組込みソフトウェア開発をめぐる新潮流
～IoT時代に求められるサービス・トランスフォーメーション～

SECシステムグループリーダー　山下　博之

社会生活や経済活動への情報通信技術（ICT）の活用が進むに伴い、組込みソフトウェアに対しても大規模・高機能化が求められるようになり、半導体技術の著しい進展がその実現を後押ししている。一方、組込みソフトウェアの大規模・高機能化が進むと、その開発効率の向上が望まれるようになり、開発環境や開発技術などが発展してきている。これらは、更に組込みソフトウェアの大規模・高機能化をドライブしている。IoT時代を迎え、組込みシステムはエンタプライズ系システムと連携し、多くの分野でサービス指向がより強くなりつつある。この状況に対応するために、組込みソフトウェアの開発スタイルの見直し、開発組織のスキルと体制の強化が求められる。本稿では、IPA/SECの調査・検討の成果や内外の公開調査結果などを通して、組込みソフトウェアの現状とIoT時代に向けた課題を概観する。

1　組込みシステム開発の変遷[1]

世界最初のコンピュータENIAC（Electronic Numerical Integrator and Computer）が開発された1946年から下ること25年、インテル社からマイクロコンピュータシステムMCS-4が1971年に発売された。このマイコンは、日本のビジコン社の依頼で開発され、プリンタ付き電卓に使われたが、ENIACに匹敵する四則演算能力を備えると共に、外部機器制御をソフトウェアによって行うものであった。これは、組込み機器の原形の一つと言えよう。

MCS-4の核となる４ビットマイクロプロセッサ4004が登場して以降、小型・低価格という特徴とプログラム制御の柔軟性から、様々な機器・システムにマイクロプロセッサが適用されていった。それに伴い、性能や機能を高めたマイクロプロセッサも次々と開発された。現在では、家電、自動車・車載機器、携帯電話/スマートフォン、テレビ、AV機器など、ほとんどの電気機器・システムにプロセッサが組み込まれ、それらの上では組込みソフトウェアが動作している。近いうちに、最近注目されている人工知能も組込みAIとして様々なシステムに搭載されることは、ほぼ間違いないであろう。

このような組込みシステムの発展過程で、組込みソフトウェアを取り巻く環境は、エンタプライズ系（情報処理システム向け）ソフトウェア開発の技術を取り込みながら進化してきた。例えば、当初はアセンブラによる開発であったが、リアルタイム制御を特徴の一つとする組込みシステム向けのリアルタイムOSとして、1979年にOS-9が、1980年にVxWorksがそれぞれ発表された。その後、1987年にITRON仕様が公開され、1998年には汎用PCなどに使われていたLinuxをベースとした高機能OSが組込みシステムでも使われ始め、2007年にはAndroidが発表されている。組込みソフトウェアの開発環境では、Embedded Javaの仕様が1997年にSun Microsystems社から発表されている。プログラミング言語関連では、1998年にC言語の標準コーディングガイドラインであるMISRA-C※1が発表されている。また、車載情報端末向けのソフトウェアプラットフォームAuto PC 1.0が、1998年にマイクロソフトから提供開始された。更に、2000年には、組込み用データベースSQLite Version 1.0がパブリックドメインソフトウェアとして公開されている。

このようなシーズとしての情報通信技術（ICT）と、ICTを活用した製品やサービス（ニーズ）とは、相互に好循環の影響を及ぼし合いながら速いペースで進展してきた。例えば、ｉモード携帯電話の登場によりモバイル・インターネットが普及し、コンテンツ配信が急速に発展した。それにより一層高まったサービス・ニーズに応える形でモバイル端末がスマートフォンやタブレットに進化すると、その高い機能・性能を活用した現地決済やオンラインゲームなどの高度なサービスが次々と生まれている、といった具合である。別の例では、自動車に通信機能が搭載されるようになると、自動車に対する様々な情報提

