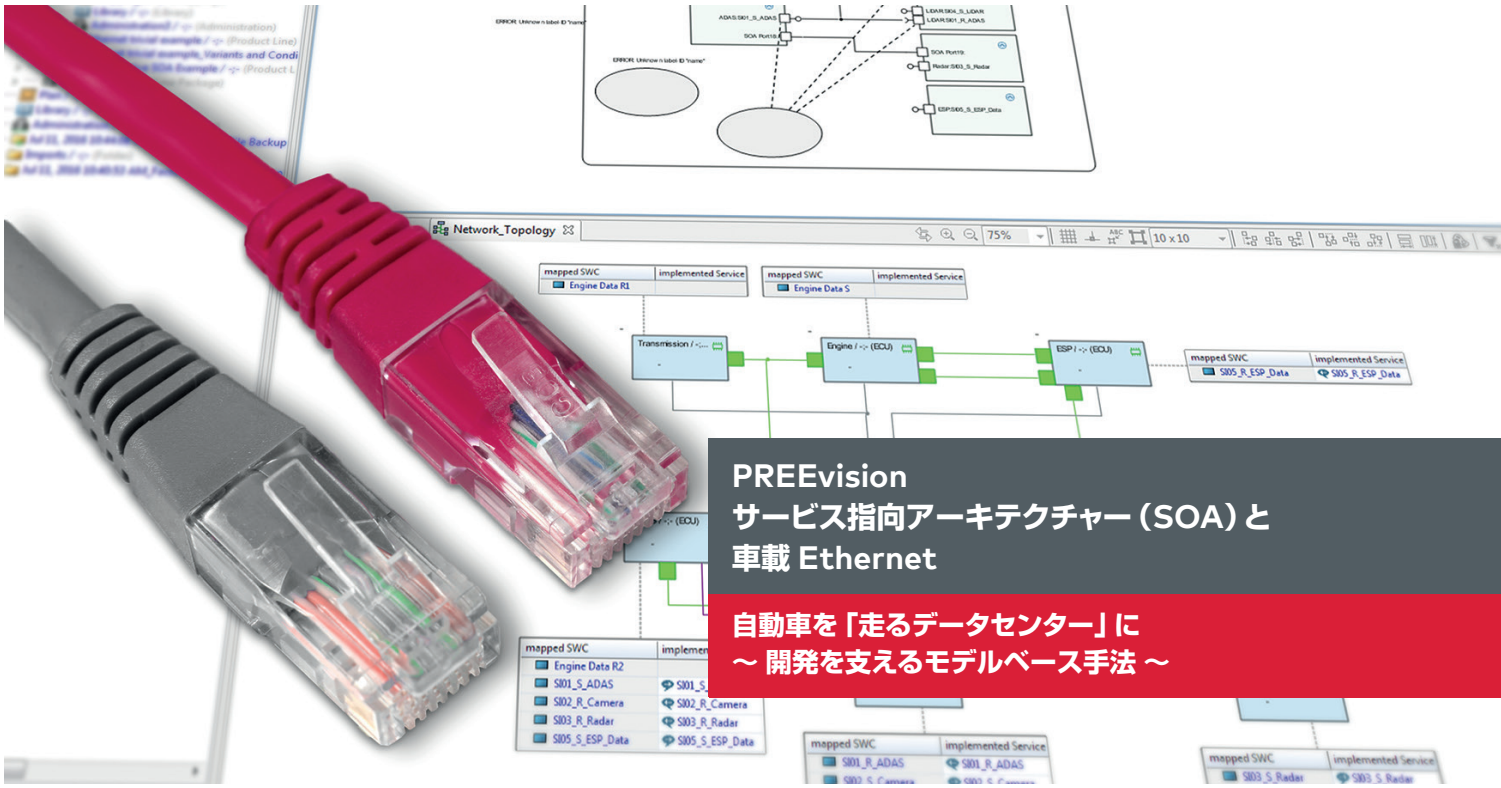
VECTOR



PREEvision
サービス指向アーキテクチャー（SOA）と車載 Ethernet

自動車を「走るデータセンター」に
～ 開発を支えるモデルベース手法 ～

サービス指向アーキテクチャー（SOA: Service-Oriented Architecture）は、長らく IT 業界で分散システムの記述と構造化に使用されてきました。近年、このサービス指向の設計は自動車業界でも急速に存在感を増しており、「自動車にも、スマートフォンで使えるような一連の機能を導入したい」という声の高まりに応えるためには欠かせない手法となっています。しかし、自動車業界における SOA は、モデル保守から自動運転や V2X 通信などのテクノロジー導入に至るまでの、新たなる要求にも対応することが求められています。

