組込みシステムへの派生 MBD 適用に向けたパラメータ設定方式の検討

秋下 耀介
株式会社日立製作所
yosuke.akishita.nw@hitachi.com

大島 浩資
株式会社日立製作所
kosuke.oshima.kn@hitachi.com

若林 昇
株式会社日立製作所
noboru.wakabayashi.mu@hitachi.com

川上 真澄
株式会社日立製作所
masumi.kawakami.ch@hitachi.com

要旨

組込みソフトウェア開発においては，ソフトウェア開発規模が増大しつつ短納期化する中で効率的に開発を行うため，開発者は派生開発により新規開発量を削減している．また，モデルベース開発(MBD: Model Based Development)を取り入れることでより効率的なアプリケーション開発を実現する． 特に制御ソフトウェアでは，ハードウェアの種類により派生製品が存在しており，類似した機能を持つソフトウェアを開発することが多い．そこで，MBD における派生開発では，ハードウェア仕様の異なる部分のパラメータ化によりモデル内で制御ロジックとハードウェア仕様を分離し，パラメータ変更のみで派生製品を実現する方法が考えられる． 一方でパラメータ設定においては，不具合を起こさないため設定対象となるパラメータの適切な選択や，パラメータ数が多い場合には設定の効率化が必要となる．

本論文では，仕様書とパラメータ間のトレーサビリティに基づき，仕様に対して変更すべき適切なパラメータを選択することで派生開発を実現するパラメータ設定方式を検討する．これを実現するため，過去の実プロジェクトの仕様書やパラメータを分析しパラメータ設定ツールを開発した．パラメータ設定ツールを実プロジェクト適用した結果，仕様に応じて適切なパラメータを選択可能であるほか，パラメータ設定工数の 94%が削減可能であり，工数削減に有効であることがわかった．

1. はじめに

輸送用機器や産業用機器のような組込みシステムに搭載するソフトウェア開発では，周辺装置の制御や情報処理機能の高度化に伴い，ソフトウェア開発規模が増大している[1]．このような状況下で，信頼性や安全性への要求を満たす高品質なソフトウェア開発技術の重要性が増している．一方でソフトウェアの要求は，顧客の要求やシステム特性，周辺装置仕様などの要因で変化するため，これらを満たすソフトウェアを効率良く開発する必要がある．特に制御ソフトウェアでは，ハードウェアの種類に応じて派生製品が存在しており，類似した機能を持つソフトウェアを開発することが多い．一般に，類似した機能を持つソフトウェアを効率良く開発するために，派生開発の考え方を取り入れて開発を行う[2][3]．また，MBD を取り入れることでより効率的なアプリケーション開発を実現する．MBD における派生開発では，ハードウェア仕様の異なる部分のパラメータ化により制御ロジックとハードウェア仕様を分離し，パラメータ変更のみで派生製品を実現することが考えられる[4][5]．

本論文で例として検証した組込みシステムでは，ベースソフトウェアを選定し，要求機能に対して新規開発ではなく，ベースソフトウェアの既存モデルを再利用し，モデル内の制御ロジックの変更かパラメータ値を設定して動作を変更することで，各プロジェクトの開発を行ってきた．プロジェクトにより要求の機能に対しパラメータ設定で対応できる量は異なるが，パラメータを設定することによって効率的に派生製品を実現している．ただし，ベースからパラメータを変更する際には，ある仕様に対し複数のモデルの中から適切なパラメータを選択する必要がある．しかし，仕様書とパラメータ間のトレーサビリティが確保されておらず，開発者は経験に基づいてベースソフトウェアの中から変更すべきパラメータを判断しており，不適切なパラメータを選択する可能性がある．また，ハードウェアとの組み合わせ過程では，重量など計画値と実際の値が変わる場合があり，シミュレーションでは再現できない不確定要素となっているため，テスト段階において設計者とは異なる設定者がパラメータの調整を行う場合がある．派生開発を繰り返す中で，パラメータ間には暗黙知の関係をもつものも存在しており，それらを考慮できずパラメータ

