デジタルプラクティス　Vol.11 No.3(July 2020)

一般投稿論文

多品種小変更型SPLE開発の経験：可変性管理の課題と解決

長峯　基[1]　中島　毅[2]　井上 克郎[3]

[1]三菱電機（株）　[2]芝浦工業大学　[3]大阪大学

ソフトウェアプロダクトラインエンジニアリング（SPLE）を成功させる重要なプラクティスの1つに製品間で異なる仕様の管理（可変性管理）があり，可変性管理の主要な技術に可変性の記述法と構成手段がある．空調機ソフトウェア開発にSPLEを適用し5年間にわたり実践したが，もととなるソフトウェア資産（コア資産）の再利用率が低下した．この問題を分析し，コア資産保守プロセスの見直しと合わせて，仕様差異を生み出す特徴の増加（フィーチャ増加）に伴う具体仕様の決定（バリアント決定）の複雑化という可変性管理の課題解決に取り組んだ．本稿では，可変性管理の課題分析に基づき，可変性管理の改善手法および支援ツールを提案する．改善手法は，可変性記述の一覧性と可読性を向上させるための製品-機能マトリクスとデータ仕様書，およびアプリケーション開発におけるコア資産の再利用率を向上させるための可変性記述から製品ソフトウェアを直接ビルドできる構成手段からなる．可変性管理の実践に基づき，可変性記述法と構成手段がコア資産の再利用にどのような影響を与えるかについて，改善前後で比較検証し，コア資産の共有と，製品ソフトウェアの構成決定の容易化によって，コア資産の再利用率が向上するという結果を得た．

1. はじめに

近年，家電製品をはじめとする組込み製品においては，他社との差別化のための高機能化，多様なユーザニーズに応えるための製品バリエーションの充実化，タイムリな市場投入が強い事業要求となっている[1]．

このような事業要求に対応するため，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）[1],[2],[3]が広く実践されている．SPLEとは，ある製品群において，共通かつ管理された機能を共有する一式のソフトウェア・システム群のことであり，体系的な再利用を特徴とする開発手法のことである[2]．

SPLE成功のためのキープラクティスの1つに可変性管理がある．可変性管理とは，対象製品群の各製品間で変わりうる仕様を定義し，複数の製品開発に利用していく活動である[6]．可変性管理が重要とされる理由は，製品間の差異の分析や定義の良し悪しによって，コア資産（共有する設計書やソフトウェア部品）の再利用性が変わり，製品群全体の生産性に影響を及ぼすためである[5],[6]．

可変性管理の主要な技術に，可変性の記述法と構成手段がある[8],[11]．可変性の記述法とは，製品間の差異の分析や定義を行うための表記法であり，可変性の構成手段とは，可変性を製品のソフトウェアとして実現する手段のことである．可変性管理の各技術については，さまざまな研究がなされる一方で，実践に基づく複数手法の比較検証や，特定ドメインに特化した実践が期待されている[9]．

筆者らは，業務用空調機（室外機）を対象に2010年から5年間SPLEを適用してきた（以降，第1期活動と呼ぶ）．第1期活動では，一時的に生産性に大きな改善があったが，時間経過とともにソフトウェア部品の再利用率が低下し，その結果製品群全体の生産性も低下した．この原因のひとつとして，開発上流工程からの体系的な再利用を促す仕組みがなかったことがある[15]．このような課題に対し，コア資産保守プロセスを，開発ロードマップに基づき，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け，幹開発では当該製品のソフトウェア部品を開発しつつ，後続する枝開発で再利用可能なソフトウェア部品を開発するプロセスへと見直した[21]．しかしながらコア資産保守プロセスの改善だけでは，仕様差異を生み出す特徴（フィーチャ）の増加に対して構成選択が難しくなるという可変性管理における課題が残る．

本稿は，第1期活動における可変性管理の問題点の分析結果に基づき，可変性管理の改善手法と支援ツールを提案するとともに，第１期の活動とそれらを適用した第2期の活動を定量的に比較検証する．改善手法は，要求仕様書に製品-機能マトリクスとデータ仕様書を導入することで，可変性記述の一覧性と可読性を向上させるとともに，これら可変性記述から製品ソフトウェアへの実装・選択方法を仕様設計時点で明確にし，かつビルドへ直結する構成手段を提供することで，アプリケーション開発におけるコア資産の再利用率を向上させる．改善手法がコア資産の再利用性にどのような影響を与えるか考察し，製品ソフトウェアの構成決定を容易化することによってコア資産の再利用率が向上するという結果を得た．

本稿第2章では，可変性管理に対する要求事項を整理し，第3章で適用対象の特徴を述べるとともに，可変性管理の改善手法を提案する．第4章では第1期活動からの改善ポイントを説明しつつ，開発における効果を定量的に比較検証する．

2. 可変性管理と技術

2.1 可変性管理

可変性管理とは，対象製品群における可変性を定義し，複数の製品開発に利用してするための活動である．ここで，可変性とは，対象製品群の各製品間で変わりうる仕様のことである[10]．

