複数プロダクトのエンタープライズアジャイル開発方法の提案と評価

田中 優之[†1]　青山 幹雄[†2]

概要：本稿ではチームを超えて複数のプロダクトを開発するエンタープライズアジャイル開発方法を提案し，適用した評価を報告する．LeSS Huge をはじめエンタープライズアジャイル開発方法は数多く存在し，日本国内においても事例が増えている．しかし，それらは既存のフレームワークに則り一つのプロダクトに対してエンタープライズアジャイル開発方法を適用している．著者らが所属するヤフーにおいては DevOps の思想が組織全体に浸透しているが外部環境（ビジネス環境，マーケット等），内部環境（組織の方針，開発リソース等）の変化により柔軟に対応するため複数のプロダクトを同時に扱い大規模にアジャイル開発を行う必要があると考えた．そこで Yahoo!Map，Yahoo!カーナビ，Yahoo!地図，地図プラットホームを開発，運用する部門においてエンタープライズアジャイルのフレームワークの一つである LeSS Huge をベースに複数プロダクトに対応するエンタープライズアジャイル開発方法を提案し，実践した．その結果，組織の開発体制を柔軟に変更し，かつ，複数のプロダクトの開発を進めることができた．本稿ではその実践結果から提案方法の有効性を示す．

キーワード：大規模アジャイル開発，エンタープライズアジャイル，Large Scale Scrum，LeSS Huge，スクラム

A Distributed Agile Software Development Method for Multi Products and Its Practical Evaluation

MASAYUKI TANAKA[†1]　MIKIO AOYAMA[†2]

Keywords: Distributed Agile Development，Agile Enterprise，Large Scale Scrum，Less Huge，Scrum

1. はじめに

大企業(エンタープライズ)のソフトウェア開発の現場にアジャイルを導入することには様々な課題が存在する．チーム運営に関する課題，他部署との調整の課題，そして組織全体の課題である[9]．小規模のスタートアップにおいては課題にならないことであっても大企業においては解決までに時間を要することは珍しくない．アジャイル開発の方法を駆使し，価値あるプロダクトを提供し続けるスタートアップが様々な分野において大企業の競合となっている．

エンタープライズアジャイルのフレームワークは数多く存在する．その中でも Large Scale Scrum，LeSS Huge[3]は Scrum Alliance が普及を推進している[14]こともあり事例が多く存在するフレームワークであり，国内における事例も年々増えてきている[10][15][17]. しかし, Large Scale Scrum, LeSS Huge においては一つのプロダクトでフレームワークを適用することが前提になっており，複数のプロダクトをエンタープライズアジャイルで開発する場合は複数の Large Scale Scrum，LeSS Huge を組織する必要がある．この場合，フレームワークのルール上ではプロダクトごとにプロダクトバックログを運用することとなる．

今日，多くの組織は複数のプロダクトを提供しており，各プロダクトが連携を取りつつ外部環境（組織の方針，マーケットの状況等），および，内部環境（組織の方針変更，開発リソース等）へ柔軟に対応することが必要となることが多い．例えば，期初に組織の方針変更（マクロな視点）に対応しつつマーケットおよびユーザのニーズ（ミクロな視点), そして開発リソースを考慮し複数のプロダクトの開発を続ける必要がある．

本稿では複数のプロダクトにエンタープライズアジャイル開発方法を適用しつつ複数のプロダクトを一つのプロダクトバックログで管理することで複数プロダクトを同時に扱い，外部環境，内部環境に対応するエンタープライズアジャイル開発方法を提案し, その実践と評価を報告する．

2. 研究課題

本稿ではエンタープライズアジャイルを複数のプロダクトに対して 1 つのプロダクトバックログを用いて開発する．エンタープライズアジャイルのフレームワークとして LeSS Huge など様々なフレームワークが存在するが複数のプロダクトに対して適用する開発方法と実践例は少ない．事例の多い LeSS，LeSS Huge においても 1 つのプロダクトに対して実施するエンタープライズアジャイル開発方法となっている．そこで，上述の背景を踏まえ以下 2 点を研究

†1 ヤフー(株)
YAHOO JAPAN CORPORATION
†2 南山大学
Nanzan University

課題（RQ）とする．
RQ1: 変化に対応可能な複数プロダクトを扱うエンタープライズアジャイル開発方法はどうあるべきか
RQ2: 提案方法論は実システムに対して有効か

3. 関連研究

3.1 スクラム

スクラム（Scrum）とはアジャイル開発方法の一つでありアジャイル開発を進めるための具体的なルールを定義しているフレームワークの一つである[8]．開発チームが定めた期間(スプリント)において作業計画（スプリント），見積もり（リファインメント），開発作業，製品の出荷，開発の振り返り（スプリントレトロスペクティブ）を繰り返し行う方法である．

スクラムにおいてはプロダクトに必要な機能，要望，修正などを一覧にしたリストであるプロダクトバックログがプロダクトの直近の開発予定から将来の開発予定まで表す情報源になる．プロダクトバックログは複数のプロダクトバックログアイテムから構成される．プロダクトバックログアイテムには仕様の詳細，優先度，見積もり，そしてそのプロダクトバックログアイテムの完了時に確認すべきテスト項目が記載される．リファインメント時にはこれらプロダクトバックログアイテムの詳細の記載，見積もりの実施，そしてプロダクトバックログ内での並び替えおよび優先度付けを行う．このプロダクトバックログの管理は煩雑になりやすいため筆者が所属する組織においてはプロジェクト管理ツール Jira Software[1]を利用している．SaaS 型の多機能プロジェクト管理ツールであり，特にアジャイル開発を行う企業において導入実績が多い．