を変更し，動作不良となる場合がある．さらに，パラメータの設定作業は人手による計算・入力を伴うため，計算ミスや変更モレなどの人的ミスが発生する可能性があるほか，パラメータが多数存在する場合は，全パラメータの設定に工数を要する．

本論文では，適切かつ効率的にパラメータを設定するため検討したパラメータ設定方式と，開発したパラメータ設定ツールについて報告する．2 章では，対象とするシステムの概要と前提とする開発プロセスを示し，本論文が扱うパラメータ設定の課題について述べる．3 章では，課題に対して検討したパラメータ設定方式と，これに基づき開発したパラメータ設定ツールについて提案する．4 章では，パラメータ設定ツールを実プロジェクトに適用した結果を示し，課題の解決に対して考察する．最後に 5 章で本論文のまとめと今後の課題について述べる．

2. 背景と課題

2.1. 対象システム

本論文の対象として，以下の特徴を持つ組込みシステムを定義する．

- 開発対象システムは，ソフトウェアが動作する装置として一つ以上のコントローラを有する．
- 開発対象システムは，コントローラ以外の機器として一つ以上のシステム内機器を有する．システム内機器はコントローラと接続され，コントローラに入力を与えるか，コントローラによる制御の対象となる．システム内機器としては，速度のような情報を取得するセンサや，モータのような物理的運動を行うアクチュエータが接続される．
- 開発対象システムは，一つ以上の外部システムと接続する．外部システムはシステムと情報を送受信する．
- 開発対象システムは，システム内機器および外部システムから受信した情報に基づいて，システム内機器を制御する．
- 開発対象システムは，外部システムが要求する情報を，システム内機器および外部システムから受信した情報に基づいて算出し，外部システムに送信する．

以上のシステムの構成例を図 1 に示す．図 1 の開発対象システムは，コントローラ，システム内機器 1, 2 で構成され，外部システム 1, 2 と接続する．開発対象システムはシステム内機器および外部システム 1, 2 から受信した情報に基づいてシステム内機器を制御し，外部システム 1, 2 が要求する情報を送信する．

図 1 の組込みシステムでは，コントローラが制御ソフトウェアを有しており，MBD においては，例として図 2 のようなブロック線図により設計される．図 2 は，アクチュエータを外部システムの入力値に基づき制御する機能を実現するブロック図の例であり，制御の最大値をパラメータとして持つ．また，センサ故障検知機能を有しており，センサ故障を判断するための下限値や，センサ有無による機能切り替えのパラメータを持つ．

本論文で例として検証した組込みシステムは，複数の制御パラメータを有しており，制御パラメータに基づき動作が変化する．パラメータの例としては，電圧などシステムの要求仕様値を元に換算して導出した制御リミット値や，機器制御信号の出力時間のような制御値や，センサの有無による機能切り替えのようなパラメータなどが挙げられる．

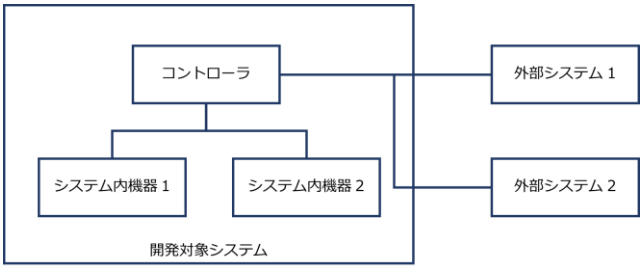

図 1 開発対象システムの構成例

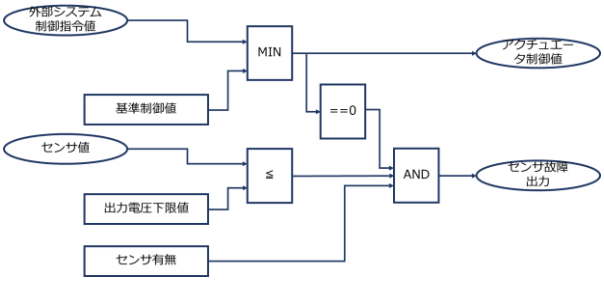

図 2 制御ソフトウェアのブロック線図例

2.2. 派生 MBD

2.1 節のシステムについて，前提となるシステム開発工程を定義する．図 3 のような V 字開発プロセスにおいて MBD を適用する．MBD ツールを使用することで，仕様化，シミュレーション，コード生成，品質や再利用性を改善する[6]．各プロジェクトの開発時には，ベースソフトウェアを選定し派生開発を実施する．なお，MBD で作ったものについては，モデルが仕様書に代わるため，積極的に再利用され得る[7]．

