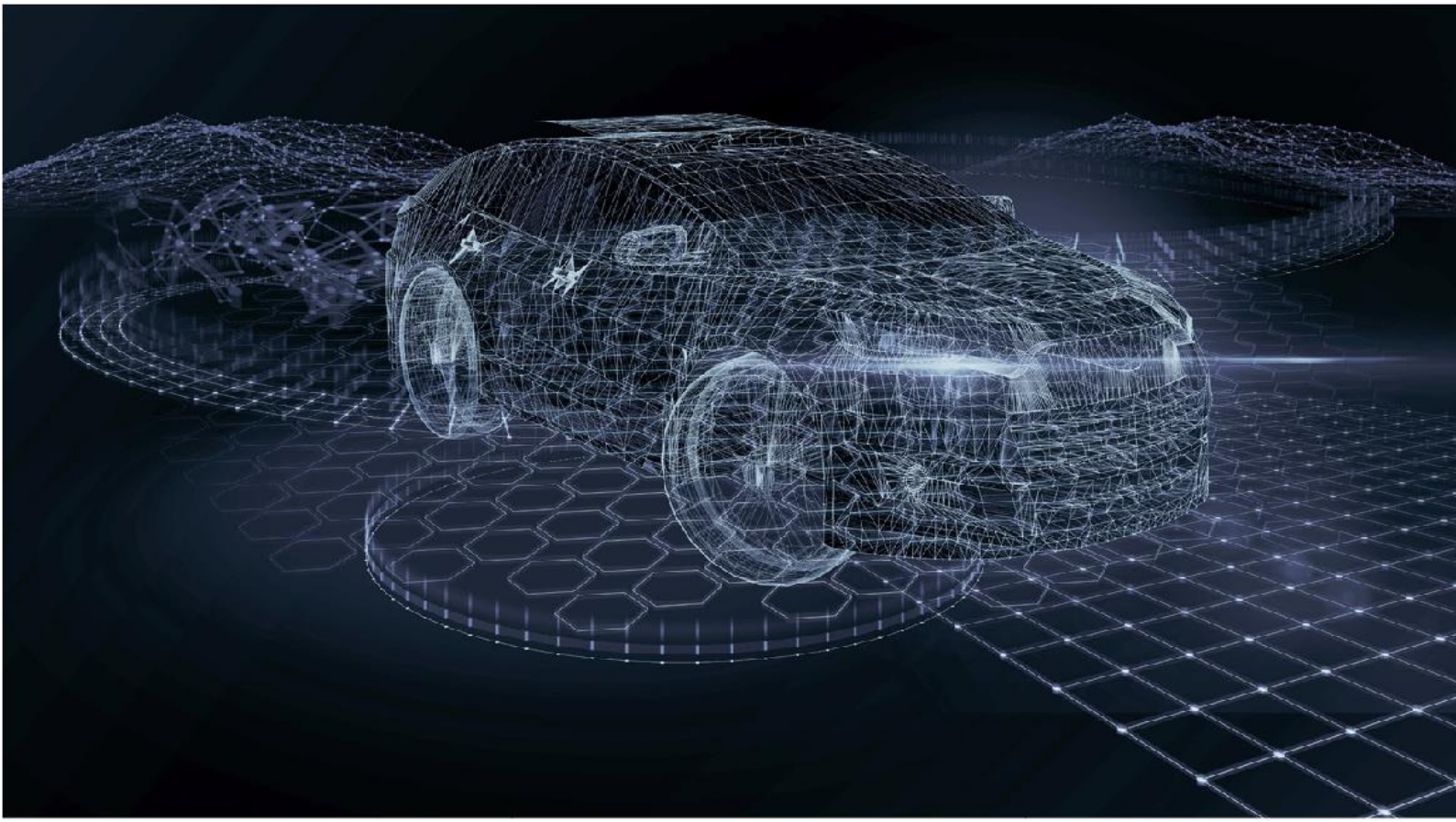
McKinsey & Company

McKinsey Center for Future Mobility

ソフトウエアを制する者が勝者となる: 自動車産業における車載ソフトウエア開発の最適化に向けて

車載ソフトウエア開発の重要性がかつてないほど高まっている中で、自動車メーカーと自動車部品サプライヤ企業どちらにおいても、この潮流を主導できている企業は限定的である。特に日本企業には、従来、自動車メーカーとサプライヤが一体となり、ともに阿吽の呼吸でハードウエア中心の開発を実施してきた歴史的な背景がある。また市場構造の特徴として、ソフトウエア開発を大手の開発ベンダーやSIerに大きく依存している。さらに、海外拠点やオフショアベンダーの管理における言語的・文化的ハードルが高いなど、日本企業独自の課題も存在する。本白書では国や企業の枠にとらわれず、自動車の車載ソフトウエアの開発に関する全体的な傾向と目指すべき方向性を網羅的に論じている。本白書が、日本企業で車載ソフトの開発に携わる方々にとって、歴史的・文化的な背景や自社の課題・特徴を鑑みながら個々の打ち手や全社的な改革を検討する際の一助となれば幸いである。

オンドレ・バーカキ、ヨハン・ダイヒマン、ステファン・フランク、ドミニク・ヘップ、小田原 浩、アンドレ・ロシャ、松原 寛、木下 暢、小泉 正剛、三宅 匠 著



2021年9月

自動車産業における車載ソフトウエア開発の重要性はかつてない勢いで高まっている。近年、自動車業界では、自動運転、コネクティビティ、電動化およびシェアドモビリティ(ACES)、という4つの**非常**に大きな**破壊的変化**が生じているが、いずれも最先端のソフトウエアによって引き起こされている。しかし、これらの破壊的変化はまだ序の口に過ぎない。ソフトウエアが中核的な役割を果たす新たなバリューチェーンでは、自動車メーカー、サプライヤに加え、他業種からの新規参入企業が、有利なポジションを獲得しようとしのぎを削っている。

ソフトウエア: 自動車産業における重要な転換点

自動車産業を取り巻く環境が急変する中で、十分なソフトウエアの開発能力を持たない自動車メーカーは量産開始の遅れや予算超過など、より大きなリスクに直面することになる。また、従来よりもはるかに革新的な製品をより速いスピードで打ち出してくる他業種からの新規参入者に後れを取るリスクも無視できない。加えて、ソフトウエアに問題が生じた場合は大規模なリコールの対象となることも想定され、さらに、ソフトウエアの脆弱性がハッキングの対象となれば、顧客の安全にも悪影響を及ぼしかねない。

マッキンゼーの調査結果によると、ソフトウエアに強みを有する企業とそうでない企業の乖離は広がっており、トップ企業の処理能力やアウトプットの品質は、下位企業の3〜6倍にまで達していることが明らかとなっている[1]。これは、ハードウエアによって生じるメーカー間の生産性の差を大きく上回っている。多くの自動車メーカーは、ソフトウエア開発に強みを持つことによるメリットを十分に認識しており、こうした開発能力を構築し業績を伸ばすために大胆な施策を打ち出している。例えば、ソフトウエア開発能力を強化するために、今後数年間で数千人に上るソフトウエアエンジニアを採用すべく動いている企業もあれば、企業のガバナンスモデルを見直し、ソフトウエア会社と提携することで、グローバルな開発拠点の構築を狙っている企業もある。

マッキンゼーの調査結果によると、ソフトウエアを大きな破壊要因と見なしている研究開発部門のトップのうち、オペレーション上必要な変革を行う準備ができていると感じている人は、わずか40%に過ぎない

[1] マッキンゼー独自のSoftCosterデータベースによる。本データベースには、様々な業種における1万4,000件を超えるソフトウエア開発プロジェクトに関するデータが格納されている。

しかしながら、我々はこうした打ち手だけでは不十分であると考える。なぜならば、真の変革は、自動車メーカーが、その運営基盤であるオペレーティングモデルをソフトウエア開発という新たな目的に合わせてアップデートしてこそ実現されるものであると考えるからである。マッキンゼーの調査結果によると、ソフトウエアを大きな破壊要因と見なしている研究開発部門のトップのうち、オペレーション上必要な変革を行う準備ができていると感じている人は、わずか40%に過ぎない[2]。ソフトウエア開発における先行企業は、業種を問わず、既にそのエンジニアリング業務に対し大胆な改善を実施している一方で、自動車メーカーは、こうした先行企業に大きく後れを取っている。特に、アジャイル開発、継続的インテグレーション(CI)およびテストの自動化が主な懸念材料となる。

こうした背景から、自動車メーカーは、ソフトウエア開発に対するアプローチを、基盤となるオペレーションモデルも含め抜本的に見直す必要がある。本白書では、自動車メーカーやサプライヤ、それ以外の自動車関連企業と緊密に協働した経験から得たマッキンゼーの知見を共有させていただく。

数字で見るソフトウエアの重要性の高まり

車載ソフトウエアの重要性の高まりを示す、いくつかのトレンドがある。その一つとして、ソフトウエアおよびエレクトロニクス市場の急速な拡大が挙げられる。この市場は、2019年から2030年にかけて年率9%で成長することが見込まれており、これは自動車自体の売上げの伸び率の2倍以上となる。その中でも特に高い成長が見込まれる領域は、ソフトウエア機能開発(年率11%)と統合テスト(年率12%)である[3]。

ソフトウエアの複雑性は増しているが、生産性は伸び悩む

車載機器の組み込みソフトウエアの複雑性は機能およびアーキテクチャの両面において増しているにもかかわらず、開発の生産性は同様のペースでは上昇していない。

マッキンゼーの調査結果によると、ソフトウエアの複雑性は過去10年で4倍となっているが、ソフトウエア開発の生産性の上昇は1.0〜1.5倍にとどまっている(図表1)。