欧米諸国ではスクラムによるソフトウェア開発が一般的になっており，最近ではソフトウェア開発現場だけではなく組織全体にスクラムの概念を適用する事例も多くなってきている[2]．一方で国内においては一般消費者向けサービスを主要な事業とする IT 企業またはベンチャー企業においてスクラムによるソフトウェア開発は年々広がってきている．その他の企業においてはスクラム等のアジャイル開発方法の導入事例は少なく，欧米諸国のような組織全体でスクラム，アジャイルの思想を取り入れる事例はほとんど無いといって良い．

3.2 DevOps

大企業においては開発者（Dev）と運用担当者（Ops）が分かれていることが前提であったが 2009 年に Flickr 社の「10+ Deploys Per Day」が Velocity 2009 において発表され DevOps の提案がされた[5]．仮想化やクラウドなどの技術の変化を背景に Dev と Ops が互いの技術をオーバーラップさせ協業することを主張されている．

スタートアップにおいてはこの DevOps の概念を実施することは組織の柔軟さも相まって導入が進んでいった．大企業においては今までと異なる思想であることもあり自動化されたインフラ環境の構築など技術的な改善をはじめ，Dev と Ops が協業する文化の醸成にも時間がかかった．しかし現在では Google，Amazon をはじめ欧米の巨大企業においても当然の文化となり[4]，筆者が所属する組織においても DevOps の文化は根付いている．

本稿においては Dev と Ops の協業と自動化によって課題解決に取り組む文化を DevOps と指すこととする．DevOps の思想は後述の LeSS，LeSS Huge において開発チームに要求されるフィーチャチームに必要な文化の一つでありアジャイルなチームを実現するために必要な概念である．

3.3 エンタープライズアジャイル

エンタープライズアジャイルには統一された定義は存在しないが，本稿では文献 [9]において言及されているように「アジャイル開発手法をチームを超えて適用する取り組みの総称」と定義する．

LeSS，LeSS Huge については共にエンタープライズアジャイルの開発方法の一つであり数少ない実績が多い手法である[13][16]．従って，本稿の課題は新たなエンタープライズアジャイルの開発方法の提案となる．

3.4 Large Scale Scrum（LeSS）

Large Scale Scrum（LeSS）はスクラムをベースにした大規模アジャイル開発のフレームワークである[3]．

LeSS はスクラムをベースにしたフレームワークであるが図 1 に示すように複数のチームから構成され，製品およびサービスに責任を持つ担当者（プロダクトオーナ）が 1 名，製品に関する作業を集約したプロダクトバックログを一つだけ持つ．このプロダクトバックログについてはスクラムにおけるそれと同様である．全ての開発チームは同じスプリント期間で作業を進め出荷，リリース可能な製品開発を進める．各スクラムチームはスプリントプランニング（スプリントプランニング 2），スプリントレトロスペクティブ (チームレトロスペクティブ)などスクラムで実施されるセレモニーを行う．

LeSS においてはさらにプロダクトオーナと実施するスプリントプランニング 1，プロダクトオーナ，各開発チームのスクラムマスターが集まるオーバーオールレトロスペクティブ，プロダクトオーナ，各開発チーム（またはチームの代表者）とプロダクトオーナが行うオーバーオールプロダクトバックログリファインメントを行う．これらはスクラムにはないルールであり，複数のスクラム開発チームが同期を取りつつ開発を行うために追加されたルールとなっている．

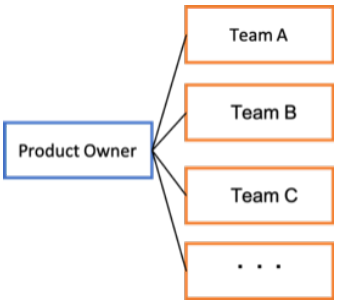

図 1　LeSS の組織構造

Fig. 1　Organizational Structure of LeSS

3.5 LeSS Huge

チーム数が 8 つ以上の場合は LeSS Huge というフレームワークを利用することが可能となる（図 2）．LeSS Huge は LeSS と同じ概念であり大きな差異はないが LeSS よりも大規模なアジャイル開発が可能となるよう設計されている[3].

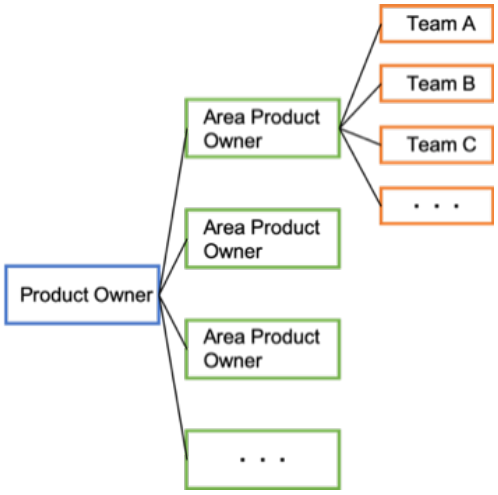

図 2　LeSS Huge

Fig. 2　LeSS Huge

LeSS Huge においても LeSS と同様にプロダクトオーナは 1 名である．しかし製品の機能（フィーチャ）毎に責任者（エリアプロダクトオーナ）を立てることでプロダクトオーナの責任と役割の委譲を行うことでスケールさせることを可能としている．

エリアプロダクトオーナは責任を持つ機能の開発を開発チームと進める．開発チームはエリアプロダクトオーナが責任を持ついずれかのエリアに所属しエリアプロダクトオーナと共に開発作業を進めることになる．