【P1: 要求定義】
顧客から提供される要求仕様書やヒアリングをベースにシステムの要求を定義する．
【P2: 基本設計】
システムへの要求に基づき，外部システムとのインタフェースを設計する．そして，要求および外部システムとのインタフェースを実現するための，ハードウェア，ソフトウェアそれぞれの仕様を定義する．このとき，ベースとするソフトウェアを選定し，ソフトウェアの変更箇所を抽出する．ソフトウェア変更箇所の抽出においては，各機能について比較表を用いつつ P1 の結果に基づき変更の必要有無を検討する．
以降プロセスではソフトウェアについて述べる．
【P3: 機能設計】
P2 の結果に基づき，ソフトウェア全体の機能を厳密に定義する．
【P4: 詳細設計】
P3 の結果に基づき，ソフトウェアの各機能を関数に至るまで詳細化する．なお，開発対象システムのソフトウェアについて，制御設計の効率化のためアプリケーションを MBD で，そのほか性能要求が厳格な個所は C 言語で開発する．MBD ツールとしては社内製ツールを利用する．社内製 MBD ツールは，モデルで使用するパラメータを一覧管理する機能を有しており，パラメータ値を設定する場合には一覧から値を入力する．
各機能についてベースソフトウェアがある場合には，開発者が機能の変更箇所に応じて該当するモデルを探し出し，処理内容やパラメータ名から制御ロジックやパラメータの変更有無を判断し，それらを修正する．
【P5: コーディング】
P4 の結果に基づき，プログラミングを実施する．このとき，MBD ツールで設計したアプリケーションについてはコードを自動生成する．
【P6: 単体テスト，P7: 結合テスト，P8: システムテスト】
P2 から P4 の仕様に基づきソフトウェアをテストする．テスト対象が関数単体からソフトウェア全体に至るよう段階的にテスト対象を拡大する．
【P9: 試運転】
顧客先環境において，システムが要求定義どおりの動作となることをテストする．

なお，MBD ツールの活用においては，ベースソフトウェアのモデルについて，モデルのパラメータを設定して動作を変更する． パラメータ設定手順として，開発者は仕様に基づいて動作を変更したいモデルから変更すべきパラメータを選択し，要求仕様などから当該パラメータの数値計算を行い，MBD ツールへ計算した値を手入力する．

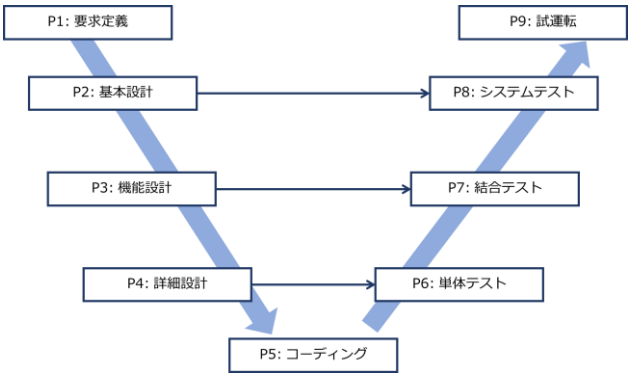

図 3 システムの開発工程

2.3. パラメータ設定における課題

2.2 節に示す派生 MBD により，モジュールの新規開発より少ない工数で開発が可能であるが，本論文の対象システムにおけるパラメータ設定では以下の課題を定義する．これらは，図 3 の P4(詳細設計)においてパラメータを設定するにあたって発生する課題である．

【課題 1】パラメータの不十分な考慮による不適切なパラメータ選択
パラメータを変更する際には，複数のモデルの中から適切なパラメータを選択する必要がある．適切なパラメータを選択するためには，上位仕様からパラメータまでトレ

ーサビリティが確保されている必要がある. しかしながら, 派生開発を繰り返す中で, トレーサビリティが確保されない場合があり, 仕様変更に対して不適切なパラメータを選択する可能性が考えられる.

