Mit DVD

iX DEVELOPER

2/2014

Auf der Heft-DVD:

Massig Software für Entwickler

IDEs: Code::Blocks, CodeLite, Eclipse, mbeddr, Visual Studio

Testversionen: Cantata, Intel System Studio

Tools: AVR Libc, GCC, GNAT GPL, GNU Binutils, MAST, openHAB, ProjectLibre, Valgrind

Bücher: Testen von Software und Embedded Systems (komplett), Software-Test für Embedded Systems (Auszüge)

Sponsored Software: Enterprise Architect 10, IDE für Raspberry Pi

Embedded Software

Ada, C, C++, MISRA C, Java, Pascal

Wie Programmiersprachen Echtzeit, Safety und Kompaktheit meistern

Modellierung

Systementwicklung mit SysML und UML

Embedded-Markt im Umbruch

Arduino, Raspberry Pi und BeagleBone

Industrie 4.0 und das Internet der Dinge

Durchbruch für die Heimautomatisierung

Qualitätssicherung

Wichtige Tools, Standards und Methoden

Debugging von Multicore-Systemen

STRATO

KEIN BLABLA

ECHTES HOSTING!
VON PROFIS FÜR PROFIS

NEU!
Verfügbarkeit: Load Balancer bieten maximalen Schutz und 99,94 % Verfügbarkeit durch echte Lastverteilung.

✓ ***Hat nicht jeder – bei uns selbstverständlich.***

NEU!
PFS (Perfect Forward Secrecy): Abhörsicherer E-Mail-Verkehr durch PFS-Verschlüsselung.

✓ ***Bei uns standardmäßig.***

NEU!
Performance Boost: Jetzt 30 % mehr RAM, SSDs in allen Datenbank-Servern und neue Caching-Technologie.

✓ ***Gibt's bei uns on top.***

NEU!
Sicherheit durch SSL: Standard- und Wildcard-Zertifikate einfach per 1-Click verfügbar – mit 256-Bit-Verschlüsselung.

✓ ***Bei uns echt günstig.***

30 Tage kostenlos testen!

PowerWeb Basic

Nach Testphase 12 Monate

0,– €/Mon.*

Getestet.de Webhosting-Anbieter Testsieger 1,9 (Gut) Test 02 / 2013

TÜV SÜD ISO 27001

HOSTED IN GERMANY 100% Qualität

STRATO.DE

Servicetelefon: 030 - 300 146 - 0

* Aktion vom 03.02. - 02.03.2014: PowerWeb Basic ganze 30 Tage kostenlos testen, Kündigung während der 30-tägigen Testphase jederzeit möglich. Danach 12 Monate für 0,- €/Mon., anschließend 4,99 €/Mon. Einmalige Einrichtungsgebühr 8,60 €. Mindestvertragslaufzeit nach Testphase 24 Monate. // STRATO AG, Pascalstr. 10, 10587 Berlin

Voneinander lernen

Softwareentwicklung für eingebettete Systeme war lange Zeit nur bedingt mit klassischer Softwareentwicklung zu vergleichen. Denn die anderen Anforderungen bei Embedded verlangten und verlangen auch heute noch besondere Lösungsansätze. Doch gerade in den letzten Jahren ist einiges passiert, was Embedded- und traditionelle Softwareentwicklung zusammenrücken lässt.

Vor einigen Jahren wurde das beim Besuch der Embedded World in Nürnberg durch Werbebotschaften zur Modellierung und zum Einsatz von Open-Source-Techniken à la Linux und Eclipse deutlich. Beides hatte zuvor schon in der klassischen Softwareentwicklung seinen Durchbruch. Etwas später folgte die Erfolgsgeschichte der Apps – streng genommen ist die allerdings stark von Usability-Aspekten geprägte Programmierung für mobile Betriebssysteme ja Embedded-Entwicklung. Den Siegeszug der Apps kann man als neueste Entwicklung im Auto-Infotainment beobachten.

Die Etablierung von Multicore-Architekturen ist ein weiterer Punkt. Parallelprogrammierung ist selbstverständlich auch in Embedded-Systemen angekommen und hier mancherorts besonders wichtig, da sich erst dadurch für eingebettete Systeme Anforderungen wie Echtzeit realisieren lassen. (Auf der parallel 2014, unserer Konferenz zur Parallelprogrammierung, haben wir auch deshalb einen Embedded-Schwerpunkt.)

Über die Jahre hinweg haben zudem vermehrt agile Prozessmodelle in die sich ansonsten dem Wasserfall-Modell verpflichtet fühlende Branche Einzug gehalten. Das sind nahezu alles Themen, die auch dieses Sonderheft zur Entwicklung eingebetteter Systeme beleuchtet.

Der neueste Trend – manche werde es Hype nennen – ist die Vernetzung unterschiedlichster Geräte, die in den Schlagwörtern „Industrie 4.0“, „Internet der Dinge“, „Smart Home“ und „(Cyber-)Physical Systems“ ihre Entsprechung haben. Quasi alle großen Software- oder Dienstleistungsanbieter, aber auch viele kleine setzen derzeit auf das dort gesehene Potenzial. Und so werden auf der diesjährigen Embedded World, aber auch auf der kurz darauf folgenden CeBIT und nicht zuletzt auf der HMI interessierte Besucher nicht darum herumkommen, sich mit einigen der damit verbundenen Techniken konfrontiert zu sehen. Selbst durch dieses Sonderheft ziehen sich diese Themen wie ein roter Faden.

Bei allen Ausnahmen, die Regeln bestätigen, mögen mir die Leser die Aussage verzeihen, dass diesen Entwicklungen mehr oder minder gemein ist, dass die Impulse von der klassischen Softwareentwicklung ausgingen. Das bedeutet nun aber nicht, dass die Embedded-Entwickler nur die Epigonen ihrer Kollegen aus der klassischen Softwareentwicklung sind. Gerade durch die Entwicklung nativer Apps und den Hype um Einplatinencomputer wie Arduino und Raspberry Pi wird das Interesse einer breiten Masse an hardwarenaher Microcontroller-Programmierung deutlich. Und viele Programmierer sind sicherlich fasziniert vom Kampf ihrer Embedded-Kollegen um weniger Bits und Bytes, die die Software auf einem Kleinstsystem erst möglich machen. Ganz zu schweigen davon, dass die Ideen und Erfahrungen von in der Embedded-Entwicklung gesetzten Disziplinen wie Requirements-Management, Software-Testing und funktionale Sicherheit für die klassische Softwareentwicklung lehrreich sein können.

Lehrreich wollen wir auch mit diesem *iX Developer*-Sonderheft sein. Wir wünschen Ihnen viel Spaß bei der Lektüre.

ALEXANDER NEUMANN

Embedded Software

Aufgrund langer Produktzyklen und hoher Fehlschlagsrisiken erwies sich die Embedded-Softwareentwicklung lange Zeit als vergleichsweise innovationsscheu. Das ändert sich nun mit Industrie 4.0 und Internet der Dinge – die Branche ist darauf gut präpariert.

ab Seite 7

Programmiersprachen

Am meisten ist in der Embedded-Programmierung nach wie vor C verbreitet, aber Ada, C++ und Java kommen ebenfalls zum Zug – teilweise deutlich zunehmend. Das liegt an neuartigen Anforderungen, für die sich das „klassische" C nicht eignet.

ab Seite 37

Qualitätssicherung

Dass bei Embedded-Systemen andere Maßstäbe bei der Qualitätssicherung angesetzt werden müssen, ist eine Binsenweisheit. Umso wichtiger ist es, dass Domänen wie Requirements Engineering, Testing oder funktionale Sicherheit mit neuen Werkzeugen und Prozessen reagieren.

ab Seite 61

Embedded im Umbruch

Die Vernetzung von Gegenständen aller Art zu einem Internet der Dinge schreitet fort. Unter dem Schlagwort Industrie 4.0 fließen die Erkenntnisse daraus in „intelligente Produktionseinheiten" ein. Doch auch abseits davon ist die hardwarenahe Embedded-Entwicklung en vogue.

ab Seite 129

Embedded im Umbruch

Diverses

Sonstiges

Alle Links: www.ix.de/ix1414SSS Artikel mit Verweisen ins Web enthalten am Ende einen Hinweis darauf, dass diese Webadressen auf dem Server der iX abrufbar sind. Dazu gibt man den iX-Link in der URL-Zeile des Browsers ein. Dann kann man auch die längsten Links bequem mit einem Klick ansteuern. Alternativ steht oben rechts auf der iX-Homepage ein Eingabefeld zur Verfügung.

