61

モデルベースによるシステム開発における プラントモデルの高精度化について[1]

1. はじめに

車載システムをはじめとする組込みシステムの開発には、モデルベース開発が定着し始めている。しかし、システム開発全体にモデルベースが適用されている例は少なく、理想通りにモデルベース開発の恩恵を受けられているケースはまだ少ないのが現状である。本編では、あるシステム開発を事例として取り上げ、モデルベース開発の利点を享受できた背景とその要因、発生した問題を検証し、それに対する解決策を考察する。さらに将来に向けての取り組みを提案する。

1.1. モデルベース開発とは

モデルベース開発は、上流（設計工程）で作成したモデルを基にシミュレーションによる検証を行いながら開発を進めていく手法である。設計工程でシミュレーションを行うことによって、設計品質向上の効果が期待できる。設計の品質が向上することで、設計の不具合による後戻りを低減できるため、生産性向上の効果も得られる開発手法である。

プログラム作成工程では自動コード生成による大幅な生産性の向上、ヒューマンエラー低減の効果が得られる。

また、シミュレーションによる検証を行うことで、設計工程でもタイミングチャートなどを出力することが可能で、実際に動作した結果の見える化が実現できる。さらに、カバレッジ確認ツールの活用により、シミュレーションによって検証されたテストカバレッジも確認できる。検証工程においては、シミュレータの活用や設計工程で作成した検証データの再利用による効率化が実現でき、モデルの再利用が進めば、開発期間／コストの大幅な削減が期待できる。

[1] 事例提供: 東芝情報システム株式会社 エンベデッドシステム事業部 三島 隆司 氏

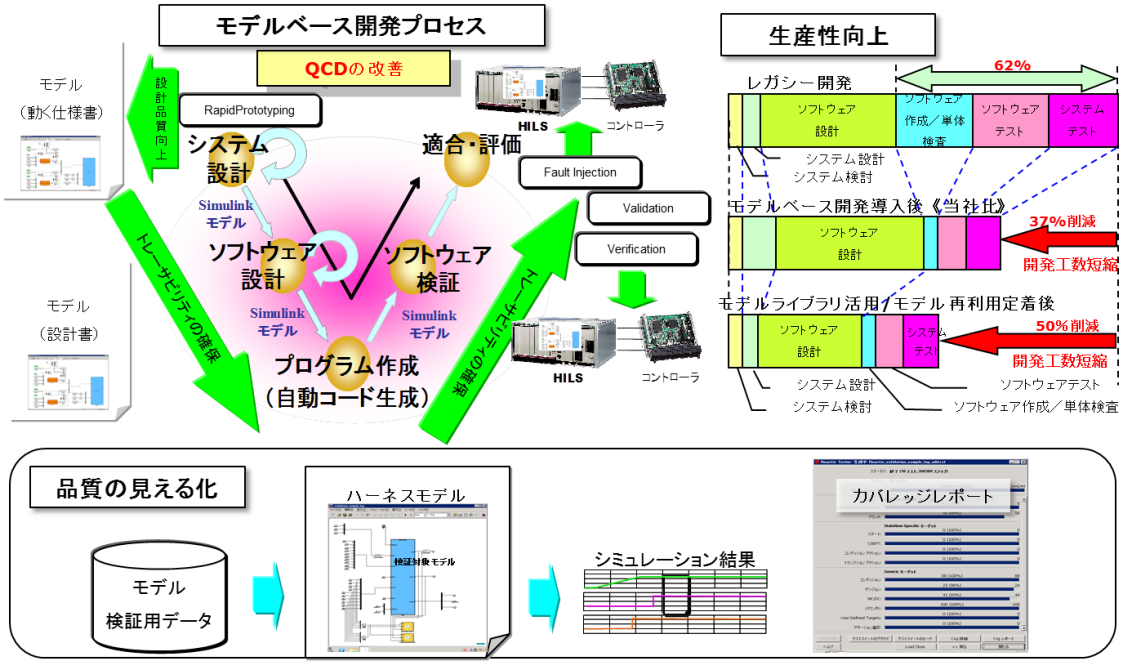

図 61-1　モデルベース開発の利点

1.2. モデルベース開発の成功のカギはプラントモデル

モデルベース開発への期待は品質の向上と生産性の向上の両立だが、なかなか期待通りの成果を上げられない場合がある。期待通りの成果を享受するためには開発プロセスの上流が重要であり、上流で行うシミュレーションの精度とテストカバレッジが品質と生産性に大きく影響する。シミュレーションの精度を高めるためには制御される側のモデル（プラントモデル）が肝心である。

モデルベース開発を行う開発者のほとんどは制御系の開発者であるため、制御モデルの作成は難なくこなすことができる。しかし、プラントモデルとなるとそれほど簡単にはいかない。プラントモデルと一言でまとめてしまっているが、機械系、電機系、化学系、流体系、他のサブシステムなどプラントモデルには様々なものがある。また、シミュレーションに要求される分解能も様々である。例えば機械系の分解能はｍｓオーダーでよいが、電機系の分解能はμｓあるいはｎｓオーダーの精度が要求される場合があり、それぞれ対象のシステムによってシミュレーションに要求される分解能が違う。理想的な振る舞いは数式でモデル化することができるが、実際のシステムになると様々なノイズや環境変化があるため、数式通りにならないケースも多い。このようにプラントモデルの作成には課題が多く、精度の高いモデルが必要となるとさらにハードルが上がってしまう。

また、大きなシステムを開発する場合は、複数のサブシステムを組み合わせることが多く、開発は別々の部門、組織で行われるケースがほとんどである。モデルベース開発を成功させるためには、複数のサブシステムをプラントモデルとしてモデル化する必要があるため、設計フェーズの歩調を合わせて、お互いの設計情報を交換しあうことが重要である。

