08 16. April 2019 8,00 € Organ der GMM VDE/VDI-GESELLSCHAFT MIKROELEKTRONIK, MIKROSYSTEM- UND FEINWERKTECHNIK

Elektronik

elektronik.de

Fachmedium für industrielle Anwender und Entwickler

Sonderdruck

Open Source Software für Safety-kritische Systeme:

So wird Linux normengerecht sicher

>> ab Seite 16

Übernahme von Mellanox soll neue Architekturen für die Rechenzentren der nächsten Generation erlauben.
>> Seite 6

Jensen Huang, Gründer und CEO von NVIDIA

Mit Simultanempfang gegen Mehrwegeschwund

>> Seite 26

Rauscharme Linearregler in der Stromversorgung

>> Seite 38

Open Source Software für Safety-kritische Systeme:

So wird Linux normengerecht sicher

(Bild: Emlix)

Sicherheitskritische Systeme müssen nach genau festgelegten Prozessen entwickelt und zertifiziert werden. Linux, entwickelt durch eine Community und ohne einen Hersteller, der ggf. in Haftung genommen werden kann, entspricht diesen Vorgaben nicht. Aber die Normen sehen Wege vor, wie Linux dennoch im Safety-kritischen Umfeld genutzt werden kann.

Von Heike Jordan und Alexander Much

Die Aufgaben von softwaregesteuerten Systemen sind in den vergangenen Jahren kontinuierlich gestiegen, auch in den Bereichen Medizintechnik und Automobilbau. Die Innovation in der Funktionsentwicklung von Systemen findet vermehrt in Software statt. In einem heutigen Oberklassefahrzeug werden mehr als 100 Steuergeräte verbaut, die mit mehr als 100 Millionen Zeilen an Programmcode programmiert werden [1]. Funktionen waren lange Zeit an die Steuergeräte selbst gekoppelt, sodass ein Mehr an Funktionen ein Mehr an Steuergeräten bedeutet hat, auch bedingt durch die Lieferketten in der Industrie. Die Zunahme von funktionalen Eigenschaften, zum Beispiel in den Bereichen Infotainment, Connectivity, Fahrerassistenz oder autonomes Fahren, führt zu einer wachsenden Anzahl an Steuergeräten und damit zu einer rasant steigenden Komplexität, da Funktionen gleichzeitig miteinander interagieren.

Entkoppelung von Hardware und Software

Eine Möglichkeit, diese Komplexität einzudämmen, liegt in der Entkopplung von Funktion und Steuergerät und damit einer Trennung von Hardware und Software. Das führt zu einer Zentralisierung von Rechenleistung in der E/E-Architektur im Fahrzeug, also dem

gesamten, durchgängigen Bordnetz, sodass Funktionen rein als Software implementiert werden und verschiedene Funktionen von mehreren Lieferanten auf einem Steuergerät koexistieren müssen (**Bild 1**). Ähnliche Entwicklungen in der Systemarchitektur finden sich beispielsweise in der Integrated Modular Avionics (IMA)-Architektur im Flugzeugbau [2].

Auch lassen sich Parallelen zum Rechenzentrumsbetrieb finden, in denen erst durch Virtualisierung und nun durch Containerisierung [7] verschiedene Funktionen auf der gleichen Hardware ausgeführt werden. Es ist davon auszugehen, dass die dadurch ermöglichten Geschäftsmodelle der „Servitization" von IaaS bis hin zu FaaS auch für den Automobilbereich übertragbar sind und sich damit neue Möglichkeiten der Zusammenarbeit ergeben. Technisch muss man bei der Konsolidierung die Rückwirkungsfreiheit zwischen Anwendungen sicherstellen, die in der Datensicherheit häufig unter den Schlagworten „Multitenancy" oder „Mandantensicherheit" diskutiert wird.