Auf der Heft-DVD

Sponsored Software

Enterprise Architect 10

Das von SparxSystems hergestellte Softwaremodellierungswerkzeug Enterprise Architect ist ein sich auf UML 2.4.1 stützendes CASE-Tool für den Entwurf und zur Herstellung von Softwaresystemen, Geschäftsprozessmodellierung und zur Modellierung beliebiger Prozesse oder Systeme. Es deckt alle Teile des Entwicklungszyklus ab und ermöglicht die Nachvollziehbarkeit von Softwareprojekten. Die Enterprise Architect Trial Edition auf der Heft-DVD entspricht dem vollen Funktionsumfang und kann 30 Tage getestet werden.

IDE für Raspberry Pi

linutronix stellt eine auf Eclipse basierende IDE zur Entwicklung für Raspberry PI zur Verfügung. Sie ist auf Windows 7 getestet und installiert sich dort selbstständig. Diese auf der DVD bereitgestellte Version enthält alle Features, die für eine effiziente Arbeit notwendig sind. Leser können hiermit ein Projekt anlegen beziehungsweise verwalten (mit dem voreingestellten Namen „linutronix“).

IDEs

Code::Blocks 13.12

Die freie, quelloffene Entwicklungsumgebung ist für die Entwicklung mit C, C++, D und Fortran gedacht. Die IDE ist leicht zu konfigurieren und erweiterbar – zum Beispiel mit Autovervollständigung, Importfunktionen, Projekt-Vorlagen, Workspaces, Klassenbrowser und Quellcode-Faltung.

CodeLite 5.4

Die freie, quelloffene und plattformunabhängige Entwicklungsumgebung eignet sich für die Entwicklung mit C und C++. Sie bietet Projektverwaltung, Autovervollständigung, Restrukturierung, Syntaxhervorhebung, Code-Faltung, Subversion-Unterstützung, Integration des GNU Debuggers und ist über Plug-ins erweiterbar.

Eclipse

Auf der Heft-DVD finden Leser die weitverbreitete Entwicklungsumgebung in den Kepler-Distributionen für Java-, C/C++- und Automotive-Entwickler.

mbeddr

mbeddr nutzt JetBrains' Language Workbench MPS (Meta Programming System), die die notwendigen Mechanismen für inkrementelle, domänenspezifische Spracherweiterungen zur Verfügung stellt. Das Tool besteht aus einer IDE für C sowie Erweiterungen und stellt außerdem einen erweiterbaren Debugger zur Verfügung.

Visual Studio Premium 2013 + Visual C++ Redistributable

Microsofts integrierte Entwicklungsumgebung als ISO-Image. Die für 30 Tage gültige Testversion enthält Tools zum Planen, Entwickeln und Testen sowie für die Diagnose und die Ausführung von Anwendungen.

Tools

AVR Libc, GCC, GNAT GPL, GNU Binutils, MAST, openHAB, ProjectLibre, Valgrind

Testversionen

Cantata

Das Tool von QA Systems automatisiert und verwaltet komplexe Aufgaben beim Testen sicherheitskritischer Anwendungen. Die Automatisierung umfasst das Erstellen der kompletten Testumgebung, die Instrumentierung für Code-Coverage-Analysen, die Generierung eines Testfallgerüsts und die Ausführung von Testcases.

Intel System Studio 2014 Beta

Die Werkzeug-Suite zum Entwickeln eingebetteter und mobiler Linux-Anwendungen enthält Werkzeuge zum Analysieren, Debuggen und Kompilieren. Die Beta-Version auf der Heft-DVD ist bis zum 30. April 2014 lauffähig. Zu der Zeit liegt die um zum Beispiel Android-Unterstützung erweiterte fertige Test-Version vor, die dann 30 Tage lauffähig ist.

Literatur

Testen von Software und Embedded Systems

Der Autor des hier vollständig als PDF-Datei gebrachten Buches (dpunkt.verlag 2010 (2. Aufl.), stellt auf pragmatische Weise klassische und moderne Testverfahren vor. Dabei zeigt er auf 359 Seiten Lösungen für technische, analytische und methodische Probleme auf, die sich in der täglichen Arbeit umsetzen lassen.

Software-Test für Embedded Systems

Das Buch von Stephan Grünfelder (dpunkt.verlag 2013) beschreibt alle wichtigen praxistauglichen Methoden des Software-Tests für eingebettete Systeme. Auf der Heft-DVD finden Leser rund 140 Seiten Auszüge aus dem ansonsten 390 Seiten schweren Buch.

Listings und Lizenzen

Die Listings zu den Heftartikeln und die Lizenzen zu den Softwarepaketen auf der Heft-DVD.

Hinweis für Käufer

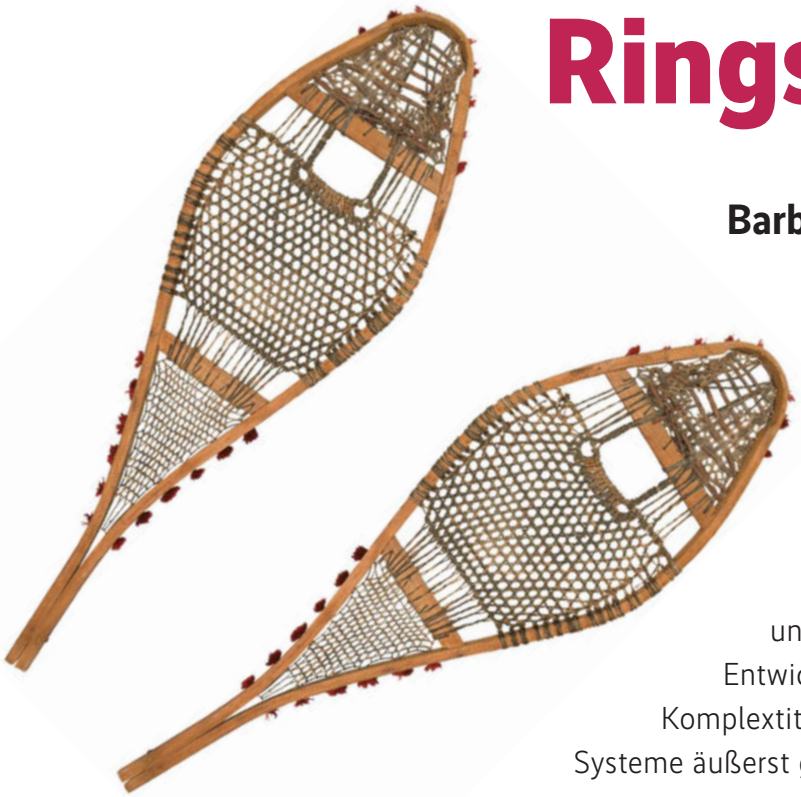
- PDF- und iPad-Version: In der *iX*-App finden Sie einen Button zum Download des DVD-Images.
- PDF-E-Book: Folgen Sie im Browser der unter „Alle Links“ angegebenen URL.

Alle Links: www.ix.de/ix1414006

Introducing into …

Aufgrund langer Produktzyklen und hoher Fehlschlagsrisiken erwies sich die Embedded-Softwareentwicklung lange Zeit als vergleichsweise innovationsscheu. Das ändert sich nun mit Industrie 4.0 und Internet der Dinge – die Branche ist darauf gut präpariert.

Trends der Embedded-Entwicklung



Ringsum

Barbara Lange

Sie sind überall: Im Auto, im Lichtschalter, im Düsenjet, im Herzschrittmacher, in der Jalousie oder in der Produktionsmaschine. Unsichtbar und oft unmerklich erfüllen Embedded-Systeme ihre Aufgaben, und es werden immer mehr. Entwickler sind durch die steigende Komplextität der einzelnen Systeme äußerst gefordert.

Die Vernetzung von Gegenständen aller Art zu einem Internet der Dinge schreitet fort. So kam auf der Consumer Electronics Show (CES) 2014 kaum ein Gerät ohne Sensor oder Netzverbindung aus, sei es nun Zahnbürste, Hundehalsband oder Auto [a]. Auch Google lässt sich keine Vernetzung entgehen: Durch den Kauf des Startups Nest Lab Inc. ist kürzlich zum autonomen Auto und der Datenbrille Google Glass noch die Haustechnik hinzugekommen. Nest bietet vernetzte Haustechnik wie Rauch- und Kohlenmonoxidmelder sowie Thermostate. Schon klar, dass viele angesichts der Datenkraken-eigenschaften des Suchmaschinenanbieters und des NSA-Skandals dabei ins Grübeln kommen.

