Architektur für komponentenbasiertes **Infotainment-Framework**

Als Zulieferer der Automobilindustrie und Anbieter verschiedener Produkte im Bereich Infotainment-, Multimedia- und Kommunikationssysteme hat Funkwerk ein komponentenbasiertes Software-Framework entworfen und realisiert, mit dem sich der Entwicklungsprozess komplexer Fahrzeugsteuergeräte leichter beherrschen lässt. Das aufgezeigte Konzept ermöglicht die durchgehende Wiederverwendung vorhandener Komponenten, ist plattformunabhängig und ermöglicht einen effizienten Entwicklungsprozess einschließlich Rapid Prototyping und Testautomatisierung. In Verbindung mit automobilen Betriebssystem, COQOS, das von Open Synergy in 2009 zur Serienreife entwickelt wurde, lassen sich Infotainmentsysteme zukünftig sogar in Autosar-Umgebungen einbinden.

1 Einleitung

Infotainment-Systeme gewinnen im Fahrzeug immer mehr an Bedeutung. Während zum Beispiel die heutige Headunit vor 20 Jahren nur in Form eines hochwertigen, im Regelfall unvernetzten Radios existierte, stellen sich heutige Headunits als komplexe, leistungsfähige und voll vernetzte Radio-Navigationssysteme mit Anzeige- und Konfigurationsaufgaben für weitere Fahrzeugfunktionen dar. Für diese Systeme existieren leistungsfähige Hardwarebausteine und verschiedene, speziell auf Infotainment-Anwendungen zugeschnittene Betriebssysteme wie das Autosar-konforme COQOS [1] oder Microsoft Auto [2].

Für die Ebene oberhalb des Betriebssystems existieren allerdings keine vergleichbar universellen Komponenten. Die Software wird oftmals individuell auf das konkrete System zugeschnitten und lässt sich nur bedingt in anderen Produkten wiederverwenden.

Für den Geschäftsbereich Automotive Communication der Funkwerk AG wurde daher die Aufgabe definiert, durch die Entwicklung eines Infotainment-Softwareframeworks eine maßgebliche Steigerung der Effizienz in der Softwareentwicklung zu erreichen.

2 Wirtschaftliche und technische Ausgangsüberlegungen

Die Entscheidung für die Entwicklung eines einheitlichen Software-Frameworks fiel bereits zu Beginn des neuen Jahrtausends mit dem Ziel, möglichst flexibel und kostengünstig auf Kundenwünsche für individuelle Telematiksysteme zu reagieren.

Für diese Anwendungsfälle sollte die Software für Client-Server-Architekturen ausgelegt werden und über ein hardwareunabhängiges Design verfügen. Einmal entwickelte und praxiserprobte Komponenten sollten einfach wieder verwendbar sein und sich als Basis für zu-

künftige Lösungen flexibel miteinander kombinieren und aufgabenbezogen konfigurieren lassen.

3 Architekturentwurf

Während sich die Softwarearchitektur im PC-Bereich schon seit vielen Jahren weg von monolithartigen Komplettpaketen hin zu dynamisch zusammengestellten Lösungen weiterentwickelt hat, finden sich in Fahrzeug-Steuergeräten häufig noch spezielle Firmware-Pakete, die optimal auf die jeweilige Aufgabe und das Steuergerät angepasst sind. Mit der zunehmenden Leistungsfähigkeit der für „Embedded Systems“ geeigneten Hardware und den wachsenden Ansprüchen der Kunden und Endanwender hinsichtlich Produkt-Aktualität und deren kurzfristiger Anpassungsfähigkeit ist es möglich und auch sinnvoll, entsprechend flexible Softwarearchitekturen in modernen Fahrzeug-Steuergeräten einzusetzen.

Die im Geschäftsbereich Automotive Communication der Funkwerk AG entstandene Softwarearchitektur wird unter der Maßgabe weiterentwickelt, die verschiedenen Anforderungen unserer Kunden abzubilden und gleichzeitig dem steigenden Kosten- und Wettbewerbsdruck gerecht zu werden.