今日の自動車用ソフトウェア開発は多くの課題に直面しています。車作りが長年にわたって複雑化してきたことに加え、SOA（「ソア」もしくは「ソーア」と発音）が登場したことで、まったく新しい開発パラダイムとプロセスが生まれつつあります。コンウェイ[1]は 1968 年にすでに次のように主張していました。

> 「システム設計は（中略）設計する組織のコミュニケーション構造と同じ形になるよう制約される」[1]

この言葉を見ると、今日の自動車のソフトウェアアーキテクチャーの大半が構造的にサイロ化し、パワートレイン、ボディー、シャーシなどのドメインに基づいて伝統的な組織構造が再現されていることもうなずけます。しかし、SOA の効果的な導入には、これとは全く異なるドメイン横断的な調整とコミュニケーションが必要であり、責任範囲は従来の部門や組織の枠内に留まりません。つまり、SOA とそのインターフェイスを決定し、開発を進めていくには、全社的な協力と連携が絶対的な前提条件となります。このような組織レベルでの変更は、人事面にも影響します。社内に新しい役割や職務要求が生じ、そのためのスペシャリストの養成や採用が必要となります。

技術面では、サービスベース通信はオンボードネットワークの帯域幅に対する需要急増の傾向を強めます。これを受け、IT 業界で実績のある Ethernet 技術の採用を進める自動車メーカーがさらに増えています。Ethernet 技術は車両内だけでなく、車両と周辺環境あるいはインターネットとの通信やアプリケーションに必要な適切な帯域幅を実装することができます。将来も引き続き使用できる電気／電子（E/E）アーキテクチャーを開発するためには、これらの要求を考慮したツールでシステム設計と保守を行うことが必要です。

サービス指向のメリット

SOA では、構築されるサービス環境の特性は、明確に定義されたインターフェイスにより決定されます。理想を言えば、これらのインターフェイスは構文的にも意味論的にも一定の形式で一義的に記述されることが望ましく、明確な定義が行われることでサービスの構造化が可能になります。たとえば階層型のアーキテクチャーが、サービスの依存関係を理解しやすくし設計規則を明確に表現するため、一般的に使用されています。また SOA で

は、個々のコンポーネント同士はサービスバスをミドルウェアとして相互に疎結合されています。このミドルウェアはサービスプロバイダー（Service Provider/サービスの提供者）とサービスコンシューマー（Service Consumer/サービスの利用者）の仲介役として機能し、システム始動時にサービスプロバイダーとサービスコンシューマーの間の通信を制御するほか、通常はデータのシリアル化も定義して、コントローラーのベーシックソフトウェア（BSW: Basic Software）やオペレーティングシステム（OS: Operating System）によって管理されます。（図 1）。

「シリアライザー」と呼ばれるこの演算規則は、物理的なバスシステムでのデータ転送を規定します。シリアライザーは、データをどのようにシリアルビットストリームに変換するか、そしてそれを受信側でどのように逆シリアル化するかを決定します。

