

モデルベース開発の MICU への適用

Application of Model-based Development to MICU

宇 治 川 登*
Noboru UJIGAWA

木 下 修 平*
Shuhei KINOSHITA

山 崎 勝 幸*
Masayuki YAMAZAKI

要 旨

車体電装部品のソフトウェア開発において、設計仕様品質向上を目的としてモデルベース開発の導入を進めている。その導入にあたって、モデルベース開発によるプロセス、および開発ツールが必要であった。そこで、開発上流の要求分析から、その達成手段の定義に至るモデルベース開発プロセスを構築した。また、開発を効率的に進めるために必要なツールを開発した。この構築したプロセスを Multiplex Integrated Control Unit の開発に試行した結果、従来の開発プロセスにて作成した制御仕様に対して、仕様の未定義、仕様不備、仕様の曖昧表現といった不完全性を 35% 削減することができた。

ABSTRACT

The introduction of model-based development is being promoted to increase specification quality in the development of software for body electrical parts. This introduction requires model-based development processes and development tools. Therefore, a model-based development process was constructed that covers from analysis of requirements in the early stages of development to definition of the methods for achieving those requirements. In addition, the tools needed to efficiently promote development were developed. The constructed process was applied to trial development of a Multiplex Integrated Control Unit, and resulted in a 35% reduction in incompleteness (undefined, conflicting or ambiguous specifications) compared to control specifications created using the previous development process.

1. はじめに

自動車における電子制御技術の重要性が高まる中、ソフトウェアの規模、複雑さは増加の一途をたどっている⁽¹⁾。一方で、電子制御ユニットに対する品質向上、開発期間短縮、コスト削減といった要求はますます高まっている。そのような中、車載ソフトウェアの品質向上を目的として、近年、モデルベース開発 (MBD) の導入が進められている⁽²⁾⁻⁽⁴⁾。

ドアロック、ワイパ、ヘッドライト、ターンシグナルランプ、ハザードランプといった車体系電装機能の制御を統合した Multiplex Integrated Control Unit (MICU) は、制御仕様をサプライヤへ提示し、その仕様を基にサプライヤが部品開発をおこなっている。

しかし、サプライヤへ提示する制御仕様において、仕様定義の基となる要求と仕様の明確な定義、流用設計仕様の

適合性を見極め方、そして曖昧さのない設計をおこなうことが課題であった。

上記の課題解決に向け、MBD による開発プロセスを構築し、MICU 開発の試行を実施した。本稿では、構築したプロセスと開発したツール、および開発の試行から得られた効果について述べる。

2. プロセス構築のねらい

MBD による開発プロセス構築のねらいを以下に定めた。

- (1) 要求と仕様を分離し、複数の観点で要求を分析し、要求定義の不足を検出できる仕組みとする。
- (2) 要求と仕様の関係を明確にし、それぞれの根拠を関連付けて残せる仕組みとする。
- (3) 準形式的な仕様記述言語を用い、仕様の曖昧さを削減できる仕組みとする。

*四輪 R&D センター

3. 開発プロセスの構築

3.1. プロセスの概要

構築したMBDによる開発プロセスは、分析観点、およびレベルから構成される (Fig. 1)。

3.1.1. 観点

開発上流での要求定義の不足を検出できるようにするため、複数の観点で分析をおこなう。分析観点を以下に示す。

- ・振舞: システムに求める振舞, 状態遷移を定義
- ・構造: システムの静的な構成, 構造を定義
- ・要求: システムに求める要求内容を定義

3.1.2. レベル

(1) システム定義 何 (what) を開発するかを明確にし、要求定義するシステムを特定する。ここでは、システムの範囲、役割、機能、他システムとの関係を定義する。

(2) 要求定義 特定したシステムに求める振舞、構造、要求を定義する。

(3) 達成手法定義 システムを実現するための具体的な達成方式 (how) を定義する。達成方式は、要求定義にて定義した振舞、構造、要求と関連付けされ、それぞれに根拠 (why) を付与する。

