

## 15-B-14

### モデルベース開発とコード解析を用いた 組込みソフトウェアの開発<sup>1</sup>

#### 1. 技術・手法の概要

本事例は、アルプス電気株式会社（以降、アルプス電気）の自動車向け組込みソフトウェア開発における、「設計・検証コストの改善」事例である。

自動車の高機能化・電子制御化に伴い、一台の自動車に搭載される ECU（電子制御ユニット: Electronics Control Unit）の数は年々増加し、また要求される機能も複雑化している。ECU の増加に伴い、実現するソフトウェアにおいては、ソースコード量の増加や ECU 間のネットワーク制御が必要になり、ソフトウェアの品質が全体の性能・信頼性に大きく影響を与えるとともに、設計・検証コストが増大する課題が表面化してきた（図 15-B-14-1）。

これらの課題を解決するため、本事例で採用した「モデルベース開発」と「コード解析」の手法を紹介する。

アルプス電気のモデルベース開発は、MDD（Model Driven Development）に TDD（Test Driven Development）の思想を取り入れ、モデルベース開発とテスト駆動開発を融合させプロセスの洗練化を図った MBTDD（Model Based Test Driven Development、アルプス電気の造語）開発である。従来開発に比べて開発工程が減ることでコストが削減でき、視覚的に理解しやすいモデルを検証することで、検証効率と品質を向上することができる。

コード解析では、市販コード解析ツールと自社開発ツールを組み合わせることで、人手で行っていた検証作業をツールに置き換え、スキルによらない検証が行え、検証コストの削減につなげることができる。

---

<sup>1</sup> 事例提供: アルプス電気株式会社 技術本部 早坂 哲 氏、宇治川 尚悟 氏

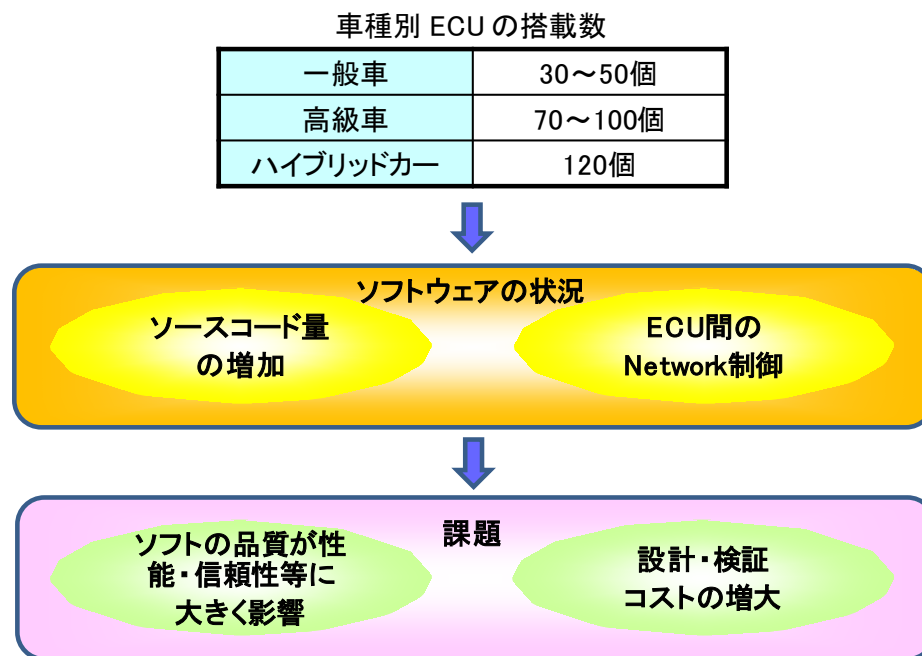


図 15-B-14-1 ECU ソフトウェアにおける搭載数の増大のインパクト

## 2. 技術・手法を導入した理由や経緯

### 2.1. モデルベース開発（MBD：Model Based Development）

年々加速度的に大きくなる車載ソフト規模の市場要求と、開発能力の関係を図 15-B-14-2 に示す。

開発現場での地道な改善活動により開発能力は毎年少しずつ向上しているが、それだけでは近年高まるソフトウェア開発の要求量と要求スピードについていくことができない状況となってきている。かつて車載ソフトの開発現場ではアセンブラベースの開発から C 言語ベースの開発に移行することで大きな開発能力向上のブレークスルーを経験した。しかし、現在ではもはや C 言語ベースの改善の積み重ねでは市場要求について行くことは困難になりつつある。第 2 のブレークスルーとして C 言語ベース開発から、制御系を含む製品開発の手法として注目されている「モデルベース開発」を設計に適用することにした。