Top-Thema: Das Internet der Dinge

Auch große Hersteller setzen auf Vernetzung im Stil des Internet der Dinge: So hat Bosch nur wenige Millimeter große MEMS-Sensoren (Micro Electro Mechanical Systems) entwickelt, die mit mikroskopisch feinen Strukturen Beschleunigung, Luftdruck, Erdmagnetfeld, Geräusche, Drehraten oder Temperatur messen. Diese Daten können sie über eine winzige Funkschnittstelle drahtlos übertragen, zum Beispiel auf das Smartphone des Nutzers. Als Energiespenderin dient eine Miniaturbatterie. Außerdem hat das Unternehmen mit „Bosch Connected Devices and Solutions“ eine Gesellschaft für das Internet der Dinge gegründet, die vernetzte Geräte samt Software entwickeln will. Zunächst soll der Schwerpunkt auf dem vernetzten Haus und dem Bereich Transport, Logistik und Verkehr liegen.

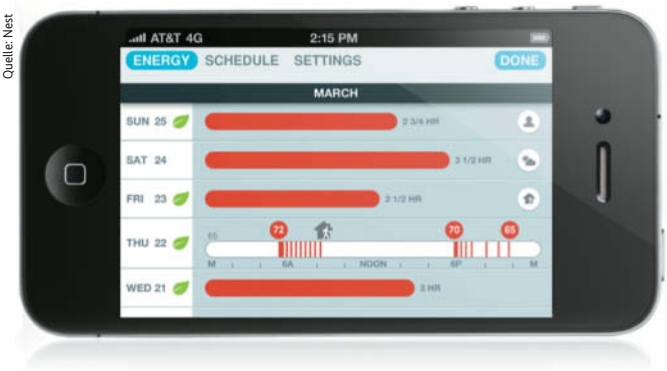
Klein geworden ist die Technik für das selbstfahrende Auto bei Audi: Auf der CES 2014 zeigte der Automobilhersteller ein autonomes Auto, dessen Elektronik, die 2013 noch einen Kofferraum füllte, nun auf eine Platine passt. Das Steuergerät namens Z-Fas wertet die Signale von verschiedenen Sensoren und Kameras aus und steuert Gas, Bremse, Schaltung und Lenkung [b].

Außerdem wollen Automobilhersteller und Google das Smartphone-Betriebssystem Android stärker in Fahrzeuge integrieren. Mit diesem Ziel haben Audi, General Motors, Honda, Hyundai, Nvidia und das omnipräsente Unternehmen Google die Open Automotive Alliance (OAA) [c] gegründet. Sie will bestehende Smartphones besser anbinden und eine angepasste

Quelle: Nest

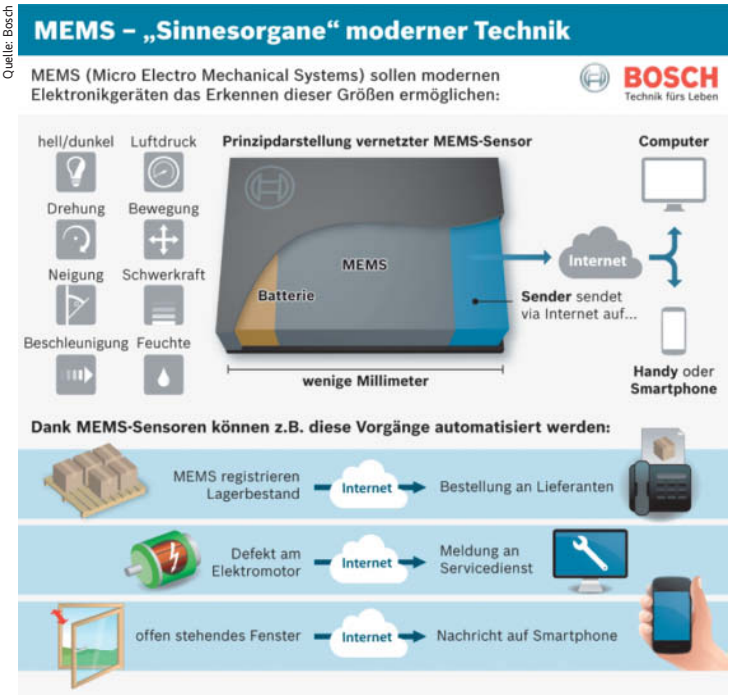
Ein netzfähiges Thermostat für die Haustechnik vom Unternehmen Nest. Anfang 2014 kaufte Google das Startup und stieg damit in das Smart Home ein (Abb. 1).

Quelle: Nest



Die netzfähigen Rauch-Kohlenmonoxidmelder und Thermostate schicken ihre Daten weiter, zum Beispiel an das Smartphone des Nutzers. Oder auch an Google (Abb. 2).

Quelle: Bosch



Prinzipdarstellung eines vernetzten MEMS-Sensors. Bosch entwickelt MEMS-Sensoren, die mit mikroskopisch feinen Strukturen Beschleunigung, Luftdruck, Erdmagnetfeld, Geräusche, Drehraten oder Temperatur messen und die Daten über eine winzige Funk-Schnittstelle übertragen (Abb 3).

Android-Version als Basis künftiger Infotainment-Systeme herausbringen.

Auf die Entwickler von Embedded-Systemen kommt also einiges zu. Für sie steigen die Herausforderungen, was sich unter anderem an der Komplexität der Systeme zeigt. Embedded-Systeme führen zunehmend mehr Funktionen in einem Gerät aus, haben aber nur begrenzte Ressourcen, Speicher und Energie zur Verfügung. Außerdem verlangen viele Anwendungen vorhersagbare Reaktionszeiten in Echtzeit. Die sich selbst steuernden Adhoc-Netze im Umfeld des Internet der Dinge stellen noch einmal ganz eigene Anforderungen. Mehrere Artikel in diesem Heft gehen auf diese Aspekte konkret ein. Die Autoren stellen beispielsweise die Betriebssysteme für das Internet der Dinge namens Contiki, TinyOS und RIOT vor (s. Artikel auf S. 136), die für Systeme mit Taktfrequenzen im einstelligen Megahertz-Bereich und nur wenige Kilobyte RAM ausgerichtet sind, wie es bei Sensornetzen häufig der Fall ist. RIOT ist für Echtzeitanwendungen optimiert.

Vernetzte Industrie braucht Sicherheit

Während das Internet der Dinge als Oberbegriff die Vernetzung von Objekten zusammenfasst, bringt Industrie 4.0 diese Vernetzung in die Produktion [1]. Noch sind große Teile des Konzepts eine Vision, denn für die Realisierung sind noch viele Dinge zu klären. Zum Beispiel gilt die interdisziplinäre Zusammenarbeit als eine große Herausforderung, denn jeder Arbeitsbereich hat seine eigene Sicht auf die Produktion. Auch müssen Standards und Normen unter anderem die Interoperabilität von Geräten verschiedener Hersteller gewährleisten. Einen Überblick gibt der Artikel auf S. 130. Er erläutert außerdem den Standard für die Automatisierungsbranche, OPC-UA (Unified Architecture), auf den sich mehr als 470 internationale Firmen in der OPC Foundation geeinigt haben.

Programmierer benötigen im Umfeld von Cyber-Physical Systems ein Spezialwissen, beispielsweise für die Arbeit mit Field Programmable Gate Arrays (FPGA). Dass grafische Werkzeuge hier quasi als „Hirnverstärker“ wirken und auch Ingenieuren ohne diese Spezialerfahrungen die FPGA-Programmierung öffnen, erläutert der Artikel auf Seite 164 anhand der Design- und Entwicklungsplattform LabVIEW von National Instruments. Die bildliche Darstellung unterstützt besonders die parallele Denkweise des Programmierers.

Natürlich ist die Sicherheit industrieller Anlagen ein Top-Thema, auch in der heutigen Industrie 3.0 noch. Wenn sich die Produktion mit der Außenwelt via Internet verbindet, umso mehr, denn im Rahmen von Industrie 4.0 werden die industriellen Netze physisch mit den IT-Netzen verbunden sein. Damit sind industrielle Netze prinzipiell den gleichen Risiken ausgesetzt wie die IT in Büros.