2. 概要

2.1. プロジェクトの概要

ここでは実際に当社が経験したプロジェクトを例に、プラントモデルの重要性について説明する。本プロジェクトは実証実験用のプロトタイプ車両の HV システムの開発である。システム構成は図 61-2 で示すとおりで、パワートレーンは既存のまま利用し、パワーマネージメントシステムを本システム用に変更する。インバータ・モータシステムとバッテリーシステムは新規開発することとなった。それぞれのサブシステムは別々の部門が担当し、当部門はバッテリーシステムを担当し、モデルベース開発を適用することとした。本編ではシステム全体を開発することを踏まえたうえで、バッテリーシステムの開発を中心に成果と課題について検証する。

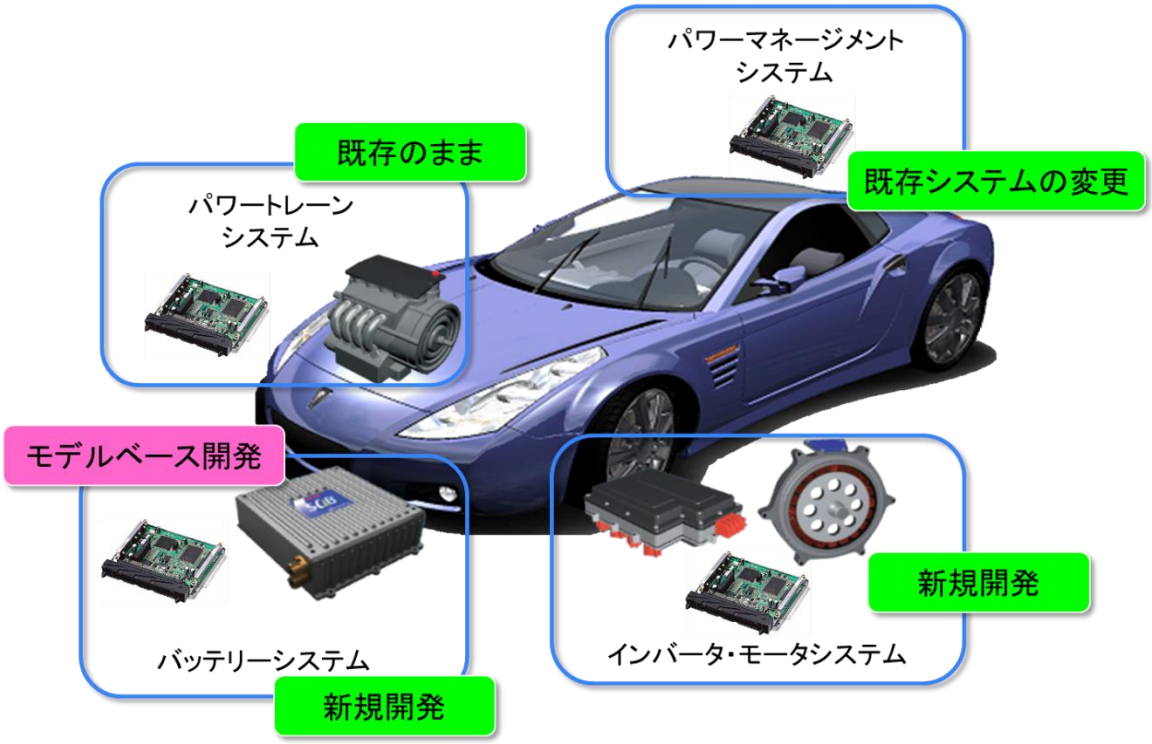

図 61-2　システム構成図

3. 取組みの目的

プロジェクトの目的（ゴール）は約 1 年間で実証実験用の車両を完成させることである。そのためにはモデルベース開発の利点を生かして、サブシステムをスケジュール通り完成させることと、車両として組み上がってからの不具合をできるだけ少なくすることが必須となる。

3.1. 解決すべき課題と対策

開発スケジュールとマイルストーンは図 61-3 で示すとおり。車両を含めた開発期間は約１年であるが、バッテリーシステムのソフトウェア開発は３か月という短い期間となった。

サブシステムごとに担当部門、担当会社が異なる体制で、新規開発のサブシステムと既存のサブシステムを組み合わせて新しいシステムを構築する計画で、仕様の決定、サブシステムごとの検証、サブシステム間の結合など各プロセスで様々な課題／問題が発生することが予想された。

車載システムの開発はモデルベース開発が浸透してきたとは言え、まだまだすり合わせ開発が残存し、サブシステムごとに開発手法も異なり、システム構築の思想も異なる状況である。

当部門が担当するバッテリーマネージメントユニット（BMU）の開発にはモデルベース開発を導入したが、他のサブシステムはレガシー開発の部分も多く、他部門、他社が担当であることからシステム全体をモデル化して俯瞰するという思想には至らならかった。そのためプラントモデルについては当部門ですべて作成する必要があった。

新規開発ではあるが、バッテリーモデルや汎用的なバッテリーマネージメントのモデルなど既存の資産があるためこれらを活用し、モデルベース開発を行うこととした。また、専用の ECU は開発せず、ラピッドプロタイプ[2] 用のリアルタイムシミュレータを利用することでさらに効率化を図ることとした。しかし、開発期間３ヶ月は非常に短く、更なる対策が必要であった。開発は細かいパーツに分けて複数名で開発することとし、最終段階でそれらを結合し、シミュレーションによる検証ができる環境（MILS[3]）の構築を並行で行うこととした。さらにリアルタイム、リアル信号でシミュレーションできる HILS[4] 環境を構築し、H/W との結合検査に備えることとした。HILS 環境を構築しておくことは、結合検査で発生した問題をタイムリーに確認、再検証することができるため効果は大きい。

---

[2] ラピッドプロトタイプとは、開発の初期の段階でリアルタイムシミュレータを ECU の代用として活用し、実プラントを使ったプロトタイプを構築し、新しい制御の実用性などを検討する手法

