デジタルプラクティス　Vol.10 No.3(July 2019)

一般投稿論文

多品種小変更型開発におけるコア資産保守・製品導出手法の改善と実践

長峯 基[1]　中島 毅[2]

[1]三菱電機（株）　[2]芝浦工業大学

近年，ソフトウェアプロダクトラインエンジニアリング（SPLE）の広がりとともに，Pohlらが紹介する参照モデルをベースとした実践事例が数多く報告されている．筆者らは参照モデルをもとに2010年からSPLEを導入したが，多品種小変更型開発の特徴ゆえに，要求仕様からソフトウェア部品への変換ミス，ソフトウェア部品の仕様不適合という2つの課題が生じ，長期的な効果を得るには至らなかった．本稿では，この2つの課題を解決するために，多品種小変更型開発におけるコア資産の保守および製品導出のための手法を改善した．このSPLE改善手法は，①開発ロードマップからアプリケーション開発を1つの幹開発と複数の枝開発に分離，②幹開発の中で，個別製品の開発を行いつつ，後続する枝開発のバリエーションを吸収可能なコア資産（ソフトウェア部品）とそのソフトウェア部品と関連付けた要求仕様書マスタを作成，③枝開発の中で，その要求仕様書マスタからのカスタマイズとソフトウェア部品の組立て，を実現するもので，上記の開発プロセスと要求仕様書の構成法からなる．本手法を，空調機開発に適用，効果検証を行い，コア資産の再利用率が向上し生産性が改善するという有効性を確認した．

1. はじめに

近年，空調機器やエレベータ，自動車等の組込み製品においては，他社製品との差別化のための製品機能の高度化，多様なユーザニーズへのタイムリな対応が求められている．そのため，これら製品向けのソフトウェア開発では，開発する機能を共有しつつ製品バリエーションに合わせて少しずつ異なる仕様を実現する複数の開発を，短期間に並行して行うことが多い（以下「多品種小変更型開発」と呼ぶ）．さらに，多品種小変更型開発は，厳しい企業間競争に勝ち抜くため，市場動向を見ながら柔軟に機能拡張することと，高品質を担保しつつ低コストで開発できることが求められている[1]．

一般に，高品質と低コストを両立させるソフトウェア開発を実現するためには，なるべく既存のソフトウェアに手を加えず再利用することが肝要である[2][3]．このため，特定のドメインの製品群に対して，ソフトウェアの再利用を体系的に行う手法として，ソフトウェアプロダクトラインエンジニアリング（以下，SPLE）が提案され，広く実践されている[1][4][5]．

SPLEは，3つの活動 ①コア資産を開発・保守するドメイン開発，②コア資産をもとに製品を開発するアプリケーション開発，③これらを系統立って行うための管理プロセスから構成される．SPLEは，ドメイン開発とアプリケーション開発を分離することで，コア資産の再利用性の向上と，顧客向けアプリケーション開発の短納期化という2つの関心事を分離して実行・管理できる利点がある[2][7]．効率的にコア資産を保守し，製品を導出するためには，これら3つの活動を円滑に実施できる組織構造およびプロセス実装が成功の主要因であるとされている[6]．

Klaus Pohlら[7]は，これら3つの活動のプロセスの組み立て方や，組織構造に応じた実践方法を参照モデルとして紹介している．また，その成果はSPLEの参照モデルとして国際標準に採用されている[8]．

筆者らは，Pohlらが紹介する参照モデルを参考にSPLEを導入し，2010年から製品導出を実施した．その結果，一時的に製品群全体の生産性が向上したが，その後コア資産の再利用率が下がり生産性が低下傾向を示すようになった．この原因を以下のように分析している[10]．多品種小変更型開発では，製品群への要求は顧客に近いアプリケーション開発の担当組織がもっているため，ドメイン開発側主導で要求事項の分析を進めることが難しい．また異なる市場に対応する多数のアプリケーション開発担当組織が，同時並行的に類似した要求仕様で開発を始めようとするため，実際にアプリケーション開発を実施するソフトウェア部門がソフトウェア部品を再利用するか新規に開発するのかの判断を誤る場合が増えてしまう．この判断誤りが，不必要な類似部品の発生を増やしコア資産を劣化させ，結果的に再利用率を落とす原因となった．

我々は，この原因分析に基づき，ドメイン開発の実施方法および，アプリケーション開発における要求仕様書の改善を図った．この改善手法は，①開発ロードマップからアプリケーション開発を1つの幹開発と複数の枝開発に分離，②幹開発の中で，個別製品の開発を行いつつ，後続する枝開発のバリエーションを吸収可能なソフトウェア部品とそのソフトウェア部品と関連付けた要求仕様書マスタを作成，③枝開発の中で，その要求仕様書マスタからのカスタマイズとソフトウェア部品の組立て，を実現するもので，上記の開発プロセスと要求仕様書の構成法からなる．ここで，ソフトウェア部品とは，コア資産の一部であり，ある機能単位にまとめられたソースコードを指す．

本開発手法を実開発に適用することで，判断誤りによる類似部品の発生を抑制し，結果，ソフトウェア部品の再利用率が39.6ポイント改善，製品群全体における生産性が55.7ポイント向上し，さらにそうした効果が持続できていることが確認できた．

本稿の第2節では，Pohlらが紹介しているコア資産保守・製品導出プロセスと，それに基づき実施した筆者らの第1期活動と課題について述べる．第3節では，第2節で述べた課題に対して，改善手法を提示し，効率的なコア資産保守・製品導出プロセス，およびそれを可能にする要求仕様書の構成法について説明する．第4節では，第3節で述べた手法の実践状況を述べるとともに，その効果や妥当性について検証する．

