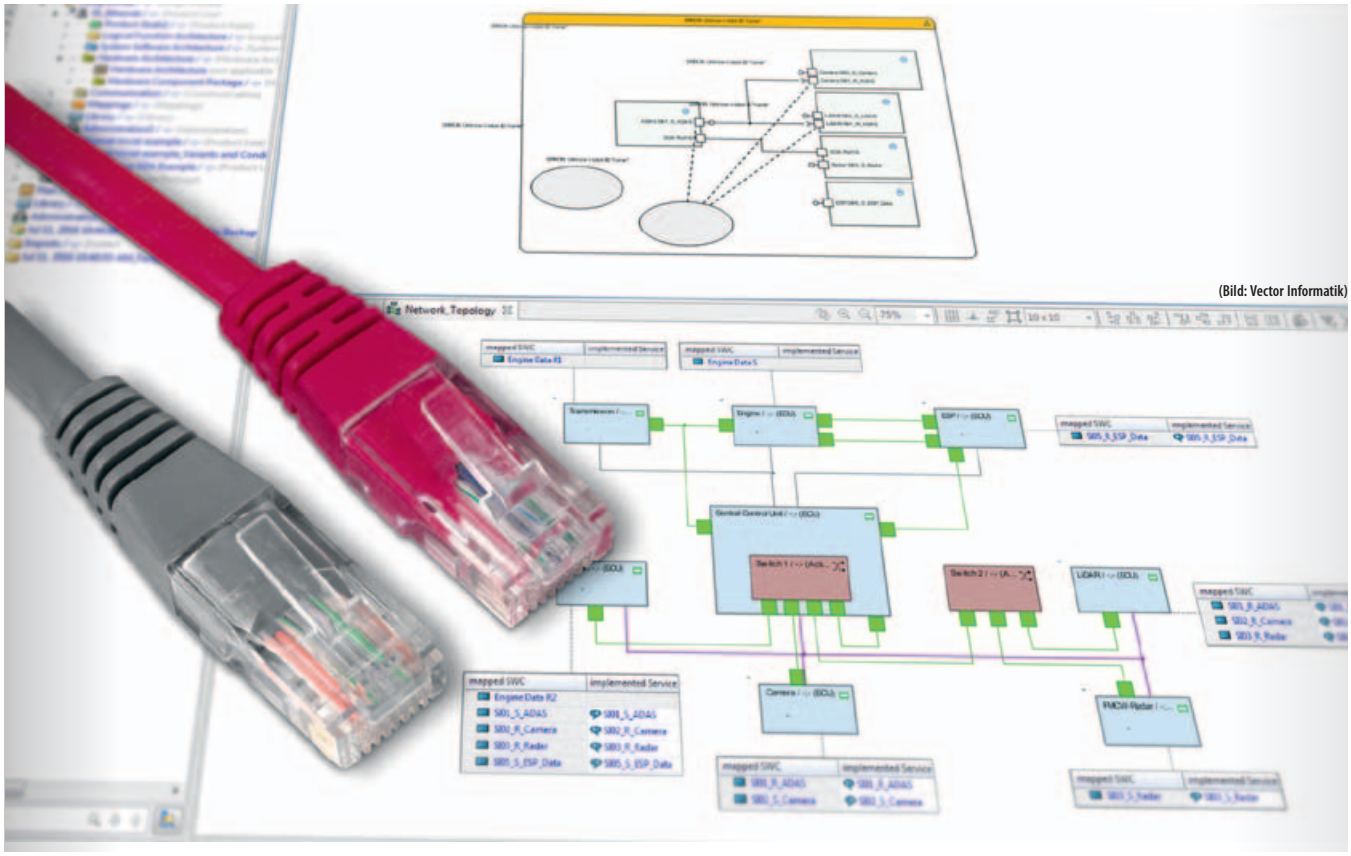
Service-orientierte Architekturen und Ethernet im Fahrzeug:

Auf dem Weg zum fahrenden Rechenzentrum



(Bild: Vector Informatik)

Service-orientierte Architekturen dienen in der IT-Industrie bereits seit Jahren dazu, verteilte Systeme zu beschreiben und zu strukturieren. Service-orientiertes Design gewinnt aber auch in der Automobilindustrie massiv an Bedeutung: Wie sonst kann der immer dringlicher werdende Wunsch, den Funktionsumfang des Automobils ähnlich den Möglichkeiten heutiger Smartphones anzupassen, zuverlässig realisiert werden? Auch zusätzliche Anforderungen aus der Modellpflege sowie die Einführung von autonomem Fahren und V2X-Kommunikation sind zu bewältigen.