可変性管理の主要な技術に可変性の記述法と構成手段がある．可変性の記述法とは，製品間の差異の分析や定義を行うための表記法のことであり，可変性の構成手段とは，製品間の可変性をソフトウェアとして実現する手段のことである．

2.2 可変性の記述法

可変性の記述法は，さまざまなものが考案されており，野田らが以下のように整理している[8],[18]．

（1）フィーチャモデル

フィーチャモデルは，製品群の持つフィーチャをツリー構造で階層的に表現する図式表現である[7]．直感的に理解しやすい利点を持つが，さまざまな情報が詰め込まれやすく現実の開発で

の適用が難しいとの指摘もある．

（2）直交可変性モデル

直行可変性モデルは，開発成果物と独立（直交）させて可変性自体をモデル化する．可変性と各開発成果物の関係を明示的に紐づける点に特徴がある[8]．複数の開発成果物に可変性情報が含まれる際に，関係性の理解や維持が容易となる．

（3）DM（Decision Modeling）

DMは，形式的なテキスト表現によるモデルで，可変点ごとに，決定すべき内容と選択肢を記載する[18]．フィーチャモデルや直交可変性モデルのようなグラフ表現に比べ一覧性に優れるが，直感性が損なわれる．

可変性記述は，アプリケーション開発者が製品の具体的な仕様を定義するために用い，その仕様が製品導出への入力となる．そのため，可変性記述法には，アプリケーション開発者が定義・レビューするための製品要求仕様書としての扱いやすさと，コア資産からソフトウェア部品を抽出しやすい情報形態であることが求められる．

2.3 可変性の構成手段

可変性の構成手段とは，可変性の実現手段のことである．Gacekらは，開発言語に依存しない構成手段について，一般的によく使われているものを整理し以下のように分類している[11],[12]．実際には，これらの構成手段を，実応用に合わせて組み合わせたり改良を加えたりして利用されている．

（1）条件コンパイル

条件コンパイルは，#ifdef（コンパイルスイッチ）によって，コンパイルする個所を特定する方法である．条件コンパイルは，既存コードに影響なく追加できるため導入障壁が低い一方で，言語とマクロの2つの言語が混在し，頻繁なコード中断やレイアウト崩れが起き，コードの可読性が悪化しやすい[13]．

（2）フレーム方式

フレーム方式は，直接ソースコードを再利用するのではなく，パターンに基づいて再利用モジュールを生成しフレームワーク部分と結合する方法である．パターンの定義および適用に成功すれば再利用率が向上するが，適用対象であるドメインを十分に分析する必要があるため，導入障壁が高く，適用範囲が狭くなりがちである[14]．

（3）パラメータ方式

パラメータ方式は，セットされたパラメータによって振る舞いを変えるコンポーネントを用いる方法である．再利用性や仕様とのトレーサビリティに優れるが，パラメータに応じた処理をすべてコンポーネントに内包するためコンポーネント自体が複雑になりがちである．

（4）静的ライブラリ

静的ライブラリは，オブジェクトファイルからなるライブラリをコンパイル・リンク時に組み込む方法である．実装時に任意の静的ライブラリを選択することで，必要な部品のみから製品を構成できる．ただし，出荷後の部品の切り替えはできない．

（5）動的リンクライブラリ

動的リンクライブラリは，ライブラリをプログラム実行中に読み込む方法である．メモリ使用量やROMサイズの節約などの利点があるが，実行時の組み合わせで振る舞いが変わるため品質保証が難しい．

可変性の構成手段は，プロダクトラインアーキテクチャおよびソフトウェア部品の形態を決定づける．また，構成手段の選択は，製品導出時のソフトウェア部品の組立て，製品ビルドの効率，ソフトウェア部品の保守のしやすさに影響を与える．したがって，構成手段の選択では，適用するドメインの可変性や開発環境への適合のしやすさが求められる[9]．

3. 可変性管理の提案

3.1　SPLE適用対象

我々は，業務用空調機（室外機）の製品群に対して，SPLEの本格適用を2010年より開始した．この製品群は，日本国内のみならず世界各国へ出荷され，また能力帯や機能の搭載有無，ハードウェア構成の違いがあるため，さまざまな製品バリエーションを持っている．比較的小さな仕様の差異をいかにソフトウェアでカバーするかが開発戦略上重要であり，多品種小変更型開発の特徴を有している．多品種小変更型開発の特徴[21]を以下に記す．

①製品の機能・性能がハードウェアに大きく依存するため，製品群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[4]．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．メカ・エレキ部門（製品部門）は部門ごとに製品用途や出荷地域が異なる固有の市場に対応している．
②製品群は，ソフトウェアから見て，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．
③製品群は，②の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る．
④①および③の特徴から，製品部門からの仕様は，コア資産に基づくものでなく，以前開発した製品に基づいたものとなりがちである（例：前年度製品をベースに，ある機能を変更する等）．
⑤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．

