Vol.2014-SLDM-165 No.7
Vol.2014-EMB-32 No.7
2014/3/15

ハードウェア仕様変更を考慮した ソフトウェアプロダクトライン型開発手法の提案

飯田 隆博†

ソフトウェアプロダクトライン(SPL)は、要求仕様からフィーチャモデルを用いて可変性を選択することで、ソフトウェアを効率的に開発するための手法として注目されている。しかし、自動車用の複雑な制御システムでは、あるハードウェアの仕様変更が他のハードウェアのサイズやギア比等の物理値変換比率に影響を与え、ソフトウェアの変更点調査に多大な工数がかるため、SPL の効果が十分に発揮できない。本研究では、ハードウェア構成要素間、ハードウェア-ソフトウェア間、ソフトウェア構成要素間の３つの仕様依存関係マトリクスを組み合わせて用いることで、ハードウェア仕様変更が間接的に影響を及ぼすソフトウェア仕様変更の追跡を可能とする手法を提案する。本手法は従来の SPL の開発プロセスの拡張であり、複雑な制御システムでも開発効率向上を見込める。

Development Method of Software Product Line to Deal with Hardware Specification Changes

TAKAHIRO IIDA†

Software Product Line (SPL) is an effective method to develop software. In the SPL process, software is developed by selecting variants on a feature model according to the requirements. However, the conventional SPL is not effective with hardware specification changes which occur in developments of control systems such as automotive electric brake systems. These changes happen frequently without functional specification changes. Because hardware specification changes affect other hardware specifications of conversion ratio like gears or size, and it takes a lot of steps to survey of software specification changes.In this paper, we propose the software development method to treat hardware specification changes in SPL. In this method, we use matrices which show dependences between each specification of hardware and software, to trace effects of a hardware specification change to other hardware or software specifications. These matrices make the tracing easy and enable to treat frequent hardware specification changes. The proposed method is expanding method of the conventional SPL process, and software development productivity obtained with SPL is maintained.

1. はじめに

近年、自動車制御システムの高度化・複雑化に伴ってソフトウェアの規模も増大しており、ソフトウェア開発効率の向上が急務となっている。自動車制御ソフトウェアでは、カーメーカ毎の仕様の違いや仕向地毎の規格や法規制等への対応により、製品のバリエーションが非常に多くなることが、開発工数の増大に拍車をかけており、開発工数が増大するおそれがある。

ソフトウェアプロダクトライン(SPL：Software Product Line) [1]は、ソフトウェア資源の再利用により開発を効率化する手法として注目されている。SPL では製品のロードマップを元に、フィーチャモデル[2]を用いることにより、機能構成を共通部と可変部に分けて開発を行うことで、ソフトウェアの再利用性を高めることができる。設計者は要求から可変部を選択しソフトウェアを構築する。

しかし、機能仕様の変更を伴わずにハードウェアの仕様変更を繰り返す製品の開発では、ソフトウェア変更が多発する。あるハードウェアの仕様変更が他のハードウェアのサイズや配置、制御比率等に影響を与えるためにソフトウェアに対しても広範囲にわたる影響を及ぼすおそれがあるため、詳細レベルでの仕様変更はさらに顕著になる。そのため、ハードウェアの仕様を考慮した開発手法が必要となる。

SPL において、ハードウェアを取り扱う取組みはこれまでも行われてきた。[3]では、ハードウェアに依存するプラットフォーム部と、ハードウェアに依存しないアプリケーション部の 2 層構成のソフトウェアを用意し、2 種類のフィーチャモデルによって取り扱う手法を提案している。

また、[4]ではカーナビゲーションシステムのように、同一のハードウェア構成であっても、異なるソフトウェアを構築することが可能であるシステムに対して、ハードウェアとソフトウェアそれぞれの視点でフィーチャモデルを組み合わせて、ある製品用の新たなフィーチャモデルを作成する手法を提案している。