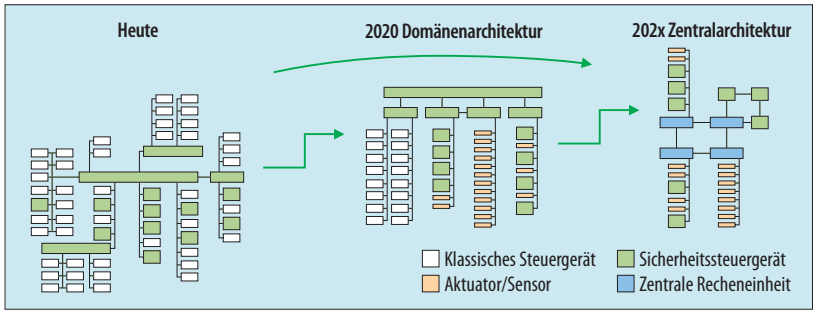

Bild 1. Evolution von Fahrzeugarchitekturen. (Quelle: Emlix)

Vor diesem Hintergrund stellt sich für Hersteller wie Zulieferer die Frage, auf welchen Ebenen eine Differenzierung möglich ist, also nur bei den Funktionen oder auch in der Middleware, dem Hypervisor oder dem Betriebssystem. Sie ist bis heute noch nicht beantwortet. Erste Tendenzen lassen sich jedoch erkennen: Mit der teilweisen Übernahme der Technik aus Rechenzentren wird man auch die Geschäftsmodelle übernehmen und sich in die dort bestehenden Strukturen einfügen. Sobald die Middleware standardisiert ist, wird die Differenzierung der Hersteller im Wesentlichen auf Funktionsebene stattfinden. Dies ist im Automobilbereich in den klassischen Steuergeräten durch AUTOSAR (AUTomotive Open System ARchitecture) erfolgt und wird aktuell mit Adaptive AUTOSAR auf die zentralen Recheneinheiten erweitert [5].

Zentralisierung von Funktionen

In der Medizintechnik hat die Konsolidierung von Steuergeräten nicht diesen herausragenden Stellenwert, allerdings nehmen auch dort die zu integrierenden Funktionen zu. Dabei beobachtet man häufig eine Modularisierung auf einer einheitlichen Software-/Hardware-Plattform. Gleichzeitig findet eine Zentralisierung bestimmter Funktionen auf einem Steuerrechner im Gerät statt: Das sind insbesondere Datenverarbeitung, -visualisierung, Daten-Logging, Bedienung sowie die Kommunikation zwischen Komponenten und mit einem lokalen Netzwerk oder dem Internet. Gerade die Prozessierung und Visualisierung sind in der Regel Safety-relevant. Ihre Integrität muss sichergestellt sein.

Nimmt ein medizintechnisches System beispielsweise Messdaten einer Sonde entgegen, ist der Pfad dieser Daten durch das Gesamtsystem sicherheitskritisch im Sinne der Unversehrtheit der Patienten. Je nach Anwendungsgebiet werden Mechanismen benötigt, um Verlust und Korruption der Messdaten über Kommunikationswege und während der Speicherung innerhalb des Systems zu erkennen oder zu verhindern. Neben der üblichen numerischen Fehleranalyse stellt sich bei der Aufbereitung der Daten die Frage nach der Verlässlichkeit der verwendeten mathematischen Bibliotheken und Algorithmen. Am Ende muss die Darstellung und fehlerfreie Übernahme der Ergebnisse garantiert sein. Dies muss auf den unterschiedlichen Ebenen – von der Architektur bis in die Verifikation – nachvollziehbar geplant und dokumentiert werden.

Eine Anbindung medizintechnischer Geräte über Weitverkehrsverbindungen scheitert hingegen häufig an rechtlichen Hürden sowie den Compliance-Vorgaben beispielsweise der Klinikbetreiber. Dies führt zu paradoxen Anforderungen im Hinblick auf die Wartung über den Lebenszyklus: Aktualität der Software versus Rezertifizierungsnotwendigkeit.

