Einsatz von Virtualisierung für sichere Softwarearchitekturen

Zentrale Steuergeräte, Mixed Criticality, Testen, Multicore

Thomas Bock und Dr. Henning Kleinwechter, Volkswagen
Dr. Ralph Sasse, OpenSynergy; Armin Stingl, iSystem
Dr. Ralf Münzenberger, Philip Rehkop und Olaf Schmidt, INCHRON

Die hohe Anzahl von Steuergeräten veranlasst die Automobilindustrie, mehrere Steuergeräte zu zentralen Einheiten zusammenzufassen. Dabei werden viele Funktionen, die zuvor auf einzelnen Steuergeräten realisiert wurden, auf einer CPU gemeinsam ausgeführt. Dieser Beitrag beschreibt Herausforderungen und Lösungen, die sich für die zeitliche Verteilung der CPU-Rechenleistung auf die einzelnen Funktionen ergeben. Der Fokus liegt auf Interferenz-Freiheit für Systeme, die ISO 26262 ASIL-relevante Funktionen mit ASIL QM-Funktionen integrieren. Der Artikel basiert auf den Ergebnissen eines gemeinsamen Vorentwicklungsprojektes von Volkswagen, OpenSynergy und INCHRON. Darin wird ein Hypervisor als Architekturansatz verwendet. Das Echtzeitverhalten und die Wirkketten werden modelliert, simuliert und anhand von Messungen verfeinert und überprüft. Die Ergebnisse ermöglichen es, Handlungsempfehlungen (best practises) für den Einsatz eines Hypervisors in Domainsteuergeräten zu formulieren.

Motivation

Zentrale Steuergeräte beinhalten immer mehr Software-Funktionen, die von unterschiedlichen Softwarelieferanten (Tier-2) unabhängig voneinander entwickelt werden. Das gilt besonders für Body-Control-Module (BCMs). Sie integrieren eine große Anzahl an Funktionen mit unterschiedlichen Eigenschaften bezüglich

- Safety,
- Security,
- Zulassungsrelevanz und
- Änderungsgrad.

Das heißt, es werden auf einer Hardware Software-Systeme mit unterschiedlichen Sicherheitsanforderungen (mixed criticality) ausgeführt. Sie müssen insgesamt die Anforderungen der ISO 26262 einhalten, so müssen sie z.B. Interferenz-Freiheit sicherstellen zwischen Systemen, die in unterschiedliche ASIL-Levels eingestuft sind. Weiterhin werden neue, innovative Funktionen mit hohem Änderungspotential zusammen mit relativ stabilen Funktionen integriert, die ggf. gesetzliche Anforderungen erfüllen müssen. Ohne Absicherungsmaßnahmen kann jede Modifikation einer Softwarefunktion das Verhalten des Gesamtsystems beeinflussen, so dass einzelne Änderungen die erneute Absicherung des gesamten Systems erfordern.
Volkswagen hat das Vorentwicklungsprojekt gestartet, um die geforderten Normen und gesetzlichen Anforderungen einhalten zu können und um den stark steigenden Test- und Rezertifizierungsaufwand zu reduzieren.

Ziele und Herausforderungen

Wenn es zukünftig in PKW nur wenige hochperformante Steuergeräte mit unterschiedlichsten Software-Funktionen geben wird, dann wird das folgende Herausforderungen mit sich bringen:

- Es muss sichergestellt werden, dass für die integrierten Software-Funktionen die zugesicherten Ressourcen (Speicher, Rechenzeit) zur Verfügung stehen und von anderen Softwarefunktionen nicht beeinflusst werden.
- Es muss sichergestellt werden, dass die im Rahmen einer Ressourcenbudgetierung festgelegten Daten den Prozessor nicht überlasten und alle Echtzeitanforderungen erfüllen.

Dabei sollen Rahmenbedingungen der Karosserieelektronik im Vorentwicklungsprojekt berücksichtig werden, wie z.B. relativ ressourcenbegrenzte Prozessoren, die i.A. lediglich über eine Memory Protection Unit (MPU) verfügen.

