マルチ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

林 健吾[†1]　青山 幹雄[†2]　古畑 慶次[†3]

概要：本稿では，マルチ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を提案する．プロダクトライン開発では，製品系列の多様性に対処すべく，可変性を備えたコア資産を構築するドメイン開発と，個別製品を導出するアプリケーション開発に問題領域を分割している．しかし，多重化したアプリケーション開発の多様性に対処した開発モデルと管理方法はなく，納期リスクが生じ得る．提案するプロセスモデルは，アプリケーション開発の反復性に着目して再利用可能なプロセス資産を構築する．プロセス資産を再利用してポートフォリオ計画と製品開発計画を立案し，開発をコントロールする方法を提案する．提案方法を自動車ソフトウェアのマルチプロダクトライン開発に適用し，マルチプロダクトラインの複雑性軽減効果と，管理性向上効果により提案方法の有効性を確認した．

An Iterative Process Model and Its Management Method for Multiple Product Line Automotive Software Development

KENGO HAYASHI[†1]　MIKIO AOYAMA[†2]　KEIJI KOBATA[†3]

1. はじめに

自動車ソフトウェア開発において，多様な要求を効率的に対応するために SPLE（ソフトウェアプロダクトライン開発）が導入されている[12][23]．SPLE は同種同系列のソフトウェア製品の多様性を吸収し，製品系列を量産するためのアプローチである[6][14][19]．

SPLE 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多様性に対処する．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導出する．SPLE に向けて，著者らはコア製品開発チーム（以下、コアチーム）と派生製品開発チーム（以下、派生チーム）で開発を分担している（図 1）．

コアチームでは，共通性と可変性を分析しながらコア資産をコアプロダクトラインとして開発し，いくつかのコア製品をリリースする．派生チームではコア資産からアプリケーションを導出して派生製品を数多くリリースする．コア資産が世代交代するときには，一部のコア資産を派生チームに委譲し，派生プロダクトラインとして維持する．

派生チームにおける製品開発は，製品当たりの開発規模は小さいが，並行開発する製品数が多い．並行開発数が増加すると，複数のプロダクトラインを並行して開発するよう進化していく[1]．本稿では，これをマルチプロダクトラインと呼ぶ．マルチプロダクトラインでは通常の SPLE よりも開発管理が複雑化する．しかし，マルチプロダクトラインの多様性を効果的に吸収して開発，管理する方法は確立されていない．特に多様な製品開発の生産性と開発量の見積りは困難で，納期リスクが発生する．

本稿では, SPLE の本質的特性である反復性に着目して，マルチ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を提案する．本方法では，アプリケーション開発において再利用可能なプロセス資産を定義し，プロセス資産を再利用しながらポートフォリオ計画とプロダクト計画を立案し，開発をコントロールする．本開発方法を自動車ソフトウェアのマルチプロダクトライン開発に適用し，マルチプロダクトラインの複雑性軽減効果と管理性向上効果により提案方法の有効性を確認した．

2. 関連研究

2.1 SPLE（ソフトウェアプロダクトライン開発）

SPLE は，同種同系列のソフトウェア製品の多様性を吸収し，製品系列を量産するためのアプローチである[6][8][19]．SPLE 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多様性に対処する．ドメイン開発は，製品系列における共通性と可変性を分析してコア資産

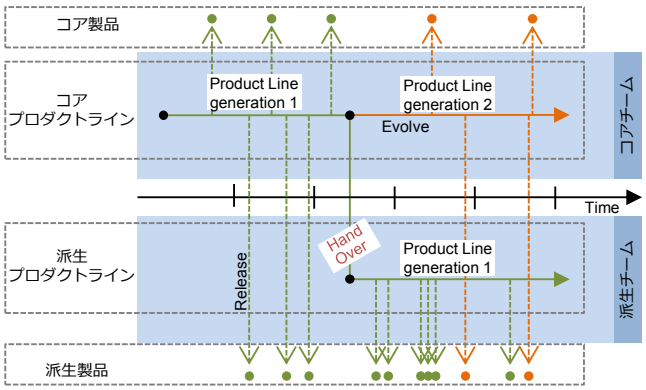

図 1 コア製品開発と派生製品開発

Fig. 1 Core Product Development and Derivative Product Development

†1 (株)デンソー
DENSO CORPORATION
†2 南山大学
Nanzan University
†3 (株)デンソー技研センター
DENSO E & TS TRAINING CENTER CORPORATION

を構築する．アプリケーション開発はコア資産から個別製品を導出する．

コア資産にはアプリケーションを構築するプロセスを蓄積したプロセス資産も含まれる[11]．しかし，アプリケーション開発における多様性に着目した，効率的なプロセス資産の運用は確立されていない．

また，SPLE におけるポートフォリオ管理の研究は少なく[20][21]，個々のアプリケーション開発の管理を連携する方法も未確立である．

2.2 ソフトウェア開発における見積りと実績の関係性

ソフトウェア開発の成功には，正確性の高い開発計画が必要である．しかし，任意の開発が完了する見積もりと実績は確率的であり，不確実性が高い[4][5][16]．ソフトウェア開発における計画とコントロール，過去の実績の再利用には，見積りと実績の確率分布を考慮すべきである．

2.3 生産管理における工程能力の測定とコントロール

生産管理では，工程能力を定義し，それを測定してコントロールしている[10][13][24]．工程能力とは，工程が一定期間，統計的管理状態であるときの能力をいう．この能力の量に着目した特性を工程許容量と呼ぶ．ソフトウェアの開発量を生産管理と同様に測定しコントロールする場合，“一定期間”，“統計的管理状態”を維持できるよう測定対象と測定範囲を定めることが重要となる．