[3] Model In the Loop Simulation：モデルを使って PC 上でクローズドループのシミュレーションを行うこと、あるいはその環境。

[4] Hardwear In the Simulation：プラントモデルをリアルタイムシミュレータに搭載し、コントローラは実ハードウェアを使ってリアルタイム、リアル信号でクローズドループのシミュレーションを行うこと、あるいはその環境。

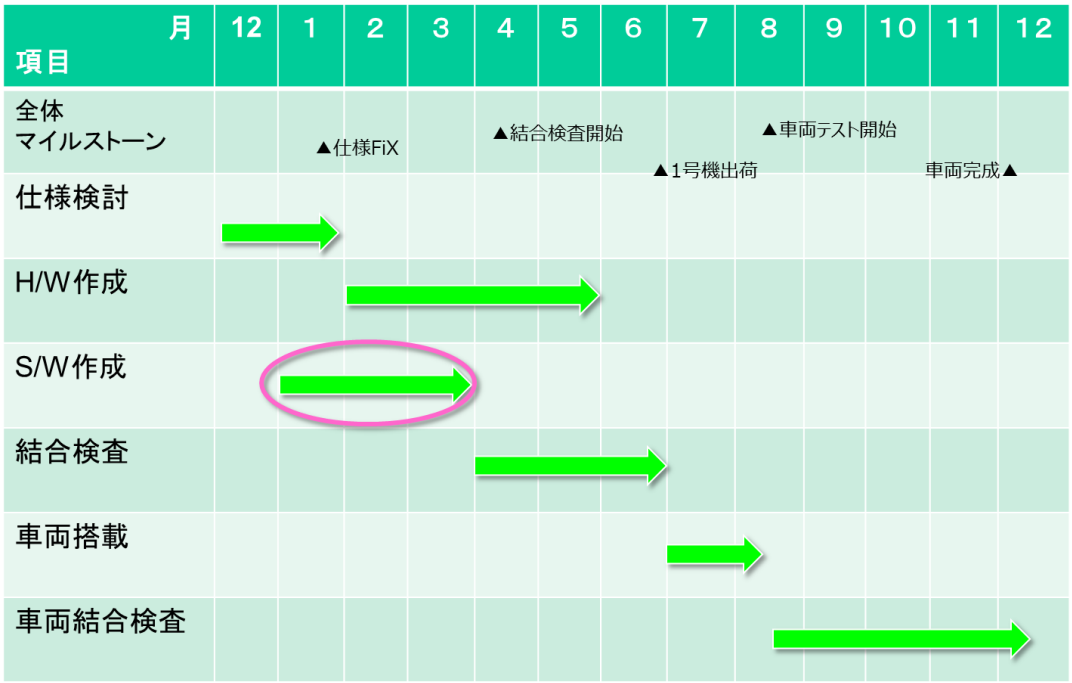

図 61-3　開発スケジュール

4. 取組みの対象、適用技術、評価・計測

4.1. モデルベース開発の導入

バッテリーシステムの構成は次ページのとおり。バッテリー制御を行うバッテリーマネージメントユニットには起動停止をつかさどる部分、バッテリーの仕様に合わせて制御をつかさどる部分、セルモニターとの通信をつかさどる部分、上位 ECU であるパワーマネージメントシステムとの通信をつかさどる部分、フェールセーフをつかさどる部分がある。

モデルベース開発で効果を得るために、システム全体をシミュレーションできる環境を整える必要がある。しかし、各サブシステムは担当部門、担当会社が異なるため、すべての仕様が開示されているわけではなく、挙動が不確定なハードウェアも存在する中でモデル化しなければならなかった。

開発期間も限られているため、システム全体を正確にモデル化することはあきらめ、BMU の周辺のプラント（バッテリー、セル監視ユニット、パワーマネージメント ECU、モータ／インバータ）のみを当部門でモデル化し、シミュレーション環境を整えることとした。

通常の開発期間であれば、モデルに仕様を盛り込んだ段階でシミュレーションを行い、シミュレーション結果を基にハードウェア開発部門や各サブシステム開発部門とのレビュー／確認を行うことができるのだが、開発期間が短く MILS 上で全体をシミュレーションできるようになるのは開発期間の後半になってしまうため、やむを得ず、ドキュメントによる仕様確認にとどめることとした。

バッテリーセルは新規開発したものを採用したことと、フェールセーフなどの組込みのためにバッテリーモジュールの開発部隊とは頻繁に打合せを行ったため、仕様についてかなり詳細に詰めることができた。これを制御モデルだけではなく、バッテリーのプラントモデルにも反映することができたため、精度の高いプラントモデルを構築することができた。こうすることにより、バッテリー制御の検証はフェールセーフも含めて MILS 上でほとんどのケースの検証が可能となった。残件としては通信エラーのトラップや時間にシビアなタイミングの検証項目のみとなった。