LeSS Huge においては新たなフィーチャを開発する必要が出てきた場合，新しくエリアを追加し対応する．この設計により新たな機能要求が出てきた際にもスケールが容易な設計になっている．しかし LeSS Huge は 1 つのプロダクトを扱うエンタープライズアジャイルのフレームワークである．別のプロダクトを扱うためには新たにプロダクトオーナを立て，別の LeSS Huge を構築することがフレームワークを最大限に活用する方法である．

LeSS Huge においては単一プロダクト開発において多くの実績があるエンタープライズアジャイルの方法である．一方で大企業においては複数のプロダクトを一つの組織で同時に開発，運用を行うケースもある．そのようなケースにおいて複数の LeSS Huge が連携して開発する方法は提案されていない．

3.6 大規模アジャイル開発におけるプロダクトオーナ

LeSS，LeSS Huge などの大規模アジャイル開発においてプロダクトオーナは多くの役割を担い重要な役割である．主な役割としては，プロダクトの新たな機能の開発計画の整理，プロダクトに関係するステークホルダとの交渉，リスクマネジメント，リリース計画の整理などがある．

図 3 に実際に大規模アジャイル開発組織のプロダクトオーナを担当する人々からのアンケートを元に作成されたプロダクトオーナの行動を示す[6].

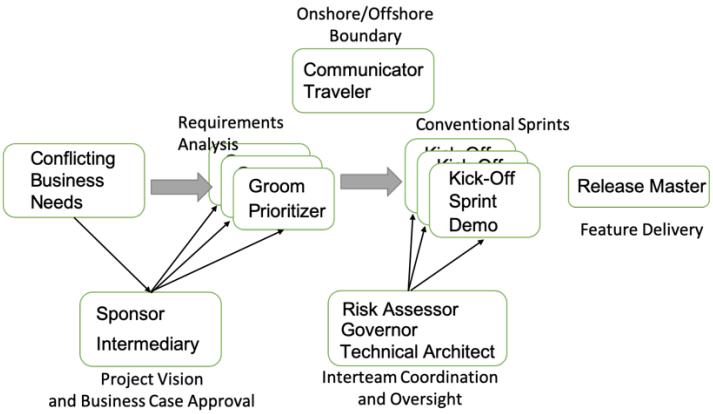

図 3　Product Owner の行動

Fig. 3　Product Owner activities

図 3 の通り大規模アジャイル開発におけるプロダクトオーナの役割は多岐に渡る．LeSS，LeSS Huge においてもプロダクトオーナがこれらの役割を担うことが順調に開発を進めるための重要な要素の一つである．しかし，これら全ての役割を担当することは容易ではない．LeSS Huge はエリアプロダクトオーナという新たな役割を定義することでプロダクトオーナの権限の委譲を行っている．その結果，LeSS と比較してより大規模なアジャイル開発に対応できるフレームワークとなっている．

3.7 ストーリポイントとベロシティ

プロダクトバックログで管理されるストーリは任意の作業量をベースに設定した相対ポイント（ストーリポイント）が割り振られている．スクラムガイドにおいては明確にストーリポイントについての記述はないがスクラム，アジャイル開発の手法においてよく利用される．ストーリポイントを用いた見積もり手法については Microsoft で実施された研究において従来のプロジェクト見積もりの手法より正確であることが示されている[11].

チームが 1 回のイテレーション（スプリント）で完了させたストーリのストーリポイントの合計値をベロシティと

いう[12]．LeSS，LeSS Huge においては特に複数のチームがスプリントを進める中で各チームのベロシティが計測されることになる．しかし計測されたベロシティをチーム間で比較をすることは非常に難しい．なぜならポイントの基準となる作業内容の調整やチーム内での任意の作業についての熟練度合い，チームメンバ数などに影響されるためである．そのため任意のスプリントにおいてあるチームが他のチームより完了したストーリポイント数が多いことはそのチームが他より優れているという指標にはならない．

4. アプローチ

図 4 に本研究のアプローチを示す．LeSS Huge をベースとし複数プロダクトに対応するため既存フレームワークの概念に以下項目を追加，変更した．

(1) 一つのプロダクトバックログに複数のプロダクトのプロダクトバックログアイテムを登録し管理する

(2) 複数のプロダクトの優先度を管理する Multi Product Owner を配置

本アプローチにおいては主に 3 つのプロダクト（Yahoo! Map アプリ，Yahoo!カーナビアプリ，Yahoo!地図）を扱い Multi Product Owner の役割を担当したメンバは 3 プロダクトの開発進捗を代理で管理する役割も担っている．各プロダクトの責任者（Product Owner）は図中に示すようにプロダクト毎に存在する．

内部環境（組織の方針変更，開発リソース等）および外部環境（ユーザのニーズ，市場の変化等）の様々な要素とその変化に対応するように複数のプロダクトを成長させていくことを可能とする．