2.4 Agile 開発

XP，Scrum に代表される Agile 開発では，開発フィーチャに着目してストーリと名付け，ストーリの開発規模を相対的に見積もったストーリポイントで管理する方法を採用している[2][15][22]．また，ストーリをタイムボックスで反復開発して生産性をコントロールし，計測した生産性を計画にフィードバックする[5][14]．測定された生産性を元に，複数プロジェクトを統括してポートフォリオ管理する方法も提案されている[14]．しかし，短期間で複数の開発が並行する場合の有効策としては提案されていない．

2.5 APLE（Agile プロダクトライン開発）

APLE は SPLE と Agile 開発を様々な方法で統合する研究分野である[7][9][18]．これらのアプローチは，2 つの方法論の利点の協調を目指す．例えば，製品進化の各段階において 2 つの開発方法を選択的に切り替える方法が提案されている[9]．また，SPLE のドメイン開発において顧客を巻き込む方法も提案されている[18]．しかし，SPLE のアプリケーション開発において，マルチプロダクトラインの開発を統括管理する方法としての組合せは提案されていない．

3. プロセスの資産化

本稿のアプローチは，SPLE のアプリケーション開発におけるプロセス資産の反復性に着目する．本章では，プロセスが反復性を備える理由と，プロセス資産の概念を示す．

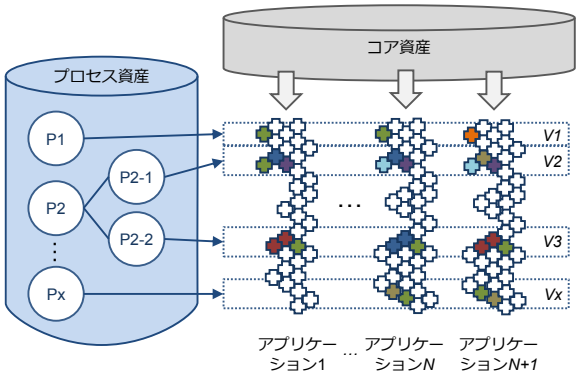

図 2 反復性のあるプロセス資産

Fig. 2 Iterative Process Asset

3.1 プロセスの反復性

コア資産を基にしたアプリケーション開発は複数回実行される．アプリケーション開発の前提となるドメイン開発では，製品系列に対する可変性の分析に基づいて，可変点を特定する．コア資産は，変異体を部分的に実装して開発される[19]．アプリケーション開発では，コア資産を再利用して，未実装の変異体を補完するか，各可変点の変異体を固定して個々のアプリケーションを導出する．各アプリケーション導出では，同一の可変点を対象に同一のプロセスを実行する．したがって，アプリケーションごとの可変点を実装するプロセスは反復性を備える．

3.2 プロセス資産の概念

図 2 にプロセス資産の概念を示す．SPLE の可変性を分類し，類似の可変性に対してはプロセスが反復可能であることに着目している．この結果，プロセス資産として蓄積するために，以下 3 つのプロセスパターンを特定した．

(1) プロセスパターン 1：アプリケーション導出時に，反復して同一の変異体（V1）を固定するプロセス（P1）．
(2) プロセスパターン 2：対象とする可変点や変異体は異なる（V2，V3）が，分析対象や試験方法が異なるだけで，類似の作業内容や作業規模であるプロセス群（P2，P2-1，P2-2）．
(3) プロセスパターン 3：コア資産が開発ライフサイクルの中で成長することで新たな可変点（Vx）が生じる．これに対応して，アプリケーション開発の反復の中で，新たに実行されるプロセス（Px）．

これらのプロセスは反復性を備えるため，プロセス資産として蓄積することで再利用可能となる．

4. 反復型プロダクトライン開発モデル

4.1 開発モデル

図 3 に提案する反復型プロダクトライン開発モデルを示す．本開発モデルは，SPLE におけるアプリケーション開発を対象とする．アプリケーション開発は，プロダクト開発とプロダクトラインポートフォリオ管理から構成される．

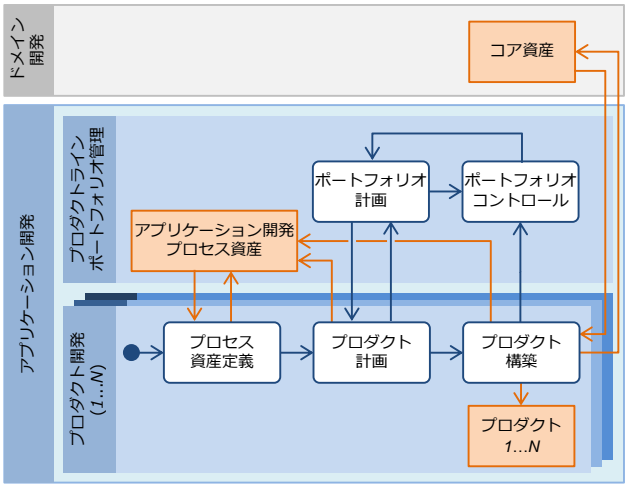

図 3 反復型プロダクトライン開発モデル

Fig. 3 Iterative Product Line Development Model

プロダクト開発では個々のアプリケーションを導出する．プロセス資産定義，プロダクト計画，プロダクト構築の 3 つのアクティビティから構成される．ここで生成されるプロダクト 1…N は，SPLE におけるアプリケーション 1…N と同義である．開発領域としてのアプリケーション開発と，個々のアプリケーション導出であるアプリケーション開発を区別するため，後者をプロダクト開発と呼ぶ．

