

## **Beitrag zur Podiumsdiskussion**

# **“UML-RT – Die Lösung für eingebettete Software?”**

Bernhard Schätz

TU München, Fakultät für Informatik, Boltzmannstr.3, D-85748 Garching (Munich)  
schaetz@in.tum.de

Aufgrund ihrer wirtschaftlichen Bedeutung rückt die Erstellung von Software für eingebettete Systeme neben den bisherigen Anwendungsgebieten (z.B. betrieblichen Informationssysteme) zunehmend in das Zentrum der Aufmerksamkeit der Informatikforschung. Eingebettete Software unterscheiden sich von klassischen Softwaresystemen in einigen Punkten: oft liegt keine direkte Benutzungsschnittstelle vor, die Software übernimmt die Regelung oder Steuerung physikalischer Prozesse, die direkt mittels Sensoren und Aktoren mit dem System verbunden sind, und schließlich weist eingebettete Software zeitkritisches Verhalten bei der Verarbeitung periodischer und aperiodischer Ereignisse auf. Typische Anwendungsgebiete eingebetteter Software sind dabei die Produktions- und Automatisierungstechnik, Luft-, Raumfahrt-, und Automobiltechnik, oder Telekommunikation.

Ursprünglich kamen bei der Erstellung eingebetteter Software ausschließlich programmiersprachliche Ansätze zum Tragen, wobei sowohl anwendungsspezifische (z.B. IEC 1131, CHILL, Protel) als auch allgemeine Programmiersprachen (z.B. Assembler, C, Ada), zum Teil mit eingeschränkten Teilsprachen (z.B. SPARK Ada), eingesetzt wurden. Wie jedoch auch in anderen Anwendungsgebieten besteht bei der Entwicklung eingebetteter Software, z.B. in der Automobiltechnik, zunehmend die Notwendigkeit, abstraktere Modelle im Entwicklungsprozess einzusetzen. Dazu trägt vor allem die drastisch zunehmende Komplexität, bei verursacht durch den gestiegenen Funktionalitätsumfang eingebetteter Software sowie die zunehmende Vernetzung eingebetteter Komponenten (z.B. ca. 50 Mio. Zeilen Objektcode für ISDN Vermittlungssoftware (1996), ca. 10 Mio. Zeilen Quellcode Kfz-Steuerungssoftware Premiumfahrzeuge (2002)). Dies wird durch die erhöhten Sicherheitsanforderungen missionskritischer Software, den gesteigerten Druck zur Wiederverwendung und Variantenbildung durch verkürzte Produktzyklen, sowie ein verteilter Softwareentwicklungs- und Zulieferungsprozess verstärkt. Abhängig vom Anwendungsgebiet haben sich dabei unterschiedliche Modellierungsansätze wie synchrone Datenflusssprachen oder diskrete ereignisgesteuerte Automaten (vgl. z.B. [Lee00]) entwickelt.

In einigen Anwendungsgebieten (z.B. Automobiltechnik, Produktionstechnik) wurde Software ursprünglich als Ersatz für `klassische' regelungs- und steuerungstechnische Ansätze (z.B. Motorsteuerung) verwendet. Entsprechend kamen daher im Kfz-Bereich Werkzeuge wie Matlab/Simulink oder ASCET-SD, und entsprechende Bibliotheken wie die Automotive Block Library von MSR-MEGMA (Manufacturer Supplier Relationship Engineering Graphic Model Exchange) mit Funktionen wie Integratoren, Kennfeldern oder

Filtern zur Anwendung. Als Konsequenz stehen in diesen Ansätzen Modellierungselemente wie synchrone Funktionsblöcke, Signale, Perioden, und Tasks im Vordergrund.

In anderen Anwendungsgebieten (z.B. Telekommunikation) wurde Software vorherrschend zur Abwicklung diskreter, ereignisgesteuerter Aufgaben (z.B. Vermittlung) eingesetzt. Entsprechend kamen hier Werkzeuge wie Telelogic Tau oder ObjecTime zum Einsatz. Dabei standen in diesen Ansätzen Modellierungselemente wie kommunizierende Komponenten, Nachrichten, Zustände, sowie Sende- und Empfangsereignisse im Vordergrund. Aufgrund der Nähe zu verteilten Informatikanwendung (z.B. CORBA-Anwendungen) fanden diese Konzepte auch bei der Modellierung betrieblicher Anwendungen ihren Einsatz. Sie wurden – wegen der Möglichkeit, reaktive Systemen mit schwachen Echtzeitanforderungen zu beschreiben – unter dem Begriff UML-RT (UML for Real-Time) zusammengefasst und schließlich in UML (z.B. [OMG04]) aufgenommen.

Während bisher bei eingebetteter Software die jeweiligen Anwendungsgebiete diskreter Ereignissysteme (z.B., Komfortelektronik) und synchroner zeitgesteuerter Systeme (z.B. Antriebsstrang) nur wenig in Berührung kamen, wachsen diese Bereiche zunehmend zusammen (z.B. aktive Fahrerassistenzsysteme). Damit stellt sich die Aufgabe, einen einheitlichen Modellierungsansatz für beide Anwendungsgebiete eingebetteter Software zur Verfügung zu stellen, um beide Aspekte eines eingebetteten Systems in Kombination beschreiben, analysieren, und bis hin zu einer Implementierung zu entwickeln.

