特集：MAZDA CX-60

22

CX-60 における車両ネットワークシミュレーターを用いた開発効率化

Development Efficiency Improvement by Network Simulator in the CX-60

佐藤　陽平 *1　西原　大樹 *2　日原　圭祐 *3
Yohei Sato　Hiroki Nishihara　Keisuke Hihara
新川　力 *4　村上　龍馬 *5
Tsutomu Shinkawa　Ryoma Murakami

要　約

近年，自動車業界は，環境問題への対応やそれに伴う電動化や自動運転技術の登場により大きく変化しており，マツダではこの変化に対応しつつお客様に“走る歓び”を提供し続けるために新たな機能開発を進めている。機能の高度化及び多様化に伴い，クルマに搭載されるソフトウェアの開発規模は増大し，かつ車両内での通信量も増加している。このような状況で，いかにソフトウェア開発の効率を上げ，短期間に高品質なモノ造りを実現するかが我々マツダの技の見せ所である。本稿では CX-60 において効率的なソフトウェア開発を実現した，車両ネットワークシミュレーターによる機能の先行妥当性確認，先行動作検証について，その考え方と成果を紹介する。

Abstract

In recent years, the automobile industry has undergone major changes due to “Responding to environmental problems through electrification” and “Autonomous driving technology”. Mazda is developing new functions in order to continue to provide customers with the “Driving pleasure”. With the complication and diversification of functions, the development scale of software installed in cars is increasing, and the amount of communication in vehicles is increasing. Under these circumstances, it is a good opportunity to show our skills to improve the efficiency of software development and achieve high-quality development in a short period of time. This paper introduces the concept and results of the improvement of software development efficiency by the advanced verification and validation of the functions using the vehicle network simulator at the CX-60.

Key words : Electronics and control, Information, Communication, and control, Integration control, Controller area network, Simulation, Control system, Software, Vehicle network

1. はじめに

近年，クルマに求められる機能は高度化，多様化している。それに伴い，クルマの制御システムは複雑化している。これらの動きは今後ますます加速すると考えられる。

特に「CASE」と呼ばれる領域について革新が進んでいる。「CASE」とはコネクティッド（Connected），自動運転（Autonomous），シェアリング（Shared）及び電動化（Electric）の頭文字を取った造語である。これらの領域における革新によって，クルマの構成及び役割が大きく変化している。

マツダにおいてはこの変化に対応しつつお客様に走る歓びを提供し続けるために多くの機能開発を実現している。例えば，幅広い運転シーンにおいて，お客様により安心いただける力強い走りを提供する Mazda intelligent Drive Select（Mi-Drive）や，クルマをドライバーに合わせより快適にドライブを楽しむドライバー・パーソナライゼーション・システムのような機能開発である。

これらの機能は複数の電子制御ユニット（Electronic

*1〜5　電子基盤開発部
Electronic Platform Development Dept.

Control Unit: ECU）間の車両ネットワーク通信により実現される。例としてマツダでの ECU 数と通信信号数の遷移を示す（Fig. 1）。機能の高度化，多様化及び制御システムの複雑化に伴い，ECU に搭載されるソフトウェアの規模は大きくなり，かつ ECU 間のやり取りする通信量も増加している。それに伴いソフトウェア開発の負担も増大している。このような状況のため，いかにソフトウェア開発の効率を上げ，短期間に高品質なモノ造りを実現するかが重要になる。

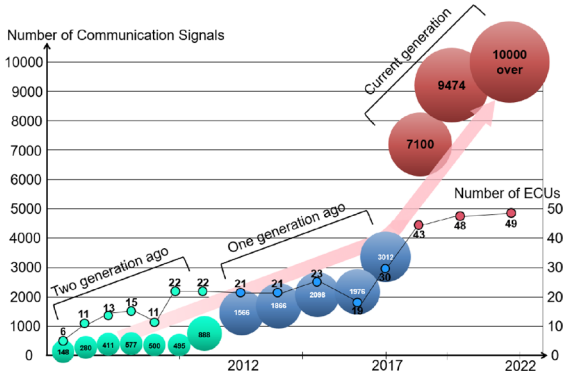

Fig. 1　Changes in the Number of Communication Signals and the Number of ECUs

2. ソフトウェア開発の効率化

2.1　ソフトウェアの開発プロセス

ソフトウェアを ECU に搭載する開発プロセスは一般的に Automotive SPICE® Process Assessment Model[1]（Automotive SPICE® は Verband der Automobilindustrie e.V.（VDA）の登録商標である）にて示される V 字プロセスが適用される。本稿では説明の簡単化のためシステムと各 ECU で開発プロセスが分かれている Version3.1 のプロセスではなく，Version2.5 のプロセスで説明する（Fig. 2）。