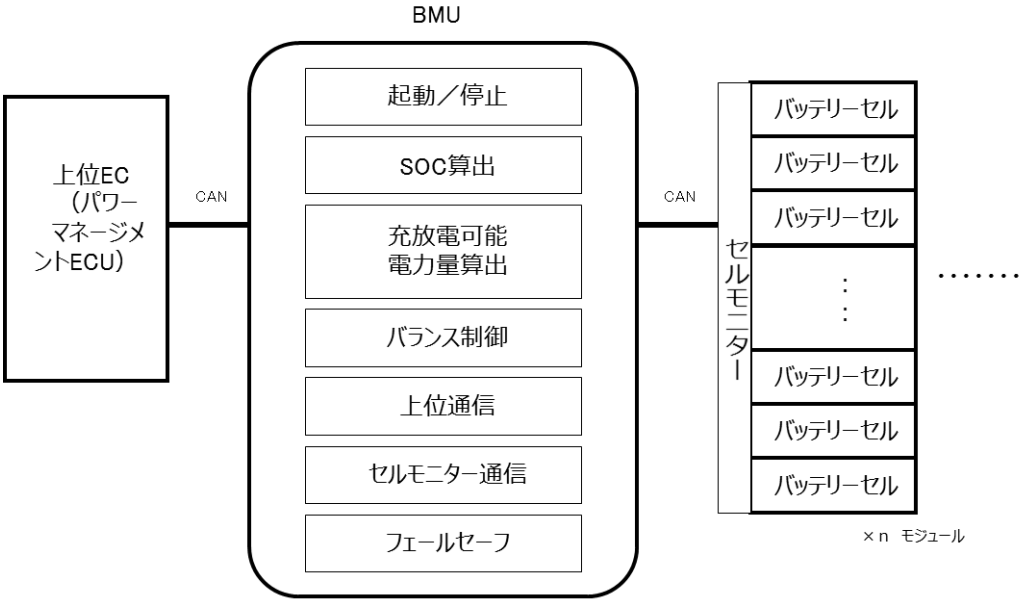

図 61-4　バッテリーマネージメントシステムの構成

4.2. HILS の導入

MILS による検証環境を構築したことで、ほとんどのユースケースをシミュレーションで確認できるようになったが、ハードウェアとの結合検査に備え、バッテリーマネージメントユニットのすべての信号をリアルタイムでシミュレーション検証を可能にする HILS を構築した。これにより MILS では検証できなかった通信エラー時の検証や、時間にシビアなタイミングの検証を可能にした。HILS に搭載したバッテリーのプラントモデルは前述の高精度モデルをそのまま搭載した。検証項目については MILS での検証シナリオを流用して自動で実行できるシステムを構築したため、ほとんど人手をかけずに MILS と同じ項目を全件検証できるようにした。

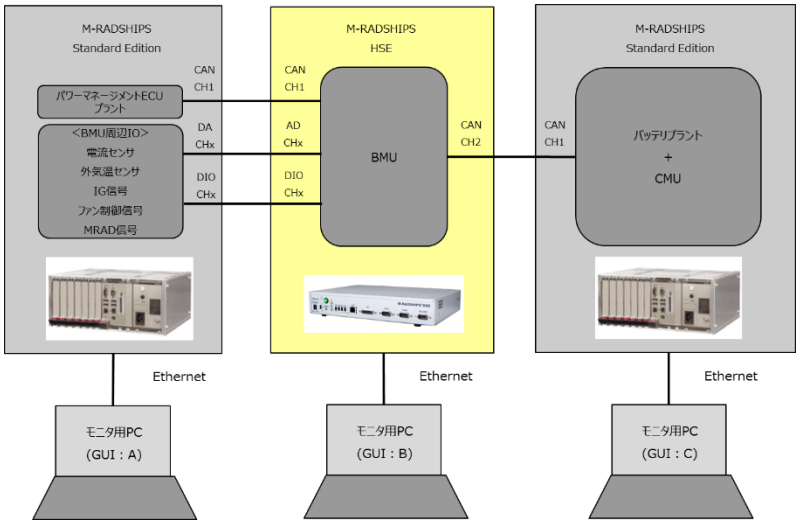

図 61-5　HILS 環境

4.3. モデルベース開発導入、HILS 導入の成果

モデルベース開発の導入、HILS の導入を行ったことにより、計画通りソフトウェア単体（MILS、HILS による検証を含め）については予定通り完成した。短い開発期間の中でも、各サブシステムをシミュレーションできるプラントモデルを作成したことで、システム全体がシミュレーション可能となった。このことが、結合検査の効率を上げることとなり、品質の向上につながった。計画時点で見込んでいたとはいえ、モデルベース開発導入の大きな成果となった。

5. 取組の実施、および実施上の問題、対策、工夫

5.1. 結合検査で問題発生

ソフトウェア単体での開発は順調であったが、バッテリーモジュールとの結合検査で問題が発生した。バッテリーセルは新規開発したものであり、BMU との結合検査までにモジュールとして組上げるというスケジュールで、シミュレータや他の治具によって確認するという工程はなく、結合検査に際しては、いわゆるぶっつけ本番状態であった。バッテリーモジュールは図 61-6 のような構成になっているが、本件のように大きな構成のバッテリーモジュールは検査する治具も大掛かりになってしまうため、バッテリーモジュール作成部門としてはスケジュール的にも、単体検査の余裕がなく結合検査がぶっつけ本番状態になったことはしかたなかったと思われる。

ぶっつけ本番のため、ハーネスの接続など単純な初期トラブルはあったものの、BMU の観点での検証は最初の２週間でほぼ完了した。しかし、バッテリーモジュールの電気的な振る舞いなど、ハードウェアに起因する検証に時間を要することとなった。このハードウェアは、本プロジェクトで新規開発したもので、ドキュメントベースの仕様書はあったが、プラント側の電気的な振る舞いがどうなるかが明確になっていなかったため、プラントモデル化できていなかった部

