マルチプロダクトライン開発における 可変性の構造分析に基づく アジャイルアプリケーション開発方法の提案と評価

林 健吾[†1]　青山 幹雄[†2]

概要：本稿では，マルチプロダクトライン開発において，可変性の構造分析に基づくアジャイルアプリケーション開発方法を提案する．プロダクトライン開発では，ドメイン開発とアプリケーション開発に問題領域を分割して製品要求の多様性に対応する．自動車システムのソフトウェア開発は多様で俊敏な開発が求められているが，多様性と俊敏性に対応した開発方法は未確立である．本稿ではマルチプロダクトライン開発における可変性構造を OVM 拡張モデルを利用して分析する方法を提案する．分析結果に基づき，開発要素を依存関係に応じて分割しインクリメンタルに開発するアジャイルアプリケーション開発方法を提案する．提案方法を自動車ソフトウェアシステム開発に適用し，可変性の複雑性に対応したテストの負荷平準化と開発俊敏性の向上効果により提案方法の有効性を確認した．

An Agile Application Development Method Based on Variability Structure Analysis for Multiple Product Line Development

KENGO HAYASHI[†1]　MIKIO AOYAMA[†2]

1. はじめに

ソフトウェアプロダクトライン開発（以下，SPLE）は，低コスト，高品質で製品系列における多様な製品を開発するアプローチである[4][18]． SPLE では，ドメイン開発とアプリケーション開発の 2 つの開発領域で多様性に対応する．ドメイン開発では，製品系列における共通性と可変性を分析してコア資産を構築する．アプリケーション開発では，コア資産から個別製品を開発する．要求の多様性に対応するために，自動車ソフトウェア開発でも SPLE は多くの開発現場で導入されている[11][20].

自動車ソフトウェア開発では，複数のプロダクトラインを並行して開発するマルチプロダクトライン開発（以下，MPLE）が実践されている[21]．例えば，Bosch のガソリンシステム開発では，市場セグメントごとにプロダクトラインを構築し分けている[15]．その他の分野でも，コンポーネントベースのアーキテクチャにおいて，相互作用や相互依存するコンポーネントごとにプロダクトラインを構築し，それらを組合わせることで，マルチプロダクトラインが構成されている[22].

一方，自動車システムを含むソフトウェアシステムの開発では，俊敏な製品進化が求められている．俊敏な製品進化に対しては，短いサイクルでインクリメンタルに製品を進化させるアジャイルソフトウェア開発（以下，ASD）が実践されている[3][9][13]．俊敏な進化と多様性に対応するため，ASD と SPLE を統合したアジャイルプロダクトライン開発（以下，APLE）も実践されている[5][9].

筆者らが従事している超音波センサシステム開発では，MPLE のアプリケーション開発に，ASD の反復型プロセスモデルを適用し，APLE の実践に取り組んでいる[8]．複数の並行するアプリケーション開発をまとめて反復的に開発管理することで，MPLE の開発における複雑性を低減し，管理性向上を実現した．

しかし，個々のアプリケーション開発で，テストの実施に負荷が集中する課題が生じた．自動車ソフトウェア開発では，可変点（Variation Point）と変異体（Variant）の組合せは膨大であり，かつ依存性が高い[23]．このため，可変点に対する実装とテストを一括して実施していた．その結果，実装とテストの負荷が突出して高くなり，テスト環境の処理能力を超えてリソースを効率的に運用できなかったり，開発終盤で不具合が検出されて後戻りを生じたり，本来であれば ASD で解決されるべき課題が残存した．

本稿では，上記の課題を，SPLE における可変性の構造を分析して，個々のアプリケーション開発をインクリメンタルな開発で実現することで解決できると仮説を立てる．可変性の構造とは，可変点の開発を制約する，可変点と変異体の組合せに生じる依存関係である．本課題を解決し，MPLE における開発工数とテスト環境の負荷を平準化することを目的に，以下の課題を定める．

Q1 : 可変性の構造をモデル化し可変点の開発制約を明らかにする方法は何であるか？

Q2 : 可変性の構造分析に基づきアプリケーション開発の負荷を平準化し，開発を俊敏化する開発プロセスはどのよ

†1 (株)デンソー
DENSO CORPORATION
†2 南山大学
Nanzan University

うなものであるか？

Q3 : 提案方法は実際の自動車システムの APLE で有効であるか？

本課題の解決方法として，可変性の構造分析に基づく，アプリケーション開発方法を提案する．本開発方法を筆者らの自動車システムの APLE に適用し，開発工数とテスト環境の処理負荷が平準化できることを確認した．

2. 関連研究

2.1 SPLE

SPLE は，低コスト，高品質で製品系列における多様な製品を開発するアプローチである[4][18]．SPLE では，ドメイン開発とアプリケーション開発の 2 つの開発領域で多様性に対応する．

ドメイン開発では，製品系列における共通性と可変性を分析してコア資産を構築する．共通性とは，製品系列内の製品間で同一の特徴点の集合である．可変性とは，製品系列内の製品間で差異が現れる特徴点の集合である．例えば，ある製品系列で警報機能を常に備えて機能差がないのであれば、その製品系列において警報機能には共通性がある．自動車のエンジンがガソリンかディーゼルかモータかで何らかの振る舞いを変えるのであれば，その製品系列においてエンジンには可変性がある．コア資産は，製品を開発するためのソフトウェア，設計，テスト仕様，プロセスなど，開発に利用可能なあらゆる資産を指す．