2. SPLE参照モデルにもとづく第1期活動と課題

2.1 SPLE参照モデル

Pohlらは，**図1**に示すような，製品群に対する要求仕様から系統的にコア資産保守・製品導出するためのフレームワークを参照モデルとして紹介している[7]．

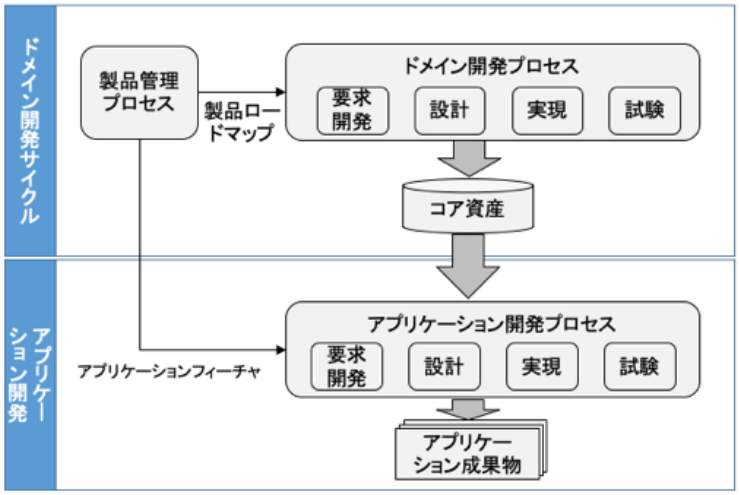

図1　SPLEフレームワーク

SPLEフレームワークは，製品群全体に対するコア資産を開発・保守するためのドメイン開発サイクル，および特定の顧客や市場向けの製品を作るアプリケーション開発から構成される．ここで，ドメイン開発サイクルは，製品管理プロセスとドメイン開発プロセスの継続的な繰り返しからなる．SPLEフレームワークは，ドメイン開発サイクルを個々のアプリケーション開発から分離することで，安定したコア資産を構築・保守し，製品群全体の開発コストを下げることを狙っている．

図1に示すように，製品管理プロセスは，製品群全体が対象とする顧客・市場の要求や経営戦略に基づき，製品ロードマップを定義する．製品ロードマップとは，製品群のリリース計画だけでなく，主要な共通フィーチャや可変フィーチャ，各製品が持つべきフィーチャ（アプリケーションフィーチャ）を含む．ドメイン開発プロセスでは，製品ロードマップをもとに，製品群の共通性と可変性を抽出し，個々の製品の開発に利用可能なコア資産を作り出す．アプリケーション開発では，ドメイン開発において開発したコア資産を活用し個別製品の開発を行う．

SPLEフレームワークにもとづく組織構造およびプロセス実装の要点は以下の3つである．
①ドメイン開発サイクルへの開発組織の割当て
ドメイン開発サイクルへの組織の割当てについては，階層型組織，マトリクス型組織それぞれでいくつかのバリエーションが紹介されている．たとえば，階層型組織構造における集中型ドメイン開発では，アプリケーション開発とは異なる独立した組織でコア資産を開発し，分散型ドメイン開発では，各製品開発組織内にドメイン開発を割り当てる．いずれにおいても，ドメイン開発と製品群全体に対する責任分担とを統合できることが重要である[7]．
②ドメイン開発での共通性と可変性の組込み
ドメイン開発サイクルで製品の共通性と可変性の組込みを行う．プロセスの実装の仕方にはいくつかのバリエーションがあり，市場ニーズや技術の将来予測に基づいて先を見越して共通性と可変性を識別しコア資産の開発を行う方法[7]と，市場ニーズにもとづく変更要求に受動的に対応しコア資産を発展させていく方法[9]とがある．
③コア資産の最大限の再利用
アプリケーション開発においては，ドメイン開発で開発したコア資産を利用することで，品質お

よび生産性が高い開発を行う．このため，アプリケーション開発では，ドメイン開発であらかじめ開発したコア資産の利用率を高め，判断誤りによる類似部品の発生を抑制することが重要である．

2.2　第1期活動の実践と課題

2.2.1　多品種小変更型開発とその特徴

空調機器やエレベータ等の多品種小変更型開発の特徴を以下に示す．

①一定期間（例：3か月～1年）の開発サイクルで機能・性能向上のための製品群の更新を行う．その際ハードウェアとソフトウェアの開発が並行する．

②製品の商品価値がハードウェアにあるため，製品群の機能・性能の仕様および開発計画は，機械（メカ）や電気（エレキ）部門が決定する[11]．ソフトウェア部門は仕様に従いソフトウェアを開発する．

③ハードウェア優先の製品開発であり，性能目標達成や市場動向の変化への対応のため，ハードウェアの仕様確定の遅れや変更が多い．その影響によりソフトウェアの仕様確定も遅れがちで，決定後も変更が多い．

④製品群は，ソフトウェアからみて，以前の開発サイクルの製品群の仕様の多くを共有し，また同一開発サイクルで開発する新機能・性能改善のための仕様も共有している．

⑤製品群は，上記共通仕様をベースにして，特定用途や特定の出荷地域向けにカスタマイズする必要があり，同一開発サイクル内におけるカスタマイズ開発数が多い（例：10～20件）．

⑥個々のカスタマイズ開発は小規模であり，短期に並行して実施される．

2.2.2　第1期活動におけるSPLEフレームワークへのマッピング

SPLEフレームワークに，多品種小変更型開発のメカ・エレキ部門およびソフトウェア部門を，割り当てようとすると，特徴②より表1の対応関係となる．

表1　SPLEフレームワークに対する担当組織

SPLE フレーワーク		対応する部門（役割）
ドメイン開発サイクル	製品管理プロセス	メカ・エレキ部門
	ドメイン開発プロセス	ソフトウェア部門（要求・設計・実現・試験）
アプリケーション開発		メカ・エレキ部門（要求） ソフトウェア部門（設計・実現・試験）

メカ・エレキ部門が，製品管理プロセスを担い，製品群の個々の製品が持つべきフィーチャの決定と，その市場投入時期を決定する．

個々の製品開発に対して，メカ・エレキ部門から担当チームが割り当てられる．各担当チームは，製品管理プロセスで決定したアプリケーションフィーチャを実現すべく，個別に要求仕様をまとめる．ソフトウェア部門は，メカ・エレキ部門の担当チームより要求仕様を受け，アプリケーションの開発を担う．

ドメイン開発は，ソフトウェア部門がプロジェクトチームを立てて担い，製品計画に沿ってコア資産の開発や保守を行う．このとき，ドメイン開発はリアクティブにコア資産を開発・保守するアプローチを採用している．なぜならば，競合の多い製品ドメインにおいては，開発要求は変更されやすく，プロアクティブなアプローチでは開発したコア資産が利用されない可能性が高まるためである．

