„Health Check“ für die Entwicklung von Echtzeitsystemen

Woran es im Entwicklungsprozess krankt[1]

Ralf Münzenberger, INCHRON
Friedhelm Stappert, Technische Hochschule Nürnberg

Viele SW-Projekte für eingebettete Systeme haben Probleme bzgl. Budget- oder Terminüberschreitungen. Gründe sind z.B., dass wichtige Anforderungen nicht früh genug berücksichtigt werden und dadurch Fehler zu spät erkannt werden, insbesondere beim dynamischen Verhalten eines Systems. Basierend auf vielen Entwicklungsprojekten wurde ein kompakter Fragenkatalog (*Real-Time Health Check*) entwickelt, der Schwachpunkte im Entwurfsprozess aufdeckt. Auf dieser Basis können Empfehlungen abgeleitet werden, wie das dynamische Verhalten von Echtzeitsystemen besser beherrscht werden kann. Typische Probleme sind: unvollständige Spezifikation von Echtzeitanforderungen, wenig robuste Architekturen, unvollständige Betrachtung des Gesamtsystems in frühen Designphasen. Vielen erfahrenen Architekten ist die Wichtigkeit dieser Aspekte durchaus bewusst. Trotzdem hat sich gezeigt, dass Timing-Fehler oft zu spät in der Testphase aufgedeckt werden.

Motivation

Im Rahmen von Entwicklungsprojekten im Bereich der Echtzeitsysteme, bei denen die INCHRON GmbH als Berater fungierte, entstand die Idee einer Analyse des Entwurfsprozesses. Das Ziel hierbei ist, die Schwachpunkte aufzudecken und herauszufinden, warum Aspekte des dynamischen Verhaltens eines Systems immer wieder so große Probleme bereiten, obwohl eine geeignete Methodik und entsprechende Tools vorhanden sind. Anhand eines kompakten Fragebogens, dem „Real-Time Health Check“, wird analysiert, wie weit diese Aspekte während des Entwurfs berücksichtigt werden.

Der Fragebogen richtet sich an System- und SW-Architekten, aber auch an Team- und Projektleiter, also allgemein an die Personen, die einen Einblick in den gesamten Entwurfsprozess haben. Idealerweise sollte der Test zu Beginn eines Projekts durchgeführt werden, wenn sich in der Vergangenheit gezeigt hat, dass immer wieder Probleme mit dem dynamischen Verhalten aufgetreten sind. Zudem ist der Test dann angeraten, wenn ein Projekt eine hohe technische oder organisatorische Komplexität aufweist. Es liegt dann ein hohes Projektrisiko vor, dass Probleme oder Fehler erst spät im Entwicklungsprozess gefunden werden, wie in [1] und [2] erläutert.

Die Auswertung des Fragebogens liefert ein detailliertes Selbstbild des Entwicklungsprozesses, d.h. eine Antwort auf die Frage: „Wo stehen wir?“. Davon ausgehend können anschließend ein Soll-Zustand und entsprechende Maßnahmen definiert werden, d.h. die Frage: „Wo wollen wir hin?“. Ziel hierbei ist es, Echtzeitaspekte besser im Entwurfsprozess zu verankern und dadurch das Risiko für zukünftige Projekte zu minimieren.

---

[1] Tagungsband Embedded Software Engineering Kongress 2015

Der im Folgenden vorgestellte Test wurde bereits bei mehreren Firmen durchgeführt, hauptsächlich im Bereich Automotive.

Der *Real-Time Health Check*

Der *Real-Time Health Check* besteht aus einer kompakten Liste von Fragen zu der Behandlung von dynamischen Aspekten während der Systementwicklung. Der vollständige Fragebogen kann online unter www.real-time-doctors.com eingesehen und ausgefüllt werden.

Anhand verschiedener Fragen wird herausgefunden, wann bestimmte Echtzeitaspekte während des Entwicklungsprozesses berücksichtigt werden, bzw. wann bestimmte Fehler bzgl. des dynamischen Systemverhaltens aufgedeckt werden. Der Fragebogen orientiert sich dabei am V-Modell. Für den Entwurf werden die Phasen „Design“, „Implementierung“, und „Test“ angenommen. Die konkreten Fragen lauten:

1. In which phase of the development cycle do you recognize the dynamic behavior of your system sufficiently?
2. In which phase do you identify the root causes for dynamic behavior problems?
3. In which phase do you consider the whole system including the environment (Sensor, actuator, peripherals, busses, networks, etc.)?
4. In which phase do you find show stoppers that are very expensive to solve?
5. In which phase do you focus on real-time requirements?
6. In which phase do you detect the following real-time errors?
    a. Load
    b. Response time
    c. Data consistency
    d. Event chain issues
    e. Jitter
7. In which phase are change requests for you most problematic?
8. In which phase do you know that changes will work?