アプリケーション開発では，コア資産から個別製品を開発する．アプリケーション開発の開発プロセスでは，要求開発，設計，実装，テストを順次実行する[18]．各プロセスは，コア資産で定義した可変点を軸に，プロセス，テスト仕様も含めて資産を再利用し，可変性をコンフィグレーションして構築することで品質，期間，コストの最適化を図る[8]．

2.2 APLE

APLE は，SPLE と ASD を統合したアプローチである[5][8][9]．これらのアプローチでは，SPLE における製品系列の多様性に対応する労力の低減と，ASD における俊敏な製品開発[3][9][13]の両方の利点を活用することを目指している．例えば，ドメイン開発に ASD を適用し，コア資産を俊敏に開発する方法が提案されている[7]．また，製品系列の特徴点であるフィーチャの決定に顧客を巻き込む方法が提案されている[16]．その他，製品系列において製品進化と並行しながらコア資産を運用する方法の提案[19]など，2 つのアプローチの統合方法は様々である．

2.3 可変性モデルと分析方法

SPLE では製品系列における製品同士の共通性と可変性の識別と分析を開発の基礎活動としている[18]．可変性には製品外部から観察できる外部可変性と，設計の詳細化で生じ，外部観察できない内部可変性に分類できる[18]．

外部可変性のモデル化には FODA（Feature-Oriented Domain Analysis）を代表とするフィーチャモデルが提案されている[12][17]. フィーチャは製品を外部から特徴づける因子である．FODA はフィーチャ同士の依存関係を，プロダクトラインを根とする木構造で表現し，分析することを可能とする．

内部可変性を含む可変性モデルとして, OVM（Orthogonal Variability Mode）が提案されている[2]．OVM はフィーチャモデルと異なり，共通性を省いて可変性に特化してモデル化する[14][18]．図 1 に，OVM の表記法とメタモデルを示す．OVM では可変性として，製品系列に存在する可変点と変異体を識別し，可変点同士，変異体同士，可変点と変異体の依存関係をモデル化する．可変点（Variation Point）は可変する特徴点であり，変異体（Variant）は可変点における選択肢である．例えば，2.1 の可変性の例では，エンジンが可変点であり，ガソリン，ディーゼル，モータが変異体となる．

OVM では，可変点，変異体と，製品開発プロセスで生成される成果物資産との依存関係も表現する．分析対象に応じて，OVM モデルは拡張されている[1][6][14].

これら，可変性の捉え方と分析方法は，SPLE の開発方法を決定づける重要な因子である．

2.4 自動車システム開発における APLE の適用

自動車システム開発において，APLE の適用例は発行媒体では確認できないことが報告されている[9]. Hohl らのサーベイでは，自動車システム開発は可変点と変異体が膨大であることによる[23]，可変性管理の複雑性の高さを，

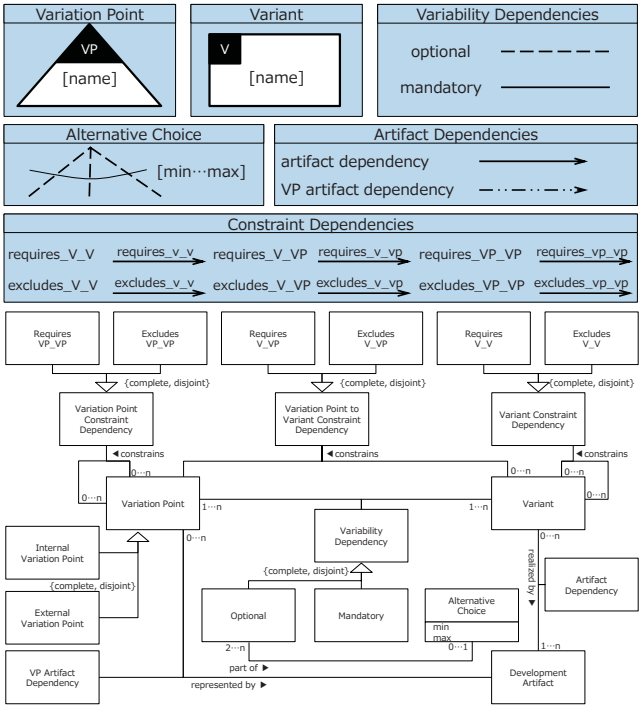

図 1 OVM 表記法とメタモデル [14][18]

Figure 1 OVM notation and meta-model [14][18]

APLE の適用を妨げる要因のひとつとして挙げている[9]. 筆者らが自動車の超音波センサシステム開発で適用した APLE[8]は数少ない適用例のひとつであるが，開発マネジメントを主眼としており，可変性管理については検討していない．自動車システム開発において，ASD と SPLE を統合した開発方法は未確立である．

3. アプローチ

3.1 研究課題とアプローチ

本稿では，前述の 3 つの課題に対して以下のアプローチを提案する．

Q1 に対して，OVM を拡張して可変性の構造を分析し，可変性における可変点と変異体の組合せに生じる依存関係から，開発の制約を明らかにする方法を提案する．

Q2 に対して，可変性の構造分析で明らかになった可変点の開発順序の制約に基づき，アプリケーションをインクリメンタルに開発することで，開発工数とテスト環境の負荷を平準化する開発モデルを提案する．

Q3 に対して，自動車システムの APLE として，筆者らが従事している超音波センサシステム開発に提案方法を適用し，その効果を評価する．

3.2 可変性の構造分析の満たすべき要件

Q1 は Q2 のアプローチの前提となる．Q1 で明らかにされる開発制約は，Q2 を解決するための条件を満たす必要がある．図 2 に，本稿のアプローチにおける，可変性の構造分析とアプリケーション開発モデルの関係を示す．