Maßgebliche Kriterien für das Einhalten dieser Anforderungen sind:

- strikte Aufgaben- und Ebenentrennung in den Klassen und Modulen
- einfache Portierung auf verschiedene Hardware und Betriebssysteme
- Konfigurierbarkeit und Wiederverwendbarkeit von Komponenten
- Verteilbarkeit von Funktionsblöcken auf verschiedene Steuergeräte
- einfache Implementierung von Tests auf Funktions-, Modul- und Komponentenebene.

3.1 Designziele

Für die Entwicklung des Frameworks wurden wesentliche Designziele zur Absicherung des komplexen Produktentstehungsprozesses festgeschrieben. Im Fall geänderter oder erweiterter Kundenanforderungen kann so flexibel auf geänderte Randbedingungen reagiert werden.

- Plattformunabhängigkeit: Das Framework soll auf Plattformen mit und ohne Betriebssystem portierbar sein. Zurzeit existieren Versionen für Windows CE, Microsoft Auto, embedded-Linux für die Anwendung im Fahrzeug sowie Windows XP und Linux für die Entwicklung auf dem PC, **Bild 1**.
- Die Komponenten eines Gesamtsystems können auf einem oder mehreren, miteinander verbundenen Steuergeräten installiert werden. Hierfür muss jede direkte Abhängigkeit zwischen Komponenten vermieden werden. Die Komponenten sind dabei lose gekoppelt und kommunizieren asynchron über einen virtuellen Nachrichtenbus. Die notwendige Kommunikation zwischen den Komponenten wird über eine Frameworkkomponente gesteuert, die über entsprechende Informationen zur Verteilung der Komponenten in Form einer Routing-Tabelle verfügt.
- Testbarkeit der Software: Die durchgehend von Standard-Komponenten abgeleiteten spezifischen Komponenten müssen sich dementsprechend auch über standardisierte Tests der Eltern-Komponenten testen lassen. Darüber hinaus werden Testkomponenten für Modul- und Systemtests erstellt.

3.2 Entwicklungsvorgaben

Um die selbst gestellten Designziele zu erfüllen, wurden die grob gefassten Anforderungen in Teilziele und Einzelkriterien aufgegliedert und wie folgt vereinbart:

- Es wird ein neu zu entwickelndes Komponentenframework eingesetzt, das frei von externen Abhängigkeiten auszulegen ist.
- Das Framework definiert einen „Container“ für Funktionen und Funktionsblöcke und garantiert so einheitlich gestaltete Komponenten. Die Komponentenverwaltung, die Funktions- und Methodenaufrufe sowie die Kommunikation innerhalb des Frameworks werden über einheitliche Verfahren realisiert.
- Das Framework wird mit Hilfe geeigneter Abstraktionsebenen plattform- und betriebssystemunabhängig realisiert. Die Anpassung an die jeweilige Hard- und Software erfolgt über eine spezielle Middleware.

Die Autoren

Dipl.-Ing., Dipl.-Inform. (FH) Thomas Kern
ist Leiter der Geschäftseinheit OEM in der Funkwerk Dabendorf GmbH in Berlin.

Dipl.-Inform. Christian Herrmann
ist Leiter Business Development in der Funkwerk Dabendorf GmbH in Berlin.

B. Eng. Katja Lampe
ist Systemarchitektin im Business Development in der Funkwerk Dabendorf GmbH in Berlin.

Dipl.-Ing. Dirk Maaßen
ist Leiter Softwareentwicklung der Funkwerk eurotelematik GmbH in Ulm.

Dipl.-Ing. Martin Eich
ist Leiter Basisentwicklung der Funkwerk eurotelematik GmbH in Ulm.

Dipl.-Inform. Hans Christian Jaud
ist Leiter Applikationsentwicklung der Funkwerk eurotelematik GmbH, Business Unit OEM in Berlin.

Bild 1: Plattformunabhängigkeit am Beispiel zweier Produkte

- Wesentliche Aufgaben des Frameworks werden als Komponenten realisiert. Hierbei werden die Komponentengruppen Service, Schnittstellentreiber, MMI und Applikation unterschieden.
- Als Programmiersprache wird C++ eingesetzt, es werden alle methodischen Vorteile der Objektorientierung genutzt. Das Framework kann aber auch auf andere Programmiersprachen wie Java erweitert werden. Falls notwendig, ist sogar die Verwendung von C möglich. Die im Fall einer prozeduralen Sprache wie C fehlenden Zugriffsschutz-Mechanismen müssen dann soweit wie möglich nachgebildet werden.