プロダクトラインポートフォリオ管理は，プロダクト開発を統括する．ポートフォリオ計画，ポートフォリオコントロールの 2 つのアクティビティから構成される．

本開発モデルの各アクティビティを以下に示す．

4.2 プロセス資産の定義

プロセス資産のモデルを図 4 に示す．プロセス資産は，1 つ以上のプロセスユニット，或いはプロセスユニット群で構成される．本節では，プロセスユニットとプロセスユニット群の定義とプロセス資産定義アクティビティを示す．

4.2.1 プロセスユニット

プロセスユニットはプロダクト構築で反復実行されるプロセスである．プロセスユニットは，ユニット名，プロセス定義，プロセス成果物，見積り規模から構成される．

見積り規模は，プロセスユニットの開発規模である．Agile 開発のストーリポイント[2][15][22]と同様にポイントで表現する．プロダクト計画時に，過去の見積り結果として参照する．プロセスユニットを登録するときには実績がないため，プロダクト計画時に他のプロセスユニットと相対的に見積りを比較して決定する．

4.2.2 プロセスユニット群

プロセスユニット群は，作業内容や作業規模が類似したプロセスユニットへの関連である．プロセスユニット群は，分類名と 1 つ以上のプロセスユニットから構成される．

4.2.3 プロセス資産定義アクティビティ

プロセス資産定義アクティビティは，プロダクト開発の計画と同時に実行する．プロダクト開発の計画において，開発プロセスを設計するために，蓄積したプロセス資産を参照しながら，プロセス資産を登録更新する．表 1 に，プロセスユニットの例を示す．

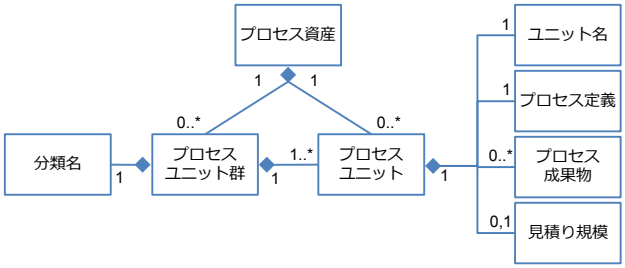

図 4 プロセス資産モデル

Fig. 4 Process Asset Model

(1) プロセスユニットの新規登録

プロセスユニット，あるいはプロセスユニット群を新規登録する．プロセスユニットは可変点に着目して作成する．プロダクト開発で管理しやすい粒度であることが望ましく，要求開発（#1），設計（#3，#4，#5，#6），実装，試験などのプロセス単位に分割する．

(2) プロセスユニット群の関連付け

可変点に変異体を追加するプロセスユニットを定義するとき，同様の作業手順や作業規模となるプロセスユニットが存在する場合は，プロセスユニット群を定義して関連付ける（#4，#5，#6）．

(3) 反復未定プロセスユニットの登録

特定のプロダクト開発のための調査など，プロダクト開発ごとに反復される予定のないプロセスも設計される．これらのプロセスは，今後反復される可能性は低くても，その他のプロセスユニットの規模見積りや，将来反復された場合の参考作業実績として参照できるよう，プロセスユニットとして登録しておく（#2）．

4.3 プロダクト計画

プロダクト計画では，プロダクトバックログの作成とプロセスユニットの規模見積り，開発スケジュールの策定を実行する．図 5 にプロダクト計画のモデルを示す．

4.3.1 プロダクトバックログの作成

プロダクトバックログの作成では，プロダクト構築で実

表 1 プロセスユニット例

Table 1 Process Unit Sample

#	分類名	ユニット名	概要	反復性
1	–	ノード間通信分析	通信仕様を走査して変異体の固定点を分析する	あり
2	–	特殊環境負荷試験	定常使用ではないプロダクトの特殊環境向けに負荷試験する	なし
3	–	コンフィグレーション	可変分析済みの可変点リストの変異体固定を設定する	あり
4	通信判別系変異体追加（設計）	エンジン種類追加（設計）	通信 X のデータからエンジン種類の変異体を追加する	あり
5		T／M 種類追加（設計）	通信 Y のデータからトランスミッション種類の変異体を追加する	あり
6		駆動方式追加（設計）	通信 Z のデータから駆動方式の変異体を追加する	あり

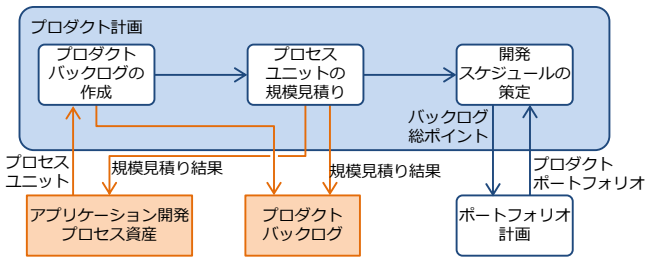

図 5 プロダクト計画のモデル

Fig. 5 Product Planning Model

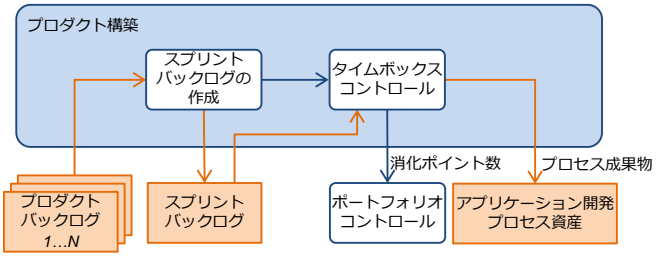

図 6 プロダクト構築のモデル

Fig. 6 Product Build Model

行するプロダクトバックログを作成する．プロダクトバックログは，Agile 開発のプロダクトバックログ[14][15][22]と同様に，開発アイテムのリストである．本開発モデルでは，ストーリではなくプロセスユニットを用いてプロダクトバックログを作成する．

