ANTSRT
-
Eine Software-Architektur für Fahrerassistenzsysteme

André Schmidt, Steffen Görzig, Paul Levi

DaimlerChrysler AG,
{andre.s.schmidt|steffen.goerzig}@daimlerchrysler.com,
Universität Stuttgart,
paul.levi@informatik.uni-stuttgart.de

Zusammenfassung Die Software-Architektur ANTS (Agent NeTwork System) wurde erfolgreich zur Erprobung zahlreicher Fahrerassistenzsysteme eingesetzt. Nach einem kompletten Redesign liegt nun eine weiterentwickelte Software-Architektur vor, welche in diesem Beitrag beschrieben wird - ANTSRT.
Sie wurde verstärkt unter dem Aspekt der Echtzeitfähigkeit entwickelt (Real Time - RT). Als signifikanteste Neuerung ergibt sich daraus die Integration von echtzeitfähigen Reglersystemen mit nicht echtzeitfähigen Softwarekomponenten zur Umgebungserfassung. Vorgestellt werden die erweiterten Bedürfnisse an die Software-Architektur, sowie die daraus resultierenden Konzepte für eine Umsetzung. Im Anschluß werden erste Erfahrungen anhand eines Prototyps vorgestellt.

1 Einleitung

Es wurden bereits verschiedene Software-Architekturen für Fahrerassistenzsysteme entwickelt. Der Versuchsträger VaMP der Bundeswehr-Universität München besaß beispielsweise eine Systemarchitektur bestehend aus vier Ebenen, genannt der 4D Ansatz [1][2]. Das von DaimlerChrysler (ehemals Daimler–Benz) entwickelte Versuchsfahrzeug VITA II (Vision Technology Application) besaß eine kontinuierliche Verhaltenssteuerung [3]. Am Lehrstuhl für Theoretische Biologie des Instituts für Neuroinformatik an der Universität Bochum wurde ebenfalls eine Architektur für Fahrerassistenzsysteme entwickelt [4][5]. In DaimlerChrysler Forschungsversuchsträgern wurden Fahrerassistenzsysteme - von der autonomen Stop&Go Fahrt bis hin zu Ampel- und Vorfahrtsassistenten - mit Hilfe des Multi-Agentensystems ANTS demonstriert [6].

Keine dieser Architekturen erfüllt jedoch alle Anforderungen, welche zur schnellen prototypischen Umsetzung zukünftiger Fahrerassistenzsysteme benötigt werden:

- Offene Plattform zur Integration verschiedenster Softwarekomponenten zur Fahrzeugsteuerung wie Sensorerfassung (Radar, Kamerasteuerung), Informationsverarbeitung (Bildverarbeiter, Sensorfusion) und Aktuatoren (Steuereingriffe/Regler).
- Zeitliche Koordination der Ausführung.
- Strikte Einhaltung der Echtzeitkriterien wenn notwendig (z. B. für Regler).
- Transparenz bezüglich Betriebssystem und verfügbarer Hardwareressourcen.
- Paralleler (deadlockfreier) Zugriff auf Daten.
- Effiziente Nutzung von Ressourcen in einer verteilten Rechnerumgebung.
- Dynamische Komponenten und gleichzeitiger Einsatz verschiedener Software-Versionen in Komponenten.

Die nächsten Abschnitte zeigen, wie diese Anforderungen konzeptionell umgesetzt wurden und welche Ergebnisse damit erzielt wurden.