Die Arbeiten am Framework sind von Beginn an darauf ausgerichtet, von mehreren Entwicklern an verschiedenen Standorten durchgeführt zu werden. Zur Absicherung des Wissenstransfers über Entwicklergenerationen (zum Beispiel für die Einarbeitung neuer Mitarbeiter) und zur Sicherstellung der Integrität aller Entwicklungsergebnisse wurden weitere Festlegungen getroffen:

- Größere Funktionsblöcke werden als Komponente ausgeführt. Sie müssen dabei der gleichen Thematik angehören. Austauschbarkeit und Wiederverwendbarkeit beeinflussen ebenfalls die Abgrenzung von Komponenten.
- Funktionalitäten der Komponenten werden durch Interfaces bereitgestellt. Von „Außen“ ist der Zugriff auf die Funktionalitäten der Komponenten nur über eines oder mehrere Interfaces möglich. Die Einhaltung dieser Forderungen ist beispielsweise mit den Mitteln der Objektorientierten Programmierung sowie der konsequenten Anwendung von „public“ und „private“ Deklarationen möglich.
- Interfaces sind virtuelle Funktionen, denen Komponenten-Nachrichten als Parameter übergeben werden. In C++ sind Interfaces Klassen mit virtuellen Funktionen, deren reale Implementierung damit verborgen ist und austauschbar bleibt.
- Jede Komponente ist über eine statische, fest konfigurierte ID (Adresse) eindeutig gekennzeichnet.
- Das Framework wird im Sinne eines Meta-Modells umgesetzt, besteht also ebenfalls aus Komponenten und Interfaces, **Bild 2**.

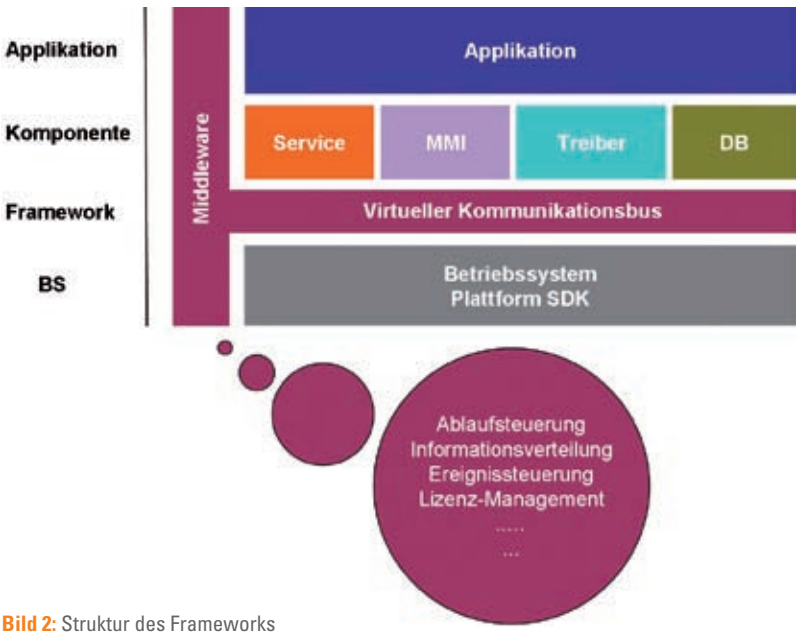

Bild 2: Struktur des Frameworks

4 Produktdesign

Die Softwarearchitektur für komplexere Produkte wie Infotainmentsysteme lässt sich im Sinne des hier vorgestellten Frameworks über vier Komponentengruppen beschreiben.