この問題は、特にインフォテイメントやADAS(先進運転支援システム)など、複雑化している大型のモジュールで深刻となっている。これらのモジュールの生産性は、従来搭載されているソフトウエアと比較して25%〜35%下回っている。

こうしたギャップの拡大は、今後の自動車産業の発展の足枷になりかねない。また、ソフトウエア開発とそのメンテナンスに費やす労力が今後さらに増加すれば、自動車産業が本来注力すべき技術革新や競合他社への対応に十分なリソースを割くことができなくなる。

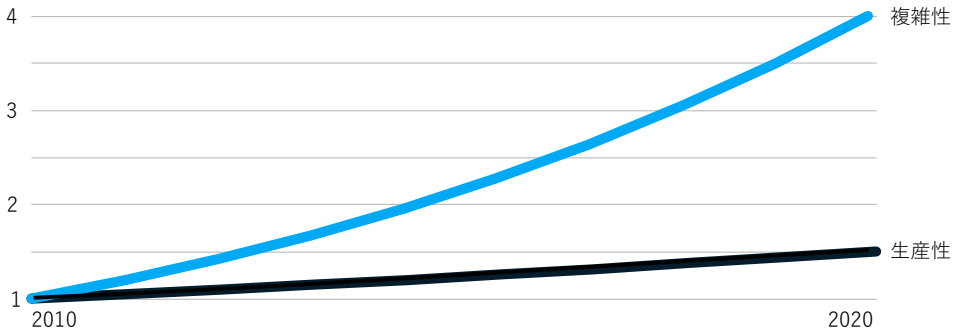
[2] McKinsey R&D of the Future Surveyによる

[3] Ondrej Burkacky, Johannes Deichmann, Jan Paul Stein, *Automotive software and electronics 2030:Mapping the sector's future landscape*(2019年7月9日), McKinsey.com (近日アップデート版が公表される予定)

図表 1

ソフトウエアの複雑性は生産性の伸びを上回るペースで増大している

ソフトウエアの複雑性および生産性の上昇の相対的推移、自動車産業における特徴を踏まえて指数化



資料: マッキンゼー独自のSoftCosterデータベース

我々が企業経営者を対象に行ったインタビューでは、「生産性が変わらないままソフトウエアの複雑性が増し続けると、ソフトウエアのメンテナンスだけで既存のソフトウエア関連の研究開発のリソースは手一杯となってしまい、技術革新に回す余力がなくなってしまう」という意見も聞かれた。こうしたギャップの拡大は、最終的にはコスト競争力を低下させ、企業の財務と風評の双方に深刻な問題をもたらしかねない。

また注目すべき点として、ソフトウエア開発に強みを持つ上位25%の企業は、下位の企業と比較して、生産性は3.0倍、処理能力は3.5倍、アウトプットの品質水準は6.0倍高くなっている(図表2)。その結果、ソフトウエアに強みを持つ上位25%の企業では、製品の市場投入までの期間が短縮され、新しいソフトウエア機能のレベルごとに開発コストが低く抑えられている。

ハードウエアに関しては、上位企業と下位企業のパフォーマンスの差はさほど顕著ではないため、ここで差別化を図るのは難しい。

複雑性を低減しつつ効率性を高める

急速に変化する環境に適応していくためには、企業は、ソフトウエアの開発およびメンテナンスの労力を低く抑え、複雑性を最小限にとどめる必要がある。これを実現するためには、各種プラットフォームや製品のライフサイクルを横断して展開しているソフトウエアのバージョン数を減らす必要がある。同時に、企業は、多くの製品においてコンポーネントの再利用・共用化を進めなければならない。また、生産性を向上させるためには、効率性の改善に取り組み、ソフトウエアの開発スピードをデジタルネイティブ企業と同程度にまで高める必要がある。ソフトウエアにおけるイノベーションは今後も引き続き活発であることが予想されるため、企業は、ソフトウエア

の開発やメンテナンスにおける生産性を高め、市場で生き残るために必要とされる製品を提供していかなければならない。

複雑性を低減し効率性を高めるためには、以下の4つの重要な側面に焦点を当てた新たなソフトウエアオペレーションモデルが必要となる。

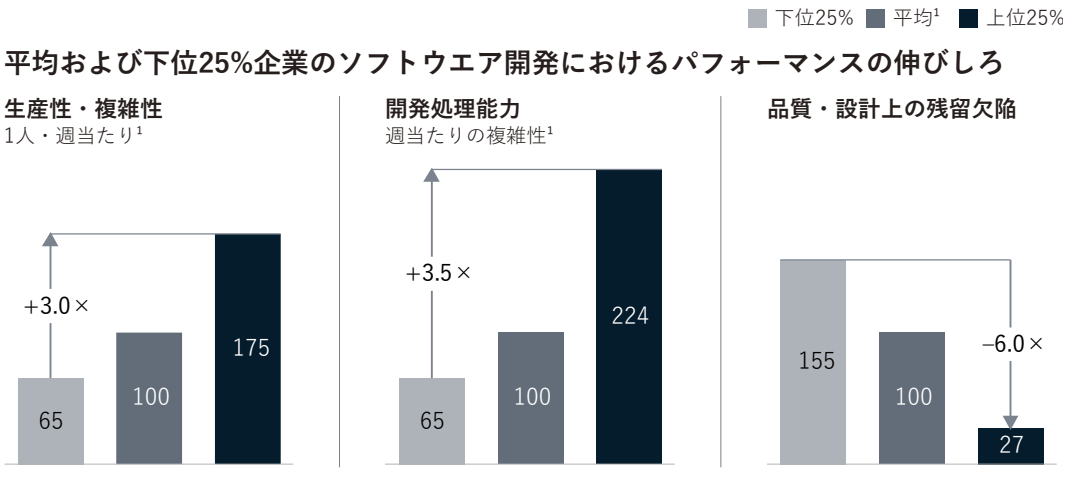
- **A. What:** アーキテクチャ、設計、要件を含め、どのようなソフトウエアを開発するか。
- **B. Where:** 拠点、人材、必要なパートナーシップを含め、組織内のどこでソフトウエアを開発するか。
- **C. How to develop:** アジャイルなどの開発手法やテストプロセスの変更などを含め、ソフトウエアをどのようにして開発するか。
- **D. How to enable:** パフォーマンスマネジメント手法やツールチェーンインフラを含め、ソフトウエア開発をどのようにして進めていくか。

まず、最初の側面である「A. what:どのようなソフトウエアを開発するか」では、複雑性を低減するため、モジュール化を前提としたアーキテクチャ、ハードウエアとソフトウエアの分離(デカップリング)、ユーザーフレンドリーな設計および要件管理の導入などを検討する。

残りの3つの側面(B、C、D)では、適切な枠組みやプロセス、インフラを提供することで、ソフトウエア開発の効率性を高めることに焦点を当てる。マッキンゼーは、自動車メーカーがソフトウエアに関する課題を克服するうえで役立つ、これらの4つの側面における11のベストプラクティスを特定した(図表3)。これら4つの側面の変革には同時に取り組むことが望ましい。

図表 2

ソフトウエアに強みを持つ上位25%の企業は、高い生産性、処理能力、品質水準を誇る品質水準を誇る

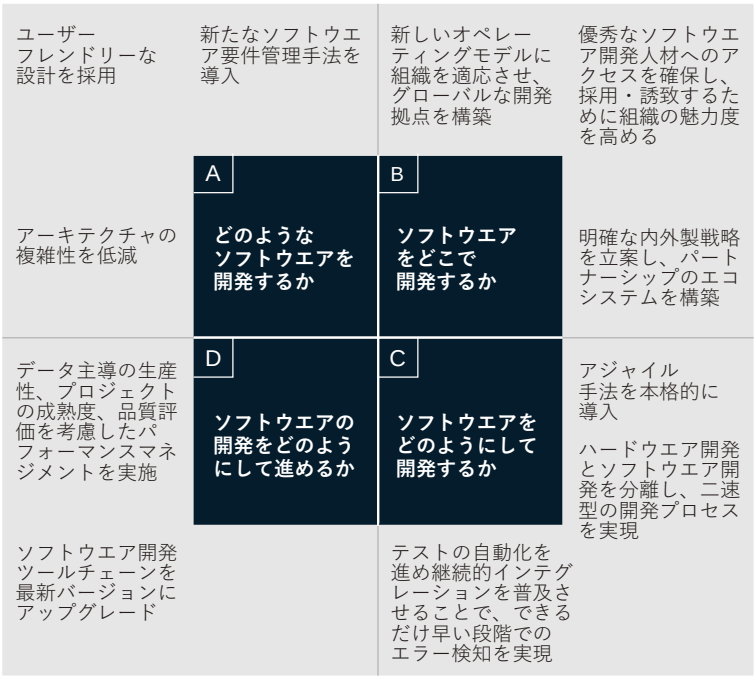


[1] 平均値を100として指数化。1人・週当たりの複雑性は1FTEの生産性を示す。週当たりの複雑性は全組織の生産性を示す
資料: マッキンゼー独自のSoftCosterデータベース

図表 3

ソフトウエア開発の新しいオペレーティングモデルでは、4つの重要な側面において変革を行う必要がある

各側面におけるベストプラクティス



資料：マッキンゼー

A. What: どのようなソフトウエアを開発するか: アーキテクチャ、設計および要件

新しいオペレーティングモデルでは、企業は、製品レベル、機能レベル、モジュールレベルのそれぞれにおいて、ソフトウエアで実現しようとしている狙いやビジネス機会を、実現可能なアーキテクチャや製品および要件に落とし込む必要がある。このプロセスを通じて、企業は、自社にとって価値を生み出すソフトウエアの種類を詳細に把握することができる。さらに、アーキテクチャの複雑性を低減したり、ユーザーフレンドリーな設計技術を採用したり、ソフトウエア要件の管理をしやすくすることもできる。