Open Source Software – mainline-compliant

Open Source Software hat in den vergangenen Jahren auch diejenigen industriellen Anwendungen erobert, aus denen sie mit Blick auf funktionale Sicherheit und Zertifizierbarkeit zunächst weitgehend ausgeschlossen war. Dies gilt bereits seit gut zehn Jahren für medizintechnische Anwendungsfelder. Im Automobilbereich erlebt Linux außerhalb des Bereiches Infotainment aktuell sein Debut auf Zentralrechnern im Fahrzeug.

In diesem Kontext stellen sich ganz zentrale Fragen, die den Entwicklungsprozess und die Pflege der Software über den gesamten Lebenszyklus betreffen. Die etablierten Standards, Zertifizierungen und Auditierungen – etwa die IEC 62304 für Medizintechnik oder Automotive SPICE und ISO 26262 für den Automobilbereich – liefern sinnvolle Prozessvorgaben in Bezug auf Planung, Entwicklung, Testen und Dokumentation von Software. Anforderungen an das Management des Lebenszyklus sind zumindest bei der IEC 62304 enthalten.

In beiden Normen nicht vollumfänglich berücksichtigt sind dabei zwei ganz wesentliche Aspekte:

- ➜ Die Besonderheiten von Open Source Software und insbesondere Embedded Linux im Entwicklungsprozess sowie beim späteren Lifecycle Management.
- ➜ Die Tatsache, dass mit dem Internet verbundene und damit global erreichbare Geräte im Hinblick auf die Datensicherheit einer aktiven Wartung und regelmäßiger Updates bedürfen.

Embedded Linux – obwohl schon deutlich reduziert gegenüber den Desktop-/Server-Distributionen – weist nicht nur eine beachtliche Komplexität auf. Es ist auch ein sehr umfangreiches Stück Software, das erst einmal „einfach da"

ist und nicht durch einen ggf. haftbaren Hersteller, sondern durch eine anonyme, auf den ersten Blick intransparente Community entwickelt und gewartet wird. Diese Software ist also außerhalb der Kontrolle des normierten Entwicklungsprozesses entstanden und muss nun normenkonform integriert und im weiteren Lebenszyklus gepflegt werden.

Dabei wird nicht mehr das gesamte V-Modell vollständig in Eigenregie durchlaufen: Entwicklungsziele, Qualitätsziele und daraus abgeleitet Anforderungen im engeren Sinne sind für den Linux-Kernel sowie die weiteren Open-Source-Komponenten zwar erarbeitet und dokumentiert, sie entsprechen aber oft nicht den Industrienormen. Ein normenkonformes und geplantes Vorgehen beginnt erst mit der Auswahl der Komponenten bzw. einer Software-Version sowie bei ihrer Komposition und Optimierung.

Auch ein Großteil der Wartung über den Lebenszyklus hinsichtlich Informationssicherheit und Qualität erfolgt durch die Community [3]. Durch sie werden Sicherheitslücken oder Fehlfunktionen aufgedeckt und neue Versionen zur Verfügung gestellt (**Bild 2**). Dies ist im Übrigen neben dem offenen Quellcode der entscheidende Vorteil von Open Source Software. Eine weltweite Community, im Wesentlichen bestehend aus den Entwicklern der sie einsetzenden Unternehmen, übernimmt kollektiv die kontinuierliche Verbesserung der Codebasis.

Das alles basiert auf einem der größten internationalen Entwicklungsprojekte [3], vertraglich abgesichert durch das Copyleft-Verfahren, das – vereinfacht formuliert – jedem alle Nutzungsrechte einräumt gegen die Auflage, Verbesserungen wiederum der Community zur Verfügung zu stellen. So findet sich Linux heute bereits in vielen kritischen Bereichen, von der Flugsicherung [8] über die Börse [9] bis hin zur Medizintechnik [6].