- Servicekomponenten: Diese Komponenten realisieren die eigentliche Funktionalität des Systems, zum Beispiel für Navigation (Routingservice, Kartenrendering, Map-Matching, et cetera), Telefonie und Kontaktverwaltung (Phonecontrol, Telefonbuch). Ein direkter Zugriff der Produktkomponenten untereinander ist nicht vorgesehen. Der Datenaustausch wird grundsätzlich über das Framework sichergestellt und durch die Applikationskomponen-

te gesteuert. Ein wesentlicher Vorteil dieser Unabhängigkeit der Komponenten zeigt sich bei deren problemlos möglichen Wiederverwendung.

- Schnittstellenkomponenten: Diese Komponenten regeln die Kommunikation über Systemgrenzen hinweg und kommen beispielsweise bei der physikalischen Einbindung von Mobiltelefonen oder anderen mobilen Geräten zum Einsatz. Schnittstellen können in Form von drahtgebundener Verbindung (Mobiltelefon-Ladeschalen, iPod-Anschlusskabel) oder auch drahtlos (Bluetooth, WLAN) realisiert werden. Auch die Anbindung an den Fahrzeugbus (beispielsweise via CAN) erfolgt über eine Schnittstellenkomponente. Wird COQOS als Betriebssystem eingesetzt, kommuniziert die Fahrzeugbus-Schnittstellenkomponente nicht direkt mit dem CAN-Controller, sondern mit dem entsprechenden virtuellen Controller. Dieser dient als Schnittstelle zur Autosar-Umgebung von COQOS.
- MMI-Komponenten: Die Mensch-Maschine-Schnittstellen-Komponenten regeln die Ein- und Ausgabe von Informationen, zum Beispiel auf dem Display oder per Sprache. Ein weiterer wesentlicher Aufgabenbereich ist die Dialogsteuerung, das heißt die Verarbeitung von Eingaben und die Umsetzung dieser Ereignisse in einen geregelten Prozessablauf.
- Applikationskomponenten: Für ein funktionsfähiges Gesamtsystem müssen die einzelnen Komponenten kontrolliert arbeiten und miteinander kommunizieren. Durch die Aggregation einzelner Service- und MMI-Komponenten werden Applikationskomponenten gebildet. Die Applikationskomponente übernimmt die Gesamtsteuerung des Systems, lädt und verarbeitet Konfigurationseinstellungen, übergibt Parameter an die restlichen Komponenten und beschreibt das Systemverhalten (wie die Abfolge von Menüs, die aktuelle Sprache).

Jede Komponente ist konfigurierbar ausgelegt und kann in ihrem Verhalten ohne Neukompilation über Parameter und Konfigurationsdateien an die jeweiligen Erfordernisse angepasst werden. So lässt sich beispielsweise die Telefonbuchkomponente hinsichtlich ihrer Kapazität (Kontaktanzahl) und Sortiervorgaben (Vorname, Nachname oder umgekehrt) leicht auf die jeweiligen Kundenwünschen abstimmen. Beispielhaft werden im Folgenden einige Komponenten näher betrachtet.

4.1 Ein- und Ausgabesteuerung

Die Ein- und Ausgabesteuerung wird über verschiedene MMI-Komponenten realisiert, die jeweils eine spezifische Aufgabe übernehmen. Für ein verdeckt verbautes Bluetooth-Telefonie-Steuergerät kommen zum Beispiel folgende MMI-Komponenten zum Einsatz:

- Display-Komponente: Diese Komponente leitet die von der Applikationskomponente kontrollierten Anzeigewünsche anderer Framework-Komponenten über die CAN-Schnittstelle (ebenfalls eine Framework-Komponente) an das Fahrzeugdisplay weiter. Die Funktionsweise entspricht einem Gateway, das heißt, hier werden die Anzeigewünsche vom intern verwendeten Nachrichtenformat in das Nachrichtenformat des Fahrzeugbusses umgewandelt und mit den Adressinformationen (beispeilsweise für das Kombiinstrument) versehen. Diese aufbereiteten Anzeigeinformationen werden anschließend über den Nachrichtenbus des Frameworks an die CAN-Schnittstellenkomponente weitergeleitet.
- Lenkradtasten-Komponente: Diese Komponente erhält über die CAN-Schnittstellenkomponente Steuerungsbefehle vom Fahrzeug, wandelt diese in das interne Nachrichtenformat des Frameworks um und leitet die Informationen an die Applikationskomponente weiter.
- Sprachsteuerungs-Komponente: Über Sprache eingehende Kommandos werden von der Sprachsteuerungs-Komponente erkannt, verarbeitet, in das interne Nachrichtenformat umgewandelt und an die Applikationskomponente weitergeleitet. In die Sprachsteuerungskomponente sind Bibliotheken externer Anbieter wie Nuance eingebunden, welche den wesentlichen Teil der Komponentenfunktionalität ausmachen.
- Sprachansage-Komponente: Die Sprachansage funktioniert vergleichbar zur Display-Komponente und bindet für die Umwandlung der Ansage in Sprache (TTS, Text To Speech) die Bibliothek eines externen Anbieters ein.