[5]では、ハードウェアも含めた全体を SysML[6]で記述したシステムモデルと SPL の連携を取る事例について報告している。また、システム全体のモデルから SPL を実施した例として、[7]ではエンジンシステム全体を概略のシステムモデルで表すことで、製品系列の起点として活用している。

既存の研究では、ハードウェアとソフトウェアはシステム上明確に分かれていることが前提であり、それらの詳細な繋がりについては、考慮されていない。しかし、仕様変

† (株)日立製作所　日立研究所
Hitachi Ltd., Hitachi Research Laboratory

Vol.2014-SLDM-165 No.7
Vol.2014-EMB-32 No.7
2014/3/15

更が繰り返される開発では、フィーチャモデルやアーキテクチャの仕様変更まで遡った場合、SPL による開発効率向上の効果が得られなくなる。

本研究では、組込み制御システムへの SPL 適用において、ハードウェア仕様の変更に対応する手法を提案する。

2. SPL による従来の開発手法

多品種開発に効果的な手法のひとつである SPL (Software Product Line)[1]では、各製品の共通部と製品個別の可変部をコア資産として開発、各製品はコア資産を組み合わせて開発することにより製品系列全体の開発工数を低減できる。SPL を成功させるために、適切な単位で共通部と可変部を定義し、長期間管理を続ける仕組みが必要となる。

SPL では一般的にフィーチャモデルをもちいて、コア資産の管理を行う。フィーチャモデルは、システムや製品の構成要素(フィーチャ)をツリー形式で表現することにより、製品の構造を表現することができる。フィーチャモデルでは、製品毎に入れ替える必要のある機能を選択肢として、製品毎に要否が異なる機能をオプションとして表現することで、単一の製品だけでなく製品種別全体に必要とされるフィーチャを全て列挙できる。

設計者はフィーチャモデルでフィーチャを選択することによって、製品の仕様を決定できる。新たな機能の開発が必要か否かの確認もフィーチャモデルによって行うことができ、不要な重複開発を無くすことができる。

図 1 に、従来の SPL による開発プロセスの概略を示す。

まず、設計者は製品開発の要求に従い、フィーチャモデルの選択及び、ハードウェア仕様の決定を行う。そして、ハードウェア仕様を元に、ハードウェアに依存するフィーチャの選択を実施する。仮に、フィーチャモデルに記されていない機能がある場合、新規部品の開発を実施することになる。

次に、フィーチャモデルの選択に従い、ソフトウェア部品を選択し、製品アーキテクチャを決定する。ソフトウェア部品は、あらかじめ SPL 全体のアーキテクチャに適合するように設計しておくことで、部品選択における IF 等の仕様の再設計は不要となる。

選択されたソフトウェア構成に対して、製品の仕様を盛り込むことで、ソフトウェア・コンポーネントの設計開発がなされる。一般的に SPL では、コア資産と呼ばれるソフト部品に対して、値を設定することで製品コードが生成される。もしも、応答性やリソースの制約が厳しい場合は、自動化を介さず、手動による設計がなされる。

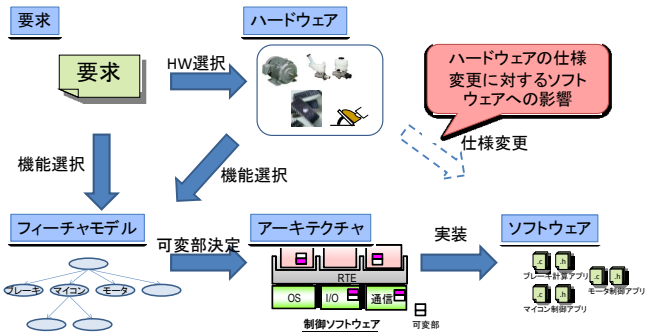

図 1 従来の SPL 開発プロセス例

3. ハードウェア仕様変更によるソフト仕様への影響

SPL では、システムの特徴をフィーチャモデルで表し、それらを実装するソフトウェアを構築することにより、効率的な再利用開発を行うことができる。しかし、従来の方式は、ひとつのハードウェアの仕様変更に対して従属的なハードウェアの変更が必要になる制御システムを取り扱うためには課題がある。本節では、いくつかのシステム例を取り上げ、ハードウェア仕様変更の取り扱う上での課題を示す。