4.3.2 プロセスユニットの規模見積り

プロセスユニットの規模見積りでは，プロセスユニットを反復実行するときに記録された見積り規模を採用する．初回登録時には，見積り規模が記録されていないため，他の記録済みのプロセスユニットと相対的に規模を見積もる．プロダクト開発自体が初めての場合は，Agile 開発と同様にプランニングポーカー[5][22]などで規模を見積もる．

4.3.3 開発スケジュールの策定

プロダクトラインポートフォリオ計画では，当該プロダクトのプロダクトバックログの総ポイント数を基に，どのタイムボックスで何ポイントを当該プロダクト開発に割り当てるかを定める．プロダクト計画では，割り当てられたポートフォリオに従って，スケジュールを策定する．

4.4 プロダクト構築

プロダクト構築では，スプリントバックログの作成とタイムボックスコントロールを実行する．図 6 にプロダクト構築のモデルを示す．

4.4.1 スプリントバックログの作成

個々のタイムボックスは Agile 開発と同様にスプリントと呼ぶ[14][15][22]．プロダクト構築ではスプリントで取り組む開発アイテムをスプリントバックログとして作成する．

スプリントバックログは，異なる複数のプロダクトバックログを集約して作成する．ポートフォリオ計画において，今回スプリントでどのプロダクトバックログからどれだけのポイントを割り当てるかが定められている．定められたプロダクトバックログから，定められたポイントを満たすだけ，プロセスユニットを引き出して集約する．プロダクトの優先度もポートフォリオ計画に従う．

4.4.2 タイムボックスコントロール

タイムボックスコントロールでは，スプリントバックログに登録されたプロセスユニットを順に実行する．タイムボックスのコントロールは，Scrum の方法を適用する[5][22]．タイムボックスコントロールの目的は，プロセスユニットの見積り規模に対する作業実績のばらつきの吸収である．スプリントバックログを生成するときに，開発チームの稼働時間上限まで使わずバッファを設ける．バッファを設けることで，各スプリントで一定した量のプロセスユニットを完了するようコントロールする（図 7）．

タイムボックスで定められた期間（1〜3 週間）の開発を完了すると，タイムボックスで完了したプロセスユニットの見積り規模の総和が定まる．見積り規模の総和はポートフォリオコントロールの入力とする．

4.5 プロダクトラインポートフォリオ管理

プロダクトポートフォリオ管理は，すべてのプロダクト開発を統括して予算や資源を管理する．ソフトウェア開発の費用のほとんどは，人的資源で費やされることから，人的資源の管理を中心に述べる．ポートフォリオ管理をする上で，工程許容量の概念を導入する．以下に，工程許容量の定義と，ポートフォリオ計画，ポートフォリオコントロールの各アクティビティについて示す．

4.5.1 工程許容量の定義

工程許容量は，“一定期間”，“統計的管理状態”を維持したときの量で定められる[10][13]．統計的管理状態とは，測定対象が何らかのばらつきがある属性を備えており，数多くの対象の属性を測定することで，ばらつきの傾向を把握し，ばらつきに応じて管理されている状態を指す．本開発モデルでは，タイムボックスにより“一定期間”を定める．プロセスユニットの見積り規模が対象であり，規模単位当たりの作業実績がばらつく属性である．作業実績がばらつくことで，一定期間内に完了するプロセスユニットの規模の総和もばらつく．タイムボックスコントロールは，このばらつきを考慮して，一定量のプロセスユニットを完了するようコントロールする．こうして得られた見積り規模の消化ポイント数を工程許容量と定義し，スプリントごとに開発チームの生産性として管理に利用する．

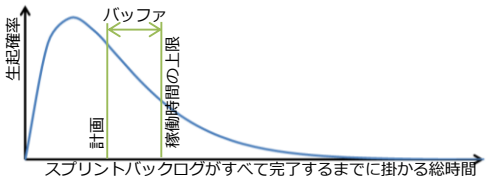

図 7 バッファによるばらつきの吸収

Fig. 7 Absorption of Variability due to Buffer

4.5.2 ポートフォリオ計画

ポートフォリオ計画では，いつ，何を，どれだけ開発するかを計画する．各プロダクト開発の規模見積りとマイルストーン，ポートフォリオコントロールから入力される工程許容量を入力情報とする．これらの入力を基に，この先のどのスプリントで，どのプロダクトのプロセスユニットを，何ポイント分実行するかを定める．

ポートフォリオ計画において，工程許容量を超えるスプリントを検出した場合，人的資源の追加投入，プロダクト計画の開発量低減やマイルストーンの交渉を検討する．

ポートフォリオ計画では，プロダクトの開発が要請された時点で，プロダクト開発を仮見積りする．プロダクト計画で詳細に見積もる前に，過去のプロダクト開発のポイント総和を利用して，中長期の計画を立てておく．

4.5.3 ポートフォリオコントロール

ポートフォリオコントロールでは，プロダクト開発におけるタイムボックスコントロールで得られた消化ポイントから工程許容量を算出し，プロダクト開発を統括した進捗を監視し，改善活動を立案・実行する．

消化ポイントと工程許容量を併せて，ポートフォリオ計画に沿っているかの進捗を監視し，この先のポートフォリオ計画を見直す．工程許容量の推移としての変化が大きい場合，タイムボックスで実行できるプロセスユニットの量が変化しているため，計画の見直しが必要となる．