Bild 3: FB6000

4.2 Applikations-Komponente

Die Applikationskomponente übernimmt zentral die Steuerung des Gesamtsystems und kontrolliert den Informationsfluss zwischen den anderen Komponenten. Sie hat ausschließlichen Zugriff auf die MMI-Komponenten und behält die Entscheidungshoheit darüber, mit welcher Priorität Ausgabe- und Eingabeinformationen zu bearbeiten sind. So kann sichergestellt werden, dass wichtige Informationen vorrangig weitergeleitet werden, eine Reizüberflutung des Fahrers vermieden wird und überschneidende, sich gegebenenfalls aufhebende Anweisungen keine unnötige Auslastung des Gesamtsystems verursachen.

5 Praktische Anwendungsbeispiele

Auf Basis des Frameworks und aufbauend auf gleichen oder ähnlichen Hardware-Bausteinen wurden bereits mehrere Produkte realisiert. Zwei davon werden unter Herausstellung der wesentlichen Merkmale kurz vorgestellt.

5.1 Infotainment und Telematikplattform FB6000

Die FB6000 beschreibt ein modulares Produkt, das aus drei eigenständig funktionsfähigen Komponenten besteht. Das Display- und Navigationsmodul (PNA) sowie die Telematikeinheit (FB4000) basieren auf Windows CE 5.0 und dem Funkwerk-Komponentenframework. Das Freisprechmodul (EGO) basiert auf einer älteren Entwicklung und hat nur Teile des Komponentenframeworks implementiert. Alle drei Module lassen sich per Kabel miteinander verbinden und können über die entsprechenden Schnittstellenkomponenten im Framework miteinander kommunizieren. Eingaben sind über Touchscreen, Funkfernbedienteil, Sprachsteuerung oder über einen Telematikserver möglich.

Für den Nutzer führen alle Eingaben zu entsprechenden Rückmeldungen des Systems, die überwiegend optisch auf dem Display, im Einzelfall aber auch akustisch über die Fahrzeuglautsprecher ausgegeben werden. Insbesondere das MMI ist dabei auf die besonderen Möglichkeiten der einzelnen Module optimiert: Der Touchscreen wird durch den PNA, Funkfernbedienung und Sprachsteuerung werden durch das EGO-Modul und Informationen vom Telematikserver durch die FB4000 kontrolliert. Zu jedem Zeitpunkt kontrolliert die Applikationskomponente der FB6000 alle Ein- und Ausgaben, so dass für den Nutzer kein Unterschied zu einem integrierten System (einem gemeinsamen Steuergerät) erkennbar wird, **Bild 3**.

5.2 Autotelefon EGO Phone

Das EGO Phone ist ein GSM-Telefon zum Festeinbau im Fahrzeug und erfüllt damit die Funktionen einer Freisprechanlage.