サービスプロバイダーとサービスコンシューマーの間の接続は、従来はシステム設計時に作成されていましたが、ミドルウェアを使用することで実行時に作成されるようになり、それによって従来のソフトウェア開発に伴うさまざまな問題への対処が可能になります。たとえば、システムの部分的なアップデートが容易になります。つまり後方互換性が確保されていれば、サービスプロバイダーに変更を加えたとしても、そのサービスのすべてのコンシューマーをそれに合わせて変更する必要はありません。サービスインターフェイスのバージョン管理とリビジョン制御を使用すれば、「ソフト」な移行のシナリオも可能です。すなわち、システムのある部分ではサービスインターフェイスの新機能を先行して使用しつつ、その他の部分では古いバージョンのサービスを引き続き使用できます。さらに、設計にフェイルセーフ機能を組み込むことも可能で、いずれかのサービスインスタンスに障害が発生しても、それと同等あるいは少なくとも互換性のあるサービスインスタンスを代わって動作させることができます。つまり、分散システムでは、サービスに関わるすべての電子制御ユニット（ECU: Electronic Control Unit）を更新しなくても済むということです。

もう一つ忘れてはならないのは、整合性のあるサービス設計が高機能な設計パターンの実装も可能にするということです。たとえば、リモートプロシージャーコール（RPC: Remote Procedure Call）はサービスベースのアーキテクチャーではよく用いられる手法ですが、かつては ECU 内でしか使われていませんでした。しかし、オンボードネットワークでもクライアント／サーバーのパラダイムの使用が可能となると、これまで車両の通信設計の主力であったシグナル指向通信が使われるケースが減っていきます。

SOA 手法 – トップダウンのアプローチ

自動車メーカーやサプライヤーが SOA のメリットを実際に活用したいのであれば、新しいシステムの設計に必要なあらゆる開発ステップに対応できる適切な手法が必要です。そして、品質が保証された設計を実現すると同時に、その複雑性をも管理することが目標となります。以下に、要求からサービスアーキテクチャー、通信設計に至る開発プロセスの手法を説明します。

