車載システム開発の技術動向と現場での取組みのご紹介

ソリューションカンパニー
ソリューション企画部
アシスタントサービスプロデューサー
彭　超

ソリューションカンパニー
ソリューションビジネス第三部
長壁　豊

ソリューションカンパニー
ソリューションビジネス第三部
山本　将史

ソリューションカンパニー
ソリューションビジネス第三部
竹村　孝

はじめに

今、世界各国で「次世代自動車の開発」を国家戦略の一つとして掲げ、積極的に取り組んでいる。日本も例外ではない。経済産業省は、自動車メーカーの電気自動車開発を支援しており、様々な政策と組み合わせて官民一体で自動車産業の次世代化を後押ししている。一方、国内の自動車メーカーも、環境規制強化に対応する次世代エコカーの開発や、自動運転をはじめとする業界新技術の研究を加速させており、大手7社の2016年度の研究開発費の合計は、過去最高の2兆8,000億円になる見通しだと報じられている。

日本の自動車産業は、長年にわたり、機械系技術者にとっての花形産業であった。[1)]しかし、現在開発が急ピッチで進められている「ADAS(エーダス:先進運転支援システム)」、「次世代車載ネットワーク」、「車載テレマティクス」といった先進技術領域は、これまで従事してきたエレクトロニクス/IT系の人材だけでは賄いきれない状況にある。そのため、自動車メーカーやサプライヤは、IT企業との新規協業を検討する必要が出てきている。

このような背景から、CACも自動車産業への参入余地は十分にあると考え、積極的に活動を行っている。今回は、「現在の動向」「業界特有の開発/管理プロセス」「エンジニア育成カリキュラム」「今後のトレンド」について詳しく述べていきたいと思う。

1. 自動車業界現在の動向

1.1 車載エレクトロニクス・ソフトウェアの割合が大幅に増加

昨今、自動車に対して「快適性」「安全性」「環境性」「信頼性」が高いレベルで求められるようになってきている。これらの研究開発が進むにつれ、自動車に搭載される機能は高度になり、車載システムの複雑さ、開発規模も大幅に増加している。現在では、高級車には一億行以上のソフトウェアコードが組み込まれていると言われている。これは、大手銀行の勘定系システムに匹敵する規模である。さらに、自動車の製造コストに占める半導体などの電子部品の割合は、2005年に20%に達し、2015年以降は40%と2倍増にまでなっている。

1.2 機能安全や業界標準への対応、技術者不足対策が急務

環境・安全・快適性への追求から機能の大規模化や複雑化が進む一方で、厳しい事業環境から、開発期間短縮の要求も高まっている。自動車メーカーやサプライヤは、ソフトウェア開発の効率化や信頼性の確保などのために「AUTOSAR(オートザー:車載ソフトウェアの標準規格。詳細は1.4参照)」の適用や、「Automotive SPICE(オートモーティブスパイス:車載ソフトウェア開発の業界標準プロセスモデル)」に基づくモデルベース開発、機能安全規格ISO26262の導入などにも柔軟に対応し、積極的に取り組む必要がある。

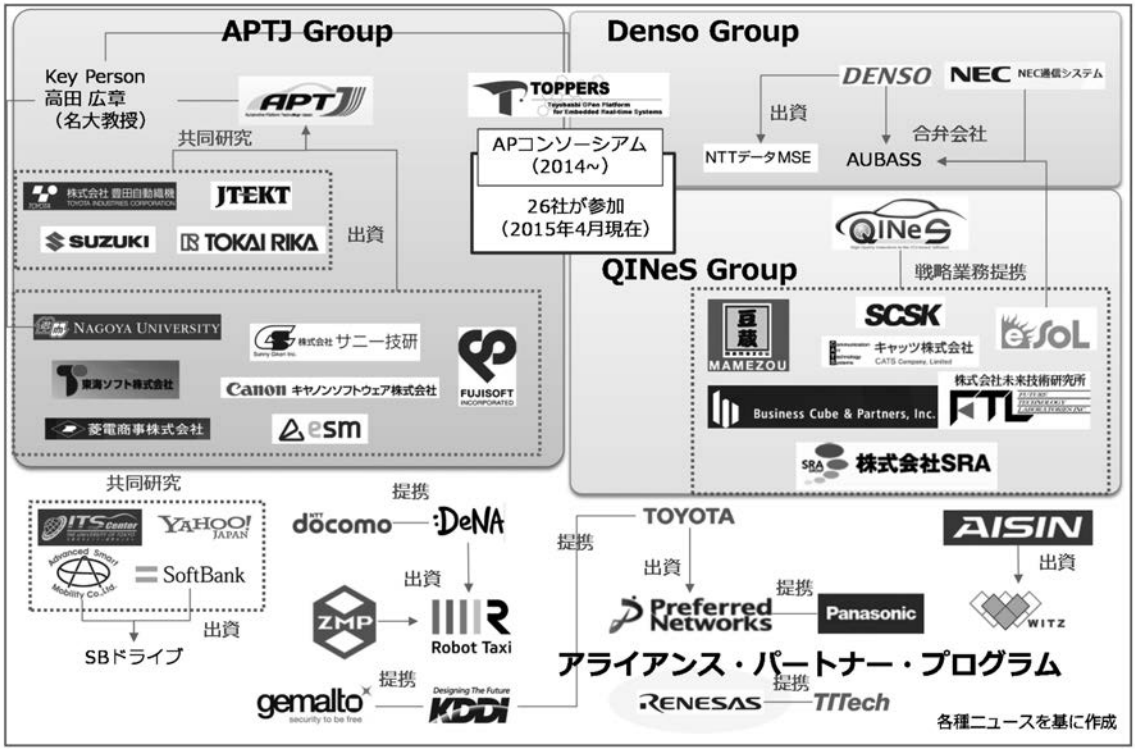
対処しなければいけない課題は山積みにもかかわらず、前述のとおり、先進技術領域を担う技術者、特に組込みソフトウェア技術者の不足は深刻なものとなっている

このように、車載ソフトウェアを取り巻く環境は大きな変革期を迎えており、市場競争力を確保していくために自動車業界・ソフトウェア開発技術者ともに大きな変化が求められている。

1.3 自動車メーカー/サプライヤ/IT企業が連携して協議を実施

自動車産業は、我が国経済におけるリーディング産業である。高い国際競争力を有し、貿易黒字の約5割を占める外貨の稼ぎ頭として、欧米競合他社にも立ち向かう必要がある。そのため、国内の自動車メーカー/サプライヤ/IT企業の3者が連携して、各社が競争すべき領域(アプリケーション)と、競争しなくてよい領域(プラットフォーム)を分け、非競争領域については共通化/標準化を進めようという戦略を立て、成果を

図1　APコンソーシアム参加企業とアライアンスパートナー関連図



上げ始めている。

例えば、APコンソーシアム（図1）は、車載制御システムのプラットフォームとなる「AUTOSAR　OS」が、欧州のソフトウェアベンダー2～3社で寡占提供されている現状に危機感を抱いて設立された。ここでは、AUTOSAR仕様をベースとして、高品質な車載制御システムプラットフォームの研究開発を行なっており、最終的には、開発したプラットフォームをグローバルトップ3のうちの1つとすることを目指している。APコンソーシアムには、大手自動車メーカーや車載機器メーカーをはじめ、国内企業28社が参加し、日本自動車産業の「ものづくり力」の維持・向上に乗り出している。[2)]

1.4 車載ソフトウェア標準規格『AUTOSAR』

車載部品の制御における電子化を測る指標として、自動車へ搭載されるECU[*1]の数が用いられる場合が多い。自動車へのECU搭載が始まった1980年代初頭では、1車種あたりのECU搭載数は1～数個程度だったものが、現在では、高級車であれば100個を超える。このような、自動車に搭載されるECUの数の増加に伴い、ECUに実装される車載ソフトウェアの規模も急激に肥大化しているのだが、これまでの車載ソフトウェアは、特定の車種、車載システムに最適化されていたため、新しい車種の開発や、既存車種でのハードウェア変更などによる、車載ソフトウェア開発・改変において、ソフトウェアの再利用が効率的に行われず、大規模なソフトウェア開発、既存ソフトウェアの改変作業が必要であった。

こうした背景を受け、肥大化する車載ソフトウェアの再利用性を向上することなどを求めて策定された標準規格がAUTOSAR（AUTomotive Open System ARchitecture）である。

（1）AUTOSARの誕生

前述のように、自動車業界では、制御システムの複雑化による、車載ソフトウェアの肥大化、開発・改変時の非効率が大きな課題であった。

そのような課題を解決するための新しい技術アプローチの必要性から、類似または同一のソフトウェアコンポーネントの継続的な繰り返し開発を止めること、さらに革新的な新機能の競争力のある開発の余地を残しつつ、基本機能上での協業の基盤を作る[*2]ことを目標として、2003年、欧州の自動車メーカー（BMW、ダイムラー・クライスラー[*3]など）や、電装部品メー

＊1）ECU：Electronic Control Unit（電子制御ユニット）、自動車などを制御するコンピュータ（ユニット）。マイクロプロセッサ、入出力機器などの周辺装置、通信モジュールにより構成されている。
＊2）AUTOSARコンソーシアム発足当初から掲げているコンセプトに「Cooperate on standards、compete on implementation」がある。これは、AUTOSARコンソーシアムに参加するパートナー企業は、AUTOSARプラットフォーム部分での「標準化では協調」して、制御アプリケーションでの「実装では競争」することを表している。
＊3）2007年のクライスラーとの関係解消により、現在はダイムラーAG。