本論文で例として検証したシステムにおいては, 仕様書とパラメータ間のトレーサビリティが確保できておらず, 開発者は経験に基づいてベースソフトウェアの中から変更すべきパラメータを判断していた. このシステムにおいては, パラメータは 1456 個存在し, 類似した名称や数値を持つパラメータが多数存在するため, 不適切なパラメータを選択する可能性がある. また, パラメータの選択には工数を要する. このシステムでは, 全パラメータのうち 400 個以上をベースソフトウェアから変更する.

【課題 2】暗黙知の関係を持つパラメータの変更漏れ

パラメータの中には, 仕様として明示されていないが同時変更が必要なパラメータなど, 暗黙知の関係を持つパラメータが存在する場合がある.

本論文で例として検証したシステムにおいては, システムのある要求値に基づき計算されたパラメータなどがあるが, 同じ要求値を参照して相関関係にあるパラメータが存在する. さらに, 各モジュールの動作のみならず, システム全体の動作を考慮して設定する必要があるパラメータも存在する. 派生開発を繰り返すほか, テスト結果に基づく変更を繰り返す中で, これらが仕様化されず暗黙知となっている. しかしながら, 同時に変更すべきものやシステム全体の動作を把握してパラメータを設定することは, 熟練した設計者でなければ困難である. さらに, ハードウェアとの組み合わせ過程では, 重量など計画値と実際の値が変わる場合があり, シミュレーションでは再現できない不確定要素となっているため, 設計者のみならずテスト実施者など複数のパラメータ設定者が存在する. したがって, 暗黙知を把握できておらずパラメータの変更漏れが発生する可能性がある.

【課題 3】パラメータ計算・入力間違い

パラメータ設定においては, 変更すべきパラメータについて要求仕様に基づいた数値計算や計算値の MBD ツール入力を行う. しかしながら, これらの作業は人手によって実施されるため, パラメータの計算間違いや, ビット位置の間違いなどパラメータ入力間違いが発生する可能性がある. また, 本論文で検証したシステムにおいては, パラメータ数が 1456 個であることから, 全パラメータを計算・入力するには工数を要する.

3. 派生 MBD 適用に向けたパラメータ設定方式

3.1. パラメータ設定方式の概要

本論文では, 2.3 章で述べた課題に対し, 以下のキーアイディアを提案する.

【キーアイディア 1】上位仕様書-パラメータ間のトレーサビリティに基づく派生開発: パラメータまでトレーサビリティを確保することで仕様に基づいた根拠あるパラメータ設定を実現し, 課題 1 で示すような経験に基づいたパラメータ選択や, 課題 2 に示すようなシステム全体の動作を考慮しないパラメータ設定を防止する. さらに, パラメータ選択に伴う調査工数を削減する.

【キーアイディア 2】入力インターフェースの共通化: 同じ入力値を参照して設定すべきパラメータについては, 要求値の入力インターフェースを共通化することで, 課題 2 に示すような同時変更が必要なパラメータの変更漏れを防止する.

【キーアイディア 3】自動計算, 自動入力: 従来の人手による作業を自動化することで属人性を排除し, 課題 3 に示すような計算や入力間違いを防止する. さらに, 計算や入力に伴う工数を削減する.

上記アイディアを実現する方法として, 図 4 に示すパラメータ設定ツールを開発する. 以降の各節では, 各キーアイディアの実現過程や, パラメータ設定ツールで実装した機能の詳細を述べる. なお, 実際の製品システムを対象にドキュメント調査やパラメータ分析などを行いパラメータ設定ツールを開発した.