Von Markus Helmling

Die Herausforderungen an die Software-Entwicklung für moderne Fahrzeuge sind gewaltig. Denn neben die seit Jahren wachsende Komplexität im Automobilbau treten mit der Einführung von Service-orientierte Architekturen (Service-oriented Architecture, SOA) zusätzlich komplett neue Entwicklungsparadigmen und Abläufe. Bereits 1968 postulierte Conway: Organisationen, die Systeme entwerfen, sind auf Entwürfe festgelegt, welche die Kommunikationsstrukturen dieser Organisationen abbilden [1].

So verwundert es nicht, dass die Software-Architektur heutiger Fahrzeuge meist siloartig strukturiert ist und traditionelle Organisationsstrukturen nach Domänen, wie Antrieb, Karosserie und Fahrwerk, abbildet. Eine wirkungsvolle Einführung einer SOA braucht aber eine völlig andere, domänenübergreifende Abstimmung und Kommunikation. Denn hier verschieben sich Zuständigkeiten über die bisherigen Abteilungs- und Organisationsgrenzen; Kooperation und Kollaboration im Unternehmen werden zur impliziten Voraussetzung für das Festlegen und Weiterentwickeln einer SOA und deren Schnittstellen. Der organisatorische Umbruch wirkt auch in den Personalbereich hinein. Es entstehen neue Rollen und Stellenanforderungen im Un-

ternehmen: Die nötigen Spezialisten müssen ausgebildet oder rekrutiert werden.

Auf der technischen Seite verstärken die Anforderungen durch eine Service-basierte Kommunikation den Trend des enorm wachsenden Bandbreitenbedarfs in der Onboard-Vernetzung. Fahrzeughersteller setzen hier zunehmend auf die in der IT-Industrie bewährte Ethernet-Technologie. Mit ihr lassen sich geeignete Bandbreiten für Anwendungen und Kommunikation innerhalb des Fahrzeugs sowie zwischen dem Fahrzeug und dessen Umwelt oder dem Internet umsetzen. Für die Entwicklung zukunftssicherer Elektrik-/Elektronik-Architekturen sind daher Werkzeuge nötig, die diese Anforderungen beim Entwerfen und Verwalten der Systeme berücksichtigen.

Vorteile der Service-Orientierung

Mit einer SOA entsteht eine durch klar definierte Schnittstellen charakterisierte Service-Landschaft. Diese Schnittstellen sind idealerweise sowohl syntaktisch als auch semantisch formal eindeutig beschrieben. Die klare Definition erlaubt es, Dienste (Services) zu strukturieren. So kommen in der Regel Schichtenarchitekturen zum Einsatz, welche die Abhängigkeiten von Services verständlich machen und klare Design-Regeln darstellen. Weiterhin sind in einer SOA die Komponenten mit einem Service-Bus als Middleware lose aneinander gekoppelt – die Middleware ist Vermittler zwischen dem Anbieter eines Service (Service Provider) und dem Konsumenten des Service (Service Consumer). Sie regelt die Kommunikation zwischen Service Provider und Service Consumer bei Systemstart und definiert in der Regel auch die Datenserialisierung (**Bild 1**). Diese „Serialisierer" genannte, eindeutige Berechnungsvorschrift beschreibt die Datenübertragung im physikalischen Bussystem. Der Serialisierer legt fest, auf welche Art und Weise die Daten in einen seriellen Bitstrom transformiert und wie sie auf Empfangsseite deserialisiert werden.

