberhard Wildermuth, Zühlke)

Beachten Sie zu diesem Thema auch den interessanten Artikel „6 Thesen zur Softwarequalität in eingebetteten Systemen" der Zeitschrift ELEKTRONIKPRAXIS.

➔ `http://bit.ly/VOOtPJ`

Bitte anklicken oder mit dem Smartphone einscannen

Weiterführende Informationen

Links:

Artikel „Klassisches Requirements Engineering und agile Entwicklungsprozesse kombinieren"

➔ `http://bit.ly/12FjY5T`

Bitte anklicken oder mit dem Smartphone einscannen.

Artikel „Requirements Engineering für Embedded-Systeme"

➔ `http://bit.ly/Va1Qga`

Bitte anklicken oder mit dem Smartphone einscannen.

Buchtipps:

- ➔ Requirements Engineering und -Management, Chris Rupp, ISBN:3-446-21664-2
- ➔ Requirements-Led Project Management, Suzanne Robertson u.a., ISBN: 0-321-18062-3

Trainings bei MicroConsult (www.microconsult.de):

- ➔ Requirements Engineering und Management für die Entwicklung in der Industrie
- ➔ Entwicklungsprozesse für Embedded-Systeme erfolgreich gestalten und optimieren
- ➔ Software-Projektmanagement: Erfolgreiches Führen von Projektteams durch alle Projektphasen
- ➔ ISTQB@ Certified Tester Foundation Level: Strukturiertes und effizientes Testen von Embedded- und IT-Systemen
- ➔ Funktionale Sicherheit (Safety) von Elektronik und deren Software: Umsetzung nach IEC 61508 und ISO 26262
- ➔ Embedded-Software-Test für C: Best Practices für den Unit-/Modul-/Komponenten-Test
- ➔ Agiles Testen und Test Driven Development von Embedded-Systemen
- ➔ Embedded-Linux für Tester, Support und Service

ANHANG

Weiterführende Informationen zu den Themen

Thema 1 - Standards und QM

Links:

Überblicksartikel zur Norm ISO 26262 bei ELEKTRONIKPRAXIS:

➔ http://bit.ly/XoTkdL

Bitte anklicken oder mit dem Smartphone einscannen

Die vollständige Norm ISO 26262 zum Download (kostenpflichtig): www.iso.org

Buchtipps:

➔ Zuverlässigkeit komplexer Systeme aus Hardware und Software, Willi Fuchs, ISBN: 3-410-32889-0

➔ Software-Qualität, Georg E. Thaller, ISBN 3-8007-2494-4

Trainings bei MicroConsult (www.microconsult.de):

➔ Funktionale Sicherheit (Safety) von Elektronik und deren Software: Umsetzung nach IEC 61508 und ISO 26262

➔ Software-Qualität: Erfolgsfaktor im Produktentstehungsprozess - Methoden zur erfolgreichen Projektumsetzung unter Berücksichtigung wichtiger Normen und Standards

Thema 2 - Safety als Strategie Qualität vor Funktionalität

Links:

Softwarequalität bei Heise.de

➔ http://bit.ly/Z8gPuE

Bitte anklicken oder mit dem Smartphone einscannen

Buchtipps:

➔ Basiswissen Softwaretest, Spillner, Linz, ISBN: 3-89864-178-3

➔ Die menschliche Seite des Projekterfolgs, Peter Siwon, ISBN: 3-898-64716-1

➔ Software automatisch testen, Dustin, Rashka, Paul, ISBN: 3-540-67639-2

Trainings bei MicroConsult (www.microconsult.de):

➔ ISTQB@ Certified Tester Foundation Level: Strukturiertes und effizientes Testen von Embedded- und IT-Systemen

➔ Funktionale Sicherheit (Safety) von Elektronik und deren Software: Umsetzung nach IEC 61508 und ISO 26262

➔ Embedded-Software-Test für C: Best Practices für den Unit-/Modul-/Komponenten-Test

➔ Agiles Testen und Test Driven Development von Embedded-Systemen

➔ Embedded-Linux für Tester, Support und Service

Thema 3 - Security als Strategie Schutz vor Viren, Würmern, Hackern & Co.