カー（ボッシュ、コンチネンタルなど）が中心となって、AUTOSARコンソーシアムが設立され、そのコンソーシアムにより、標準規格となるAUTOSARが策定された。[3)]

つまり、AUTOSARとは、標準規格策定を目指して組織された団体であり、かつ、その策定された標準規格そのものである。

AUTOSARコンソーシアムは、「Core Partners」、「Premium Partners」、「Associate Partners」から成り（図2）、2017年１月現在、103の企業・団体で構成されている。[4)]

（2）AUTOSARの重要性

現在、国内の主要自動車メーカー・電装部品メーカーは、そのほとんどが、AUTOSARコンソーシアムに参画しており、AUTOSAR導入に乗り出している（図3）。

また、近年では、機能安全（2.2参照）などの安全基準に対応するために、AUTOSAR導入が国際的に進んでおり、近い将来、AUTOSARに準拠していない自動車は、海外での販売が難しくなることが予想される。

しかしながら、日本国内でのAUTOSAR対応は、欧州に比べ明らかに遅れており、国内での自動車市場において、AUTOSAR対応（普及）は、喫緊の課題と言える。

このため、国内（海外でも同様であるが）において、車載ソフトウェア領域への参入する企業にとっては、AUTOSARは必ず押さえておくべき知識・技術領域であると言える。

（3）レイヤードアーキテクチャ[5)]

AUTOSARのアーキテクチャは、「BSWレイヤー」、「RTEレイヤー」、「アプリケーションレイヤー」の3つのレイヤーから構成される、階層化されたソフトウェア・アーキテクチャであるレイヤードアーキテクチャをとっている（図4）。

図2　AUTOSARコンソーシアム参加メンバーの権利と義務

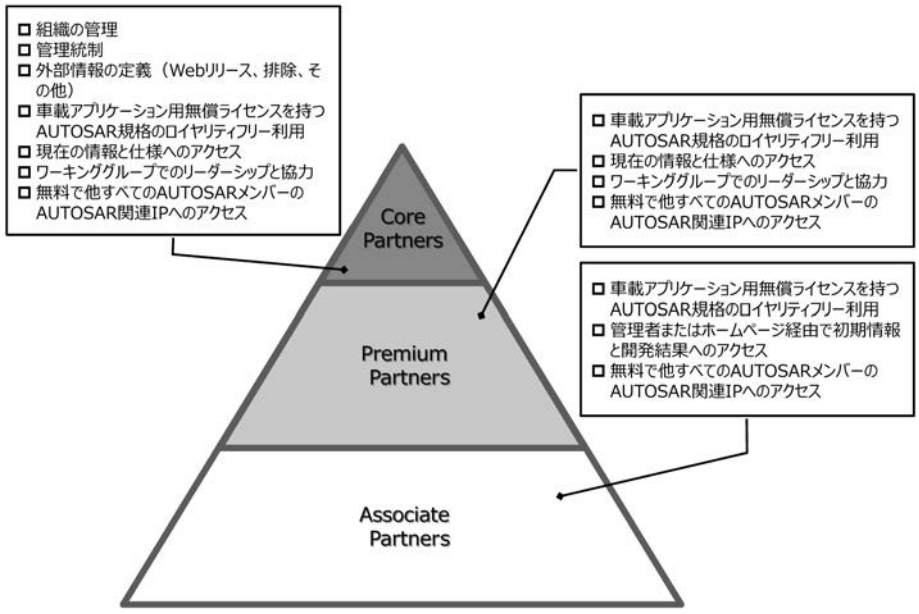


図3　AUTOSARコンソーシアム国内参加団体（2017年１月現在）

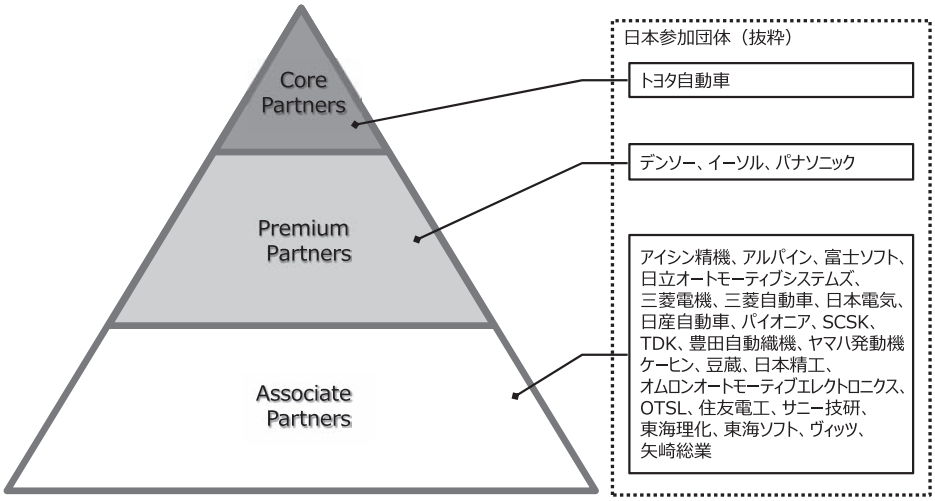
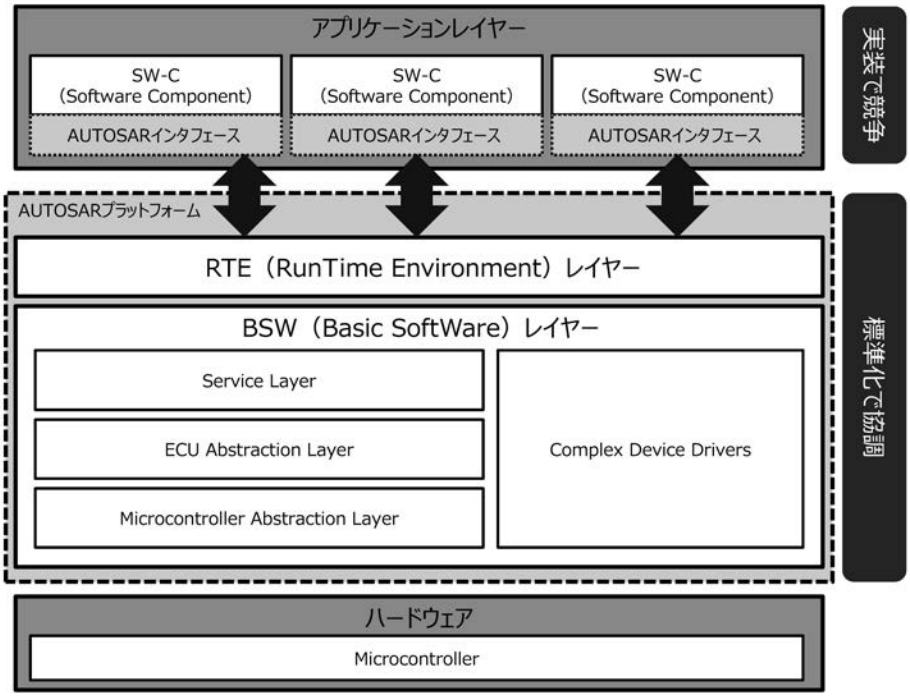


図4　AUTOSARレイヤードアーキテクチャ



・BSW（Basic SoftWare）

アプリケーションレイヤーに実装されるソフトウェアコンポーネントとハードウェアを繋ぐレイヤーで従来の、OS、ドライバ、ミドルウェアに相当する。

・RTE（RunTime Environment）

アプリケーションレイヤーに実装されるソフトウェアコンポーネント同士、ソフトウェアコンポーネントとBSWレイヤーを繋げるインタフェースを提供する。

これにより、アプリケーションレイヤーに実装されるセンサ、アクチュエータのアプリケーションであるソフトウェアコンポーネントは、AUTOSARインタフェースを介してRTEレイヤーと接続されることになり、ソフトウェアコンポーネントは、ハードウェアを意識することなく実装が可能となる。

また、BSWレイヤーは、さらに細分化され、OS、車両ネットワーク管理、メモリサービス、診断サービス、ECU状態管理の各機能を提供する「Service Layer」、周辺機器/デバイスへのアクセス、マイクロコントローラーへのインタフェースAPIを提供する「ECU Abstraction Layer」、マイコンへの直接アクセス機能を実装するレイヤーである「Complex Device Drivers」、マイクロコントローラー内蔵の周辺機器や、メモリにマッピングされた外部デバイスへ直接アクセスするソフトウェアモジュールを実装するレイヤーである「Microcontroller Abstraction Layer」から構成されている。

（4）AUTOSARの今後

AUTOSARは、2005年にR1.0が発表されて以降、ほぼ毎年リリースが続いており、2009年に、R4.0が発表されて以降は、R3.XシリーズとR4.XシリーズとにAUTOSARのバージョンが分岐している（次ページ図5、図6）[3)4)]。

2016年11月時点での最新バージョンは、R3.Xシリーズが R3.2.2であり、R4.XシリーズがR4.2.2である。R4.0以降、R4.Xシリーズには、機能安全ISO26262仕様（メモリ保護、時間保護等）が追加されている。

今後、R4.Xシリーズには、機能安全だけでなく、セキュリティ、高速化（より高速な車載ネットワーク仕様の取り込み）、エコ等に関する新たな仕様が追加され、2017年以降は、R4.Xシリーズが主流となり、現状ECUに搭載されているR3.XシリーズからR4.Xシリーズへのシフトが急速に加速していくと予想される。