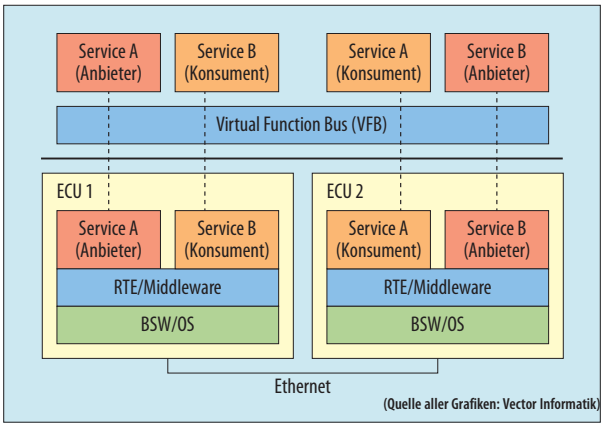


Bild 1. Schematische Darstellung von Services und Middleware.

Durch den Einsatz der Middleware entsteht die Verbindung zwischen Service Provider und Service Consumer zur Laufzeit – und nicht wie bisher zur Design-Zeit des Systems. Dadurch werden verschiedene Probleme der bisherigen Software-Entwicklung adressiert. Zum einen können Teil-Updates des Systems durchgeführt werden: Sofern eine Abwärtskompatibilität sichergestellt wird, wirkt sich die Änderung eines Service nicht zwingend in einer Software-Anpassung auf alle Konsumenten des Service aus. Durch den Einsatz eines Versionsmanagements und der Revisionierung von Service-Schnittstellen sind so auch „sanfte" Migrationsszenarien möglich: So können Teile des Systems eine ältere Version eines Service verwenden, gleichzeitig nutzt ein anderer Teil bereits einen neuen Aspekt der Service-

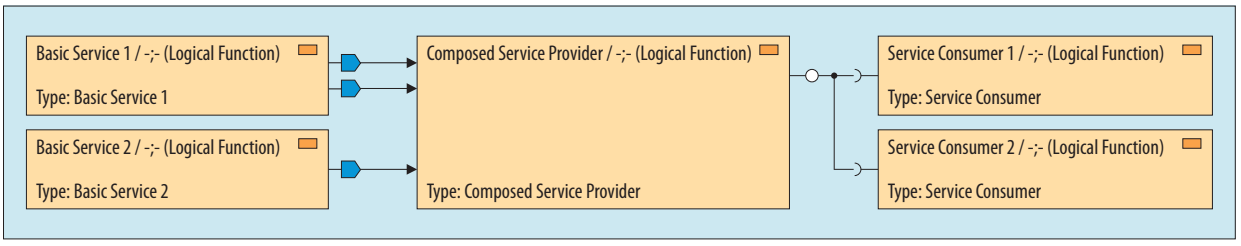


Bild 2. Logische Architektur zur statischen Beschreibung der Service-Orchestrierung.

Schnittstelle. Darüber hinaus kann Ausfallsicherheit per Design eingeführt werden: Beim Ausfall einer Service-Instanz kann eine andere, gleichwertige oder zumindest kompatible Service-Instanz an deren Stelle treten. Das hat in verteilten Systemen die Folge, dass nicht notwendigerweise alle an einem Service beteiligten Steuergeräte aktualisiert werden müssen.

Ein konsequentes Service-Design lässt nicht zuletzt auch mächtigere Design-Patterns in der Implementierung zu: So sind beispielsweise Remote Procedure Calls (RPC) in einer Service-Architektur gängige Praxis. RPCs kamen bisher nur innerhalb eines Steuergeräts zum Einsatz und sorgen nun dafür, dass Client-Server-Paradigmen auch in der Onboard-Vernetzung genutzt werden. In der Folge wird dadurch die bisher stark ausgeprägte Signalorientierung im Kommunikations-Design von Fahrzeugen verdrängt.