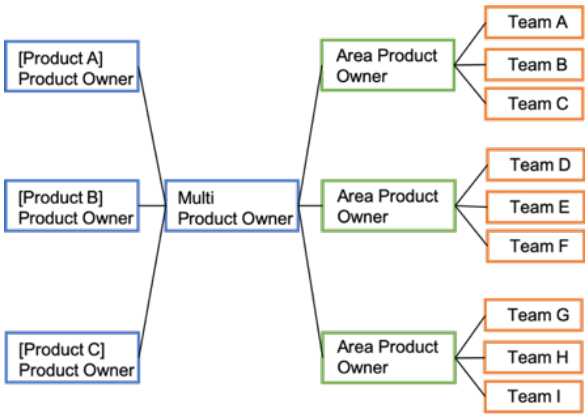

図 4　LeSS Huge で複数プロダクトを扱う

Fig. 4　LeSS Huge for Multiple Product

5. マルチプロダクトアジャイル開発方法論

5.1 開発方法論の概要

本稿で提案するマルチプロダクトアジャイル開発方法論を図 5 に示す．

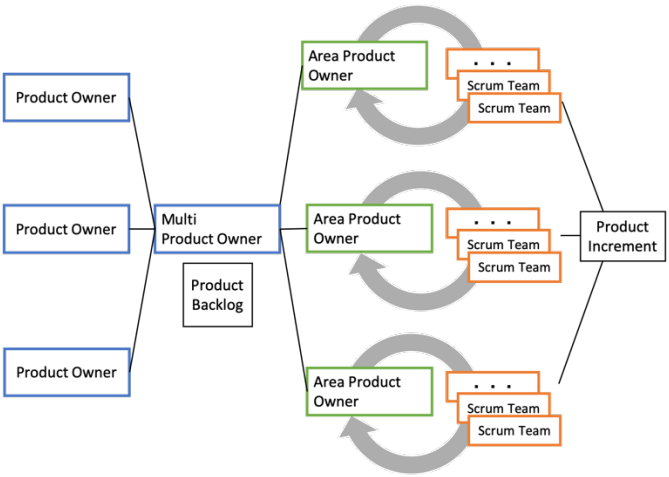

図 5　マルチプロダクトアジャイル開発方法モデル

Fig. 5　A Distributed Agile Software Development Method for Multi Products

マルチプロダクトアジャイル開発方法モデルは LeSS Huge をベースにしているが，プロダクトバックログの管理方法，複数プロダクトの優先度を管理するマルチプロダクトオーナ(Multi Product Owner)の役割の追加の 2 点が拡張した点である．

5.2 マルチプロダクトのバックログの統合管理

マルチプロダクトアジャイル開発方法モデルにおいては 1 つのプロダクトバックログで複数プロダクトのプロダクトバックログアイテムを管理する．

LeSS，LeSS Huge においてはプロダクトバックログには 1 つのプロダクトのプロダクトバックログアイテムのみが管理される．これに対して，マルチプロダクトアジャイル開発方法モデルにおいては，扱うプロダクト全てのプロダクトバックログアイテムを統合して管理することで複数プロダクトの開発優先度を柔軟に管理することを可能とする．

プロダクトバックログの整理はスクラムや LeSS，LeSS Huge と同様にリファインメントの時間を利用して整理を進める．複数のプロダクトのプロダクトバックログアイテムが一つのプロダクトバックログで管理可能とするためには，管理方法を工夫する必要がある．この点については JIRA Software[1]が提供する機能を用いてプロダクトバックログを継続的に整理することが一つの解決方法である．

5.3 マルチプロダクトオーナ(Multi Product Owner)の役割

マルチプロダクトアジャイル開発方法モデルにおいては扱う複数プロダクトのフィーチャの開発優先度を外部環境，内部環境に対応して柔軟に判断する役割が必要になる．この役割を担うメンバを Multi Product Owner と呼ぶこととする．

Multi Product Owner は個別のプロダクトの責任は持たない．外部環境，内部環境を踏まえたプロダクトバックログの管理に責任を持つことで，組織とプロダクトの成長を全

体最適する．

5.4 プロダクトオーナ(Product Owner)の役割

Product Owner は担当するプロダクトの成長に責任を持つ．プロダクトの新たな機能の開発，不具合の改修は開発チームが実施するがプロダクトに関する最終的な責任は Product Owner にある．そのため Product Owner はプロダクトが目指すビジョンを開発チームに伝えて，共有する．具体的な方法は Product Owner を担当する個人により様々になる．

マルチプロダクトアジャイル開発方法モデルにおいては Product Owner の意見を Multi Product Owner と議論することで組織とプロダクトの状況に合わせて全体最適化を図る．

5.5 エリアプロダクトオーナ(Area Product Owner)の役割

Area Product Owner は任意のプロダクトの機能の開発に責任を持つ．Area は複数の開発チームから組織され，Area が担当する機能の開発を Area Product Owner は管理する．

マルチプロダクトアジャイル開発方法モデルにおいて，Area Product Owner は担当する Area が開発しているプロダクトを Product Owner と開発状況について共有を随時行う．そして組織全体の状況についても Multi Product Owner と連携を取ることで担当する Area が全体最適されたプロダクトの開発に集中できるよう管理する．

5.6 開発プロセス

開発プロセスについては LeSS Huge で定義されているプロセスと同様に進める．図 6 に木曜日に始まり，水曜日に終わる 1 週間スプリントを実施する例を示す．各チームのセレモニー実施時間はチームにより様々に設定することも可能である．図 6 においてはまとまった開発作業時間を確保することを目的に月曜日，金曜日にはセレモニーを少なくすることで開発効率を上げる工夫を行なっている．

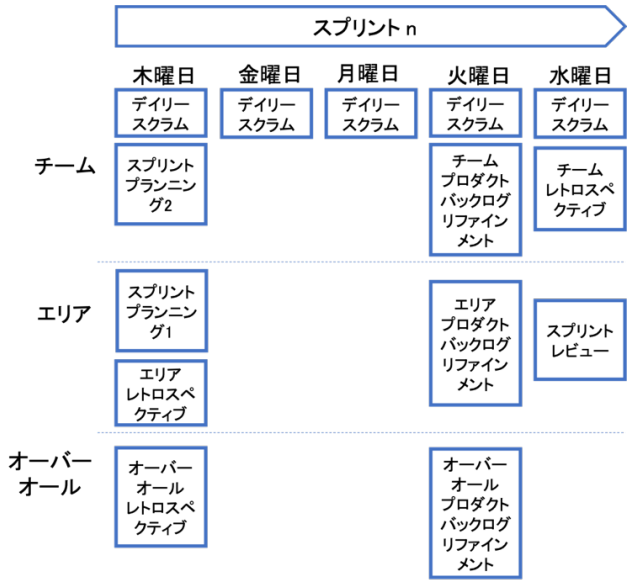

図 6 スプリント

Fig. 6 Sprint

5.7 Multi Product Owner，Product Owner，Area Product Owner 間のコミュニケーション構造

マルチプロダクトアジャイル開発方法モデルにおいては Multi Product Owner，Product Owner，Area Product Owner のコミュニケーションを円滑に進めることが重要になる．