アプリケーション開発では，ドメイン開発で開発したコア資産をもとに，製品ソフトウェアを組み上げることを志向した．

2.2.3　第1期活動における課題

第1期活動では，2つの課題[10]があった．以下にそれらの課題を整理して示す．

［課題1］ 要求仕様からソフトウェア部品再利用へ変換ミス

アプリケーション開発（要求開発）において，メカ・エレキ部門の担当チームは，ソフトウェア部門が保守するコア資産を構成する参照アーキテクチャとソフトウェア部品に関する知識を持たずに，要求仕様書を記述する．

ソフトウェア部門は，そうした要求仕様を解釈し，可変点の識別に基づき，利用可能なソフトウェア部品の開発をする．しかし，複数の開発が並行するため，要求仕様の解釈時に，可変点の見逃しが頻繁に起き，結果的に利用可能なソフトウェア部品を利用できずに不必要な開発が増加してしまう．

［課題2］ソフトウェア部品の仕様不適合

製品ロードマップ作成時点では，製品群のラインナップ，並びに各製品の持つアプリケーションフィーチャおよび可変点は決まっていても，可変点に対応するバリアントであるソフトウェア部品の具体的な仕様は，ハードウェアの詳細な仕様が決まるまで確定しない．極端な場合，その状態は対象となる新機能を盛り込んだアプリケーション開発の完了間際まで続く．

このため，アプリケーションの開発以前に，ドメイン開発（特に設計から試験）を実施しようとすると，仕様を仮決めして，ソフトウェア部品を作ることになり，作成したソフトウェア部品は，アプリケーションにそのまま利用できない場合が多くなる．すなわち，アプリケーション開発で個別に修正する事態が起こりやすくなる．

要求変更が多い開発には受動的に対応するアプローチ[9]が適していると思われるが，特徴⑤により同一開発サイクル内におけるカスタマイズ開発数が多い場合，統制なくすべての要求変更を受動的に対応することは困難である．

課題1の要求仕様からソフトウェア部品への変換ミスおよび課題2のソフトウェア部品の不適合が頻繁に発生することにより，コア資産であるソフトウェア部品の再利用率が低下し，類似したソフトウェア部品が増加する状況が生じる．

3. 第2期活動におけるコア資産保守・製品導出プロセスの改善

多品種小変更型開発に対する2つの課題を解決するためのコア資産保守・製品導出手法を改善した．改善手法は，開発プロセスおよび要求仕様書の構成法からなる．

3.1　開発プロセス

改善した開発プロセスを**図2**に示す．本プロセスは，対象製品群の同一販売期間向けの開発サイクルに対応するもので，1開発サイクルは1つの製品管理プロセス，1つの幹開発プロセス，および複数のアプリケーション開発プロセス（枝開発）からなる．

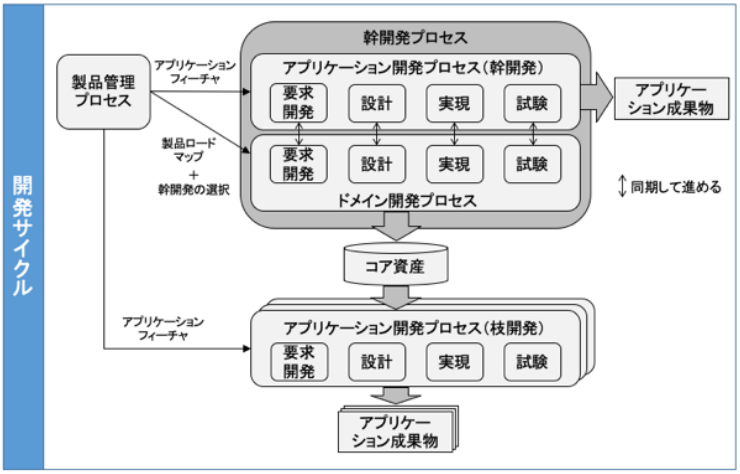

図2　改善した開発プロセス

3.1.1　開発プロセスの詳細

改善した開発プロセスの詳細と，多品種小変更型開発の2つの課題の解決について以下に示す．

（1） 製品管理プロセス

本プロセスの主要な目的は，開発サイクル（たとえば1年間）の製品開発計画を定めることと，計画された製品の中から幹開発の選定を行うことである．

製品管理プロセスでは，メカ・エレキ部門が，市場のニーズや，他社動向，販売戦略等に基づき，長期の開発計画，および当該開発サイクルの詳細な開発計画を定める．当該開発サイクルの詳細な開発計画には，製品ラインナップ，開発時期，主要な新機能や性能目標，ハードウェアの構成を記述する．

これらの情報に基づき，当該開発サイクルの開発対象製品群の中から，以下の条件に適合するものを幹開発対象として選定する：
①開発する新機能数が最多また次点のもの
②当該開発サイクルで新たなハードウェアを搭載するもの
③本来は枝開発であるが，幹開発と同時に開発ができるもの（ハードウェアが幹開発と同等であるもの）

幹開発対象製品以外は枝開発対象製品とする．

（2） 幹開発プロセス

幹開発プロセスの目的は，選定した具体的な製品のアプリケーション開発（幹開発）を進めることと，後続するアプリケーション開発（枝開発）のために当該開発サイクルで利用するコア資産の保守を行うドメイン開発を実施することである．

幹開発プロセスは，この目的のために，メカ・エレキ部門の当該製品担当者とソフトウェア部門との混成チームを構成し，ドメイン開発プロセスおよびアプリケーション開発プロセスにおける要求開発・設計・実現・試験を，同時並行で進めていく．当該プロセスの入出力を以下に示す．

入力：製品ロードマップ，
　　アプリケーションフィーチャ，コア資産
出力：幹開発対象製品（アプリケーション成果物），
　　保守されたコア資産（要求仕様書マスタおよび
　　開発したソフトウェア部品を含む）

当該製品の具体的な要求仕様と製品ロードマップより分析・抽出した可変性に基づき，可変点に対応するバリアント（ソフトウェア部品）を識別する．たとえば，可変点のフィーチャ「圧力保護」が，幹開発は圧力スイッチによる検出，別の枝開発では異なるハードウェア構成（圧力スイッチが付かない等）の製品の開発が枝開発で計画されているとする．この場合，スイッチありとスイッチなしの両方に対応したソフトウェア部品を識別する．