Normgerechte „Unbedenklichkeitsbescheinigung“

Formal erlauben es einige Normen, Linux als „gegeben“ hinzunehmen und erst von der Übernahme in das eigene Softwaresystem an aktiv normengerecht weiter zu entwickeln und zu konfigurieren. In der IEC 62304 geschieht dies über das Konstrukt der SOUPs, der „Software Of Unknown Provenance“. Gemäß IEC 62304 ist dies „eine Software-Komponente, die bereits entwickelt und allgemein verfügbar ist und die nicht entwickelt wurde, um in das Medizinprodukt integriert zu werden, oder bereits fertig entwickelte Software, für die angemessene Aufzeichnungen zum Entwicklungs-Prozess nicht verfügbar sind.“

Im Automobilbereich erlaubt die ISO 26262 ein ähnliches Vorgehen mit dem erklärten Ziel, dass „die Wiederverwendung von qualifizierten Software-Komponenten eine Neuentwicklung existierender Software-Komponenten vermeidet […] oder den Einsatz kommerziell verfügbarer Software (Commercial Of The Shelf, COTS) ermöglicht“.

Dieses Vorgehen ist besonders bei einem Betriebssystem interessant, da hier wesentliche Mechanismen für die Rückwirkungsfreiheit zwischen sicherheitskritischen und nicht sicherheitskritischen Softwarekomponenten umgesetzt werden. Üblicherweise erbt damit nur dieser Teil des Betriebssystems das Integritätslevel der höchsten Einstufung aller Funktionen. Das bedeutet nicht, dass das gesamte Betriebssystem als sicherheitskritisch eingestuft wird.

Sicherheitsanforderungen in der funktionalen Sicherheit beziehen sich immer auf das Gesamtsystem und mögliche Auswirkungen. Ein leider häufig anzutreffender Denkfehler ist, die Funktion mit den Komponenten zu verwechseln: Sicherheitsanforderungen sind immer Anforderungen zugewiesen und nicht Komponenten. Solange beispielsweise ein Ausfall einer Komponente zuverlässig erkannt wird und nicht zu einer Verletzung einer Sicherheitsanforderung auf Systemebene führen kann, braucht die Komponente nicht als sicherheitskritisch eingestuft zu werden.

Mainline-Kernel: Intensive Tests stellen Qualität sicher

Ein in der Medizintechnik etabliertes Vorgehen beim Einsatz von Linux ist, direkt von der Originalquelle zu starten. Solche Versionen werden „Mainline“ oder auch „Vanilla“ genannt. Dort erfolgte Änderungen unterliegen einem etablierten und strikten Review-Prozess durch die Community und durch so genannte Maintainer. Erst nach deren Freigabe wird ein Stück Code in den Kernel integriert.

In den vergangenen Jahren ist außerdem die Qualität beim Testen solcher Vanilla-Kernels deutlich besser geworden und wird von verschiedenen Mitgliedern in der Community übernommen. Erwähnenswert sind in diesem Zusammenhang beispielsweise das Linux Test Project (LTP) [10], die kontinuierliche Integrationsumgebung des 0-Day Testing Service [11] oder syzkaller [12].

Nach der Integration eines Vanilla-Kernels ins Projekt verändert man selbst direkt im Kernel nichts mehr. Dem Projekt obliegt „nur“ die Aufgabe der Konfiguration und der Absicherung im Projektkontext. Sind für das Projekt Erweiterungen oder direkte Änderungen notwendig – das können beispielsweise Security-Patches sein –, so dürfen diese im ersten Schritt nicht die bereits erreichte Absicherungsqualität senken. Auch ohne normative Anforderungen ist dringend zu empfehlen, die Komponenten, aus denen sich das Gesamtsystem zusammensetzt, auf die essenziell benötigten Teile zu begrenzen und weitere Abhängigkeiten bewusst auszuschließen. Für eine spätere Härtung [13] des Systems ist dies die Voraussetzung.