3.1 電動ブレーキシステム

自動車の電動ブレーキなどでは、ハードウェアの開発を進めながらソフトウェアを開発していくため、サイズの制約や、踏み心地、騒音などを対策するためハードウェア仕様の変更と対応したソフトウェアの変更が繰り返される。またハードウェアを構成する各部品の結びつきが強いため、ハードウェアの一部の変更が広範囲の変更につながる。図 2 では相互影響が複雑な制御システムの例として、電動ブレーキシステムを示す。電動ブレーキシステムでは、ブレーキペダルの踏み込み量を元に CPU で計算されたブレーキ力を計算する。計算されたブレーキ力はモータを回転させボールねじ等に伝えることで前後運動となり、マスタシリンダのピストンを動かす。マスタシリンダに圧力がかけられることで、要求通りのブレーキ力が液圧として発生する。さらに電動ブレーキシステムは、横滑り防止装置やブレーキキャリパ、タイヤへと物理的に繋がっており、これら部品の経年劣化や動作中の振動などを考慮した、制御が必要となる。これら部品間に密接なつながりがあるため、

図 3 に電動ブレーキシステムにおいて、仕様変更の影響が波及する例を示す。モータの回転音が問題になる場合、モータの径や筐体の変更で解決を図る。ここで、モータの仕様変更はモータ制御ソフトの変更につながるが、ソフトウェアの変更で解決できない場合は、他のハードウェアに追加の変更を加える。もしモータを変更したならば、モータの回転をピストン運動に変えるボールねじが変わること

Vol.2014-SLDM-165 No.7
Vol.2014-EMB-32 No.7
2014/3/15

があり、これらはマスタシリンダが発生させる液圧に影響する。マスタシリンダの仕様変更はブレーキペダルの動作に影響を与えうる。さらに、システムの外にあるブレーキキャリパの経年劣化等によって、ブレーキシステムが発生させるべき摩擦力の計算に影響がでる。

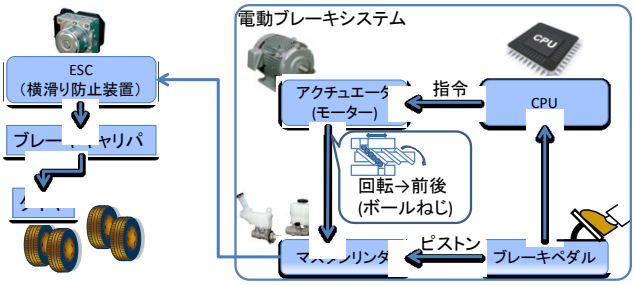

図 2 電動ブレーキシステムの構成

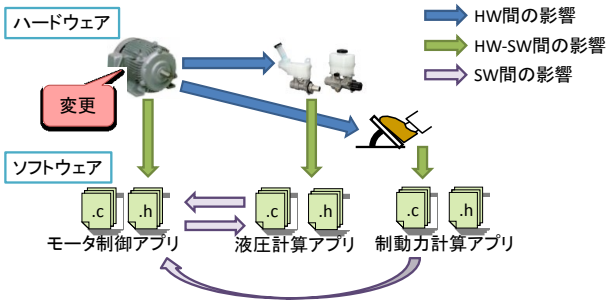

図 3 ハードウェア仕様変更の影響波及例

3.2 モータ駆動システム(EV)

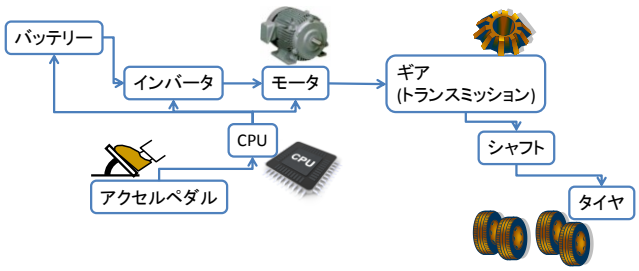

図 4 モータ駆動システムの構成

図 4 に示すように、電気自動車(EV)のモータ駆動システムはバッテリー、ギア、アクセルペダル、タイヤ等の詳細仕様に影響される。アクセルペダルの踏み込み量に対して、一定の加速を行う制御だけでも、バッテリーの残量や経年劣化、ギア比やタイヤの摩耗等も考慮した回転制御を行わなければならない。