3.1.3. 成果物

観点、およびレベルごとに、作成する成果物を Fig. 2 に示す。主に、System Modeling Language (SysML)⁽⁵⁾ のダイアグラム、および MATLAB/Simulink, Stateflow⁽⁶⁾ にて成果物を作成する。ここでは、SysML にて定義された複数のダイアグラムの中から、各分析観点、およびレベルごとに最低限必要なものを選定した。

3.2. プロセス詳細

分析観点、およびレベルの枠組みで構成したプロセスの詳細なワークフローを Fig. 3 に示す。

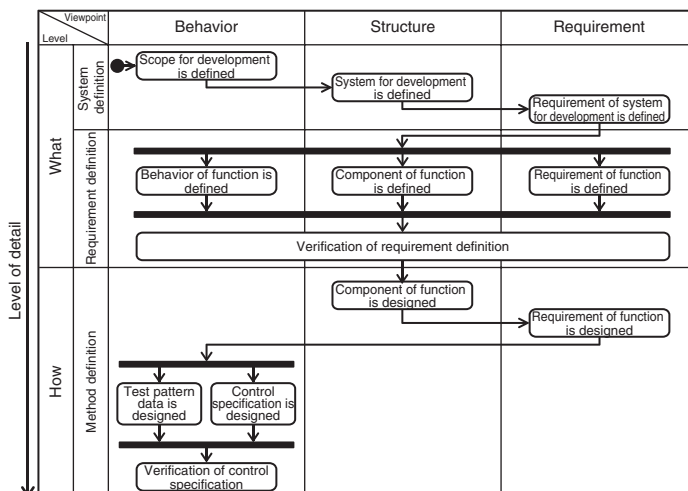


Fig. 3 Workflow of development

3.2.1. システム定義

(1) 開発対象範囲の定義 ユースケース図を用いて、開発の対象とするシステムを特定する。また、特定したシステムに関するユーザ、他システムを定義する (Fig. 4)。