2. 業界固有の開発/管理プロセス

本章では、CACがこれまで他業界向けに経験してきたものとは違った、自動車業界固有の開発/管理プロセスについて述べる。

2.1 Automotive SPICE（オートモーティブスパイス）

Automotive SPICE（以下、ASPICE:エースパイス）は、ISO/IEC15504（Software Process Improvement and Capability dEtermination）に基づいて策定された「車載システム開発」のためのプロセスモデルである。[6)]

ASPICEの各プロセスは、「プロセス目的」と「プロセス成果」、それを実現するための「基本プラクティス」と「作業成果物」で構成されている。

図5　AUTOSARリリース年表

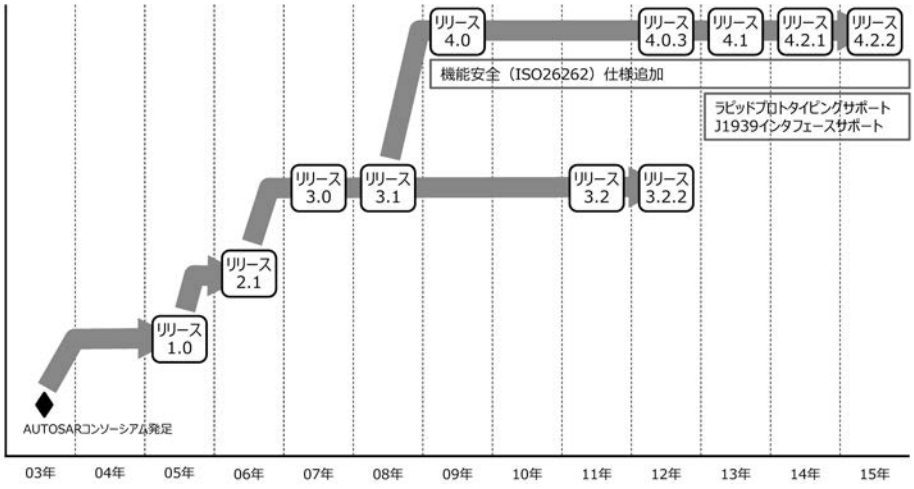


図6　AUTOSAR開発スケジュール

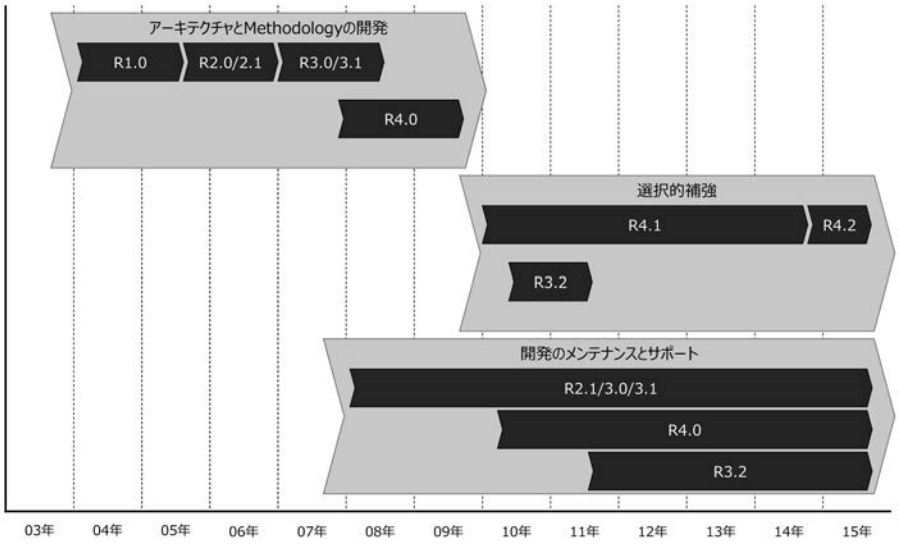


図7　ASPICE V3.0のプロセス参照モデル（PRM）

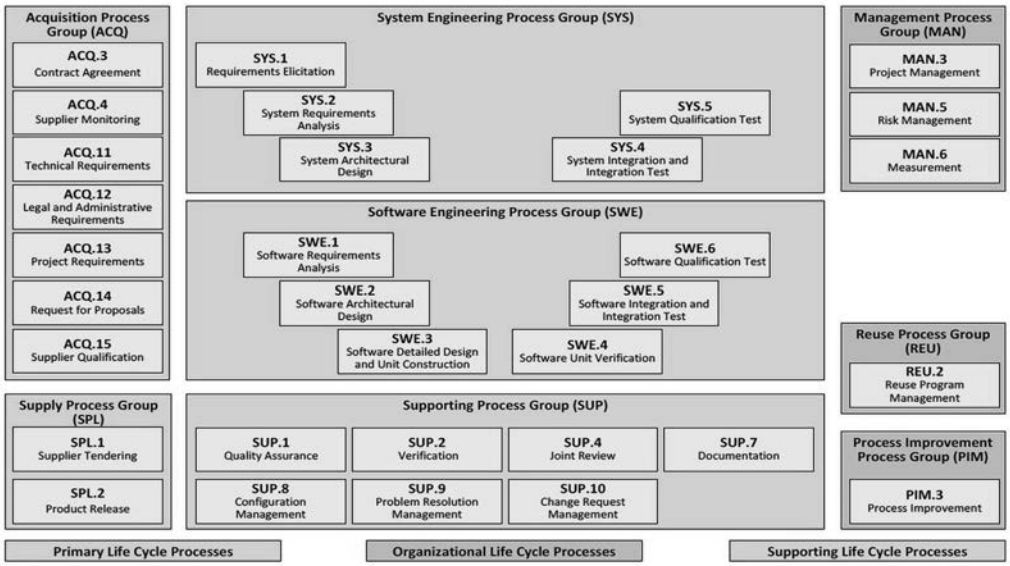
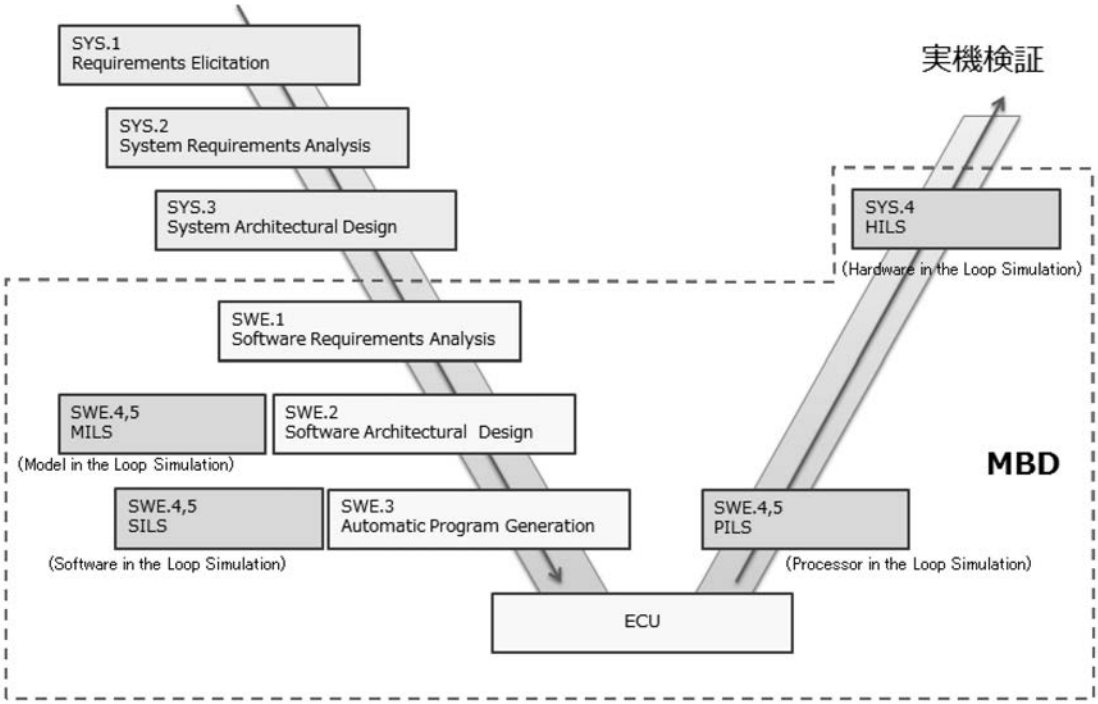


図8　MBDのプロセスイメージ



また、能力レベルの達成指標として「プロセス属性」と「達成成果」、「共通プラクティス」と「共通リソース」が定義されている。能力レベルはCMMIと同様に5段階あるが、ASPICEではプロセスごとに能力レベルがマッピングされるのが特徴となる。

当社が車載ソフトウェア開発を実施する上で必要なプロセス群は、ソフトウェアエンジニアリングプロセス群（SWE）、管理プロセス群（MAN）、支援プロセス群（SUP）、取得プロセス群（ACQ）の一部となる。勘違いしやすいのがシステムエンジニアプロセス群（SYS）である。ここでの「システム」とは、ハードウェア構成も含めたシステムを指す。

通常、このフェーズはTier1サプライヤが担当し、ソフトウェア開発者はSRS（Software requirement specification）をもとに開発する。