3.2　第1期活動における可変性管理の課題

第1期活動における可変性管理プロセスについて，各コア資産の可変性記述と構成手段およびソフトウェア部品のアプリケーション開発での使われ方を**図1**に示す．

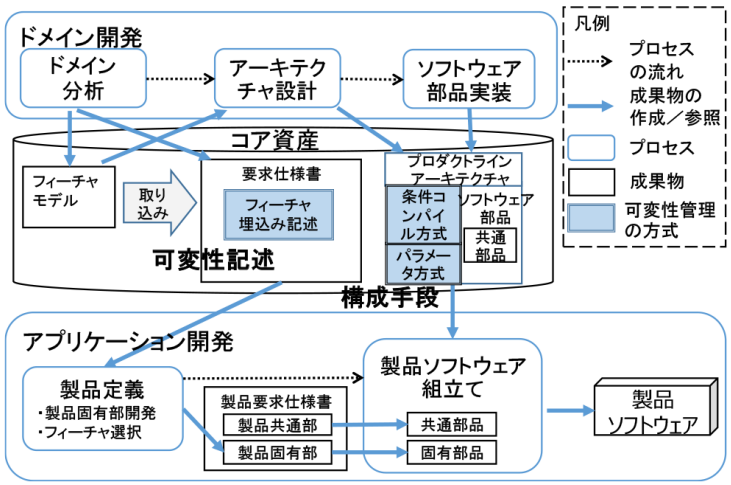

図1　第1期活動における可変性管理手法

ドメイン開発では，コア資産である要求仕様書とプロダクトラインアーキテクチャおよび共通的なソフトウェア部品を開発する．要求仕様書は，フィーチャモデルに基づいて構成しており，製品機能内の仕様差異をフィーチャに基づく可変性として記載する（フィーチャ埋込み記述）．たとえば，圧力保護機能の検出方式の差異である「圧力スイッチによる検出」と「温度推定による検出」の2つ

を，フィーチャに対応する検出方式の仕様（バリアント）として記載している．プロダクトラインアーキテクチャは，条件コンパイル方式を採用し，先述した製品機能内の仕様差異を，ソフトウェア部品内のコンパイルスイッチで実現している．

アプリケーション開発では，当該製品での固有制御を製品固有部として新しいフィーチャとともに製品要求仕様書に埋め込み，ソフトウェアの固有部品として実装する．この固有部品を登録テーブルに登録するとともに，条件コンパイルで有効にして固有の製品ソフトウェアを生成する[15]．また，パラメータ差異はフィーチャの組み合わせによって具体的な値を決定する方式とした（パラメータ方式）．

製品定義（アプリケーション開発）では，コア資産に関する知識を持たない製品部門からの仕様をもとにソフトウェア部門がソフトウェア部品を再利用／開発する（特徴①）ことになるが，複数の開発が並行する（特徴⑤）ため，仕様の解釈時に，可変点の見逃しが頻繁に起き，結果的に類似したソフトウェア部品（固有部品）を開発してしまう[21]．これは可変性管理における以下の2点の課題が原因となっている．

[課題①]要求仕様書の目次構造により，内部に記載された開発済みの制御方式を共有できず，利用可能なソフトウェア部品を使わずに新たな仕様として要求されることが多い．
[課題②]製品系列の増加に伴うフィーチャ数の増加によって，製品部門のフィーチャ選択に基づいて具体的なソフトウェア部品を#ifdefの組合せで決定するのが困難になった．