In der Entwicklung von Softwarearchitekturen hat sich AUTOSAR als Standard zunehmend durchgesetzt. Hinsichtlich der Interferenz-Freiheit definiert AUTOSAR ein Partitionierungskonzept (OS-Applications) zum Speicherschutz. Für eine Rechenzeitzusicherung für Applikationssoftware verzichtet AUTOSAR auf explizite Mechanismen. Hier besteht die Möglichkeit, die Laufzeit einzelner Runnables oder Tasks zu überwachen. Das ist jedoch ein aufwändiges und zum Teil ressourcenintensives Konzept der Ausnahmebehandlung von Laufzeitüberschreitungen.
Deshalb wurde im Vorentwicklungsprojekt auf eine Softwarearchitektur fokussiert, bei der die Software-Systeme in virtuellen Maschinen (VMs) eingekapselt werden. Ein Hypervisor übernimmt dabei die Steuerung der VMs. Diese Architektur wird in anderen Domänen bereits erfolgreich eingesetzt. Sie erlaubt es, die Rückwirkungsfreiheit sowohl hinsichtlich Speicherzugriffen als auch Rechenzeit sicherzustellen. In Abb. 1 wird diese Architektur zusammen mit Anforderungen für drei zu integrierende Softwaremodule exemplarisch dargestellt.

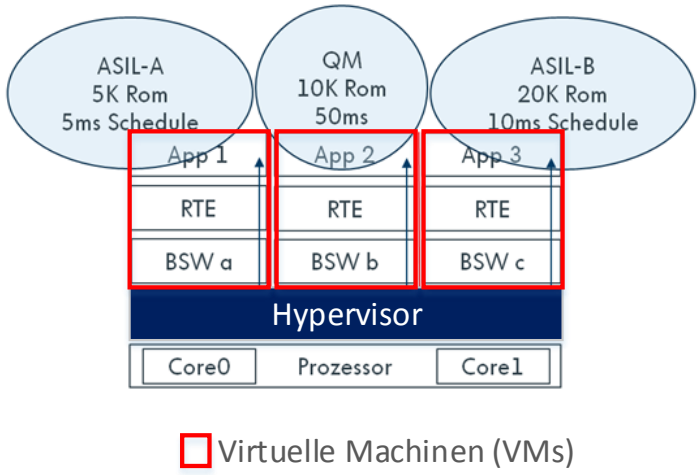

Abb. 1 Softwarearchitektur mit Virtualisierung

Ziel des hier beschriebenen Vorentwicklungsprojekts war zum einen eine Portierung einer Softwarearchitektur mit Virtualisierung auf einen aktuellen Prozessor für BodyControlModules. Zum anderen sollten die spezifischen Eigenschaften einer solchen Architektur mittels Simulation untersucht werden, insbesondere hinsichtlich des Timing- und Echtzeitverhaltens. Die hieraus gewonnenen Erkenntnisse ermöglichten die Definition von Best Practices für die Auslegung einer Hypervisor-basierten Softwarearchitektur.

Virtualisierung

Dank des Hypervisors können beispielsweise mehrere AUTOSAR-Softwarestacks nebeneinander auf einer CPU laufen. Der Hypervisor isoliert die einzelnen VMs. Jeder VM stehen die Ressourcen der CPU in dem Umfang zur Verfügung, als würde sie allein auf der CPU ausgeführt. Die Gastbetriebssysteme, RTE usw. können so unabhängig von anderen VMs programmiert und betrieben werden. Mehrere VMs teilen sich einen Core. In Multicore-Systemen können zusätzlich VMs über mehrere Cores verteilt werden. Die Aufgabe des Hypervisors ist dabei, die beeinflussungsfreie Unabhängigkeit der einzelnen VMs hinsichtlich Zeit (timing/scheduler) und Speicher (Flash/RAM) zu garantieren und gleichzeitig auch die gewollte Kooperation der VMs über Kommunikations- bzw. Sharingmechanismen zu ermöglichen bzw. zu überwachen.