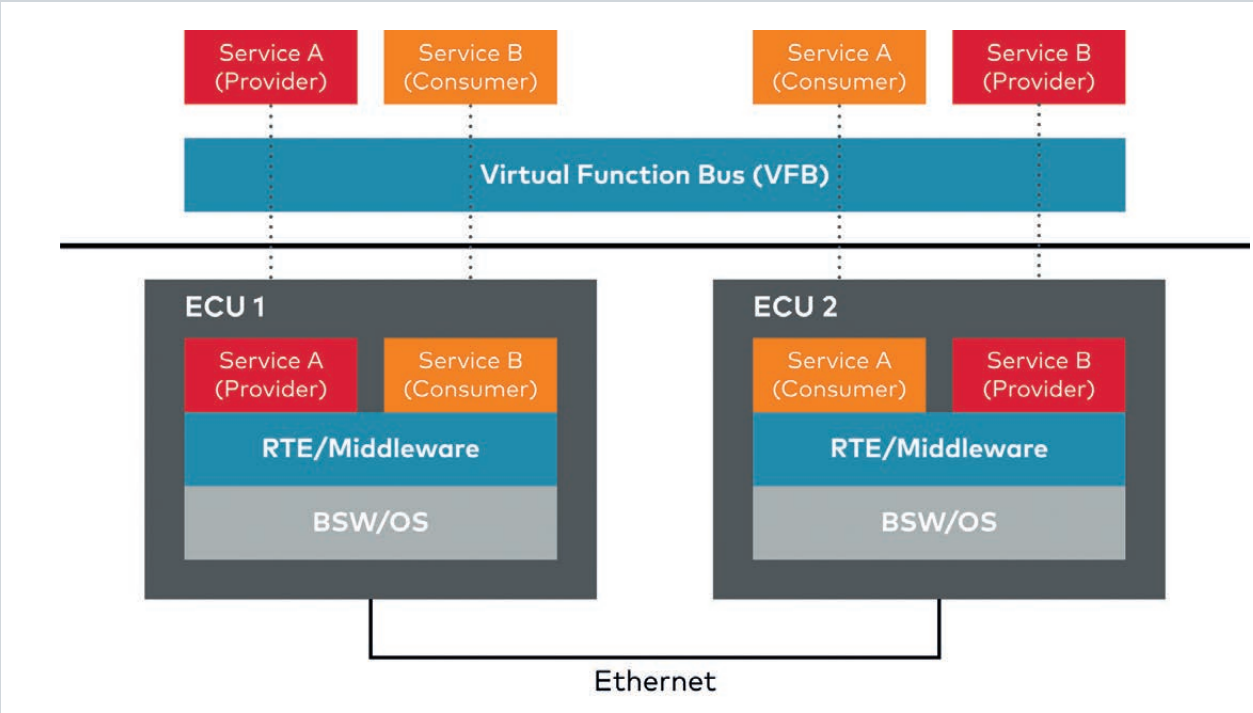


図 1：サービスとミドルウェアの概略図

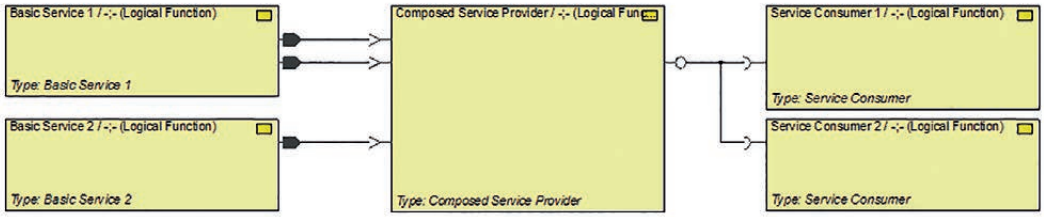


図 2 : サービスのオーケストレーション（構築）を静的に記述した論理アーキテクチャー

ここでの目標は、AUTOSAR Classic に対応した車両または車両サブシステム用のシステム記述を作成することです。ただしこの手法は AUTOSAR Adaptive のコンテキストあるいは AUTOSAR 以外のシステムにも応用できます。

開発プロセスは新機能のアイデアから始まり、次にそれをフィーチャー（利用者から見たシステムの機能・特徴）の形に詳細化します。最近では、これはユースケースに従って構造化されるのが普通で、標準規格である統一モデリング言語（UML: Unified Modelling Language）に基づくユースケースダイアグラムの記述法が有効であるとの定評があります。ユースケースにより、フィーチャーのカタログからサービスを導出できます。また、サービスを構造化し、依存関係を描出するにはさまざまなビューが必要です。

すでに定着している、わかりやすいビューの 1 つが、機能の依存関係を表す情報フロー指向のビューです。ベクターのシステムツール「PREEvision」には、このビューのためのいわゆる「論理アーキテクチャー図」が用意されています。この図では、ブロックダイアグラムの形で、（ハードウェアやソフトウェアによる実装に依存しない）サービス機能と、機能間の接続関係（イベントチェーン）を表現します（図 2）。

さらに、サービスアーキテクチャーの記述に特化した UML プロファイルの「SoaML」では、「パーティシパント（Participant/参加者）」と呼ばれる論理ノードが定義されています。パーティシパントは、サービスのプロバイダーとしてもコンシューマーとしても動作し、サービスの交換はいずれの場合も「SOA ポート（SOA port）」と呼ばれるポートを介して行われます。これらによってサービス間の依存関係を可視化できます。たとえば複数の（基本）サービスのコンシューマーであるパーティシパントが、それらを組合わせて実現する高品質のサービスのプロバイダーとなるようなケースを可視化します。この設計ステップは「オーケストレーション（Orchestration/構築）」と呼ばれ、制御機能に広く使用されるイベントチェーンのビューに似ています。

ただし、完全に動的なシステムの場合、システム内には 1 つのサービスの複数のインスタンスが異なるバージョンで存在しており、前述のようなパーティシパントへのサービスの配置は、実行時になってはじめて確定します。自動車システムでは今のところ、サービスは特定のハードウェアに静的に配置されるのが主流であるため、これを ECU にサービスのロール（プロバイダー／コンシューマー）を割り当てたものと解釈することもできます。それとは別に、パーティシパントを使用すれば、実行時に生成されるさまざまな構成を可視化することも可能です。このように、論理サービスノードをパーティシパントとして表現することは、解析ツールとしても有効だといえます。なお、こうしたシステムをテストおよび保護するには、これらの構成を把握し、文書化しておくことが重要な基盤となることを付け加えておきます（図 3）。