分である。

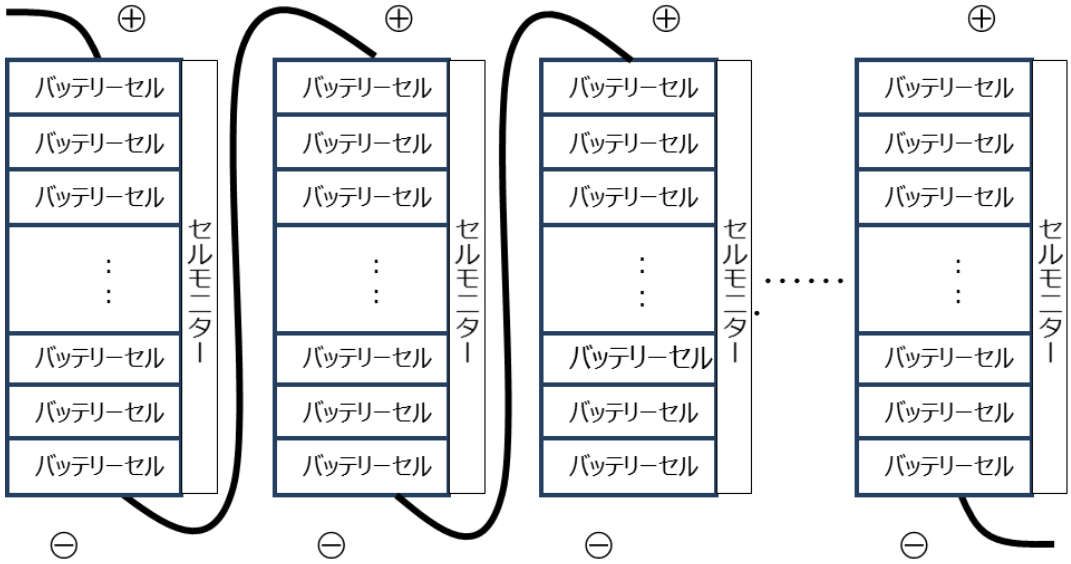

図 61-6　バッテリーモジュール

5.2. 部門間での進め方の違いが問題の原因

問題となった新規開発のハードウェアには、仕様をすり合せた際には定数で提示されていた。しかし、実は現物との調整（キャリブレーション）が必要な値があった。この調整は、ソフトウェアとハードウェアが互いに変更しながら行う必要があった。キャリブレーションが必要であることが結合検査の時点で発覚したのだが、これはおそらくバッテリーモジュール作成部門では想定できていたはずである。しかし重大なことだとは思っておらず結合検査の時点ですり合せればよい事項だと考えていたと思われる。開発の早い段階で問題が想定できるような事項でも後工程で対応するレガシー開発の慣習を引きずったものであり、早い段階で想定できる問題に対応することができるモデルベース開発の利点を消してしまうことであった。

ハードウェアを制御するというシステムを作成する場合、キャリブレーションが発生することは想定しなければならない。設計時点でキャリブレーションが必要なパラメータなのか、そうでないのかを把握しておかなければ、場合によっては設計工程まで後戻りが発生してしまうことになる。

本件の場合、キャリブレーションを予定していなかったパラメータに関してはプログラム領域にデータを持っていた。したがって、キャリブレーションを行うためにはプログラム（＝モデル）を修正しなければならなかった。モデルは設計書であり、トレーサビリティをキープするために厳格な管理を行うルールになっている。本件のルールではモデルを変更した場合、シミュレーション確認を行い合格となってから構成管理の DB をアップデートしてリリースするということになっていた。しかし、開発ツールやシミュレーション環境なども現場に持ち込むことは難しい。そのため、モデルの更新をルール通りに行うことができなくなり、なし崩しで、現場でモデルの変更を行いそのままリリースするという運用にせざるを得なくなってしまった。

結果として次のような問題に発展してしまった。

① バッテリーモジュール作成部門のモデルベース開発に対する不信感

これまでの開発であれば、現場で“ちょちょっと”直して簡単に調整できたのになぜそんなに手間がかかるのか？何とかして欲しいという要望が現場から上がる。これによってなし崩しの現場対応が始まってしまう。

② 現場に立ち会ったソフトウェア開発者の長時間拘束

最初からキャリブレーション用の変数にしておけば、キャリブレーション作業はバッテリーモジュール作成部隊に任せて問題が発生したときだけ対応すればよかった。しかし、モデル自体を修正しなければならいため、ソフトウェア開発者がずっと張り付くことが必須になってしまった。工数的にもモチベーション的にも悪影響を及ぼしてしまった。

③ 現場でのモデル変更が当たり前となり、構成管理に乱れが発生

現場でのモデルの変更が当たり前になると、構成管理が乱れ開発元での管理ができなくなり、管理者側にもフラストレーションがたまることとなってしまった。

④ 現場でのモデルの修正によってシミュレーション環境との乖離

現場で起きた問題を現場でアップデートするようになり、開発元でのシミュレーション環境との乖離が大きくなってしまった。最終的には開発元でのシミュレーションができない状態になってしまった。

きっかけはそれほど大きな問題だとは考えていなかったが、小さな問題が火種となりそれが積み重なって大きな問題となってしまった。それでも何とか期日通りに結合検査を完了し、バッテリーシステムを車両への組み付け工程に向けてリリースすることができた。

現場ではモデルベース開発についての不信感もあったが、最終的にはソフトウェア開発部門からの提案により、バッテリーシステムの検証には図 61-7 のような HILS を導入した検証環境を構築した。この環境を活用したことによる効率化が期日通りの出荷に大きく貢献することとなった。

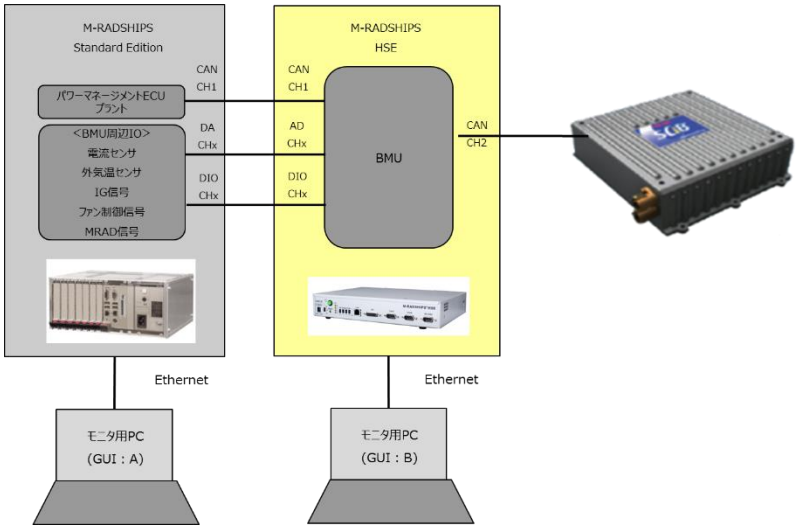

図 61-7　バッテリーシステム検証用 HILS システム

5.3. 車両結合検査での問題発生

車両と結合して行う車両結合検査を開始した時点でいきなり問題が発生した。上位 ECU であるパワーマネージメント ECU とバッテリーシステムの起動シーケンスがかみ合っておらず、起動しなかったのである。起動・停止のシーケンスについては設計時点で何度もすり合せたのだが、開発者間の意志疎通ができていなかった。