A1. アーキテクチャの複雑性を低減

マッキンゼーの調査結果によると、車載ソフトウエアは、モジュール構成が不十分であるため設計が複雑になっており、結果としてプロジェクト全体の工数を増やす原因となっている。加えて、車載ソフトウエアにおいては、アーキテクチャとコンポーネントの境界が明確でないことが多く、その結果、新たな機能を追加する際には膨大な数のコンポーネントを変更しなければならなくなる。こうした相互依存性があるため、ソフトウエアに不具合

が生じた際には、どのモジュールが原因で、どの開発チームが担当したのかを追跡するために、多くの時間と高度な専門知識が必要となる。

これらの課題に対処するためには、企業は、標準化とモジュール化を徹底的に進め、これらをプラットフォームを横断して展開し、ソフトウエアの複雑性を低減していく必要がある。

また、企業は、ソフトウエアをハードウエアから切り離すことに注力し、サービス重視の設計を採用していくことが重要となる。

アーキテクチャの分離(デカップリング): 企業は、強力なミドルウェア層を導入することで、ハードウエア機能を抽象化し、上層部で使用されている標準APIを介してソフトウエアの機能やサービスを利用できるようになる(図表4)。このソフトウエアアーキテクチャでは、プラットフォーム間の共用性を高め設計の複雑性を低減することができるため、同じソフトウエアを何度も繰り返し開発する必要がなくなる。

企業は、ハードウエアとソフトウエアの分離に加え、オペレーティングシステムの標準化を進め、異なるコンポーネント間の互換性を確保していく必要もある。既に多くの企業がこのようなオペレーティングシステムの開発に着手したことを公表しているが、現時点では、すべてを網羅した手法はまだ確立されていない。

また、各社とも、このようなオペレーティングシステムのどこに注力し、どのような機能を持つ必要があるのかを明確に定義できていない。

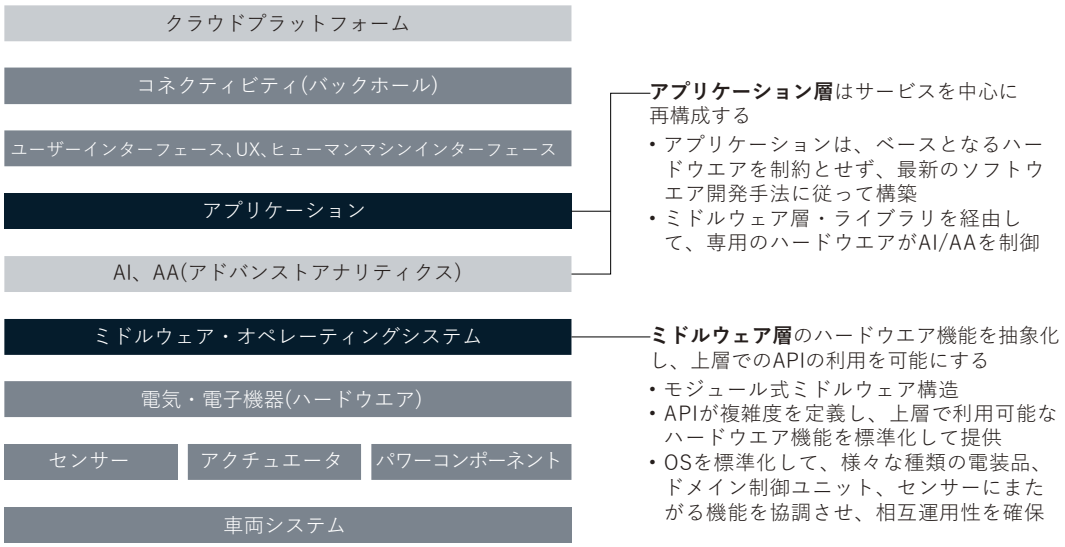
企業は、明確なアーキテクチャ上の原則やガイドラインに従うことで、システムやソフトウエアの複雑性に効果的に対処することができる。ハードウエアとソフトウエアの分離を進めていくことで、様々なメンバーがモジュール開発に参加することができる。また、ソフトウエア開発のモジュール化を進めることで、コードの再利用が促進され、共用性が高まることで、全体として必要なコードの量を減らすことが可能になる。多くの企業が、ソフトウエアプロダクトライン開発(SPLE)を導入し始めており、同種・同系列のソフトウエア製品を効率的に生産するためにソフトウエア資産の共用・再利用の拡大を進めている。このアプローチにより、1つのソフトウエアで、複数の製品、製品バリエーション、製品群に対応することが可能となり、異なるハードウエア上でも機能させることができる。SPLEを用いることで、複数の製品についても開発やテストは1回実施すればよいことから、トータルで必要な労力を大幅に削減することができる。

別の言い方をすると、ハードウエアをソフトウエアから切り離すことで、ハードウエアの「民主化」を進めることができる。ハードウエアは、基本的な演算、メモリ、入出力、電源供給機能を提供する一方で、ソフトウエアは、エンドユーザーの機能を定義する。仮想化やコンテナを使用すれば、同一のハードウエア上で複数のソフトウエア機能を利用することができ、これを、必要に応じて(元のハードウエアの故障時などに)他のハードウエアに動的に分散させることもできる。一方、ADAS(先進運転支援システム)のようにリアルタイムで稼働させる必要のあるアプリケーションについては、効率性を維持するために

も、特定のハードウエアに特化したソフトウエア開発が必要になる。

図表 4

企業は、ハードウエアおよびソフトウエアを分離し、強力なミドルウェア層を有するアーキテクチャの構築を目指すべきである



資料：マッキンゼー

サービス指向アーキテクチャ(SOA): アーキテクチャは、想定するサービスの定義に従って、ビジネスやユーザーのニーズを盛り込んだものでなければならない。サービス指向アーキテクチャを構築することにより、部門を超えて、中核となる要素やインターフェースを標準化することができる。また、企業は、ハードウエアやソフトウエアの個々の要素について設計の標準化を図り、従来の性能を損なうことなく、演算機能やストレージなどのリソースを、サポートされている他のデバイスや機能とともに拡張することもできる。自動車メーカーにとって、サービス指向アーキテクチャの構築は特に重要となる。

車両からクラウドシステムへの高速な接続を実現し、モデルの速やかなアップデート/アップグレードを可能にすることで、製品の長期的な価値を高めることができる。

A2. ユーザーフレンドリーな設計を採用

業界を問わず、ユーザーフレンドリーな設計を重視し、理想的なユーザーエクスペリエンス(UX)の実現に注力している企業は、競合を上回る収益をあげている[4]。ACESが普及し、

[4] McKinsey Design Index Scores 詳細については、 Benedict Sheppard, Hugo Sarrazin, Garen Kouyoumjian, and Fabricio Dore, "The business value of design," *McKinsey Quarterly*, 2018年10月25日, McKinsey.comをご参照

ソフトウエア・デファインド・ビークル(SDV)が一般的になるにつれ、これらの機能は、自動車メーカーが総合的な競争力を高めるうえでますます重要となってくる。自動車産業は、優れたソフトウエアのUXの設計や顧客価値の最大化という側面においては他産業に後れを取っており、トップ企業であっても油断はできない。

自動車メーカーは、設計のベストプラクティスに則り、新たなソフトウエアサービスにエンドユーザーからのフィードバックを反映させるというプロセスを、製品の市場投入の前後両方において高頻度で繰り返し実行し、改善していく必要がある。加えて、ソフトウエアのアップデートや新機能の追加を週単位もしくは月単位で行うデリバリーモデルを採用し、継続的な改善を可能にすることも重要となる。これらのモデルで想定されるサイクルは、従来のハードウエアの開発期間をベースとした数年のサイクルよりも大幅に短いものとなる。新たなデリバリーモデルを採用することによるもう一つのメリットは、企業が顧客と直接的な接点を持つことにより利用者から継続的にフィードバックを得ることができるため、要件を最適化し、優れたUXを提供することが可能になる点である。

このような顧客とのやり取りは、自動車メーカーがアップグレードを行うためにハードウエアに依存していた時代には実現不可能であった。この時代には、顧客との接点は、ディーラーネットワークを通じた営業活動や市場調査の際に限られていた。こうしたモデルを実現するためには、自動車メーカーは、ソフトウエアや電子機器のアーキテクチャをサポートするだけでなく、無線通信でのアップデートを可能にするツールチェーンを実装するなど、一定の前提条件を満たす必要がある。

自動車メーカーがUXの分野でリーダーを目指すのであれば、データを徹底的に活用する必要がある。車載ソフトウエアやセンサーの数も増えているため、自動車メーカーは自動車の利用に関して膨大な量の顧客情報にアクセスすることができるようになった。自動車メーカーは、これらのデータを活用して、顧客にとって最も重要な機能を特定すると同時に、オーバースペックの機能や全く使用されていない機能を把握できるようになった。こうした知見は、将来のモデルの機能要件や優先度に反映される。

ソフトウエア開発を成功させるためには、顧客からのフィードバックに基づいて要件を継続的に調整・修正することが不可欠である

最後に、新しいデリバリーモデルは開発効率の改善にも大きく貢献する。自動車メーカーは、ソフトウエアを頻繁に変更、適応、修正していくことになるため、プロジェクトの開始時に詳細な要件を指定する必要はなくなる。要件定義に費やす時間が短くなれば製品の量産までの時間も短縮できる。