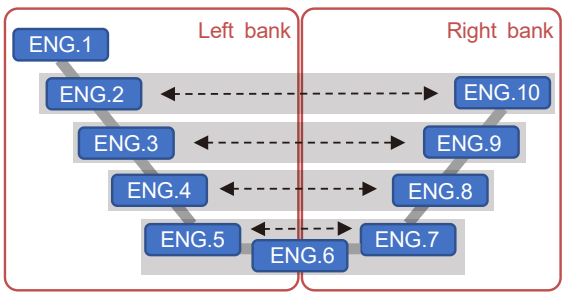

Fig. 2　V-Type Software Development Process

V 字プロセスでは左バンクにおいて次の工程を順に実施する。まずクルマとして機能で実現したいことである要件を収集する（ENG.1）。次にシステムとして必要な要件が何かを分析し（ENG.2），その要件が実現できるようにシステムアーキテクチャを設計し，各 ECU に要件の配分を決定する（ENG.3）。更に配分された要件を分析し（ENG.4），各 ECU としての設計に落とし込む（ENG.5）。各 ECU 内のソフトウェアは複数の構成要素（モジュール）から成り立っており，各構成要素間の機能配分を設計する工程と，各構成要素そのものの設計は ENG.5 で実施する。ソフトウェアの実装は（ENG.6）で実施する。

右バンクにおいては次の工程を順に実施する。まず，実装に対するソフトウェアの動作検証を行う（ENG.7）。その次に ECU が要件を満たしているかソフトウェアの妥当性確認（ENG.8）を実施する。次にシステムの動作検証（ENG.9）を行い，最後にシステムが要件を満たしているかの妥当性確認（ENG.10）を実施する。

「動作検証」は設計（仕様）のとおりシステム及びソフトウェアが動作するか検証するという意味である。また，「妥当性確認」とはシステム及びソフトウェアが要件（要求）を満足するか確認することを意味している。

これらの各工程は Fig. 2 に示すように，それぞれ各層ごとに対構造になっている。つまり，各左バンクの工程での要件（要求）及び設計（仕様）を対になる右バンクの妥当性確認及び動作検証時にて確認，検証するプロセスとなっている。

2.2　マツダでの開発効率向上の考え方

左バンクの時点で早期に誤りを発見できた場合，右バンクにおいてシステム検証やシステムの妥当性確認にて誤りを発見した場合に比べて，改修にかかる時間を抑えることができる。Kelly, John C. らによる論文「An Analysis of Defect Densities Found During Software Inspection」[2]では，左バンクで発見した誤りを修正する時間を 1 とした場合，右バンクで発見した誤りを修正する時間は 10〜34 倍だと報告されている。

そこでマツダは左バンクでの仕様策定に注力し，ENG.4 のインプットとなるサプライヤーへの提示仕様の精度を向上する。それにより，右バンクの ENG.8 の段階でサプライヤーより成果物として受け入れる ECU のソフトウェアおける提示仕様の誤りを撲滅することで，サプライヤー工程での実装不備による誤りの検証に集中できる状況を創り出す。この考え方を通じて，V 字プロセス全体で問題が流出しないことを保証できる無駄のない開発プロセスの構築を目標とした。

左バンクで行う要件収集（ENG.1），要件分析（ENG.2/ENG.4），仕様設計（ENG.3/ENG.5）は，右バンクの動作検証（ENG.7/ENG.9），妥当性確認（ENG.8/ENG.10）を実施することで仕様の正しさが初めて分かる。つまり，左バンクの精度を上げるためには右バンク相当の動作検証，妥当性確認を何らかの形で早期に実施する必要がある。そこで実現したい機能について要求のとおりに動作するシミュレーターを左バンク時点にて作成する。実際に動くシステムで相互理解を行い，人によ