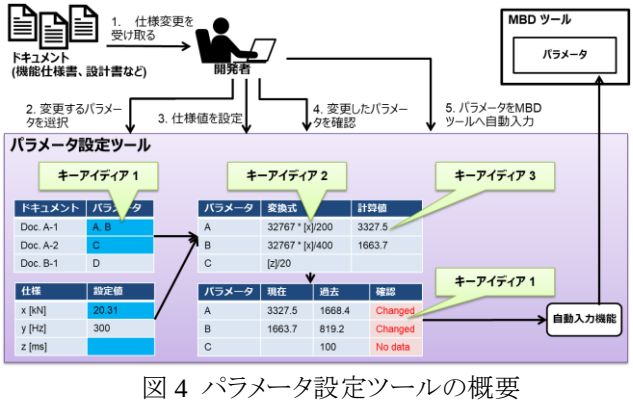

図 4 パラメータ設定ツールの概要

3.2. 上位仕様書-パラメータ間のトレーサビリティに基づく派生開発

パラメータと仕様書間のトレーサビリティを明確にすることで，ある仕様変更に対して変更すべきパラメータを示す．ハードウェア仕様の異なる部分をパラメータ化しているため，類似した機能をもつ製品であれば，パラメータ変更のみで同じ制御ロジックを用いて派生製品を実現することができる．

トレーサビリティを明確にするため，我々は対象システムについて 23 種のドキュメント調査を行うほか，過去の仕様変更やパラメータ変更実績を分析した．さらに，システムとしての動作考慮の必要があり安易に変更できないパラメータなどの暗黙知について，変更実績をベースに開発者へ繰り返しヒアリングを行った．この結果，対象システムのパラメータ 1456 個のうち 558 個はプロジェクトの中で変更する可能性が高いが，残りのパラメータは変更する可能性が低く，変更する場合にシステムとしての動作が不安定になる可能性を考慮して変更可否を議論すべきパラメータであるとわかった．

パラメータ設定ツールにおいては，各パラメータについて仕様とのトレーサビリティを示すタグを埋め込むほか，ベースソフトウェアのデフォルト値から変更する際には変更可否の議論が必要なパラメータを明示する．

3.3. 入力インターフェースの共通化

同じ要求値を参照して相関関係にあるような同時変更が必要なパラメータについて，パラメータ設定における入力インターフェースを共通化することで，パラメータの変更漏れを防止する．入力インターフェースを作成するため，我々はパラメータ値とパラメータの計算式を分析し，同時変更の必要性があるパラメータ候補をリスト化した．その後，開発者にそれらの関係性をヒアリングし，仕様化されていないパラメータの関係を明確にした．

3.4. 自動計算・自動入力

パラメータ設定ツール内に計算式を実装することで自動計算を実現し，MBD ツールへの入力機能を実装することで自動入力を実現する．

表 1 従来とツール適用後のパラメータ設定作業

作業項目	従来	ツール適用後
ベースソフトウェア調査によるパラメータ選択	ベースソフトウェアのパラメータを調査し，経験に基づき変更可否を判断	パラメータ設定ツール内のパラメータ変更可否情報やトレーサビリティを元にパラメータ選択
パラメータ設定根拠の調査	パラメータ設定設定根拠となるドキュメントを探す．または経験を根拠とする．	トレーサビリティに基づきドキュメントを確認
パラメータ計算	ベースソフトウェア調査やパラメータ設定根拠の調査で得られた値から手計算	仕様との差分やトレーサビリティに従い設定値を入力し自動計算
MBD ツールへ入力	MBD ツールへ計算したパラメータを手入力	パラメータ設定ツールから自動入力
パラメータ設定検証	パラメータを一つずつ確認	ツール内で変更内容を一括確認
手順・環境構築等の情報共有	散在したドキュメントやツールを共有	ツール内に必要な情報を集約
パラメータ設定不良対応	パラメータ選択間違いや計算・入力間違いなどの修正	-

3.5. パラメータ設定ツールを用いた開発手順の定義

適切なパラメータを選択し，誤りなく入力を行うため，以下の手順でパラメータ設定ツールを運用することを考えた (各手順は図 4 の手順に対応する).

1. 仕様変更を受け取る．パラメータを調整する場合であっても，仕様書の変更から行うことで，要求仕様を意識して設定する．
2. 仕様変更のあったドキュメントに基づいてパラメータを選択する．
3. 電圧などの仕様値を入力する．

4. パラメータが変更できていることを確認する.
5. MBD ツールへ自動入力する.