A3. 新たなソフトウエア要件管理手法を導入

歴史的に見て、自動車産業は、統合されたバリューチェーンにおける要件管理において先駆的な存在であった。しかしながら、自動車メーカーは従来ハードウエアの要件管理に注力しており、その確立されたプロセスはソフトウエアの要件管理においては最適とは言い難い。今後、車載ソフトウエアの重要性が増し大きな差別化要因となることが予想される中、自動車メーカーはソフトウエアの要件管理に向けて新たな手法を導入する必要がある。マッキンゼーの調査によると、車載ソフトウエアの要件があまりにも細分化され過ぎて開発スピードが遅くなっているため、変革の必要性は非常に高いと言える。

自動車メーカーは顧客価値に基づいて要件を分類・統合する必要がある。ユーザーに直接影響する要件(いわゆるユースケースと呼ばれるもの)は最初の階層に統合すべきである。一方、特定の機能に必要なメモリ容量などの実装のための技術や要件(いわゆる実行要素と呼ばれるもの)は、別の階層に分類・統合すべきである。このアプローチをとることで、自動車メーカーは、ソフトウエア開発において価値を創出するための優先順位を適切に設定することが可能となる。企業が要件を複数の階層に分類・統合する際には、以下を考慮する必要がある。

戦略及び顧客価値を反映した要件設定を行う: ソフトウエア開発を成功させるためには、顧客からのフィードバックに基づいて要件を継続的に調整・修正することが不可欠である。企業は、初期的には事業戦略や目的を反映したソフトウエア要件を設定すべきであるが、顧客からのフィードバックや開発の進捗状況に応じて要件を定期的に調整していく必要がある。

エンド・ツー・エンドのトレーサビリティ(追跡可能性): 原材料の調達から製造、そして廃棄まで追跡可能な状態)を確保: バリューチェーン全体で要件を綿密に追跡することで企業は不要な労力を省き、開発を加速することができる。しかし、これを行うためには、その開発プロセスやツールチェーンにおいて、定義から承認までの要件プロセスがすべて詳細に追跡可能になっていなければならない。

トレーサビリティを確保することにより、企業は要件(顧客の視点)、必要な機能(開発者の視点)、成果物(テスト実施者の視点)について明確に把握することができる。

企業は高度に統合された少数のツール、もしくは対応するインターフェースを備えたそれぞれの専用ツールを活用し、以下の4つのステップを通じて、トレーサビリティをエンド・ツー・エンドで確保する必要がある。

- 要件のトレーサビリティ:機能からコンポーネントに至るまでの要件を詳細に定義する。

— バックログの管理: チームがソフトウエアの開発スプリントごとの要件のカバレッジを管理する際に役立つ(次のステップと密接に関連)。

— コード変化点の追跡: バックログ項目の更新を含む。

— 要件の評価: 各評価の進捗および評価結果(Pass/Fail)をチェックする。

これらのツールを使用して要件をリンクさせトレーサビリティを担保することで、開発者は、プロジェクトのどのフェーズにおいても効率的に変更作業を行い、ASPICEやUNECE5[5]などの規格で定義されているエンド・ツー・エンドのトレーサビリティ要件を満たすことができる。このアプローチでは、変更が生じた場合に、どの製品のどの部分が影響を受けるかを迅速かつ明確に把握することが可能となる。アジャイルプロセスではこのような要件変更は頻繁に発生する(例えば、顧客からのフィードバックに基づく変更など)が、これは望ましいものであり、企業はこうした変更に柔軟に対応できるようプロセスやツールを整備する必要がある。従来のソフトウエア開発におけるウォーターフォール型のプロセスでは、このような変更の発生は稀であり、通常は想定されていない。

オーバースペックを回避し明確にカテゴリ化: 企業は、ソフトウエア要件を特定し分類するためのベストプラクティスを確立することで、テストを簡易化することができる。優れた要件仕様は、明確で曖昧さがなく、他の要件から独立してテストすることが可能である。事業ポートフォリオの管理と同様に、要件もそのタイプ毎に区別して管理する必要がある。一般的なカテゴリとしては、法規制、安全性、戦略的・本質的改善、顧客価値、コスト削減などが挙げられる。さらに、企業は、要件間の相互依存関係についても透明性を確保し、明確に把握しておく必要がある。多くの企業は、これらをルール化してソフトウエア開発のプロセスや研修カリキュラムに組み込み、ソフトウエア開発のプロセスや見直し体制の最適化を図っている。

優先順位を決定し継続的に調整: 企業は、具体的なビジネスケースや戦略的目標に加え、開発段階(例えばテスト中)に得られた顧客からのフィードバックや学びを踏まえてソフトウエア要件を評価し、優先順位を決定する必要がある。さらに企業は、定期的に要件を再評価し、透明性の高い形でバックログを管理していく必要がある。

多くの企業は、幅広い知識を持つ人材をプロダクトオーナーとして任命している。彼らは、トレードオフの評価や、機能横断的に編成されたチームの指揮、要件に関する組織内の多くの異なる認識の共通化などを行っている。また、プロダクトオーナーは、ベストプラクティスを順守し、要件やユースケースのバックログを管理する責任も担っている。

B. Where:ソフトウエアをどこで開発するか: 組織、拠点、人材およびパートナー

ほとんどの自動車メーカーでは、大規模なソフトウエア開発に対応するための組織体制が

[5] Automotive Software Performance Improvement and Capability determination; United Nations Economic Commission for Europe.

整っていない。組織にソフトウエア開発担当の役員がいないことや、十分な数のソフトウエアエンジニアや設計者を確保できていないことなど、自動車メーカーには多くの課題が存在する。新しいオペレーティングモデルは、ソフトウエア開発に必要な組織体制、拠点および人材戦略に加え、外注戦略や他社との協業を踏まえたエコシステムの構築を定義することで、これらの課題に対処することが可能となる。

B1. 新しいオペレーティングモデルに組織を適応させ、グローバルな開発拠点を構築

組織レベルでは、自動車メーカーのほとんどが、ソフトウエアの重要性を高める要因であるACESのトレンドに対応する体制が整っていない。例えば、自動車メーカーの多くは、ソフトウエアに関する意思決定に時間がかかる傾向があり、また多くの企業が、車両プラットフォームにおけるソフトウエアの包括的な所有権やエレクトロニクス戦略、および関連予算を明確にしていない。新しい環境で競争力を維持していくためには、企業は組織体系を見直す必要がある。大きな目標の一つは、開発プロセス全体を通じて、アーキテクチャ定義や要件定義、開発プロセスにおけるインターフェースの数を削減することである。開発のすべての段階で部門間の情報共有や協働を促進することで、企業は重複する作業を減らし、既存および新たなケイパビリティの向上に注力することが可能になる。多くの企業は、アーキテクチャを一元管理化し、組織全体でベストプラクティスを共有する取り組みに着手し始めている。例えば、ある自動車メーカーでは、5,000を超えるソフトウエア開発者を一元管理する機能を構築した。しかし、依然としてほとんどの自動車メーカーが、ソフトウエア開発において分散型のアプローチをとっている。

組織体系: 自動車メーカーがソフトウエアの開発手法を改善するために採用できる組織体系モデルがいくつか存在する。企業にとって最適な選択肢となるのは、各企業の優先事項を反映したものであり、その優先事項には、意思決定の迅速化、インターフェースの数の削減、責任の明確化などが含まれる。さらに重要なのは、社内の各部署(研究開発、調達、製造、営業およびアフターサービスなど)から出されるソフトウエアに関する要件の調整を包括的に行うことである。多くの場合、ソフトウエア要件やライフサイクルは部門によって異なるため、包括的な調整を行うことで、シームレスなユーザーインターフェースを構築し、開発プロセスを効率化することが可能になる。企業は、このような組織体系の見直しに加え、カルチャーギャップを解消し、全部門の業務プロセスを調和させるための取り組みに大胆な投資を行うべきである。このような取り組みには継続的なチェンジマネジメントが必要になるが、自動車産業におけるソフトウエア開発機能の強化のためには非常に有効と言える。

組織体系を設計する際には、自動車のソフトウエア開発部門について、その企業の中での役割や、製品もしくはプロジェクト、技術を考慮することが望ましい。しかしながら、既存の組織体制では、これらの側面のいずれか1つに偏ることが多い。

まずは、1つ目のモデルである機能的な役割を重視した組織体系について説明する。このモデルでは、製品やプラットフォームに特化し

たプロジェクトには、各機能部門から選出されたメンバーが参画する。製品や技術に関する事項については、間接的な報告という形で各機能部門に伝達される。このモデルでは、新しい製品・プロジェクトや技術に対応するために改めて組織を再編成する必要がないため、組織は最大限に柔軟な運営を行うことができる。しかしながら、この組織体系下での開発効率は必ずしも最良というわけではない。そのため、プロジェクト管理、アーキテクチャ、人員配置など、部門を超えた機能の必要性が高まるだろう。

2つ目のモデルは、プロジェクトを重視した組織体系である。具体的には、特定の顧客や、車両モデル、個別車両、プラットフォームなどに関するプロジェクトを中心に据え組織体系を構成する。製品や技術に関する事項については、間接的なレポートラインや既存のプロセスを介して各プロジェクトに伝達される。この組織体系では、顧客と最終製品を最大の焦点に据えるが一方で冗長性が生じるリスクや、異なる製品やプロジェクトグループ間の障壁となり、技術の共有の妨げになる可能性もある。