Multi Product Owner はプロダクトについての責任を負ってはいないが，Product Owner と議論した結果である組織および外部環境の状況を考慮した優先度を把握し，設定する．Multi Product Owner は議論した内容を用いてプロダクトバックログの整理を進める．より詳細な開発計画については Area Product Owner とも議論し整理を進める．このとき，Area Product Owner が担当するフィーチャは優先度に応じて変化する．そのため随時 Product Owner とも直接コミュニケーションをするケースがある．当面の予定が決まっているエリアにおいては図 7 に示すように直接 Product Owner と連携をとることも発生する．

このように Multi Product Owner，Product Owner，Area Product Owner は相互に密に連携を取ることでプロダクトバックログを継続的に整理する．

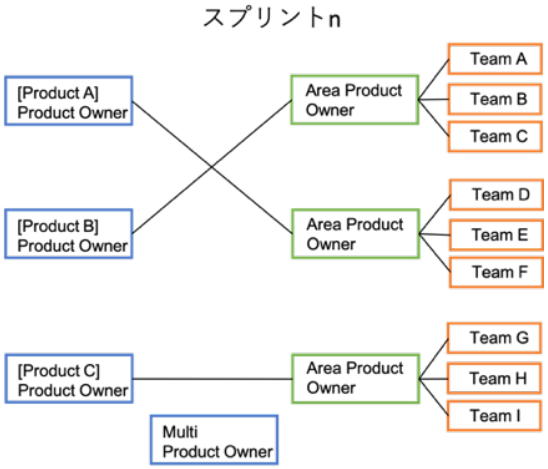

図 7 Product Owner と Area Product Owner の情報交換

Fig. 7 Communicate Structure between Product Owner and Area Product Owner

5.8 担当するプロダクトとエリアのパターン分類

マルチプロダクトアジャイル開発方法では複数のプロダクトを一つのプロダクトバックログで管理している．そのため任意のスプリントにおいてはプロダクトバックログの優先度に応じて各エリアは担当するプロダクト，フィーチャは変わることが発生する．この場合，任意のスプリントにおいて各エリアと担当するプロダクトの関係を，次の 3 パターンに分類した．

(1) 任意のプロダクト開発に集中する

図 8 に示す任意のスプリントにおいては Product A の大規模なフィーチャの開発に集中し，全てのチームが Product A の開発に集中する．ただし，各エリアが対応するフィー

チャはそのエリア内で完結するよう進める．そのため，同じプロダクトを開発しているが，フィーチャは異なるため並行開発が進めやすい状況にある．

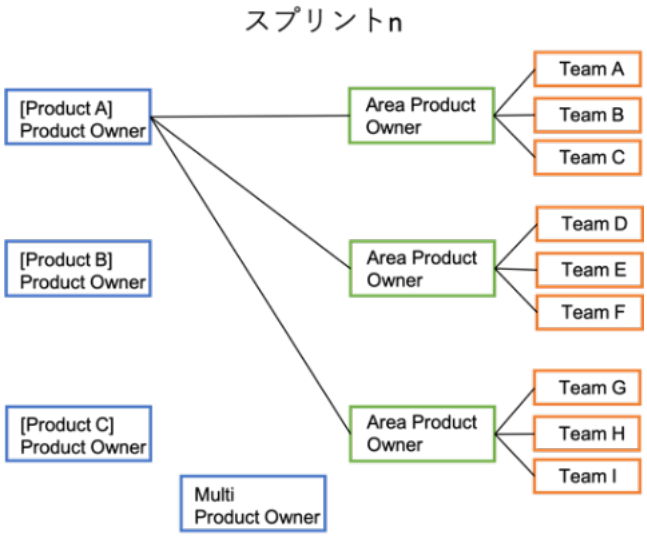

図 8　パターン 1

Fig. 8　Pattern 1

(2)　任意のプロダクト開発は行わない

図 9 に示す任意のスプリントにおいては Product A, Product B の開発に集中しており各エリアの Area Product Owner は担当するフィーチャの Product Owner と連携し開発を進める. 開発を担当する Area が割当てられていない状況においても，通常のスクラムと同様に今後予定されているフィーチャのリファインメントは Product Owner, Multi Product Owner, Area Product Owner を中心に準備を進める．