2 Softwareagenten

Nach Wooldride und Jennings [7] können Agenten über ihre Eigenschaften definiert werden:

> „Ein Agent ist ein Hardwaresystem oder softwarebasiertes Computersystem, das bestimmte Kriterien erfüllt:
> **Autonomie** Agenten besitzen eine eigene Kontrolle über internen Zustand und Aktionen. Es existiert keine direkte Steuerung durch den Menschen.
> **Soziale Kompetenz** Agenten kommunizieren und interagieren miteinander. Die Art der Kommunikation wird durch eine 'Agent Communication Language' beschrieben.
> **Reaktivität** Agenten nehmen ihre Umgebung wahr und reagieren in angemessener Form und in vorgegebenem zeitlichen Rahmen auf Veränderungen.
> **Proaktivität** Agenten führen nicht nur einfache Reaktionen aus, sondern arbeiten zielgerichtet. "

Die Betrachtung der Agenten als reale oder abstrakte Objekte mit speziellen Zuständen ermöglicht, agentenbasierte Systeme auf objektorientierte Konzepte abzubilden. Statische Modelle beschreiben die strukturellen Relationen zwischen verschiedenen Agenten. In dynamischen Modellen können Interaktionen (Kommunikation) zwischen den Agenten spezifiziert werden. Funktionale Modelle können dazu verwendet werden, das Verhalten der einzelnen Agenten darzustellen. Für eine vollständige Beschreibung der Agenten sind einige Modifikationen und Erweiterungen objektorientierter Vorgehensweisen einzuführen. Neben den Zuständen und dem Verhalten der Agenten muß eine Darstellung ihrer Ziele möglich sein. Weiterhin müssen Interaktions- und Verhandlungsprotokolle der Agenten modelliert werden können. Die verschiedenen agentenorientierten Modelle können in Software-Architekturmodellen zusammengeführt werden.

3 Software-Architektur ANTSRT

Software-Architekturen beschreiben den grundlegenden Aufbau eines Software-Systems in Form von Komponenten und deren Beziehungen zueinander sowie zur Umwelt [8]. Die Komponenten müssen eine Reihe von Diensten bereitstellen, welche die Anforderungen an das System realisieren. Der erste Schritt in der Anforderungsanalyse bestand darin, diese Dienste zu identifizieren und nach Gemeinsamkeiten (ähnliches Aufgabenspektrum) in Gruppen zusammenzufassen. Gleichzeitig waren die Interaktionen zwischen den Diensten (Dienstgruppen) zu erfassen. Das Ergebnis spiegelt sich in einem dreistufigen Architekturmodell (Bild 1) wider. Auf der untersten Ebene findet eine Abstraktion von eingesetzten Hardwaretreibern und Betriebssystemdiensten statt. Darauf aufbauende Dienste können in einer heterogenen Umgebung mit verschiedenen echtzeit- und nichtechtzeitfähigen Betriebssystemen eingesetzt werden. Den Kern des Softwaresystems bildet die zweite Schicht (Framework-Ebene) mit Schnittstellen zu den Applikationskomponenten.

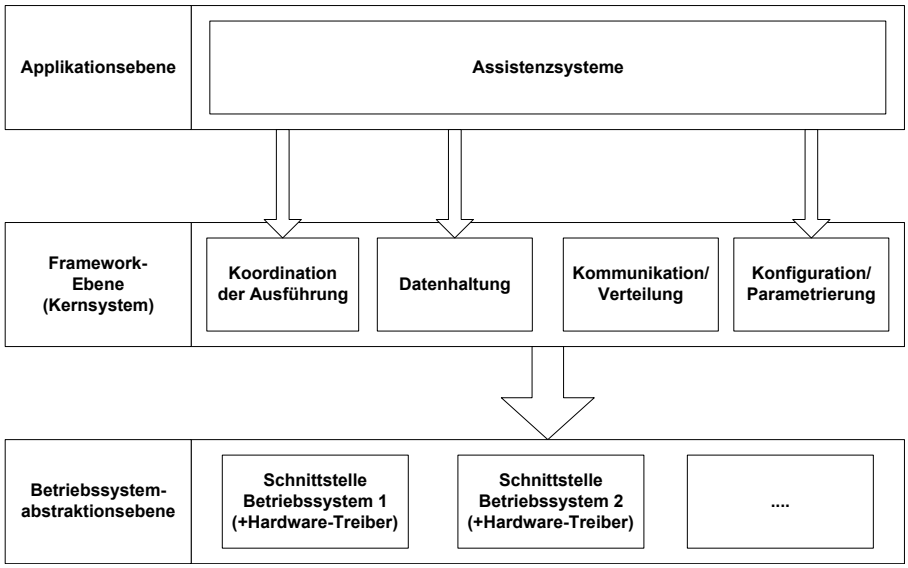

Abbildung 1. Schichtenmodell der Software-Architektur ANTSRT.