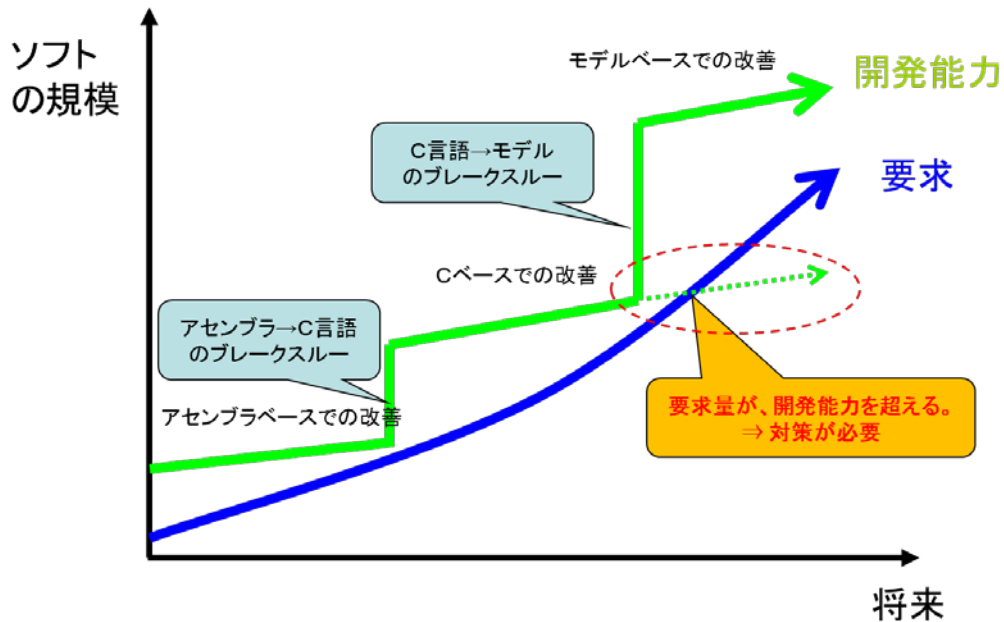


図 15-B-14-2 モデルベースの開発の必要性

一般的にモデルベース開発の目的は、「複雑なシステムを効率的に開発すること」で、下記のような利点がある。

- ① 仕様書の明確化と再利用性の向上
- ② シミュレーションをしながら仕様を決定できる
- ③ 仕様書から自動でコードが生成されるために、バグが入りにくい
- ④ 制御対象と制御装置を同時に開発できる。

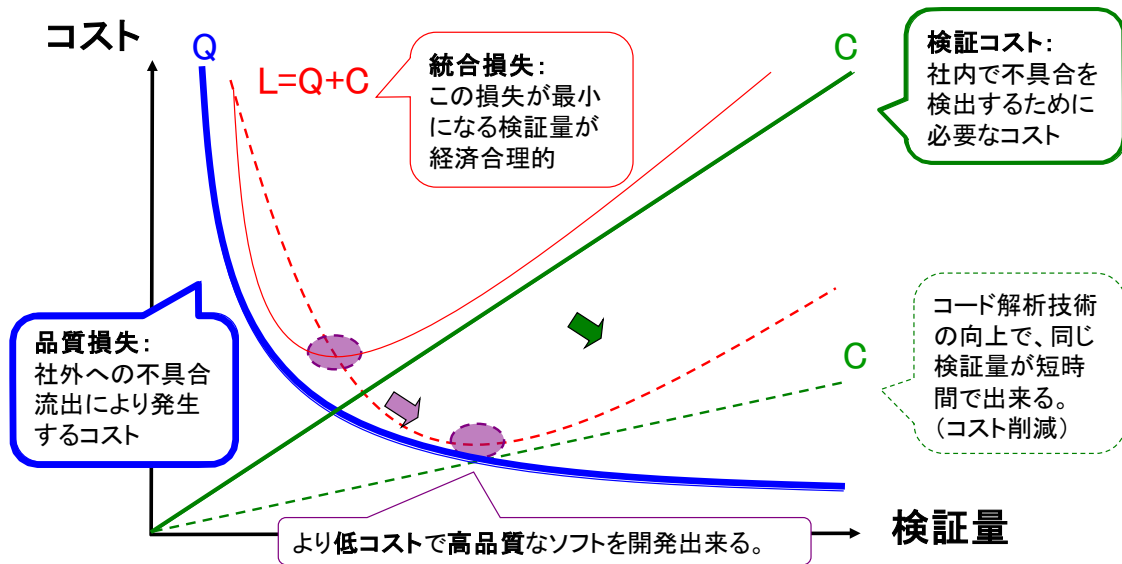
モデルベース開発の手法はソフトウェアの開発規模の増大にも対応できる手法であり、モデルの作成ができると、ソースコードが自動生成されるので、プログラミングのミスが防止でき、品質の向上が見込めるとともに生産性の向上も期待できる開発手法と言える。

## 2.2. コード解析

設計においてはモデルベース開発を適用したが、検証については、下記の理由から各種ツールを利用した「コード解析」を実施することとした。

- ① 人による検証からツールによる検証への置き換え
- ② 顧客要求に応じた各プロジェクトのサポート
- ③ ソフトウェア品質の可視化

コード解析を実施することで、今後増え続けるであろう検証負荷を軽減し、開発効率及び開発品質を向上する。図 15-B-14-3 に示すように、コード解析の目標は、統合損失（品質損失＋検証コスト）を最小にする点を求めると共にさらにそれを小さくすることである。



※コード解析技術向上による「検証量の増加」は、設計者の検証量や検証時間を増やすものではありません。

図 15-B-14-3 コード解析の目標

### 3. スムーズな導入と活動定着のための事前準備や工夫

#### 3.1. ワーキング活動と開発体制

MBD 開発に興味のある有志を中心にワーキンググループ（WG）を構成し、2 週に 1 回程度の頻度で定期的に勉強会や討論会（いわゆる小集団活動）でツールのカスタマイズ等を実施した。MBD の WG 活動については既に 10 年以上継続している。WG 活動の結果として、従来のハンドコーディングと同等のコードサイズを実現する MBD 開発ができるようになった。