アプリケーションを ASD のフレームワークでインクリメンタルに開発するためには，開発要素（ストーリ）が INVEST であることが望ましい[3]．INVEST とは，Independent（独立している），Negotiable（対話を引き出す），Valuable（ユーザ価値を提供する），Estimable（見積り可能である），Small（小さい），Testable（テスト可能である）の各単語の頭文字をとった指針である．これらの内，Negotiable，Valuable，Estimable は新たな機能や可変点の開発では重要であるが，MPLE のアプリケーション開発の設計済みの可変点の変異体実装では，条件が満たされている．

Q1 では，INVEST の Independent，Small，Testable を満たすべく可変点の開発を分割できることが条件となる．

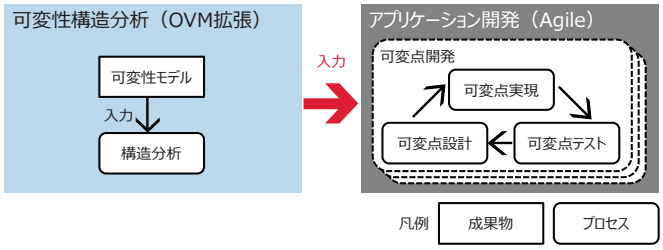

図 2 可変性構造分析とアプリケーション開発の関係

Figure 2 Relation between Variable Structure Analysis and Application Engineering

Independent については，可変点の部分集合が，その他の可変点の集合に対して依存関係を持たなければよい．

Small は定性的な尺度であるが，開発チームが定めた反復単位であるタイムボックス[3][13]の期間に収まるか否かである．依存関係を外に持たない部分集合が Small を満たさない場合はさらに分割する必要がある．

Testable は依存関係のある可変点のテスト可能性を表す．依存先の可変点が実装されていないと，依存する可変点は Testable でない．Testable であるためには，依存される可変点は，その他の可変点に先立って開発される必要がある．

したがって，Q1 では，Q2 で Small な可変点の開発に分割できるように，可変性の構造を分析することで，Independent で Testable を満たす可変点の開発順序の制約を分析できることが条件となる．

4. 可変性のモデル化と構造分析

MPLE において ASD を適用するためには，可変点の集合を独立した部分集合に分割する必要がある．そのため，本稿では，OVM モデルを拡張して，可変性の構造を分析し，可変性における可変点と変異体の組合せで生じる開発順序制約を分析する方法を提案する．

4.1 可変性構造分析のための OVM 拡張モデル

可変点の依存関係を整理し，依存関係と開発における分割方法を対応付ける．その結果として，新たな依存関係を導入し，新たな依存関係を表現可能とする OVM の拡張モデルと表記法を示す．

4.1.1 可変点の依存関係と分割方法

可変点の依存関係と分割方法を整理すると，3 つのパターンに分類できる（図 3）．2 つの可変点 vp1 と vp2 の依存関係を例として以下に示す．{*} は開発ストーリとしてまとめられる可変点の集合を示す．VPo は vp1，vp2 と依存関係のない，vp1，vp2 以外の可変点の集合を示す．→は依存関係による開発の順序制約を示す．

(1) vp1 と vp2 に依存関係が存在しない場合：

{vp1，vp2，VPo}は（{vp1, VPo}, {vp2}）または（{vp1},

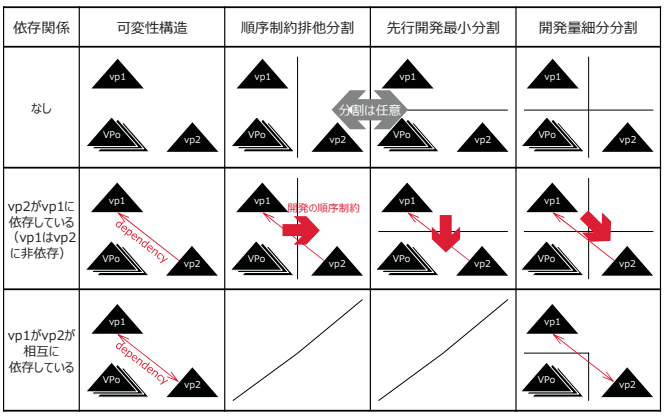

図 3 可変性の構造と開発分割方法

Figure 3 Variability Structure and Development Dividing

{vp2, VPo})，({vp1}, {vp2}, {VPo}) に分割できる．集合同士は任意の順序で開発することができる．

(2) vp2 が vp1 に依存しており，vp1 は vp2 に依存していない場合：

{vp1, vp2, VPo}は ({vp1, VPo}→{vp2}) または ({vp1}→{vp2, VPo})，({vp1}→{vp2}, {VPo}) に分割できる．vp2 を含む集合が，vp1 を含む集合よりも後で開発する制約が生じる．

(3) vp1 と vp2 が相互に依存している場合：

{vp1, vp2, VPo}は ({vp1, vp2}, {VPo}) だけに分割できる．vp1 と vp2 の開発は分割することはできない．vp1 と vp2 はいずれかを統合テストする前には，共に実装が完了していることが制約として生じる．

4.1.2 OVM 拡張モデルと表記法

既存の OVM（図 1）では，依存関係として Requires（必要とする）と Excludes（排他する）が定義されている．これらの依存関係では前項で示した(3)のパターンを明示できないため，新たな依存関係として Co-Requires（共に必要とする）を導入する．

Requires，Excludes，いずれの依存関係も付与されていない可変点同士は，前項(1)のパターンとして識別できる．いずれかの依存関係が付与されている場合は，前項(2)のパターンとして識別できる．具体的な分析方法は 4.2 で示す．