Links:

Bundesamt für Sicherheit in der Informationstechnik: www.bsi.de

Bitte anklicken oder mit dem Smartphone einscannen

Buchtipps:

➔ Software Inspection, Gilb, Graham, ISBN: 0-201-63181-4

➔ Viren Würmer und Trojanische Pferde, Andreas Winterer, ISBN-10: 3-815-82265-3

Trainings bei MicroConsult (www.microconsult.de):

➔ Security: Kryptografie richtig anwenden

➔ Requirements Engineering und Management für die Entwicklung in der Industrie

➔ Embedded-C: Effektiver Einsatz von Programmiermethoden und -tools für Embedded-Anwendungen

➔ Objektorientierte Softwareentwicklung: Spezielle Programmierprinzipien

➔ Design Patterns (nicht nur) für Embedded-Systeme

➔ Funktionale Sicherheit (Safety) von Elektronik und deren Software: Umsetzung nach IEC 61508 und ISO 26262

Thema 4 - Entwicklungsprozess

Einführung in die Entwicklung gemäß IEC 61508:

➔ http://bit.ly/TI4Tt3

Bitte anklicken oder mit dem Smartphone einscannen

Thema 5 - Die Zukunft im Blick

Links:

Nachbericht über BAIKEM-Netzwerktreffen: „Embedded Systems - Effizienzsteigerung durch modellbasierte Softwareentwicklung".

➔ http://bit.ly/11AxOHz

Bitte anklicken oder mit dem Smartphone einscannen

Buchtipps:

➔ Nationale Roadmap Embedded-Systems, Werner Damm, Reinhold Achatz, Klaus Beetz, Prof. Dr. Dr. h.c. Manfred Broy, Heinrich Daembkes, Klaus Grimm, Peter Liggesmeyer, ISBN: 978-3-642-14498-1

Informationsangebote von MicroConsult:

➔ Embedded Software Engineering Kongress, www.ese-kongress.de. Jährlich im Dezember. Über 100 Vorträge und Seminare aus Industrie und Forschung rund um das Thema Embedded Software Engineering aus Industrie und Forschung

Thema 6 - Qualitätsmerkmale

Links:

Artikel „Klassisches Requirements Engineering und agile Entwicklungsprozesse kombinieren"

➔ http://bit.ly/12FjY5T

Bitte anklicken oder mit dem Smartphone einscannen.

Artikel „Requirements Engineering für Embedded-Systeme"

➔ http://bit.ly/Va1Qga

Bitte anklicken oder mit dem Smartphone einscannen.

Buchtipps:

➔ Requirements Engineering und -Management, Chris Rupp, ISBN:3-446-21664-2

➔ Requirements-Led Project Management, Suzanne Robertson u.a., ISBN: 0-321-18062-3

Trainings bei MicroConsult (www.microconsult.de):

➔ Requirements Engineering und Management für die Entwicklung in der Industrie

➔ Entwicklungsprozesse für Embedded-Systeme erfolgreich gestalten und optimieren

➔ Software-Projektmanagement: Erfolgreiches Führen von Projektteams durch alle Projektphasen

➔ Funktionale Sicherheit (Safety) von Elektronik und deren Software: Umsetzung nach IEC 61508 und ISO 26262

Impressum

Herausgeber:
MicroConsult GmbH
Charles-de-Gaulle-Str. 6
81737 München
Tel. +49 89 450617-0
info@microconsult.de
www.microconsult.de

Projektleitung und Redaktion:

Peter Siwon,
MicroConsult
p.siwon@microconsult.de

Autoren:

Alexander Sedlak,
freier Journalist
alexander.sedlak@arcor.de

Marcus Gößler
MicrocConsult
m.goessler@microconsult.de

& Peter Siwon, MicroConsult

Lektorat:
Sabine Pagler