また、コード解析においても同様にコード解析 WG を構築し手法の学習や習得に努めてきた。活動は 2008 年 11 月から 2 週に 1 時間程度の周期で実施してきた。

##### 3.1.1. 組織内の役割分担

MBTDD WG、コード解析 WG、コード解析専任チーム、各プロジェクトの役割を図 15-B-14-4 組織内の役割分担に示す。MBTDD WG は MBTDD 技術の教育や各プロジェクトから上がる改善要望から MBTDD 技術の改善を行っている。コード解析 WG はコード解析技術の改善を、コード解析専任チームは各種ツールを使用してコード解析を実行、各プロジェクトは設計者担当分のコード解析を実施と解析依頼の結果を確認することを主に実施する。

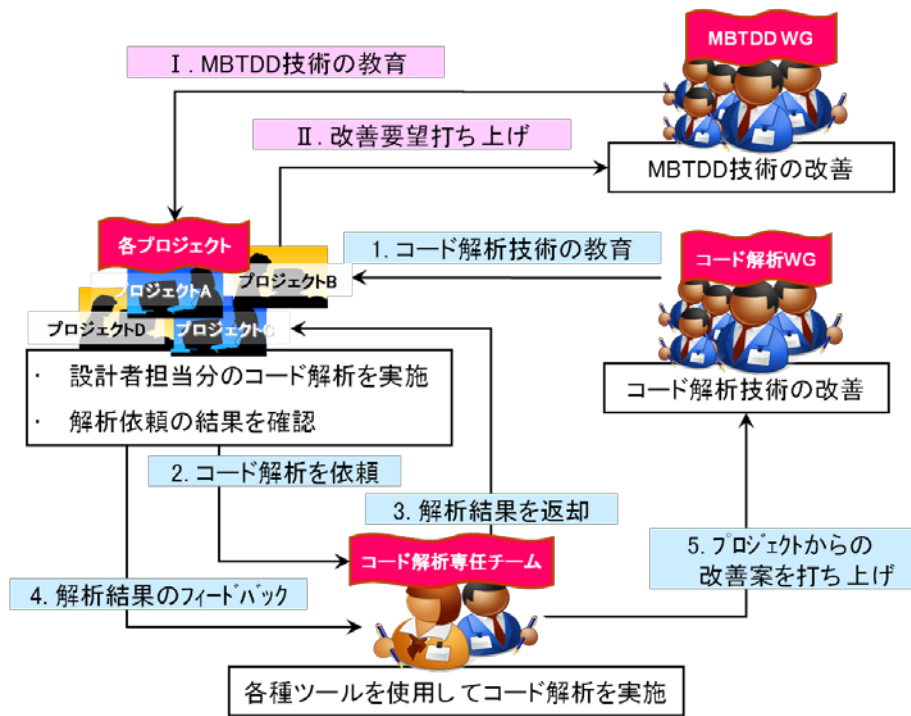


図 15-B-14-4 組織内の役割分担

3.1.2. 開発工程ごとの解析項目

MBTDD WG が主に V 字モデルの左側の開発プロセス、コード解析 WG が主に V 字モデルの右側の開発プロセスの環境整備や教育を担当している。

コード解析技術では、開発工程により専任者による体制を構築しており、単体テストでは設計者によるコード解析体制、結合テストでは専任者によるコード解析の体制とした（図 15-B-14-5）。

解析種別により複数の解析ツールを用途に応じて使い分けている（表 15-B-14-1）。また、図 15-B-14-5 に示すように、コード解析（専任者）が使用する解析ツールと開発工程、コード解析（専任者）が使用する解析ツールと開発工程を、それぞれ決めている。

表 15-B-14-1 コード解析ツール一覧

解析種別	検証項目の例	ツール
ランタイムエラー	配列オーバーフロー、ゼロ割など	Polyspace®
MISRA-C	MISRA-C 2004 対応	QAC®+M2CM
ソフトウェアメトリクス	サイクロマティック複雑度、実行行数	QAC®+自作
コーディングルール	社内コーディングルールへの適用	QAC®+自作
ソフトウェア構造	タスク間インタフェースなど	Imagix4D
コードクローン	コードクローン率	CCFinder