上で述べた SOA ポートのタイプは、複数の側面を明確に記述したサービスインターフェイスによって決まります。まず明記する必要があるのはそのポートが果たすロール、すなわち、それがサービスをプロバイド（提供）するのか、それともコンシューム（消費）するのかです。ここではインターフェイスの特性を構文的に記述しますが、その際は利用可能なメソッド、フィールド、イベントなどの、一般的に有効な、実装テクノロジーに依存しない属性だけでなく、メソッドのパラメーターのように、ミドルウェアがサポートし、この定義の時点で既知であるデータ型も記述します。これにより、サービスの静的な振る舞いを完全に記述できます。

SOA ポート間のサービスコントラクトのセマンティクスは、多くの場合はテキスト形式で記述します。特に、サービスのロール間の時間シーケンスを指定し、プロトコルを表現します。この設計ステップは「サービスコレオグラフィー」とも呼ばれます。シーケンスの視覚化には一般に、標準規格の UML で提案されているコラボレーション図またはメッセージシーケンス図（MSC: Message Sequence Chart）が使用されます。

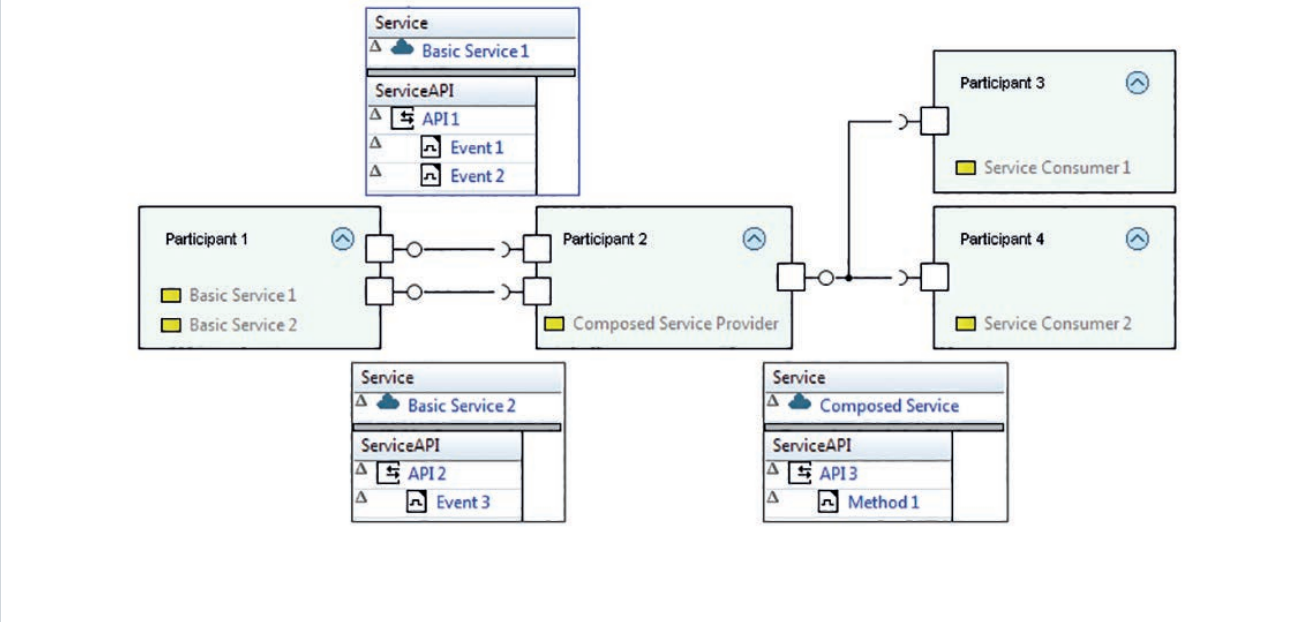


図 3 : 論理サービスノード（パーティシパント）を定義するための「PREEvision」の SOA ダイアグラム

実装テクノロジーに依存しない、形式的なサービスの記述は、通常はインターフェイス記述（IDL: Interface Description Language、インターフェイス記述言語）により定義・情報交換されます。PREEvision のような最新のシステム設計ツールでは、この記述に基づき実装の外部構造を導出できます。具体的には、この記述に基づき AUTOSAR Classic のアプリケーションソフトウェアコンポーネント（Application SW-C）による実装の外部構造を導出できます。