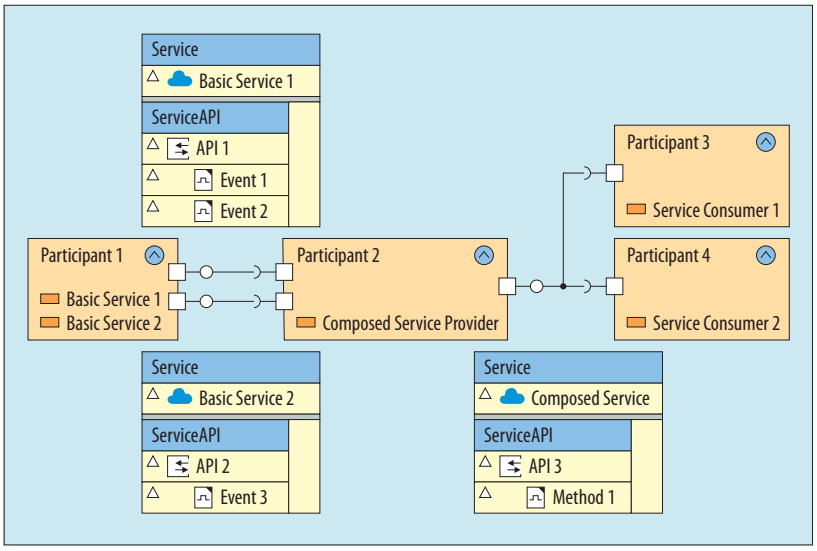


Bild 3. SOA-Diagramm zur Definition von logischen Service-Knoten (Participants) in PREEvision.

SOA-Methodik als Top-Down-Ansatz

Möchten Automobilhersteller und Zulieferer die Vorteile einer SOA ausspielen, brauchen sie eine geeignete Methodik, die den Entwurf neuer Systeme mit allen notwendigen Entwicklungsschritten unterstützt. Es gilt hierbei die Komplexität zu beherrschen und gleichzeitig qualitätsgesicherte Designs zu ermöglichen. Die im Folgenden beschriebene Methodik erläutert den Entstehungsprozess von den Anforderungen über die Service-Architektur bis hin zum Kommunikations-Design. Das Ziel dabei ist eine AUTOSAR-Classic-konforme Systembeschreibung für ein Fahrzeug oder Fahrzeug-Teilsystem. Die Methodik lässt sich aber ebenso im Kontext von AUTOSAR Adaptive oder auf Systeme außerhalb von AUTOSAR anwenden:

Am Anfang steht die Idee einer neuen Funktion, deren Merkmale festgehalten werden. Das geschieht heute meist nach Anwendungsfällen strukturiert. Bewährt und etabliert hat sich dafür die Notation der Use-Case-Diagramme aus dem UML-Standard. Mit Use Cases lassen sich aus einem Katalog von Merkmalen Dienste ableiten. Um Dienste zu strukturieren und Abhängigkeiten aufzuzeigen, werden verschiedene Sichten benötigt. Eine naheliegende und bereits etablierte Sicht orientiert sich am Informationsfluss und veranschaulicht so die Abhängigkeiten von Funktionen. Im Systemdesign-Werkzeug PREEvision hat sich dafür die sogenannte „logische Architektur" etabliert: Hier lässt sich die logische Architektur in Form eines Blockdiagramms abbilden und beschreibt so unabhängig von der Umsetzung in Hardware oder in Software den funktionalen Zusammenhang einzelner Service-Funktionen durch Wirkketten (**Bild 2**).

Ergänzend definiert das auf die Beschreibung von Service-Architekturen spezialisierte UML-Profil SoaML logische Knoten, welche „Participants" genannt werden. Participants treten sowohl als Anbieter als auch als Konsumenten von Services auf – der Austausch erfolgt in beiden Fällen über sogenannte SOA-Ports. Mit ihnen werden Abhängigkeiten zwischen Services sichtbar – beispielsweise wenn ein Participant einen oder mehrere (Basis-)Services konsumiert und andererseits höherwertige Services anbietet. Dieser Entwurfsschritt wird als Orchestrierung bezeichnet und ist ähnlich der weitverbreiteten Wirkketten-Sicht für Steuerungs- und Regelungsfunktionen.