る理解の差異を抑えて開発プロセスを加速した上で先行妥当性確認，先行動作検証を行い左バンクの精度を向上する（Fig. 3）。

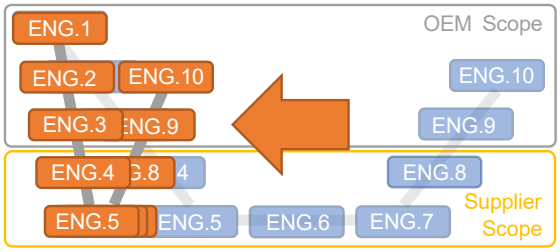

Fig. 3　Front Loading of Process Using Simulation

先行妥当性確認と先行動作検証は「目的指向と全体俯瞰」「抽象化・モデル化」「反復による発見と進化」の考え方を用いることで早期に漏れなく効率よく実施できる環境を実現する。具体的には，要求の実現を最終目標として，システム全体の妥当性確認及び動作検証を実施することで誤った要求分析及び設計（ENG.2/ENG.3/ENG.4/ENG.5）を防ぐ。そしてシステムの異常処理などを含む完全な模擬ではなく，確認するべきことに対し必要十分な模擬を行うことでシミュレーターの開発期間の短縮を実現しつつ，シミュレーター環境という利点を生かし，条件を変えた妥当性確認を高速に繰り返すことによって要求，仕様の高速育成を行う。

左バンクの開発プロセスについて，上記取り組みにて精度は上がるが，これは ENG.2 のアウトプットであるクルマとしての要求動作をシミュレーターに実装し，マツダの期待値に対する作検証，妥当性確認を行っているものであり，実際に車両に搭載する ECU の動作検証を行っているわけではない。そのため，右バンクではシミュレーターに対する動作検証と同等の動作検証が車載 ECU に対しても必要となる。

ここで動作検証（ENG.7）をマツダの期待値と車載 ECU それぞれで実施し，同一であることが確認できれば，シミュレーターと車載 ECU が仕様設計（ENG.5）に対して同一の動きとなっていることが確認できる。また本稿では ECU ごとにシミュレーターを作成し，車両ネットワーク通信でつなぐことで，タイミングの小さなズレやノイズに対してロバストでなければならないという車両ネットワーク通信の特性を生かし，シミュレーターと車載 ECU の差異を考慮しなくてよい構成としている。それにより ENG.7 より右バンク後段の工程における動作検証及び妥当性確認の結果はシミュレーターでの結果を流用できる（Fig. 4）。

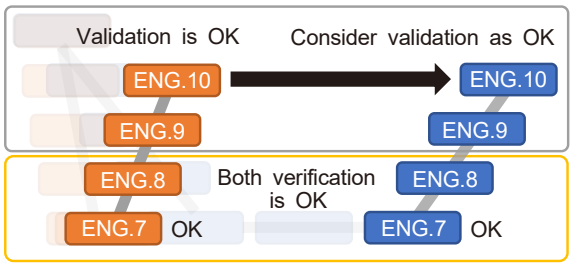

Fig. 4　Consider ECU Validation Result as OK

2.3　サプライヤーでの開発効率向上の考え方

ここまでの開発効率化の考え方により，元の左バンク完了前に「実現したい機能について要求のとおりに動作するシミュレーター」を手に入れることができる。これは ENG.8 のサプライヤー成果物の先取りである。ここでソフトウェアの要求分析，設計及び実装（ENG.4/ENG.5/ENG.6）の工程について考える。これらの工程は機能を実現するために配置された各 ECU で実施される。各 ECU はそれぞれ異なるサプライヤーにて開発されるため，いずれかの工程において誤りがあり機能全体として不整合が起きた場合，各 ECU を集めて実施するシステム検証（ENG.9）まで問題を検出できない。例えばある ECU が起動中は機能が動いていないとして故障を意味する信号 A を一時的に出すが，通信相手の ECU はその仕様を誤解または見落とし初期値 B を出すものと理解しているような場合である。これはこの工程において，通信相手となる ECU の動きを相手 ECU の仕様書でしか確認できず，そこから仕様を解釈し，相手 ECU の実際の動きを確認できないので理解誤りを完全には防げないためである。ENG.4/ENG.5/ENG.6 の工程時にサプライヤーへ通信相手 ECU の「ENG.8 成果物相当のシミュレーター」を提供することで，システム検証の前に各 ECU 間の整合を取れるようにし，システム検証（ENG.9）からの手戻りを撲滅する。