ASPICEのプロセス参照モデルは、標準的なスクラッチ開発がベースとなっている。

従って、後述するMBD（Model Based Development）を適用する場合、別途テーラリングが必要となる。

MBDのテストはモデルシミュレーションが基本となるため、ソース自動生成後にBack to Backの検証が求められる。

ASPICEのプロセスアセスメントモデルでは、設計要素とテスト仕様の一貫性を担保するため、双方向トレーサビリティを確立することが義務付けられている。

車載組込みソフトではシステム開発の規模が大きくかつ、複雑なコンポーネント構成になることが多いため、ツールを使用してトレーサビリティを担保するケースが一般的だ。

アセッサーによっては、ツール使用を必須にしたり、トレースする時間を測定するといった例もある。また、自動車機能安全規格（ISO26262）も考慮しなければならない。ISO26262の詳細は次項で述べるが、自動車機能安全規格では、ASPICEとは別に品質に関する規格が各プロセスで定義されている。

実際にアセスメントを受ける場合、Intacs™（International Assessor Certification Scheme）認定のアセッサーに依頼するが、特に注意したいのが自動車メーカーやTier1サプライヤから契約ごとにアセスメントを求められる場合である。

対象となるプロセスも各社ごとに異なり、アセッサーも先方が指定、それにかかる費用は基本ベンダー側が持つといったケースが多い。

つまり、1度取得すれば良いというわけではなく、顧客や製品カテゴリが変わる度にアセスメントを受けなければならないリスクが予想される。ちなみに、日本での認定アセッサーは30名程度とまだまだ少ない。

当社はASPICEにおける能力レベルは0からのスタートとなる。レベルを1段上げるのにかかる期間は最短でも1年と言われている。これは多くのドキュメントや標準化を作成する手間がかかるためである。また、レベル2以降から「管理のための監視」が必須になり、SEPGやSQAなど組織的な対応も求められる。今後のビジネス展開（当社の立ち位置）や開発手法など、様々な要因やリスクを考えた上で取り組まないと無駄な投資になりかねない。特に機能安全規格に準拠した開発ツールやテストツールの選定も重要となり、単にCMMIの焼き直しだけで済まないのは間違いない。

2.2 機能安全とは

機能安全とは、故障が発生してもシステムや機器の安全性を確保できる機能を実装しておくという考え方である。そうした安全に関係する事項を規格化したものが機能安全規格であり、2000年には国際規格としてIEC61508（電気・電子・プログラマブル電子安全関連系の機能安全）が策定されている。

機能安全は、しばしば本質安全との対比で説明される。本質安全とは、「機械が人間や環境に危害を及ぼす原因そのものを低減あるいは除去する」ことである。対して、機能安全とは、「機能的な工夫により安全機能を導入して許容できるレベルの安全を確保する」ことである。[8)]

これを立体交差と踏切の例に例えると、鉄道と道路の交差において、踏切を立体交差にして、踏切内で事故に遭遇する可能性を無くすことが本質安全であり、踏切に警報機や遮断機を設置して、完全な安全が保障されるわけではないが、許容できるレベルの安全を確保することが機能安全である。

現在の自動車には、多数の電気電子部品とそれらを制御するマイコンが搭載されており、それらの何らかの不具合による機能不全が、自動車、交通関係者（ドライバー、同乗者、歩行者等）に及ぼす危険性を安全機能の設置により許容可能なレベルに低減する考え方が自動車における機能安全である。[13)]

（1）ISO26262

機能安全規格は、「IEC61508」策定後、これをベースとして製品カテゴリごとに専用の規格が策定されてきた。そのうちの自動車分野向けがISO26262である。車載電気/電子（E/E）システム向けの機能安全規格であり、その対象は車両に搭載する電気電子機器だけでなく、ソフトウェアも含むコンピュータである。ISO26262は、Part1～Part10で構成されている（図9、表1）[9)]。

自動車のE/Eシステムの固有の必要性に準拠するように策定され、車両総重量が3,500kgまでの量産乗用車に適用される。各Partにおける規定のプロセスにより、システムを設計する必要がある。

（2）ASILの決定

ISO26262では、機能安全のレベルをASIL（Automotive Safety Integrity Level）と呼ばれる4段階の指標で分類する。ASILは、リスクの発生確率やドライバーの制御性、事故発生時の結果の重大性によりリスクを推定して決定する。また、ASILは、システムに採用されている技術に対するものではなく、ドライバーやその他の道路利用者に対する危険性を規定するものである。

図9　ISO26262全体構成図

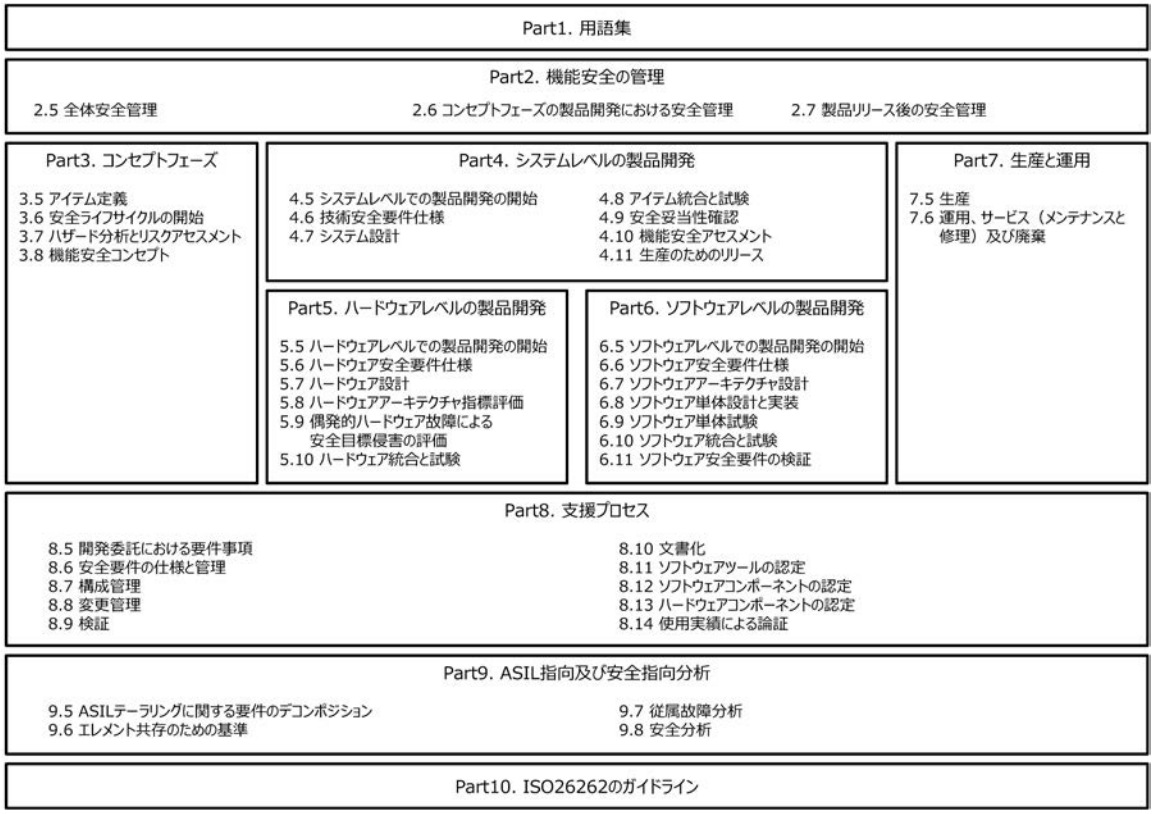


表1　ISO26262Part説明

構成Part		説明
Part 1	用語集	規格で使われる用語の定義。
Part 2	機能安全の管理	安全関連系の開発に関わる組織や人物が満たすべき要件を規定する。
Part 3	コンセプトフェーズ	ASIL決定の手続きとしてのアイテム定義、ハザード分析とリスクアセスメント、安全目標から機能安全コンセプトを作成する手続きを行う。
Part 4	システムレベルの製品開発	Part3で得られた機能安全コンセプトを技術安全コンセプトに詳細化し、システム設計に反映する。
Part 5	ハードウェアレベルの製品開発	確率論に基づくハードウェアの偶発的故障発生確率の計算、技術安全コンセプトに沿って開発する安全メカニズムの実装、および、その結果として得られる安全目標を侵害する確率の定量的評価に基づいた設計と支援を行う。
Part 6	ソフトウェアレベルの製品開発	V字プロセスに従って、技術安全コンセプトからのソフトウェア安全要件の導出、ソフトウェアアーキテクチャ設計からソフトウェア統合試験、ソフトウェア安全要件検証を行う。
Part 7	生産と運用	量産やサービス、市場の監視、廃棄するまでの安全要件を規定する。
Part 8	支援プロセス	サプライヤへの開発委託、支援系プロセス、及びソフトウェアツール、ソフトウェアコンポーネント、ハードウェアコンポーネント認定に関する要件を規定し、複数のプロセスに横断的に関与する。
Part 9	ASIL指向及び安全指向分析	ASILの取り扱いと技術的な分析手法の指針を規定し、Part8同様、複数のプロセスに横断的に関与する。
Part 10	ISO26262のガイドライン	Part1～9で記載困難な特定項目の解説や、事例を示したガイドライン。

ASILの定義

ASILはリスクアセスメントによる次の3つの指標（S、E、C）で決定する。

S：過酷度　（Severity）

E：ハザードの発生頻度　（probability of Exposure）

C：回避可能性　（Controllability）

過酷度（S）：

Severityは、「潜在的な危険な状況における1人以上の個人に対して発生する可能性のある危害程度の見積もり」と定義されている。つまりは、ドライバーまたは他の交通関係者が受ける障害の重さの見積りである。