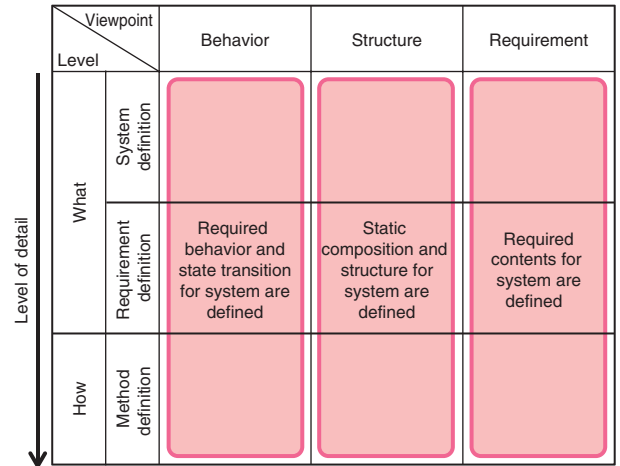


Fig. 1 Relationship between process and analysis viewpoints

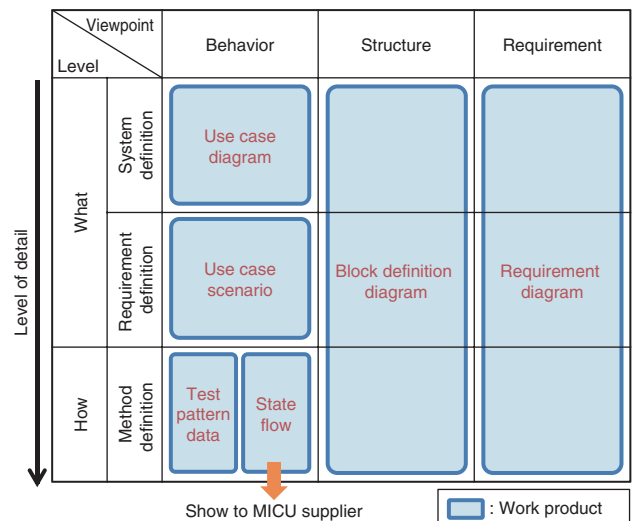


Fig. 2 Work product of each viewpoint and level

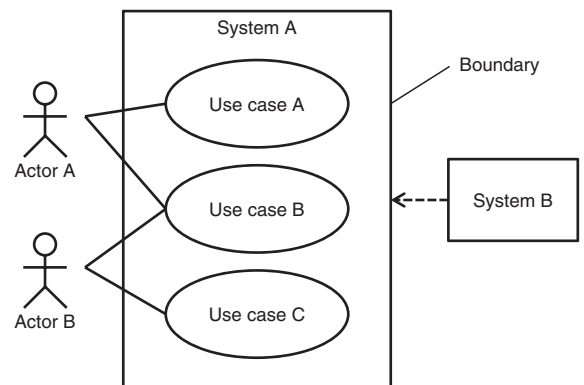


Fig. 4 Example of use case diagram

- ・ システムの利用者の定義: システムを利用するヒト (運転手など), およびモノ (他システムなど) をアクターとして定義する.
- ・ システムの機能の定義: アクター視点でシステムに必要な機能をユースケースとして定義する.
- ・ システムの利用者と機能の関係の定義: ユースケースと, ユースケースを実行する (利用する) アクターを関連付ける.
- ・ システムの境界の定義: ユースケースをシステムの境界枠で囲み, 開発対象のシステムを定義する.

(2) 開発対象システムの定義 ユースケース図を基に, 開発対象システムをブロック定義図として定義する (Fig. 5).

- ・ ユースケース図の境界枠をブロック定義図の構造ブロックとして定義し, システムの概要, ボデータタイプなどの情報を付与する.

(3) 開発対象システムの要求定義 ユースケース図を基に, 開発対象システムの要求を要求図として定義する (Fig. 5).

- ・ ユースケース図のユースケースを要求図の要求ブロックとして定義する.

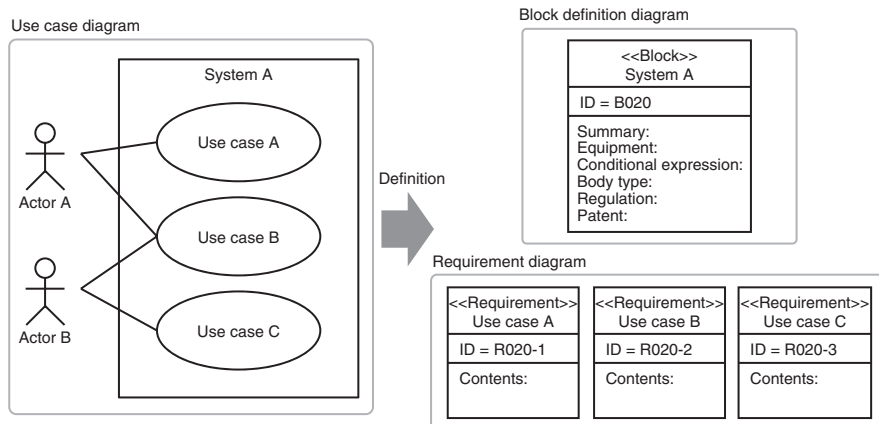


Fig. 5 Relationship of each diagram in system definition process

Scenario	
ID	UC020-1-1
Summary	...
Actor	Driver
Order	Contents
Precondition	Ignition is OFF
1	Driver turns on ignition
2	System outputs relay
3	End of Scenario
Postcondition	Relay is ON

Fig. 6 Example of use case scenario

3.2.2. 要求定義

(1) 機能の振舞分析, および定義 ユースケース図を基に, 機能の振舞をシナリオにて定義する (Fig. 6).