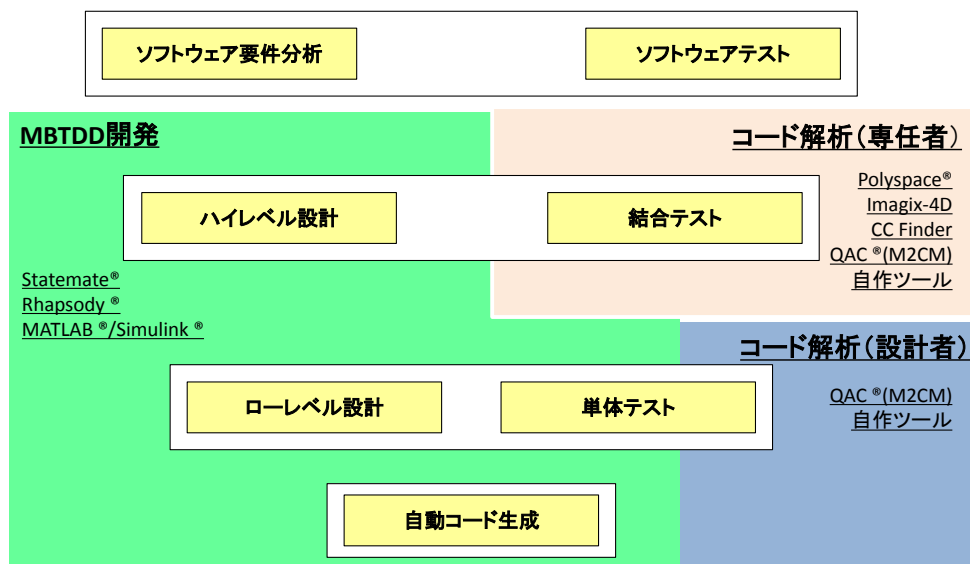


図 15-B-14-5 開発工程ごとの専任体制

### 3.2. 教育の取組

小集団活動を通して教育の資料を作成し教育コンテンツを整備してきた。モデリングのノウハウ、ユニットテストのノウハウ、リファクタリングの各種ノウハウを Tips 集として教育資料としてきている。MBD 開発では現在、1 か月の OFF-JT 教育である研修コース（図 15-B-14-6）を用意しており、この OFF-JT 後 3～6 月の OJT で一人前になることができる。コード解析では 2 日間の研修コースを新人教育などに取り入れている。

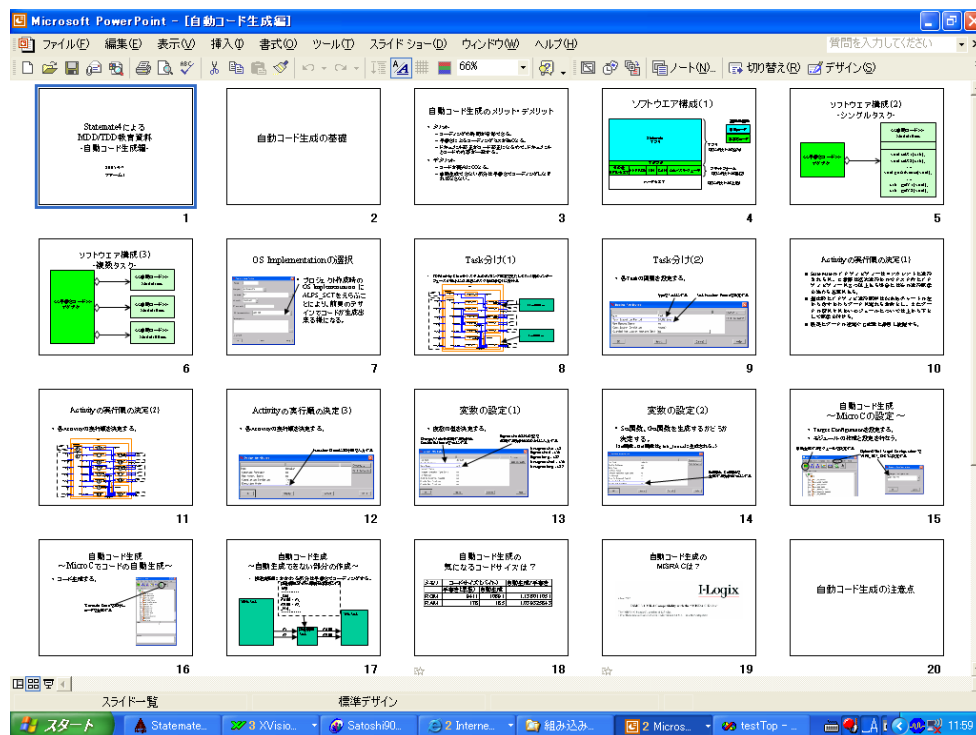


図 15-B-14-6 OFF-JT 教育資料

## 4. 効果測定の方法とその結果

### 4.1. モデルベース開発

#### (1) 適用状況

現在、アルプス電気では適材適所でモデルベースの開発を実施している。適用対象事例を表 15-B-14-2 に示す。適材適所のモデルベース開発では、モデル仕様に基づいて開発プロセスを構築する手法である広義の MBD (Model Based Development) と、組込み制御システム開発の狭義の MBD があり、車載用組込みソフトで狭義の MBD は全体の 20%程度にしか適用できない。このため、残りの 80%については MDD (Model Driven Development) で開発し効率化を目指している。

表 15-B-14-2 モデルベースの開発状況

名称		適用対象項目	手法	使用ツール
広義の MBD	狭義の MBD	制御系アルゴリズム開発	制御ブロックモデリング	MATLAB®/Simulink®
	MDD	アプリケーション開発 (製品機能実装部)	構造化モデリング	Statemate®
		プラットフォーム開発 (ハードウェア制御部)	オブジェクト指向 モデリング	Rhapsody®

#### (2) MBTDD 開発

アルプス電気では MDD (Model Driven Development) に TDD (Test Driven Development) の思想を取り入れ、モデルベース開発とテスト駆動開発を融合させてプロセスの洗練を行う MBTDD (Model Based Test Driven Development、アルプス電気の造語) 開発を実施している。MBTDD 開発では、MDD のモデルシミュレーションの代わりに、モデルから生成されるコードについて直接テストを実施する。ここに TDD の思想を取り入れテストとモデルの洗練を繰り返し実施する。コードではなくモデルを洗練することで本当の意味での設計の洗練になる。