Mangelnde Sicherheit hängt aber auch mit internen Unzulänglichkeiten zusammen, wie ein Bericht von Kollegen der Schwesterzeitschrift c’t im Mai 2013 gezeigt hat: Der Zugriff auf Hunderte von Industrieanlagen, darunter Fernwärmekraftwerke, wichtige Rechenzentren, eine Justizvollzugsanstalt und ein Stadion war ganz leicht, da sie zwecks Fernwartung via Internet sperrangelweit offen standen [2].

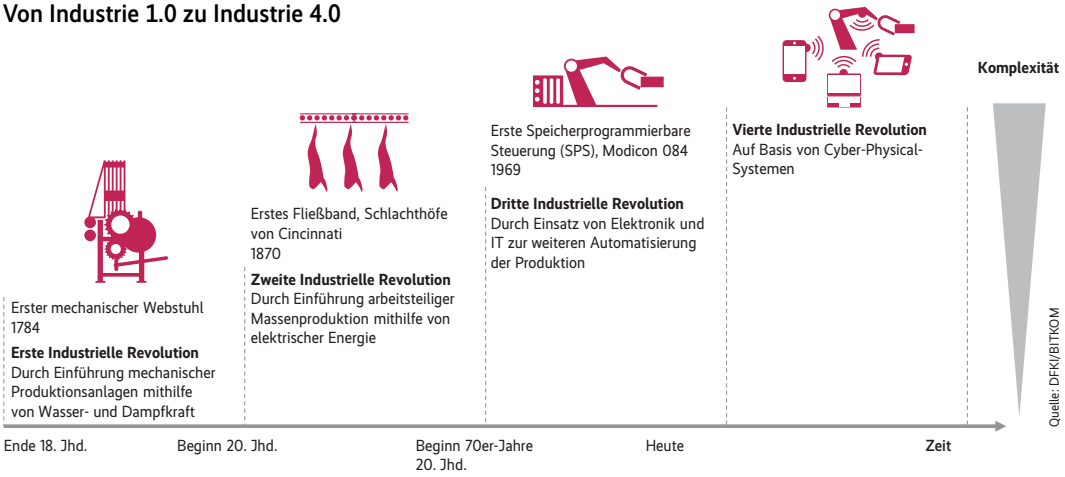
Viele Unternehmen haben in puncto Sicherheit noch Nachholbedarf, was auch der Verband Deutscher Maschinen- und Anlagenbau (VDMA) kürzlich herausgefunden hat: 63 Prozent von 70 befragten Mitgliedsunternehmen des Verbands erwarten eine steigende Anzahl an „Security-Vorkommnissen“. Bei 29 Prozent haben sie bereits zu Produktionsausfällen geführt. Nur 57 Prozent der Unternehmen kennen allerdings einen der gängigen Security-Standards und weniger als ein Drittel wendet diese Standards an [d].

Um das Risiko von Manipulation und Ausspähungen bei der Machine-to-Machine-Kommunikation zu verringern, fordert der Verband der deutschen Internetwirtschaft eco, dass Hersteller bei den Steuerungseinheiten nicht nur Garantien für die Hardware übernehmen, sondern zusätzlich Software-Updates und Bugfixes anbieten.

Standards regeln funktionale Sicherheit

In sicherheitskritischen Systemen, zum Beispiel in Flugzeugen oder Fahrzeugen, müssen viele Anwendungen strenge Anforderungen an die funktionale Sicherheit nach Normen wie IEC 61598 erfüllen, damit es nicht zu Fehlfunktionen kommt. Aus IEC 61598 haben einige Branchen eine auf sie zugeschnittene Version abgeleitet, darunter die Automobilindustrie mit ISO 26262. Denn im Auto müssen Steuergeräte, die die Funk-

Von Industrie 1.0 zu Industrie 4.0



Während die Internet-Leute noch bei Version 2.0 oder höchstens 3.0 sind, ist der Zähler in der Industrie schon eine Stufe weitergerückt. Man rechnet dort in industriellen Revolutionen (Abb. 4).

tionen des Motors oder der Bremse regeln, unbedingt richtig funktionieren. Eine nach eigenen Angaben erste SIL-4-Zertifizierung für Echtzeitbetriebssysteme auf Mehrkern-Prozessoren hat das Betriebssystem PikeOS von Sysgo erhalten. Der Sicherheits-Integritätslevel (SIL) definiert Kriterien für die funktionale Sicherheit. SIL 4 ist die höchste Stufe. Aufgrund fehlender Zertifizierungen haben Anwendungen mit Mehrkern-Prozessoren oft nur einen Kern genutzt. Nun können laut Sysgo auch sicherheitskritische Anwendungen von der Leistungsfähigkeit von Mehrkern-Prozessoren profitieren.

Zur Klassifizierung der Sicherheitslevel führt ISO 26262 unter anderem den Begriff ASIL (Automotive Safety Integrity Level) ein, der von ASIL A bis D als höchsten Sicherheitslevel die funktionale Sicherheit gewährleisten soll. Der Artikel auf Seite 80 gibt einen Überblick über Aspekte der funktionalen Sicherheit sowie AUTOSAR und geht besonders darauf ein, wie Entwickler von Steuergeräten Software aus Projekten übernehmen können, die nach älteren AUTOSAR-Standards erstellt wurden. AUTOSAR soll den Austausch von Software auf Steuergeräten und ihre Wiederverwendbarkeit erleichtern.

Für sicherheitskritische Systeme, die gesetzlichen Auflagen unterliegen wie in den Bereichen Automotive, Luftfahrt oder Medizin gelten außerdem sehr hohe Anforderungen an Tests und an das Requirements Engineering, wie mehrere Artikel in diesem Heft konkret erläutern.

Parallelität und Modellierung

Außerdem stellen Mehrkernprozessoren besondere Hürden auf: Da die Ansprüche an Embedded-Software steigen, reicht ein Prozessor oft nicht mehr aus. Mit Mehrkernprozessoren können viele Funktionen, die zuvor auf mehreren Geräten verteilt lagen, nun in einem Gerät laufen. Das erschwert aber die ohnehin schon schwierige Aufgabe des Debugging weiter, wie der Autor des Artikels auf Seite 77 ausführt. Denn nun laufen viele Prozesse parallel und beeinflussen sich gegenseitig, da sie sich Ressourcen wie Hauptspeicher, Cache, Busse oder Peripherie teilen müssen. Dadurch können Wechselwirkungen zwischen Programmen entstehen. Entwickler stehen vor der Aufgabe, die Rechenoperationen zu koordinieren. Die meisten Werkzeuge sind aber auf die neuen Herausforderungen nur unzureichend vorbereitet. Noch sind die Reproduzierbarkeit von Fehlern und das Debugging zeitkritischer Software nicht zufriedenstellend gelöst. Abhilfe schaffen könnte das Tracing, das das Verhalten der Software aufzeichnet, sodass man anschließend in Ruhe analysieren kann.

In der Embedded-Programmierung hat sich der modellgetriebene Ansatz (Model Driven Development) etabliert, um den Ansprüchen einer Wiederverwendung und Abstraktion von Software entsprechen zu können. Ein Beispiel dafür ist der bereits genannte Standard AUTOSAR in der Automobilindustrie, der versucht, das Verhalten von Steuergeräten auf einem hohen Abstraktionsniveau anhand von Modellen zu beschreiben.

Programmiersprachen für Embedded Systems

Mit der Unified Modeling Language (UML) beschäftigen sich mehrere Artikel, zum Beispiel der auf Seite 86, der am Beispiel einer Temperaturüberwachung und -regelung die Konzepte der modellbasierten Systemerstellung zeigt. Außerdem geht er darauf ein, wann eine Codegenerierung aus UML in C für eingebettete Systeme sinnvoll ist. Wie gesagt, Embedded-Systeme werden zunehmend komplexer. Die Entwicklungszyklen sind lang. Für den Markterfolg ist aber Schnelligkeit gefragt. Ob sich dafür agile Methoden wie Scrum oder Kanban eignen, diskutiert der Artikel auf Seite 158. So stand die Automobilindustrie zwar agilen Methoden vor einiger Zeit noch skeptisch gegenüber, testet aber diesen Ansatz mittlerweile auch in der Fahrzeugentwicklung schrittweise.