ポートフォリオコントロールでは，工程許容量，すなわち生産性が高く安定することを目指して，開発チームのプロセスを改善するための施策を検討する．目的は，生産性を高めることと，生産性を安定させて開発計画の見通しの正確性を高めることである．

5. 自動車ソフトウェア車両展開開発への適用

マルチプロダクトライン開発のための反復型プロセスモデルを，自動車ソフトウェアの車両展開開発に適用した．車両展開開発は，コア製品を派生開発して多様な車両に展開していく開発である．著者の一人は車両展開開発の開発チームリーダとして開発に参加している．

5.1 開発の背景と課題

自動車ソフトウェアの開発組織とプロダクトライン，製品リリースの流れを図 8 に示す．同一製品系列を 2 つの組織で分担開発している．SPLE におけるドメイン開発を主として担うコア製品開発チームと，アプリケーション開発を担う車両展開開発チームである．

コア製品開発チームは，コア資産（コアプロダクトライン）を開発し，可変点を決定する．アプリケーションとしていくつかのコア製品（Product A，B，E）を開発する．

車両展開開発チームは，コア資産から派生製品（Product C，D）を導出する．アーキテクチャの変更を伴う大きな変更があった場合，製品系列を保守するために，コア製品開発チームから車両展開開発チームにコア資産を委譲する．車両展開開発チームは，委譲されたコア資産（派生プロダクトライン）を進化させて，派生製品（Product F，H，I）を継続してリリースする．

コア製品開発チームは，次世代のコア資産を進化させてコア製品（Product G）をリリースする．車両開発フェーズによっては，プロトタイプ開発が必要となる．コア資産が安定していない場合，プロトタイプ（Product H V1，H V2）は車両展開開発チームで開発する．再利用可能な開発資産は，コア製品開発チームによってコア資産へ取り込まれる．

このような開発形態においては，多くの派生製品をマルチプロダクトラインを利用して車両展開開発チームで並行的に開発，管理する方法が課題となる．

5.2 適用期間と対象プロジェクト数

車両展開開発チームにおいて，本稿で提案したプロセスモデルと管理方法を適用した．

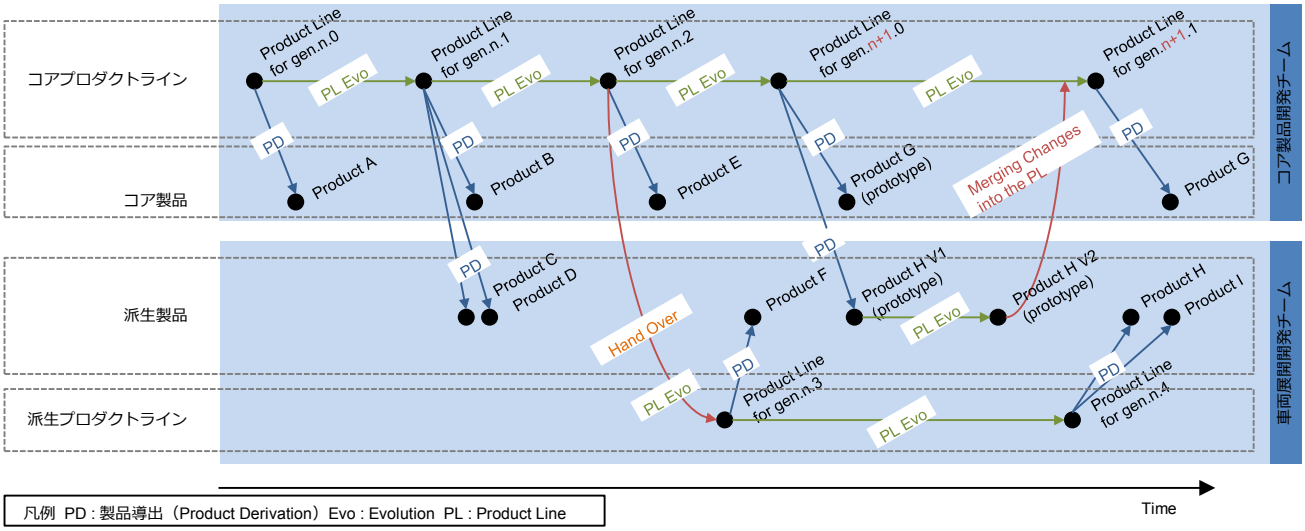

図 8 自動車ソフトウェア開発の開発組織と製品リリース

Fig. 8 Development Organization of Automotive Software Development and Product Release

適用期間は，2015 年 7 月～2016 年 4 月の 10 か月間である．1 スプリントは 2 週間としており，期間中に 22 スプリントを実行している．チームメンバは 3 スプリント目以降で増員しており，その後の人員の変動はない．期間中に 11 のプロジェクトに取り組んだ．これらのプロジェクトで適用して得られたデータで本方法の有効性を評価した．

6. 評価

6.1 評価方法

提案する開発モデルの効果を以下観点で評価した．

(1) プロセスユニットの反復性
(2) マルチプロダクトラインの複雑性軽減
(3) プロセス資産の規模見積りによる管理性

評価データは，開発リーダとして開発に参加した著者の一人が，車両開発の実プロジェクトで収集した結果に基づく．(1)は本開発モデル提案の前提条件の検証であり，(2), (3)は本開発モデルによる効果の検証に位置づく．