バッテリーモジュールと BMU の結合検査で起きた問題と同じく、現場ですり合せることを優先した計画となっていたため、上流でのすり合せ時の意識に食い違いがあったと思われる。仕様のすり合わせがきっちりできていたならば、シミュレーション可能な部分であり不具合の発生を防ぐことができたはずである。問題の発生場所が起動シーケンスであったため、ここを直さなければ検証が進まないため、即対応を迫られることになった。これをきっかけに現場でのすり合せ対応が始まってしまった。しかし、大きな問題はこの一点で、結果的にこれ以降は大きな問題がは発生することなく車両結合検査を終えることができた。

6. 達成度の評価

6.1. プロジェクトの振り返り

短い開発期間のプロジェクトでもモデルベース開発を導入することで、想定通りの成果を得られた。ソフトウェア単体の部分に限れば十分な成果を得られた。しかし、結合検査以降の工程については課題もあった。結合検査以降もしっかり成果を得られた部分は、やはりプラントモデルを精度高く作り込めた部分であった。プラントモデルがしっかり作り込めなかった部分は、現物合わせのすり合わせ開発となり、シミュレーション確認できた部分に比べ 10 倍以上の時間を要してしまった。

スケジュールを再度確認してみると、すり合わせ開発を想定したスケジュールになっていることがわかる。もちろんバッテリーシステム以外のサブシステムはレガシー開発で行われたためしかたない部分はあるが、全面的にモデルベース開発ができないまでも、シミュレーションを活用することは可能だと考えられる。“タラレバ”ではあるが、最初にシステム全体をシミュレーションすることを計画し、結合検査時のすり合わせ部分を排除することができれば、図 61-8 のように S/W 作成期間をもう少し長くとることができた可能性がある。ソフトウェア開発期間がもう少し長く取れれば、シミュレーションによる仕様詰めを行うことができたはずである。そうすれば全体的にはスケジュールを縮められた可能性はある。

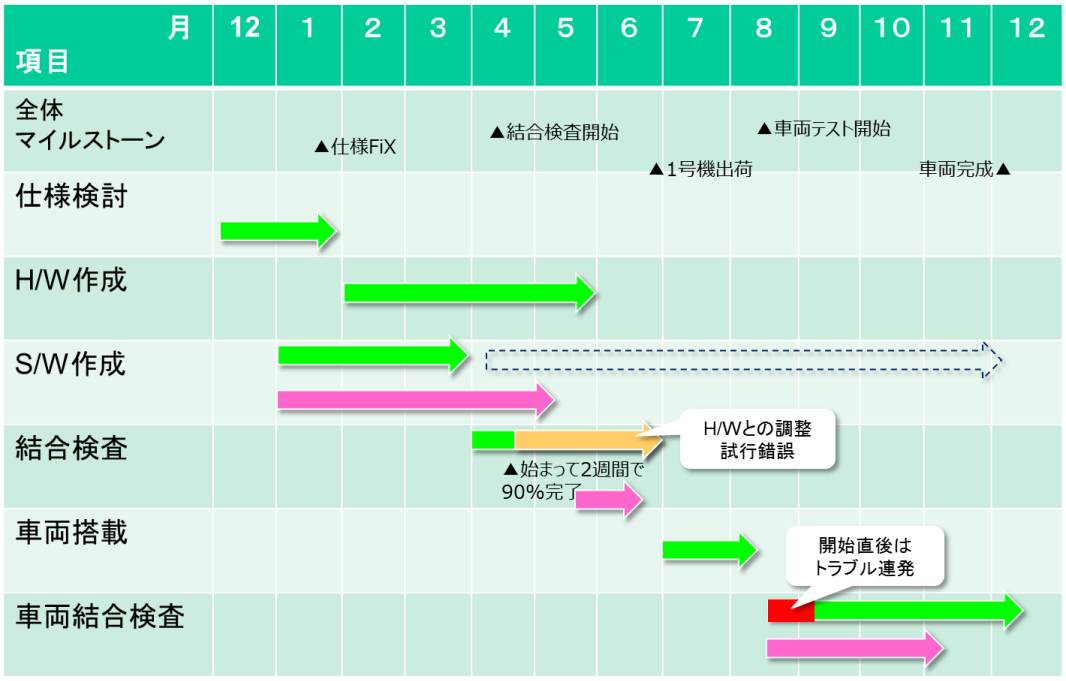

図 61-8 “タラレバ”のスケジュール

6.2. モデルベース開発の利点を最大限に生かす

モデルベース開発の利点を最大限に生かすためには本プロジェクトをどうするべきだったか。もう少し突っ込んで振返ってみると、以下のポイントが挙げられる。

① 新規開発のハードウェア導入への対応

本件では新規開発のハードウェアを導入した部分に見事に問題が発生した。新規開発のハードウェアの問題は、問題が起きた個所以外にもいくつか存在したが、仕様書から振る舞いが想定できたものは、プラントモデルに組み込んでシミュレーション確認を行った。しかし、問題が起きた個所については、振る舞いが不明確という理由で、プラントモデルが作成できなかった。リスクがあることが解っていながら結果的に放置してしまったことになる。