識別した可変点およびソフトウェア部品を盛り込んだ要求仕様書マスタを作成する．要求仕様書マスタは，第3.2節で詳述するように，アプリケーション開発（幹開発）の要求仕様書であると同時に，可変点と識別されたバリアントリストを組込み，当該開発サイクルの対象製品全体に対応した雛形となっている．

識別したソフトウェア部品の仕様の決定と実装は，アプリケーション開発の設計・実現時に並行して行う．

このように，改善後のプロセスでは，幹開発製品を対象とした具体的なアプリケーション開発と，ドメイン開発を同期して進めることができるので，ソフトウェア部品の仕様の確度が向上する．さらに，並行して開発するハードウェアの仕様の決定遅延や変更は，幹開発プロセスの中で吸収できる．これらから，提案プロセスにより，課題2「ソフトウェア部品の仕様不適合」の問題を解決することができる．

（3）アプリケーション開発プロセス（枝開発）

アプリケーション開発プロセス（枝開発）の目的は，コア資産であるソフトウェア部品を再利用し，枝開発対象製品のアプリケーション開発を高品質かつ低コストで実施することである．

アプリケーション開発プロセス（枝開発）は，従来と同様，メカ・エレキ部門の担当チームが要求仕様書を作成し，その仕様書に基づき，ソフトウェア部門が開発を実施する．当該プロセスの入出力を以下に示す．

入力：アプリケーションフィーチャ，
　　コア資産（要求仕様書マスタおよび
　　開発したソフトウェア部品を含む）
出力：枝開発対象製品（アプリケーション成果物）

アプリケーション開発用の要求仕様書は，コア資産の要求仕様書マスタをベースとして，設定された可変点からバリアントを選択し，必要なパラメータを設定することで作る．それにより，バリアントに紐づけられたソフトウェア部品の再利用が確実に実施できるようになる．

要求仕様書マスタにコア資産の再利用を考慮した可変性を直接織り込んであるので（詳細は第3.2節），課題1「要求仕様からのソフトウェア部品の変換ミス」が減り，コア資産を有効に再利用することができるようになる．

3.1.2　開発プロセスの実施例

図3は，提案する開発プロセスを一実施例として時間軸に投影したものである． 製品開発プロセスで選択された幹開発が行われ，具体的な製品開発とともに，当該開発サイクルで利用可能なソフトウェア部品と要求仕様書マスタが開発され，コア資産として登録される．それに続いて，登録されたコア資産を用いた，特定用途向けや出荷地域ごとのカスタマイズを行うアプリケーション開発（枝開発）が，開発ロードマップに沿って，順次並行して行われていく様子を表している．

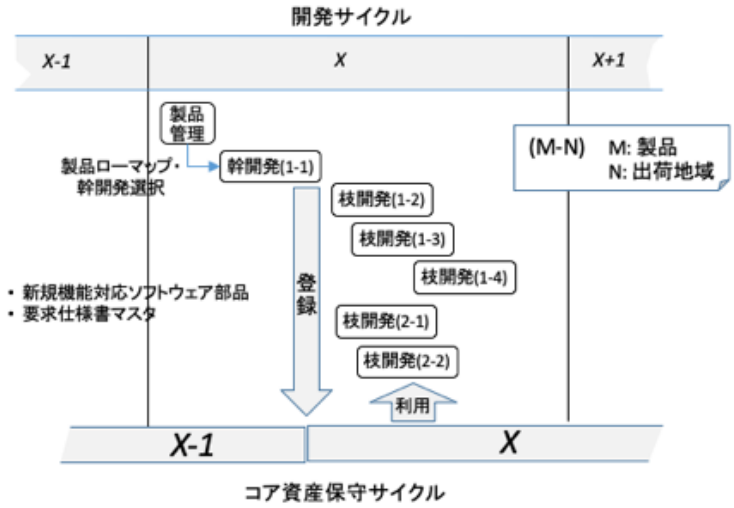

図3　改善プロセスの実施例

3.2　要求仕様書の構成および運用方法

3.2.1　要求仕様書の構成に求められる要求事項

3.1節の開発プロセスが有効に機能するためには，要求仕様書が以下3つの要求事項を満たす必要がある．

R1）製品群で必要とされる可変性を表現できること．

R2）アプリケーション開発の要求開発を担当するメカ・エレキ部門の要員にとって理解しやすいものであること．

R3）要求仕様書とソフトウェア部品の間のトレーサビリティが確保できること．

改善手法は，これらの要求事項を満たすように，次に示す要求仕様書の構成法を用いる．

3.2.2　可変性の分析

筆者らは，多品種小変更型開発に本改善手法を実装する以前に，5年にわたるSPLE実適用を経ている[10]．その適用結果の分析から，要求事項の変化はおおむね以下の3つに分類できることが分かった．

①機能の有効／無効

②デバイス制御方式の差異（同一目的・同一機能だが，デバイス違いから検出方法や演算方法が異なるもの）
③ハードウェア・出荷地域違いによるパラメータ差分

これら3種類の要求事項の変化は，ソフトウェア的には以下の2種類の可変性によって吸収できることが分かった．
・ソフトウェア部品の選択（①，②に対応）
・パラメータ値の設定（③に対応）

3.2.3　要求仕様書の構成

改善手法では，上記2種類の可変性を要求仕様レベルで選択できるような要求仕様書の構成とすることによって，要求事項R1の達成を図った．

図4に提案する要求仕様書の構成を示す．要求仕様書は制御仕様書とデータ仕様書の2つの文書からなる．制御仕様書は，製品群に搭載されるすべての機能について記述し，各機能の開始条件や制御内容等の仕様が記載された製品群全体で共有される文書である．後述するように，すべての機能を記述したうえで，製品ごとに搭載される機能を明確化している．データ仕様書は，各製品における具体的なパラメータの設定値が記載された文書を指し，製品ごとに持つ．

図4　要求仕様書の構成

（1） 制御仕様書

制御仕様書は，システム構成と製品機能の2つの章を持つ．「システム構成」は，製品―機能マトリクスによって表現される．製品―機能マトリクスとは，縦軸に製品群に搭載されている全機能，横軸に各製品がそれぞれ列挙され，交点に各製品での機能の有効／無効が明示された表の