危険事象のそれぞれに対しS0、S1、S2、S3の過酷度のクラスの1つを割り付ける（表2）。

ハザードの発生頻度（E）：

Exposureは、「分析対象の故障モードと組み合わさると、運用状況が危険になる可能性がある状態」と定義されている。想定される運転状況の特性に応じ、その期間、もしくは、その発生頻度のどちらかの指標による見積りである。危険事象のそれぞれに対しE0、E1、E2、E3、E4の発生頻度のクラスの1つを割り付ける（表3）。

回避可能性（C）：

Controllabilityは、「当事者のタイムリーな反応により、場合によっては外部方策の支援により、特定された危害または損害を回避する能力」と定義されている。潜在的リスクのあるドライバーまたは他の人物が、特定の危害を回避するために危険事情を十分に抑制することができる確率の見積りである。危険事情のそれぞれに対しC0、C1、C2、C3の回避可能性のクラスの1つを割り付ける（表4）。

ASILの決定：

前述の、過酷度（S）、ハザードの発生頻度（E）、回避可能性（C）それぞれの割り付けられたクラスより、ASIL（A、B、C、Dの4段階）を決定する。ASIL−Dは4段階のうち、安全性の重要なプロセスで最も厳しいテスト規定となる（表5）。

ISO26262では、上記により決定されたASILレベル（A～D）により、各種試験制約を設定している（表6）。

表2　過酷度のクラス

クラス	S0	S1	S2	S3
内容	障害なし	軽度及び中程度の障害	重度及び生命を脅かす障害（生存の可能性がある）	生命を脅かす障害（生存がはっきりしない）
Sポイント	0	1	2	3

表3　ハザードの発生頻度のクラス

クラス	E0	E1	E2	E3	E4
内容	可能性なし	可能性が非常に低い	可能性が低い	可能性が中程度	可能性が高い
Eポイント	0	1	2	3	4

表4　回避可能性のクラス

クラス	C0	C1	C2	C3
内容	一般的に回避可能	容易に回避可能	通常は回避可能	回避困難または回避不可
Cポイント	0	1	2	3

表5　ASILの決定

ポイント合計（Sポイント ＋ Eポイント ＋ Cポイント）				
10	9	8	7	6以下または E・C・S何れかが0
ASIL D	ASIL C	ASIL B	ASIL A	QM （通常の品質管理）

表6　ASILレベルに基づく試験制約

試験制約項目			ASIL			
			A	B	C	D
ソフトウェアユニット検証 ソフトウェア統合テスト		要件に基づく試験	++	++	++	++
		インタフェース試験	++	++	++	++
		不具合混入試験	+	+	+	++
		リソース試験	+	+	+	++
		バックトゥバック試験	+	+	++	++
試験ケースの導出方法		要件分析	++	++	++	++
		同値クラスの作成と分析	+	++	++	++
		境界値分析	+	++	++	++
		エラー予測	+	+	+	+
100% 網羅率（カバレッジ）	ソフトウェアユニット検証	C0カバレッジ （命令網羅率）	++	++	+	+
		C1カバレッジ （分岐網羅率）	+	++	++	++
		MC/DC(*) （Modified Condition/Decision Coverage）	+	+	+	++
	ソフトウェア統合テスト	機能カバレッジ	+	+	++	++
		コールカバレッジ	+	+	++	++

＋：記載された手法を推奨する
＋＋：記載された手法を強く推奨する

表6（＊）　MC/DC詳細

Modified Condition/Decision Coverage（MC/DC）カバレッジ

MC/DCは、国際技術標準 DO-178B(RTCA)に準拠している。
これは、ソフトウェア証明のための基準を指定したものであり、対象となるソフトウェアには、
航空業界で使用される重要機器およびシステムのためのリアルタイム組込みシステムも含まれる。
DO-178Bに従うと、完全な(100%の)MC/DCカバレッジを得るには、次の3つの条件を満たす必要がある。
a. 各「判断文」が、少なくとも1回すべての可能な結果を得ている。
b. 1つの「判断文」中の各条件が、少なくとも1回すべての可能な結果を得ている。
c. 1つの「判断文」中の個々の条件が、単独で全体の「判断文」の結果を左右する。

例:

```
if(     ( ON == bit_SW_UP )
    && ( OFF == uint8_SW_UP_o )
    && ( UINT8_PWR_MAX > uint8_led_pwr ) ){
      uint8_led_pwr = uint8_led_pwr + UINT8_PWR_STEP ;
}
```

上記例のC1/C2およびMC/DCのカバレッジ100%全網羅試験入力は以下通り:

条件①	条件②	条件③	必須テストパタン		
(ON == bit_SW_UP)	(OFF == uint8_SW_UP_o)	(UINT8_PWR_MAX > uint8_led_pwr)	C1	C2	MC/DC
T	T	T	○	○	○
F	T	T	任意一つ	○	○
T	F	T		○	○
T	T	F		○	○
T	F	F		○	—
F	T	F		○	—
F	F	T		○	—
F	F	F		○	—

(3) ソフトウェア開発と機能安全

ある組織が機能安全を考慮した開発を行う能力があること、あるいは考慮して開発したことを示すためには、その根拠を論理的に説明できる必要がある。ISO26262でも、安全要件のトレーサビリティを求めている。そのため、要件ごとに固有の番号を付し、紐付けして管理する。細かい粒度での管理が必要であり、ソースコードでは関数単位相当まで実施する。このようにして、設定した安全目標が確実に実装もしくは試験されたかどうかを開発工程の全体で追跡可能にすることが求められている。トレーサビリティが適切に管理されると、上位要件に対する下位要件の抜け漏れ、関数の不具合による要件への影響などの効率的な発見と対処が可能になる。[7)]

2.3 モデルベース開発(MBD : Model-Based Development)[10)]

モデルベース開発(MBD)は、単に「モデルを用いた開発」ではない。自動車業界におけるMBDは、「制御ソフトおよび制御対象を、シミュレーション可能な数学モデルにより表現し、制御ソフトの開発・検証を行う手法」である。

モデルには、開発対象であるコントローラーと、コントローラーの制御対象(プラントと呼ぶ)の2つがある。制御ソフト開発において、プラントモデルが必要となる理由は、クローズドループという特性にある。常に制御対象からのフィードバック情報を入力・計算して次の結果を出力する必要があるためだ(図10)。

「シミュレーション可能な数学モデル」とは、制御ロジックをソースコードより抽象度の高いモデル(ブロック線図)で記述し、モデルの状態で実行(シミュレーション)し、制御ロジックの検証を可能とすることを指す(図11)。

自動車業界におけるMBDの導入はヨーロッパが先行しているが、日本の自動車メーカーにも普及しており、日産自動車では、エンジン制御ソフトの全ソフト開発にMBDを採用していると言われている。また、車両1台分の機器をシミュレートするHILシミュレータ(後述)の構築も進んでおり、デンソーやアイシンといったTier1サプライヤにとっては、MBDへの対応は必須である。特に、ADAS(先進運転支援システム)、自動運転の領域では、安全性を検証するためにあらゆる状況を想定したテストが必要であり、天候等の外部状況と車両の状態を組合せた膨大なケースの検証をする上で、MBDは必須の技術・手法となっている(図12)。

MBDの実現には、モデルを「実行」、即ち、ブロック線図による数学モデルを数値シミュレーションできるツールの存在が不可欠である。自動車業界では、MBDのデファクトツールとしてMathWorks社のMATLAB/Simulinkを使用している。

図10　2つのモデルと2種類の制御方式

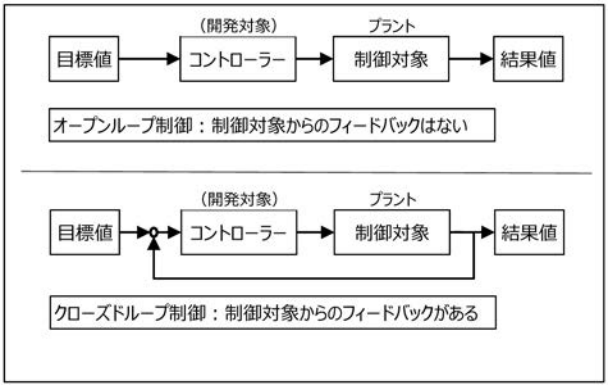


図11　MBDにおけるモデル(ブロック線図)