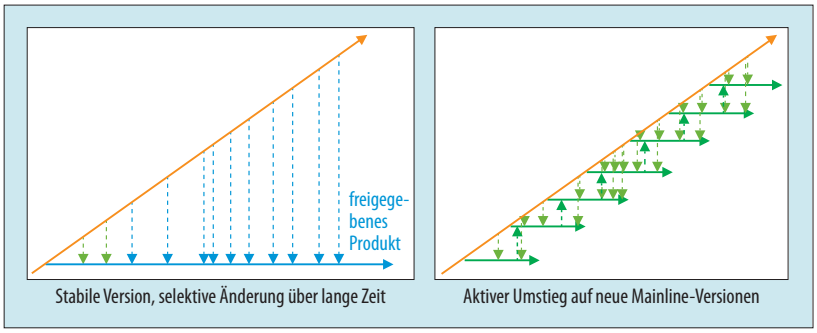

Bild 2. Wartungsstrategien bei Verwendung eines Linuxkernels. (Quelle: Emlix)

zung. Ebenso reduziert es Wartungs- und ggf. Rezertifizierungsaufwände signifikant, denn dafür müssen sämtliche Änderungen am Mainline-Code gemäß den Vorgaben der einschlägigen Normen geplant, dokumentiert und getestet werden, um eine vollständige Rückverfolgbarkeit der Änderungen und deren Verifikation zu gewährleisten.

Testwerkzeuge der Mainline-Community verwenden

In der Qualitätssicherung kann durchaus auf die durch die Community entwickelten und gepflegten Testsuiten zurückgegriffen werden. Obschon sie sich auf eingebetteten Systemen – also etwa Steuerrechnern – in der Regel nicht ohne Weiteres ausführen lassen, lohnt sich ihr Einsatz. Sind sie korrekt konfiguriert, lässt sich mit ihnen zumindest verifizieren, dass sich die Komponenten eines spezifischen Systems in den gewählten Konfigurationen erwartungsgemäß verhalten.

Darüber hinaus ist jedoch die Ausfallsicherheit des Gesamtsystems zentrale Anforderung. Vor diesem Hintergrund kommen funktionalen Tests der eingesetzten Komponenten beziehungsweise ihres Zusammenspiels in einer Funktionalität eine hohe Bedeutung zu. Sämtliche Tests sollten reproduzierbar zur Verfügung stehen.

Dieses wiederum sollte Bestandteil einer Wartungsstrategie sein. Systeme in der Medizintechnik und im Automobilbau haben oft eine mehrjährige Einsatzdauer. Eine Wartungs- und Update-Strategie ist entscheidend für den Einsatz von Software im sicherheitskritischen Bereich, besonders für Systeme mit Anforderungen im Bereich funktionaler oder Informationssicherheit. Das betrifft kommerzielle im gleichen Maß wie Open Source Software.

Vor dem Hintergrund der IEC 62304 muss es auch für das Linux-System einen solchen definierten Wartungsprozess geben. Der Linux-Kernel wird durch die Community konstant weiterentwickelt. Für frühere Versionen gibt es die sogenannte „longterm maintenance“. So ist die Version 4.9 am 11. Dezember 2016 erschienen und wird voraussichtlich bis Januar 2023 gepflegt, also ungefähr sechs Jahre [14]. Für die Verwendung von Linux in einem Projekt bedeutet dies, die Veröffentlichung von neuen Ständen zu beobachten und deren Kritikalität und Auswirkung auf das Produkt zu beurteilen.

Die Komponenten des Linux-Systems sind Bestandteil einer Liste aller SOUP-Komponenten, in der nicht nur die Herkunft der Sourcen, sondern auch der Zweck der Software im Gesamtsystem beschrieben sein muss. Für alle SOUP-Komponenten müssen Verbesserungen, Bugfixes und neue Versionen einem Monitoring sowie einer vorab definierten Relevanzprüfung unterzogen werden. Systematische Fehler sind in komplexen, softwarebasierten System trotz Anwendung gängiger Industrienormen nicht vermeidbar. Erweist sich ein Bugfix gemäß der festgelegten Kriterien als relevant für Funktion oder Datensicherheit des Systems, muss er übernommen werden.