こうしたリスクを緩和するためには、強力なプラットフォームとアーキテクチャ機能の構築が有効である。

3つ目のモデルは、ネットワーク、ヒューマンマシンインターフェース、バックエンドなどの技術やドメインに焦点を当てた組織体系である。このモデルでは、製品毎のプロジェクトに技術部門のメンバーが配置される。このアプローチでは、間接的なレポートラインや既存のプロセスを介して技術や役割に焦点を当てることが求められる。このモデルでは、技術やドメインに関する深い専門性が形成される一方で、プロジェクトのスコープや要件、仕様については柔軟性がほとんどないため、たとえプロジェクト期間中にこれらが変更されたとしても対応は難しい。企業は、通常、多くの製品を開発・維持する際にこの組織体系を採用する。

自動車メーカーがソフトウエア競争に勝ち残るためには様々な面で優位性を実現する必要があるが、優秀な人材の確保と維持は最も重要な要素と言える

B2. 優秀なソフトウエア開発人材へのアクセスを確保し、採用・誘致するために組織の魅力を高める

自動車メーカーがソフトウエア競争に勝ち残るためには様々な面で優位性を実現する必要があるが、優秀な人材の確保と維持は最も重要な要素と言える。

自動車メーカーのほとんどが、ソフトウエア開発を大幅にアウトソースし、戦略的パートナーシップに大きく依存している。ACESのトレンドの勢いが増す中でソフトウエアの重要性は大幅に高まっており、ソフトウエア開発の生産性についても一定の上昇が予測されるが、ソフトウエアエンジニアの需要は2030年までに3〜4倍に増加すると見込まれている。ソフトウエア人材は、テック企業やその他の業種の企業とも争奪戦となっており、自動車産業が優秀な人材を獲得するためには採用制度を抜本的に変革していく必要がある。そうでない限り、必要な人材の確保は今後ますます難しくなるであろう。

自動車業界における人材確保において留意すべき点は、人材の多様性である。マッキンゼーのベンチマーキングによると、多様な経歴を有する人材で構成されたチームは、同質的なチームに比べて開発の生産性が10%以上上回っている。多様な人材を確保していくにあたり、以下の取り組みが有効であると考えられる。

ソフトウエア開発人材の獲得のためグローバル市場へのアクセスを高める: 包括的な拠点戦略は、企業がコストを抑えながらソフトウエア開発活動を拡大し、関連するケイパビリティを構築し、企業のキャパシティを高めるうえで非常に役立つ。また、優秀な人材の獲得を巡る競争にも有効と言える。一部の革新的な企業は、自動運転の開発拠点を、デジタル人材が豊富な地域に設立している。しかし、大手企業の多くが、これまでの事業拠点やハードウエアの開発拠点を維持しており、このことが、優秀なソフトウエア開発人材を獲得・維持するうえで足枷となっている可能性がある。もし、このような企業が、デジタルにおける重要地域に進出し、拠点を設けてグローバルに事業を展開することができれば、近隣の大学や教育機関から優秀な人材を獲得することができる。企業は、また、このようなつながりを強化するために、大学と共同でフェローシッププログラムなどを開発し、新卒の学生や専門スキルを有する人材を確保することも可能となる。

グローバルな拠点の構築には様々なメリットが期待できるが、遠隔の拠点を管理するためには適切なオペレーティングモデルが必要となる。例えば、リサーチによれば、ソフトウエア開発の生産性は、開発プロジェクトに新たな拠点が加わるごとに、平均して10%程度低下するという結果が出ている。

ソフトウエア開発人材にとって魅力ある職場づくりを行う: マッキンゼーの調査結果によると、ソフトウエアエンジニアが就職において重視する要素は、報酬、キャリア機会、業務内容および企業文化となっている。現在、自動車産業はこのいずれのカテゴリにおいてもテック企業に後れを取っている。この差を縮めるためには、前述のすべての要素について、ソフトウエア人材の確保に的を絞った訴求価値の強化を行う必要がある。雇用の安定性や

社宅の提供など、従来の福利厚生を強調するだけでは十分とは言えない。

自社の組織能力の構築を推進: 企業は、ハードウエア開発経験を有する既存の人材に対し、可能な限り再教育を提供し、ソフトウエア関連のポジションに就かせることが望ましい。例えば、企業はハードウエア開発のプロジェクトマネージャーにソフトウエア開発プロジェクトの監督を任せられるよう訓練することができる。しかし、こうした試みが成功するのは、ソフトウエアの専門家と再教育を受けたハードウエア開発の専門家のバランスが適切な場合に限られる。このバランスは企業により異なり数値化することは難しいが、大まかなルールとして、ソフトウエア開発の専門家と再教育を受けたハードウエア開発の専門家の割合が2対1の場合に成功する確率が高いことが多数の企業で立証されている。最良の結果を得るためには、企業は、社員にとって魅力的な研修や実地訓練プログラムを開発し、社員が迅速にソフトウエアの専門用語や必要な知識を習得できるよう尽力する必要がある。また、学習は一時的なものではなく、生涯にわたり継続して行うべきものであることも社員にアピールしていく必要がある。

社員に再教育を行うことは人材不足の解消に効果的であるが、注意すべきは、特定の業務に長期間にわたり従事してきた社員の多くは、問題解決のためのパターンが身についてしまっている点である。ハードウエア開発に従事した経験を持ちソフトウエア開発の再訓練を受けた社員は、予期せぬ事態が発生した場合に、アジャイル型のアプローチではなく、ウォーターフォール型のアプローチで問題を解決しようとする傾向がある。

そのため、企業は、社員に再訓練を行う際には、スキルだけでなく新しい考え方を習得させる必要がある。これが成功すれば、生粋のソフトウエアの専門家と再訓練を受けた元ハードウエアの専門家の差異を解消することができる。逆に、これを怠ると、全社的なアジャイル手法への移行を妨げることになる。

ソフトウエア人材は、テック企業やその他の業種とも争奪戦となっており、自動車産業が優秀な人材を獲得するためには採用制度を抜本的に変革していく必要がある

明確なキャリアパスを提供: ほとんどの自動車メーカーは、テック企業と比較して、キャリアパスや成長機会の定義が不明確である。自動車メーカーでは、スペシャリストとしてのキャリアパスがまだ広く提供されておらず、さらに企業の多くが、特定の役職や職位に期待される職務内容を明確に定義していない。そのため、スキルをさらに高めたいと願っているエキスパートも、他の選択肢がないことから、管理職に就かざるを得ない状況にある。

自動車メーカーが人材の定着率を高めるためには、全階層について、特定のスキルに紐づいた明確なキャリアパスを形成することを考慮する必要がある。キャリアパスには、スペシャリスト向けのものもあれば、昇進を目的としたものもある。後者の場合は、垂直的な昇進(例えば、ジュニア開発者からシニア開発者、そしてチームリーダーへ)を提供し、同様に、6ヵ月程度の水平方向の業務のローテーションを実施して、リーダーシップやプロジェクトマネジメント、ソフトウエア開発などに関連する様々なスキルを習得させることができる。また、企業は、全社にこれらを展開するために、機能別や学際的な専用のトレーニングプログラムを開発することも検討すべきである。キャリアパスを明確にすることは、多くの場合、企業の効率性を高めることにもつながる。なぜなら、エキスパートは、各分野の非専門家よりも一般的に生産性が高いからである。

B3. 明確な内外製戦略を立案し、パートナーシップのエコシステムを構築

これまでに獲得した競争優位性を維持し、その他大勢のハードウエア開発企業の一つと見なされないようにするためには、企業は、明確な顧客戦略を立案し、自社の差別化要因や訴求価値を特定・強化することに加え、開発アーキテクチャをモジュール化する必要がある。

企業は、これらを完了した後に、以下の3つの側面に焦点を当てることで明確な内外製戦略を立案することができる(図表5)。

- 開発プロセスの各フェーズ(システム統合や受け入れテストなど)
- ソフトウエア技術の蓄積
- インフォテインメントやパワートレインをカバーするソフトウエアドメインやモジュール

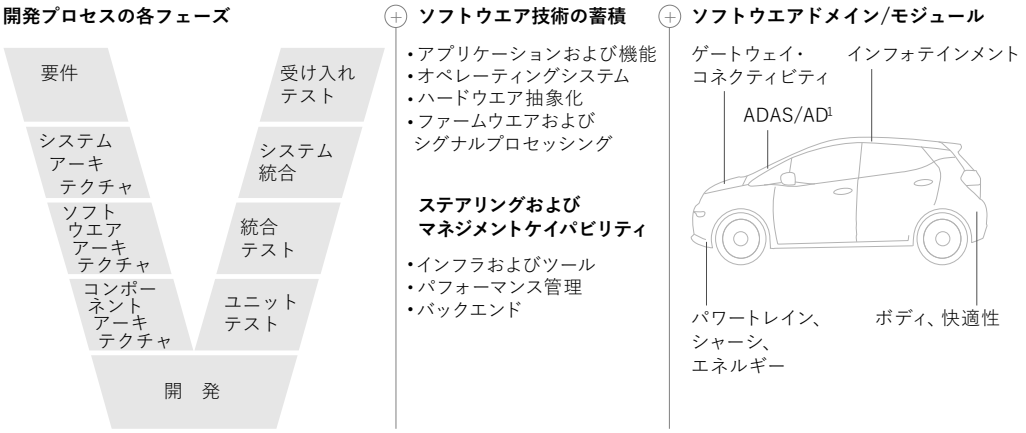
これらの側面を踏まえて、企業は、内外製戦略を自社の総合的な経営戦略と整合させるための制御点を特定し、定義する必要がある(例えば、重要な知的財産、品質基準、独自のイノベーションなど)。

もちろん、企業は、内外製の意思決定を行う際に、調達のしやすさなどを考慮する必要もある。また、このほかにも考慮すべき要因としてコストなども挙げられるが、これは決定事項にはならないことが多い。