前項(3)のパターンは，Requires の依存関係が双方向に成立している場合に規定される．双方向の依存関係を 2 つの単方向の依存関係で表現することは煩雑となる．そのため，Co-Requires（共に必要とする）という依存関係を定義し，双方向の依存関係を明示できるよう OVM を拡張する．図 4 に，拡張した OVM の表記法とメタモデルを示す．

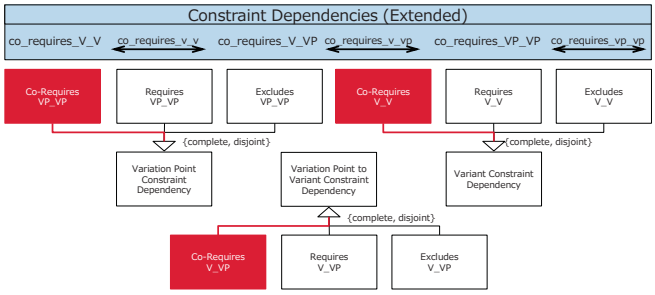

図 4 OVM の拡張表記法とメタモデル

Figure 4 Extention of the OVM notation and meta-model

OVM 拡張モデルを利用して可変性の構造を表現した例を図 5 に示す．本稿では，関連した表記法も拡張した．従来の OVM では，可変点から開発資産への依存矢印は，可変性構造図から他の開発資産の表現図へ直接依存関係を矢印線として記述している．これに対し本稿では，分析時に関心事に集中できることを目的として，複数開発資産と分析図を分割した．そのため，可変点，変異体に識別番号を付与し，各開発資産の表現図内で識別番号領域（▲や■）による関連付けを可能とした．

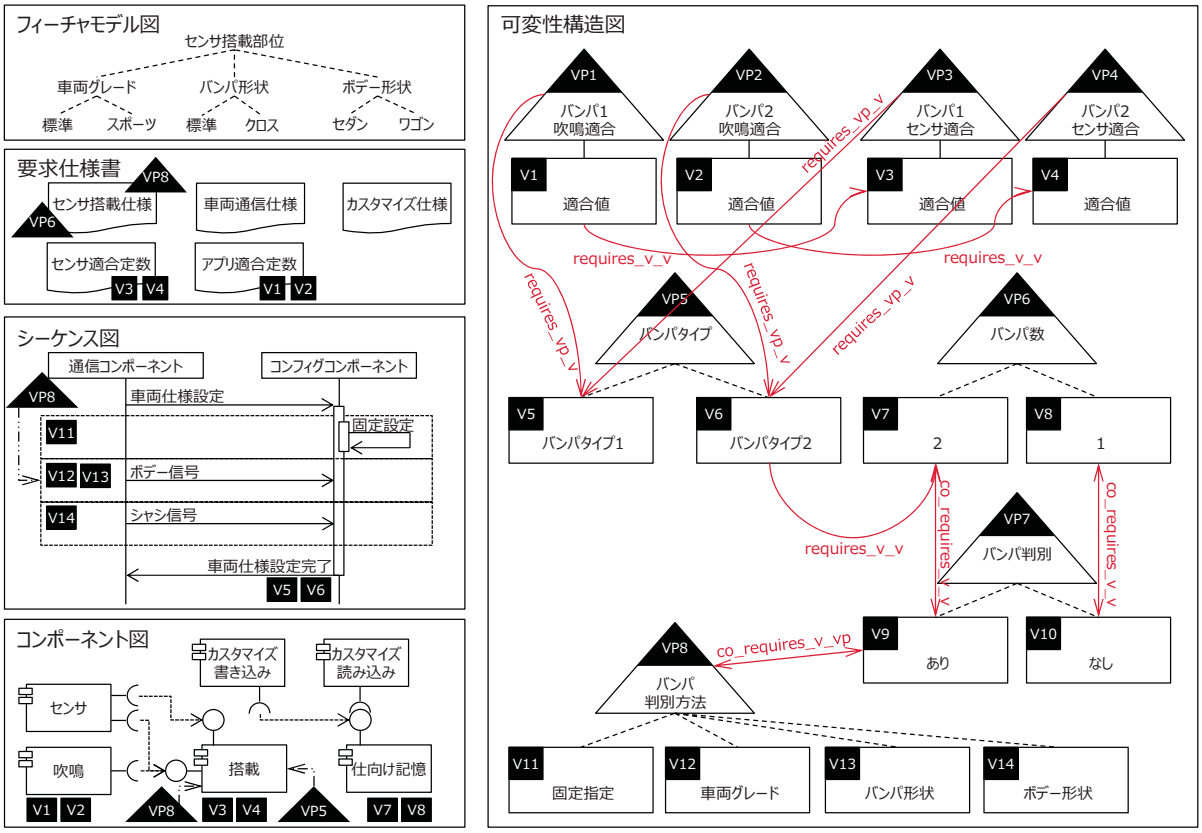

図 5 拡張 OVM による可変性の構造記述例

Figure 5 Sample of Variability Structure Representation by Extended OVM

4.2 MPLE における可変性構造の分析方法

OVM 拡張モデルを用いて MPLE における可変性の構造を分析する方法を示す．MPLE における OVM 拡張モデルの生成方法を示し，開発順序制約の分析方法を示す．

4.2.1 MPLE における OVM 拡張モデルの生成

本項では，MPLE における OVM 拡張モデルを生成するスコープを定義する．スコープ定義は，MPLE の形態に基づく定義と，モデルの生成タイミングに基づく定義の 2 つの基準がある．これらの 2 つの基準に基づき，適切にスコープを定義する必要がある．