シナリオでは, ユースケース図に登場するアクターとシステムのやりとり, およびその時のシステムの振舞を定義する. ここでは, システムとして想定されるさまざまなシチュエーションでの振舞を定義する.

- ・ 基本シナリオの定義: 各ユースケースに対し, 関連するアクターとシステムのやりとりを基本シナリオとして定義する.

- ・ 拡張シナリオの定義: 基本シナリオの事前条件, 事後条件, 事後条件に至る手順, アクターのいずれかが異なるパターンを導出し, 拡張シナリオとして定義する.

(2) 機能の構成要素の定義 システム定義工程で定義したブロック定義図を基に, システムに必要な構成要素をブロック定義図に定義する (Fig. 7).

システムを構成する要素を, 構造ブロックとして定義する. また, 構造ブロック間の関係を定義する. 定義手順を以下に示す.

- ・ 要求を満足させるために必要なシステムの構成要素を, 入力, 制御, 出力の観点で構造ブロックとして定

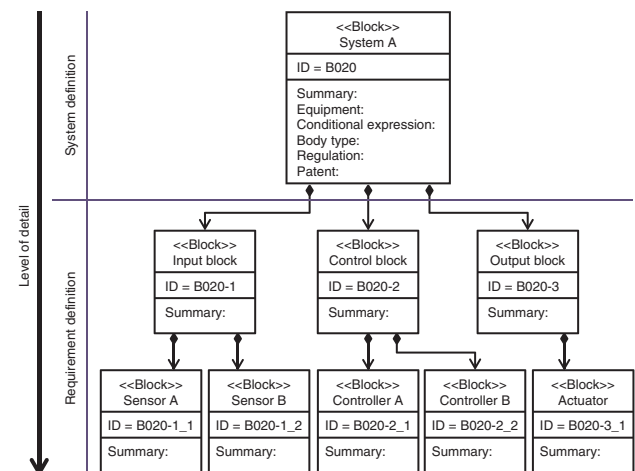


Fig. 7 Example of block definition diagram in requirement definition process

義する。

- ・ 定義した構造ブロックを上位構造と下位構造の関係で関連付ける。
- ・ 構造ブロック間でやりとりする情報を入出力情報として各構造ブロックに定義する (Fig. 8)。

(3) 機能要求の分析, および定義 システム定義工程で定義した要求図を基に, 機能ごとの要求を要求図に定義する (Fig. 9)。

機能に求める要求を, 要求ブロックとして定義し, 要求ブロック間の関係を定義する。定義手順を以下に示す。

- ・ システムの利用シーンを考慮し, システム定義工程で定義したユースケースに対する詳細要求 (操作, 条件, 出力, 要求競合時の優先順位など) を抽出して, 要求ブロックとして定義する。
- ・ 定義した要求ブロックを上位要求と下位要求の関係で関連付ける。
- ・ 各要求の根拠を関連付ける。