In volldynamischen Systemen, bei denen innerhalb des Systems mehrere Instanzen eines Service in unterschiedlichen Versionen vorliegen, ist diese Darstellung der Allokation von Services auf Participants allerdings nur ein sich zur Laufzeit ergebendes, mögliches Szenario. Weil im Automobilbereich aber aktuell hauptsächlich statische Zuweisungen von Services auf dedizierte Hardware vorgenommen wer-

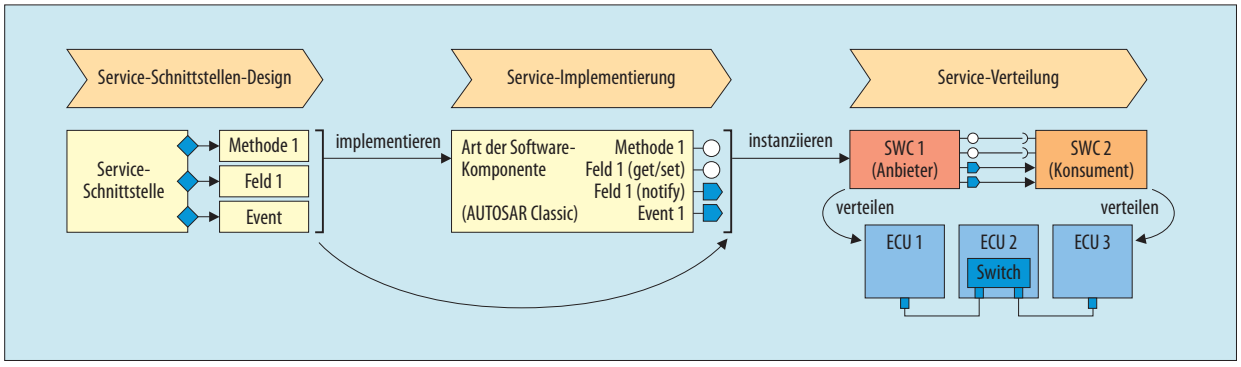


Bild 4. Vom Service-Entwurf zum Kommunikations-Design.

den, kann diese Darstellung auch als Allokation einer Service-Rolle (Provider/Consumer) auf ein Steuergerät (ECU) verstanden werden. Davon abgesehen werden mit Participants verschiedene, sich zur Laufzeit ergebene Konstellationen sichtbar. Die Darstellung von logischen Service-Knoten als Participants kann so auch als hilfreiches Analysewerkzeug angesehen werden. Diese Konstellationen zu kennen und zu dokumentieren ist nicht zuletzt für den Test und die Absicherung solcher Systeme eine wichtige Grundlage (**Bild 3**).

Die oben angesprochenen SOA-Ports werden durch ein Service-Interface typisiert, das durch mehrere Aspekte eindeutig beschrieben wird. Zunächst ist zu klären, welche Rolle ein Port einnimmt – bietet er einen Service an oder konsumiert er ihn? Die Eigenschaften des Interface werden hier syntaktisch beschrieben: allgemeingültige und technologieunabhängige Eigenschaften wie verfügbare Methoden, Felder und Events – aber auch zu diesem Zeitpunkt bereits bekannte, von der Middleware unterstützte Datentypen, zum Beispiel die der Parameter einer Methode. Das statische Verhalten eines Service ist damit vollständig beschrieben. Die semantische Beschreibung des Service-Vertrags (Service Contract) zwischen den SOA-Ports erfolgt häufig textuell. Hier werden unter anderem zeitliche Abläufe zwischen Service-Rollen festgelegt und Protokolle abgebildet. Dieser Entwurfsschritt wird auch Service-Choreografie genannt. Zur Visualisierung von Abläufen werden meist die von der UML vorgeschlagenen Kollaborations- oder Message Sequence Charts (MSC) eingesetzt.

Die zunächst technologieunabhängige, formale Beschreibung eines Service wird typischerweise über eine Schnittstellenbeschreibung (Interface Description Language, IDL) transportiert. In modernen Systemdesign-Werkzeugen wie PREEvision kann hiermit bereits die äußere Struktur der Implementierung abgeleitet werden, beispielsweise für die Anwendung im Kontext von AUTOSAR Classic. Es entstehen so Implementierungsrümpfe, die für die weitere Detaillierung des Designs und dessen Implementierung benötigt werden. Außerdem werden die Rolle des Service und die Beschreibung des Service Interface übernommen.