3.3　可変性管理における提案手法

3.3.1　第2期の可変性管理プロセスと提案手法

第2期活動における可変性管理プロセスについて，各コア資産の可変性記述と構成手段およびソフトウェア部品のアプリケーション開発での使われ方を**図2**に示す．

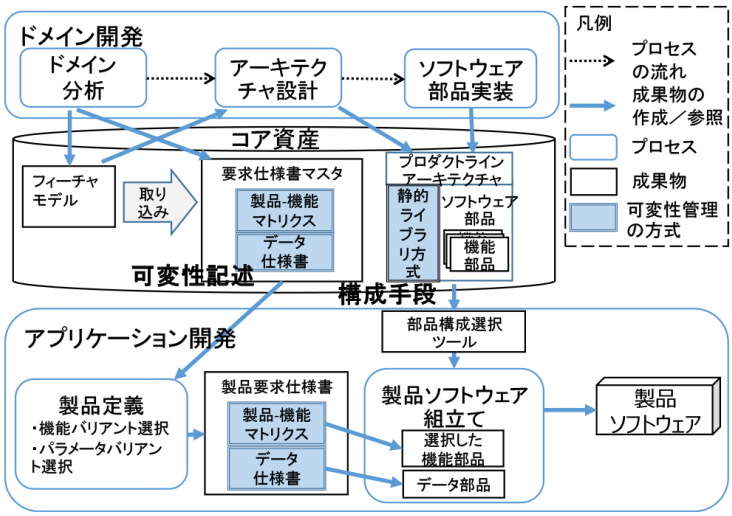

図2　第2期活動における可変性管理

第1期活動からの主な変更点は，以下である．

①要求仕様書の可変性記述法をフィーチャ埋込み記述から製品–機能マトリクスおよびデータ

仕様書へ変更
②プロダクトラインアーキテクチャの構成手段を条件コンパイル方式から静的ライブラリ方式に変更
③構成手段を支援する部品選択ツールを整備

本変更によって，製品部門と開発済みの製品機能を共有できること（課題①への対応）と，ソフトウェア部品の選択を容易にすること（課題②への対応）を狙っている．

3.3.2 可変性記述法の提案

提案手法における可変性記述法（変更点①）を説明する．要求仕様書の構成を**図3**に示す．

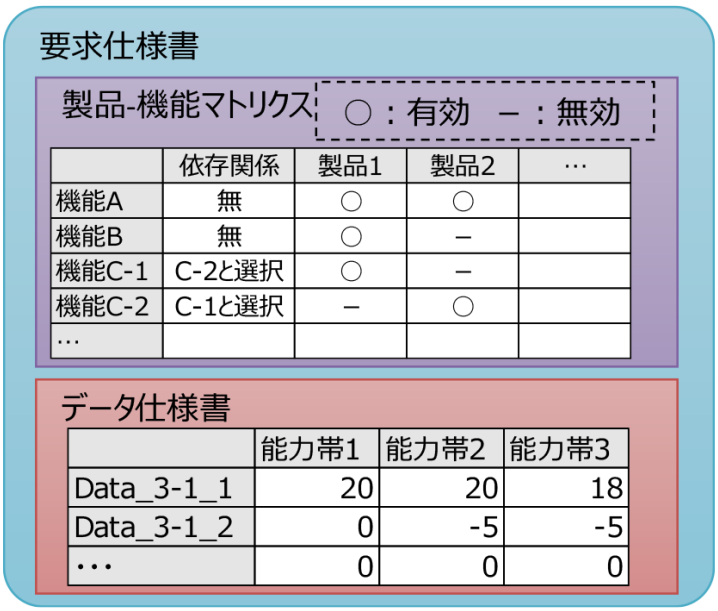

図3　要求仕様書の構成

要求仕様書（要求仕様書マスタ）は製品–機能マトリクスとデータ仕様書からなる．製品–機能マトリクスは，縦軸に製品群に搭載されている機能，横軸に製品として，交点に各製品での機能の有効／無効を明示した表である．縦軸の機能一覧においては，全機能を列挙するとともに，制御方式に差異がある類似機能を依存関係とともに明示している．図3においては，製品1，製品2を含む製品群全体で機能A，機能B，機能C-1，機能C-2を有する可能性があることを意味し，機能C-1と機能C-2はあるフィーチャによって機能が択一的に選択される機能であることを示している．製品1は機能A，機能B，機能C-1を有する製品であることを定義している．機能C-1および機能C-2とは，具体的には，圧力スイッチによる圧力保護と温度センサによる圧力推定を行う圧力保護をそれぞれ別の製品機能として定義し，製品機能の目次レベルで可変性があることを明示するとともに，択一的な関係であることを示す．

データ仕様書とは，製品個々のパラメータを一括して記載する文書である．チューニング可能なパラメータに対し，[Data_3-1_1]のようなタグを本文中に埋め込んでおき，タグの付いたパラメータの設定値をデータ仕様書に記載する．たとえば，Data_3-1_1は温度閾値を意味し，能力帯1と能力帯

2では20度だが，能力帯3では18度であることを表す．この見直しは，パラメータ差異がフィーチャの組み合わせだけで決定せず，ハードウェア構成や出荷地域等の複合的な要因により決定するという分析結果に基づいている．

要求仕様書マスタを用いた開発例を図 2に基づいて示す．ドメイン開発において作成した要求仕様書マスタには，開発済みの製品機能と製品機能のパラメータが記載できるデータ仕様書が含まれている．アプリケーション開発では，要求仕様書マスタをカスタマイズすることで，個別の製品要求仕様書を作成する．具体的には，各アプリケーション開発（製品定義）において，搭載する製品機能を選択することで製品－機能マトリクスの担当製品に対応する縦の欄を完成させ，当該製品用のデータ仕様書に能力帯やハードウェア構成の違いによるパラメータ設定を記載する．

表1に第1期と第2期の可変性記述法の違いを整理する．

表1　可変性種別と要求仕様書の対応関係

可変性種別	第 1 期	第 2 期
機能有無	目次に登場する機能	製品 - 機能マトリクス
制御方式	機能内の可変点	
パラメータ		データ仕様書

第1期活動においては，制御方式やパラメータの差異は製品機能内の可変点として記述していたが，第2期活動では，制御方式が異なる製品機能を異なる製品機能として定義したうえで，すべての製品機能を製品–機能マトリクス上に列挙し，機能の搭載有無を表の交点に記載している．また，データの差異については個々に可変点として記述する方式から，データ仕様書に一括で記載する方式に切り替えている．すわなち，具体的なバリアント（制御仕様）を決めるために複数のフィーチャを段階的に選択する方式から，制御仕様を直接的に選択できる方式に切り替えた．