6.2 プロセスユニットの反復性の評価

試行した 11 のプロジェクトで実行されたプロセスユニットについて，プロセスユニット群として反復した回数の統計を図 9 に示す．本プロジェクトでは，プロセスユニットを要求開発，設計，実装，試験に分割しているため，分類ごとに集計している．

プロセス資産には，合計で 268 のプロセスユニットが登録された．その内，プロセスユニット群（単体のプロセスユニットを含む）は 15 種あった．15 種のプロセスユニットは，実行されたプロセスユニット全体の 84.7%を占めている．要求開発は，プロダクトを対象に実行されるため，プロジェクトの数が最大の反復回数となっている（A）．設計，試験は変異体を対象に実行され，かつ対象が異なっても類似のプロセスユニット群を反復することが多いため，プロジェクト数を超えて反復されている（E，F，K）．実装は変異体の固定で反復されるため，設計の反復回数よりは少ないが，同様にプロジェクト数を超えてプロセスユニット群が反復される結果が得られた（J）．

15.3%のプロセスユニットは，反復を伴わない一過性の調査や，プロトタイプ製品リリースのために派生プロダクトラインを進化させるフィーチャ開発であった．

車両展開開発のプロダクトライン開発において，プロセスユニットに反復性があることが明らかとなった．

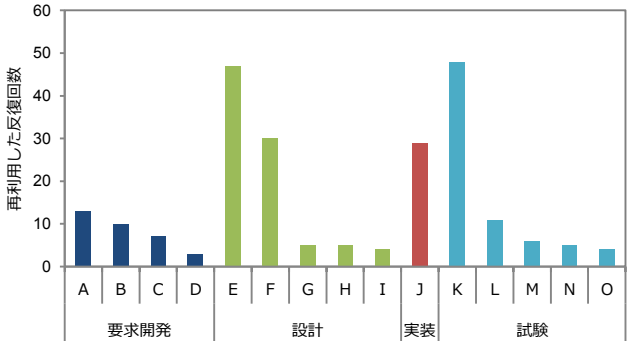

図 9 プロセスユニットの反復回数の統計

Fig. 9 Statistics of the Number of Iterations of the Process Unit

6.3 マルチプロダクトラインの複雑性軽減の評価

マルチプロダクトライン開発において複雑性が軽減され，多様性に対処できるかを評価した．図 10 に，プロダクトユニットの規模総和について，プロダクト計画時の見積りと実績の関係を示す．11 の試行プロジェクトの内，立ち上げ時の 2 プロジェクトと，フィーチャ開発を含む 2 プロジェクトを除く，7 プロジェクトを評価対象とした．

見積りと実績を対応させた結果を線形近似すると，相関係数が 0.8 を超えており，線形の強い相関が確認できた．開発規模が大きくなっても，規模あたりの開発コストは一定であった．マルチプロダクトライン開発で開発が並行しても，コスト変動が抑えられ，多様性に対する複雑性が軽減されているといえる．

6.4 マルチプロダクトライン開発管理性の評価

マルチプロダクトライン開発の管理に本管理方法が有効であるかを確認するため，開発の安定性と見積りの正確性を評価した．

6.4.1 開発の安定性評価

開発管理の容易性の指標として開発量の予測可能性がある．予測可能性を評価する代替特性として，開発の安定性を次のように定義して評価する．

提案した開発モデルでは，タイムボックスによって開発をコントロールする．各スプリントで消化されるプロセスユニットの見積り規模の総ポイント数の変化が一定であれば，工程許容量の変化が一定となる．工程許容量の変動が少ないことを，開発が安定している状態であるとする．

スプリントごとの消化ポイントの推移を図 12 に示す．赤線は，7 スプリントごとの移動平均であり，工程許容量を示す．工程許容量が計算できる第 7 スプリント以降において，工程許容量の変化率が 20%以内である場合を，変動が少なく開発が安定しているとする．図 13 に工程許容量の変化率の推移を示す．第 8～第 22 スプリントの 15 スプリントすべてが変化率 20%に収まっていた．その内，11 のスプリントは変化率 10%に収まっていた．変化率が 10%を上回

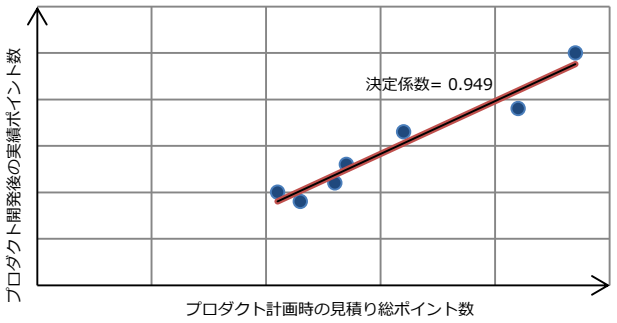

図 10 プロダクト計画の見積りと実績の関係

Fig. 10 Relationship between Estimates and Results of Product Plan

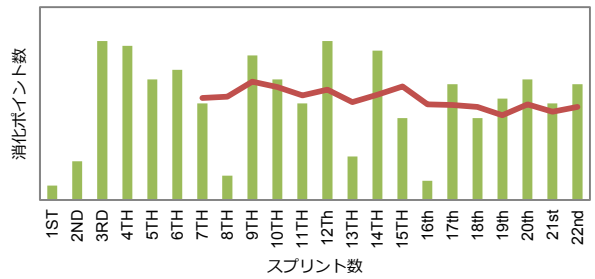

図 12 スプリントごとの消化ポイントの推移

Fig. 12 Trends of the Digestive Point of Each Sprint

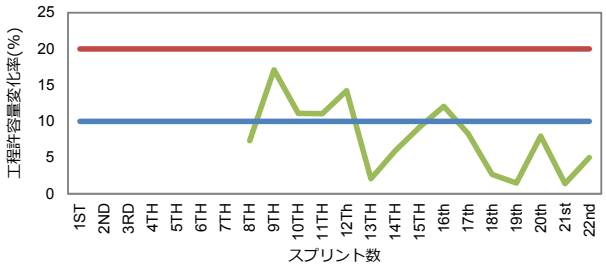

図 13 工程許容量の変化率の推移

Fig. 13 Trends of the Rate of Change of Process Capacity

る 4 スプリントの要因は以下である.