【脚注】

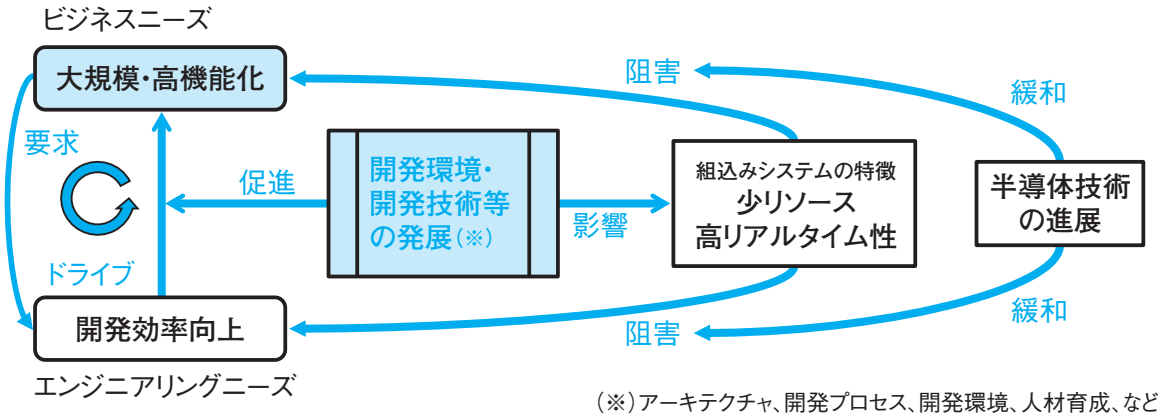
※1　欧州の自動車関連ソフトウェア業界団体MISRA（The Motor Industry Software Reliability Association）が策定。

供サービスが生まれたが、これが自動車からの情報収集のニーズを喚起し、双方向の通信機能が具備されるようになった。その結果、気象や渋滞などの自動車から発信される様々な広域の環境データを集約して提供するというサービスに発展した。

前者のように、もともとITサービスの端末などとしてITサービスと共に使用されてきた組込み機器・システムも、後者のように、当初は単独で使用されていたものがネットワークを介して接続されるようになってきた組込み機器・システムも、もはや区別なく、ITサービスと連携して発展するようになってきた。IoT時代には、この傾向はますます強くなるものと思われる。

また、社会生活や経済活動へのICTの活用が進むに伴い、組込みソフトウェアに対しても大規模・高機能化が求められるようになった。組込みシステムの少リソース、高リアルタイム性という特徴は、組込みソフトウェアの大規模・高機能化を阻害するものであるが、半導体技術の著しい進展が、その実現を後押しする形となった。一方、組込みソフトウェアの大規模・高機能化が進むと、その開発効率の向上が望まれるようになった。それを実現・推進するのが、開発環境や開発技術などの発展である。これらは組込みソフトウェアコード量の増大を招き得ることから、組込みシステムの特徴は開発効率の向上をも阻害するものであるが、半導体技術の進展がそれを緩和している。

このように、開発効率の向上が更に組込みソフトウェアの大規模・高機能化をドライブするという循環が生じている。（図 1 参照）



(※)アーキテクチャ、開発プロセス、開発環境、人材育成、など

図1　組込みソフトウェア開発の循環

ここでポイントとなる開発環境・開発技術などとしては、アーキテクチャ、開発プロセス、開発環境（ツールなど）、人材育成などが考えられる。本稿以降では、組込みソフトウェアを中心とするこれらに関する最近の動向について、それぞれの専門家に寄稿していただいた記事を掲載する。本稿では、その前段として、これらについての状況を概観し、簡単な考察を加える。