これらの方式の切り替えによって以下の2点が改善した．

①要求仕様書における可変性の一覧性向上

第1期では，500ページ程度の要求仕様書の中から可変点を探し出し，当該製品に適合するバリアント（制御仕様）を選択する必要があったが，第2期では，5ページ程度の製品–機能マトリクスの中から対象の製品機能を見つけ出し制御仕様を確認の上，選択することができるようになった．このように可変点およびバリアント（制御仕様）の検索が容易となったため，開発済みの製品機能の再利用率が向上した．

②要求仕様書における可変性の可読性向上

第1期では，目次に登場する機能の搭載有無を決定したのちに，機能内の可変点ごとに，フィーチャを組み合わせてバリアント（制御仕様やパラメータ）を決定していたが，第2期では，製品–機能マトリクスによって製品機能を取捨選択し，各機能のパラメータを当該製品用のデータ部品によって決定するという構成としたため，具体的なバリアントを決定するのが容易になった．製品–機能マトリクスは，製品機能を取捨選択する（搭載有無を決定する）というシンプルな表記法となっているため，熟達技術者でなくともその意味を理解することが容易であり，プロダクトラインアーキテクチャを理解していない製品部門でもコア資産の確認が容易となる．

3.3.3　可変性の構成手段の提案

可変性記述の見直しに基づき，仕様設計時点で明確にしたソフトウェア部品としてのモジュール性をそのままに設計・実装できるようにするため，**表2**に示すように，構成手段を静的ライブラリ方式に変更し，パラメータについてはフレーム方式を併用することとした．

表2　可変性種別と構成手段の対応関係

可変性種別	第 1 期	第 2 期
機能有無	条件コンパイル方式	静的ライブラリ方式(パラメータはフレーム方式と併用)
制御方式差異		
パラメータ	パラメータ方式	

機能有無や制御方式差異に対する構成手段として，第1期活動では，実装のしやすさから条件コンパイル方式を採用していた．これに対して，第2期活動では，すべての機能が製品ー機能マトリクスによって，機能の有無で表現できるようにするため，静的ライブラリ方式へと見直した．

パラメータ差異に対する構成手段ついても同様に，構成手段をすべて静的ライブラリ方式に統一した．これは，可変性記述法の見直しによって，パラメータ差異を，製品機能内のバリアントから製品ごとのバリアント（データ部品）として扱うことにしたためである．さらに，データ部品については，パラメータやソースコード自体を再利用対象とするのではなく，データ仕様書からソースコードへのパターン化した変換ロジックを再利用対象としている（フレーム方式）．具体的には，データ仕様書の各パラメータは，能力帯によらず一定の値をとる定数型，および能力帯ごとに設定値が異なる能力帯テーブル型の2種で記載し，パターンに応じて自動的にソースコードと取得関数に展開している．

これらの構成手段の変更によって，仕様からソフトウェア部品の実装方法が単純化しソフトウェア部品開発時の作業者の担当分けのしやすさや実装ミスの減少が効果として得られ，また製品との対応付けが明確になり，ビルド管理を含む構成管理も容易となった．

3.4　可変性管理におけるツール支援

構成手段の見直しに合わせ，構成選択を効率化する目的で支援ツール（以降，部品構成選択ツール）を開発した．構成を**図4**に示す．

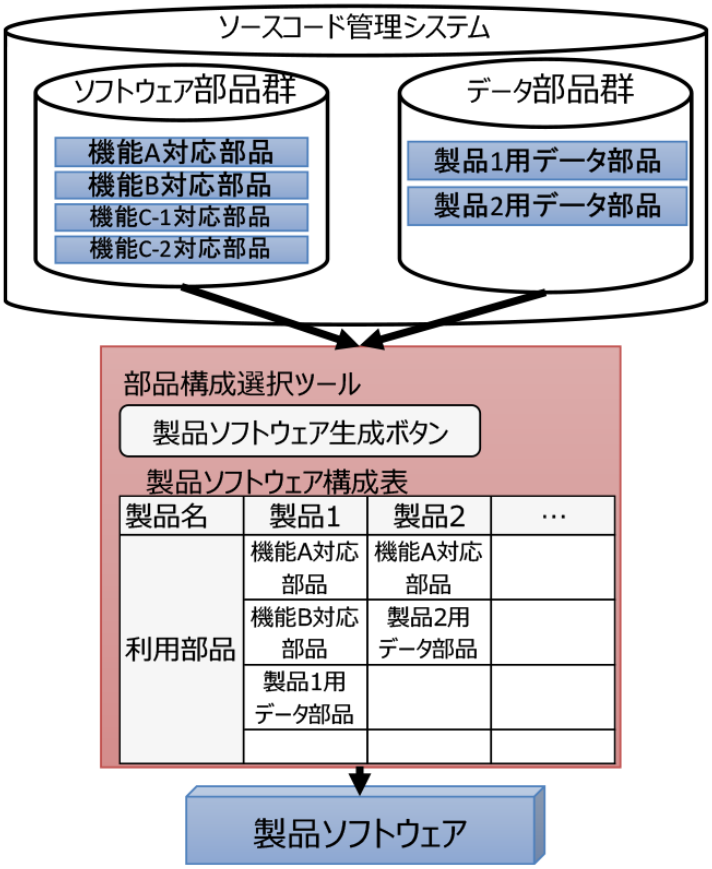

図4　部品構成選択ツールの構成

部品構成選択ツールは，製品ソフトウェア構成表と製品ソフトウェア生成ボタンから構成されるツールで，ソースコード管理システム上にあるソフトウェア部品群とデータ部品群を入力として，製品ソフトウェアを出力する．製品ソフトウェア構成表は横軸に製品，縦軸に当該製品で利用するソフトウェア部品を列挙した表で，要求仕様書の製品－機能マトリクスに基づいて生成する．