図表 5

内製・外製についての意思決定を行う際は、3つの側面について考慮すべきである

自動車の例



[1]ADAS: 先進運転支援システム、AD: 自動運転

資料：マッキンゼー

自動車メーカーは、意思決定の質を高めるために、自社の優先事項や市場状況を踏まえて各要素を検討していくことが望ましい。

企業がソフトウエアの自社開発を決定する際には、その決定が社内の技術的な生産能力に及ぼす影響を評価し、既存の従業員が必要なケイパビリティをすべて備えているかどうかを踏まえて、自社の組織体系やプロセスを精査する必要がある。企業が適切なケイパビリティや十分な生産能力を持たない場合には、他社の買収や合弁などの選択肢を検討し、重要な制御点を掌握できるようにする必要がある。

企業がソフトウエア開発の外製を決定した場合には、調達モデルを詳細に定義する必要がある。加えて開発パートナーの選出と契約の締結については、綿密な評価・精査を実施する必要がある。また、複雑なソフトウエアシステムの部分的な外製を検討する場合には、企業は契約先を多くとも2〜3社に抑えるべきである。

マッキンゼーの調査結果から、これ以上提携先が増えると、生産性が65%以上低下する可能性があることが分かっている。

包括的な内外製戦略を決定する際には、企業は、できるだけ標準およびオープンソースを活用すべきである。なぜなら、これらを活用することで、ソフトウエア開発において非常に大きなメリットを享受することができるからである。もちろん、企業がオープンソースを活用する際には、明確なルールやプロセスを策定し、ライセンシングや法的責任およびメンテナンスについて細心の注意を払う必要がある。また、自動車メーカーやサプライヤがオープンソースのコンポーネントを自社製品に組み込む場合には、正式な契約を締結する必要がある場合もある。

自動車メーカーは、これまでソフトウエア開発よりもハードウエア開発を重視してきた。しかし、今やソフトウエアは製品開発における主要な価値創出要素となっており、自動車メーカーはこうした考え方や従来の開発プロセスを改めるべき時期に来ている

最後に、自動車メーカーは、エコシステムを共同で構築するための戦略的パートナーシップを締結する必要がある。こうしたコネクションを持つことで、企業は互いに学び合いながら開発を促進し、コストを抑えることができる。加えて、共同開発することで、市場参入の遅れに伴うリスクも低減することができる。

優れた内外製戦略では、自動車メーカーは、自社の差別化要因となり得る機能を内製し、重要度の低いソフトウエアの開発をプロバイダーや外部業者に委託する。このような手法をとることで、自社でソフトウエア開発人材を多く確保する必要がなくなる。

C. How to develop: ソフトウエアをどのように開発するか: アジャイル手法、分離(デカップリング)、テスト

自動車メーカーは、ソフトウエア開発よりもハードウエア開発を重視してきた。しかし、今やソフトウエアは製品開発における主要な価値創出要素となっており、自動車メーカーはこうした考え方や従来の開発プロセスを改めるべき時期に来ている。これを実現するためには、アジャイル手法を全社規模で導入し、ハードウエアとソフトウエアの開発プロセスを分離し、テストの自動化と継続的インテグレーションを推進していく必要がある。

C1. アジャイル手法を全社規模で導入

アジャイル手法をハードウエアとソフトウエアの開発の双方に採用することで、企業は生産性を向上させ、急速な環境の変化に迅速に対応することが可能になる。業種を問わず、アジャイル変革を実現した企業は、生産性と実行スピードを30%向上させると同時に、市場投入時点の製品の欠陥率を70%以上削減している[6]。また、アジャイル手法は、予算や期限、品質に関連するプロジェクトのリスクを低減することにより複雑性という課題を克服するうえで、非常に重要な役割を果たす。

[6] マッキンゼー独自のSoftCosterデータベースによる

しかし、現時点でソフトウエア開発にアジャイル手法を継続的に採用している自動車メーカーは限定的である。特に先端技術開発においては、多くの企業がパイロットを実施しているが、大規模にアジャイル手法を採用している企業はほんの一握りである。

採用率が低い要因の一つとして、自動車のアプリケーションには非常に特殊な要件が多くあるため、標準的なアジャイル手法を全社規模で採用することが難しい点が挙げられる。さらに、自動車に関する一連のアジャイル手法は、システムの複雑性やハードウエア開発との複雑な相互依存性に対応可能なものである必要があり、サイバーセキュリティや車両の安全性、品質に関する厳格な規制要件にも対応できるものでなければならない。

ハードウエア開発との相互依存性: 自動車メーカーがハードウエアとソフトウエア間の相互依存性を管理するために有効な手法として、開発手法と要件管理手法を組み合わせた、全社もしくは組織全体におけるアジャイル手法の採用が考えられる。これらの手法を合わせて導入することで、ハードウエアとソフトウエアを円滑に統合し、開発のタイミングを同期することができる。

全体的なアーキテクチャ、インテグレーションおよびテストを実施するための古典的な開発手法は、製品のライフサイクルを統合的に管理し、企業が規制要件を満たすうえで役立つ。

企業は、開発手法を用いて、車両やドメインの全体的なアーキテクチャを定義することができる(図表6)。これにより、アジャイルチームに対し、全体の概要や境界条件について質の高いインプットを提供することが可能になる。アジャイルチームは、これらのインプットに基づいて、コンポーネントの開発やテストを実施する前にソフトウエア要件の詳細を詰めることができる。開発サイクルの終了時には、各ドメインと車両の統合が完了し、テストが無事に終了すればシステム全体が完成され、チームは開発プロセスを完了することになる。

プロジェクトマネジメントの観点からは、制御ユニットとドメインアーキテクチャの特殊要素の優先順位と必要な同期ポイントについて、全ファンクションで整合を図ることが目標となる。例えば、企業は、開始時期に制約のある必要な機能に優先的に取り組み、進捗状況を追跡する必要がある。

一方、優先度の低い機能については、要件全体の完了指標と同時に監視すればよい。

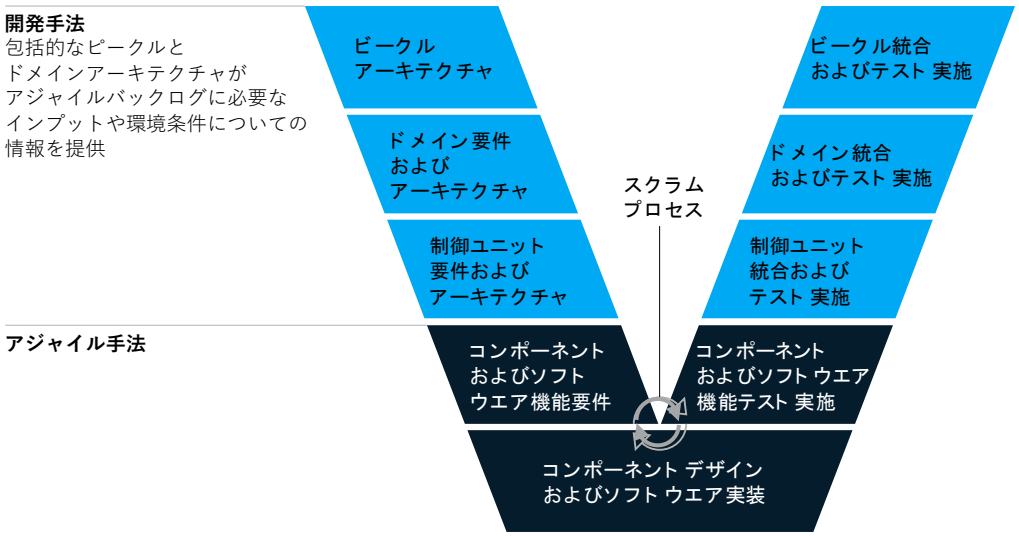
規制要件および業界基準: アジャイル手法は、自動車産業においても有効に機能し、大幅な効率性の向上をもたらすが、アジャイル手法を導入するにあたっては、関連する法規制や、ハードウエアとソフトウエアの本質的な統合が不可欠であることを考慮する必要がある。例えば、ISO 26262規格(機能安全に関する認証)を順守するための認証を取得するために成果物の作成を求められるなど、従来の作業プロセスを変更する必要に迫られる場合もある。従来のウォーターフォール型開発では、認証を得るための成果物は、一般の開発手法(計画立案、顧客要件の定義、システムアーキテクチャの設計、ソフトウエア要件の定義、ソフトウエアの実装、ソフトウエアのテスト実施)と同様の手順で作成される。アジャイル開発では、最適なアプローチとして、一般の開発

手法とは逆の認証方式をとる。すなわち、すべてのソフトウエア要件とコードが確定したプロジェクトの終了時、つまり製品の市場投入の直前に、認証を得るための成果物を作成する。このプロセスのもとでは、成果物を複数回にわたり作成する必要はなく、プロジェクトの開始段階から認証担当者と必要な情報を共有し認識を共通化することで、無駄な作業を最小限に抑えることができる。ハードウエアとソフトウエアを分離させることで、ISO 26262の認証作業はさらに簡略化される。なぜなら、ソフトウエアを再利用する場合、そのコンプライアンスの手続きにおいて、同ソフトウエアが他のアプリケーションで問題なく機能していることを、安全性を裏づける根拠の一つとして挙げることができるからである。