図 15-B-14-7 に人手によるコーディングをしていた従来プロセスと MBTDD プロセスの比較を示す。

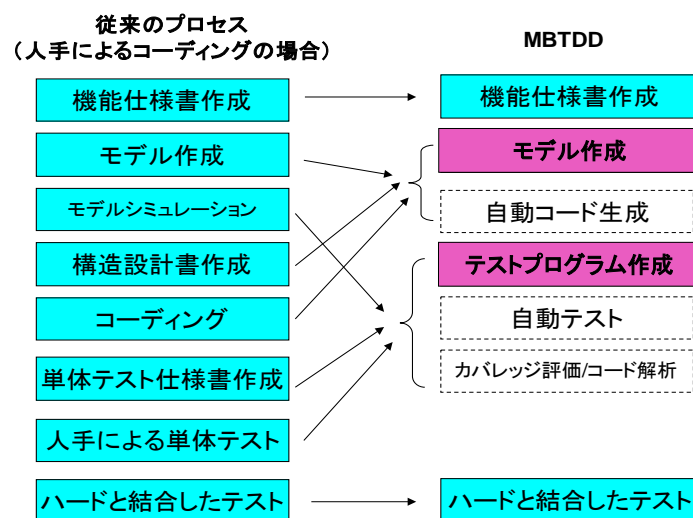


図 15-B-14-7 従来のプロセスと MBTDD プロセスの比較

従来プロセスでもモデルは使用していたものの、その主な目的はシミュレーションによるモデルの動作検証であった。モデルシミュレーションで動作検証が済んだモデルをコーディングするために必要な情報を記述した構造設計書を作成し、それに基づいて人手でコーディングしていた。また単体テストも単体テスト仕様書を作成し、人手による単体テストを実施していた。

MBTDD では、モデルの中に従来の構造設計の情報を埋め込み、最適なコードが生成されるようにカスタマイズしたツールのコード生成機能を用いてモデルからボタン一つでコードを生成している。また単体テスト仕様書をプログラム化し実施を自動化した。

これらの改善によりモデルシミュレーション工程を無くし、モデルができたらボタン一つでコード生成し単体テストプログラムを走らせることで従来のモデルシミュレーション、コーディング、人手による単体テストの工程を無くすことができた。また単体テストの実施を自動化し何度でも簡単に実施できるようにしたことと合わせ、この工程で簡単にテストカバレッジ及び設計者自身によるコード解析もできるような環境を整備した。結果としてモデル作成工程とテスト工程を行ったり来たりしながらモデルの洗練と単体テストの洗練を行うようになった。

### (3) MATLAB®/Simulink®の活用事例

MATLAB®/Simulink®をカーナビディスプレイの開閉制御に活用した例である(図 15-B-14-8)。実機での動作検証前に MATLAB®/Simulink®でシミュレーションし、システム成立性の確認を行った。

また、モデルとコード解析を使った作業(図 15-B-14-11)を繰り返すことで、ロバスト性が高く、高品質のソフト開発を行うことができた。



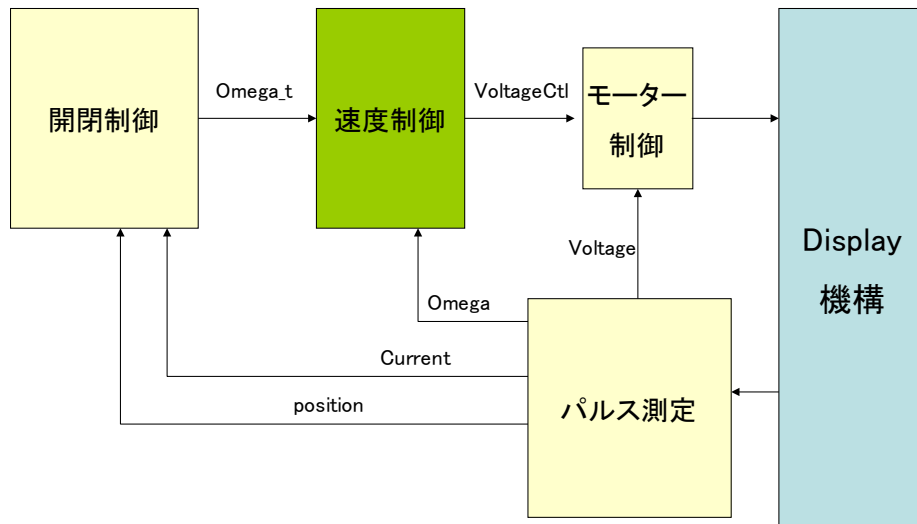


図 15-B-14-8 MATLAB®/Simulink®の活用事例

## 4.2. コード解析

### 4.2.1. コード解析ツールの選定

アルプス電気では複数の市販コード解析ツールと自社開発ツールを組み合わせ使用しており、より精度の高いコード解析を達成するため、市場のコード解析ツールのベンチマーキングを実施しツール選定を行っている。以下にその評価方法と結果を示す。