Eine eigene Rubrik beschäftigt sich mit Programmiersprachen. Am meisten verbreitet ist nach wie vor C, aber C++, Pascal, Ada oder Java kommen ebenfalls zum Zug. Unterstützend in der Programmierung sicherheitskritischer Bereiche wirkt MISRA, ein Kodierstandard, der ursprünglich von der Automobilindustrie erarbeitet wurde. Der Standard sollte anhand von Regeln für C gewährleisten, dass eingebettete Software in Komponenten unterschiedlicher Hersteller fehlerfrei


Quelle: Tinkerforge

Mit stapelbaren Microbausteinen von Tinkerforge kann man Anwendungen durch das Zusammenstecken realisieren, zum Beispiel eine Überwachung des Serverraums (Abb. 5).

funktionieren. Mittlerweile ist MISRA zu einem Quasi-Kodierungsstandard für Sicherheits- und lebensentscheidende Anwendungen geworden, und seit März 2013 gibt es die Version MISRA C:2012, die auch die C-Variante C99 unterstützt. Im Artikel auf Seite 54 stellt der Autor die aktuelle Version vor und diskutiert die Frage, wie man sie richtig einsetzt.

Die Programmiersprache Ada bietet nützliche Features für das Speichermanagement, wie man im Artikel auf Seite 38 erfahren kann, denn Ada beschreibt nicht nur die Struktur der Daten, sondern auch ihre Anordnung im Speicher. Für Embedded-Programmierer ist das Speichermanagement besonders wichtig, da die Ressourcen begrenzt sind. Der Autor beschreibt anhand praktischer Beispiele, wie man diese Funktionen nutzen kann. Dass man Embedded-Systeme nicht unbedingt in C oder Assembler programmieren muss, sondern auch C++ hierfür nehmen kann, erläutert der Artikel auf Seite 48 konkret am Beispiel von „Blinking Lights“, der Embedded-Variante des klassischen Beispiels „Hello, World“.

Java wird durch spezielle Erweiterungen wie die Java Virtual Machine JamaicaVM von aicas hart echtzeitfähig. Mit Java ME adressiert Oracle mittlerweile den Markt der Embedded Devices und das Internet der Dinge. Nach der Einschätzung von Branchenkennern hat Java hier aufgrund der vorherrschenden zahlreichen proprietären Lösungen und Protokolle und des prognostizierten Wachstums eine echte Chance [e].

Gemeinsam mit ARM will Oracle in den nächsten Jahren außerdem die Standard Edition der Java-Plattform (Java SE) für ARM-32-Bit-Plattformen anpassen und den Einsatz mit ARMv8-64-Bit ermöglichen. Außerdem ist das Java 7 JDK in das Repository der Rasperry Pi Foundation eingezogen. Mehrere Artikel beschäftigen sich mit Themen rund um Java. Wie man die Sprache C erweitern kann, beschreibt der Artikel auf Seite 92 am Beispiel einer Heizungssteuerung mit Arduino und der Open-Source-Software mbeddr, die auf JetBrains MPS basiert. Mit mbeddr kann man bestehende Sprachen erweitern oder neue erstellen.

Mit dem für die Embedded-Entwicklung zentralen Thema Echtzeit beschäftigt sich ein Überblick (s. S. 21) über den Markt für Echtzeit-Betriebssysteme. Auch Linux ist seit der Kernel-Version 2.6.24 echtzeitfähig geworden und eignet sich vor allem für nicht sicherheitskritische Anwendungen. Die Produkte decken die vielfältigen Bedürfnisse unterschiedlicher Branchen, Sicherheitsstufen, Hardwarearchitekturen und Einsatzbereiche ab.

Alles auf einer Platine

Neue Aspekte für die Embedded-Welt halten die Einplatinen-Rechner wie Raspberry Pi oder Arduino bereit. Mit ihnen können auch Einsteiger in die Mikroelektronik einsteigen. Außerdem entstehen durch Erweiterungen der offenen Boards zunehmend Anwendungsmöglichkeiten für professionelle Entwickler. Zum Beispiel koppelt eine Brücke namens GertDuino die beiden Einplatinen-Rechner. Mit stapelbaren Microcontroller-Bausteinen namens Bricks und Bricklets von Tinkerforge lassen sich auch kleinere industrielle Anwendungen schnell realisieren [3].

Mehrere Artikel in diesem Heft beschäftigen sich mit diesen Open-Source-Hard- und Softwareplattformen. So geht der Artikel auf Seite 12 auf einen Disput auf einer Konferenz zwischen zwei „Lagern“ ein: den Arduino-Fans und den etablierten Embedded-Entwicklern. Ist Kreativität wichtiger als die technische Effizienz des Systems? Leidet die Qualität von Systemen darunter, dass nun auch durchschnittliche Programmierer oder die sogenannten Maker Steuerungen realisieren können? In vielen Bereichen löst sich die Embedded-Szene vom Bitzählen, da die Ressourcen, Speicher und Rechenleistung zwar immer noch begrenzt, aber doch tendenziell umfangreicher werden.

Onlinequellen

[a] „Internet der Dinge“: Dinge sind da, aber sie sprechen nicht miteinander
http://heise.de/-2083768.html

[b] Audi und BMW automatisieren das Auto
http://heise.de/-2076764.html

[c] Open Auto Alliance
www.openautoalliance.net

[d] VDMA-Studie: Status Quo der Security in Produktion und Automation 2013/2014 (November 2013)
pks.vdma.org/documents/105969/142443/VDMA+Studie+Security/82324cfa-2df6-4c4e-ae21-490a26e30d0c

[e] Lars Röwekamp; Kommentar: Vom „Feature-Phone“ zum „Internet der Dinge“ – die Renaissance von Java ME
http://heise.de/-1968194.html

Fazit

Wer Systeme für eingebettete Systeme erstellt, hat es mit vielen aktuellen Themen zu tun: So scheint die Branche derzeit entschlossen, das Internet der Dinge in allen Bereichen umzusetzen, sei es nun im Auto, im Haushalt oder in der Produktion. Zudem werden Sicherheitsfragen noch wichtiger, als sie es ohnehin schon sind. Leser bekommen in diesem Sonderheft ganz konkrete Informationen und Anregungen, wie sie Embedded-Systeme, die etwa mehrere Prozessoren nutzen, vorhersehbar in Echtzeit reagieren und Standards der funktionalen Sicherheit erfüllen müssen, erfolgreich entwickeln, debuggen und testen können. Aber nicht überall gelten so hohe Sicherheitsanforderungen wie in Fahrzeugen, Flugzeugen oder bei medizinischen Geräten. Hier eröffnen die Einplatinenrechner Arduino, Raspberry Pi und Co. auch Nicht-Experten einen Einstieg in die Embedded-Welt. (ane)

Literatur

[1] Barbara Lange; Alles autonom; Die Produktion steuert sich selbst; *iX* 7/2013, S. 108

[2] Louis-F. Stahl; Gefahr im Kraftwerk; Industrieanlagen schutzlos im Internet; c’t 11/2013; S. 78 ff.

[3] Barbara Lange, Offene Pläne, Professionelle Ansätze für Open-Source-Hardware, *iX* 2/2014, S. 86

[4] Andreas Graf; Aus einer anderen Welt; Softwareentwicklung für eingebettete Systeme; in iX Special 1/2010 „Programmieren heute“, S. 132

[5] Barbara Lange; Entkoppelt; Trends der Embedded-Entwicklung; in iX Developer 1/2013 „Programmieren heute“, S. 159

Barbara Lange
ist IT-Journalistin und Inhaberin des Redaktionsbüros kurz und einfach in Lengede.

Alle Links: www.ix.de/ix1414008

Embedded-Programmierung im Umbruch

Scharf argumentiert

Tam Hanna

Aufgrund langer Produktzyklen und hoher Fehlschlagsrisiken erweist sich der Embedded-Bereich als vergleichsweise innovationsscheu. Doch gilt hier, dass die Zeit des Zählens von Bits so langsam vorbei ist. Das liegt auch an Systemen wie Arduino und Raspberry Pi, die einen einfacheren Einstieg in Embedded ermöglichen.

Eine kleine Veranstaltung in Berlin wurde vor einigen Monaten zur Bühne für einen handfesten Generationenkonflikt in Sachen Embedded-Programmierung: Im Rahmen einer QA-Session gerieten zwei Streitparteien aneinander, die unterschiedlicher nicht sein können. Auf der einen Seite stand Massimo Banzi, der Miterfinder des Einplatinen-Computers Arduino, auf der anderen ein pensionierter Rüstungsmanager einer osteuropäischen Miliz.