2　組込みシステムのアーキテクチャ

組込み機器・システム及びそれらの開発は、一般に、次のような特徴を持つ：

- 利用可能なコンピューティング資源が少ない
- リアルタイム性の要求が厳しい
- 外部環境とのインタラクションが多い
- 顧客の要求が明確ではない

これらの特徴について、大規模・高機能化の影響を簡単に分析する。

(1) 利用可能なコンピューティング資源が少ない

とくに、組込み機器・システムに搭載可能なメモリサイズが大きな制約である。半導体技術の著しい進展により、この制約は緩和されるかに見えるが、組込み機器・システムの大規模・高機能化に伴って組込みソフトウェアが必要とする資源量も増大する。また、組込みソフトウェア開発の効率向上などを目的とするコード生成ツールや機能パッケージなどの使用により、機能当たりのコードサイズが増大している。従って、相対的にはこの状況は大きくは変わらない。

(2) リアルタイム性の要求が厳しい

マイクロプロセッサやメモリの性能向上に伴い、リアルタイム性要求を満たすことが容易になるかに見えるが、プラットフォームやコード生成ツールなどの使用により、各処理の所要ステップ数は増大している。従って、相対的にはこの要求の厳しさは大きくは変わらない。

(3) 外部環境とのインタラクションが多い

組込み機器・システムが前面となって、外部の様々なデータの収集が行われる。IoTでは、その種類や数が飛躍的に増大する。従って、組込みソフトウェア開発への要求もますます増大する。

(4) 顧客の要求が明確ではない

家電製品やスマートフォンのように、多くの組込み機器・システムは、顧客からの明示的な要求に基づいて開発されるわけではない。これらの場合、開発者サイドが経験と想像に基づいて機器・システムへの要求を設定し、未来予測をその要求に反映し、多くの関係者（関連部署）と協力しながら開発を進める。近年、通信技術などの進展により、市場からのフィードバックを迅速に行う仕組みを設けることができるようになり、短いサイクルで機能を積み上げ、評価しつつ、製品の価値を高めていくスタイルが普及しつつある。

一般的には上記のような特徴を有する組込みシステムであるが、システムの種別により採用するアーキテクチャは異なる。

例えば、スマートフォンでは、単一のプロセッサ上で多数のアプリケーションが動作する。これらのアプリケーションは、頻繁に更新される。このような機器は、パーソナルコンピュータやメインフレームなど、単一のコンピュータと類似した構成と言える。

他方、自動車では、エンジンやブレーキ、パワーウィンドウなどの制御対象対応に専用のプロセッサが存在する。プロセッサの合計は数十個に及び、最近では100個程度搭載した車もあるという。各プロセッサ上のアプリケーション、すなわち制御プログラムは、更新されることは多くない。なお、各プロセッサ間の連携については、当初は少なかったものの、高機能化に伴い急激に増加傾向にある※2。このような構成は、エンタプライズ系の機能分散システムあるいは広域のサービス連携システムと類似しているとも言える。レスポンスタイム制約の下で、(制御)データの配置と交換に工夫が必要となる。組込みシステムでは、とくに、リアルタイム性の制約が厳しいため、ハードウェアとソフトウェアとの協調が重要となる。

更に、継続的に機能拡張を行うシステムにおいては、アーキテクチャの問題は一層重要度が増す。また、大規模・高機能化に伴うシステム構造の階層化傾向が顕著になってきている。このような状況で、機能拡張を効率的に行いつつ、システムの品質を維持し続けるために、システムの特徴に応じた適切なアーキテクチャを採用する必要がある。

3　組込みシステムの開発プロセス

2013年のET-West講演でクリストファー・テイトは、日米における組込みデバイス（機器・システム）を比較しており[2]、それを図２に示す。同図によれば、日本のデバイスは米国のデバイスに比べ、組込みソフトウェアの比率が小さい。また、米国のデバイスには、日本のデバイスにはない「インターネットを介した改善」の部分が含まれる。概念的ではあるが、米国では当時から既に組込みソフトウェアのオンライン更新の導入が進んでいたことが分かる。なお、組込みソフトウェア開発データ白書[9]によれば、日本でも約半数がオンライン保守可と回答している。