製品ソフトウェア構成表の中からビルドしたい製品を選択して，製品ソフトウェア生成ボタンを押下することで，各ソフトウェア部品をソースコード管理システムから取得しMakefileを生成する．このとき，データ部品は，データ仕様書をもとにデータ形態に応じて取得関数を含むソースファイルに変換され，Makefileに組み込まれる．このように生成されたMakefileを実行することで製品ソフトウェアを生成できる．

このような支援ツールを開発することによって，仕様設計時点で明確になったソフトウェア部品としてのモジュール性を活かして効率的に製品ソフトウェアを構成することが可能となる．

4. 提案手法の適用評価

4.1　第1期からの改善ポイント

要求仕様書における可変性記述法においては，フィーチャ埋込記述から製品–機能マトリクスおよびデータ仕様書に見直すことで，製品部門が開発済みの製品機能を把握・選択することが容易となる，つまり仕様が共通化されることを狙っている．

また，可変性の構成手段においては，条件コンパイル方式から静的ライブラリ方式に見直すことで，製品ソフトウェア組立て時に，各ソフトウェア部品のリンク有無のみを決定すればよくなる．つまり，製品構成決定のパラメータ数が低減できることを狙っている．

4.2　開発における評価

第3章で述べた改善手法を，2015年度開発製品より適用した．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　仕様共通化率Rc

対応製品にあった仕様を見落としなく選択できてきることを評価するために，仕様共通化率を用いる．仕様共通化率Rcは次式として定義する．

$$Rc = （A / B） \times 100 \quad (1)$$

A：複数製品で使用される機能数
B：製品群全体での全機能数

なおRcは比較のため，2014年を1.00とした相対値Rc-yyで示す．Rc-yyは次式と定義する．

$$Rc\text{-}yy = yy年のRc \ / \ 2014年のRc \quad (2)$$

仕様共通化率は複数製品で利用された仕様の割合を示す指標となっている．本手法においては，3.3.2項の要求仕様書において選択する仕様に対応して，粒度の揃った製品機能を部品化していること，各部品内にはパラメータ差異以外の可変点を埋め込まない方式を採用している．そのため，仕様共通化率を用いることにより，コア資産として作成した製品仕様がどれだけ予定とおり再利用されているかを評価できると考える．

図5に示すように，第1期活動では仕様共通化率は1.00～1.35倍と変動がなかったが，第2期活動においては，15年に2.56倍，16年には2.68倍と，約2.5倍の仕様共通化率を達成している．

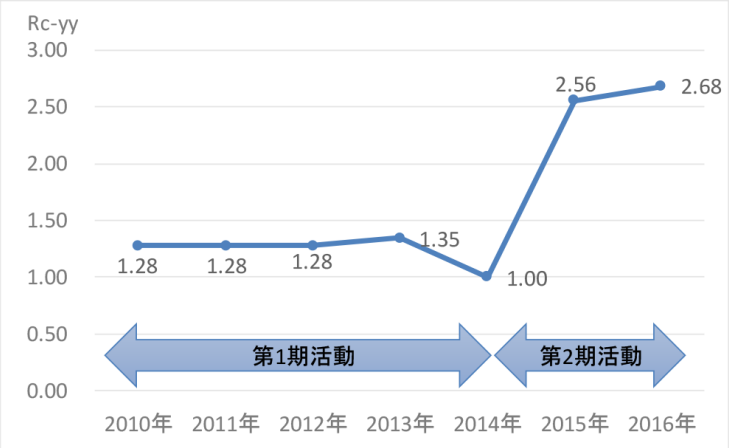

図5　仕様共通化率

これは，第1期活動では，仕様書の目次構造の問題から製品部門と開発済みの制御方式が共有できておらず，類似した新たな仕様を開発してしまっていたが，第2期活動では，製品–機能マトリクスで一覧化できることによって500ページもの要求仕様書の中から可変点を探すのではなく，5ページ程度のマトリクスにすべての可変点が表現されたことで，製品部門との共有が促進できた成果と考える．