Typische Fehler beim Echtzeit-Verhalten sind Fehler bzgl. Prozessorlast, Antwortzeiten von Tasks und ISRs (Interrupt Service Routinen), Datenkonsistenz, Wirkketten (z.B. Datenflusslatenzen vom Sensor bis zum Aktuator) und Jitter (z.B. Start-to-Start Jitter von Tasks, ISRs, Funktionen), wie aus Frage 6 ersichtlich. Je später solche Fehler aufgedeckt werden, desto aufwändiger und damit teurer wird die Korrektur.

Für den Test suchten sich die Teilnehmer ein repräsentatives Entwicklungsprojekt aus. Auf der Basis dieses Entwicklungsprojekts beantworteten sie für jede Frage einzeln, in welcher Phase (Design, Implementierung, Test) das jeweilige Thema behandelt wurde.

Für die Auswertung werden die Antworten unterschiedlich gewichtet und in einem „Radar-Diagramm“ (je Frage eine Radiale) dargestellt. Je größer die umschlossene

Fläche im Diagramm ist, desto höher ist das Risiko, Probleme bei der Entwicklung von Echtzeitsystemen zu bekommen. Die Risikobewertung wird grob in „Grün“ („Excellent Real-Time Health“, innerer Kreis), „Gelb“ („Medium Real-Time Health Risk“, mittlerer Kreis) und „Rot“ („High Real-Time Health Risk“, äußerer Kreis) eingeteilt.

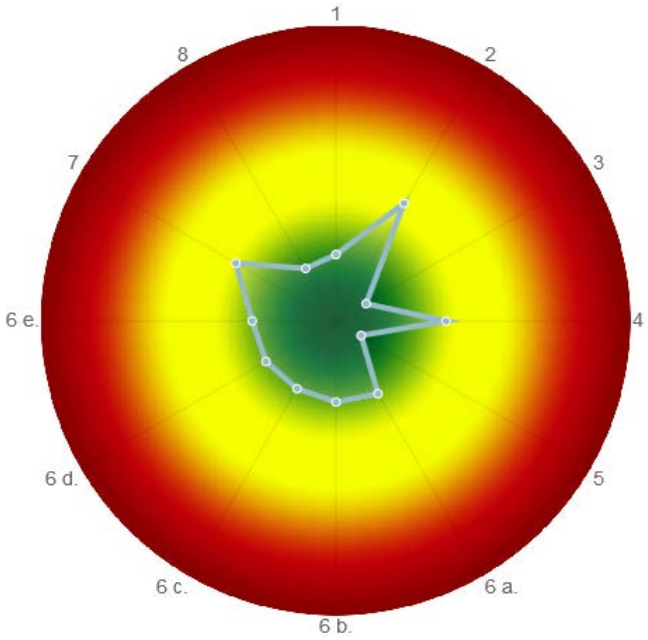

Abb. 1: Beispiel, geringes Echtzeit-Risiko im Entwicklungsprozess

Teilnehmer, die den Status „Grün“ erreichen, gehen bereits methodisch vor und setzen geeignete Tools während des Entwurfs konsequent ein. Die notwendigen Aufgaben, Artefakte und Rollen sind ein integraler Bestandteil des Entwicklungsprozesses. Diese Teilnehmer kennen i.d.R. ihren Entwurfsprozess sehr genau und wissen, wo sie stehen. In Abb. 1 ist eine Auswertung mit „Excellent Real-Time Health“ zu sehen.

Der Status „Gelb“ weist darauf hin, dass die Teilnehmer noch gewisse Schwachpunkte in ihren Prozessen haben, meist wissen sie aber auch bereits, dass es diese Schwachpunkte gibt.

Teilnehmer mit „rotem“ Status schätzen sich selbst häufig falsch ein. Sie sind sich der Probleme zwar bewusst, wissen aber nicht genau, wo die Ursachen sind. Dies zeigt sich in der detaillierten Auswertung im folgenden Abschnitt.