また、図 4 のような構成を取る場合、電子制御されるモータから先の、ギア、シャフト、タイヤは、全て機械的な機構で繋がっているため、以降の摩耗や環境による微細な変化を考慮した制御ソフトウェアの設計/実装が必要になる。

さらに、より高度な運転アプリケーションを導入するためには、タイヤに備えられるブレーキパッドへ力を与えるブレーキシステムとの連携を、電子的にも機械的にも考えたソフトウェア作りが重要である。

3.3 課題のまとめ

本節で示したように、ハードウェアの仕様変更頻発による構成要素間の従属的な影響関係は把握困難であり、フィーチャモデルで扱うと、フィーチャ分岐数が増大し、また、相互依存関係が発生するためツリー構造を維持できない。上記課題を解決するためには、ハードウェア変更が間接的にソフトウェアへの影響を整理する仕組みが必要となる。

4. 提案手法

ハードウェア(HW)仕様変更の影響波及を容易に追跡するため、我々は下記 3 つの依存関係を管理するトレーサビリティマトリクスを用いたソフトウェア(SW)への影響分析手法を提案する。

1. HW 間依存関係マトリクス
2. HW-SW 依存関係マトリクス
3. SW 間依存関係マトリクス

4.1 HW 間依存関係マトリクス

図 5 に HW 間の依存関係マトリクス概要を示す。横軸に変更対象となるハードウェアのコンポーネントを、縦軸に影響先のハードウェアのコンポーネントを記す。

本ハードウェアマトリクスにおける依存関係の定義は以下の二つである。

1. 配置寸法の依存関係
   ひとつの部品サイズ変更による、他の部品配置との干渉
2. 力学的依存関係
   ギア比などの動力伝達手段や、液圧・空圧等の動力伝達手段の変更による他の動力伝達手段の特性変更

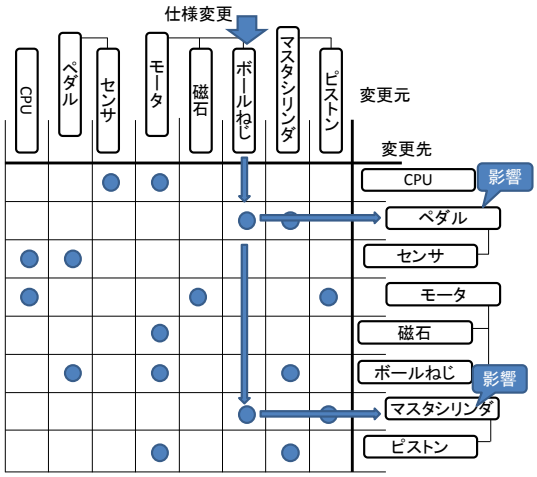

図 5 HW 間依存関係マトリクス

Vol.2014-SLDM-165 No.7
Vol.2014-EMB-32 No.7
2014/3/15

例えば、ブレーキペダルはマスタシリンダとは力学的な依存関係にあり、ドライバーがペダルを操作した際の液量制御に影響を与える。また、CPU 基板とモータは配置寸法の依存関係があり、片側の変更がもう片側のサイズに影響を与える。

4.2 HW-SW 依存関係マトリクス

図 6 にハードウェアとソフトウェア間の依存関係マトリクス概要を示す。横軸に変更元のハードウェアのコンポーネントを、縦軸に影響先のソフトウェアのコンポーネントを記す。ハードウェアのコンポーネントが変更になった際には、影響対象のソフトウェアのコンポーネントがわかるように印を入れる。本マトリクスにおいて、あるハードウェアと、あるハードウェアの特性を制御するソフトウェアモジュール間に影響関係があるとする。