(1) MPLE の形態に基づくモデルのスコープ定義

OVM 拡張モデルは，依存関係にある可変点と変異体の集合をスコープとしている．しかし，MPLE の形態によっては，スコープをさらに分割することで，関心事を限定し，分析コストを低減できる．

MPLE では，次の 2 つの形態を取り得ることから，それぞれに応じたスコープの定義が必要となる．

(a) コア資産ベーススコープ定義

1) MPLE の形態：製品は特定のコア資産から開発するが，並行運用するコア資産が複数存在する[15].
2) モデルのスコープ：コア資産ごとに OVM 拡張モデルを生成する．コア資産同士は並行に運用されるが，依存関係は独立して扱えるためである．

(b) モノリシックスコープ定義

1) MPLE の形態：製品の構成コンポーネントが，コンポーネントごとにコア資産を個別に有する[22].
2) モデルのスコープ：すべてのコア資産をスコープに含める．可変性の依存はコンポーネントを横断して現れる．スコープを分割すると分析すべき依存関係を見落とすリスクが生じる．

(2) モデルの生成タイミングに基づくモデルのスコープ定義

OVM 拡張モデルを生成するタイミングによっても，モデルを生成するスコープを分割することで，関心事を限定し，分析コストを低減できる．

(a) ドメイン開発時

1) モデルの生成タイミング：ドメイン開発時に可変点を設計するときにモデルを生成する．
2) モデルのスコープ：設計対象となるすべての可変点，変異体をスコープとしてモデルを生成する．

(b) アプリケーション開発時

1) モデルの生成タイミング：アプリケーション開発時に可変点を設計するときにモデルを生成する．
2) モデルのスコープ：要求分析や設計工程で分析対象となる可変点，変異体，コア資産にスコープを限定する．アプリケーション開発では開発期間の制約がドメイン開発時よりも強く，すべての可変点，変異体を対象にモデル生成することは，投資対効果として低いため，今回の開発で変更を加える可変点を優先してモデルを生成する．但し，依存関係の見落としが発生し得るような，コンポーネント単位などでのスコープ限定は避ける．

4.2.2 可変性の依存関係による開発制約の分析

可変性の依存関係によって，アプリケーション開発における開発アイテムの分割と開発順序に制約が生じる．これらの制約を明確にすることで，合理的な開発計画が設計可能となる．これらの制約は，4.1.1, 4.1.2 で示した通り，OVM 拡張モデルを用いて分析が可能である．開発アイテムの分割への制約と分割後の開発順序制約を示し，分析において明確にすべき，依存元と依存先における可変点と変異体の組合せの作用と，Excludes 依存関係の扱いについて述べる．

(1) 開発アイテムの分割方法の分析

開発アイテムの分割方法は，4.1.1 で示した通り，OVM 拡張モデルにより，可変性の依存関係を明らかにすることで分析できる．依存関係を明らかにした後は，どの程度の開発量に開発アイテムを分割したいかによって任意の分割が可能である．

(2) 開発順序制約の分析

開発順序の制約も，4.1.1，4.1.2 で示した通り，OVM 拡張モデルにより，可変性の依存関係を明らかにすることで分析できる．但し，可変性としての依存関係と，開発順序制約としての依存関係は，依存の方向で一致しない場合がある．OVM 拡張モデルで定義した依存関係ごとの順序制約を図 6 に示す．

可変点同士に依存関係がない場合，Requires の依存関係の場合，Co-Requires の依存関係の場合は，可変性の依存関係と順序制約が一致する（図 6－(A)，(C)，(D)）．Excludes の依存関係の場合に順序制約が反転する．詳細な理由を(4)にて述べる．

(3) 依存元と依存先における可変点と変異体の組合せ

依存元と依存先の対象は，可変点と変異体で組合せが存在する．しかし，どのような組合せでも，可変点同士の依

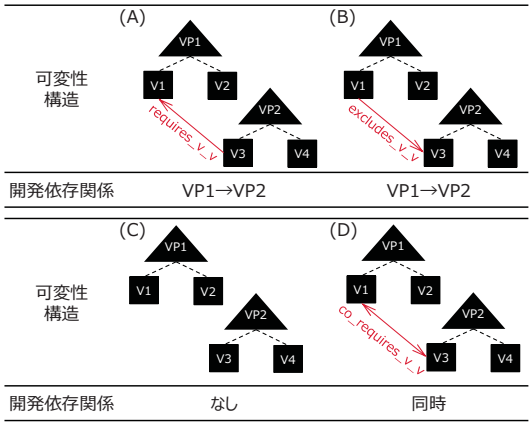

図 6 可変性の構造と開発順序制約

Figure 6 Variability Structure and Constraints of Development Sequence

存関係として扱える．例えば，図 6－(A)のように，VP2 の変異体 v3 から VP1 の変異体 v1 に Requires の依存があるとする．VP2 で変異体を決定するためには，VP1 において v1 が選択されるか否かが決まってからでないと，VP2 で v3 と v4 が選択可能であるのか，v4 のみ選択可能であるかが定まらない．

(4) Excludes 依存関係の扱い

Excludes の依存関係は，依存の方向を反転させた Requires と同じ扱いとする．例えば，図 6－(B)のように，VP1 の変異体 v1 から VP2 の変異体 v3 に Excludes の依存があるとする．VP2 で変異体を決定するためには，VP1 において v1 が選択されるか否かが決まってからでないと，VP2 で v3 と v4 が選択可能であるのか，v3 が排他されて v4 のみ選択可能であるかが定まらない．

図 5 で示した OVM 拡張モデルであれば，図 7 で示す可変点の順序集合に沿って，インクリメンタルに開発できる．

{VP6, VP7, VP8}→{VP5}→{VP3, VP4}→{VP1, VP2}

図 7 可変点の集合を分割した開発順序生成例

Figure 7 Example of Development Sequence Based on the Separation of Variation Points