Auswertung

Der Fragebogen wurde bereits von 35 Teilnehmern beantwortet. Auch wenn die Ergebnisse nicht repräsentativ sind, zeigen sie doch einige interessante Trends.

Insgesamt zeigt sich (siehe Abb. 2), dass nur 11% der befragten Firmen den Status „Grün“ erreichen. Fast die Hälfte (46%) hat dagegen ein hohes Risiko.

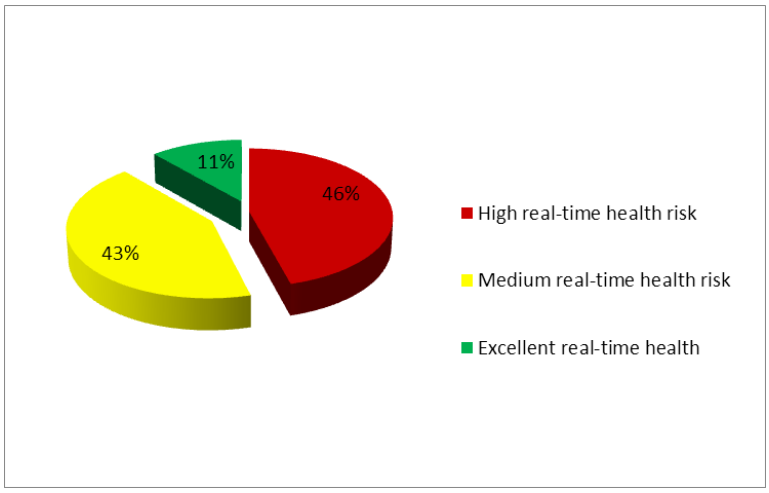

Abb. 2: Auswertung aller Fragebögen

Methodisches Vorgehen führt zum Ziel

Interessant ist hierbei, dass alle „grünen“ Teilnehmer bereits methodisch vorgehen, entsprechende Tools anwenden und im Entwicklungsprozess verankert haben (Abb. 3). Bei den Teilnehmern, die noch keine strukturierte Vorgehensweise etabliert haben, zeigen dagegen fast zwei Drittel (65%) den Status „Rot“, wie in Abb. 4 dargestellt.

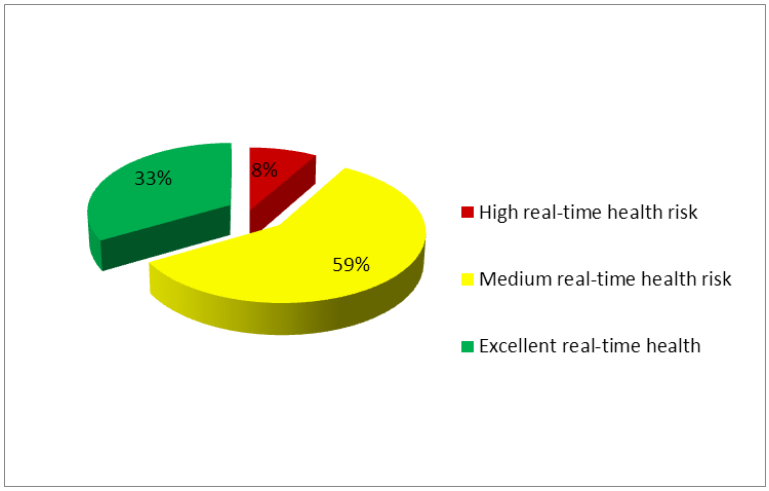

Abb. 3: Auswertung, Teilnehmer mit strukturierter Vorgehensweise

Ein typisches Beispiel für einen „grünen“ Entwicklungsprozess zeigt Abb. 1. Hier werden Echtzeitaspekte frühzeitig und ausreichend berücksichtigt und Fehler im dy-

namischen Verhalten rechtzeitig erkannt und behoben. Die Prozesse zeigen einen hohen Reifegrad.