2.4　開発効率化の実現手段

2.2 節及び 2.3 節の開発効率化を下記 3Step で実現した。

（1）左バンクでの仕様の妥当性確認

Controller Area Network（CAN）インターフェースのレベルでさまざまな ECU を模擬するために「ネットワークシミュレーター」（Network Simulator：以下 NewS と呼ぶ）の作成／要件（ENG.1）から仕様の作成／仕様のとおり動作するソフトウェアの実装により，実際の動作を再現し，要求が満たせていることを確認する「仕様の妥当性確認」を行う。模擬対象は設定値の変更や画面遷移のような入力と出力が明確に対応している機能である。この際の模擬は状態遷移による記述を用いてインターフェースのレベルで行う。これにより今までの Model Based Development（MBD）[3] で活用してきたモデルと

インターフェースのレベルで接続し活用することができる。

CX-60 はMAZDA3 同様 Body Control Module（BCM）[4] を中心とした車両ネットワークトポロジーを採用しており，この BCM がさまざまな機能を実現している。NewS は一番複雑な CAN インターフェースをもつ BCM を模擬できるよう，Single board computer（SBC）に CAN インターフェースを多数接続し，構成している。また，BCM は一部 Local Interconnect Network（LIN）及び専用配線での電源供給による制御を行っているため，NewS には LIN インターフェースマイコン，リレーを接続し制御できる機能をもたせている（Fig. 5）。

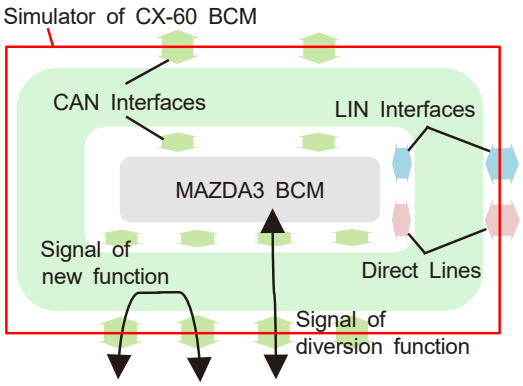

Fig. 5　Structure of Simulator of CX-60 BCM

この BCM は MAZDA3 より採用されている ECU である[4]。CX-60 でも同様の構想にて車両制御を実現するため，MAZDA3 から流用する機能と，CX-60 で新規に開発する機能が混在する。NewS を使って仕様の妥当性確認を行う際，BCM を模擬するためソフトウェアを作成するが，流用機能，新規機能両方の模擬ソフトウェアを作成することは，BCM の再開発が新規開発に追加され，その開発規模から動作検証を実施しての流用部分の等価性の保証が難しい。そのため，流用，新規が混在する ECU の模擬の際には Fig. 5 のように MAZDA3 のユニットを内部に抱え込む構成とし，流用機能は内部のユニットに通信を通過させ，処理するようにし，新規機能を SBC で処理を行うことで，流用機能の等価性と，新規機能の実装を両立した。

NewS で動作させるソフトウェアはライブラリなどの開発環境が整っており，人間が読んで理解しやすく，昨今よく使用されているプログラミング言語の Python（Python は Python Software Foundation の登録商標である）で作成する[5][6]。CAN，LIN，電源制御線は別途個別のモジュールで処理を行う。処理をモジュール化することで，モジュールごとの開発を可能とする。使用機会の多いプログラミング言語を選択すること，モジュールごとの開発とすることで，開発人員の調達を容易とし，各機能の並行開発を可能とすることで，車載 ECU そのものの開発をより容易かつ早く期待の動きを模擬することができる。

加えて，BCM の模擬の他に機能の実現のためには表示を行う ECU や車両各部を駆動する ECU など，その他の ECU も新規機能の模擬が必要である。その他の ECU の新規機能は PC 上のソフトウェアとして必要に応じた精度で実現する。例えばディスプレー表示はそのまま PC 上に表示を行い，ドアの開閉やシート移動などはリスト選択やラジオボタンの選択にて動作の入力，ステータスの表示を行うようにするなどである（Fig. 6）。

