8 Schlussbemerkungen

Ob Mobiltelefon, PDA, Waschmaschine oder Kraftfahrzeug – immer mehr entscheidet ein immer größer werdender Anteil an Software über den Erfolg eines Produkts. Jedes Gerät soll mit jedem kommunizieren können, möglichst klein und dennoch kostengünstig sein. Intelligente Kleidung, Fahrerassistenzprogramme im Auto und Drive-by-wire sind nur einige der inzwischen schon realisierten Beispiele. Sie alle teilen die Eigenschaft, von einem sogenannten eingebetteten System gesteuert zu werden.

Unter einem solchen System versteht man ein in ein umgebendes technisches System eingebettetes und mit diesem in Wechselwirkung stehendes Computersystem. Eingebettete Systeme finden sich in fast allen modernen technischen Systemen. Beispiele sind Bremsassistenten, Fahrdynamiksteuerungen (intelligente Tempomaten, engl. Adaptive Cruise Control) und Fahrzeugumfeldschutzsysteme in Kraftfahrzeugen ebenso wie Verkehrsleitsysteme, medizinische Geräte, Telekommunikationsgeräte oder Produktionsteuerungsanlagen. Auch in der Robotik, Automatisierungstechnik, Luft- und Raumfahrttechnik und in zahlreichen Konsumgütern leisten sie gute Dienste. Sie übernehmen dort komplexe Steuerungs-, Regelungs- und Datenverarbeitungsaufgaben und verleihen dem Gerät damit wettbewerbsentscheidende Zusatzeigenschaften.

Dies hat Embedded Systems zu einer der schnellstwachsensten Branchen auf dem Gebiet der Informatik gemacht. Gerade vor dem Hintergrund aktueller Trends der Informationstechnologie (IT) wie Outsourcing bzw. Offshoring kann die Softwareentwicklung eingebetteter System eine konstruktive Gegenmaßnahme sein, um interessante und gut bezahlte IT-Arbeitsplätze auch zukünftig im Lande halten zu können.

Hierfür ist allerdings bei vielen Verantwortlichen ein Umdenken erforderlich. Bei dem Entwurf eingebetteter Systeme ist neben der Hardwareentwurfstätigkeit mindestens gleichermaßen die Softwareentwicklung wichtig. Diese beinhaltet – teilweise spezifische – Analysemethoden, Beschreibungsverfahren, Programmiertechniken,

Vorgehensmodelle, Qualitätssicherungsmechanismen und Betriebssystemkonzepte. Hinzu kommt eine übergeordnete Sicht, die das gekoppelte Verhalten der Hardware- und Softwarekomponenten betrifft, kurz das Hardware/Software-Codesign.

In Abhängigkeit von der Anwendungsdomäne und ihrem spezifischen Kostendruck variiert der physikalische Aufbau eingebetteter Systeme stark. Er reicht von Ein-Chip-Computern mit Stückkosten von wenigen Eurocent für den Einsatz in preiswerten Konsumgütern über Steuergeräte in Flugzeugen und Automobilen bis zu modular aufgebauten, schrankgroßen Rechnern in der Automatisierungstechnik. Trotz dieser Unterschiede weisen alle eingebetteten Systeme einen gleichartigen logischen Aufbau auf, bei dem an die Stelle der Mensch-Maschine-Schnittstelle eine Schnittstelle (Sensorik, Aktuatorik) zu einem umgebenden technischen System tritt. Gerade in diesem Punkt unterscheiden sich eingebettete Systeme maßgeblich von anderen Informationssystemen. Bei letzteren werden Benutzereingaben in einem iterativen Prozess gelesen, bearbeitet und schließlich in Ausgaben transformiert.

Die Beschreibung der Systeminteraktion mit einer technischen Umgebung stellt die Entwickler eingebetter Systeme jedoch vor Herausforderungen, die sich von denen bei der Entwicklung „herkömmlicher“ Informationssysteme maßgeblich unterscheiden. Der Mensch verfügt über wesentlich flexiblere Reaktionsmöglichkeiten als eine technische Umgebung. Im Unterschied zum Menschen muss diese u. a. automatisch und sinnvoll auf Fehlermeldungen reagieren können und ggf. sogar einen Systemabsturz erkennen können. Systemantworten müssen unter Umständen in einem fest definiertem Zeitfenster erfolgen (vgl. Airbag- oder ABS-Steuerungen). Geschieht die Systemreaktion inkorrekt oder vielleicht auch nur geringfügig zu spät, kann dies fatale Folgen für System und Umgebung induzieren. Ein irrtümlicherweise ausgelöster Airbag ist ebenso nutzlos und gefährlich wie ein zu spät aufgeblasener. Letztendlich kann ein inkorrekt arbeitendes eingebettetes System Gesundheit und Leben seiner Benutzer schädigen.

In Kombination mit der Tatsache, dass einmal in den Massenmarkt ausgelieferte eingebettete Systeme nur schwer bzw. verbunden mit einem hohen Kostenaufwand korrigierbar sind, führt dies zu besonderen funktionalen und nicht-funktionalen Qualitätsanforderungen an die Hard- und besonders Softwareentwicklung eingebetteter Systeme. Allerdings haben, wie die alljährliche ADAC-Pannenstatistik zeigt, Rückrufaktionen hiesiger Premiummarken in den vergangenen Jahren deutlich zugenommen.

Die meisten eingebetteten Systeme sind reaktive Systeme, die beständig mit ihrer technischen Umgebung reagieren. Ihre gewissenhafte, formale und methodische Behandlung ist von größter praktischer Bedeutung und Kern regen wissenschaftlichen Interesses. Für einen systematischen Entwicklungsprozess werden formale und dennoch intuitive Beschreibungstechniken im Softwareentwurf (Statecharts, Esterel usw.) ebenso benötigt wie Standard-Prozessmodelle (z. B. das V-Modell XT) und Referenzarchitekturen (vgl. OSEK, AUTOSAR, HIS). Oftmals sind hier sogar Modelle nötig, die über die klassischen Modelle für Hardware oder Software Systeme hinausgehen (wie etwa Hardware/Software-Codesign oder hybride Systeme).

Ziel dieses Buches war es, einen kompakten und dennoch ausreichend tiefgehenden Überblick über ausgewählte aktuelle Spitzentechnolgien der Softwareentwicklung eingebetter Systeme zu geben, die zwar Stand der Wissenschaft sein mögen, aber – obwohl weitgehend fundiert und werkzeugunterstützt – noch nicht Stand der Technik sind. Damit soll es Studenten und Praktikern gleichermaßen als Leitfaden und Ideenmotor dienen.

Weiterer Forschungsbedarf besteht insbesondere bei der Entwicklung mathematischer Systemmodelle und formaler Semantiken für Beschreibungssprachen, der semantischen Integration unterschiedlicher Modellierungstechniken, methodischer Richtlinien, einer umfassenden Werkzeugunterstützung, insbesondere bei der Qualitätssicherung, sowie der möglichst automatischen Generierung von performanten, ablauffähigen Systemen.