(1) 長期連休で稼働日が少数のスプリントを含む [1 回]
(2) フィーチャ開発など反復性のないプロセスユニットが多いスプリントを含む [2 回]
(3) 計画外プロセスが発生したスプリントを含む [1 回]

これらの要因を除くことで，開発の反復性が高くなると開発の安定性が高くなることが明らかになった.

6.4.2 見積りの正確性の評価

開発管理の容易性の指標として開発量の予測可能性がある．開発量の予測可能性を計画時の開発規模の見積りと実績の比較として評価した.

ポートフォリオ計画，プロダクト計画での規模見積りと，プロジェクト完了時の実績の比較を図 11 に示す．対象は 6.3 と同様であるが，開発中に顧客要因で要求が変化したプロジェクトを除外し，6 プロジェクトで評価した.

反復を重ねるごとに，プロジェクトの実績規模が減少した．これは，プロダクト開発が反復されることで，開発への経験不足で実施したムダな作業が省かれていったことと，開発要員が作業に習熟したことで，作業が効率化されプロセス規模が縮小されたことによる.

尚，ポートフォリオ計画とプロダクト計画では，プロダクト計画の方が正確性が高い．誤差率が増加したプロジェクトがあるが，その後は減少した．ポートフォリオ計画ではおおむね誤差率が 30%以下，プロダクト計画では誤差率が 15%以下であった．見積り誤差は初期のプロダクト定義において 60%，承認されたプロダクト定義において 25%というデータ[5]と比較して，高い正確性が得られた.

7. 考察

7.1 プロセスユニットの反復性

SPLE では，ドメイン開発で適切に可変点が分析されていれば，アプリケーション開発での，可変点やフィーチャの追加開発の発生は少ない．プロダクト構築において反復したプロセスユニットは，主に変異体の固定，もしくは変異体の追加開発であった．変異体の追加開発は，可変点が異なっても手順や規模の差異は小さかった．かつ，あるプロダクト構築で変異体が追加されると，その後のプロダクト構築では変異体を追加する割合が減り，変異体を固定するプロセスユニットの割合が増えた．そのため，設計，試験の両プロセスにおいて，反復されるプロセスユニット群が数多く作成できたと考えられる．初期のプロダクト開発でプロセスユニットはパターン化され，反復を重ねることで定着していく傾向がみられた.

7.2 マルチプロダクトラインの複雑性軽減

本開発プロセスモデルを適用した結果，開発規模によらず，見積りと実績は線形性を示した．従来の開発では，開発規模が大きくなると，指数関数的に開発コストは大きくなる[3]．試行において強い線形性を示した（図 10）のは，SPLE におけるアプリケーション開発の可変点同士の独立性が影響したと考えられる.

アプリケーション開発は，可変点に対する開発が主な活動となる．可変点同士の依存関係が少なくなるように適切に設計されていれば，可変点は互いに独立して開発できる．反対に，ドメイン開発での設計が不十分であると，複雑性は軽減されず，従来の開発と同様に開発コストは指数関数的に増大する可能性が高い.

また，本試行では，ポートフォリオ計画でタイムボックス単位の並行開発数を抑制していた．その結果，並行開発数増大による開発の複雑性が軽減されたと考えられる.

7.3 マルチプロダクトライン開発管理性

実際の開発へ適用した結果，本管理方法によるマルチプロダクトライン開発の管理性は良好な結果が得られた．開発の安定性と見積もりの正確性についてそれぞれ考察する.

7.3.1 開発の安定性

開発の安定性はタイムボックスコントロールと，SPLE

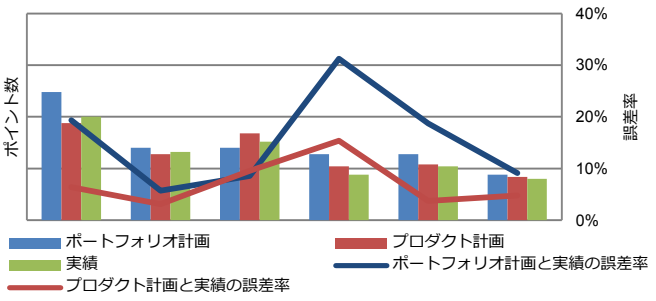

図 11 プロダクトの計画と実績の比較

Fig. 11 Comparison of the Product of the Planned and Actual

の反復性が大きく影響している．

プロセスユニットの見積りと実績では，実時間実績上のばらつきは大きい．タイムボックス化することで，各スプリントのバッファ消費率が変動するだけで，スプリントごとの消化ポイントは一定に保てた．その結果，規模実績においては安定性の向上が得られた．

しかし，フィーチャ開発など，反復性がなく経験もない開発では，見積り規模に応じた時間実績の乖離が大きい．その結果，一定のポイントを消化できないスプリントが生じた．反復性によって相対的な規模見積りが有効に働く．見積りと実績のばらつきが度抑えられることで，本管理方法が有効である．