5. 可変性の構造分析に基づくアジャイルアプリケーション開発方法

5.1 開発モデル

本稿では，図 8 に示すインクリメンタルなアジャイルアプリケーション開発モデルを提案する．本モデルは，OVM 拡張モデルによる可変性の構造分析に基づいて機能する．

本モデルは，アプリケーションバックログ設計，機能開発，可変点開発の 4 つのアクティビティで構成される．以下，各アクティビティの詳細を示す．

5.2 アプリケーションバックログ設計

4.2.2 による OVM 拡張モデルの分析で明らかになった分割方法を用いて，INVEST の方針に従ってアプリケーションを開発するバックログを設計する．バックログは，製品の開発に必要なストーリの集合であり，開発順序を規定した開発アイテムのリストである[3].

バックログは，OVM 拡張モデルの分析結果と開発規模の見積もりを比較して設計する．依存関係が複数の可変点を連絡する場合，開発分割方法は複数案できる．そのため，分割方法を変えながら，開発規模の見積もりを繰り返すことで，規定のタイムボックスに収まる分割方法を採択して，ストーリ定義してバックログ設計する．

本バックログに従って，機能開発と可変点開発のストーリをインクリメンタルに開発していく．

5.3 機能開発

製品固有の機能開発は，従来の ASD と同様に開発する．ストーリごとにテストまでの工程を一通り実施するため，

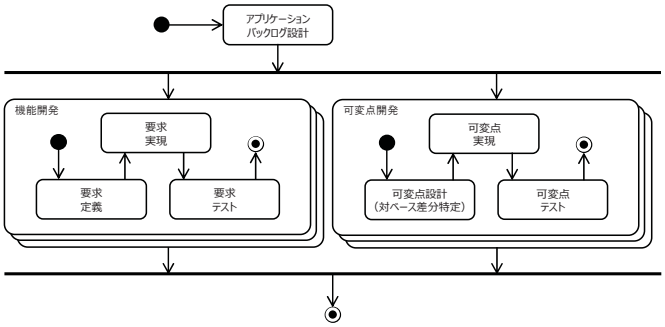

図 8 インクリメンタルなアプリケーション開発モデル

Figure 8 Incremental Application Engineering Model

テスト工程の工数とテスト環境の負荷を開発期間を通して平準化することができる．

尚，機能開発の中で新たな可変点が発生した場合は，OVM 拡張モデルを用いて，既存の可変点と新たな依存関係が生じないかを確認して設計する．

5.4 可変点開発

可変点開発では，変異体の実装を対象とする．機能開発と同様に，ストーリごとにテストまでの工程を一通り実施するため，テスト工程の工数とテスト環境の負荷を平準化することができる．また，可変点の集合の依存関係が明らかになっているため，その後のストーリ開発において，回帰テストを計画しなくてもよく，開発後期でのテスト工数の集中を軽減することができる．

6. 自動車システム APLE への適用

可変性の構造分析に基づくインクリメンタルなアジャイルアプリケーション開発方法を，自動車システムの APLE として，超音波センサシステム開発に適用した．筆者らの一人はチームリーダとして開発に参加している．

適用期間は 2016 年 12 月～2017 年 2 月の 3 か月間である．1 スプリントは 1 週間とし，9 スプリントで開発を終えている．開発人員は 3 名，テスト環境は 1 台である．

評価の比較対象として，2016 年 8 月～10 月で開発した類似規模の製品開発を 1 プロジェクト選択した．本プロジェクトはインクリメンタル化前の APLE として筆者らが提案した開発方法で開発している[8]．各プロジェクトで得られた実データで本方法の有効性を評価した．

7. 評価

7.1 評価方法

可変性の構造分析に基づくアジャイルアプリケーション開発方法を以下の観点で評価した．

(1) バリューストリームの速度向上

(2) テスト工数とテスト環境の負荷平準

これらの評価で有効性が確認できることで，自動車システムの APLE での有効性を評価する．

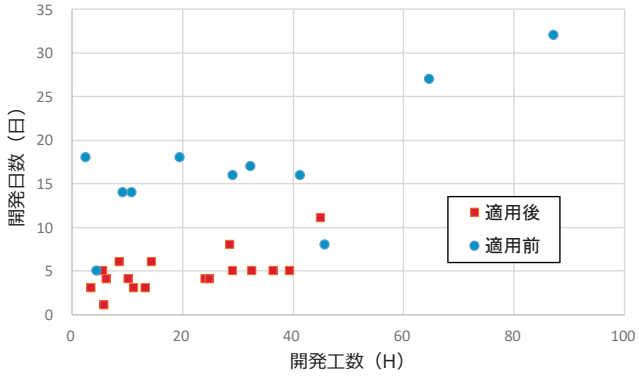

図 9 適用前後のバリューストリーム速度

Figure 9 Value Stream Speed Before and After Application

表 1 開発工数と日数の平均と標準偏差

Table 1 Average and Standard Deviation of Development Effort and Days

	適用前	適用後
アイテム数	11	17
開発工数:平均(H)	31.53	19.99
開発工数:標準偏差(H)	25.49	13.02
開発日数:平均(日)	16.82	4.82
開発日数:標準偏差(日)	7.21	2.15

7.2 バリューストリームの速度向上

バリューストリームの速度向上効果として，提案方法の適用前後における，開発アイテムの要求分析からテスト完了までの開発工数と開発日数を比較して評価した．

図 9 に，開発工数と開発日数を軸に取った，適用前後の開発アイテムごとの散布図を示す．また，表 1 に，同一のデータに対して，開発工数と開発日数それぞれで平均と標準偏差を求めた結果を示す．