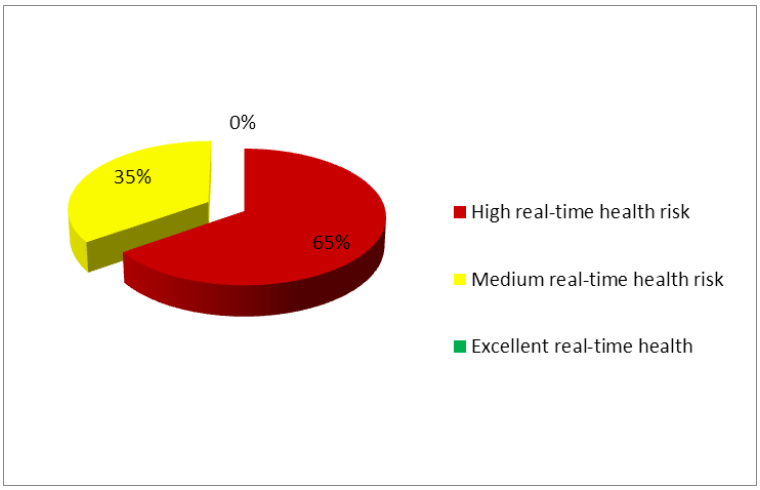

Abb. 4: Auswertung, andere (keine strukturierte Vorgehensweise)

Einen typischen Vertreter mit mittlerem Risiko zeigt Abb. 5. Hier kommt es immer noch zu Problemen, das Team ist sich der Problematik aber durchaus bewusst. Mit geeigneten Maßnahmen können die Prozesse optimiert werden, um den „grünen“ Status zu erreichen.

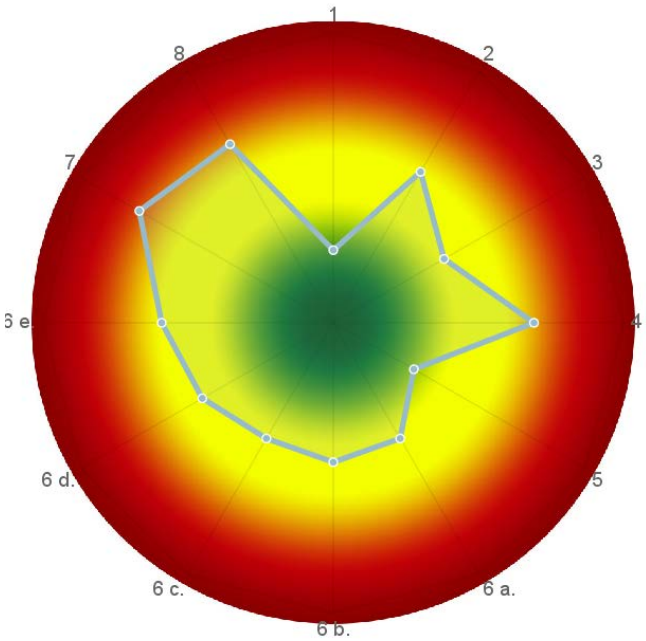

Abb. 5: Beispiel, mittleres Echtzeit-Risiko im Entwicklungsprozess

Hohes Echtzeitrisiko - es geht auch besser

Auch Vertreter mit hohem Risiko (Abb. 6) sind sich häufig bewusst, dass das Thema Timing zu spät bzw. nicht ausreichend berücksichtigt wird. Bei der Analyse der Gründe, warum der Entwicklungsprozess nicht geändert wird, sind folgende Punkte besonders häufig aufgetreten:

- Keine Zeit oder Budget für eine ausreichende Betrachtung von Timing während der Designphase.
- Es stehen die notwendigen Tools im Unternehmen nicht zur Verfügung.
- Das notwendige Know-how ist nicht vorhanden.
- Die Echtzeitanforderungen können erst während der Implementierungsphase genau spezifiziert werden.
- Die Netto-Ausführungszeiten der SW können erst in der Testphase ermittelt werden. Eine Budget-Abschätzung ist nicht möglich.