Der Disput fand seinen Auslöser durch Banzis Aussage, dass sein Computersystem jedem den Zutritt in die faszinierende Welt der Mikroelektronik ermöglichen würde. Dank diverser Hochsprachen seien nun auch Künstler, Grafiker und Architekten zum Konstruieren von Gadgetry befähigt – und das ohne langwieriges Studium der internen Architektur des verwendeten AVR-Controllers. Von Seiten des Managers kam daraufhin der Einwand, dass die damit erstellten Produkte aus technischer Sicht nur mangelhaft sein könnten – wer die interne Architektur der zugrunde liegenden MCU (Microcontroller Unit) nicht verstehe, erstelle keine effizienten Applikationen. Der wortgewandte Italiener erwiderte darauf sinngemäß, dass ihm die Effizienz völlig egal sei – wichtig sei die Kreativität, die durch seine Hardware freigesetzt werde.

Das Publikum teilte sich schnell in zwei Gruppen auf. Die Mehrheit der Teilnehmer der Konferenz schlug sich aufseiten des Arduino-Schöpfers. Die verschwindend geringe Minderheit fokussierte ihre Kritik vor allem an der als Entwicklungsumgebung verwendeten Sprache Processing. Diese ist alles andere als hardwarenah, der aus der Hochsprache generierte Assembler-Code lässt sich nicht ohne Weiteres nachvollziehen.

Wieso Processing?

ARM-Prozessoren haben eine derartig komplexe interne Architektur, dass das Erstellen einer in Assembler gehaltenen Anwendung nur für die diensterfahrenen Softwareentwickler realisierbar ist. Durchschnittliche Programmierer sind mit dem Erlernen des Hardwareaufbaus zumeist überfordert, der Einsatz einer Hochsprache ist deswegen in den häufigsten Fällen gerechtfertigt.

Arduinos basieren auf ATMega-Prozessoren aus dem Hause Atmel. Sie sind mit Sicherheit etwas komplexer als der althergebrachte PIC16F84A von Microchip – die interne Architektur lässt sich trotzdem ohne allzu große Probleme an einem Nachmittag begreifen. Aufgrund der eher geringen Rechenleistung und des einfachen Aufbaus ist der Einsatz von Hochsprachen aus technischer Sicht alles andere als sinnvoll. Zudem „übt“ das Programmieren in Assembler das Gehirn der Entwickler. Ein kleiner Teil der Kritiker ging aus diesem Grund sogar so weit, Banzi vorzuwerfen, dass er dafür verantwortlich wäre, dass die nachkommenden Programmierer zunehmend weniger „praktische Erfahrung“ mitbrächten. Die Mehrheit der Anwesenden ging jedoch davon aus, dass die Kreativität der Nutzer wichtiger sei als die technische Effizienz des resultierenden Systems.

Das beste Gegenbeispiel fand sich einige Wochen später im nahen Umfeld des Autors. Eine technisch nicht sonderlich begabte Lebensgefährtin eines Elektronikers nutzte einen Arduino, um ihre Kaffeemaschine vom Bett aus fernsteuern zu können – wenn die Dame den oberen Stock ihrer Wohnung in Richtung Küche verlassen hatte, stand eine Tasse Betriebsstoff bereit.

Werte schaffen

Mit Sicherheit hätte sich dieser Aufbau auch mit einem PIC16F84A und Assembler realisieren lassen; das resultierende Gerät wäre um einige Cent billiger und würde sicherlich mit etwas weniger Rechenleistung auskommen. Der springende Punkt liegt an anderer Stelle. Die Dame konnte ihre Fernsteuerung selbst – das bedeutet ohne Hilfe von Technikern oder ihres Mannes – konstruieren. Dadurch schuf sie eigenständig einen Wert, den sie ohne Arduino niemals hätte realisieren können. Ihr Leben wurde somit um ein technisches Gerät reicher, das sie sonst nicht (oder nur mit viel Aufwand) hätte erhalten können.

Auf gesamtgesellschaftliche Sicht gesehen entstand dadurch eine nicht unwichtige Veränderung. Die Nutzerin lernte, dass sie durch Wille, Fleiß und Einarbeitung ein komplexes technisches Gerät realisieren kann.

Zeit kostet Geld

Beim Deployment von Arduino und Co. darf ein „weicher" Vorteil nicht unterschätzt werden. Für Quereinsteiger vorgesehene Entwicklungsumgebungen bringen im Allgemeinen weitaus mehr Debugger-Unterstützung mit – wer einmal einen Fehler in einem Programm für einen PIC16F84A gesucht hat, kann ein Lied über diese mühevolle und wenig ergiebige Sisyphusarbeit singen. Die dazu notwendige Zeit kann bei kleinen Produktionsmengen den Preis für teurere Prozessoren mehr als aufwiegen.

Doch damit nicht genug: Die meisten Arduino-Konfigurationen bestehen aus Kombinationen von tausendfach erprobten Komponenten. Diese sind oft von mehreren Anbietern verfügbar – bei elektrischen Problemen ist es leicht, technische Unterstützung zu finden. Ein selbst konstruiertes Mainboard kann hier in mehrerlei Hinsicht nicht mithalten. Erstens ist es heute alles andere als einfach, Kleinserien zu beordern: Wenn man die Platinen nicht selbst zusammenbauen möchte, kostet schon allein das Ätzen des Mainboards ein kleines Vermögen. Zweitens ist man bei einer Eigenentwicklung nie vor simplen Fehlern gefeit. Softwarefehler lassen sich vergleichsweise einfach reparieren – das Entwerfen einer neuen Platine kostet Länge mal Breite. Zu guter Letzt besteht noch das Risiko von Konflikten mit den Regulierungsbehörden.

Logische Abwehrreaktionen

Etablierte Entwickler von Embedded-Systemen stehen diesen Gedanken mit Sicherheit alles andere als erfreut gegenüber. Dabei handelt es sich – zumindest bis zu einem gewissen Grad – um eine logische Abwehrreaktion, die die Verteidigung der eigenen Pfründe avisiert. Solange das Entwerfen von Embedded-Steuerungen eine komplizierte Aufgabe darstellt, lässt sich mit Beratungsdienstleistungen viel Geld verdienen. Wenn erst einmal jeder durchschnittliche Programmierer zum Realisieren eingebetteter Steuerungen befähigt ist, sieht die Situation völlig anders und für die Berater alles andere als befriedigend aus.

Viele Unternehmen gehen beim Verteidigen ihrer Pfründe sogar so weit, dass sie ihre Kunden mit Absicht schädigen. Der Einsatz programmierbarer Suchköpfe könnte im Rüstungsbereich zu einem Quantensprung führen – die etablierten Unternehmen setzen nur deshalb nicht auf diese Technologie, da sich ihre Margen dadurch wesentlich verschmälern würden.

Pragmatismus ist immer falsch

Eine alte Weisheit im Bereich des Software Engineering besagt, dass es keine silbernen Kugeln gibt. Das gilt auch für die Entwickler von Embedded-Systemen: Bei einem für die Massenfertigung vorgesehenen Produkt ergibt das Einsparen einiger Cents an Hardwarekosten Sinn. In diesem Fall wäre die Verwendung von Assembler und einem „kleinen" Controller mit Sicherheit gerechtfertigt – für ein in Kleinserie produziertes System sieht die Kalkulation naturgemäß völlig anders aus.

Daraus folgt, dass das Erlernen einer Assembler-Sprache meist keine vergebene Liebesmüh ist. Ein in Elektronik versierter Programmierer findet in den meisten Fällen jede Menge an Aufträgen. Für einen in Hochsprachen entwickelnden Programmierer ist die Lage weitaus weniger eindeutig: Heutige Compiler sind so weit von der Hardware entfernt, dass der Assembler-Code kaum mehr nachvollziehbar ist. Zudem unterscheidet sich die Hardware eines ARM- oder x86-Prozessors so stark von einem primitiven Microcontroller, dass Abstraktionen nicht mehr ohne Weiteres möglich sind.

Das Ende einer Ära

Die von Banzi vertretene Sicht der Dinge ist genauso falsch wie die des pensionierten Managers. Der beste Weg liegt, wie so oft, in der Mitte. Auch wenn es für den einen oder anderen mit Assembler aufgewachsenen Veteranen schwer zu akzeptieren ist: Die Branche lebt heute in einer Zeit, in der Speicher und Rechenleistung nur mehr eine untergeordnete Rolle spielen. Wer seine Programmiererkarriere unter Palm OS begann, zeichnete in der Anfangszeit Speicherlayouts auf kariertes Papier. Beim Palm IIIc war das nicht mehr notwendig – heutige Smartphones haben ein oder zwei Gigabyte Arbeitsspeicher.