したがって，反復性を備えない開発においては本開発モデルと管理方法は適さない．但し，Agile 開発のように，ストーリ開発で要求分析から検査までのプロセスを実行し，ストーリ単位でプロセスを反復することで学習効果を得る方法もある．反復性を見出して管理対象を選択することが，管理性の向上に重要であるといえる．

7.3.2 見積りの正確性

SPLE における見積りの正確性は，その反復性に依存している．SPLE の反復回数が増えると，開発実績が蓄積されて学習が進む．しかし，開発チームの成熟度が向上することで，作業の効率化やムダの削減によって開発規模の見積りに対して実績は小さくなっていった．実績の蓄積による学習に加えて，開発チームの成熟度を考慮することで，見積りの正確性の向上が期待できる．

7.4 APLE との比較

提案方法は，SPLE に Agile 開発の管理方法を組み合わせている．本アプローチは APLE の研究分野に属すると言える．APLE のその他の開発方法との比較を考察する．

Hansen は製品進化の各段階において 2 つの開発方法を選択的に切り替える方法を提案している[9]．本稿の方法では，開発初期の技術力が未熟であっても，開発方法を継続的に取り組むことで開発を安定させ，技術力を成熟させていくことが可能である．

Noor は，SPLE のドメイン開発において顧客を巻き込む方法を提案している[18]．本稿の方法では，対象を限定せず，より適用場面が多い問題領域を取り扱っている．

提案方法は，プロセス資産の利用とアプリケーション開発における開発管理方法に着目している．SPLE で取扱いの少ない問題領域を補完しているため，SPLE に基づく様々な開発方法との組み合わせも可能であると考えられる．

8. 今後の課題

本稿の適用では，スプリントの消化ポイントは開発を通して一定であった．しかし，チームの生産性は成熟度の向上と共に上昇している．見積り粒度の粗さによって生産性の変化が吸収されたと考えられる．適切な見積もり粒度の算出方法と，粒度変更による管理性への影響を評価する．

9. まとめ

車両展開開発のプロダクトライン開発において，反復性に着目することで，プロダクトラインの多様性に対処して管理性を向上させるため，反復型プロセスモデルと管理方法を提案し，自動車ソフトウェア開発の実プロジェクトに適用した．その結果，SPLE が反復性備えており，その反復性を利用してマルチプロダクトライン開発の複雑性を軽減でき，開発を安定させて管理性を向上させることが可能となった．

謝辞 本研究の推敲にあたり，多大な協力をいただいた開発チームのメンバに感謝する．

参考文献

1) M. Aoyama, Continuous and Discontinuous Software Evolution, Proc. of the IWPSE 2001, ACM, 2001, pp. 87-90.
2) K. Beck, Extreme Programming Explained, Addison-Wesley, 2000.
3) B. W. Boehm, et al., Software Cost Estimation with COCOMO II, Prentice Hall, 2000.
4) F. P. Brooks, Jr., The Mythical Man-Month, Addison-Wesley, 1995.
5) M. Cohn, Agile Estimating and Planning, Prentice Hall, 2005.
6) S. Deelstra, et al., Product Derivation in Software Product Families, J. of Systems and Software, Vol. 74, No. 2, 2005, pp.173-194.
7) J. Díaz, et al., Agile Product Line Engineering – A Systematic Literature Review, Software: Practice and Experience, Vol. 41, No. 8, 2011, pp. 921-941.
8) C. Elsner, et al., Variability in Time- Product Line Variability and Evolution Revisited, Proc. of VaMoS 2010, 2010, pp.131-137.
9) G. K. Hanssen, et al., Process Fusion: An Industrial Case Study on Agile Software Product Line Enineering, J. of Systems and Software, Vol. 81, No. 6, 2008, pp. 843-854.
10) 石川 馨, 新編品質管理入門 B 編, 日科技連出版社, 1966.
11) L. G. Jones, Software Process Improvement and product Line Practice, Softare Engineering Institute, 2004.
12) S. Kato and N. Yamaguchi, Variation Management for Software Product Lines with Cumulative Coverage of Feature Interactions, Proc. of the SPLC 2011, IEEE Computer Society, Aug. 2011, pp. 140-149.
13) 小暮 正夫, 工程能力の理論とその応用, 日科技連出版社, 1975.
14) D. Leffingwell, Agile Software Requirements, Addison-Wesley, 2011.
15) R. C. Martin, Agile Software Development, Prentice Hall PTR, 2003.
16) S. McConnel, Software Project Survival Guide, Microsoft Press, 1998.
17) 蓑谷 千凰彦, 統計分布ハンドブック, 朝倉書店, 2010.
18) M. A. Noor, et al., Agile Product Line Planning, J. of Systems and Software, Vol. 81, No. 6, 2008, pp. 868-882.
19) K. Pohl, et al., Software Product Line Engineering: Foundations, Principles and Techniques, Springer, 2005.
20) J. Savolainen, et al., Combining Different Product Line Models to Balance Needs of Product Differentiation and Reuse, High Confidence Software Reuse in Large Systems, LNCS Vol. 5030, Springer, May 2008, pp.116-129.
21) K. Schmid, Scoping Software Product Lines, Proc. of the 1st Software Product Lines Conference, Kluwer Academic, Aug. 2000, pp.513-532.
22) K. Schwaber, Scrum Development Process, Business Object Design and Implementation, Springer, Oct. 1995, pp. 117-134.
23) C. Tischer, et al., Bosch Gasoline Systems, F. van der Linden, et al. (eds.), Software Product Line in Action, Springer, 2007, pp. 133-148.
24) J. M. Yuran, Quality Control Handbook, McGraw-Hill, 1951.