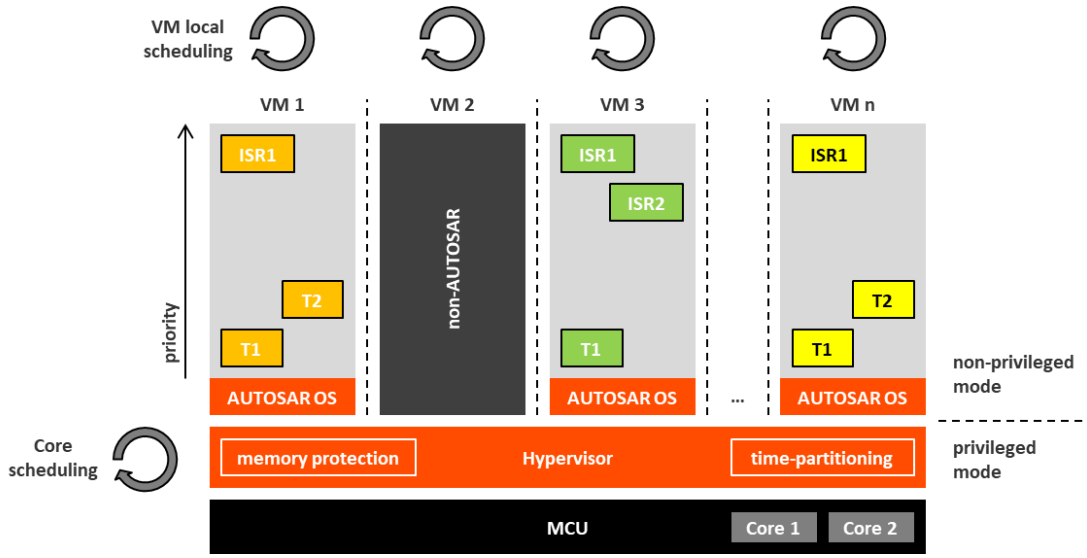

Abb. 2: Hierarchisches Scheduling mit Hypervisor

In Abb. 2 wird die erste (globale) Stufe des Schedulings gezeigt, die im Hypervisor implementiert ist. Dafür wurde ein rein statischer, TDMA-basierter tabellengesteuerter Ansatz gewählt. Die Ergebnisse des Projektes zeigen, dass die Zykluszeiten nicht zu kurz (steigender Overhead durch häufige VM Wechsel) und nicht zu lang (steigende Latency durch geplant spätere Bearbeitung) gewählt werden sollten. Die zweiten (lokalen) Stufen setzen die Gastbetriebssysteme in den jeweiligen VMs um.

Für den Schutz von Speicherbereichen stehen im BCM-Umfeld nur die eingeschränkten Möglichkeiten einer MPU zur Verfügung. Die beim VM-Wechsel notwendige Umprogrammierung der MPU kann aber durch die gezielte Wahl der Speicherzuordnung zu VMs so minimiert werden, dass nur die geringen Unterschiede zwischen den VMs geändert werden müssen.

Für Interrupts, die von aktuell nicht aktiven VMs behandelt werden, wurde ein pre-emptives, Budget-basiertes Konzept eingeführt, das die geplanten Ausführungszeiten der VMs weiterhin garantiert (fast Interrupts), allerdings auf Kosten der Auslastung der betroffenen CPU-Cores.

Timinganalysen

Ein wesentliches Ziel des Vorentwicklungsprojektes bestand darin, Timingeffekte durch den Einsatz von Virtualisierung systematisch zu untersuchen und anschließend Best-Practices für solche Systeme zu entwerfen. Hierfür wurde ein Timingmodell der dynamischen Architektur mit prioritätsbasiertem Scheduling (AUTOSAR OS) für ein Steuergerät mit Virtualisierung erstellt. Die Netto-Ausführungszeiten der Software, das Mapping der Softwarekomponenten auf Tasks sowie die Stimulation der ISRs stammen aus einem Serienentwicklungsprojekt. Gleichzeitig wurde ein Timingmodell des Serienentwicklungsprojekts erstellt, und zwar auf dem gleichen Abstraktionsniveau wie das System mit Virtualisierung. So ist das dynamische Verhalten mit und ohne Virtualisierung vergleichbar.

Für die Timinganalysen wurde der Echtzeitsimulator chronSIM verwendet. Eine wesentliche Erkenntnis, die hierbei gewonnen wurde, ist, dass durch den Einsatz eines Hypervisors mit Zeitscheiben für einzelne VMs Timingeffekte auftreten, die gegenüber einem reinen prioritätsbasierten Scheduling überraschend sind:

1. Eine geeignete Wahl der Periodendauer des Hypervisors und der Zeitscheibenlänge der einzelnen VMs hat eine erhebliche Auswirkung auf die Einhaltung der Zeitanforderungen. Selbst bei einer niedrigen Auslastung des Gesamtsystems kann es zu Echtzeitverletzungen innerhalb einer VM kommen.
2. Im Gegensatz zu einem System mit prioritätenbasiertem Scheduling müssen bei einem System mit Hypervisor zeitkritische Wirkketten sorgfältig im Entwurfsprozess geplant werden, wenn der Datenfluss über mehrere VMs geht. Dies betrifft die Ausführungsreihenfolge der VMs, wenn die Schritte einer Wirkkette möglichst aufeinander erfolgen sollen. Dies gilt sowohl auf Single- als auch auf Multi-Core-CPUs.
3. Die Abbildung von Softwarekomponenten auf Tasks und Tasks auf VMs kann sehr leicht ungünstig erfolgen, bezüglich der benötigten Last je VM (siehe Punkt 1) oder der Einhaltung der max. Latenzzeit von zeitkritischen Wirkketten (Punkt 2).
4. Da Interruptverarbeitung bei der Virtualisierung aufwendig ist, sollte möglichst Polling verwendet werden.

Insgesamt besteht eine wesentliche Erfahrung durch das Projekt darin, dass bereits in der Designphase das zeitliche Verhalten analysiert und die dynamische Architektur optimiert werden müssen. Wichtig ist es hierbei, die Zeitanforderungen frühzeitig eindeutig zu spezifizieren und die Zeitbudgets für die Software der einzelnen Zulieferer festzulegen. Im Folgenden werden zwei der oben erwähnten Punkte weiter erläutert.

Punkt 1: In Abb. 3 ist im unteren Teil ein zeitlicher Ausschnitt des OS-Status bei der Ausführung von Tasks in den entsprechenden VMs zu sehen, und im oberen Teil ist die Auslastung des Gesamtsystems zu erkennen. Die Abarbeitung des Tasks VM3_APP_20ms in der VM 3 wird in dem vorgesehenen VM-Zyklus nicht fertig, obwohl nur noch wenig Zeit für die vollständige Ausführung benötigt wird. Die vollständige Abarbeitung des Tasks VM3_APP_20ms muss bis zum übernächsten Aufruf von VM 3 warten, was in diesem Fall der zweifachen Zykluszeit des Hypervisors von 10 ms entspricht. Zu dieser Echtzeitverletzung kommt es aufgrund einer Hochlastsituation von VM 3, obwohl in anderen VMs noch Rechenkapazität zur Verfügung steht.

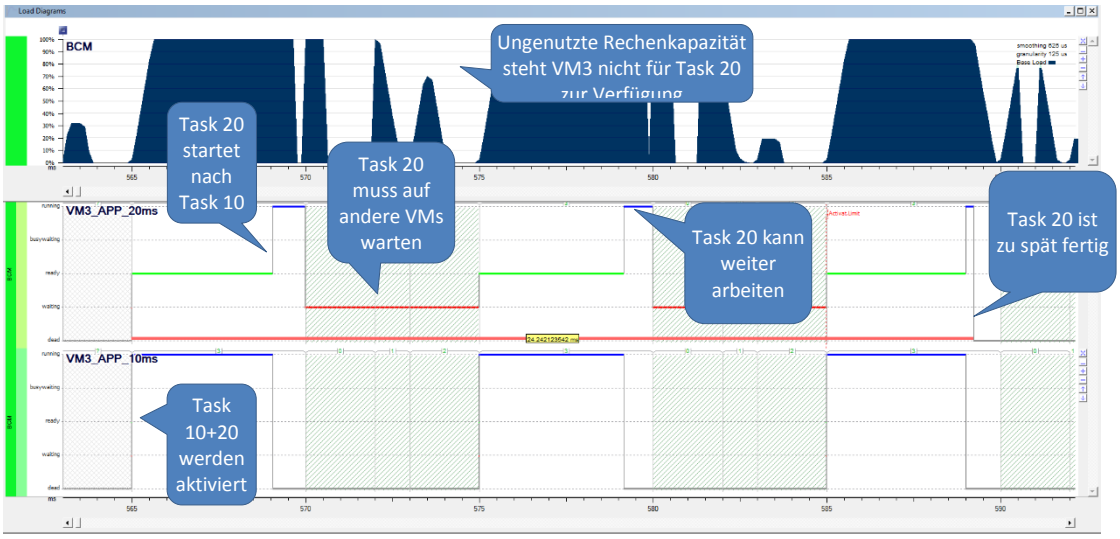

Abb. 3: Last- und Task-Zustandsdiagramm aus chronSIM mit Echtzeitverletzung, da VM3 in Hochlastsituation ist

Punkt 2: In Abb. 4 sind die Verläufe von zwei Wirkketten dargestellt. Bei der blauen Wirkkette ist Ausführungsreihenfolge der VMs und der Tasks für die Abarbeitung der Wirkkettenschritte optimiert, so dass es zu einer kleinen Wartezeit zwischen der Ausführung von aufeinanderfolgenden Wirkkettenschritten in VMs kommt. Anders ist dies bei der lila Wirkkette, die deshalb die maximale Latenzzeit deutlich übersteigert.