#### (1) 評価方法

解析ツールの選定では以下の 5 つのステップで評価方法を定義している。

- ① 社内ソフトウェアでよく実装してしまう不具合を選別  
⇒過去の不具合について洗い出しを行い、類型化を図る
- ② 不具合がない 3 種類のソースコードを用意  
⇒3 種類のソースコードを使い解析ツールを評価
  - ・ Statemate®ツールによる自動生成コード
  - ・ Rhapsody®ツールによる自動生成コード
  - ・ MATLAB®/Simulink®による自動生成コード
- ③ 各ソースコードに、選別した不具合を埋め込む  
⇒②のソースコードに意図的に不具合を挿入
- ④ 社内外の解析ツールで不具合理め込みコードを解析
- ⑤ 解析結果を分析し、結果を比較

#### (2) 解析ツールの評価基準

理想的なコード解析ツールとは、①すべての不具合を検出すること（＝不具合検出率が高い）、および、②正しいコードを誤って不具合としない（＝有効警告率が高い）ことである。評価基準を表 15-B-14-3 に示す。

表 15-B-14-3 評価基準

評価項目	計算式	基準
不具合検出率	検出された埋め込み不具合の数/ 埋め込み数	高い方がよい
有効警告率	埋め込み不具合に対する警告数/ 総警告数	高い方がよい

## (3) 評価結果

図 15-B-14-9 に市販のコード解析ツール（ツール A~F）と自社開発ツール（ALPS）の評価結果を示す。自社開発ツールは、特に重要な不具合検出率が一番高く、有効警告率も中間に位置しており、現在のツール選定の妥当性が確認できた。また、同時に、本評価結果をもとに改善点も明確になり、改善することができている。

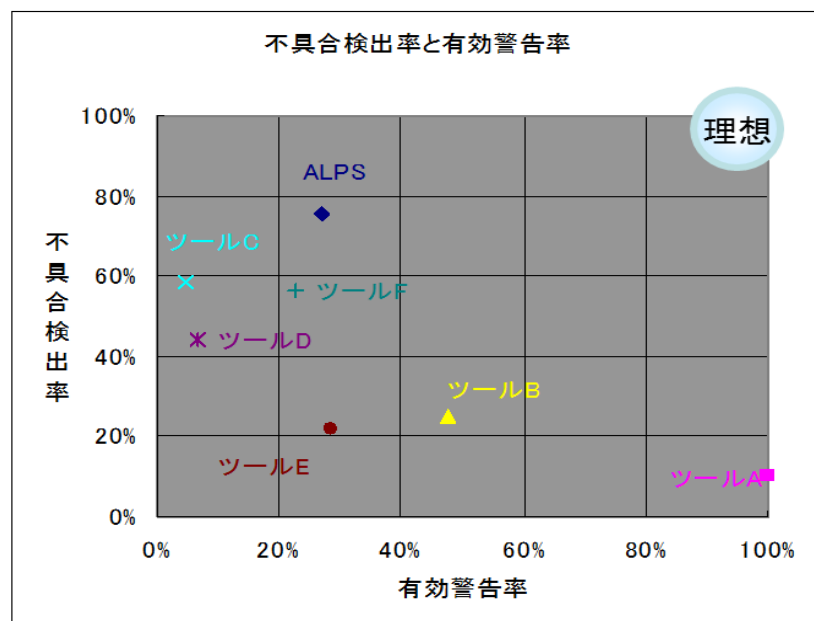


図 15-B-14-9 解析ツールの評価結果

#### 4.2.2. ソフトウェア品質の可視化

##### (1) ソフトウェア品質モデル

ISO/IEC 9126-1 (Software engineering - Product quality - Part1: Quality model) で定義しているソフトウェア品質モデル (表 15-B-14-4) の中から「信頼性、効率性、保守性、移植性」の 4 つの品質特性を使いソフトウェア品質を評価している。

表 15-B-14-4 ソフトウェア品質モデル (ISO/IEC 9126-1)

品質特性	品質副特性	主な内容
機能性	合目的性	目的から求められる必要な機能の実装の度合い
	正確性	
	相互運用性	
	セキュリティ	
	標準適合性	
信頼性	成熟性	機能が正常動作し続ける度合い
	障害許容性	
	回復性	
	標準適合性	
使用性	理解性	分かりやすさ、使いやすさの度合い。 いわゆる「使い勝手」、「使いやすさ」、「操作性」の概念です。
	習得性	
	運用性	
	魅力性	
	標準適合性	

品質特性	品質副特性	主な内容
効率性	時間効率性	目的達成のために使用する資源の度合い
	資源効率性	
	標準適合性	
保守性	解析性	保守(改訂)作業に必要な努力の度合い
	変更性	
	安定性	
	試験性	
移植性	標準適合性	別環境へ移した際そのまま動作する度合い
	環境適応性	
	設置性	
	共存性	
	置換性	
	標準適合性	