Da UML-RT sich stärker auf reaktive als auf eingebettete Systeme ausrichtet, wurde mit einem zusätzlichen UML-Profil [OMG03] ein Metamodell entwickelt, um die in der Echtzeitprogrammierung eingesetzten Konzepte und Begriffe (z.B. Ereignis, Aktion, Ressource, Schedule) aufzunehmen. Der Schwerpunkt wird dabei darauf gelegt, ein Ressourcenmodell in die UML zu integrieren, also Begriffe wie Ressource, Aktion, oder Dauer in Bezug zu setzen zum Begriffsmodell der UML mit Begriffen wie Verhalten, Zustand, Übergang, oder Nachricht. Ähnlich wie UML selbst konzentriert sich das Profil bewusst darauf, eine Sprache zur Beschreibung der Ressourcen eingebetteter Systeme – möglichst unabhängig von konkreten Notationen oder Werkzeugimplementierungen - zur Verfügung zu stellen, nicht aber den Einsatz der Sprache in der Entwicklung vorzugeben.

Die Bereitstellung eines umfassenden Metamodells für reaktive Softwaresysteme und die Integrationen von Sprachelementen zur Ressourcenbeschreibung stellen sicher eine wichtige Voraussetzung für die werkzeugtechnische Behandlung eingebetteter Systeme dar. Für die Bereitstellung eines umfassenden Modellierungs- und Entwicklungsansatzes für eingebettete Software sind aber darüber hinaus noch einige weitere Aufgabenstellungen zu betwähigen. Dazu gehört es insbesondere,

- die elementaren Modellierungskonzepte für reaktive Software (z.B. Komponenten, Kanäle, Ereignisse) gegenüber den Implementierungskonzepten (z.B. Objekt, Attribut, Methodenaufruf) wie in UML-RT in den Vordergrund zu rücken, um die Modellierung anstatt der Implementierung zu betonen
- *domänenspezifische Konzepte* (z.B. Funktion, Periode, Ausführungszeit; aber auch Prozessor, Kommunikationspfad, Task, Schedule) in das allgemeine Metamodell (z.B. in Form eines Profils für eingebettete Systeme) zu integrieren (nicht nur in Form der domain-specific concept models für Ressourcen, wie in UML-RT vorgesehen) um so die unterschiedlichen *anwendungsspezifischen Sichten* auf eingebettete Software zu unterstützen.

- diese neu eingeführten domänenspezifische Konzepte zu einzelnen Teilansätzen (z.B. synchrone Datenflussfunktionen, zeitdiskrete ereignisgesteuerte Systeme) zusammenzufassen und geeignete *einfache Ausführungsmodelle* für diese unterschiedlichen domänenspezifischen Modelle und Abstraktionsebene zu definieren (z.B. Abstraktion von Zeiteigenschaften für synchronen Datenfluss, zeitsynchrone signalbasierte Kommunikation), um so die Komplexität der einzelnen Sichten zu beschränken
- die eingeführten Teilansätze den *verschiedenen Abstraktionsebenen* bei der Modellierung verteilter eingebetteter Software einzuziehen (z.B. die funktionale Architekturebene mit Funktionen und Diensten, die logische Architekturebene mit Clustern kommunizierender Komponenten, die technische Architekturebene mit Steuergeräten und Kommunikationspfaden), um sie in einen domänenspezifischen Entwicklungsprozess (z.B. ITEA EAST/EEA, vgl. [FvdB+03]) zu integrieren
- die Modelle der verschiedenen Abstraktionsebene in einen *Entwicklungsprozess* zu integrieren und *Übergänge* zwischen diesen Modellen (z.B. von Funktionsblöcken mit verzögerungsfreier Berechnung und Kommunikation zu Komponenten mit Verzögerung) zu definieren. Die Bereitstellung solcher Übergänge - sowohl mittels analytischer (z.B. Konsistenz zwischen dem Datenfluss auf Funktionsblockebene und seiner Realisierung als Nachrichtenaustausch auf Komponentenebene) also auch generativer Verfahren (z.B. Berechnung von Kommunikationsschedules für den Übergang von der logischen zur technischen Architektur) – ist nötig, um einen werkzeuggestützten Entwicklungsprozess bis hin zur Implementierung in Hard- und Software zu unterstützen.

Naturgemäß spielt die Modellierung reaktiver Anteile und die Erfassung von Ressourcenanforderungen eine wichtige Rolle bei der Erstellung von eingebetteter Software. Für die Beherrschung der Entwicklungsproblematik ist dies aber nur ein erster Schritt. Hier ist letztendlich eine Untergliederung der verwendeten Modellierungsansätze und des Entwicklungsprozesses in domänenspezifische, handhabbare Teile die Voraussetzung, um die stetig steigende Komplexität eingebetteter Systeme in den Griff zu bekommen.

## Referenzen

- [FvdB+03] Freund, U. von der Beeck, M. Braun, P. Rappl, M. *Architecture Centric Modeling of Automotive Control Software*. SAE 2003-01-0856, Detroit 2003.
- [Lee00] Lee, E. *What's Ahead for Embedded Software*. IEEE Computer, 2000.
- [Lyo98] Andrew Lyons. *UML for Real-Time Overview*. Whitepaper, www.rational.com, 1998.
- [OMG03] Object Management Group. *UML<sup>TM</sup> Profile for Schedulability, Performance, and Time Specification*. Version 1.0 formal/03-09-01. www.uml.org, 2003.
- [OMG04] Object Management Group. *Unified Modeling Language: Superstructure*. Version 2.0. OMG Adopted Specification ptc/03-08-02. www.uml.org, 2004.