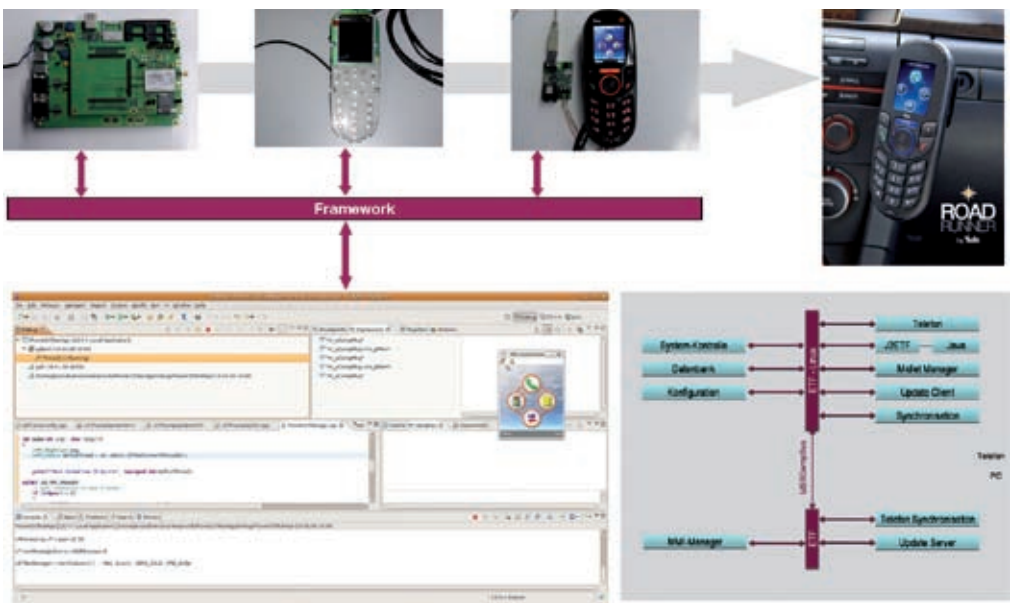

Bild 4: Schnelle Entwicklungsprozesse durch Rapid Prototyping und Hardware-in-the-Loop

Bild 5: Headunit-Prototyp CeBIT 2009

Bei der zeitkritischen Neuentwicklung eines derartigen Produkts zeigt sich ein herausragender Vorteil des plattformunabhängigen Framework-Konzepts. Zum Beispiel konnte die Softwareentwicklung vollkommen unabhängig von der Verfügbarkeit der Hardware gestartet und weitestgehend fertiggestellt werden. Noch vor dem ersten Hardware-Muster konnten so Abstimmungen mit dem Kunden bezüglich der Menüstrukturen, -abläufe, Symbole und Bedienhandlungen erfolgen. In den darauf folgenden Schritten wurden schrittweise Softwarekomponenten auf die Hardware-Muster portiert und deren Verhalten, Funktionsfähigkeit und Fehlerfreiheit verifiziert, **Bild 4**.

6 Bewertung und Ausblick

Mit dem Konzept des komponentenbasierten Infotainment-Frameworks kann die Entwicklung von einfachen und komplexen Produkten jederzeit beherrscht und damit zeitgerecht und qualitativ hochwertig umgesetzt werden. Hierzu sind alle Komponenten in sich abgeschlossen, haben klar definierte Schnittstellen und daraus resultierende Aufgaben. Durch die variable Zusammenstellung der Komponenten können auch vollkommen unterschiedliche Produkte auf gleicher technischer Basis entstehen.

Die klar definierten Schnittstellen ermöglichen nicht nur ferngesteuerte (PC-basierte) Unittests, sondern auch die Aufzeichnung von Nutzereingaben und Systemreaktionen. In späteren Testszenarien können die aufgezeichneten Nachrichten wieder in das System eingespielt und als Basis für weitgehend automatisierte Tests von Einzelkomponenten und des Gesamtsystems verwendet werden.

Auch die Kombination von Fahrzeug- und Infotainmentfunktionen wird mit Hilfe des Autosar-konformen COQOS-Betriebssystems leicht möglich. Für kommende Generationen von Fahrzeugsteuergeräten ergeben sich so erhebliche Vorteile für den Produktentstehungsprozess und die erreichbaren Qualitätsansprüche. Die von Funkwerk auf verschiedenen Messen ausgestellten Konzepte und Prototypen zeigen den Weg und das Ziel deutlich auf, **Bild 5**.

Literaturhinweise

[1] OpenSynergy, COQOS. ATZe 01/2009 S. 40ff
[2] Microsoft, MS Auto. ATZe 06/2008 S. 36ff
[3] Continental, Fahrerarbeitsplatz. ATZe 01/2008 S.42ff