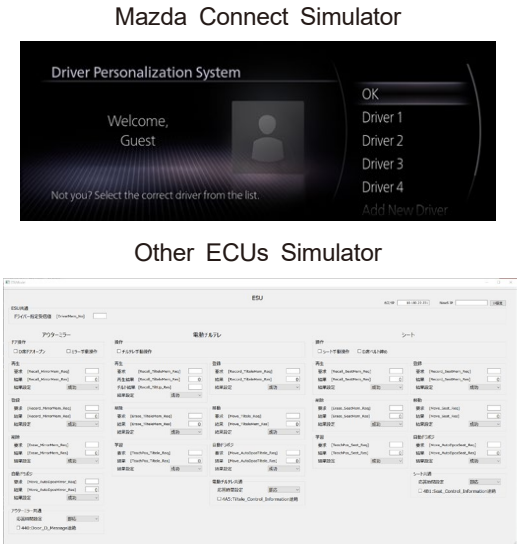

Fig. 6　Simulation of ECUs

このその他の ECU 模擬を行う PC 上のソフトウェアからは最終的に CAN 信号を出力する必要があるので，NewS にて CAN 信号と Ethernet 上の UDP パケットを相互変換し，PC 上ソフトウェアからの CAN 信号通信を実現する。PC 上での表示と NewS による CAN 通信処理を合わせたその他 ECU のシミュレーター全体を「Proof of Concept（PoC）by NewS」と呼ぶ（Fig. 7）。

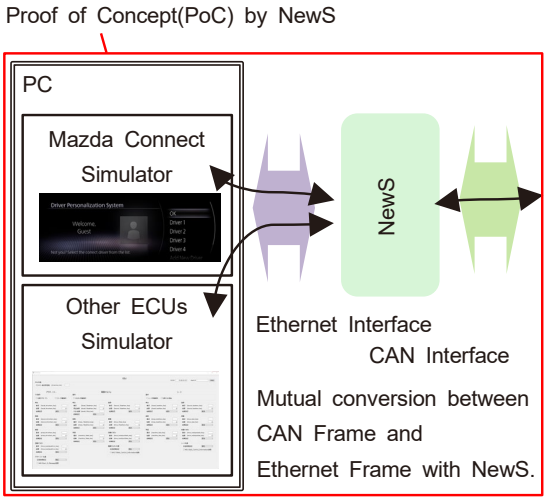

Fig. 7　Proof of Concept（PoC）by NewS

これらの要件（ENG.1）とおり動作するNewSと複数のPoC by NewSを組み合わせることで，システム全体の動きを模擬し，仕様の妥当性確認を実現する（Fig. 8）。

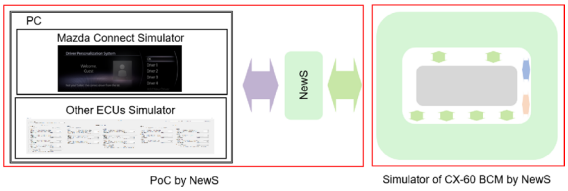

Fig. 8　Entire Validation Environment

（2）右バンクでの高速自動検証

2.2節で述べたように，右バンクでは今まで妥当性を確認してきた仕様と，車載ECUの挙動が等価であることを確認する必要がある。その確認に費やす費用と時間を最小とするため，車載ECUでの検証（ENG.7）における検証範囲は維持したまま，可能な限り自動で検証を実施する手段を開発した。自動検証は検証シナリオの自動入力とマツダ期待値であるシミュレーター出力値と車載ECUの出力値比較にて実現している。この自動検証システムをStraightforward test with equality by NewS：以下SweNと呼ぶ。SweNの構成を（Fig. 9）に示す。