これにより実装の本体部分が作成されますが、これは次の設計ステップ、つまりさらに詳細な設計と、詳細設計に基づく実装のベースとして必要なものです。さらに、サービスのロールとサービスインターフェイスの記述も、AUTOSAR Classic による実装に引き継がれます。このアプローチには整合性があり、サービスのロールやサービスインターフェイスの記述を変更しても、同期機能によりその変更をテクノロジー固有の実装に自動的に反映できるというメリットがあります。その結果、IDL 記述からソフトウェア設計インターフェイスを導出し、それらを AUTOSAR Classic プラットフォーム上でセンダー／レシーバー、あるいはクライアント／サーバーのインターフェイスとして表現することが可能になります。さらに適切なファイル型（例：ARXML バージョン）を指定することで、導出された設計情報を外部ファイルにエクスポートすることもできます。

現在、AUTOSAR Adaptive プラットフォームに特別な「サービスインターフェイス」を導入することが検討されています。これは先に述べたような属性をそのまま含み、実装テクノロジーに依存しないサービス記述を最大限に活用できるようにするのが狙いで、最終的には各種の実装テクノロジーに基づく外部構造を、同じサービス定義から、ボタン 1 つで導出できるようにすることを目標としています。このステップの後には、起動時のミドルウェア固有の振る舞いの規定と、そのパラメーター化が必要です。このプロセスを「サービスディスカバリー」と呼びます。

通信設計

これ以降の手順では、データ供給と通信記述の構造は、使用される通信バスのテクノロジーとミドルウェアに依存します。AUTOSAR では、サービスディスカバリーとデータシリアル化のための変換機能として、Service Oriented Middleware over Internet Protocol (SOME/IP) がデフォルトで使用されます。

ただし、AUTOSAR は他の変換機能やミドルウェアの使用にも対応しています。通信チャンネルに Ethernet と Internet Protocol (IP) を使用する場合は、IP アドレス、トランスポートプロトコル、ポートを指定してソケットを定義する作業が不可欠です。また、AUTOSAR Adaptive ではソケットアドレスを記述するだけで十分ですが、AUTOSAR Classic のベーシックソフトウェアのスタックは FlexRay あるいはもっと広く使用されている Controller Area Network (CAN) バス、Local Interconnect Network (LIN) サブバスなどを使用するシグナルベース通信用に設計されているため、AUTOSAR Classic の場合はシグナルレベルも指定しなければなりません。統合型のシ