上記手順を, 2.2 節の開発プロセスおける P4 に組み込む. パラメータ設定ツール導入前後で, 表 1 のようにパラメータ設定作業が変化した.

4. 評価

4.1. 評価方法

パラメータ設定方式の有効性は, 実プロジェクトへのツール適用によって評価する. 定量的評価として, 従来のパラメータ設定作業に関する工数とパラメータ設定ツールを用いた場合の作業工数を計測・ヒアリングして比較評価する. さらに定性的評価として, 開発者へのヒアリングを実施しパラメータ設定ツールの効果や改善点のフィードバックを得る.

ツール適用前後での作業工数計測・ヒアリングは同じソフトウェア開発者へ実施する. また, ツール適用前後のプロジェクトは同じソフトウェアをベースとしているが, プロジェクトが異なるため, 変更量や変更するパラメータは異なる(同じパラメータも含まれる).

4.2. 評価結果

パラメータ設定ツール適用の工数削減効果を表 2 に示す. 表 2 の「ベースソフトウェア調査によるパラメータ選択」から「パラメータ設定検証」までは, 1 パラメータあたりの各作業時間を計測し, プロジェクトの中で変更する可能性が高いパラメータの数 (558 個)を乗算, 人日換算した値である. 「手順・環境構築」と「パラメータ設定不良対応」は, 各作業時間をヒアリングし抽出した値である. パラメータ設定ツール適用の結果, 1 プロジェクトあたりのパラメータ設定作業について 18.36 人日を要していたものが 1.44 人日となり, 従来に対して 94%の工数を削減した.

また, パラメータ設定ツールを使用した開発者へヒアリングを実施した結果, 表 3, 4 のようなフィードバックを得た. パラメータ設定の作業負荷削減や, 変更するパラメータ・参照ドキュメントの明示, 一度の入力によるパラメータ変更が実現できている旨の回答を得た. 一方で, ツールへの入力に関する改善や, パラメータ変更に伴う確認に関する機能拡張などの要望を得た.

4.3. 考察

従来に対して 94%の工数を削減できることから, ベースソフトウェアについてモデル内のパラメータ変更で派生製品を実現する場合には, パラメータ設定ツール適用が工数削減に有効であると言える. またパラメータ設定間違いは起きていないことがわかる.

課題 1 に対し, パラメータ設定者はトレーサビリティに基づきドキュメントを参照しつつパラメータを設定していることから, 不適切なパラメータ選択を防止しているとわかる. 課題 2 に対し, パラメータ設定ツールは変更するパラメータの明示や設定値入力に伴う自動計算により, 従来の暗黙知の関係を持つパラメータについて変更漏れを防止しているとわかる. 課題 3 に対し, パラメータ設定ツールは要求仕様などの設定値を元にパラメータを自動計算し, MBD ツールへ自動入力しており, パラメータの入力・計算間違いを防止しているとわかる. ただし, パラメータ設定ツールにはユーザビリティの観点で改善点があることが判明しており, 今後アップデートが必要である.

本論文では組込みシステムの1種についてパラメータの分析やツール開発を行った. 一方で本論文のアプローチは他の組込みシステムについても同様に適用可能であると考えており, 今後他の派生 MBD プロジェクトへの展開により効果を検証し, 手法の汎用化を図る.

5. まとめ

組込みソフトウェア開発では, 開発効率化のため MBD ツールを用いるほか, 派生開発によりソフトウェア資産を流用する. 特に制御ソフトウェアでは, ハードウェアの種類により派生製品が存在しており, 類似した機能を持つソフトウェアを開発することが多い. そこで派生開発を実現する方法として, ハードウェア仕様の異なる部分のパラメータ化によりモデル内で制御ロジックとハードウェア仕様を分離し, モデル内のパラメータを設定してモジュールの動作を変更することが考えられる. 一方でパラメータ設定においては, 不具合を起こさないため設定対象となるパラメータの適切な選択や, パラメータ数が多い場合

には設定の効率化が必要となる.