4.2.2　製品の構成決定パラメータ数Pp

製品機能の具体的な仕様を効率的に決定できていることを評価するために，構成決定パラメータ数を用いる．パラメータ数Ppは次式と定義する．

Pp = average（Nf／製品機能）　（3）
Nf：フィーチャ数（選択肢の数）

なおPpは比較のため，2014年を1.00とした相対値Pp-yyで示す．Pp-yyは次式と定義する．

Pp-yy=yy年のPp　/　2014年のPp　（4）

構成決定パラメータ数は，1つの製品機能を実現するためにいくつの可変点を決定しなければならないかを示す指標となっている．本改善手法においては，ソフトウェア部品内にはパラメータ差異以外の可変点を埋め込まない方式としているため，目論見とおり機能有無の選択とパラメータチューニングのみで製品機能を決定できているかを評価できる．

図6に示すように，第1期活動では，Ppが年々増加しているのに対し，第2期活動では，Ppが56%低減し，その後の増加はない．なお，第2期活動でPpが減っているのは，本手法の導入に合わせて既存のコンパイルスイッチを削減するリファクタリングを実施したためである．

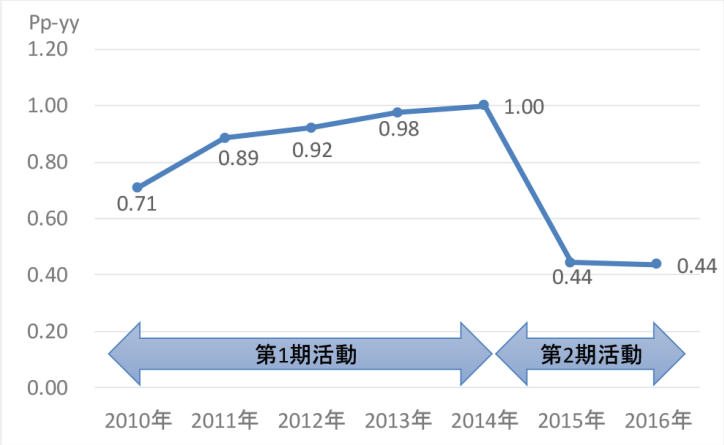

図6　製品構成決定のパラメータ数

第1期活動では，製品系列の増加に伴いフィーチャ数が増加し，コンパイルスイッチが部品内に散在する結果となったが，第2期活動においては，部品のリンクおよびデータ部品の決定のみでよくなったったため，フィーチャ数が増加しても，構成決定が容易な状態を維持できた．

本手法はコア資産保守プロセスの見直しと合わせて適用することで，幹開発で開発したソフトウェア部品の枝開発における認知度が向上し，さらに，枝開発における構成決定が容易化できたことで，ソフトウェア部品の再利用率向上を実現している[21]．

4.3　妥当性に対する脅威

本改善手法における製品–機能マトリクスは，製品と具体的な仕様（バリアント）を直接結び付ける表であり，バリアント選択の根拠となるフィーチャが登場しない．そのため，1つの可変点に多くのバリアントが存在するような製品群の場合，なぜその仕様を選択しているのか，その根拠がわかりにくくなり，後述するディシジョンモデルと比較して，製品部門が正しくバリアント（制御方式）を選択できないという問題も発生しうるが，現時点ではバリアントの選択肢が多岐にわたる場合が少なくその選択理由が比較的自明であるため，特に大きな問題を生じていない．

データ仕様書およびデータ部品については，多くの製品群ではパラメータの違いで製品間の差異を実現しているものも多いと考えられるので，個々のパラメータの違いを可変性とするのではなく，データ部品としてまとめて可変性としてしまう手法は他のドメインでも有効と考える．

5.　関連研究

5.1　可変性記述法

可変性記述法に関する研究は数多く行われている．Czarneckiらは，レベルの異なるフィーチャが混在することによるフィーチャモデルの崩壊を回避するため，段階的構成を提唱している[16]．段階的構成では，フィーチャモデルを段階的に特殊化し，製品の構成を決定していくやり方を採る．ClassenらもMLSC（Multi-Level Staged Configration）と呼ばれる複数レベルの段階的構成によって製品構成を実現している[17]．これは，1つのモデルと表によって各ステークホルダとの共有と製品構成を実現した我々のやり方とは異なっている．

1つのフィーチャモデルと表によって製品導出を実現している事例もある．Czarneckiらは，フィーチャモデルとディシジョンモデルによって共通性と可変性の抽出および製品導出支援を実現している[18]．ディシジョンモデルは，縦軸にフィーチャ，横軸にフィーチャを決定するための質問やフィーチャの型や範囲が記述されており，フィーチャ決定を支援している．フィーチャモデルと表を併用するという点で我々の取組みに近いが，表を機能と製品の星取表とし製品導出に利用した点が異なっている．

5.2　可変性の構成手段

Faulkらは最上位のクラス図で製品群の共通性を表し，サブクラスにより可変性を表現する手法を提案している[19]．Faulkらの提案は，サブクラスを切り替えることで製品固有のニーズを実現している点に特徴がある．本稿は，類似機能を別機能として扱う点，つまり最上位のレベルで可変性を切り替えている点がFaulkらの取り組みと異なっている．

Greenfieldらは，モデル駆動開発によって製品ソフトウェアの生成を実現している[20]．本取り組みは，OOA&D（object oriented analysis and design）をベースとしたモデル駆動開発で，モデル内部で可変性を管理し，コードを自動生成している点に特徴がある．パラメータ差異等の小さな可変性によって変異体を増加させない点に本稿との差異がある．