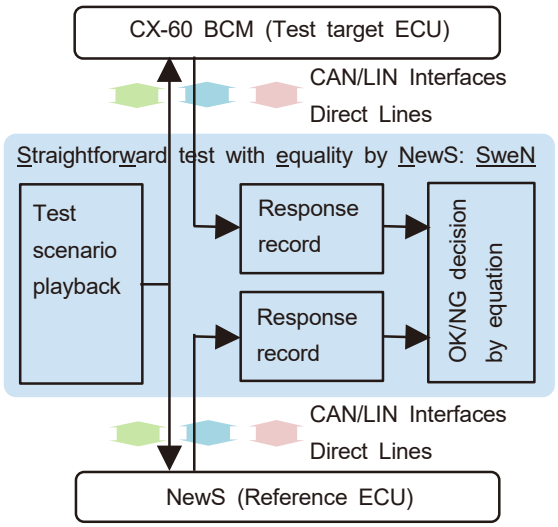

Fig. 9　Structure of SweN

車載ECUに対して必要なことは「シミュレーターでの検証と同等の検証」を実施することである。そこで左バンクで行ったシミュレーターの検証を右バンクでも活用する。このシミュレーター検証時のシミュレーターを車載ECUに置き換えることで，車載ECUが仕様とおりに動作しているか検証することができる。検証試験の入力はCANインターフェース，LINインターフェース，電源制御線のリレー制御インターフェースを介して実施し，ソフトウェア制御にて自動化する。

次に出力値の比較についてである。左バンクで作成したシミュレーターはこの時点で妥当性確認を完了しており，要件（ENG.1）を満たしている。そこで，シミュレーターと車載ECUに同時に上記の検証試験の入力を与え，出力が一致しているかを比較することで検証結果の自動判定を実現する。

この自動判定では出力のパターンにより，14種の判定方法を使い分けることで出力のパターンに合った適切な判定を実現している。例として出力変化タイミングの一致を比較する判定方法や信号送出間隔を比較する判定方法がある。このように複数の判定方法を用いるのは，機能により値の何が一致すれば判定をOKとできるかが異なるためである。例えばカウンター値であればシミュレーターと車載ECUの起動タイミングによってその開始タイミングは変わりうる。そのため，絶対値の比較では誤って判定NGとしてしまう。この機能の本質は1ずつカウントアップすることであり，比較するべきはそれぞれの出力値における前回値との差分である。

（3）ENG.5/ENG.6/ENG.7での開発効率化

（1）の妥当性確認にて準備した各ECUのシミュレーター群はCANを介して接続しているため，それらを車載ECUに置き換えることが可能である。サプライヤーでは開発対象の置き換えた車載ECU以外は期待値のとおりに動くシステムとしてシステム観点での開発を他の車載ECUが開発，リリースされる前から効率的に進めることが可能である（Fig. 10）。

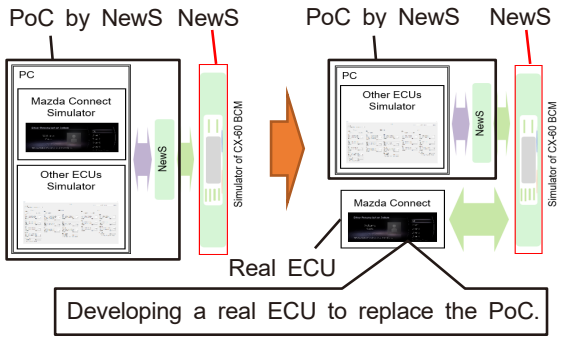

Fig. 10　Development Real ECU of Simulation Environment

3. CX-60でのソフトウェア開発効率化

3.1　左バンクでの妥当性確認

まず，BCMに対する新規要求を収集，整理し明確化した（136項目）。その後，要求に基づき各ECUの仕様を設計し，各ECU間の通信インターフェースを合意した。これらの情報を基に，本稿のソフトウェア開発効率化を実施した。

各要件の仕様確定時期に合わせ，シミュレーターの動作検証及びシミュレーターでの妥当性確認は3サイクルに分けて実施した。動作検証ではシミュレーターに対し

てトータル 16515 テストを試験し，シミュレーターが仕様のとおりできていることを確認した。妥当性確認ではトータル 580 テストを実施し，仕様書の確らしさを確認した（Fig. 11）。

Prior Verification in Left bank

	Number of Test	OK	NG
Cycle1	11,247	11,247	0
Cycle2	4,687	4687	0
Cycle3	581	581	0
Total	16,515	16,515	0