Diese formalen Festlegungen entbinden jedoch wiederum nicht davon, die eingesetzte Software und das Gesamtsystem dem Stand der Technik entsprechend zu entwickeln und zu testen. Das System sollte über Überwachungsmechanismen verfügen, um Fehler, die zu sicherheitsrelevanten Ausfällen führen können, zu detektieren und angemessen zu reagieren. Auch dies ist unabhängig davon, ob im System Open Source eingesetzt wurde oder nicht.

Das Erreichen des Notwendigen für die funktionale Sicherheit oder für die Informationssicherheit ist nicht in Stein gemeißelt, sondern unterliegt einem Konsens aus Herstellern, Lieferanten und Zertifizierungsstellen, der sich über die Zeit hin weiterentwickelt. Daher sind Kooperationen wie das Projekt Sil2LinuxMP bei OSADL [15] oder aktuell ELISA (Enabling Linux In Safety Applications) bei der Linux Foundation [16] hilfreich, um für diesen Konsens zu arbeiten. *jk*

Referenzen

[1] https://www.visualcapitalist.com/millions-lines-of-code/, Stand 2019-03-03.
[2] https://en.wikipedia.org/wiki/ARINC_653, Stand 2019-03-10.
[3] https://www.linuxfoundation.org/publications/2017/10/2017-state-of-linux-kernel-development/, Stand 2019-03-10.
[5] https://www.auosar.org/, Stand 2019-03-03.
[6] https://conf.qtcon.org/system/attachments/121/original/Gerloff.StrategicUseOfFreeSoftwareatSiemens.pdf%3F1473084229, Stand 2019-03-06.
[7] https://en.wikipedia.org/wki/Virtualization#Containerization, Stand 2019-03-03.
[8] https://archive.fosdem.org/2017/schedule/event/air_traffic_control/, Stand 2019-03-03.
[9] https://www.pressebox.de/inaktiv/suse-linux-ag/Deutsche-Boerse-Systems-AG-konsolidiert-mit-SUSE-LINUX-Enterprise-Server/boxid/11948, Stand 2019-03-03.
[10] https://linux-test-project.github.io/, Stand 2019-03-03.
[11] https://01.org/lkp/documentation/0-day-test-service, Stand 2019-03-03.
[12] https://github.com/google/syzkaller, Stand 2019-03-03.
[13] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Grundschutz/Leitfaden/GS-Leitfaden_pdf.pdf?__blob=publicationFile&v=3, Stand 2019-03-06.
[14] https://www.kernel.org/category/releases.html, Stand 2019-03-03.
[15] https://www.osadl.org/SIL2LinuxMP.sil2-linux-project.0.html, Stand 2019-03-05.
[16] https://elisa.tech/, Stand 2019-03-05.

Alexander Much
der in Erlangen und Cambridge Mathematik und Informatik studiert hat, arbeitete zunächst als Linux-Entwickler bei SuSE Linux in Nürnberg, bevor er 2003 zu Elektrobit Automotive wechselte. Dort ist er verantwortlich für das System Engineering und dabei insbesondere für die Themen funktionale Sicherheit, Informationssicherheit und Open Source.
Alexander.Much@elektrobit.com

Heike Jordan
Nach dem Studium der Molekulargenetik und selbstständiger Tätigkeit im Umfeld der Nuklearmedizin kam Heike Jordan 2006 zu Emlix. Dort ist sie seit 2012 als geschäftsführende Gesellschafterin verantwortlich für das Multiprojektmanagement und dabei insbesondere für zertifizierungspflichtige Entwicklungsprojekte sowie Security und Lifecycle Maintenance.
hjordan@emlix.com