Der Embedded-Bereich erweist sich aufgrund langer Produktzyklen und hoher Fehlschlagsrisiken als vergleichsweise innovationsresistent. Trotzdem gilt auch hier, dass die Zeit des „Bitzählens" vorbeigeht. Schnelle Arduinos rechnen Kreise um klassische Microcontroller. Die auf Seite 146 vorgestellten Einplatinen-Computer bieten Rechenleistungen, die vor zehn Jahren im Workstation-Bereich üblich waren.

Avioniker erlebten vor einigen Jahrzehnten eine ähnliche Umstellung: Analogrechner galten als erprobt, konnten aber mit den Leistungen ihrer digitalen „Kollegen" nicht mithalten. Im Laufe der Zeit stellten alle Unternehmen um – wer als Erster „sprang", konnte sich im Rennen um die Technologieführerschaft bessere Plätze sichern. (ane)

Tam Hanna
befasst sich seit der Zeit des Palm IIIc mit Programmierung und Anwendung von Handheldcomputern. Er entwickelt Programme für diverse Plattformen, betreibt Onlinenews-Dienste zum Thema und steht für Fragen, Trainings und Vorträge gern zur Verfügung.

bbv
Software Services

Attraktiv und einfach ist das Infoterminal der intelligenten Paketbox plus. bbv entwickelte in kürzester Zeit die Software für den Prototypen der René Koch AG. Das agile Vorgehen bewährte sich, die Funktionalität und die Qualität überzeugen.

www.bbv.ch

Luzern · Zug · Bern · Zürich · München

Werkzeuge und Methoden für die
Embedded-Softwareentwicklung mit Open Source

Rutsch- und schlagfest

Andreas Graf

Ein Ende der Trends „zunehmende Komplexität", „stärkere Vernetzung" und „Parallelität" ist in der Entwicklung eingebetteter Systeme nicht abzusehen. So erweitern Industrie und Forschung permanent die vielfältige Werkzeuglandschaft, die eine breite Palette von Methoden absteckt. Zunehmend haben sich hier Alternativen aus Open Source und Community Source im industriellen Einsatz etabliert.

Im Gegensatz zu kommerziellen Werkzeugen, die oft mit hohen Lizenz- und Einstiegskosten einhergehen, ermöglichen es die Open-Source- und Community-Source-Tools, sich mit den neuen Entwicklungen auseinanderzusetzen, ohne sich in Kosten stürzen zu müssen oder durch eine Demo-Lizenz zeitlich beschränkt zu sein. Dabei decken die verfügbaren Werkzeuge inzwischen den gesamten Entwicklungsprozess ab.

Gerade der erste Prozessschritt, das Erfassen und Verwalten der Anforderungen, ist für technik- und funktionsorientierte Entwickler oft ein ungeliebtes Kind. Dementsprechend gab es hierfür lange Zeit kaum ernstzunehmende Open-Source-Entwicklungen. Mit der zunehmenden Bedeutung von Entwicklungsstandards wie ISO 26262 (sämtliche Links zu Produkten und Stadards unter „Alle Links" – s. Ende des Artikels) rückt das Thema mehr in den Fokus, da ein durchgängiges Anforderungsmanagement nun rechtlich als Stand der Technik anzusehen ist. Zudem hat die Automobilindustrie den neuen OMG-Standard ReqIF (s. Artikel auf S. 66) zum Austausch von Anforderungen initiiert und vorangetrieben. ReqIF soll die Grenzen der Anforderungswerkzeuge aufbrechen und bereitet damit den Boden für eine neue Landschaft von Tools.

Eine der ersten Implementierungen des Standards, das Requirements Management Framework (RMF), ist als Open-Source-Software unter dem Dach der Eclipse Foundation veröffentlicht worden. Grundlage ist ein Framework zum Lesen, Schreiben und Modifizieren von ReqIF-Modellen, das auch als Integrationstechnik fungieren kann. Darauf aufbauend stellt es den Anwendern eine grafische Oberfläche zum Erfassen von Anforderungen bereit.

Hohe Komplexität der Varianten meistern

Beim Erfassen der Anforderungen sieht sich der Entwickler mit der Variantenkomplexität konfrontiert: Systeme werden heute meist nicht mehr als Einzelsystem, sondern als Teil einer Produktlinie mit unterschiedlichen Ausprägungen entwickelt. Viele Branchen sehen das Management von Produktlinien als ein Kernthema, aber die Automobilbranche beschäftigt sich aufgrund der hohen Konfigurierbarkeit der Fahrzeuge besonders damit. Hierbei konnte sich weder in der Terminologie noch in der Werkzeuglandschaft eine Vereinheitlichung durchsetzen. Das Variantenmanagement gliedert sich in vier Hauptschritte: Zu Beginn erfassen die Projekte die relevanten Merkmale des Produkts, die sogenannten „Features". Beispielsweise ließe sich ein Fahrzeug entweder mit Benzin- oder Dieselmotor bauen. Die Architekten modellieren das über zwei sich ausschließende Features. Ebenso werden unterschiedliche Getriebevarianten, Infotainment-Systeme et cetera und deren komplexe Abhängigkeiten erfasst – so schließen manche Motortypen bestimmte Getriebe-

typen aus oder es existieren besondere Beschränkungen für einen bestimmten Zielmarkt.

Ziel der Erfassung ist es zu systematisieren, welche Entwicklungsartefakte für welche Konfigurationen relevant sind – also welche Soft- und Hardware in einem konkreten Fahrzeug tatsächlich verbaut wird. Die Prozessbeteiligten ordnen Entwicklungsartefakte wie Anforderungen, Modelle, Software, Hardware und Testfälle den Features zu. Im einfachsten Fall wird ein Feature direkt einem Artefakt zugeordnet. Häufig ist es aber erforderlich, komplexere Bedingungen zu formulieren. So kann eine Anforderung genau für den Fall relevant sein, dass ein Automatikgetriebe im US-Markt betrieben und somit eine logische UND-Verknüpfung zwischen Features gesetzt wird.

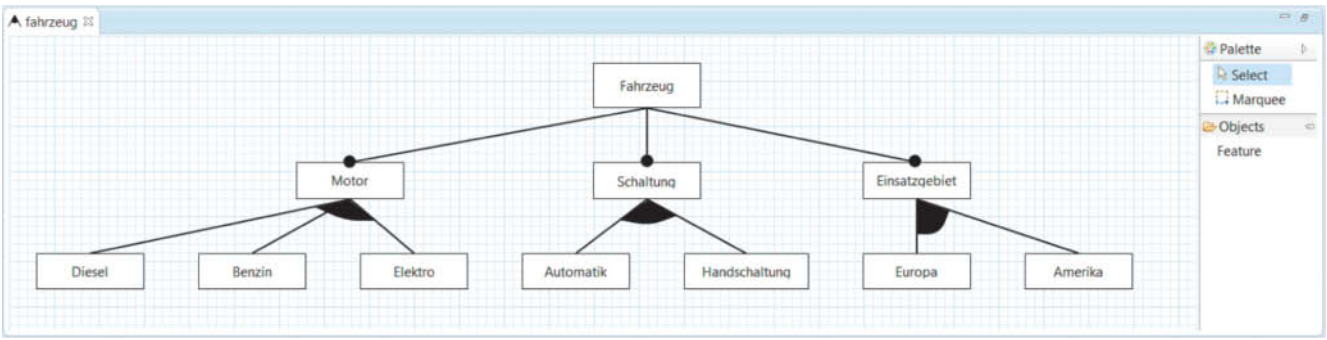
Die Gesamtheit der Artefakte, die auf die Produktlinie zutreffen, wird auch als 150-Prozent-Modell bezeichnet. Die Entwickler leiten nun die relevanten Informationen für eine Ausprägung – zum Beispiel ein Fahrzeugmodell – aus. Dazu beschreiben sie zuerst die konkrete Produktvariante über das Variantenmodell, in dem sie auswählen, welche Features zu dieser Variante gehören sollen. Die komplexen Abhängigkeiten sind dabei manuell kaum zu bewältigen. Die Konfigurationsverantwortlichen verlassen sich hier auf Werkzeuge, die die Korrektheit der Konfiguration überprüfen oder sogar Vorschläge machen. Im letzten Schritt, der konkreten Ausprägung, werden Ausleitungen der Artefakte auf Basis der zugeordneten Varianten erzeugt und somit alle Informationen entfernt, die für diese Variante nicht relevant sind.