(4) 要求定義の検証 Fig. 10 に示すように, 要求定義工程にて作成した成果物を照合し, 以下の観点で検証する。

(a) 要求図とシナリオの照合

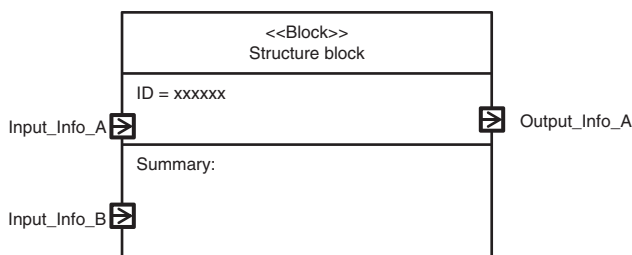


Fig. 8 Example of definition of input and output information to structure block

- ・ シナリオを満足させるための要求が定義されている。
- ・ 全てのシナリオが要求に関連付いている。

(b) シナリオとブロック定義図の照合

- ・ シナリオを満足させるための構造が定義されている。
- ・ シナリオを満足させるための入出力情報が定義されている。

(c) ブロック定義図と要求図の照合

- ・ 要求を満足させるための構造が定義されている。
- ・ 全ての要求が構造に関連付いている。
- ・ 要求を満足させるための入出力情報が定義されている。

3.2.3. 達成手法定義

(1) 機能構成要素の仕様化 要求定義工程にて定義したブロック定義図を基に, 具体的な構成部品をブロック定

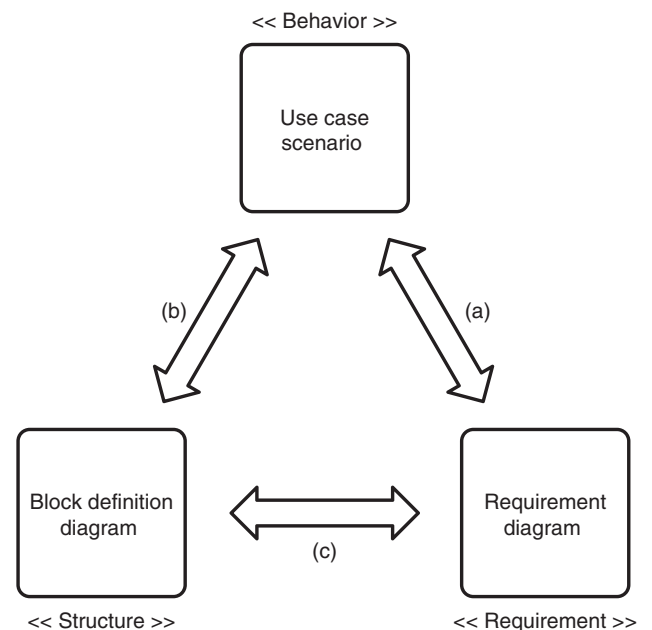


Fig. 10 Verification of work product

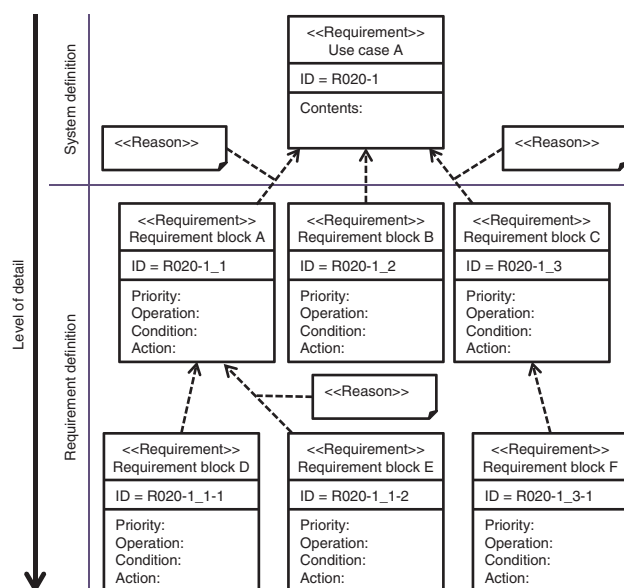


Fig. 9 Example of requirement diagram in requirement definition process

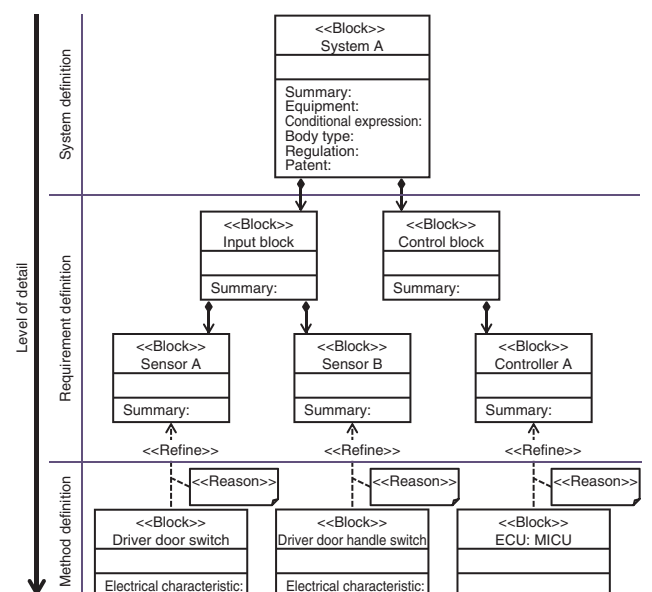


Fig. 11 Example of block definition diagram in method definition process

- ・要求定義工程にて、入力、制御、出力の観点で定義した構造ブロックに対して、具体的な構成部品（スイッチ、ECU、アクチュエータなど）を構造ブロックとして定義する。この際、選定根拠の情報やそのエビデンスを、構造ブロックに関連付ける。この関連付けられた情報により、マイナーチェンジや仕向け地ごとの仕様変更といった差分開発において、流用設計仕様の適合性を見極めを可能とする。

- (2) 機能要求の仕様化 要求定義工程にて定義したブロック定義図、および要求図を基に、達成方式を検討し、要求図の要求と関連付ける。

(3) テストパターンの作成 要求定義工程で定義したシナリオ、およびブロック定義図にて定義した入出力情報を基に、テストパターンを作成する (Fig. 12).

制御に必要な詳細イベントと、イベントの成立によって発生する事象を、状態、および遷移条件にて定義する。

- | Time
(sec) | Signal | | | | | | |