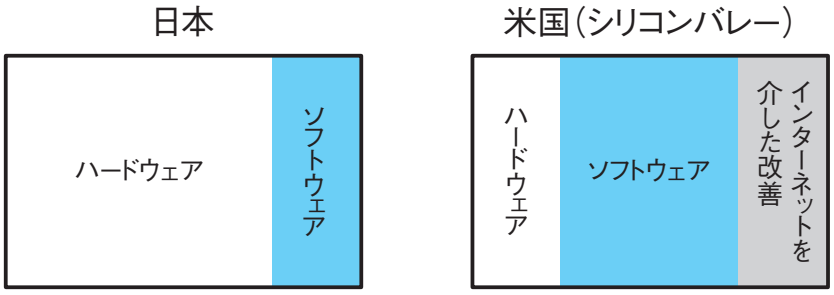


図２　日米における組込みデバイスの比較

クリストファー・テイトに倣ってIoT時代の組込みデバイスのモデルを描くと、図３の左上のようになろう。最近では、FPGA（Field Programmable Gate Array）などのPLD（Programmable Logic Device）の大規模化が進んでハードウェア論理の書換えが容易となっており、その状況を盛り込んだ。IoT時代には、組込みデバイスなどによって収集された各種データ（デバイスやその周辺環境の状態、デバイスの利用状況、など）がクラウドなどに集積され、ビッグデータ解析などに利用されると想定されている。デバイスのハードウェア論理やソフトウェアの更新の仕組みと合わせて、図３のモデルに含めた。

このようにして収集した組込みデバイスの利用者（エンドユーザ）の声を分析し、デバイスの機能拡張や修正を行うことができる。このようなエンドユーザからのフィードバックは、デバイスの開発開始時点では掴み得なかった「真の要求」と言える。

このようなデバイスの（定期的な）更新を適切に行うための有力なソフトウェア開発手段が、アジャイル開発である。

【脚注】

※2　例えば自動車の場合、エンジン制御が単独ではあり得ない状況であり、燃費、排ガス（環境）のトレードオフに対応するためにエアコン、シャーシー系との連携が必須になっているとのことである。