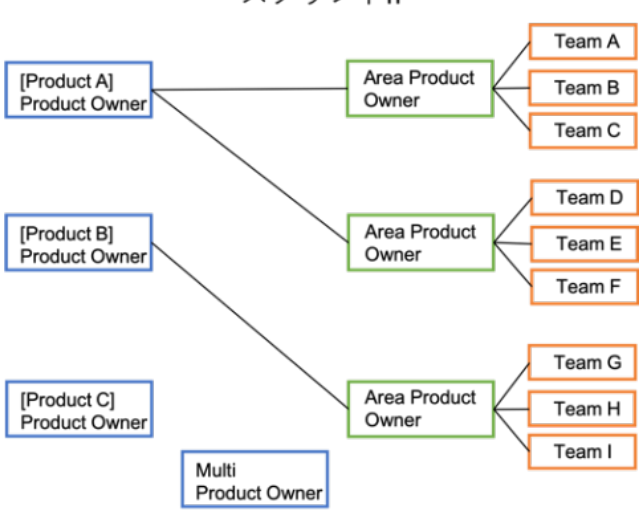

図 9　パターン 2

Fig. 9　Pattern 2

(3)　全てのプロダクトの開発を並行して進める

図 10 に示す任意のスプリントにおいては各 Product にそれぞれ Area が割り当てられ, 担当するフィーチャの開発を進める．

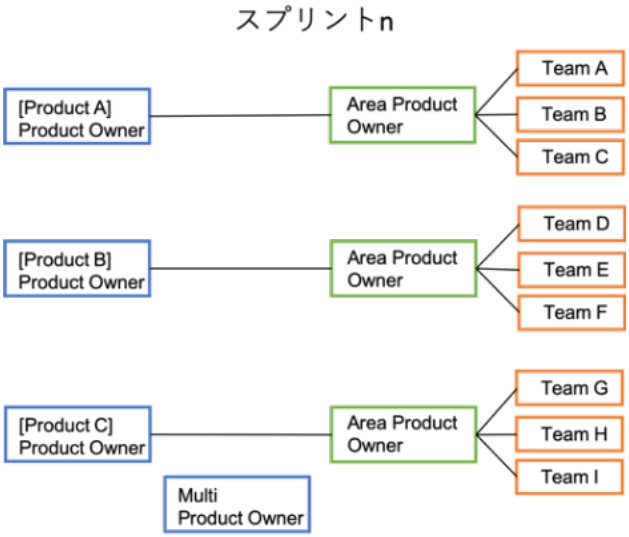

図 10　パターン 3

Fig. 10　Pattern 3

6.　提案方法論の適用

2017 年 11 月 30 日（木）から 2018 年 10 月 10 日（水）の期間に Yahoo! Map アプリ, Yahoo!カーナビアプリ, Yahoo! 地図を扱う部門において適用した事例を示す．組織変更のより変動があったがチーム数は常時 10 チーム前後，各チームは 5 名前後であり各チームにはスクラムマスターの役割を担うメンバは 1 名であった．

6.1　直近 3 スプリントの移動平均ベロシティ

図 11 に 2017 年 11 月 30 日(木)から 2018 年 10 月 10 日(水)の期間に提案開発方法を実施した直近 3 スプリントの移動平均ベロシティを示す．スプリントは木曜日に始まり水曜日に終了する 1 週間で行っている．

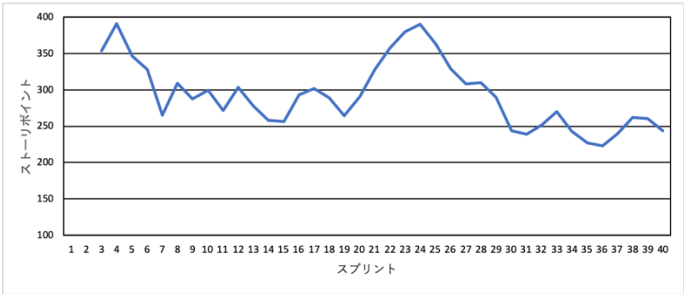

図 11　直近 3 スプリントの移動平均ベロシティ

Fig. 11　Moving Average Velocity over the Last 3 Sprints

6.1.1 直近 3 スプリントの移動平均ベロシティとストーリポイント

各スプリントのベロシティは各スクラムチームにおいてメンバの勤怠状況、稼働日または稼働時間の少ないスプリントなどの影響を受けやすい．そのため直近 3 スプリントの移動平均ベロシティを評価指標として，その影響を少なくした．また，ストーリポイントは相対ポイントであるためチーム間でポイントに対する作業量の認識を一致させる

ことは難しいが，組織全体でストーリポイントの基準となる作業量の調整を行う取り組みをしたことで認識のずれが少なくなるようにした．

6.1.2 組織変更による影響

筆者が所属する組織においては毎年 4 月と 10 月に大きな組織変更が行われる．スプリント 16 は 2018 年 3 月 29 日(木)に始まり 2018 年 4 月 4 日(月)に終わるスプリントであった．このスプリント 16 中にも実際に組織変更の通知があり変更に合わせ開発組織の再編を実施した．

組織変更により開発チームのメンバ構成，チーム数，そして Multi Product Owner，Area Product Owner の変更があった．この期間については移行期間として開発業務より体制立ち上げを重視し，業務の引き継ぎに時間を充てた．

6.1.3 大規模開発期のベロシティの上昇

スプリント 1 からスプリント 6 の期間に直近 3 スプリントの移動平均ベロシティが上昇している．この期間において，図 8 に示すような体制で一つのプロダクトのフィーチャを複数エリアで開発を進めていた．平均ベロシティ値が高い傾向にあるのはこの影響であると考えられる．

スプリント 20 からスプリント 29 の期間においても直近 3 スプリントの平均ベロシティが上昇している．この期間では，夏休みやお盆などの季節行事に合わせた開発が重なっていたことが主な要因であった．また，開発体制については図 9 と図 10 示した体制を開発状況に応じて柔軟に切り替えつつ，複数のプロダクトの開発を並行して進めた．

6.2 コミット率

図 12 に 2017 年 11 月 30 日(木)から 2018 年 10 月 10 日(水)の間に提案方法を適用した組織全体のコミット率を式(1)で定義する．

$$\text{コミット率}\ (= \frac{\text{完了したストーリ(ポイント)}}{\text{プランニング時に決定したストーリ(ポイント)}}) \quad (1)$$