本論文では, パラメータ設定を適切かつ効率的に行うためのパラメータ設定方式を検討した. パラメータ設定方式では, 仕様書とパラメータ間のトレーサビリティに基づき, 仕様に対して変更すべき適切なパラメータを選択することで派生開発を実現する. またこれを実現するためパラメータ設定ツールの開発に取り組んだ. パラメータ設定ツールにおいては, 各パラメータと仕様書のトレーサビリティや暗黙知となっているパラメータ間の関係を明確化することで, パラメータの設定間違いを防止した. さらに, 自動計算や MBD ツールへの自動入力によりパラメータの設定工数を削減した. パラメータ設定方式の有効性を確認するため, ツールを実プロジェクトへ適用した結果, パラメータは適切に設定できており, パラメータ設定工数の 94%が削減可能であることがわかった. したがって, ベースソフトウェアを選定し, モデル内のパラメータ値を設定して動作を変更する場合には, 本論文で提案するパラメータ設定方式が工数削減や不具合防止に有効であると言える.

今後の課題として, 派生 MBD におけるパラメータのトレーサビリティ管理方法を検討することを挙げる. 本論文では, 派生 MBD 適用に向けたパラメータ設定方式を実現するため, 実際の製品システムについてパラメータを分析しトレーサビリティを明確にした. しかし, システムのさらなる機能高度化や試験結果に応じ新たなパラメータが拡充される場合には, 拡充されたパラメータのトレーサビリティの精度や粒度などにばらつきが発生し, その後の派生製品の品質への影響が見込まれる. そのため, パラメータのトレーサビリティについて管理方法を検討し, 後の派生製品の品質を維持していくことを考える.

表 2　1 プロジェクトあたりのパラメータ設定ツールの適用効果(人日)

作業項目	従来	ツール適用後	効果
ベースソフトウェア調査によるパラメータ選択	3.00	0.10	−2.90
パラメータ設定根拠調査	3.00	0.10	−2.90
パラメータ計算	1.20	0.02	−1.18
MBD ツールへ入力	1.55	0.00	−1.55
パラメータ設定検証	6.00	1.20	−4.80
手順・環境構築等情報共有	0.52	0.02	−0.49
パラメータ設定不良対応	3.10	0.00	−3.10
計	18.36	1.44	**−16.92**

表 3 パラメータ設定ツールの良かった点

開発者	良かった点
A	・パラメータ設計作業が楽になった. ・チェック/変更するパラメータがツール内での明示により削減されている.
B	・参照するドキュメントが示されているため探す手間が省ける.
C	・パラメータが一覧化できる. ・試験中にパラメータの一括変更をやりやすくなる.

表 4 パラメータ設定ツールの改善点

開発者	改善点
A	・パラメータ設定部分が見づらい.
B	・ある時点のパラメータと現在のパラメータをツール内で比較したい.

参考文献

[1] 経済産業省, “2012 年版ものづくり白書 第 2 章 我が国ものづくり産業が直面する課題と展望” 2012.

[2] Pohl, K., Böckle, G. and van der Linden, F. J.: Software Product Line Engineering, Springer-Verlag(2005). 林 好一, 吉村健太郎, 今関 剛(訳):ソフトウェアプロダクトラインエンジニアリング ソフトウェア製品系列開発の基礎と概念から技

法まで. 株式会社エスアイビー・アクセス, 2009.
[3] 筒井賢, “プラットフォーム開発, XDDP 開発, そしてSPL 開発へ 〜 移行のきっかけ(取り巻く環境)と成功/失敗のステップ 〜”, 第4回 アフォード・フォーラム, 2013.
[4] Dziobek, Cristian. “Functional variants handling in simulink models.” MathWorks Automotive Conference 2008, 2008.
[5] Polzer, Andreas, Stefan Kowalewski, and Goetz Botterweck. "Applying software product line techniques in model-based embedded systems engineering." 2009 ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software. IEEE, 2009.
[6] Liebel, G., Marko, N., Tichy, M., Leitner, A., & Hansson, J. “Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice.” Software & Systems Modeling, 17(1), 91-113, 2018.
[7] IPA, “組込みシステムの先端的モデルベース開発実態調査報告”, 2012.