図表 6

自動車メーカーは、総合的な開発手法にアジャイルソフトウエア開発手法を取り入れるべきである

開発手法とアジャイル手法の組み合わせ



資料：マッキンゼー

Automotive SPICEなどの業界基準では、現在、車載ソフトウエアに関する全要件のトレーサビリティを確保すること、および使用したプロセスやツールを監査対象とすることが義務づけている。要件のトレーサビリティについては、アジャイル手法と互換性があり、自動化されたツールチェーンにより効率的に実現することができる(セクションA3: 要件管理、D2: 標準ツールチェーンを参照)。しかしながら、プロセスやツールに対する監査の義務づけは、アジャイル手法の特徴である継続的な改善システムの足枷となる恐れがある。「純粋なアジャイル」チームは、その作業プロセスやアプローチをすべて独自の裁量に基づいて改善することができるが、自動車メーカーは一定の基準を順守する必要があり、チーム

間でアクションの足並みをそろえなければならないことから、進捗スピードが遅くなる可能性がある。

アジャイル手法を使用するタイミング

自動車メーカーのソフトウエア開発チームは、バックログをあらかじめ定義する必要があり、厳格な監査を実施するよう義務づけられているが、ほぼすべてのアジャイル手法を容易に採用することができる。これにより、チームメンバーの働き方は劇的に変化する。

アジャイル開発手法は自動車産業においても有効に機能し、大幅な効率性の向上をもたらすが、アジャイル手法を導入するにあたっては、関連する法規制や、ハードウエアとソフトウエアの本質的な統合が不可欠であることを考慮する必要がある

アジャイル型に移行するにあたり、企業は、オペレーションの設計についてマクロレベルの変革を伴う重大な意思決定を行う必要がある。これには、ポートフォリオ管理、リソース管理、およびプロジェクトマネジメントが含まれる。一般的に、アジャイル手法を全ファンクションに大規模に取り入れ新しい働き方を採用できるのは、自動車メーカーにおいては「ソフトウエアファクトリー」のような先端技術開発ユニットに限られる。しかしながら、例外も存在する。例えば、自動車産業の先進企業の中には、企業規模でアジャイルを導入するためのフレームワークであるSAFe(Scaled Agile Framework)などのアジャイル手法を全社で採用し、研究開発業務全体の舵取りを行っているところもある。

一方、個別のチームがアジャイル手法を採用する場合には、その原則を順守すべきである。例えば、チームは、機能横断的に編成され、同じ場所で作業し、一定の期限を設定して反復的にテストを実施することなどが重要となる。

他の業界と同様に、アジャイル開発のメリットは、個別の機能の開発を担当するチームの

運営にアジャイル手法を適用した場合に最も顕著となる(図表 7)。

図表 7

アジャイル手法は、様々な側面における変革を促進する

変革の例(非網羅的)

カテゴリ	事　例	導入前	導入後
体　制	プロダクトチームの構成	パートタイムのスタッフで構成	フルタイムのスタッフで構成
		機能毎にチームを構成	機能横断でチームを構成
働き方、役割および責任	成果主義 ベースの評価・目標設定	活動ベース	成果主義もしくは重要業績評価指標ベース
		事前に決定されたプロジェクト計画に照らして評価	継続的評価
	継続的なリリース頻度	ウォーターフォール型、年に 3〜4 回	アジャイル型、週次もしくは日次
	意思決定	ヒエラルキー型、上層部が実施	プロダクトオーナーレベル(何を開発するか)、およびアジャイルチームレベル(どのようにして開発するか)
技術およびアーキテクチャ	モジュール式もしくは分離型のソフトウエアアーキテクチャ	モノシリック(分割されていない1つのモジュールで構成)	サービス中心
	拡張性のあるITインフラ	固定的なアーキテクチャ。寄せ集めのITシステムや非完全なデータモデルで構成	モジュール式ITアーキテクチャ。合理的で拡張性のあるITインフラ

資料：マッキンゼー

自動車メーカーがアジャイル型に移行する場合は、以下を実行する必要がある。

- 本格的な実装を可能にするために、アジャイルプロセスに開発サプライヤを参加させる。
- 事前に明確に定義された仕様を前提とした従来の調達契約からスプリントベースの開発パートナーシップへの移行に伴い、それに適応するよう調達プロセスを変更する。
- 拠点戦略やサプライヤと自動車メーカーの協働を阻む法律上の課題を解消する。

C2. ハードウエアとソフトウエアを分離し、二速型の開発プロセスを実現

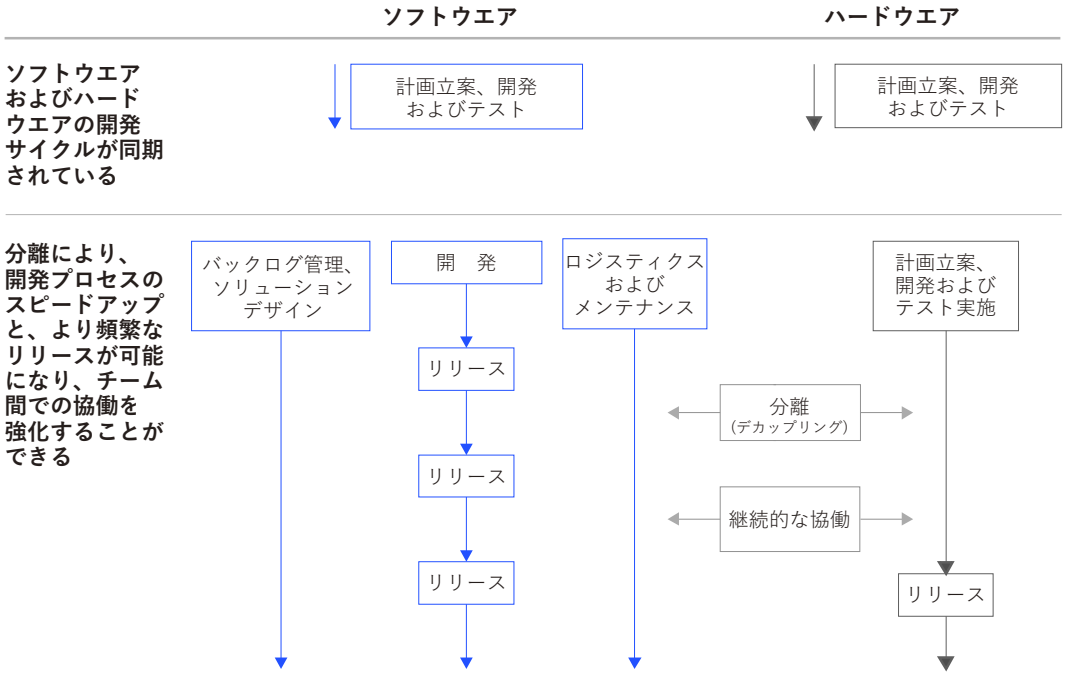
自動車メーカーは、ソフトウエアの開発プロセスをハードウエアから分離させることで、よりダイナミックな開発サイクルを実現することができる。この開発サイクルでは、厳格ではるか先に設定された車種ベースのSOP(生産開始日)に縛られることなく、頻繁にソフトウエアをリリースすることが可能となる(図表 8)。製品とライフサイクルの管理をハードウエアから切り離すことは、車種ごとのSOP志向から脱却するための鍵となる。

これを実現するためには、それぞれのバックログやロードマップを別々に管理し、ハード

ウエアとソフトウエアの開発のマイルストーンを明確に定義して一定の整合性を持たせる必要がある。また、ベンダーと協働する場合には、開発プロセスの分離を可能にするために契約内容を見直す必要があるかもしれない。

図表 8

企業は、ソフトウエアとハードウエアの開発サイクルを分離させることで多くのメリットを享受することができる



資料：マッキンゼー

最後に、ソフトウエアやインテグレーションのテストやデプロイメントの自動化を進めていく必要もある。

システム開発チームは、アジャイル手法と同様に、ハードウエアとソフトウエアのバックログの分離が可能となるよう、両チームが使用する。

インターフェースを定義・管理し、全体で同期がとれているかを確認することもできる。もちろん、ハードウエアとソフトウエアが分離されていれば、企業はアーキテクチャのフリーズポイントを別々に設定できるため、システム全体で同期をとる必要はない。

アジャイル手法と同様に、ソフトウエアとハードウエア開発を分離するためには、いくつかの前提条件を満たす必要がある。具体的には、アーキテクチャが標準化されモジュール化されていること、堅牢なテスト手法があることなどが挙げられる。先述した通り、企業は強力なミドルウェア層を導入することでハードウエアの機能を抽象化し、上層部で使用されている標準APIを介してソフトウエアの機能やサービスを利用できるようになる。

しかし、このような分離を実行するうえで最も重要な要素となるのは、堅牢なテスト手法である。そのため、企業は、テスト車両を提供しメンテナンスするための方法を定義する必要がある。具体的には、ハードウエア・イン・ザ・ループ手法あるいはソフトウエア・イン・ザ・ループ手法のどちらかを採用するか、もしくは、より広範なシミュレーションインフラを導入するかを決定する必要がある。

アジャイル手法の導入やハードウエア・ソフトウエア開発手法の分離は、いずれもバリューチェーンの下流に大きなインパクトをもたらし、特に調達部門に大きな影響を与える。例えば、調達部門は、従来のウォーターフォール型の調達プロセスから、アジャイルや分離された開発手法に対応可能な調達プロセスにシフトする必要がある。つまり、今後自動車メーカーは、特定のベストプラクティスに従って、ソフトウエアの戦略上の重要性を継続的に評価し、要件がどのように進化していくかを理解し、どの調達モデルが各ケースにおいて最適かを判断するなどを実行していく必要がある。このような変化に対応するためには、ソフトウエア開発における総保有コスト(TCO)の概念に加え、戦略的なパートナーシップに基づく新たな提携モデルも必要になる。