<u>ではどうするべきだったのか？</u>

振る舞いが不明確であってもプラントモデルは作るべきだった。プラントモデルを作れば何が不明確で、どこを決めなければならないか？何を現物と合わせる必要があるかを認識することができたはずである。少なくともその時点でハードウェア作成部門とそのことについて話す機会が得られたはずである。そうすればキャリブレーションが必要だということも、あらかじめ認識できたはずである。このことからも、プラントモデルの作成は新規開発のハードウェア導入によるリスク対策としても役立つと考えられる。

さらにリスクを軽減する施策としては、ラピッドプロトタイプを導入することをお勧めしたい。キャリブレーションを行う必要がある場合、本件のようにソフトウェアだけでなく、ハードウェアも調整する必要がある場合もある。ラピッドプロトタイプを導入すれば結合

検査工程の前にキャリブレーションを行うことが可能になる。そうすることによって、結合検査工程では必要最小限の調整にとどめることが可能となる。結合検査の工程を短くすることができれば、ハードウェア、ソフトウェアのそれぞれの開発工程に余裕が生まれることになる。

② プラントモデルの精度向上

本件では、上位 ECU との起動シーケンスに不整合があり車両結合の工程に大きな影響が出てしまった。シーケンス制御が必要な部分のセルモニターなどは、仕様確認とシミュレーション確認に加えて HILS での確認を行った。しかし、上位 ECU とは仕様確認にとどめ、上位 ECU のプラントモデルは当部門で作成した。以降、当部門で作成したプラントモデルを使ってシミュレーション確認、HILS での確認を行っている。プラントモデルを作成する時点では、インプットとなる仕様がすり合っていなかったことに加え、電源投入時の過渡応答と起動時間の問題もあったが、これらの問題を予測できなかったことはしかたない。

<u>では、これらの問題に対してはどうするべきだったのか？</u>

起動シーケンスのシミュレーション結果のレビューを上位 ECU 担当者と行っておくべきだった。仕様のすり合わせでは見えなかった部分がシミュレーション結果を見ることで気が付いた可能性はある。さらに電源投入時の過渡応答とそれぞれのサブシステムの起動時間をプラントモデルに盛り込んでおくべきだった。もちろんプラントモデル作成時点では、各々の時間については完全に Fix しているわけではないが、ある程度の想定範囲で時間を振ってシミュレーションすることによって、問題の発生個所の想定ができた可能性はある。

起動停止はどんなシステムでも問題になるケースが多いが、問題が発見されるのは開発工程の後工程であることが多い。起動停止部分だけでも各サブシステムのプラントモデルの精度を高めることで最終工程でのトラブルを未然に防ぐことが可能となる。

7. 今後の取組と考察

本編の中でモデルベース開発の利点を享受するためには、プラントモデルの精度上げるべきだと論じてきたが、プラントモデルの精度を上げることはそれほど簡単なことではない。冒頭でも述べたが、すべての現象を数式で表すことは非常に難しく、予測できないことも多々あるのが現実である。その一つの突破口としてプラントモデルの作成に機械学習を用いることを検討している。

7.1. 機械学習によるプラントモデルの作成

図 61-9 のように機械学習には様々なものがある。深層学習（Deep Learning）や人工知能なども機械学習の一種である。機械学習の特徴としてはデータがあればモデル化できるというところである。機械学習はコンピュータを用いてデータから、パターンや特徴を解析し、学習するシステムであり、人間には解析が難しい大量で不揃いなデータを使って学習し、モデル化することができることが特徴である。

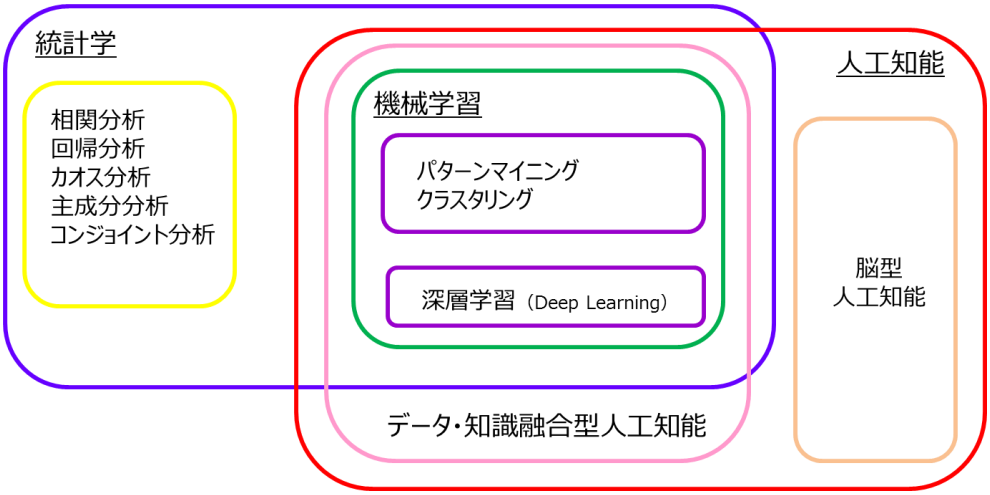

図 61-9　機械学習の種類

従来、数式では表しきれないプラントのモデル化や、できあがったプラントモデルをブラッシュアップする際には実機よりデータを取得し、人の手でチューニングを行うというのが一般的であった。この時、人の手で行うためデータを取得する際には、変化要素である様々なパラメータのうち、一つだけを変化させ、他のパラメータを一定になるような環境を構築する必要があった。たくさんのパラメータが存在する場合は、この方法をパラメータの数分繰り返してデータを取得しなければならない。このデータの計測には、時間と労力がかかる。ものによっては環境を一定にして計測することが不可能な場合もある。

機械学習を活用することで、人間には解析できない不揃いなデータであってもモデル化することができるため、上記のように環境を一定に保って測定したデータを用意する必要が無くなりデータの計測の時間と労力を削減できると考えている。まだ実験的な取り組みだが、リチウムイオンバッテリーを題材にして機械学習によるモデル化を実験した例があるので以降で紹介す