適用前に対して適用後は，バリューストリームの速度を示す開発日数の平均は 16.8 日から 4.8 日と，3.5 倍に向上した．標準偏差は割合としては 42.9%から 44.6%と，ばらつきの程度に有意差はないが，絶対値としてのばらつきは 7.21 日から 2.15 日と 70%低減した．

適用後の開発アイテム数は増加した．アイテムごとの開発工数の平均は31.53Hから19.99Hと63%程度に縮小した．ばらつきとしての標準偏差は絶対値で25.49Hから13.02H，相対値で 80.8%から 65.1%に改善された．開発アイテムの開発規模が小さくなり，アイテムごとの開発工数，開発日数のばらつきが縮小し，バリューストリームの速度が 3.5 倍に向上する結果が得られた．

7.3 テスト工数とテスト環境の負荷平準

テスト工数とテスト環境の負荷平準効果として，テスト環境の使用率を計測し，テスト負荷のばらつきを評価した．

図 10 に提案方法を適用する前後でのテスト環境使用率を示す．適用前では，開発の後半（16 日目以降）でテストが開始され，終盤に向けて使用率が上がっている．適用後では，開発開始後 5 日目にはテストが開始され，開発終盤でやや高めになるものの，使用率は一定水準を保った．

表 2 にテスト環境使用率の平均値と最大値を示す．分母

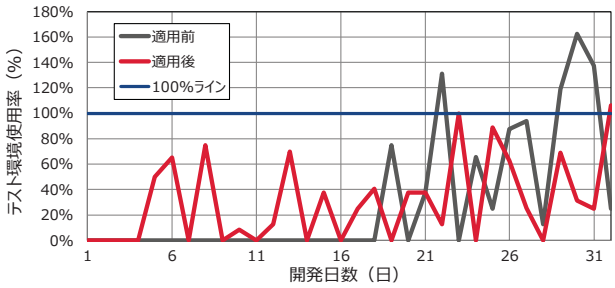

図 10 適用前後のテスト環境使用率

Figure 10 Test Environment Usage Rate Before and After Application

表 2 テスト環境使用率の平均と最大

Table 2 Average and Maximum of Test Environment Usage Rate

	適用前	適用後
テスト日数（日）	12	20
テスト工数（H）	77.75	78.32
平均（%）	80.99	48.95
最大（%）	162.50	106.25

はテスト環境の使用日数である．適用前では，テスト環境の使用日の平均値は約 80%であり，最大の使用率は約 160%である．適用後では，平均値は約 50%と 30%低減し，最大値も約 106%と，50%以上低減した．テスト工数は両プロジェクトで同等であり，テスト日数は適用後では約 1.67 倍となっていた．これらの結果から，テスト環境の使用率が平準化されたことが確認できた．

テスト工数には，報告書の作成工数など，テスト環境を使用する以外の工数も含まれる．しかし，テスト工数の多くは環境使用の工数であり，テスト環境使用率で近似できる．したがって，本評価結果から，提案方法によって，テスト工数の負荷が平準化されることが確認できた．

8. 考察

8.1 OVM 拡張モデルによる開発制約の明確化（Q1）

8.1.1 OVM 拡張モデルによる可変性の構造分析により可変点の開発制約は明らかにできたか？

明らかにできた．7.2 で示した通り，適用前と比較して開発アイテムの粒度を平均63%縮小して分割することができた．7.3 で間接的に示される通り，各開発アイテムは独立してテストを実施できており，Independent で Small かつ Testable なストーリ定義に成功している．

8.1.2 OVM 拡張モデルでしか開発制約は求められないか？

求められないことはない．例えば，フィーチャモデルでも realized-by の関係を定義し，可変点と可変点を実現する開発コンポーネントとの関係を分析する方法が提案されている[10]．しかし，OVM では共通性を捨象して可変点，変異体の依存関係のみを抽出して分析できることから，開発制約を求めるという関心事に集中できるメリットがある．

8.1.3 Co-Requires の関係は分析にどれくらい寄与するか？

モデルを描いて開発制約を分析する場合に効果的である．Co-Requires は Requires の双方向の関係を集約する．

Co-Requiresの関係が多いときにRequiresのみで表現すると，表現が煩雑になり分析に誤りが混入しやすくなる．

何らかの自動分析ツールを開発する場合，Co-Requires で同時開発の制約を明示することで，分析ルールを単純化して解析効率の向上が可能であると考えられる．

8.2 アプリケーション開発の負荷平準化と俊敏化（Q2）

8.2.1 提案モデルで負荷平準化と俊敏化は実現できたか？

実現できた．7.3 で示した通り，テスト工数とテスト環境の負荷が平準化できた．また，7.2 で示した通り，バリューストリームの速度を向上させ，開発が俊敏化できた．

8.2.2 提案モデルは効果を実現する必要条件であるか？

必要条件ではない．負荷平準化と俊敏化の効果は，提案モデルのベースとなっている，ASD におけるインクリメンタル開発によって実現されている．提案モデルでなくても，インクリメンタルな開発であれば効果が期待できる．

8.2.3 提案モデルが果たしている有用性はどこか？

インクリメンタルな開発を実現したときの効果を阻害リスクを軽減できる．可変性の構造が不明な場合，分割された開発アイテム同士が相互作用を起こすことで，開発後半で以前の開発アイテムにデグレードが生じたり，デグレードを検出するための回帰テストの工数が増加したり，効果を阻害するリスクが生じ得る．OVM 拡張モデルによる可変性の構造分析でこれらのリスクを軽減して，インクリメンタル開発への移行を容易にする．