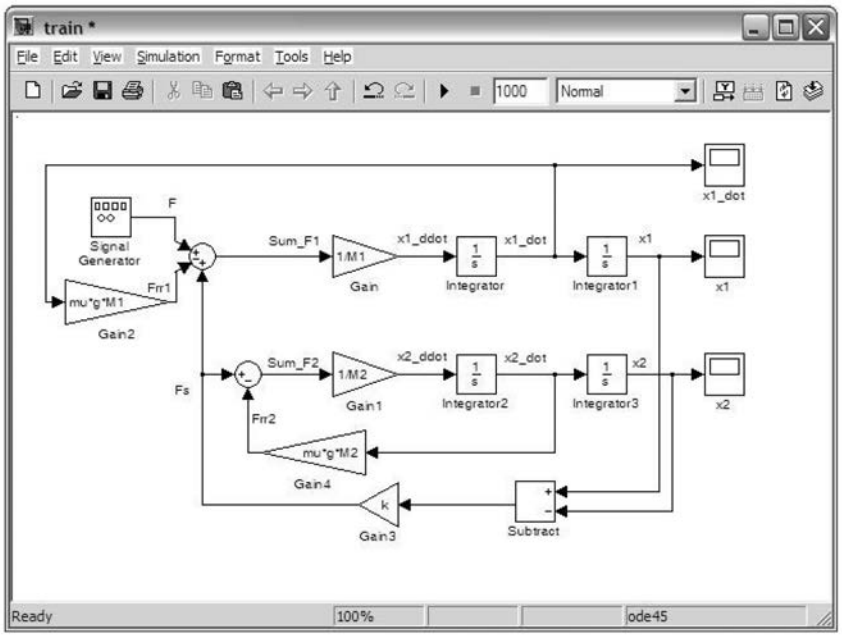


図12　従来プロセスとMBDの開発プロセスの違い

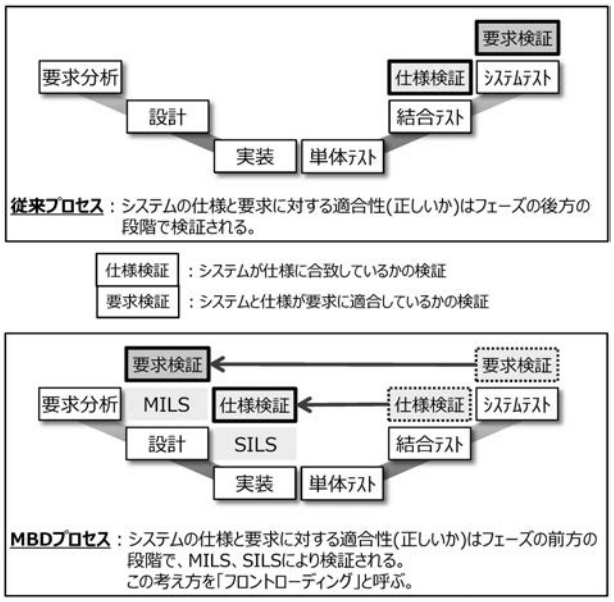
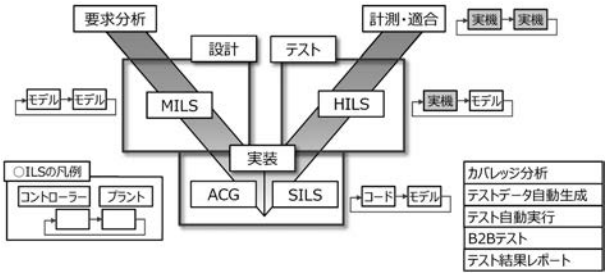


図13　MBDの開発プロセスと手法・技術



ちなみに、MATLAB/Simulinkは、自動車産業のみならず多くの産業界で非常に高いシェアを持ち、100万人以上の技術者に利用されている。

モデリングツールを前提とした、MBDのプロセスを構成する手法、技術には以下のものがある（図13）。

MILS：Model In the Loop Simulationの略。コントローラーもプラントもモデルの状態でシミュレーション（テスト）を行う

ACG：Auto Code Generationの略。モデルからCソースコードを自動生成する。モデルを正確にCで表現することに加えて、ISO26262機能安全が推奨するMISRA−C*4のコーディング規約に則ったコードを生成する

SILS：Software In the Loop Simulationの略。コントローラーはCソースコードを使用し、プラント側はモデルの状態でシミュレーションを行なう

B2Bテスト：Back to Back testの略。MILSのテスト結果とSILSのテスト結果を突き合わせ同じ結果となることを確認することにより、ACGによるソースの正しさ、ACGの正しさを検証する。ISO26262でも実施が要求されている

HILS：Hardware In the Loop Simulationの略。コントローラーはコードをロードしたECU（電子制御ユニット。「実機」と表現）を使用し、プラント側はモデルの状態でシミュレーションを行う。ただし、プラントモデルはリアルタイム（実時間）で動作させる必要があり、HILS専用機器を使用する。

MBDの数学モデルは「動く仕様書」と呼ばれている。MBDは動く仕様書であるモデルを設計フェーズからテストフェーズまで徹底的に利用することにより、開発効率と品質を劇的に高める手法と言える。

これまでCACが携わってきた業務アプリは勘定系が主だったため、利用する数式も、貸借平均の原則など四則演算の結果がイコールであるというものだった。しかし、制御アプリで扱う式の基本は、ニュートンの運動方程式:F（力）=m（質量）×a（加速度）となり、加速度は速度の微分、速度は距離の微分となり、微積分が入ってくる。

MATLAB/Simulinkには数学モデルを構成する要素（ブロック）を、「加算器」、「積分器」などと呼ばれる制御アルゴリズムを表現する上で必要な「演算器」をライブラリとして多数持っており、これらの「演算器」を配置して制御ロジック（モデル）を作成することができ、微分方程式を数値計算（シミュレーション）する「ソルバー」にてシミュレーション実行し、検証することができる（図14）。当然のことながら、モデリングには、制御工学（機械工学、数学）の知識が要求される。

3. トレーニング

現在CACでは、社内トレーニングおよび社外研修を活用してすべてのカリキュラムを習得した要員を「MBD開発者」として当カンパニー内で技術者認定を行っている。

3.1 スキルマップ

開発技術も従来型と異なっている。開発言語は、組込みC、C++、PERLとなるが、求められるものはプログラミングスキルだけではない。業務システムの知識も当然必要だが、機械制御の基礎知識（理工系大学の数学、一般工学知識）も車載システム開発には必要となってくる（図15）。

以下は、このような基礎知識に加えて、車載システム開発に必要なスキルを開発フェーズに沿って配置した図である（図16）。数学・工学系の基礎知識、AUTOSARなどの開発規格の知識、マイクロコントローラー（以下、マイコン）とそれを制御するためのプログラミングスキルで構成されている。次項では、これらの技術を習得するためのカリキュラムを紹介する。

3.2 育成カリキュラム

育成カリキュラムは、下記3ステップに分かれている（表7）。

現在、当カンパニーの担当チーム初期メンバーがStep1の

＊4）英国自動車産業ソフトウェア信頼性協会（Motor Industry Software Reliability Association）が発行した組込みCプログラミング標準