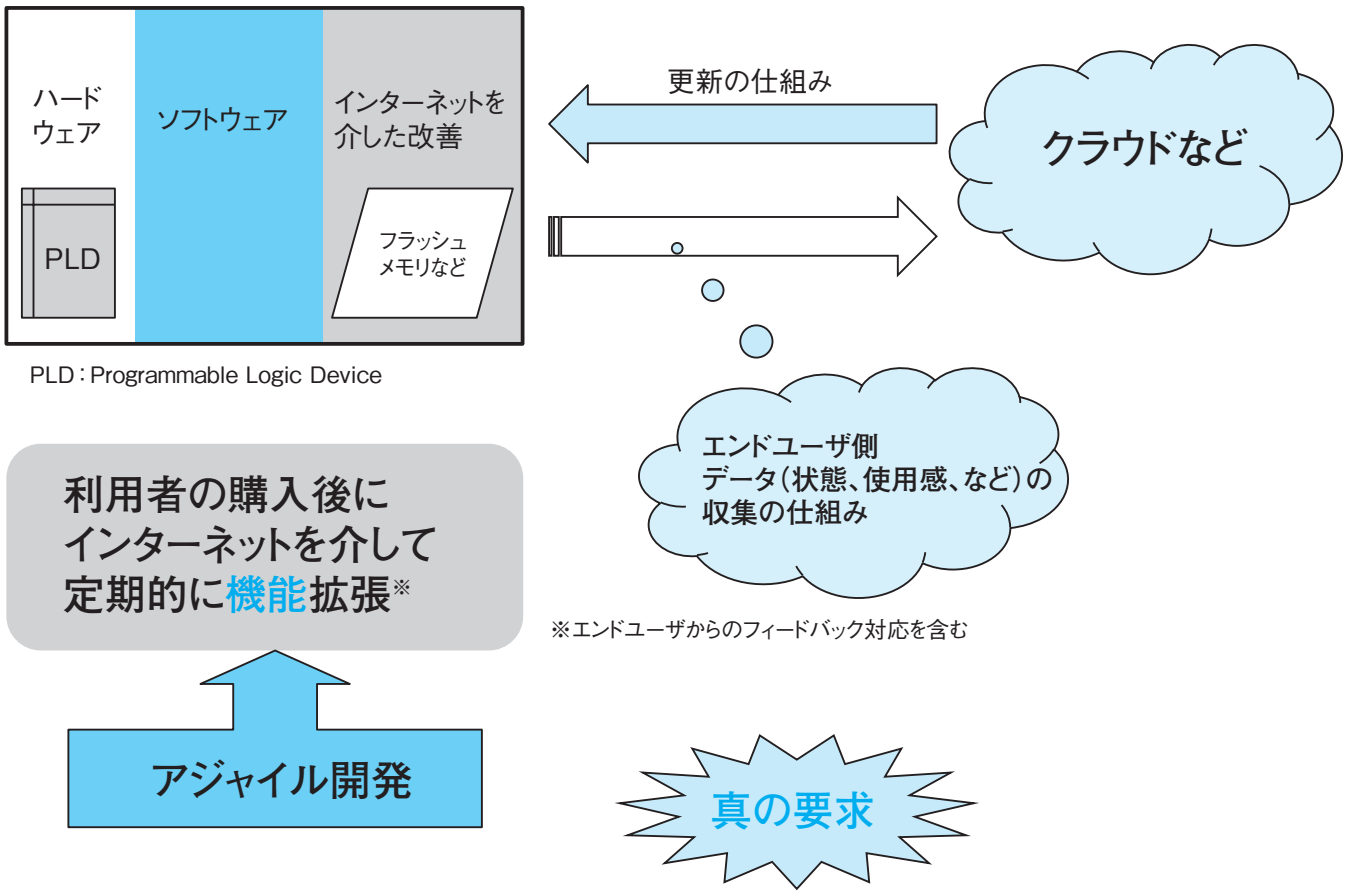


図３　組込みデバイスとアジャイル開発

4　組込みシステムの開発環境

ソフトウェアの開発プロセス（スタイル/手法）を効果的に運用するためには、適切なツールを活用することが重要である。そして、アプリケーション・ソフトウェアの要求管理、設計、実装、テスト、リリース管理などのライフサイクルにわたって一貫した管理が有効である。このような各フェーズ対応に適切なツール群を活用した統一的なプロセス管理は、ALM（Application Lifecycle Management）と言われる。

筆者が各開発現場の声を聞いたところでは、残念ながら我が国では、このようなツールが十分に活用されているとは言えないという印象を持っている。OECDの調査においても、我が国の労働生産性は先進国の中でも下位に位置する[3]。労働集約性の高いソフトウェア開発の現状がこの調査結果と合致している。この状況は、一つには、多重下請けというソフトウェア産業の構造的な問題が起因していると思われる。すなわち、大手ITベンダの系列ごとに独自のツールを使用することになり、汎用的なツールやオープンソースソフトウェアの普及がなかなか進まないのである。

こうした状況は、モデルベース開発（MBD：Model Based Development）といった技術の普及が進まない理由の一つになっているのではないかと思われる。MBDは、大規模・複雑化するソフトウェアに対し、効率的で高品質な開発を可能とする有力な手法であるが、専用のツールが欠かせない。後述の、MBDの全面的採用に踏み切ったトヨタ自動車のような事例[4]が、今後増えていくことを期待したい。

5　組込みシステムの開発組織・人材

前述のモデルにおけるもう一つのポイントは、組込みデバイスとエンタプライズ系システムとの連携である。これはIoTという語自体から当然と言えるが、組織・人材の観点で、より重要な意味を持つ。