Wesentliche Aufgaben sind die Steuerung der Anwendung, konsistente Datenhaltung und eine transparente Verteilung der Applikationskomponenten. Die hier angesiedelten Dienste stellen eine zeitlich korrekte Ausführung der Fahrerassistenzsysteme in einer verteilten Umgebung sicher. Eine weitere Gruppe von Diensten ist für eine flexible Konfiguration bzw. Parametrierung von Anwendungen zuständig. Sie stellen eine Schnittstellensprache zur Verfügung, mit

deren Hilfe die Struktur und das gewünschte Verhalten der im Versuchsträger eingesetzten Assistenzsysteme sinnvoll definiert werden können. In der dritten Ebene sind die eigentlichen Softwarekomponenten der Applikation angesiedelt.

Ausgehend von den identifizierten Dienstklassen im Architekturmodell erfolgt eine Verfeinerung der Dienste in mögliche Strukturkomponenten und deren Beziehungen. Im Weiteren wird genauer auf die einzelnen ermittelten Kernkomponenten der Framework-Ebene eingegangen.

Hauptfaktor für die Definition der neuen Softwarestruktur ist das zu garantierende Verhalten der Assistenzsysteme. In den Versuchsträgern operieren verschiedene Teilsysteme zusammen, um das Fahrzeug zielgerichtet steuern zu können. So sind beispielsweise Hindernis- oder Fahrspurerkennung, die Umsetzung von speziellen Fahrerwünschen oder Aktuatoreingriffe mit unterschiedlichen zeitlichen Restriktionen parallel zu bedienen. Desweiteren muß das Verhalten der Teilsysteme in Abhängigkeit externer oder interner Reize steuerbar sein. Die Abbildung dieser Anforderungen resultiert in ANTSRT in einer Zustandsmaschine (Scheduler in Abbildung 2), welche einen dirigierenden Einfluß auf die Aktivitäten der Teilsysteme besitzt. Die Zustände spiegeln beispielsweise die aktuelle Umgebung des Fahrzeugs (Autobahn, Innenstadt, Landstraße) wider.

Jedes Teilsystem besteht aus einer Reihe von applikationsspezifischen Softwarekomponenten. Dazu gehören sogenannte funktionale Einheiten und Administratoren. Funktionale Einheiten implementieren die Verarbeitung von Umgebungsinformationen (z.B. Objekterkennung). Administratoren hingegen sind für die Rekonfiguration des Gesamtsystems bedeutsam. Sie definieren, unter welchen Umständen aus einem Systemzustand in einen anderen gewechselt wird (interne Reize). Beispielsweise kann bei dem Verlassen der Autobahn eine Anpassung der Fahrerassistenzsysteme an die neuen Straßenverhältnisse durch Administratoren ausgelöst werden. Beide Arten von Softwarekomponenten können in sequentieller oder paralleler Form zusammengesetzt werden. Die notwendige Koordinierung der Teilsysteme wird von eigenständigen, dem Haupt-Scheduler untergeordneten Steuerungskomponenten abgedeckt.

Die Kommunikation der funktionalen Einheiten und Administratoren untereinander erfolgt ausschließlich über eine zentrale Datenbank. Der entscheidende Vorteil ist, daß für eine erfolgreiche Kommunikation nur die Struktur der auszutauschenden Informationen (Datenbankeinträge) festgelegt sein muß. Aufgrund der gestiegenen Anforderungen bezüglich Echtzeitfähigkeit und Parallelität ist die Datenbank mit einem deadlockfreien Transaktionsmanagement und einem Priority-Ceiling-Protocol versehen [9] [10].