る。

7.2. 採用した手法

図 61-9 で示した通り機械学習と言っても様々な手法がある。本件ではパターンマイニングを行い教師データあり、Regression という手法を採用した。

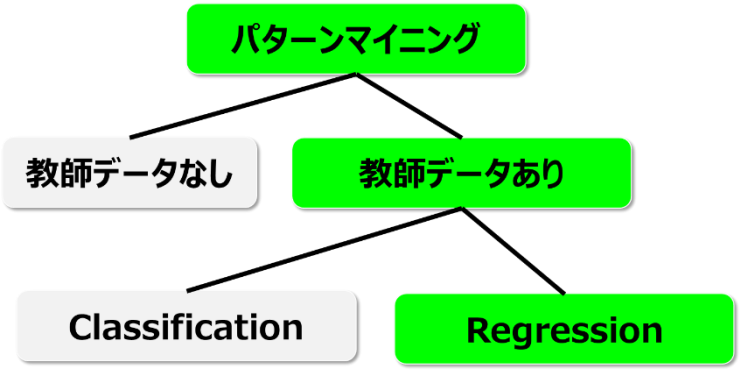

図 61-10　機械学習の手法

パターンマイニングは、データ集合から特徴的なパターンを見つけ出し、関連するパラメータなどを見つけ出すことから始める。本件でモデル化対象としたリチウムイオンバッテリーの特徴を形成するパラメータは電圧、電流、温度、SOC（充電残量）、電流の印加時間などであり、電圧が出力となるモデルを作ることとした。機械学習では教師データありと教師データなしの手法がある。教師データなしの手法ではデータの集合の中から、似たもの同士をいくつかの集合に分類するという手法であるため、本件には適さない。本件では、あらかじめ計測してあるいくつかのデータを教師として、入力に対する出力を予測するモデルを作成することとした。結果の出力方法として Classification（分類）と Regression（数値回帰）の 2 種類があるが、本件では出力として分類ではなく数値が必要なため、Regression を採用することとした。

リチウムイオンバッテリーの充放電特性のモデル化のトライアルを行った結果、教師データの数が重要だということがわかった。当初 1200 秒 12000 レコードのデータを教師データとした場合図 61-11 のように傾向は再現できているようだが、ピークが一致しなかった。

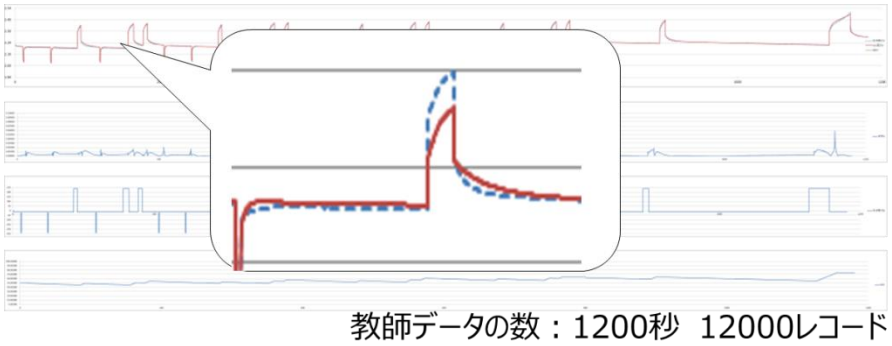

図 61-11　トライアル 1

次に、教師データのサンプル数を増やしてみると、傾向だけでなくピークもほぼ再現されている。

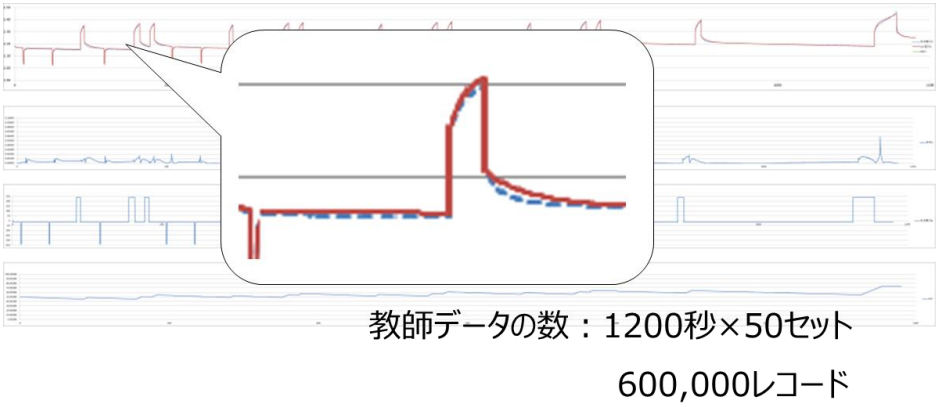

図 61-12　トライアル２

次に、温度や電流の利用領域を変更してトライアルを行った。当たり前だが教師データがカバーしている範囲に関しては、前述と同様の結果が得られた。しかし、教師データがカバーしていない領域については不安定で傾向が一致しないケースもある。

7.3. まとめ

機械学習を活用してプラントモデルを作成する手法は利点も多く、しっかり特徴を理解して活用することでプラントモデルの精度を上げることができる。これによりモデルベース開発の利点をさらに多く享受できるようになると考えられる。しかし、機械学習を活用して作成したプラントモデルの信頼性については現在のところ証明することが難しいという課題もある。今回のトライアルで、関係する教師データの品質と網羅性が、できあがったプラントモデルの精度に影響することがわかった。機械学習を活用したプラントモデルの信頼性については、教師データの素性と数量、カバー領域を使って指標化できると、利用者の安心感を得られる可能性がある。まだまだこれから検討を進める必要があるが、機械学習の利用は今後のキーテクノロジーの一つになると考えられる。

掲載されている会社名・製品名などは、各社の登録商標または商標です。

独立行政法人情報処理推進機構 技術本部 ソフトウェア高信頼化センター（IPA/SEC）