技術者は、機器・システムだけを見ていては不十分である。すなわち、イノベーションに結び付きにくい。機器・システムの機能拡張の仕組み（サーバー／バックヤード／クラウド側）の理解と、機能拡張項目選定のトリガ（利用者の声を捉える仕組み）の理解が必要となる。このことは、組織としては組込み系とエンタプライズ系の協働、個々の人材としては両スキルの獲得ということに帰着する。

また、製品やサービスのアーキテクチャとは別に、組織構造としてのアーキテクチャも重要である。顧客のニーズに応えた製品やサービスを継続的に提供し続けるための組織とは、どのような機能を有し、それらがどのように連携するべきか？　ちなみに、コンウェイの法則とい

うものがあり、「設計組織により生み出されるシステムは、その組織とそっくり同じ構造となる」のだそうである。

対象システムにどのようなアーキテクチャを採用するか、多様な開発スタイルのうちから、システムのどの部分にどの開発スタイルや手法/ツールを用いるかを決定するためには、高度なデザイン力が求められる[5]。なお、組込みソフトウェア関連人材のスキルとしては、IPAは組込みスキル標準（ETSS Series）[6]を作成し、現在は、一般社団法人スキルマネージメント協会（SMA）[7]がその普及啓発を行っている。

上述のように、最近は、組込みシステムに関するスキルだけ保持していれば十分ということはなく、エンタプライズ系の素養もある程度は必要になってきている。これは、すべての人材に言えるというわけではないが、少なくとも組織としては保持すべきものである。従って、「組織スキル」ということになろう。

6 IPA/SECの取り組み

IPA/SECでは、その設立以来、組込み系のソフトウェア・エンジニアリングに関する取り組みを行ってきた。その成果は、次の書籍などにとりまとめ、公開している。

- ESxR（Embedded System xxx Reference）シリーズ[8]
  - 組込みソフトウェア開発向けコーディング作法ガイド（ESCR）
  - 組込みソフトウェア向け開発プロセスガイド（ESPR）
  - 組込みソフトウェア向けプロジェクトマネジメントガイド（ESMR）
  - 組込みソフトウェア開発向け品質作り込みガイド（ESQR）
  - 組込みソフトウェア向け設計ガイド（ESDR）
  - 組込みソフトウェア開発における品質向上の勧め［テスト編～事例集～］（ESTR）
  - 組込みソフトウェア開発における品質向上の勧め［バグ管理手法編］（ESBR）
- 組込みスキル標準（ETSS Series）[6]
- 組込みソフトウェア開発データ白書[9]

また、最近では、機器やシステムの障害事例を分析し、教訓としてとりまとめている[10]。これは、経験を開発・運用プロセスの改善につなげるものである。

これらの成果、特にESxRシリーズについては、IoT時代に対応して改訂する必要があるかもしれない。本特集の記事がその促進材料となることを期待したい。

最近、組込み機器・システムは大規模・複雑化しており、ほかのシステムや人との連携も増大している。このような機器・システムにおいて、安全上のリスクを設計時に分析したり、トラブルを解析したりする技術に対しても、従来とは異なる考え方が求められるようになってきている。更に、セーフティとセキュリティの両方を考慮した効率的な設計を求められている。セーフティは、一般に、いったん設計した後で要件が変わることはあまりない。他方、セキュリティについては、提供後にも新たな脅威が発生することがあり、提供者には顧客のもとにある既存の製品・サービスにおいてもその脅威への対応が求められる。IPA/SECでは、このような傾向に対応すべく、次のような新しい技術の検討にも着手している。

- ハザード分析（STAMP/STPA）／障害診断技術（事後V&V）[11]
- Safety&Security設計

システムの大規模化・複雑化に伴う影響の一つとして、トラブル解析の困難さの増大がある。IPA/SECでは、機器やシステム障害事例の分析に取り組んでいる[10]が、次のような事例が発生している：