|---------------|----------|----------|----------|----------|----------|----------|----------|
| | Input | | | | | Output | |
| | Signal_A | Signal_B | Signal_C | Signal_D | Signal_E | Signal_F | Signal_G |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1.00 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 2.00 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 3.00 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 4.00 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5.00 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 6.00 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 7.00 | 0 | 0 | 2 | 0 | 1 | 0 | 1 |
| 8.00 | 0 | 0 | 2 | 0 | 1 | 0 | 1 |
| 9.00 | 0 | 0 | 2 | 0 | 1 | 0 | 0 |
| 10.00 | 0 | 0 | 2 | 0 | 1 | 0 | 0 |

Fig. 12 Example of test pattern data

- ・ 必要に応じて状態および遷移条件の追加, 削除をおこなう.

(5) 制御の検証 Stateflow, およびテストパターンを照合し, 定義した制御仕様を検証する.



4.1. 入出力情報の同期

そのため、リスト形式のユーザインタフェースにて入出力の属性情報を編集し、その編集した内容をモデルへ反映できるツールを開発した。本ツールを用いて、モデルの内容をリスト形式へ出力することもできる (Fig. 14)。

このツールを利用することにより、入出力情報の付与に
関わる作業時間をツールの導入前に対して 1/5 に短縮した。

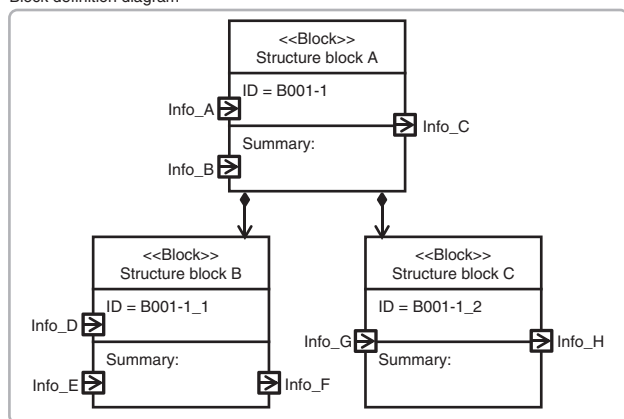
要求ブロック、構造ブロックには、ユニークな ID を付与した。しかし、システムの規模に比例して各ブロックの数は増加するため、モデル上で ID を一つずつ定義することに多くの作業時間を必要とした。また、モデルの変更を繰り返すことによって、ID の重複が発生した。