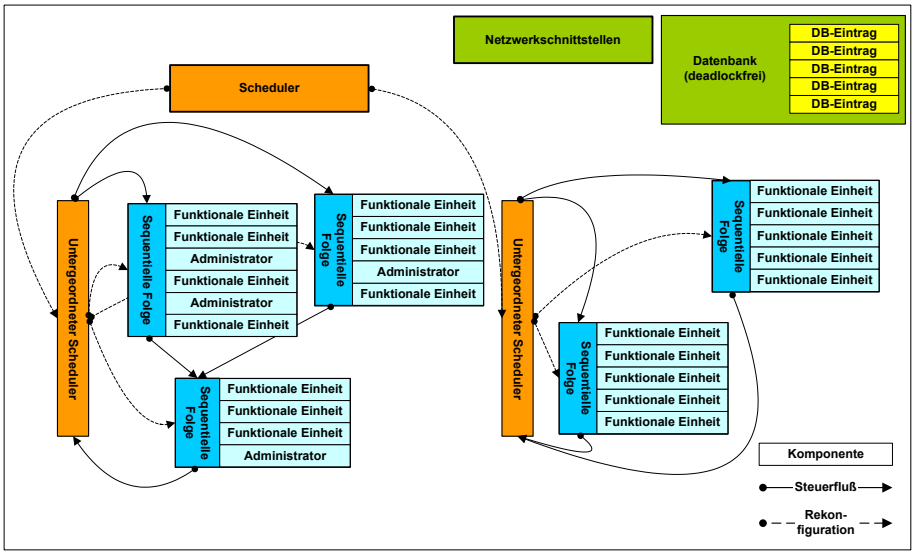

Abbildung 2. Komponentenstruktur der Software-Architektur ANTSRT.

Die Datenbank dient gleichzeitig für einen transparenten Datenaustausch über verschiedene Rechner hinweg. Einzelne Applikationskomponenten müssen nicht wissen, auf welchem Rechner die benötigten Informationen erzeugt werden. Das Kernsystem von ANTSRT stellt sicher, daß die Daten in den lokalen Datenbanken zur Verfügung stehen, bevor diese benötigt werden.

Zwischen den Instanzen des Softwaresystems ANTSRT wird die Kommunikation durch bereitgestellte Dienste des Frameworks realisiert. Diese definieren Richtlinien, wie Datenbankobjekte in Datenformate des verwendeten Kommunikationsprotokolls umzusetzen sind und umgekehrt. Um die Kommunikation zwischen Rechnern zeitlich steuern zu können (Datenkonsistenz), wird eine Zweiteilung des Übertragungsvorgangs vorgenommen. Einerseits übernimmt eine zentrale Einheit die Ansteuerung der Kommunikationsressourcen. Andererseits sind in den Steuerfluß der Applikationen an den benötigten Stellen Komponenten eingefügt, die relevante Daten aus der Datenbank an die zentrale Kommunikationskomponente weiterreichen, beziehungsweise empfangene Daten in die Datenbank eintragen.

4 Werkzeuge

Parallel zu ANTSRT werden Werkzeuge entwickelt, welche die Applikationsentwicklung auf Basis der neuen Softwarearchitektur unterstützen. Ein Schwerpunkt liegt in der automatischen Erzeugung von Software-Rahmenstrukturen für neue Komponenten von Fahrerassistenzsystemen. Beispielsweise können mittels einer graphischen Oberfläche Prototypen neuer funktionaler Einheiten sowie

deren Datenbankschnittstellen definiert und automatisch in Code umgesetzt werden (Abbildung 3). Gleichzeitig bietet das Werkzeug eine Unterstützung bei der Verwaltung und der Übersetzung der erzeugten Codes an.