コミット率についてはスプリントプランニング時に各チームがそのスプリントにおいて達成可能なラインを合意した上での結果となっているため直近 3 スプリントの移動平均とする必要はないと考えた．

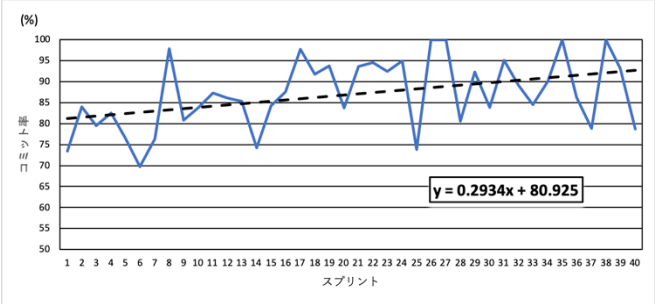

図 12　コミット率

Fig. 12　Commitment Rate

コミット率は 70%から 100%の間で変動しているが，体制の立ち上げ初期を除くと組織全体として概ね 80%のコミット率は維持できている状況であった．最大値で 100%となり，最小値は開発体制をはじめ年末年始休暇を終えた後のスプリント 6 における 69.7%であった．

スプリント 6，14，25 では前後のスプリントと比較してコミット率が低下している傾向がある．これらのスプリントにおいては緊急性がなく先送りにしていた，いわゆる Undone ワーク[3]をまとめて実施し，解消することに開発チームが取り組んだ結果と考えられる．このような Undone ワークを解消する周期は，開発する機能の規模によって変化するであろう．

7. 評価

7.1 RQ1: 変化に対応可能な複数プロダクトを扱うエンタープライズアジャイル開発方法はどうあるべきか

本稿で提案したマルチプロダクトアジャイル開発方法論は変化に対応できることがベロシティ，コミット率の推移より明らかになった．図 12 に示したコミット率の推移が上昇傾向にあること，実際に組織変更があった際にも機能していることから図 13 のような開発体制を取り組織が変化に対応できると考える．

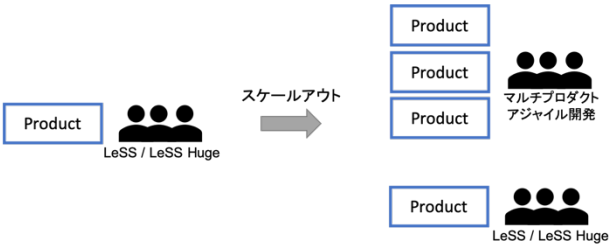

図 13　スケールアウト

Fig. 13　Scale out

LeSS，LeSS Huge においてはプロダクトごとに組織を構成しスケールアウトしていくこととなる．しかし，組織内外の変化に対応するエンタープライズアジャイル開発方法としては提案の開発方法を用い図 13 に示すように開発組織とプロダクトを柔軟に配置できることが必要と考える．

7.2 RQ2: 提案方法論は実システムに対して有効か

複数のプロダクトの開発を組織全体の優先度に応じて変更しつつ開発を進めた．直近 3 スプリントの移動平均ベロシティ，コミット率は 70%から 100%の間で変動が見られたが，組織変更時の開発組織の再編中にも 80%前後のコミット率を維持できた．また期間中のコミット率は上昇傾向にあった．これらのことより，提案した開発方法が有効であったと評価できる．

8. 考察

8.1 Multi Product Owner の役割

Multi Product Owner が担当する役割は図 3 における Requirements Analysis， Project Vision and Business Case Approval に相当する．LeSS Huge において LeSS のプロダ

クトオーナの権限委譲をすることでより大規模な開発へ対応しているように，提案方法が複数のプロダクトを扱うため LeSS Huge におけるプロダクトオーナの役割を権限委譲することで定義されている．そのため Multi Product Owner の役割は大規模アジャイル開発に必要な役割の一部となっており，提案方法においてもより大規模な開発へ対応するために権限委譲された役割である．

8.2 本アプローチにおける Product Owner の役割

提案方法において Product Owner の役割は本来の LeSS，LeSS Huge のフレームワーク内で示される役割から変化した部分があった．Product Owner は Area Product Owner とも連携はするが，提案方法では Multi Product Owner と密に連携することが多くなった．そのため Product Owner からは開発チームの状況が把握しにくいという意見があった．

この点については図 3 を参考に Product Owner，Multi Product Owner，Area Product Owner が担当する役割を明示することで改善が可能と考える．

8.3 フィーチャ間の依存関係による影響

開発規模がある程度大きな案件では特定のフィーチャがリリースされていないことによる影響（開発の遅延などによる）を受け別のエリアやチームが開発しているフィーチャを任意のスプリントにリリースできないことがある．この問題は，提案方法においても発生した．しかし，提案方法固有の問題ではなく LeSS Huge を始めエンタープライズアジャイル開発を実践する場合に発生しやすい問題である．

対策としては，Product Owner，Area Product Owner が密に連携をとり開発状況を整理することでフィーチャ間の依存関係による遅延発生を防ぎつつ，発生した場合にはストーリの分割を行うなど他のストーリに着手することを準備する方法が考えられる．

8.4 アジャイル開発組織のスケールアウト

LeSS，LeSS Huge において複数プロダクトを扱う場合はプロダクトごとに組織を拡大させていくことになる．しかし，今回の提案手法によって複数のプロダクトを 1 つの組織でアジャイル開発をする有効性を示せた．このことより，図 13 で示したように LeSS，LeSS Huge のプロダクト中心の組織だけでなく，プロダクトと組織の状況（外部環境，内部環境）を考慮したアジャイル開発組織の構築とスケールアウトが可能と考える．