Der Vorteil dieses durchgängigen Ansatzes: Änderungen an der Rolle eines Service oder der Beschreibung der Service-Schnittstelle können durch Synchronisationsmechanismen automatisiert auf die technologiespezifische Implementierung übertragen werden. Aus den IDL-Beschreibungen werden dann Software-Design-Schnittstellen abgeleitet und in der AUTOSAR-Classic-Plattform als Sender-Receiver- und Client-Server Interfaces abgebildet. Wurden bereits zur Technologie passende Dateitypen gewählt, können diese automatisiert übernommen werden. Für die AUTOSAR-Adaptive-Plattform wird aktuell die Einführung eines speziellen „Service Interface" diskutiert. Es soll die oben beschriebenen Eigenschaften direkt beinhalten und den Vorteil einer technologieunabhängigen Beschreibung von Services in vollem Umfang nutzen. Angestrebt ist das Ableiten von Implementierungsrümpfen für unterschiedliche Technologien aus derselben Service-Definition per Knopfdruck. Das Middleware-spezifische Startup-Verhalten soll im Anschluss an diesen Schritt festgelegt und parametriert werden. Dieser Prozess wird als „Service Discovery" bezeichnet.

Kommunikations-Design

Die Bedatung und die Strukturen der Kommunikationsbeschreibung hängen im weiteren Vorgehen von der eingesetzten Bustechnologie und der Middleware ab. AUTOSAR verwendet als Transformer für die Service Discovery und die Datenserialisierung standardmäßig SOME/IP. In AUTOSAR ist aber auch der Einsatz anderer Transformer und Middleware vorgesehen. Werden Ethernet und IP als Kommunikationsweg eingesetzt, ist die Definition von Sockets mit IP-Adresse, Transportprotokoll und Port ein wesentlicher Arbeitsschritt. Und während für AUTOSAR Adaptive eine Beschreibung von Socket-Adressen ausreichen wird, ist für AUTOSAR Classic auch die Signalebene zu definieren – denn der AUTOSAR-Classic-Basis-Software-Stack ist auf eine signalbasierte Kommunikation ausgelegt. An dieser Stelle bieten integrierte Systemdesign-Werkzeuge einen entscheidenden Vorteil: Mit ihnen ist es möglich, die Signalkommunikation abzuleiten. Notwendig ist dafür die Service-Beschreibung und die Information, an welcher Stelle ein Service im System vorgesehen ist – also welches konkrete Steuergerät den Provider und welches den Consumer implementiert. PREEvision übernimmt an dieser Stelle auch weitere Eigenschaften aus der Service-Beschreibung in die Kommunikationsbeschreibung, beispielsweise Service- und Methoden-IDs (**Bild 4**).

Warum PREEvision?

Das Entwicklungswerkzeug PREEvision unterstützt den methodisch durchgängigen Entwurf einer Service-orientierten Architektur. Der Anwender wird von der Definition der Service-Schnittstellen über die Festlegung der Interaktion von Diensten bis hin zu einem AUTOSAR-konformen Ethernet-Design durch einen integrierten Workflow geführt. Sollen neben Ethernet auch andere Bustechnologien wie CAN, LIN oder FlexRay eingesetzt werden, können auch gemischte Topologien entworfen werden. PREEvision begleitet Systemdesigner so bei der anspruchsvollen Aufgabe, klassisches Embedded Design mit moderner Service-Orientierung und der nötigen Back-End-Kommunikation zu verbinden – und unterstützt so die Transformation des Automobils zum fahrenden Rechenzentrum. *eck*

Referenz

[1] Gesetz von Conway: de.wikipedia.org/wiki/Gesetz_von_Conway (9. Januar 2017)

Dipl.-Ing. (FH) Markus Helmling
ist seit 2012 bei Vector Informatik als Product Management Engineer für PREEvision zuständig. Sein Schwerpunkt ist das AUTOSAR-Kommunikationsdesign.