ことである．機能を有効／無効をこのマトリクス上で選択することにより，第3.2.2項記載の要求事項の変化①に対応することができる．

縦軸の機能一覧においては，制御方式が異なる類似機能を依存関係付きの別機能として扱う．類似機能とは，機能的な要求は同じでハードウェア構成等の事情により制御が異なるバリアントであり，第3.2.2項記載の要求事項の変化②に対応する．たとえば，同じ「保護機能」でも，圧力センサで直接圧力を測るものと，圧力センサの代わりに温度を測定することにより圧力を推定するものとがあるなどである．こうした機能は，代替の関係にあるので，図4の製品―機能マトリクスのC-1とC-2のようにいずれかの選択であることを明示する．このようにメカ・エレキ部門には類似機能の存在とバリエーションを知らせている．

「製品機能」は，当該製品が持つ機能を，開始条件や制御内容等の項目について記述する．「システム構成」において一覧化されたすべての機能をここに記述する．これらの機能は（Func_A）のようなタグを有しており，下流工程の文書およびソースコードの該当個所には本タグが付与されている．これによりソフトウェア部品と明示的に紐づけられるため，上流工程での変更を下流工程に誤りを混入させることなく伝達することが可能となる．

上記の章構成および記述方法により，開発済みの機能の存在とその仕様を知ることができる．

（2） データ仕様書

製品機能の中には，同一の仕様であっても製品ごとにパラメータが異なるものが多い．これらの差異は，データ仕様書に記述する．具体的には，制御仕様書の本文中には[Data_3-1_1]のようなタグを記述し，このタグと同一のタグをデータ仕様書に記述することで，トレーサビリティを確保する．外出しするチューニングデータは，能力帯，出荷地域，ハードウェア構成，製品シリーズを含む．

データ仕様書により，第3.2.2項記載の要求事項の変化③に対応する，製品別のパラメータ差分を吸収する個所が特定でき，また製品ごとのデータ部品を自動生成することで，パラメータ違いのみでソースコードを修正するような無駄な開発を防ぐことができる．

3.2.4　幹開発と枝開発における要求仕様書の利用方法

改善手法では，幹開発プロセスにおいて，提案する要求仕様書を作成する．これを要求仕様書マスタと呼ぶ．

幹開発プロセスは，①要求仕様書マスタの作成，②可変点とそのバリアントである「機能」に対応するソフトウェア部品の作成，③機能とソフトウェア部品のリンク付けを行う．要求仕様書マスタは，幹開発対象製品の要求仕様書でもある．要求仕様書マスタとソフトウェア部品は，コア資産に登録する（図2）．これにより，要求事項R3「要求仕様書とソフトウェア部品の間のトレーサビリティを確保」の達成を図ることができる．なお，この作業は，幹開発プロセスに割り当てられたメカ・エレキ部門の幹開発製品の担当者とソフトウェア部門との混成チームで行うことで，適切に行うことができる．

アプリケーション開発（枝開発）では，要求仕様書マスタをカスタマイズすることで，個別の要求仕様書を作成する．各アプリケーション開発プロセス（枝開発）において，要求仕様書マスタの「システム構成」の担当製品に対応する縦の欄を完成させ，当該枝開発用のデータ仕様書に能力帯や出荷地域ごとのパラメータ設定を記載する．

2つの可変性を組み込んだ要求仕様書を利用することにより，各アプリケーション開発プロセス（枝開発）において，メカ・エレキ部門が，要求仕様書を個別に作らず，個別製品の要求事項を，選択式で行うことができる．これにより，要求事項R2「メカ・エレキ部門の要員にとって理解しやすいものであること」の満足を図る．

R1-3を満たす要求仕様書を用いることにより，提案プロセスを用いて，要求仕様から変換ミスなくソフトウェア部品をより確実に活用することが可能になる．

4. 適用と評価

4.1 適用対象，経緯および方法

典型的な多品種小規模型開発である空調機（室外機）を対象として，提案するコア資産保守・製品導出手法を適用・評価を実施した．

（1）適用対象

空調機（室外機）は，日本国内のみならず世界各国へと輸出・展開している典型的な多品種小変更型の製品群である．製品群は，以下のバリエーションを持っている．
・能力帯（顧客層に対応）
・出荷地域別／能力帯別のハードウェア構成の違い
・特定用途向けの機能の搭載の有無

1年を1開発サイクルとして，機能・性能の向上を実現する製品群のソフトウェア開発を実施する．上記のバリエーションを持つ製品群を30製品程度開発する．各製品の開発は，小規模でシステム試験まで含んで3～6カ月程度の工期が設定されている．これらの開発の多くは，前年度製品群からの機能の流用率が90%と高い．

開発組織は，メカ・エレキ部門，ソフトウェア部門からなり，メカ・エレキ部門がハードウェアと各製品の開発，ソフトウェア部門がコア資産の開発・保守，および各製品のソフトウェア開発を実施する．

（2）適用の経緯

適用対象に対して，以下の3段階で実施している．
①準備段階（2006～2009年）：ドメイン開発を実施し，コア資産を構築した段階．プロダクトラインアーキテクチャを開発しソフトウェア部品をコア資産として整備した．
②第1期活動（2010～2014年）：準備ステージで構築したコア資産を用いて，個別製品を本格的に開発した段階．Pohlらが紹介する参照モデル[7]をベースに，コア資産の保守・製品導出を行った[10]．
③第2期活動（2015年～）：第1期活動の結果，生産性と再利用率の低下があり，その原因分析[10]に基づき改善手法を適用した段階．

適用対象ドメインは，SPLE準備段階以前から国内に安定したビジネス基盤をもっており，製品群の品質もすでに高い状態であった．第1期から第2期にかけ海外市場への製品展開が増加していくことが予想されており，これに伴い，ソフトウェアの開発規模の増加，それに伴う開発人員の質と量の確保と同時に，従来と同様の品質を維持すること求められていた．

（3）適用方法

上記③第2期活動において，改善手法を実組織において以下のように実装した．

①メカ・エレキ部門とソフトウェア部門が，開発サイクルの最初にワーキンググループを作り，開発ロードマップに沿って幹開発となる対象製品を決定する．具体的には，国内市場向けの性能・機能における最上位製品を幹開発とし，国内市場向けの廉価製品や海外市場向けの製品を枝開発とした（図3）．

②幹開発は，幹開発の担当者とソフトウェア部門のドメイン開発担当者がチームを組み，対象製品の開発を進めつつ開発サイクル内で共有する機能，およびバリエーションを実現するための要求仕様書マスタ＋ソフトウェア部品群を開発していく．幹開発を相対的に開発費の多い国内市場向けの最上位製品としたため，コア資産の開発や保守のコストおよび要員は，幹開発の担当部門が受け持つ．

③枝開発は，要求仕様書マスタを，メカ・エレキ部門の担当者がカスタマイズして枝開発対象製品の要求仕様書を作り，ソフトウェア部門がその仕様書に基づきソフトウェアを開発する．枝開発の途中で，想定してなかった仕様変更があり，新たな実装が行われた場合には，要求仕様書マスタとソフトウェア部品への追加を実施し，コア資産の保守を行う．

4.2　適用前後の比較と評価

上記に示した適用対象，経緯，および方法で，本手法を，2015年度開発製品より適用を実施した．適用開始直前の2014年度の各指数を100として，各指標の推移を示し，適用の効果を示す．各年度のプロットは，各年度の開発で得られたデータの平均値とした

.