Prior Validation in Left bank

	Number of Test	OK	NG
Cycle1	64	64	0
Cycle2	209	209	0
Cycle3	307	307	0
Total	580	580	0

Fig. 11　Developing ECU in Simulation Environment

ここで具体例として CX-60 で新規搭載されたドライバー・パーソナライゼーション・システムでの成果を紹介する。BCM が外部センサーからの情報を基に，登録された「誰」が「ドライバー」として乗っているかを判断し，判断した「ドライバー」に合わせた設定を各 ECU に通知，設定するシステム[7] である。複数の ECU がドライバーや設定の情報をやりとりしながら機能を実現するため，本取り組みにより先行妥当性確認，先行検証を行うことで，このやりとりでの認識違いによる不具合を事前に潰し込むことができた。

3.2　右バンクでの高速自動検証

車載 ECU の検証では BCM の検証を対象とし，自動化により（Fig. 12）のような多数の検証を毎回の車載 ECU アップデートごとに実施し，以前実装していた機能が意図せず動作しなくなるデグレード不具合の流出を防ぐことができた。

CX-60 BCM ENG.7 Verification by SweN

	Test Case	OK	NG	Concern	Issue
Ver.1	1,442	1,442	0	9	0
Ver.2	2,625	2,595	30	17	2
Ver.3	12,463	12,372	91	18	8
Ver.4	13,247	13,247	0	0	0
Ver.5	13,136	13,136	0	0	0
Ver.6	13,239	13,239	0	0	0
Ver.7	13,233	13,233	0	0	0

Fig. 12　BCM Verification Result

また自動化を行ったことにより，左バンクで実施に数か月かかっていた 13000 程度のテストケースを 1 週間程で実行可能とした。

3.3　ECU 設計及び実装の効率化

最後に，今回トライアルとして NewS をマツダから BCM とは異なる車載 ECU を開発するサプライヤーへ貸出し，ECU 設計及び実装時の対向動作確認に活用いただいた。これにより，各 ECU を持ち寄って試験を行うシステム検証まで BCM での新規機能の動作を見ていなくても，NewS での動きを元にシステムの動きを整合させ，7 件の不具合をシステム検証前に検出でき，計画どおりのシステム検証を実施できた。

4. おわりに

本稿では，2 章ではシミュレーターにて ECU の動きを模擬することで左バンク時点にて妥当性確認を実現することで手戻りを防止し，サプライヤーにはマツダの期待動作をするシミュレーターを貸出することでサプライヤーの開発効率を上げる考え方を説明した。3 章ではその考え方を実現するイネーブラである NewS と SweN の説明と，CX-60 での具体例を説明した。

今後，車はインターネットと接続し，ソフトウェアが Over the Air（OTA）でアップデートされ，機能が逐次提供される変化の速い時代となる。そのような時代においてはお客様の期待を確実に素早く実現し続けることができるよう，クルマの機能開発は確実に，効率よく実施できなければならない。

お客様に更なる“走る歓び”を実感頂けるようソフトウェア開発力を進化させていく所存である。

参考文献

（1）Automotive SIG : Automotive SPICE® Process Assessment Model, https://www.automotivespice.com/fileadmin/software-download/automotiveSIG_PAM_v25.pdf（2010）

（2）Kelly. Hon C., Joseph S. Sherif, and Jonathon Hops. : An Analysis of Defect Densities Found During Software Inspection, Journal of Systems and Software Vol.17, No.2, pp.111-117

（3）藤川智士：マツダの目指すモデルベース開発，マツダ技報，No.31，pp.44-47（2013）

（4）薬師寺英明：新型 MAZDA3 の車両電子制御システム，マツダ技報，No.36，pp.149-153（2019）

（5）The RedMonk Programming Language Rankings: January 2022, https://redmonk.com/sogrady/2022/03/28/language-rankings-1-22/（2022）

（6）2021 Developer Survey : https://insights.stackoverflow.com/survey/2021（2021）

（7）末永修滋ほか：CX-60 ドライバー・パーソナライゼーション・システムの紹介，マツダ技報，No.39，pp.116-121（2022）

■著 者■


佐藤 陽平


西原 大樹


日原 圭祐


新川 力


村上 龍馬