C3. テストの自動化を進め継続的インテグレーションを普及させることで、できるだけ早い段階でのエラー検知を実現

組み込みソフトウエアの数は増え続け、その機能の継続的なアップデートに対するニーズも高まっている。こうした中、自動車メーカーは、バグやインターフェース上のエラーを可能な限り早期に発見し解消する必要に迫られている。仮にエラーの解消が遅れた場合、新たに膨大なバグが発生してしまい、そのエラーの追跡に膨大なリソースが必要となり、開発の遅れや検証作業の増加などを招く恐れがある。

アジャイル手法と同様に、継続的インテグレーションやテストの自動化を本格的に導入している自動車メーカーはほとんどない。しかし、こうした傾向を覆し新たな手法の導入を進めれば、自動車メーカーは、劇的に生産性を向上させると同時に、ローンチリスクを低減させることも可能になる。我々の経験では、新たなアプローチを導入した企業の多くが、生産性を40%以上向上させながら、バグを60%以上削減することに成功している。

こうした企業の先例に倣うためには、企業は相互関係にあるソフトウエア開発の2つのベストプラクティスを採用することが望ましい。まずは、1日数回、コードを共通のレポジトリに保存・統合し、自動でテストするシステムを導入する。コードを早期に統合することで、開発者は「早い時点で失敗」し、継続的インテグレーションやツール、自動化を活用することで、エラー要因を容易に特定することができる。サプライヤは、システムレベルでこうしたメリットを独自で実現することができる。また、車両システムレベルでは、知的財産関連の制約をクリアし、コーディングを内製するか、自動車メーカーがサプライヤとコードを共有するために「ホワイトボックス」手法を導入する必要がある。2つ目のベストプラクティスは、テスト主導の開発および自動化の導入である。これは、コーディングを開始する前にテストを定義し、コード統合後にテストを自動的に実行するプロセスである。テスト部門ではなく、ソフトウエアの設計者や開発者自身が、顧客とともに開発プロセスの

中でテストを繰り返し実施し、改善していく。これにより、開発者は、コーディングを行う前に、システムをどのように使用し実装するかについて検討する必要がある。

こうしたベストプラクティスを採用することで、包括的で自動化されたテストケースにより、高品質で持続可能なスプリントを実現することが可能になる。

D. How to enable: ソフトウエア開発をどのようにして進めるか: パフォーマンスおよびツールチェーン

ソフトウエア主導のオペレーティングモデルで効率を向上させるためには、最適なパフォーマンスマネジメント手法を採用し、ソフトウエア開発ツールチェーンを構築することで、最先端のソフトウエアインフラを構築する必要がある。

D1. データ主導の生産性、プロジェクトの成熟度、品質評価を考慮したパフォーマンスマネジメントを実施

ソフトウエアの複雑化に伴い、自動車メーカーは、生産性やプロジェクト成熟度、品質に関する標準化されたデータ駆動型の評価基準を用いて、パフォーマンスマネジメントシステムをアップグレードする必要がある。事実に基づいてリアルタイムでパフォーマンスマネジメントを実現し、時間やコスト、品質に関するソフトウエアの潜在的な課題に積極的に対応していくためには、自動化されたデータ駆動型のインサイトが不可欠となる。

最適なパフォーマンスマネジメントは、以下の3つの項目において優れている。

- コスト、時間、品質などのKPIをカスケード接続し、各KPIが総合的な事業上の目標やオペレーション上のタスクとリンクした包括的なシステムを構築する。
- 課題を上申するための効率的なシステムを構築する。例えば、短時間で簡潔な報告や意思決定、上申などを行うための定例ミーティングなど。
- ソフトウエア開発において生産性と品質の客観的な評価手段を確保するために、ツール、自動計測テクノロジ、コード品質の評価基準などを活用する。

最先端の標準開発ツールチェーンの実装は、テストの自動化やアジャイル手法の導入により生産性を30～40%上昇させるための重要な手段となる

D2. ソフトウエア開発ツールチェーンを最新バージョンにアップグレード

自動車メーカーがソフトウエア開発における効率性を高めるために、継続的な統合に対応し、APIを活用した標準化されたツールチェーンを導入することは有効である。このツールチェーンは、ソースコード管理プロセスやビルド、継続的インテグレーションおよびテストの自動化(テスト実行、テスト判定、テストレポートの生成)に特化したツールで構成されることが一般的である。前述の通り、このツールチェーンは、要件管理に必要な全ツールをシームレスに統合する。

高度に自動化された統合ツールチェーンを構築することで、開発プロセスにおいて多くのメリットを享受することができる。例えば、業務の複雑性を低減し、外部のビルディングブロックが提供する技術を活用することにより、社内のソフトウエア開発業務の効率性を高めることができる。マッキンゼーの調査結果および経験によると、最先端のツールチェーンを導入することで以下を実現することができる。

- 開発プロセスを迅速化し、コードのリリースやビルドに要する労力を90%削減する。
- 厳格な品質ゲートを活用することで、バグを最大50%削減する。
- 1日当たり数千に上るコードの編集を自動化する。

全体で見ると、標準化された最先端の開発ツールチェーンの実装は、テストの自動化やアジャイルの導入により生産性を30〜40%上昇させるための重要な手段となる。このレベルのパフォーマンスを実現できれば、トップパフォーマーとなることも夢ではない。自動車メーカーは、このようなソフトウエア開発ツールチェーンを、継続的な統合や標準APIを活用する高い生産性を誇る企業となるための中核として捉える必要がある。このようなツールチェーンを導入することで、全開発プロセスを通じて、ソフトウエア・ハードウエアの開発ツール間のインターフェースにおいて効率的な自動化を実現することができる。

加えて、テストランなど複数のプロセスも自動化することが可能になる。最終的な目標として、開発スピードを加速させ、早期段階でのテストを可能にすることを目指す。

使用されるツールチェーンやツールの数が管理不能なレベルにまで増加し、透明性を確保することが難しくなる場合も多くある。こうした問題を回避するためには、経験豊富なツールチェーンマネージャーがツールを定期的に精査し、必要に応じて改善措置を講じていく必要がある。

変革を実現する

多くの企業は、ソフトウエア開発により価値を創出しACESのトレンドがもたらす機会を捉えるために、既に本格的な取り組みを開始している。

通常、変革の道のりは、単一の課題もしくはトピックに狙いを定めることから始まる。例としては、ソフトウエアプロジェクトの立て直しのための緊急の救済策の策定、ソフトウエアの内製もしくは外製についての意思決定、ソフトウエア開発のパフォーマンスを改善するための変革などが挙げられる。しかし、ほとんどの企業は、このような単一のテーマを解決しただけではそのインパクトは限定的であることを認識しており、これは特に企業が改善を維持するために不可欠な基盤を備えていない場合に当てはまる。世界に通用するソフトウエア開発能力を構築するためには、企業は自社の変革に一気通貫で取り組む必要がある。

各企業の状況により、変革に向けてとるべきアプローチは異なる。最初に注力する領域を決定する際には、以下を参考にするとよい。

What: 製品アーキテクチャおよび内製・外製の決定: 社内のコントロールポイントとパートナーシップのエコシステムを含め、目標とするソフトウエアアーキテクチャを決定することを目的とし、プラットフォームを横断して、競争力のあるソフトウエアを効率的に開発できるようにする（セクションAおよびB3を参照）。

How: 生産性を向上させる方法およびソフトウエアの開発手法: ここでは、アジャイル手法、継続的インテグレーションおよびテストの自動化など、ソフトウエア開発の主要な効率化手法を組み合わせて、ソフトウエアの研究開発に関する生産性を向上させることに重点を置く（セクションCおよびD2を参照）。

Where: 合理的なコストで優秀な人材を確保: ソフトウエア開発を強化するために必要となる人材が不足している企業は、まずこのテーマに注力する。これを実現するためには、人材へのアクセスと研究開発コストの両方を最適化するために、企業の活動拠点を増やし、ソフトウエア人材にとって魅力ある組織づくりを行う必要がある（セクションB2参照）。

How to set up: 組織およびガバナンス体制: もう一つの着手点として、最適な組織体系を定義することが挙げられる。まず「部署と報告体制（“boxes and lines”）」を決定し、ソフトウエアの開発を加速させるためのオペレーティングモデルを、指揮系統およびパフォーマンスマネジメントを含め明示する（セクションB1およびD1を参照）。

自動車業界が抱えるソフトウエアの複雑性および生産性に関わる課題を克服するには、車載ソフトウエアの研究開発を包括的に変革することが不可欠である。CTOやCEOは、企業が現在の競争力を維持し今後も成功していくために、これらの課題に最優先で取り組むことはもちろん、変革の長い道のりに備える必要がある。組織体系やオペレーティングモデルを含め、ソフトウエア開発に関連するすべての課題に対処するためには、数年をかけて変革を行っていくことになるだろう。

オンドレ・バーカキは、マッキンゼーミュンヘンオフィスのシニアパートナー、ドミニク・ヘップは、同オフィスのアソシエイトパートナー。ヨハン・ダイヒマンは、シュトゥットガルトオフィスのパートナーであり、ステファン・フランクはハンブルクオフィスのコンサルタント。アンドレ・ロシャはマドリードオフィスのパートナー。小田原 浩は東京オフィスのシニアパートナー、松原 寛は同オフィスのアソシエイトパートナー、木下 暢および小泉 正剛、三宅 匠は同オフィスのコンサルタントを務める。

本稿の作成にあたり、Silviu Apostu、Georg Doll、Julia Gößwein、Anna Gomulec、Virginia Herbst、Shannon Johnston、Maximilian Lemmens、およびHenrik Rochlitzに多大なる協力を頂いた。
ここに感謝の意を表したい。

Designed by Global Editorial Services