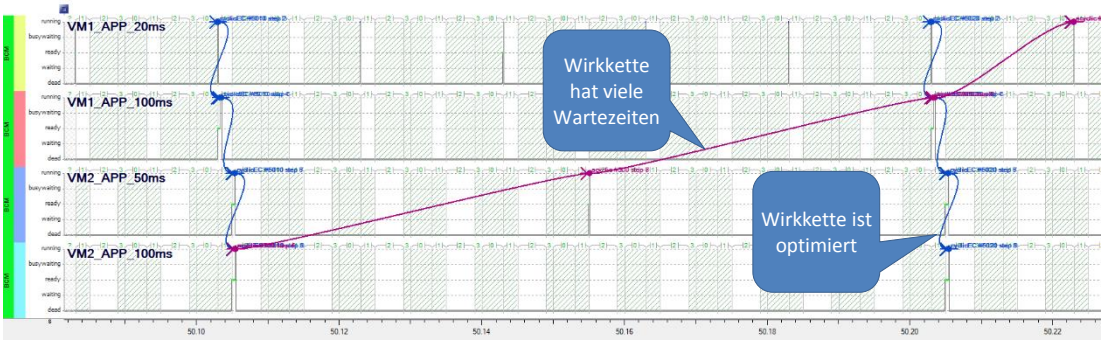

Abb. 4: Latenzzeitoptimierter (blau) und nicht optimierter (lila) Wirkkettenverlauf

Flankierend zur modellbasierenden Timing-Analyse wurden im Projekt Messungen mit winIDEA von iSystems durchgeführt. Gemessen wurden VMs, Tasks, ISRs und Runnables.

Best Practices beim Einsatz von Virtualisierung

Der Entwurf eines Systems mit Virtualisierung sollte mehrstufig erfolgen. In einem globalen Architekturentwurf werden die Eigenschaften des Hypervisors festgelegt. Hierfür werden von allen Projektpartnern (OEM, Tier-2 Lieferanten) wichtige Informationen über deren Teilsystem benötigt. Zudem werden erste Verhandlungen mit den Projektpartnern über deren Zeitbudgets für die Software geführt. Der lokale Architekturentwurf hat den Scope auf das Scheduling innerhalb des Steuergerätes und der VMs.

Der von Volkswagen erarbeitete Prozess sieht fünf Aufgaben vor. Eine Erfahrung des Projekts liegt darin, dass mehrere Iterationen der Aufgaben notwendig sind, um eine robuste und skalierbare Architektur zu entwerfen:

1. **Anforderungen erheben**
   Der Softwarearchitekt muss zu Beginn des Projekts eine Ressourcenplanung erstellen, aus der für jede Softwarekomponente das zur Verfügung stehende Laufzeitbudget, Speicher und andere Ressourcen hervorgehen.
2. **Clustering von Software-Applikationen**
   Die Anzahl der VMs kann anhand der Kriterien (a) ASIL Level, (b) Security Relevanz, (c) Software-Lieferant (d) Notwendigkeit von Interferenz-Freiheit und weitere Kriterien durchgeführt werden.
3. **Globaler Architekturentwurf**
   Aus den Daten zur Budgetierung und der Rückmeldungen der einzelnen Softwarekomponenten wird ein Timing-Modell erstellt. Die geplanten Budgets und Ressourcen werden mit Hilfe eines Modells in der vorgesehenen Architektur durch Simulation verifiziert. Aus den Ergebnissen können Optimierungen abgeleitet werden.