##### (2) ソフトウェア品質の可視化の目的

ソフトウェア品質について、4 つの品質特性を 5 段階評価したものをレーダーチャートで可視化している。これにより以下の効果が得られる。

- ① ソフトウェア品質状況を数値化して定量的に把握する
- ② 定量的データに基づき、品質の良否を判断できる
- ③ 品質が悪い場合の改善ポイントが明確になる

また、その結果として、品質のよいソフトウェア開発が可能になる。

- ① 不具合が除去されている
- ② 仕様変更に対応できる
- ③ 再利用が可能である

#### 4.2.3. コード解析 WG の成果

各プロジェクトの品質状況を可視化し、モニタリングすることで、早期に適切な対処を行っている (図 15-B-14-10)。

## M3 技術部コード解析月報 (2014 年 12 月)

2015/01/08

コード解析 WG

## 1. 解析結果推移

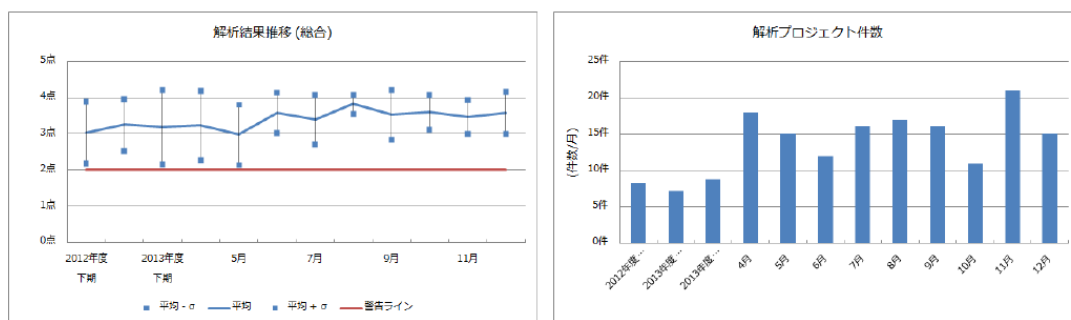


図 15-B-14-10 品質状況のモニタリング

従来レビューで活用していたチェックリストを見直し、コード解析で代用できるものを削減した。その結果、30%以上のチェック項目がコード解析で対応できることがわかり、レビュー時間の削減に大きく貢献した。また、レビューからコード解析に検証手法を変えたことで、設計者のスキルに依存しない、安定した検証が短時間でできるようになった。

## 4.2.4. 複数の市販解析ツールと自社開発ツールの活用

市販ツールには、それぞれ得意な解析分野があるが、一方でアルプス電気のソフト開発では必要としない機能もある。アルプス電気では各市販ツールの解析のためのパラメータの設定をカスタマイズしている。さらに独自のフィルターをかけることで、効率的な解析を実施することが可能となっている。また、市販の解析ツールのカスタマイズで対応できないものはツールを自作し活用している。

## 4.2.5. 自動生成におけるコード解析の必要性

モデルベース開発は、モデル自体の文法チェックや一貫性のチェックが実施でき、また、視覚的に理解しやすい開発手法である。しかし、自動コード生成をするためにはプログラム言語に近い記述も必要であり、人為的なミスやスキル不足による間違いがそのまま自動で生成されたコードに反映されてしまう。そのため、自動生成コードでもコード解析で不具合の検証を行う必要がある (図 15-B-14-11)。なお、制御系アルゴリズム開発には MBTDD 開発を適用していない (表 15-B-14-2) ため、シミュレーション (図 15-B-14-11) が必要だが、MBTDD 開発が適用できるアプリケーション開発やプラットフォーム開発では不要になる。

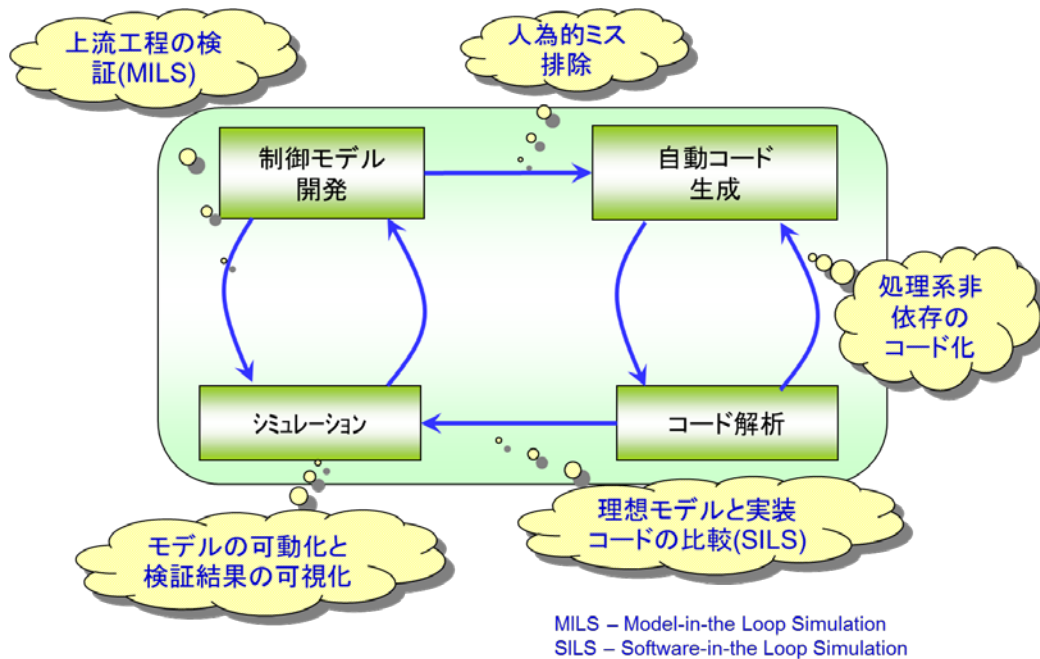


図 15-B-14-11 モデルとコード解析を使った開発(MATLAB®の例)