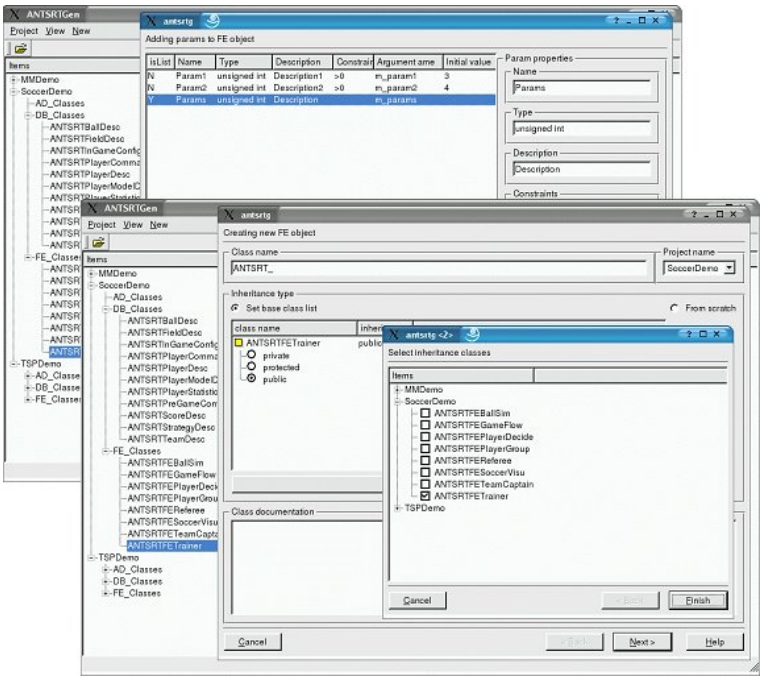

Abbildung 3. Der Komponentengenerator von ANTSRT.

Ein zweites Werkzeug erlaubt es, mit wenigen Handgriffen komplette Fahrerassistenzsysteme aus existierenden Software-Komponenten zusammenzustellen und deren Verhalten im Rahmen von ANTSRT zu definieren (Abbildung 4). Hierfür werden dem Anwender verschiedene Sichten zur Verfügung gestellt. In einer Sicht kann beispielsweise die statische Struktur des gewünschten Assistenzsystems definiert und parametriert werden. Andere Sichten erlauben es, die Zustandsmaschine zu modellieren und das Verhalten der Software-Komponenten in jedem Zustand festzulegen.

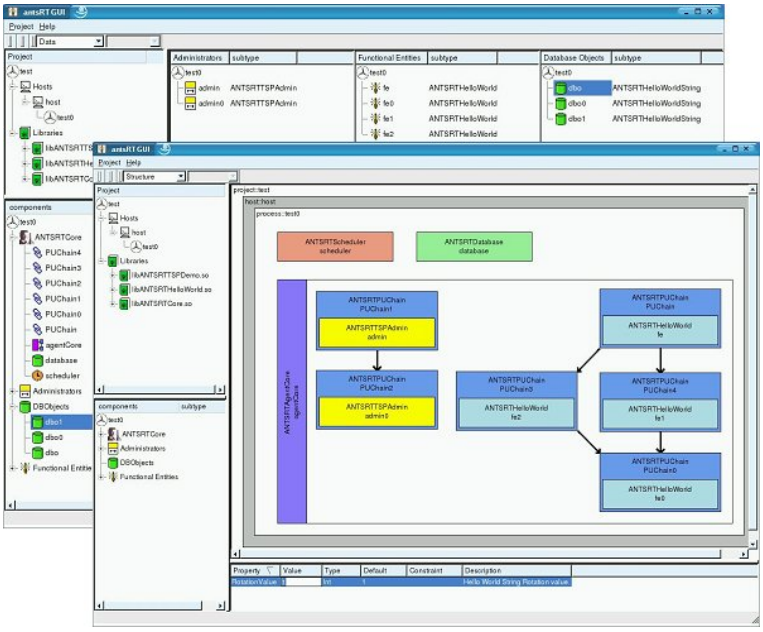

Abbildung 4. Erstellung von Fahrerassistenzsystemen mit dem ANTSRT Modeller.