履修を完了し、組込み開発エンジニアとなっている。次項では、Step1の組込み開発エンジニア育成カリキュラムを例に、実際の研修内容を紹介する。

図14　Simulinkによるモデル作成とシミュレーション

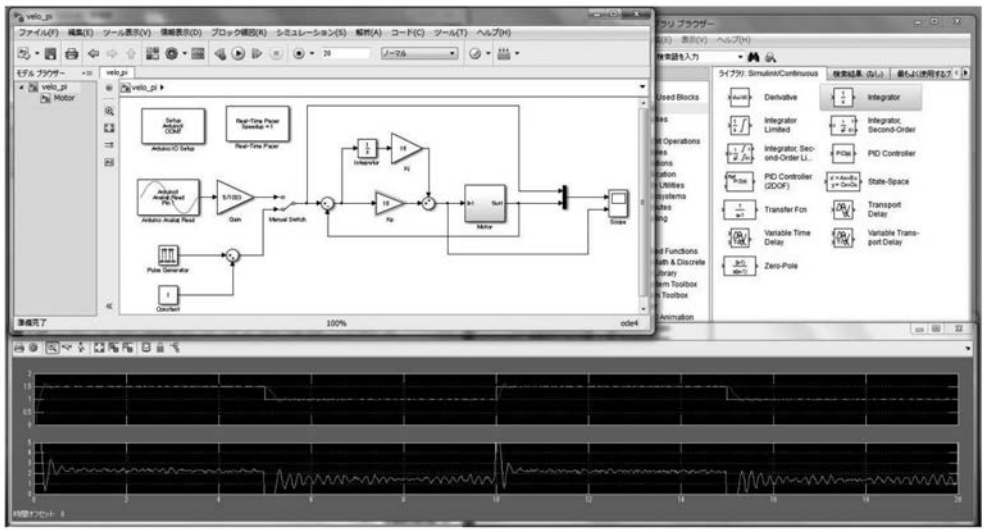


図15　自動車制御システムにおけるデータの流れ

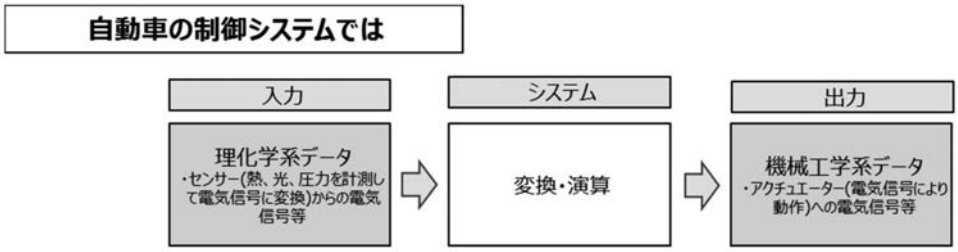
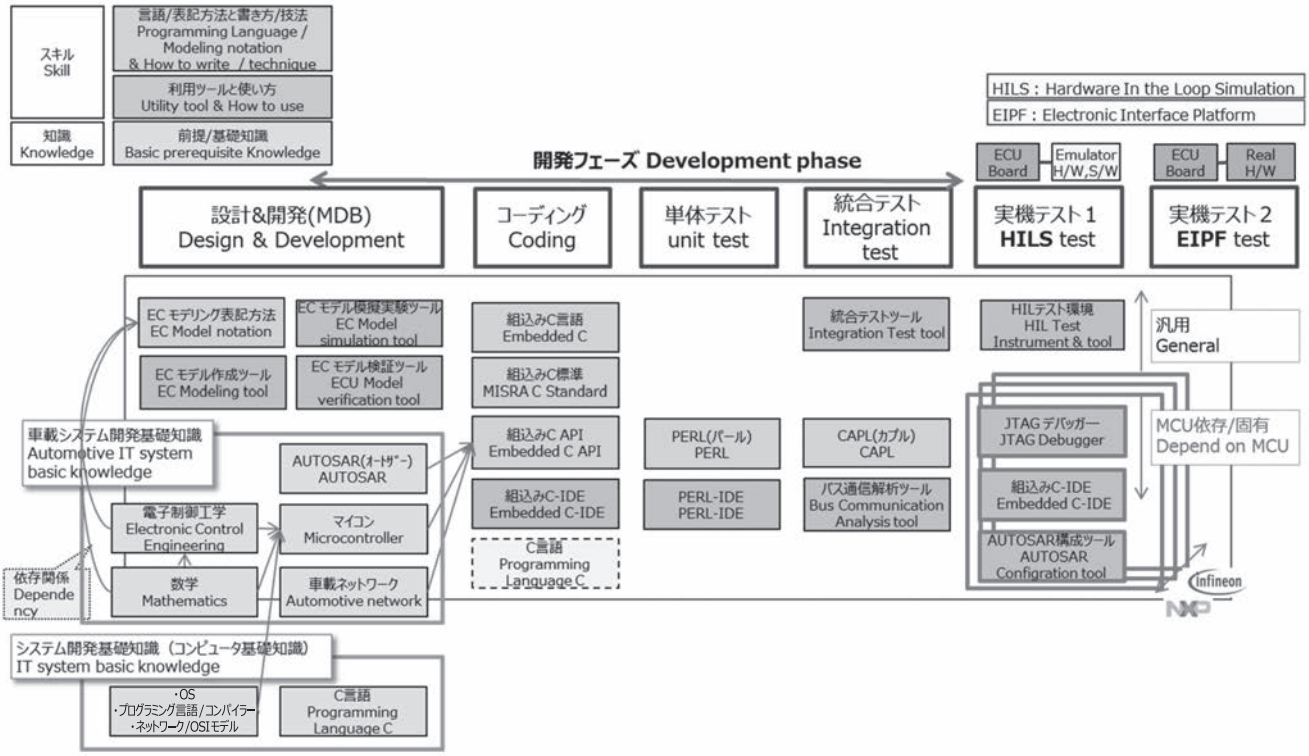


図16　車載システム開発フェーズ別スキル配置図



3. 3 トレーニング内容

Step1の組込み開発エンジニアを育成するための社内トレーニングは全11工程で構成されている（表8）。

マイコンは、ハードウェアの制御を行っており、ボタンやキーからの入力を受け取り、あらかじめ設定されたプログラムに基づいてモータやLCDなどの周辺回路を制御する。このマイコンの構成（図17）を理解し[11]、設計仕様に基づき制御するためのプログラミングをトレーニングする。これを習得することにより、開発エンジニアとして活躍することができる。

マイコンの機能図（Block Diagram）[12]とトレーニングカリキュラムのマッピングは図18を参照してもらいたい。

実際のトレーニングでは、写真1にあるとおり実習用の入出力装置（基盤）とECUを作成し、入出力装置を制御するためのプログラミング（図19）を記載し、入出力装置の制御方法と、組込みC言語を習得していくことになる。

表7　育成カリキュラム

段階	呼称	説明	担当フェーズ
Step1	組込み開発エンジニア	組込みＣプログラマー	コーディング・単体テスト
Step2	MBD エンジニア	モデリング技法の習得、モデルベースでの設計・開発者	設計、開発 モデリング、モデル検証
Step3	テストエンジニア	HILS テストエンジニア	統合テスト、HILS テスト

表8　Step1習得カリキュラム

項番	タイトル	説明
1	Introduction	組み込みシステムの核となるマイコンの基礎知識の習得
2	Port Register	CPU と外部のセンサーやスイッチなどの信号のやり取りをするための周辺機能
3	Timer – Interruption	Timer とは、時間が経過したことを知らせる装置のことで、経過時間を測る以外にも指定した周期のパルスを出力したり、入力パルスの周期を測ったりすることにも使用する
4	Analog Digital Converter (ADC)	ADC とは、アナログ信号をデジタル信号に変換するために用いる電子回路である
5	Universal Synchronous Asynchronous Receiver Transmitter (USART)	USART とは、シリアル信号とパラレル信号を互いに変換することができる回路である
6	Controller Area Network (CAN)	CAN とは、耐ノイズ性の強化を考慮して設計され、相互接続された機器間のデータ転送に使われる規格である
7	Sensor	ある対象の状態を監視したり検知や計測などを行うデバイスや機器のこと。物体検知、温度、音、光、磁気、熱、振動、速度、圧力など検出対象によって種類も多数ある
8	Serial Peripheral Interface (SPI)	SPI とは、マスタとスレーブ間における非同期（全二重／半二重）または同期シリアル通信を提供する。基本的にオンボードにおける通信方式で、物理層付近を規定した仕様である
9	Watchdog Timer	ある一定時間ごとに、相互で何らかの通信を取り合うようにつくられたもの。この通信が途切れた時点で装置が故障したりハングアップ等の不正動作が発生したことがわかる
10	Pulse Width Modulation (PWM)	PWM とは、半導体を使った電力を制御する方針の１つである。オンとオフの繰り返しスイッチングを行い出力される電力を制御する
11	Input Capture Unit (ICU)	ICU とは、外部から入力された信号の立ち上がり、立ち下がりまたは両方のいずれかを検知すると、カウンタの値をCapture Compare Register：CCRへ取り込む機能のこと