- 通信回線制御ボード内のバッファメモリ（キュー）のオーバフローにより機能停止したが、そのバッファメモリはプログラムからは見えない内部バッファであり、動作中のキューの状態が全く分からない。
- 機器間通信スイッチ内のメモリの読み書きデータ不正により機能停止したが、スイッチ内のメモリの動作状況について、外部からは全く分からない。

かつての通信機器には、プロセッサやメモリが搭載されていなかったため、監視対象となる状態は、処理中/アイドルの別やデータの有無程度であり、あまり監視の必要はなかった。その後、それらの機器は次第にインテリジェント化し、今では、プロセッサや相当容量のメモリを含むのが普通である。にもかかわらず、機器の監視が従来の考え方のままであるならば、何が起きているかを適確かつ迅速に把握することはできない。

これに関連して、興味深い事例がある。自動運転支援で有名なテスラ社製の電気自動車では、事故を起こした車のログの解析により、次のことが分かったという[12]：

「該当車のログデータを分析したところ、この車両は正常に手動運転される状態にあり、事故が起きた際にも、また、その数分前にも、自動運転やクルーズコントロールの設定はされていなかったことが確認された。また、データは、約9.7km/hの速度で走行中に、突然、アクセルペダルが100％踏み込まれたことを示している。ドライバーのこの行動に伴い、車両は指示通りに駆動力を増加して加速した。」

また、事故などによりエアバッグが展開した場合、その事実などが事故直後に自動車メーカに自動的に通知されるというセーフティ＆セキュリティ機能が、米国では比較的広く浸透しているという。ただし、通信回線の容量によっては、事故の詳細データのすべてが転送されないこともあるらしい[13]。

いずれにしろ、トラブル解析を容易にするために、適切なログの取得機能が必要である。また、故障の予兆を監視するための機能も有効であろう。これらの機能は、ハードウェアとソフトウェアとの協調により、効率的に実現される。

7 まとめ～IoT時代に向けて

本誌第44号において、IoT時代に求められるソフトウェア開発スタイルについての特集[5]を掲載した。同特集の各記事は、必ずしも組込みソフトウェア向けではないが、IoT時代の組込みソフトウェア開発に対する多くの示唆を含んでいる。

組込みソフトウェアの開発では、従来、多くの現場で派生開発[14]が行われてきた。これは、既存のソフトウェアに変更・追加・削除を行うことにより新たな要求機能などを具備するソフトウェアを作るものである。IoT時代では、とくにコンシューマ向け機器においては、外部環境の変化に機敏に対応した商品/サービスの投入が求められることから、いわゆるアジャイル開発[15]も必要となる。このアジャイル開発と派生開発とは対立するものではなく、両立するものであると考える。各現場の経験などの状況に応じ、派生開発をアジャイルに回す、あるいはアジャイル開発に派生開発の要素を採り入れる、などの工夫を行うことが有効であろう。更に、制御のパターンやルールがある程度整理され、パラメータ化できるような部分に対しては、超高速開発[16]を採用することも考えられる。

ここで、新しい開発スタイルを全面的に採用した事例を紹介しておく。

トヨタ自動車は、2015年12月に発売した新型「プリウス」から、ハイブリッドシステムに用いる制御ソフトウェアの開発プロセスを全面的に刷新した[4]。

新型プリウスでは、より高度な制御が求められたほか、制御に対する要求が多様になった。このため、制御ソフトウェアを全面的に刷新すると共に、開発プロセスにも全面的にモデルベース開発を取り入れた。

具体的には、複数の海外ベンダから自動コード生成ツールやソフトウェア検証ツール、構成管理ツールなどを導入し、開発プロセスの効率化を図っている。制御ソフトそのものも、品質を確保しつつ新機能を迅速に追加できるようにするため、処理を集約したり、モード選択の組み合わせ数を減らしたりするなど、ソフトウェアの構造を簡略化した。これらにより、ソフトウェアの複雑度は従来に比べて半減したという。また、従来はソフトウェアの一部の開発をサプライヤに委託していたが、今回はすべて内製化したという。