– 91 –

付与するツールを構築し、「全ての ID を付与しなおす」、「付与されていない要素(要求, 構造ブロック)に対してのみ ID 付与をおこなう」、「全ての ID を消去する」といった作業を自動でできるようにした。

このツールを利用することにより, ID 重複の発生を完全に防止するとともに, ID 付与に関わる作業時間をツールの導入前に対して 1/2 に短縮した。

Block definition diagram



List of input and output information

Structure ID	Structure name	Signal name	In/Out	Value	Initial value	Description
B001-1	Structure block A	Info_A	In	...	...	...
B001-1	Structure block A	Info_B	In	...	...	...
B001-1	Structure block A	Info_C	Out	...	...	...
B001-1_1	Structure block B	Info_D	In	...	...	...
B001-1_1	Structure block B	Info_E	In	...	...	...
B001-1_1	Structure block B	Info_F	Out	...	...	...
B001-1_2	Structure block C	Info_G	In	...	...	...
B001-1_2	Structure block C	Info_H	Out	...	...	...

Fig. 14 Synchronous tool of input and output information

5. プロセスの検証

構築したプロセスを検証するため, MBD による開発プロセスに沿って MICU 開発の試行を実施した。構築したプロセスの良し悪しを判断する評価尺度として, 制御仕様に対するサプライヤからの仕様確認件数を利用した。

サプライヤからの仕様確認の内訳から, 仕様未定義や未記載, 仕様不備, 仕様曖昧(複数の解釈ができる)に分類される仕様確認の件数を削減できれば, 従来の開発プロセスと比べて, 構築したプロセスの方が, 開発品質が良いとした。

MICU サプライヤの協力のもと, MBD による開発プロセスを適用して作成した制御仕様を, 従来の開発と同様の観点でサプライヤにて確認をおこなった。結果, 仕様未定義や未記載, 仕様不備, 仕様曖昧に分類される仕様確認の件数は, 35% 削減した。

6. まとめ

今回構築した MBD による開発プロセスでは, 振舞, 構造, 要求の三つの観点でシステムを分析することにより, 何のために, どのようなシステム構成で, 各構成要素がどのような振舞をするのかを整理することができた。さらに, 整理した各成果物を照合することで, 要求の品質を高めることができた。また, 要求と要求根拠, および達成手法を関連付けることで, 流用設計仕様の適合性を見極めを可能とすることができた。

今後, 構築した開発プロセスを量産開発へ展開して, 着実な品質向上を図るだけでなく, 効率化の観点での改善を引き続き進める。また, 複数部門での成果物の運用ルールの整備や, MBD による開発プロセスの普及に取り組んでいく。

参考文献

- (1) 車載ソフト巨大化に立ち向かう, 日経 Automotive Technology, 2007/11 号, 日経 BP 社, p.82-83(2007)
- (2) <http://sec.ipa.go.jp/reports/20120323.html>, 2012/8/23
- (3) 水谷将和, 八幡忠孝, 亀井雄一, 早瀬憲児, 金田匡弘: ボディ系量産 ECU へのモデルベース開発の適用, 三菱自動車テクニカルレビュー, No.22, p.56-59(2010)
- (4) 渡部知樹, 吉野弘規, 塩ノ谷陽介, 関孝幸: 電子制御 AWD システムにおける新 MBD ソフトウェア開発プロセス, Honda R&D Technical Review, Vol.23, No.1, p.22-28
- (5) <http://www.omg.sysml.org/>
- (6) <http://www.mathworks.co.jp/products/>

著者



宇治川 登



木下 修平



山崎 勝幸