6. さいごに

本稿では，空調機開発を対象に，コア資産保守プロセスの見直しと合わせて導入した可変性管理の改善手法を評価した．改善手法は，要求仕様書に製品-機能マトリクスとデータ仕様書を導入することで，可変性記述の一覧性と可読性を向上させ，仕様設計時点で可変点およびバリアントの選択方法を明確にするものである．さらにこれら可変性記述から製品ソフトウェアのビルドへ直結する構成手段を提供することで，アプリケーション開発におけるコア資産の再利用率を向上させる．本改善手法によって，ソフトウェアへの理解の浅い製品部門とコア資産を共有することに成功し，仕様共通率を約2.5倍に高めている．さらに構成決定パラメータ数を56%低減し，構成決定を容易化している．この結果，コア資産の再利用率向上を実現している[21]．これらの評価結果に基づき，可変性管理はステークホルダとのコア資産共有と構成決定を容易化することが重要であるという知見を得た．

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）Clements, P. and Northrop, L. : Software Product Lines : Practices and Patterns and Addison-esley (2002).
3）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).
4）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).
5）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).
6）野田夏子：ソフトウェア再利用の新しい波—広がりを見せるプロダクトライン型ソフトウェア開発—：2. プロダクトラインの可変性管理—可変性のモデル化とアーキテクチャ設計，情報処理，Vol.50, No.4, pp.274-279 (2009).
7）Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E. and Peterson, A. S. : Feature-oriented Domain Analysis (FODA) Feasibility Study (No. CMU/SEI-90-TR-21).

Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst (1990).
8）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).
9）Bosch, J., Florijn, G., Greefhorst, D., Kuusela, J., Obbink, J. H. and Pohl, K. : Variability issues in Software Product Lines. In International Workshop on Software Product-Family Engineering, pp.13-21, Springer (2001).
10）ISO/IEC 26550 : 2015 Software and Systems Engineering—Reference Model for Product Line Engineering and Management
11）Gacek, C. and Anastasopoules, M. : Implementing Product Line Variabilities. In ACM SIGSOFT Software Engineering Notes, Vol.26, No.3, pp.109-117, ACM (2001).
12）Svahnberg, M., Van Gurp, J. and Bosch, J. : A Taxonomy of Variability Realization Techniques. Software : Practice and Experience, Vol.35, No.8, pp.705-754 (2005).
13）Kästner, C. : Virtual Separation of Concerns : Toward Preprocessors 2.0, It-Information Technology Methoden und Innovative Anwendungen der Informatik und Informationstechnik, Vol.54, No.1, pp.42-46 (2012).
14）Neighbors, J. M. : Draco : A Method for Engineering Reusable Software Systems, Software Reusability, 1, pp.295-319 (1989).
15）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, pp.220-226, ACM (2016).
16）Czarnecki, K., Helsen, S. and Eisenecker, U. : Staged Configuration Through Specialization and Multilevel Configuration of Feature Models, Software Process : Improvement and Practice, Vol.10, No.2, pp.143-169 (2005).
17）Classen, A., Hubaux, A. and Heymans, P. : A Formal Semantics for Multi-level Staged Configuration, VaMoS, Vol.9, pp.51-60 (2009).
18）Czarnecki, K., Grünbacher, P., Rabiser, R., Schmid, K. and Wąsowski, A. : Cool Features and Tough Decisions : A Comparison of Variability Modeling Approaches, In Proceedings of the Sixth International Workshop on Variability Modeling of Software-intensive Systems, pp.173-182, ACM (2012).
19）Faulk, S. R. : Product-Line Requirements Specification (PRS) : An Approach and Case Study, Fifth IEEE International Symposium on. IEEE, pp.48-55 (2001).
20）Greenfield, J. and Short, K. : Software Factories : Assembling Applications with Patterns, Models, Frameworks and Tools, In Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, pp.16-27, ACM (2003).
21）長峯　基，中島　毅：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，デジタルプラクティス，Vol.10, No.3，pp.639-655 (2019).
22）Berger, T., Rublack, R., Nair, D., Atlee, J. M., Becker, M., Czarnecki, K. and Wąsowski, A. : A Survey of Variability Modeling in Industrial Practice, In Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems, p.7, ACM (2013).

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．

井上克郎（正会員）inoue@ist.osaka-u.ac.jp
　1984年大阪大学大学院基礎工学研究科博士後期課程修了（工学博士）．同年大阪大学 基礎工学部情報工学科助手．1984年～1986年，ハワイ大学マノア校コンピュータサイエンス学科助教授．1991年大阪大学基礎工学部助教授．1995年同学部教授．2002年より大阪大学大学院情報科学研究科教授．ソフトウェア工学，特にコードクローンやコード検索 などのプログラム分析や再利用技術の研究に従事．

投稿受付：2019年11月6日
採録決定：2020年2月25日
編集担当：福田晃（九州大学大学院）