4.2.1　ソフトウェア部品の再利用率

多品種小変更型開発へのSPLE適用上の課題1「要求仕様のソフトウェア部品への変換ミス」，すなわち開発／再利用の判断ミスの解決を見るため，ソフトウェア部品の再利用率を評価する（**図5**）．

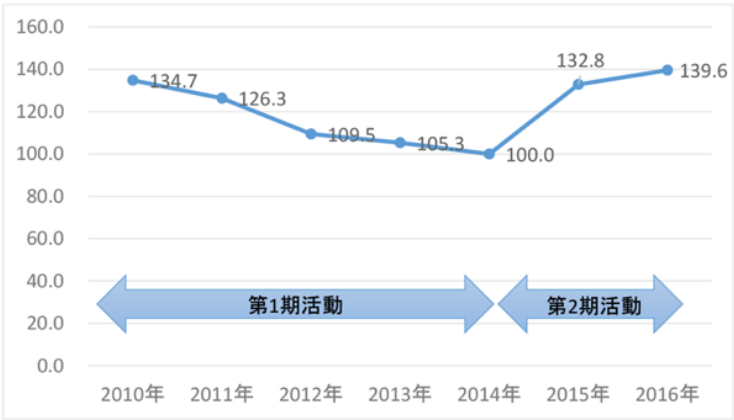

図5　ソフトウェア部品再利用率（2014年度を100）

ソフトウェア部品再利用率Rpは次式と定義する.

Rp＝Nr / Np （1）

Nr：複数製品で使用されているソフトウェア部品数

Np：コア資産に登録されているソフトウェア部品数

（1）適用前と適用後の比較

図5に示すように，ソフトウェア部品再利用率は，第1期活動では，2年目より，徐々に低下し4年間で34.7ポイント低下しているのに対し，本手法適用段階では，対14年度比で15年度に32.8ポイント，16年度は39.6ポイントの改善が見られる.

（2）ソフトウェア部品への変換ミスの防止効果の評価

第1期活動時は，ソフトウェア部品の増加に伴い，再利用率の低下が見られた．本手法の適用後は，提案する要求仕様書の導入により，ソフトウェア部品の変換ミスが減り，再利用率が高まっていると考える．また，この時期開発人員が増加しており，慣れによる習熟度の向上が見込めない状況であったことを加味すると，本手法の有効性が評価できる．さらにこの効果が2年目も高い水準を維持できていることから，ソフトウェア部品数が増加しても，変換ミスを防止する効果が維持できる手法と言える.

4.2.2　類似部品発生率

多品種小変更型開発へのSPLE適用上の課題2「ソフトウェア部品の仕様不適合」，すなわちソフトウェア部品がそのまま使えない状況の発生という課題の解決を見るために，類似部品発生率（ソフトウェア部品1つあたりのブランチ発生数）を評価する（**図6**）.

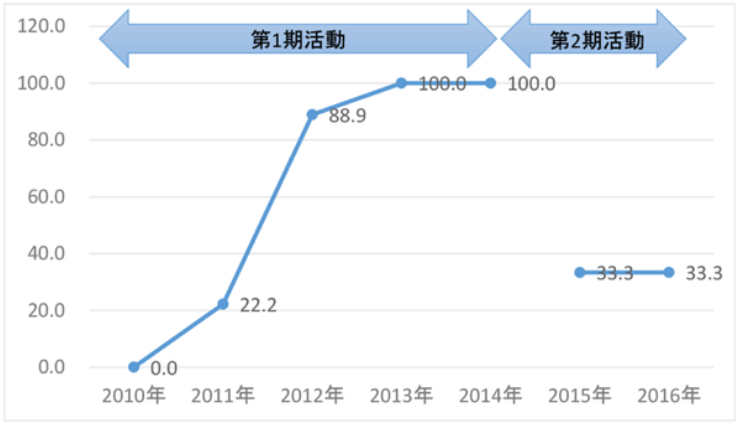

図6　類似部品発生率（2014年度を100）

類似部品発生率Rsは次式と定義する.

Rs＝Ns / Np （2）

Ns: 同一ソフトウェア部品からのブランチ数の総和

Np：コア資産に登録されているソフトウェア部品数

（1）適用前と適用後の比較

図6に示すように，類似部品発生率は，第1期活動段階は，年々増大していった．本手法適用段階では，15年度から16年度はブランチ数の増加はなかった．なお，15年度のブランチ数が33.3ポイントに減少したのは，活動の見直しに合わせて，類似部品を統廃合したことによるものである．

（2）ソフトウェア部品の仕様不適合の防止効果の評価

第1期活動時は，登録したソフトウェア部品をそのまま使えず，類似したソフトウェア部品を多数派生させてしまっていた．本手法適用後は，ソフトウェア部品の仕様の確度が上がったことにより，ブランチの発生数がなくなり，ソフトウェア部品をそのままの状態で利用する場合が増えていることを示しており，ソフトウェア部品の仕様不適合が軽減できる手法であることが分かる．市場の拡大に伴い，細かなバリエーションが増加しているなか，類似部品を発生させなかったことは，本手法の効果であると考える．