   Das Modell dient dazu, die Einhaltung der Ressourcengrenzen über die gesamte Entwicklungszeit sicherzustellen, daher ist eine zyklische Aktualisierung von Beginn an einzuplanen.
4. **Abstimmung und Finalisierung des Zeitbudgets je Projektpartner**
   Auf der Basis des globalen Architekturentwurfs werden die Zeitbudgets je Software-Komponente und mit den Projektpartnern verhandelt und festgelegt. Bei jeder Änderung ist eine erneute Timinganalyse durchzuführen.
5. **Lokaler Architekturentwurf**
   Für die einzelnen VMs (Scheduling, Mapping von Software-Komponenten auf Tasks etc.) empfiehlt sich eine ähnliche Vorgehensweise wie in Punkt 3, um auch hier die lokalen Ressourcengrenzen nicht zu überschreiten. Durch Timinganalysen werden die VM-lokalen Zeitanforderungen überprüft.
6. **Kontinuierliche Verifikation der Architektur**
   Durch die entwicklungsbegleitende Überwachung der Ressourcen und des dynamischen Verhaltens der Software wird die Funktion der einzelnen Komponenten und des Gesamtsystems sichergestellt.

Eine kontinuierliche Überwachung der Zeitbudgets und Einhaltung der Zeitanforderungen ist nach jeder Integration zwingend notwendig. Bei jedem relevanten Change Request sind die obigen Punkte durchzuführen.

Zusammenfassung und Ausblick

Die zunehmende Integrationsdichte in hochperformanten ECUs erhöht auch die Gefahr von Ressourcenkonflikten und gegenseitiger Beeinflussung einzelner Funktionsmodule. Durch den Einsatz von Virtualisierungskonzepten können diese Herausforderungen beherrschbar werden. Für den effektiven Einsatz eines Hypervisors in Domänensteuergeräten sind eine detaillierte Timing-/Speicher-/I/O-Ressourcenplanung während der Designphase und eine kontinuierliche Überwachung während der Entwicklung mittels Simulation sehr empfehlenswert.

Eine begrenzte Virtualisierungsunterstützung durch den Prozessor schränkt die Anwendbarkeit der Mechanismen im Hypervisor ein und erhöht den

Implementierungsaufwand. Sie stellt aber kein prinzipielles Ausschlusskriterium für dieses Feature dar. Da mit der stark steigenden Funktionsintegrationsdichte eine stetige Performanzerhöhung bei den Prozessoren einhergeht, werden sich Virtualisierungskonzepte schnell etablieren.

Autoren

Thomas Bock arbeitet seit mehr als 15 Jahren in der Entwicklung der Karosserielektronik bei Volkswagen. Er betreut die Entwicklung von Software Funktionsmodulen in diesem Umfeld als Projektleiter und legt dabei einen besonderen Schwerpunkt auf die dynamische Software Analyse.

Dr. Henning Kleinwechter betreut bei Volkswagen die Softwarearchitektur und Steuerung von 3rd Party Applikationssoftware für Body-Controller-Steuergeräte. Er hat mehr als 15 Jahre Erfahrung als Projektleiter und Software-Architekt für modellbasierte Softwareentwicklung bei Carmeq und Daimler.

Dr.-Ing. Ralph Sasse ist Lead Solution Engineer bei OpenSynergy. Er hat über 20 Jahre Erfahrung im Software Design und der Architektur sicherheitskritischer eingebetteter Systeme im Automobilbereich. Er war bereits für Daimler und Texas Instruments in Forschung und Entwicklung tätig.

Armin Stingl ist seit 2013 bei der iSYSTEM AG tätig und betreut hier die Definition, Validierung und Markteinführung von neuartigen Debug- und Trace-Tools. Mit über 20 Jahren Berufserfahrung war er in die Entwicklung von mehreren RISC Cores involviert. Als System Ingenieur war er bei namhaften Halbleiterherstellern vor allem für die Architekturdefinition zuständig. Im Rahmen dieser Tätigkeit war er bei der Entwicklung von on-chip Debug und Trace Lösungen für mehrere Prozessoren beteiligt

Dr.-Ing. Ralf Münzenberger ist als Mitgründer der INCHRON GmbH für den Bereich Professional Services als Geschäftsführer verantwortlich. In mehr als 160 Projekten hat er Kunden rund um das Thema Design von robusten dynamischen Architekturen, sowie bei der Sicherstellung der Echtzeitfähigkeit insgesamt, weitreichend unterstützt. Dies umfasst im Einzelnen Themen wie Architekturoptimierung, Migration von Single-Core nach Multi-Core, funktionale Sicherheit und Prozessberatung.