図17　マイコン構成図

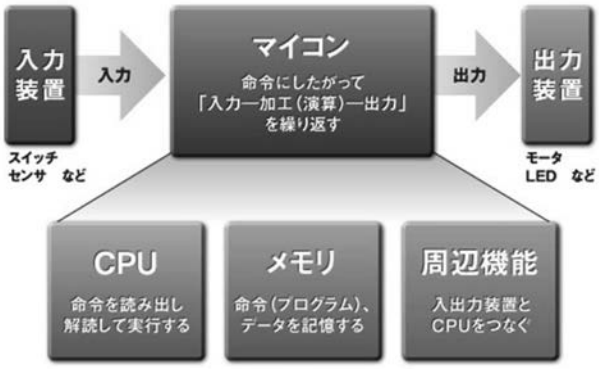


写真1　実習用の入出力装置基盤

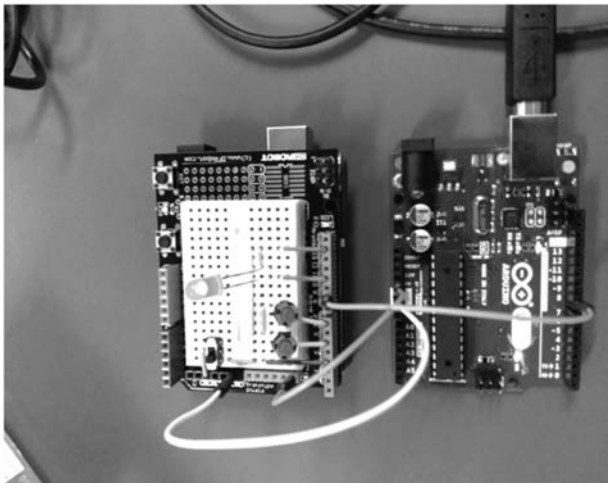


図18　マイコンの機能図（Block Diagram）とトレーニングカリキュラムのマッピング

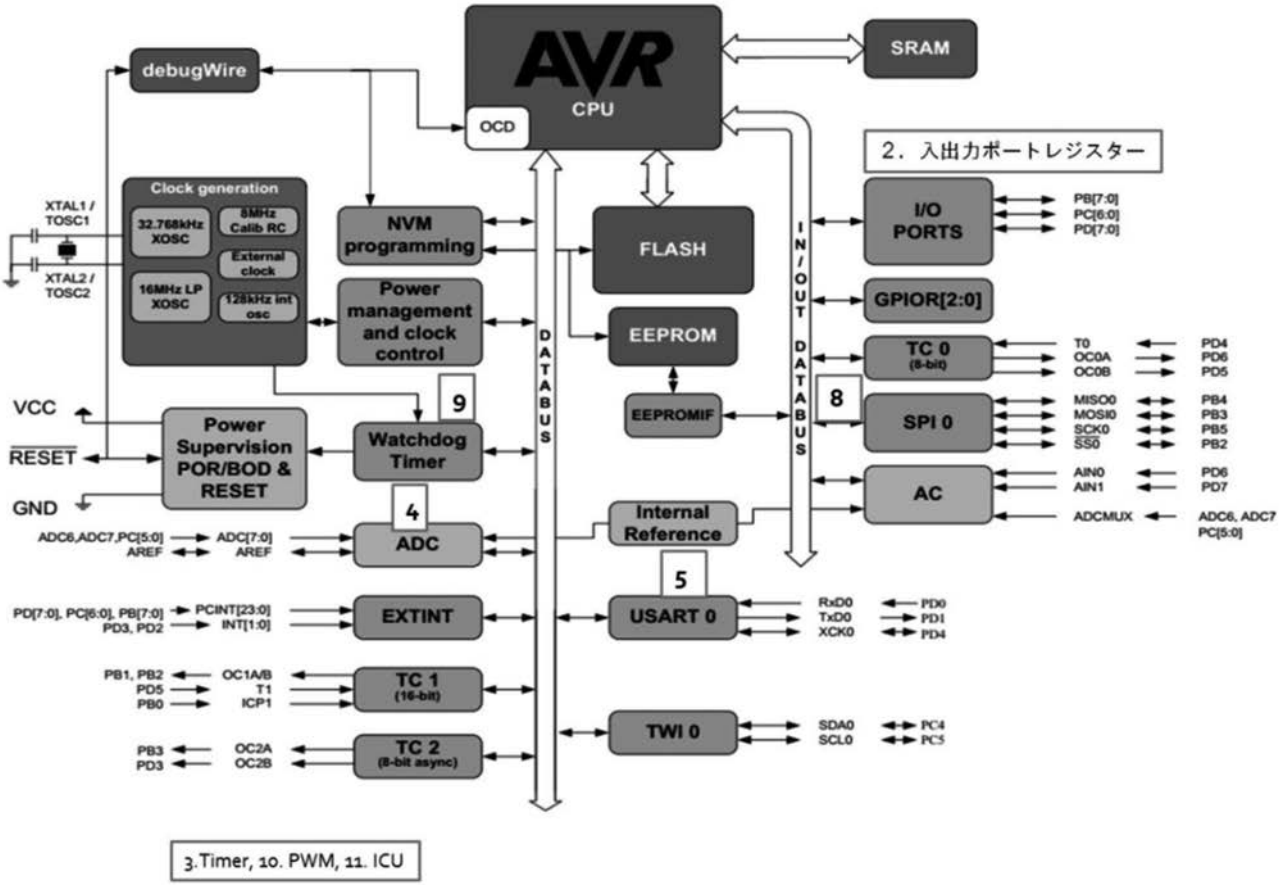
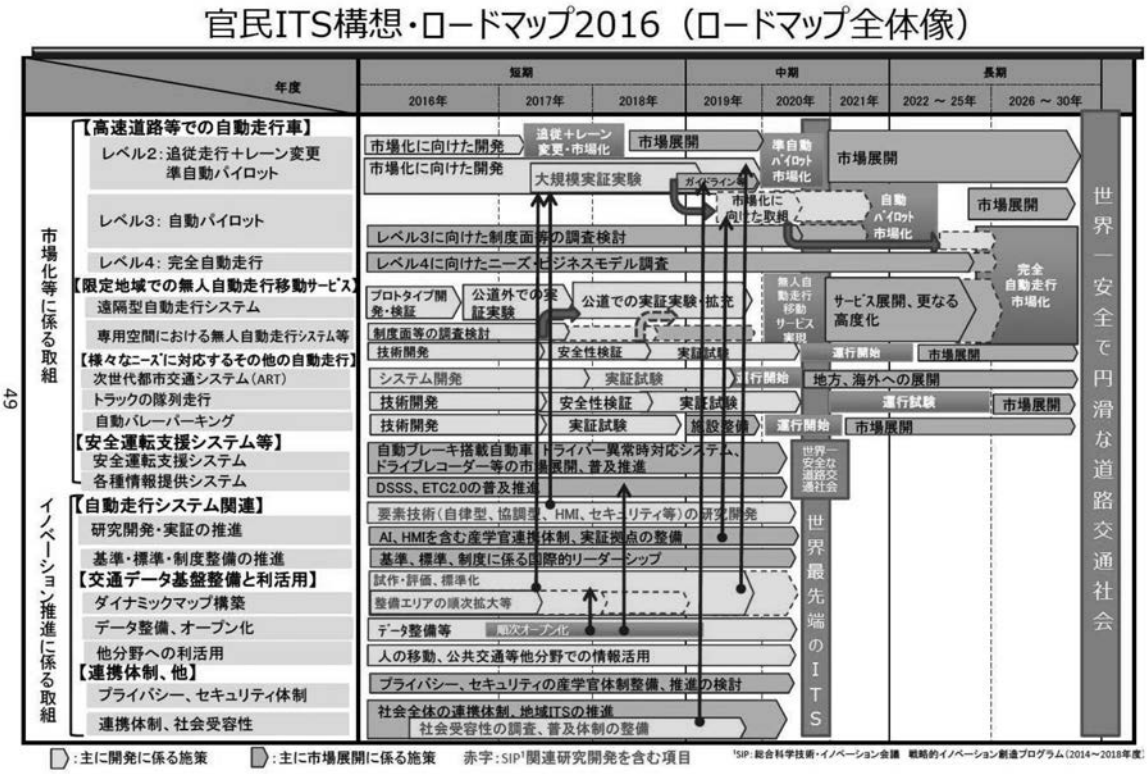


図19　入出力装置を制御するためのプログラミング例

```
//----------------------ADC REGISTER --------------------------
void adc_init(void){
    ADCSRA |= ((1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0));   //prescaler = 128, ADC_ref_clock = 16Mhz/128 = 125Khz
    ADMUX |= (1<<REFS0);                            // voltage reference from Avcc (5v)
    ADCSRA |= (1<<ADEN);                            // turn on ADC
    ADCSRA |= (1<<ADSC);                            // do an initial conversion

    /*
    ADMUX   :   ADC Multiplexer Selection Register
    ADCSRA  :   ADC Control and Status Register A
    ADCL/H  :   ADC Data Register Low/High
    */

}

uint16_t adc_read(uint8_t channel){
    ADMUX &= 0xF0;                      // clear the older channel that was read
    ADMUX |= channel;                   // define the new ADC channel to be read
    ADCSRA |= (1<<ADSC);                // start a new conversion
    while(ADCSRA & (1<<ADSC));          // wait until the conversion is done (ADSC becomes 0 again)
    return ADCW;                        // return the ADC value of the chosen channel
}

//---------------------USART REGISTER-------------------------
void usart_init(void)
{
    /*Set baud rate */
    UBRR0H = (uint8_t)(BAUD_PRESCALLER>>8);
    UBRR0L = (uint8_t)(BAUD_PRESCALLER);

    UCSR0B = (1<<TXEN0);                // enable transmitter (0x08)
    UCSR0C = (3<<UCSZ00);               // set character size 8 bits (0x06)
}

void usart_send(unsigned char data){
    while(!(UCSR0A & (1<<UDRE0)));      // wait for empty transmit buffer
    UDR0 = data;                        // put data into buffer, sends the data
}

void usart_putstring(char* StringPtr){
    while(*StringPtr != 0x00)           // check if there is still more chars to send
    {
        usart_send(*StringPtr);         // send one char at a time
        StringPtr++;                    // increase the pointer to read the next char
    }
}
```

表9　官民ITS構想・ロードマップ2016（ロードマップ全体像）



おわりに

ここまで紹介してきた車載ソフトウェア開発のトレンドは欧州の自動車メーカー・部品メーカー・ソフトウェアベンダーが中心となって作り出したものであり、現状では日本勢は優れたモノ造り品質力をもちながらもグローバル競争のなかで欧州発の標準に追従せざるを得ない状況となっている。

今後、自動走行システムのイノベーションの進展に伴い、国際的なルール策定の動きは自動車単体から道路交通システムや自動車データ関連ビジネスへと広がっていくと考えられる。日本においても、「2020年までに世界一安全な道路交通社会」を構築、2030年までに「世界一安全で円滑な道路交通社会」を構築・維持することを目指すとの政策目標が立てられ、制度設計においても国際的な連携をしつつも日本が世界をリードするというスタンスで取り組むこととされている(官民ITS構想・ロードマップ2016)。

このような標準化競争がある中で、欧州・米国メーカーのオフショア開発を請け負う当社グループのAccel Frontline Limited(インド)とCAC(日本)が連携を強めることで、日米欧三極の車載ソフトウェア開発技術の最新情報・経験を共有し柔軟な対応がとれる体制を構築する。それは必ずお客様の課題を解決するグローバル・将来視点での技術提供につながるはずである。

参考文献

1)「グローバル化の新潮流　自動車産業における環境変化と経営課題」、P51、知的資産創造2015年4月号、野村総合研究所、2015年

2)「APコンソーシアム～紹介と参加のお誘い　2014年3月(改訂版)」、P5～6、名古屋大学大学院情報科学研究科附属組込みシステム研究センター
<https://www.nces.is.nagoya-u.ac.jp/press/131120_ap-conso-intro.pdf>

3)「AUTOSAR入門」、VECTOR　Eラーニング、Vector Japan
<https://elearning.vector.com/vl_autosar_introduction_jp.html>

4) AUTOSAR(AUTOSAR公式サイト)
<https://www.autosar.org/>

5)「AUTOSARで変わる車載ソフトウェア開発」、EDN Japan
<http://ednjapan.com/edn/articles/0910/01/news123.html>

6)「SPICE3.0プロセス基礎トレーニング資料」、Business Cube & Partners

7) 茂野一彦、「自動車用機能安全規格ISO26262の紹介」、MSS技報Vol.23、三菱スペース・ソフトウエア、2013年
<http://www.mss.co.jp/technology/report/pdf/23-05.pdf>

8) 安倍秀二、「自動車の機能安全と部品安全～ISO26262の概要～」、一般社団法人電子情報技術産業協会(JEITA)、2015年
<http://www.jeita.or.jp/japanese/exhibit/2015/1111/pdf/02_Functional.pdf>

9)「ISO26262機能安全規格とは?」、NATIONAL　INSTRUMENTS
<http://www.ni.com/white-paper/13647/ja/>

10) dSPACE　Japan監修、「モデルベース開発−モデリング、プラント・モデル、コントロール・モデル」、日経BP社、2013年

11)「マイコン入門(1)マイコンの基本構成、動作」、ルネサスエレクトロニクス
<https://www.renesas.com/ja-jp/support/technical-resources/engineer-school/mcu-01-basic-structure-operation.html>

12) ATmega328/P設計書、P13、Atmel Corporation

13)「機能安全(ISO26262)」、一般財団法人日本自動車研究所
<http://www.jari.or.jp/tabid/112/Default.aspx>