4.2.3　製品群全体の生産性

本改善手法が製品群全体に与える効果をみるために，製品群全体の開発の生産性を評価する．

生産性Pは次式と定義する．

$$P = S / C \quad (3)$$

S: 当該年度で開発した新規・改造コードの規模+
Σ（各ソフトウェア部品のコード規模×再利用数）
C: Σ（開発プロジェクトのソフトウェア開発費用）

図7に製品群全体における生産性の推移を示す．

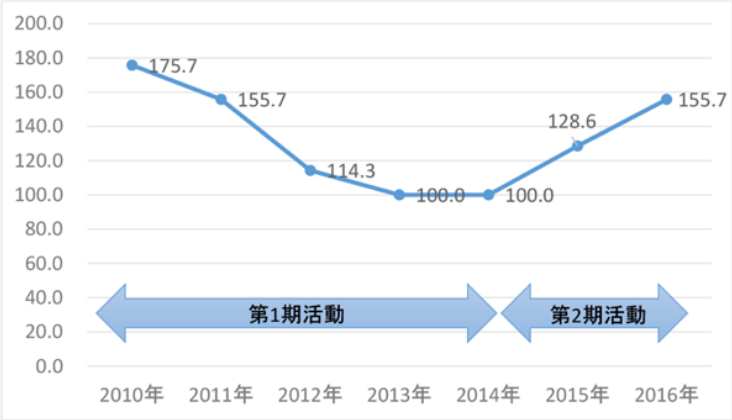

図7　製品群全体における生産性（2014年度を100）

（1）適用前と適用後の比較

図7には，第1期活動段階である2010年度からのデータも示している．この段階は，コア資産の保守や製品導出が適切に行われず[10]，適用2年目より生産性の低下が見られ，4年間で75.7ポイントの低下があった．

第2期活動段階においては，生産性が，1年目の2015年度は2014年度比で28.6ポイント，2年目の2016年度は55.7ポイントの改善があった．

（2）改善手法の生産性に与える効果の評価

第1期活動段階は，ソフトウェア部品の増加と修正によるコア資産劣化により再利用率が下がり，その結果生産性が下がり続けていた．本改善手法を適用することで，部品の再利用率が上がり，結果として生産性の改善の効果があった．

事業上の目標である製品群全体の開発コストの低減は，生産性の向上により達成することができた．第1期から第2期にかけ急激に年間開発数が増え，第1期の10件程度から30件以上に増加したが，開発リソースの調達に破綻なく柔軟に対応することができた．また，製品の品質についてはSPLE導入以前から問題がない状況ではあったが，年間開発数の増加の中で同様の品質を維持することができた．

4.3　考察

4.3.1　改善手法の有効性の評価

第4.2節に示した適用結果より得た改善手法の有効性に関する評価を以下にまとめる．

・多品種小変更型開発の課題1「ソフトウェア部品への変換ミス」については，ソフトウェア部品再利用率が向上・持続できていることから，提案する要求仕様構成法は，この課題を解決できていると評価する．

・多品種小変更型開発の課題2「ソフトウェア部品の仕様不適合」については，類似部品発生率が抑制できていることから，提案する開発プロセスがソフトウェア部品の仕様の確度を上げる効果があり，この課題を解決していると評価する．

・製品群全体について生産性が改善し，2年目も効果が持続していることから，本手法が多品種小変更型開発に対して持続性のある手法となっていると評価する．

4.3.2　妥当性に対する脅威

本手法は，製品要求を決定するメカ・エレキ部門と，コア資産を開発するソフトウェアの部門が分離された組織を対象とすることを前提としている．その他の組織構造での有効性については未評価である．

また，適用対象は，開発規模の小さい，流用率が高い製品群の並行開発を対象としている．並行開発数，流用率や開発規模の範囲に関して本手法の適用上の限界がどこにあるか，不明である．

本改善手法の要求仕様書の構成法は，機能選択とパラメータ変更の2種類の可変性表現とその実装方式に強く依存している．この特徴は，空調機分野において，2006年から進めたSPLEの導入・適用活動の評価から発見した事項である[10]．他の組込みドメインにおいても適用可能性は高いと考えるが適用評価は未実施である

.

空調機の分野においても，新しい機能を作り出すアクチュエータの採用など大きな変更が入る開発の場合があり，その場合プロトタイピングなどの別の開発プロセスと組み合わせ対処する必要がある．

5. 関連研究

本稿で課題としたコア資産の再利用や要求仕様書への可変性の組込みについて他研究論文との差異を整理する．

5.1　コア資産再利用のための組織的課題

組織が，SPLE開発へ移行する際に生じるリスクを軽減するためのアプローチに関する研究や実践事例は数多く報告されている．Schmidらは，プロジェクト統合型というアプローチを提唱している[12]．プロジェクト統合型は，組織内で並走する複数のプロジェクト成果物をマージすることによりコア資産を開発するものである．また，Rubinらも先行製品からの差分開発によってSPLE移行を進めている事例を報告している[13]．

これらのアプローチは，SPLE導入移行時のコア資産の開発と利用を対象にし，移行期特有のリスクやコストの低減の問題を扱っているのに対し，改善手法は，一定期間の開発サイクルに合わせ継続的に行うコア資産の保守と製品導出を対象とし，多品種小変更型開発特有の要求仕様からソフトウェア部品への変換ミス，ソフトウェア部品の仕様不適合を扱っている点が異なっている．

Fogdalらは，調整部門を設け，コア資産の利用を促進する手法を提唱している[14]．調整部門は，並行する独立した開発プロジェクトに横断する形で設けられた組織であり，開発は行わずコア資産の周知・再利用の促進のみを行う．これらの取り組みによりコア資産の利用率が向上したことが報告されている．

Adamらは，統合アプリケーション開発と呼ばれる手法を提唱している[15]．これは，各アプリケーション開発において，製品開発プロセスで作られた開発戦略を参照しコア資産を開発・登録するものである．並行する開発が多い多品種小規模開発には，コア資産の構成管理が複雑になるため，適さないアプローチと考える．

5.2　要求仕様書への可変性の組込み

SPLEの製品導出において，各アプリケーションの製品機能を，段階を設けて構成していく段階的構成（Staged configuration）が有用な手法として知られている[16]．提案する要求仕様書の構成法は，機能の有効／無効および制御方式の差異，並びにパラメータ差分の2段階による段階的構成を採っていると見ることができる．

SPLEでは，共通性と可変性を，モデル図を使って表現し，ドメイン分析，コア資産設計，製品導出のそれぞれの局面で利用する事例が多い．モデル図としては，フィーチャモデル[17]やその発展形，それ以外のクラス図[18]などが提案されている．改善手法は，機能の選択とパラメータの変更の2段階の可変性を要求仕様書に直接埋め込んでいる．これは，直接の要求の出し手であるメカ・エレキ部門の人が読んで，理解できることを重視したためである．