5 Ergebnisse

Die Softwarearchitektur ANTSRT wird bereits prototypisch eingesetzt. In einem Versuchsträger kommt unter anderem das Assistenzsystem autonomes Stop & Go-Fahren zum Einsatz. Das Fahrzeug kann selbstständig vorausfahrenden Fahrzeugen folgen. Die Realisierung der Softwarearchitektur ANTSRT unterstützt aktuell die Betriebssysteme Linux sowie die Echtzeiterweiterung RTAI. Je nach vorhandener Hardware stehen für die Interaktion zwischen den Rechnern und Steuergeräten Mechanismen zur Kommunikation über CAN bzw. TCP (Ethernet/Myrinet) zur Verfügung.

6 Ausblick

Mit der Einführung weiterer Assistenzsysteme für ANTSRT schreitet auch die Entwicklung der Softwarearchitektur voran. Ein primäres Entwicklungsziel ist es, die Portabilität der Software durch Unterstützung des echtzeitfähigen Betriebssystems LynxOS zu erhöhen. Weiterhin wird die Integration diverser Sicherheitsmechanismen forciert. Exemplarisch ist eine Überwachung der Ausführung aller Assistenzsysteme (Watchdogs) inklusive einer Unterstützung von Rückfallebenen bei aufgetretenen Fehlersituationen zu nennen. Kurzfristig sind weitere

Verbesserungen in den Kommunikationsmechanismen der Software-Architektur - insbesondere bei der verteilten Synchronisation der Assistenzsysteme - geplant.

Literatur

1. Maurer, M., Dickmanns, E.D.: A SYSTEM ARCHITECTURE FOR AUTONOMOUS VISUAL ROAD VEHICLE GUIDANCE. In: IEEE Conference on Intelligent Transportation Systems ITSC'97, Boston, MA (1997)
2. Maurer, M.: Flexible Automatisierung von Straßenfahrzeugen mit Rechnersehen. PhD thesis, Universität der Bundeswehr München (2000)
3. Reichardt, D.: Kontinuierliche Verhaltenssteuerung eines autonomen Fahrzeugs in dynamischer Umgebung. PhD thesis, Universität Kaiserslautern (1996) Forschung F1M/IA Daimler–Benz.
4. Handmann, U., Kalinke, T., Tzomakas, C., Werner, M., v. Seelen, W.: An Image Processing System for Driver Assistance. In: 1998 IEEE International Conference on Intelligent Vehicles, Stuttgart (1998) 481–486
5. Handmann, U., Leefken, I., Tzomakas, C., v. Seelen, W.: A flexible Architecture for Driver Assistance. In: SPIE'99, SPIE's International Symposium on Intelligent Systems and Advanced Manufacturing; Conference "Intelligent Transportation Systems", Boston, USA (1999)
6. Görzig, S.: Eine generische Software-Architektur für Multi-Agentensysteme und ihr Einsatz am Beispiel von Fahrerassistenzsystemen. Shaker Verlag, Aachen, ISBN 3-8322-1470-4 (2003)
7. Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. Knowledge Engineering Review (1995)
8. Architecture Working Group of the Software Engineering Standards Committee of the IEEE Computer Society: Recommended Practice for Architectural Descriptions of Software- Intensive Systems. Standard 1471-2000 (2000)
9. Ramamritham, K.: Real-time databases. International Journal of Distributed and Parallel Databases (1996) 199–226
10. Ramamritham, K., Sivasankaran, R., Stankovic, J.A., Towsley, D.T., Xiong, M.: Integrating temporal, real-time and, active databases. SIGMOD Record **25** (1996) 8–12