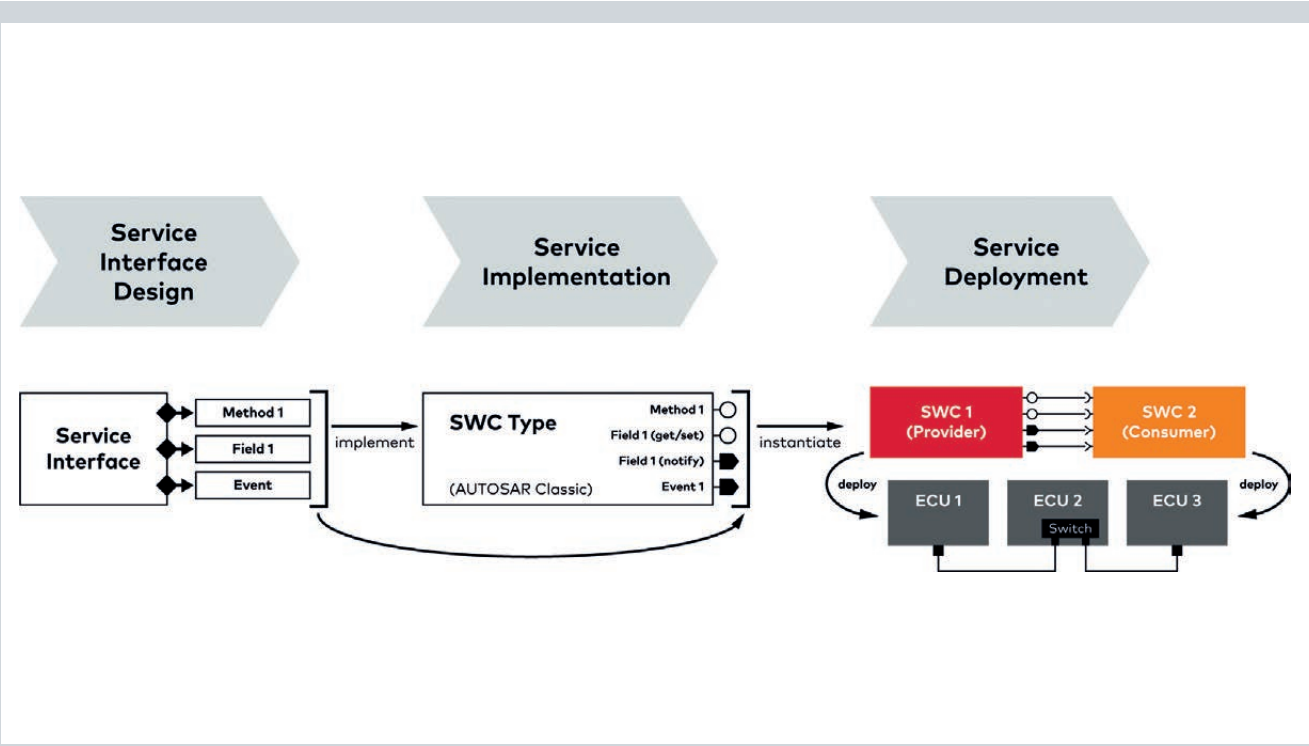


図 4 : サービス設計から通信設計まで

ステム設計ツールはこの点に圧倒的な強みを持っており、以前のステップの作業成果物を基にしてシグナル通信を導出することができます。これにはサービス記述と、システム内のどこでサービスが提供されるのかについての情報、すなわち、どの ECU がプロバイダーとなり、どの ECU がコンシューマーとなるのかの情報が必要ですが、PREEvision はこうした際に、サービスやメソッド ID といったさらに詳しい情報もサービス記述から抽出し、通信記述に転送します（図 4）。

まとめ

ベクターのモデルベース E/E 開発ツール「PREEvision」は、方法論に基づいた整合性あるサービス指向アーキテクチャーの設計をサポートします。ユーザーはサービスインターフェイスの定義からサービスのインタラクションの指定、そして AUTOSAR に対応した Ethernet 設計に至るまで、すべてを統合されたワークフローに沿って進めていくことができます。Ethernet に加えて CAN、LIN、FlexRay などの他のバステクノロジーも使用する場合は、混合型トポロジーも設計できます。システム設計者は PREEvision を使用することで、従来の組込設計と最新のサービス指向を用いたバックエンド通信とを組み合わせるという困難な課題を解決できます。このように PREEvision は、自動車を「走るデータセンター」へと変貌させることを強力にサポートします。

参考文献 :

[1] Conway's law: http://en.wikipedia.org/wiki/Conway%27s_law (January 19, 2017)

執筆者 :

Markus Helmling (Dipl.-Ing. FH)
Vector Informatik GmbH にて PREEvision プロダクトマネージメントエンジニアとして勤務。

本稿は、ドイツで発行された『Elektronik automotive, magazine Special issue, Automotive Ethernet – March 2017』に掲載された記事内容を和訳したものです。

画像提供元 :
見出し画像および図 1 ～ 4 : Vector Informatik GmbH

■ 本件に関するお問い合わせ先
ベクター・ジャパン株式会社 営業部
（東京） TEL: 03-5769-6980 FAX: 03-5769-6975
（名古屋）TEL: 052-238-5020 FAX: 052-238-5077
E-Mail: sales@jp.vector.com