この刷新に先立ち、トヨタ自動車社内では、数年前から、制御システム基盤開発を推進する部署がハイブリッドシステム開発の部署にMBDのメリットを説いて回っていたという。そうこうするうちに、3年前に、自動コード生成ツールの性能向上、すなわち、手書きの場合と同程度のコードを生成できることが明らかになったことなどを契機に、MBDの採用を決断したという。その前後で、2年間ほどをかけて、技術者へのMBDの教育を行ったという。

以上のように、組込み機器・システムは、従来の制御対象に特化し、いったん設計・提供された後はほぼそのままライフサイクルを全うするものから、顧客ニーズに応えて変更、拡張されるものへと変化してきている。いわば、組込み機器・システムのサービス化が進んでいると言えよう。このような動きに対応するために、組込み機器・システムの開発組織には、「サービス・トランスフォーメーション」が求められる。これを実現するために必要なことは、開発スタイルのみならず、アーキテクチャ、開発環境や組織・人材など、企業全体（エンタープライズ）のレベルで取り組む、「エンタープライズシステムズエンジニアリング」であろう。

【文献】

[1] 拡大し続ける「組込みシステム技術」の29年史, Tech総研, http://next.rikunabi.com/tech/docs/ct_s03600.jsp?p=001274

[2] クリストファー・テイト：イノベーションを生み出す日本へ, 再び ～ソフトウェアとハードウェアの対話が,日本に強さもたらす～, ET-West 2013 ヒートアップセッションHU-5講演, 2013年6月14日, 大阪.

[3] 平成27年版 労働経済の分析, 厚生労働省, 平成27年9月.

[4] 阿部眞一：第4世代プリウスとモデルベース開発の活用, dSPACE Japan User Conference 2016, 平成28年6月3日.

[5] 山下博之, 室修治：多様化が進むソフトウェア開発スタイル, SEC journal, Vol. 11, No. 4, IPA/SEC, 2016年3月1日.

[6] 組込みスキル標準（ETSS Series), http://www.ipa.go.jp/sec/softwareengineering/std/etss.html#200

[7] 一般社団法人スキルマネージメント協会, http://www.skill.or.jp/index.php

[8] IPA/SECの組込み系成果, http://www.ipa.go.jp/sec/softwareengineering/std/emb.html

[9] 組込みソフトウェア開発データ白書2015、及び、組込みソフトウェア向けプロジェクトマネジメントガイド［定量データ活用編］, http://www.ipa.go.jp/sec/reports/20151116.html

[10] 重要インフラ分野のシステム障害への対策, http://www.ipa.go.jp/sec/system/index.html

[11] 「はじめてのSTAMP/STPA ～システム思考に基づく新しい安全性解析手法～」の公開, http://www.ipa.go.jp/sec/reports/20160428.html

[12] テスラ,「モデルXが勝手に加速した」というオーナーの主張を否定, AutoBlog日本版, 2016年6月9日, http://jp.autoblog.com/2016/06/09/tesla-rejects-claim-unintended-acceleration-model-x/

[13] 野辺継男:テスラEVでまた事故,喫緊の課題は「機能告知の徹底」, 日経テクノロジーオンライン, 2016/07/09, http://techon.nikkeibp.co.jp/atcl/column/15/415543/070900042/

[14] 清水吉男：IoTの時代における派生開発の対応, SEC journal, Vol. 11, No. 4, IPA/SEC, 2016年3月1日.

[15] 平鍋健児：変化を味方につけるアジャイル開発, SEC journal, Vol. 11, No. 4, IPA/SEC, 2016年3月1日.

[16] 関隆明：ユーザーイニシアティブを可能にする超高速開発, SEC journal, Vol. 11, No. 4, IPA/SEC, 2016年3月1日.