## 5. 導入時の工夫・苦労や適用途中での課題と対策

### (1) メモリサイズとの戦い

車載用の組込機器のメモリサイズの制限は相当に厳しく、Statemate®やRhapsody®のツールのデフォルト設定では手書きの2倍以上になってしまい、実用に耐えられなかった。この問題は、ツール設定のカスタマイズ等の工夫を行うことで、手書きの時と同等のメモリサイズを実現し、対応できた（表 15-B-14-5）。

表 15-B-14-5 同じモデル（設計）でのメモリサイズ比較

メモリ種別	手書き	自動生成（ツール）	
		デフォルト	カスタマイズ
ROM	1	2.4	1.2
RAM	1	2.0	1.1

また、同一仕様における従来開発と MBTDD 開発でのメモリサイズ比較を表 15-B-14-6 に示す。

表 15-B-14-6 実際のプロジェクトに適用した時のメモリサイズ比較

プロジェクト	メモリ種別	従来開発（手書き）	MBTDD 開発
A	ROM	1	0.6
	RAM	1	0.9
B	ROM	1	1.0
	RAM	1	0.8

## (2) カスタマイズマニュアルの作成

ツールのカスタマイズの必要性については(1)でもふれたが、ツールを提供しているベンダからの情報だけでは満足に行くカスタマイズができず、数年間の試行錯誤の連続で何とかカスタマイズできたのが実状である。このため習得した技術を社内の有志と共有するためツールのカスタマイズマニュアルを作成し、更なる改良に役立つようにしている。

## (3) 抵抗勢力に対する取り組み

MBTDD への移行については、従来手法での成功体験を多く持っているベテランからは強く抵抗された。このため導入推進派が抵抗勢力のプロジェクトに入り一緒に開発することで、一つひとつ確実に成功事例を作り理解を得るようにした。

現在は、車載システムについては 100%水平展開を実施済みである。

## 6. 技術や手法の導入後の改善活動、今後の課題

MBD 開発後は、モデルの中に設計情報を書き入れているので、あいまいになりがちな設計書を作成する必要はなくなった。ソースについても自動生成されるので、コーディング作業は不要になった。今後、モデルベース開発では、モデル用のライブラリの充足、および、マイコン依存部やデザインパターン等の実装を簡単にするフレームワークを整備することで、再利用率を上げる予定である。コード解析では、有効警告率を 25%から 40%程度まで向上させることを目標に、Polyspace®、Imagix4D などのツールの解析設定のカスタマイズ、および、自作のツールの改良を実施する。

## 7. 結果と考察

## (1) 品質について

新手法を導入する際に、品質の目標をバグ件数半減としていたが、結果は表 15-B-14-7 に示すように大幅に改善できた。バグの数が減ったことと、バグが出ても原因の特定や対策ができるようになりバグ対策の仕事が劇的に減った。打ち合わせの多くを占めていたバグ対策が激減し、モデルやテストの洗練に多くの時間を割く文化が育成された。

表 15-B-14-7 品質の結果

プロジェクト	バグ件数
A	約 1/4
B	約 1/3

## (2) 生産性について

新手法を導入する際に、開発速度の目標を2倍とされていたが結果は表 15-B-14-8 に示すように改善できた。

表 15-B-14-8 開発速度の結果

プロジェクト	実績
A	約 2.5 倍
B	約 2.0 倍

従来レビューで活用していたチェックリストを見直し、コード解析で代用できるものを削減した。その結果、30%以上のチェック項目がコード解析で対応できることがわかり、レビュー時間の削減に大きな改善が図れた。

## (3) 開発工程ごとの専任体制による効果

開発工程ごとに専任者によるコード解析の体制を構築した（図 15-B-14-5）ことにより、結合テスト段階で、専任者による高度な解析を行うことができ、ソフトウェアの信頼性を確保できた。

また、コード解析専任チームを設置したことで以下の効果も得られた。

- ① 市販ツールを使用するための技術が集約できる。⇒解析ノウハウの蓄積
- ② 市販ツールのライセンスが有効に活用できる。
- ③ 市販ツールの選定を一定の基準で行える。
- ④ 設計者からの改善要求や解析の傾向を分析し、WG を中心にコード解析技術の改善がスムーズに行える。

## (4) 検証への影響

以前はソースコードをコーディングしていたためコード網羅率などを意識して検証していたが、新手法導入後はソースコードが自動生成されることから、ほぼモデル中心の検証へとシフトしている。コードについては設計者用のコード解析ツールやコード解析専任チームで指摘された事項についてピンポイントで確認テストを実施している。

## 8. まとめ

車載向け ECU ソフトウェアの開発において、モデルベース開発とコード解析を導入することにより、品質向上と生産性向上の目標を十分達成することができた。具体的には、Statemate®、Rhapsody®、MATLAB®/Simulink®等のモデルベースツールをカスタマイズし TDD と組み合わせて使用することと、QAC®、Polyspace®、Imagix4D、自作ツール等のコード解析ツールを適材適所に使用することで、従来の人手によるコーディング及び人手による検証に比べ、劇的に開発効率と品質を改善した。また、レビューからコード解析に検証手法を変えたことにより、設計者のスキルに依存しない、安定した検証が短時間でできるようになった。

参考文献

- [1] アルプス電気株式会社 宇治川 尚悟： 車載向け ECU のコード解析とモデルベース開発への取り組み MATLAB EXPO 2013

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)