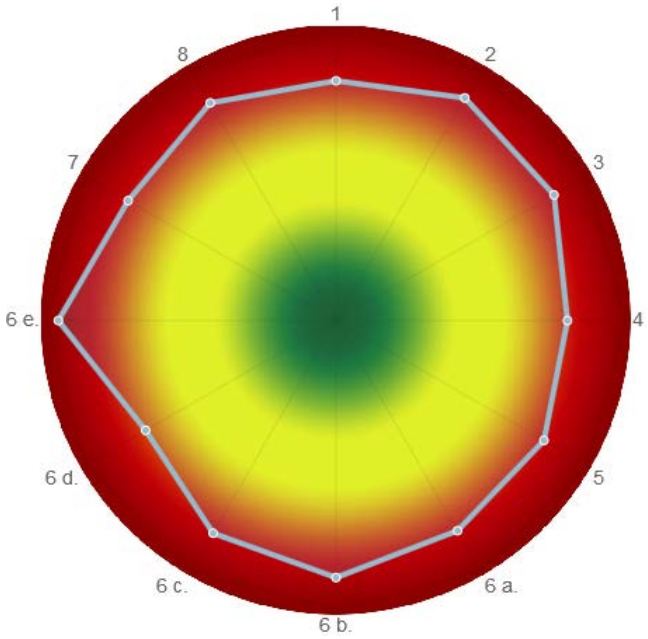

Abb. 6: Beispiel, hohes Echtzeit-Risiko im Entwicklungsprozess

Ein konkretes Eingehen auf die Argumente ist leider im Rahmen dieses Artikels nicht möglich. Die Testergebnisse der Teilnehmer mit einem geringen Echtzeit-Risiko zeigen allerdings eindeutig, eine frühzeitige Berücksichtigung des Themas Timing ist im Entwicklungsprozess möglich und vermeidet zeitaufwändige und teure Re-Designs in späten Entwicklungsphasen. In den Veröffentlichungen [6] bis [9] wird konkret aufgezeigt, wie dies in Entwicklungsprozessen umgesetzt werden kann und welche Vorteile sich daraus ergeben.

Selbstbild - Fremdbild

Bei einigen Teilnehmern stimmt die Selbsteinschätzung nicht mit dem tatsächlichen Bild überein, wie in Abb. 7 aufgezeigt. Der Teilnehmer geht davon aus, dass das dynamische Systemverhalten rechtzeitig und ausreichend berücksichtigt wird (Fragen 1, 3). Trotzdem werden die entsprechenden Fehler erst spät aufgedeckt (Frage 6). Hier wäre eine genauere Analyse zu empfehlen, um die Ursachen dieser Diskrepanz zu erklären und die entsprechenden Schwachpunkte aufzudecken.

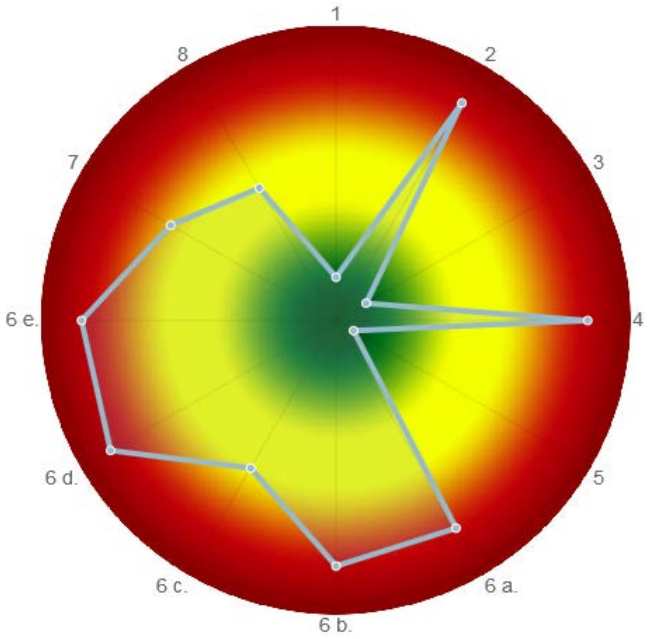

Abb. 7: Beispiel, inkonsistente Selbsteinschätzung

Etliche Teilnehmer mit einem mittleren bis hohen Echtzeit-Risiko füllten einen zweiten Fragebogen aus, um genauer zu spezifizieren: „wo wollen wir hin?“. Darauf aufbauend wurden entsprechende Prozessschritte definiert und im Entwicklungsprozess verankert.

Aktuelle Forschung

Die Grundlagen zur Behandlung von Timing-Aspekten beim Systementwurf sind hinreichend erforscht. Beispiele sind die Projekte TIMMO / TIMMO2USE [4] oder Amalthea [3]. In TIMMO und dem Nachfolgeprojekt TIMMO2USE wurde beispielsweise die Architekturbeschreibungssprache „EAST-ADL“ um eine Komponente zur Beschreibung des dynamischen Verhaltens eines Systems erweitert. Basis dieser Sprache sind Events, d.h. beobachtbare Ereignisse während der Laufzeit, und bestimmte Regeln bzgl. des zeitlichen Ablaufs dieser Events. Mit diesem Formalismus können sowohl Timing-Anforderungen als auch -Eigenschaften eines Systems erfasst werden [10].