表形式で可変性の表現する方法も多数ある．Stoiberらはフィーチャモデルと決定表の2つの表記を用いて製品導出を支援する方法を提唱している[19]．製品とフィーチャの星取表（Product-feature matrix）を用いて製品管理プロセスにおけるスコーピングを支援するもの[20]や，可変性を分析する手法[21]などがある．改善手法は，この製品とフィーチャの星取表を要求仕様書に組み入れ，製品導出までを系統的に行うことを狙っている．

6. さいごに

本稿では，多品種小変更型開発を対象に，要求仕様からソフトウェア部品への変換（開発／再利用の決定）誤りを軽減するため，コア資産保守・製品導出手法を改善し実践した．この改善手法は，製品ロードマップに基づきソフトウェア再利用戦略上重要な個別製品開発を幹開発として

選定し，当該製品開発におけるアプリケーション開発と同時に後続する製品（枝開発）の可変性を設計・実装することでコア資産を開発し，後続製品での再利用を促進する開発プロセスと，それを可能にする要求仕様書の構成法からなる．

多品種小変更型開発に改善手法を適用した結果，ソフトウェア部品の再利用率の改善，類似部品発生率の維持抑制が可能となった．この結果，高品質状態を維持したまま，事業上の目標である製品群全体における生産性を改善することができた．これらの結果から，改善手法の有効性を確認した．

参考文献

1）吉村健太郎，ダルマリンガム ガネサン，ディルク ムーティック：プロダクトライン導入に向けたレガシーソフトウェアの共通性・可変性分析法，情報処理学会論文誌，Vol.48， No.8，pp.2482-2491（2007）．
2）城谷まりな，岸　知二：SPL開発におけるペアワイズ法を用いたテスト手法について，第78回全国大会，pp.319-320 (2016)．
3）田原圭祐，野田夏子：ユースケースからのフィーチャモデル導出の提案，情報処理学会研究報告，Vol. 2014-SE-183, No.3, pp.1-8 (2014)．
4）Clements, P. and Northrop, L. : Software Product Lines : Practices and Patterns, Addison-Wesley (2002)．
5）Van der Linden, F. J., Schmid, K. and Rommes, E. : Software Product Lines in Action : The Best Industrial Practice in Product Line Engineering, Springer (2007)．
6）Capilla, R., Bosch, J. and Kang, K. C. : Systems and Software Variability Management, Concepts Tools and Experiences, Springer (2013)．
7）Pohl, K., Bockle, G. and van der Linden, F.：ソフトウェアプロダクトラインエンジニアリング，（株）エスアイビー・アクセス (2009)．
8）ISO/IEC 26550 : 2015 Software and Systems Engineering — Reference Model for Product Line Engineering and Management
9）Buhrdorf, R., Churchett, D. and Krueger, C. W. : Salion's Experience with a Reactive Software Product Line Approach. International Workshop on Software Product-Family Engineering, Springer, pp.317-322 (2003)．
10） Nagamine, M., Nakajima, T. and Kuno, N. : A Case Study of Applying Software Product Line Engineering to The Air Conditioner Domain, The 20th International Systems and Software Product Line Conference, ACM, pp.220-226 (2016)．
11）西　康晴：製品品質の決定要因としての組込みソフトウェアと組込みソフトウェア・クライシス，品質，日本品質管理学会誌，Vol.34, No.4, pp.343-349 (2004)．
12）Schmid, K. and Verlage, M. : The Economic Impact of Product Line Adoption and Evolution, IEEE Software, Vol.19, No.4, pp.50-57 (2002)．
13）Rubin, J, Czarnecki, K, and Chechik, M : Managing Cloned Variants : a Framework and Experience. The 17th International Software Product Line Conference, ACM, pp.101-110 (2013)．
14）Fogdal, T., Scherrebeck, H., Kuusela, J., Becker, M. and Zhang, B. : Ten Years of Product Line Engineering at Danfoss : Lessons Learned and Way Ahead, The 20th International Systems and Software Product Line Conference, ACM, pp.252-261 (2016)．
15）Adam, S. and Schmid, K. : Effective Requirements Elicitation in Product Line Application Engineering : An Experiment, International Working Conference on Requirements Engineering : Foundation for Software Quality, Springer, pp.362-378 (2013)．
16）Czarnecki, K., Helsen, S. and Eisenecker, U. : Staged Configuration Using Feature Models, International Conference on Software Product Lines, Springer

Berlin Heidelberg, pp.266-283 (2004).
17）野田夏子，岸　知二：プロダクトライン開発における可変性のモデル化手法，コンピュータソフトウェア，Vol.31, No.4, pp.66-76 (2014).
18）Faulk, S. R. : Product-Line Requirements Specification (PRS) : an Approach and Case Study, Fifth IEEE International Symposium on. IEEE, pp.48-55 (2001).
19）Stoiber, R. and Glinz, M. : Modeling and Managing Tacit Product Line Requirements Knowledge. Managing Requirements Knowledge, Second International Workshop, IEEE, pp.60-64 (2009).
20）Loesch, F. and Ploedereder, E. : Optimization of Variability in Software Product Lines. Software Product Line Conference, 11th International, IEEE, pp.151-162 (2007).
21）John, I. : Using Documentation for Product Line Scoping, IEEE Software, 27 (3), pp.42-47 (2010).

長峯 基（非会員）Nagamine.Motoi@ce.MitsubishiElectric.co.jp
2006年筑波大学第三学群工学システム学類卒業．同年三菱電機（株）入社．

中島 毅（正会員）tsnaka@shibaura-it.ac.jp
1984年早稲田大学大学院修士課程修了．同年三菱電機（株）入社． 2008年早稲田大学大学院博士課程修了，博士（工学）．現在，芝浦工業大学情報工学科教授，ソフトウェア工学に関する研究に従事．著書に『IT Text ソフトウェア開発改訂第2版』（共著，オーム社）．技術士（情報工学／総合技術監理）．IEEE CS，電子情報通信学会，電気学会各会員．

投稿受付：2018年9月10日
採録決定：2019年1月24日
編集担当：居駒 幹夫（青山学院大学）