9. 今後の課題

9.1 Multi Product Owner の役割の明確化

Multi Product Owner の果たす役割は新たな概念でありより明確に責任範疇について明示する必要がある．

Multi Product Owner は各 Product に対して責任を持っているのではなく内部環境，外部環境など様々な要素を統合的に考えた上での優先度付けに責任を持っている．Product Owner が担当する役割と明確に差別化した責務を明示するとこで，提案方法がより機能すると考える．

9.2 Multi Product Owner，Product Owner 間の同期コミュニケーション

提案方法においては LeSS Huge をベースにしており LeSS Huge が定める範囲での担当者間でのセレモニーの実施は行なっている．しかし，提案方法において追加している Multi Product Owner という役割は明確にコミュニケーション方法を新たに定義する必要がある．特に，Multi Product Owner，Product Owner 間のコミュニケーションは提案方法が機能するための重要な要素であり，スプリント毎に同期可能なセレモニーを追加で定義する必要がある．

9.3 本手法の適用可能な条件の整理

本事例においては 3 つのプロダクトを扱っているが，全てのプロダクトに地図という共通の機能が存在する．各エリアは様々なフィーチャを担当することを要求されているが，共通して地図に関係するフィーチャを主に担当しているケースが多く，地図というドメインおよびフィーチャに詳しいメンバが多い状況であった．そのため，あるチームが今まで経験したことのないプロダクトの開発を始めたケースにおいても地図というドメイン知識が豊富なケースもあり，それらが提案方法の有効性に寄与していた可能性が考えられる．

そのため対応するプロダクトに共通のフィーチャとは異なるドメインを扱うようなケースにおいても提案方法の適用が有効かどうかについては検討の余地がある．

10. まとめ

複数のプロダクトを 1 つのプロダクトバックログ上で，並行して開発するエンタープライズアジャイル開発方法を提案し，3 つの市販プロダクトとそのプラットフォームの開発に適用し，その効果を確認した．このような市販プロダクト開発では，外部環境および内部環境に対応した．このことより，LeSS，LeSS Huge のプロダクト中心のアジャイル開発組織のスケールアウトだけでなく，組織の状況を考慮したアジャイル開発組織の構築とスケールアウトが可能と考える．

参考文献

[1] Atlassian, Jira Software, https://ja.atlassian.com/software/jira (参照 2019-05-04).
[2] B. Schlatmann, ING's Agile Transformation, McKinsey & Company, https://www.mckinsey.com/industries/financial-services/our-insights/ings-agile-transformation?cid=other-eml-ttn-mkq-mck-oth-1712 (参照 2019-04-01).
[3] C. Larman, and B. Vodde, Large-Scale Scrum More with LeSS, Addison-Wesley, 2016.
[4] G. Kim, et al., The DevOps Handbook, IT Revolution Press, 2016 [榊原 彰 (監修), 長尾 高弘(訳), The DevOps ハンドブック: 理論・原則・実践のすべて, 日経 BP 社, 2017].
[5] J. Allspaw, 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr, Velocity 2009, https://www.slideshare.net/jallspaw/10-

deploys-per-day-dev-and-ops-cooperation-at-flickr, (参照 2019-04-01).
[6] J. M. Bass, and A. Haxby, Tailoring Product Ownership in Large-Scale Agile Projects, IEEE Software, Vol.36, No2, Mar./Apr. 2019, pp.58-63.
[7] J. Smed, et al., DevOps: A Definition and Perceived Adoption Impediments, Proc. XP 2015, LNBIP Vol. 212, Springer, May 2015, pp. 166-177.
[8] K. Schwaber, and J. Sutherland, The Scrum Guide, Nov. 2017, https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf (参照 2019-04-01).
[9] 川口 恭伸, ほか, 楽天でのエンタープライズアジャイルとDevOps, 情報処理学会デジタルプラクティス, Vol. 7, No. 3, Jul. 2016, pp. 243-251.
[10] 貝瀬 岳志, ほか, スクラム実践入門, 技術評論社, 2015.
[11] L. Williams, et al., Scrum + Engineering Practices: Experiences of Three Microsoft Teams, Proc. of ESEM 2011, IEEE Computer Society, Sep. 2011, pp. 463-471.
[12] M. Cohn, Agile Estimation and Planning, Prentice Hall, 2005, [安井 力, 角谷 信太郎 (訳), アジャイルな見積もりと計画づくり, 毎日コミュニケーションズ, 2009].
[13] M. Kalenda, et al., Scaling Agile in Large Organizations: Practices, Challenges, and Success Factors, J. of Software: Evolution and Practice, Vol. 30, No. 10, Oct. 2018, pp. 1-24.
[14] Scrum Alliance, Scaling Works to Fit Your Needs, https://www.scrumalliance.org/get-certified/scaling (参照 2019-04-01).
[15] 鈴木 友也, インターネットバンキング案件における大規模スクラムの適用, Agile Japan 2017, Apr. 2017, https://www.agilejapan.org/2017/img/session/document/A-5.pdf , (参照 2019-04-01).
[16] T. Dingsøyr, et al., Agile Development at Scale: The Next Frontier, IEEE Software, Vol. 36, No. 2, Mar./Apr. 2019, pp. 31-36.
[17] 塚越 啓介, 今井 恵子, 小さく始める大規模スクラム, Mar. 2018, https://www.slideshare.net/keisuketsukagoshi/ss-59705472 (参照 2019-04-01).