Auch im AUTOSAR Standard der Automobilindustrie existiert eine Spezifikation für Timing Beschreibungen („AUTOSAR Timing Extensions“) [5].

Zusammenfassung

Der *Real-Time Health Check* ist eine effiziente Methode, Schwachpunkte im Entwurfsprozess zu finden, wenn es um Echtzeitaspekte geht. Architekten und Projektleiter können sich damit schnell einen Eindruck verschaffen, wo sie stehen und welche Maßnahmen ggf. notwendig sind, um das Risiko von zu spät gefundenen Problemen zu minimieren. Die Testergebnisse zeigen eindeutig, dass ein systematischer Ansatz, bestehend aus einer Methodik, den notwendigen Prozessschritten sowie den Einsatz von geeigneten Tools, zu einem geringen Echtzeit-Risiko führt. 65% der Teilnehmer, die dies nicht berücksichtigen, haben ein hohes Echtzeit-Risiko, während 33% der Teilnehmer, die dies berücksichtigen, ein geringes Echtzeit-Risiko haben.

Literatur

[1] U. Brodtmann: Return on Investment bei Tools – Automobil Elektronik 3/2013.

[2] R. Münzenberger: Dynamisches Verhalten. Steuergeräteentwicklung im Spannungsfeld von OEM und Zulieferer. Vortrag im Konferenzband 16. Internationaler Fachkongress in der Automobil-Elektronik, 19. und 20. Juni 2012, Ludwigsburg.

[3] AMALTHEA: An Open Platform Project for Embedded Multicore Systems, siehe http://amalthea-project.org/.

[4] TIMMO-2-USE (Timing Model - TOols, algorithms, languages, methodology, USE cases), siehe https://itea3.org/project/timmo-2-use.html.

[5] Specification of Timing Extensions: AUTOSAR_TPS_TimingExtensions.pdf, siehe http://www.autosar.org/documents/

[6] T. Jäger, I. Houben, R. Münzenberger: Modellbasierte Architekturentwicklung und Simulation. Tagungsband Embedded Software Engineering Kongress 2015.

[7] J. Meyer, I. Houben, R. Münzenberger: Kommunikationsoverhead bei Multi-Core-Systemen früh beherrschen. Tagungsband Embedded Software Engineering Kongress 2014.

[8] A. Wolfram, M. Makarov, T. Kramer, W. Ramisch, R. Münzenberger: Design of Robust System Architectures for Automotive ECUs; Conquest 2009, Nürnberg; www.isqi.org/conferences/conquest.

[9] R. Münzenberger, M. Dörfel, C. Dietrichs, U. Margull, G. Wirrer: Entwurf echtzeitfähiger Steuergerätesoftware in FlexRay-Netzwerken. KFZ Entwicklerforum 2007, Ludwigsburg.

[10] F. Stappert, J. Jonsson, J. Mottok, R. Johansson: A Design Framework for End-To-End Timing Constrained Automotive Applications. Embedded Real-Time Software and Systems (ERTS), Toulouse, France, 2010.

Autoren

Dr.-Ing. Ralf Münzenberger ist als Mitgründer der INCHRON GmbH für den Bereich Professional Services als Geschäftsführer verantwortlich. In mehr als 150 Projekten hat er Kunden rund um das Thema Design von robusten dynamischen Architekturen und bei der Sicherstellung der Echtzeitfähigkeit insgesamt weitreichend unterstützt. Dies umfasst im einzelnen Themen wie Architekturoptimierung, Migration von Single-Core nach Multi-Core, funktionale Sicherheit oder Prozessberatung.

Kontakt: ralf.muenzenberger@inchron.com
Internet: www.inchron.com
Email: info@inchron.com

Prof. Dr. Friedhelm Stappert ist seit 2011 an der Technischen Hochschule Nürnberg in der Fakultät Informatik tätig und vertritt dort das Lehrgebiet „Echtzeitsysteme und Embedded Systems“. Seine Forschungsschwerpunkte sind Methoden und Architekturen für die Entwicklung von eingebetteten Systemen, insbesondere im Automobilbereich.

Kontakt: friedhelm.stappert@th-nuernberg.de
Internet: www.th-nuernberg.de