8.3 自動車システムの APLE における有効性（Q3）

8.3.1 提案方法は自動車システムの APLE で有効か？

有効である．自動車システムの実開発に適用し，7.2，7.3 の評価で示した効果を得ることで，自動車システムの APLE において，提案方法が有効であることを示した．

8.3.2 提案方法は大規模開発や膨大な可変点にも有効か？

SPLE に基づいて可変点を適切に設計しようとしている開発であれば有効である．適切な設計とは，依存関係を持つべき可変点だけに依存制約が束縛されている設計を指す．

拡張 OVM モデルで可変点の依存関係をモデル化した場合を考える．製品全体の開発規模が大きくても，適切な設計に基づけば，依存が閉じたモデル図の規模は小さい．モデル図の単位でインクリメンタル開発のバックログを設計することで，大規模開発でも効果が期待できる．

大規模開発で提案方法を取り組むにあたり，適切な設計であるかを判定できる評価方法や，スケールアップするために複数のチームが分担開発するための具体的な方法については今後の課題として取り組む必要がある．

9. まとめ

自動車システムの MPLE において， OVM 拡張モデルを用いて可変性の構造を分析した結果に基づき，独立で開発量の小さな開発アイテムに分割しインクリメンタル開発するアジャイルアプリケーション開発方法を提案し，自動車システム開発の実プロジェクトに適用した．その結果，MPLE における可変性の複雑性に対応し，テスト工数とテスト環境の負荷平準化を果たし，開発の俊敏化を実現した．自動車システムの APLE として開発の俊敏性を向上できた．

参考文献

[1] M. Aoyama, An Extended Orthogonal Variability Model for Metadata-Driven Multitenant Cloud Services, Proc. of APSEC 2013, IEEE, Dec. 2013, pp. 339-346.
[2] S. Bühne, et al., Why is it not Sufficient to Model Requirements Variability with Feature Models?, Proc. of AuRE04, IEEE, Sep. 2004, pp.5-12.
[3] M. Cohn, Agile Estimating and Planning, Prentice Hall, 2005.
[4] S. Deelstra, et al., Product Derivation in Software Product Families, J. of Systems and Software, Vol. 74, No. 2, 2005, pp.173-194.
[5] J. Díaz, et al., Agile Product Line Engineering- A Systematic Literature Review, Software: Practice and Experience, Vol. 41, No. 8, Jul. 2011, pp. 921-941.
[6] F. R. Frantz, Quality-Aware Analysis in Product Line Engineering with the Orthogonal Variability Model, Software Quality J., Vol. 20, No. 3-4, Sep. 2012, pp. 519-565.
[7] G. K. Hanssen, et al., Process Fusion, J. of Systems and Software, Vol. 81, No. 6, Jun. 2008, pp. 843-854.
[8] 林 健吾, 他, プロダクトライン開発のための反復型プロセスモデルと管理方法の提案と適用評価, SES 2016 論文集, 情報処理学会, Aug. 2016, pp. 195-202.
[9] P. Hohl, et al., Searching for Common Ground, Proc. of ICSSP ‘17, ACM, Jul. 2017, pp. 70-79.
[10] M. Janota, et al., Formal Approach to Integrating Feature and Architecture Models, Proc. of FASE 2008, LNCS, Vol. 4961, Springer, 2008, pp. 31-45.
[11] S. Kato, et al., Variation Management for Software Product Lines with Cumulative Coverage of Feature Interactions, Proc. of SPLC 2011, IEEE, Aug. 2011, pp. 140-149.
[12] K. Lee, et al., Concepts and Guidelines of Feature Modeling for Product Line Software Engineering, Proc. of ICSR-7, Springer, 2002, pp. 62-77.
[13] D. Leffingwell, Agile Software Requirements, Addison-Wesley, 2011.
[14] H. M. Mærsk-Møller, Cardinality-Dependent Variability in Orthogonal Variability Models, VaMos’12, ACM, Jan. 2012, pp. 165-172.
[15] A. Metzger, et al., Software Product Line Engineering and Variability Management, Proc. of FOSE 2014 (ICSE 2014), ACM, May-Jun. 2014, pp. 70-84.
[16] M. A. Noor, et al., Agile Product Line Planning, J. of Systems and Software, Vol. 81, No. 6, 2008, pp. 868-882.
[17] O. Oliinyk, et al., Structuring Automotive Product Lines and Feature Models, Requirements Eng. J., Vol. 22, 2017, pp. 105-135.
[18] K. Pohl, et al., Software Product Line Engineering, Springer, 2005.
[19] B. Rumpe, et al., Agile Synchronization between a Software Product Line and its Products, Informatik 2015, LNI Vol. 246, Springer, Sep.-Oct. 2015, pp. 1687-1698.
[20] C. Tischer, et al., Bosch Gasoline Systems, F. van der Linden, et al. (Eds.), Software Product Line in Action, Springer, 2007, pp. 133-148.
[21] M. Steger, et al, Introducing PLA at Bosch Gasoline Systems: Experiences and Practices, SPLC 2004, LNCS Vol. 3154, Springer, Aug.-Sep. 2004, pp. 34-50.
[22] B. Tekinerdogan, et al., Supporting Incremental Product Development using Multiple Product Line Architecture, Int’l J. of Knowledge and Systems Science, Vol. 5, No. 4, Oct.-Dec. 2014, pp. 1-16.
[23] L. Wozniak, et al., How Automotive Engineering is Taking Product Line Engineering to the Extreme, SPLC 2015, ACM, Jul. 2017, pp. 327-336.