Die Methoden- und Tool-Landschaft ist hierbei recht zersplittert. Die Standards EAST-ADL und AUTOSAR definieren Modelle und Austauschformate für Variantenmodellierung, und bei der OMG ist der Standard CVL (Common Variability Language) eingereicht, aber noch nicht veröffentlicht. Dazu kommen die spezifischen Ansätze der Werkzeuge. Aus dem Open-Source-Bereich seien hier die Feature IDE der Universität Magdeburg und das Eclipse-Feature-Model-Projekt erwähnt, die sich zum Modellieren der Features und Varianten eignen. Für eine Integration mit Anforderungen, Softwarearchitekturen et cetera ist in diesem Bereich allerdings noch spezifischer Entwicklungsaufwand erforderlich, um die Zuordnung zu Artefakten und die Ausprägung vornehmen zu können.

Modellierung der Funktionen

Die zunehmende Variantenvielfalt und Komplexität bringt viele Projekte dazu, eine abstraktere Systemsicht auf das Produkt einzunehmen, in der sie noch keine detaillierten Designentscheidungen wie Hardware- und Softwarearchitektur fällen. In dieser Phase modellieren sie die abstrakten Zusammenhänge zwischen den logischen Funktionen und deren Ein- und Ausgaben. Erst in späteren Phasen entscheiden sie dann über die konkrete Realisierung einer Funktion.

So benötigen in einem Fahrzeug einige Funktionen die Information, ob der Fahrer anwesend ist oder nicht. Die Funktion „Anwesenheit ermitteln“ wird in dieser Phase abstrakt beschrieben, zum Beispiel mit einem booleschen Ergebniswert als Ausgabe. Erst in späteren Designschritten entscheiden die Entwickler über die Umsetzung der Funktion in Hard- und Software. Sie



Feature-Modellierung mit dem Eclipse Feature Model (Abb. 1)

könnten einen Sensor im Sicherheitsgurt verbauen oder den Fahrer über eine Innenkamera finden.

Unterstützung bieten die methodischen Ansätze, die die Sicht auf das gesamte System behandeln. Einer der bekanntesten Vertreter ist die SysML, eine mit der UML verwandten Modellierungssprache für das Systems-Engineering (s. Artikel auf S. 86). Sie wird von den meisten UML-Werkzeugen unterstützt. Die SysML-Blöcke mit ihren Ein- und Ausgabe-Ports repräsentieren das Netz der logischen Funktionen. Die französische Luftfahrtindustrie treibt dabei das quelloffene Werkzeug Topcased voran. Allerdings überarbeiten die verantwortlichen Firmen derzeit sowohl das Werkzeug als auch die Organisation dahinter grundlegend – es rückt zur Eclipse Foundation und näher an das Eclipse-UML-Werkzeug Papyrus. Hier scheint es sinnvoll, auf das nächste Release im Rahmen von Eclipse 4.4 im Juni 2014 zu warten, um das Werkzeug auszuprobieren.

Einen standardisierteren Ansatz bietet EAST-ADL, der von der Automobilindustrie getrieben wird und ein definiertes Meta-Modell für funktionale Architekturen sowohl auf Analyse- als auch auf Designebene bietet. EAST-ADL eignet sich für den Einsatz in unterschiedlichsten Projekten, ist aber besonders dafür ausgerichtet, in Kombination mit AUTOSAR eingesetzt zu werden. Neben UML-Profilen liegen direkte Umsetzungen von EAST-ADL vor. Im Rahmen des Projekts EATOP entsteht zum Beispiel eine Eclipse-basierte Werkzeugunterstützung.

Softwarearchitekturen und Implementierung

Im Bereich der Softwarearchitekturen spielt die Unified Modeling Language nach wie vor eine wichtige Rolle. Da sie per se nicht auf einen spezifischen Einsatzzweck ausgerichtet ist, wird sie zunehmend mit sogenannten Profilen für den Einsatz in eingebetteten Systemen ergänzt. Verschiedene Werkzeuge unterstützen den hierzu von der OMG veröffentlichten Standard MARTE. Er ergänzt die Basis-UML um Modellierungselemente für Timing, Ressourcen-Verbrauch et cetera, damit sich das System frühzeitig analysieren lässt. Diese Analyse liegt allerdings außerhalb des Funktionsumfangs der meisten UML-Werkzeuge. Der Entwickler benötigt noch ein dediziertes Analysewerkzeug und einen Konverter von MARTE in das Eingabeformat des Werkzeugs. Hierfür hat der französische Konzern Thales einen Satz von Eclipse-Plug-ins zur Integration von UML/MARTE mit dem freien Analysewerkzeug Cheddar veröffentlicht. Im Automobilbereich bietet AUTOSAR teilweise ähnliche Modellierungsmöglichkeiten und sollte in diesem Fall definitiv verwendet werden, da AUTOSAR hier der Industriestandard ist.

Schon seit den frühen 90er-Jahren liegt mit ROOM (Real Time Object Oriented Modeling) eine Modellierungssprache für Echtzeitsysteme vor, deren Konzepte teilweise in die UML eingeflossen sind. Aufgrund des dedizierten Einsatzzwecks wird ROOM von seinen Befürwortern als schlanker und besser geeignet angesehen. Mit Eclipse eTrice liegt eine ROOM-Werkzeugunterstützung inklusive Code-Generatoren für C++ und C vor, die sich im industriellen Einsatz bewährt hat.

UML, AUTOSAR und ROOM zeigen, dass der Trend zu modellgetriebener beziehungsweise modellbasierter Software ungebrochen ist. Obwohl viele Funktionsentwickler diese Begriffe auf Werkzeuge wie Matlab/Simulink beziehen, ist der Einsatz von Modellen deutlich vielfältiger. Neben standardisierten Modellierungssprachen wie UML gewinnen weiterhin domänenspezifische Sprachen, die für einen bestimmten Zweck hin entwickelt werden, an Bedeutung. Eines der bekanntesten Werkzeuge ist hier Eclipse Xtext (s. Artikel auf S. 104), dessen Stärke neben der Open-Source-Lizenz die relativ geringe Einstiegshürde bei der Definition eigener Sprachen und Generatoren darstellt. Auf Xtext basiert auch die von der Infotainment-Initiative GENIVI definierte Interface-Beschreibungssprache Franca, die gegenwärtig außerdem als einer der Ansätze für eine Brücke zwischen Infotainment (Genivi) und klassischen Steuergeräten (AUTOSAR) diskutiert wird.

Tools, Tools, Tools

Mehr Unterstützung für die Kombination unterschiedlicher Modellierungsprachen bietet im Vergleich zum Parser-basierten Xtext der projizierende Ansatz von JetBrains' MPS (Meta Programming System). Im Embedded-Bereich macht sich ihn das Open-Source-Werkzeug mbeddr (s. Artikel auf S. 92) zunutze. Es kombiniert verschiedene Sprachen für Anforderungen, Implementierung, Testen und Verifikation eingebetteter Systeme in einer integrierten Werkzeugumgebung. Im Gegensatz zu anderen Modellierungsansätzen finden sich Entwickler mit C-Kenntnissen hier deutlich näher an ihrer gewohnten Programmiersprache, die um abstraktere Konzepte wie Zustandsautomaten ergänzt wurde und somit einige Nachteile von C ausgleicht. Der projizierende Ansatz ermöglicht es beispielsweise, einen Zustandsautomaten direkt als Code, Tabelle und Diagramm parallel nebeneinander anzuzeigen und synchron zu editieren. Nutzer und Dritte können jederzeit modulare Spracherweiterungen erstellen, die sich nahtlos in das C-Derivat integrieren.

Grafische domänenspezifische Sprachen mit Diagrammnotationen sind dagegen deutlich weniger verbreitet. Einer der Gründe ist sicherlich, dass im Gegensatz zu textuellen DSLs kaum nennenswerte kostenfreie oder Open-Source-Werkzeuge verfügbar sind. Grafische DSL-Werkzeuge waren dagegen lange mit hohen Lizenzkosten verbunden. Das ändert sich, da die französische Firma Obeo im Eclipse-Projekt Sirius Teile ihres Werk-