例えば、モータの磁石が変わる場合、モータのトルク特性が変わるため、トルク計算制御ロジックに修正が入る。また、最もハードウェアの操作に近い電流制御も磁石の特性を吸収する制御が必要となるため、影響が存在する。

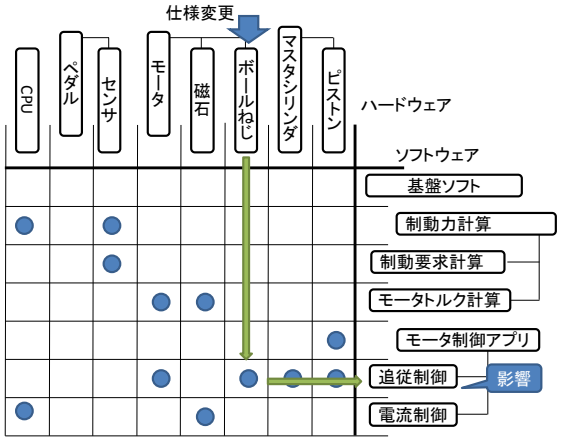

図 6 HW-SW 依存関係マトリクス

4.3 SW 間依存関係マトリクス

図 7 にソフトウェア・コンポーネント間の、依存関係マトリクスを示す。ソフトウェア・コンポーネント間の影響関係は、図 8 の制御フローに基づき分析する。例えば、モータトルクの計算は、後段の追従制御に制御値を受け渡すためモータトルク計算と、追従制御間に影響有りとしてチェックを入れることになる。

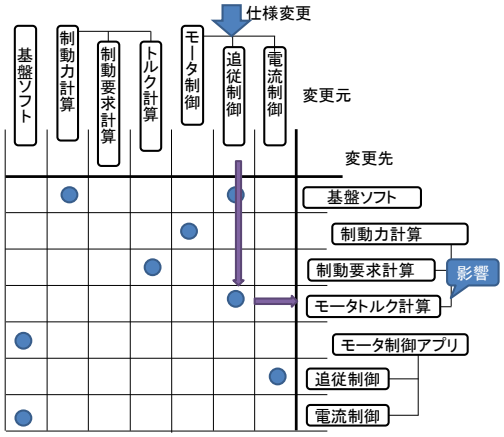

図 7 SW 依存関係マトリクス

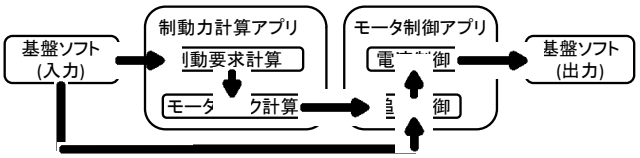

図 8 ソフトウェアの制御フロー

4.4 提案手法を用いた SPL 開発プロセス

図 9 に各マトリクスを用いた影響分析の概要を示す。ハードウェアの仕様変更があった際に、設計者は HW 依存関係マトリクスを元に影響があるハードウェアを確認する。次に、変更のあったハードウェアが影響を与えるソフトウェアを HW-SW 依存関係マトリクスを用いて追跡する。最後に、SW 間の影響は SW 依存関係マトリクスで追跡する。以上により、ひとつのハードウェア仕様変更が影響を与えるソフトウェア仕様を抽出できる。ハード仕様変更とソフト仕様変更の結果は、両者を対応させる形で管理する。

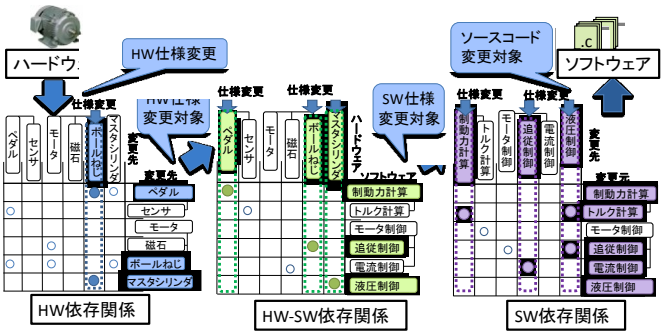

図 9 ハードウェアの仕様変更を追跡する 3 マトリクス

図 10 に提案手法を用いた際の開発プロセスを示す。本手法の適用によりハードウェアの仕様変更がソフトウェアに与える影響を管理できるようになった。

Vol.2014-SLDM-165 No.7
Vol.2014-EMB-32 No.7
2014/3/15

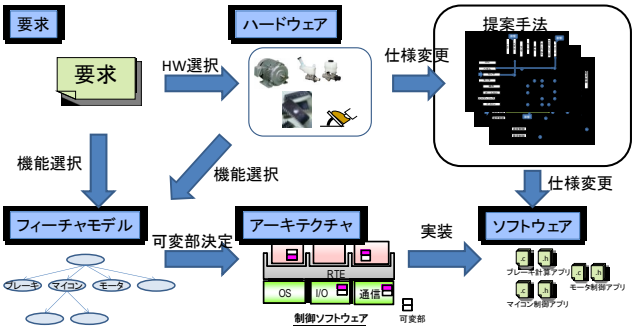

図 10 提案手法を用いた SPL 開発プロセス

5. 考察

提案手法を用いることで、ハードウェアの仕様変更時の影響把握が可能になる。本手法適用により、より詳細なレベルでハードウェアの仕様影響を管理可能になった。

従来、依存関係の複雑さにより、再設計を繰り返すことによる大きな変更工数がかかっていた。本手法により、再設計箇所の発見を早期化することができるため、大幅な工数削減効果が期待できる。

また、上記の開発プロセスでは、フィーチャモデル等の上位設計は変更せずに再利用できるため、SPL による開発効率向上を維持することができる。

HW-SW 依存関係マトリクスを用意することにより、本手法の用途以外にも、ハードウェア仕様からフィーチャ選択する際に利用することができる。

6. まとめと今後の課題

本研究では、開発中にハードウェアの仕様変更が多発する制御システムにおいて、ハードウェアの仕様変更がソフトウェアに及ぼす影響を管理する手法を提案した。

今後の課題として、SysML 等のモデル言語により、上位の概念でハードウェアやソフトウェア間の関係を管理し、依存関係マトリクスを機械的に導出することが挙げられる。

試行の結果、ハードウェアをマトリクスに分解するための適切な粒度の尺度が実務上重要であることがわかった。ハードウェアをマトリクスに分解するための適切な粒度を決める手法が必要である。

参考文献

1) Klaus Pohl, Gunter Bockle, Frank J. van der Linden: Software Product Line Engineering, German, Springer-Verlag (2005)

2) Kyo C. Kang, Sholom G. Cohen, James A. Hess, William E. Novak, A. Spencer Peterson: Feature-oriented domain analysis (FODA) feasibility study, Carnegie Mellon University Software Engineering Institute, 1990.

3) 細合 新太郎, 岸 知二: 2 層フィーチャモデルを用いた開発手法の提案と実装: IPSJ SIG Technical Report (2008)

4) Christopher Brink, Martin Peters, Sabine Sachweh : Configuration of Mechatronic Multi Product Lines, In Proceedings of the 3rd international workshop on Variability & Composition. ACM, 2012. p. 7-12.

5) Cosmin Dumitrescu, Raul Mazo, Alain Dauron, Camille Salinesi: Bridging the gap between product lines and systems engineering. An experience in variability management for automotive model based systems engineering, 17th International Software Product Line Conference

6) Object Management Group: OMG System Modeling Language (OMG SysML™),V1.1, OMG document number formal/2008-11-02, November 2008, http://www.omgsysml.org

7) Christian Tischer, Birgit Boss, Andreas Müller, Andreas Thums, Rajneesh Acharya, Klaus Schmid: Developing long-term stable product line architectures, In Proceedings of the 16th International Software Product Line Conference-Volume 1. ACM, 2012. p. 86-95.