デジタルプラクティス　Vol.11 No.3(July 2020)

特集投稿論文

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実践

田中 優之[1]　青山 幹雄[2]

[1]ヤフー株式会社　[2]南山大学

本稿ではチームを超えて複数のソフトウェアプロダクトを一括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，実践した経験と評価を報告する．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，それらは既存の開発方法に則り1つのソフトウェア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．筆頭筆者が所属するヤフーにおいてはDevOpsの行動原理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）の変化に柔軟かつ俊敏に対応するため複数の大規模プロダクトに並行してアジャイル開発を行う必要があると考えた．そこでスマートフォン向けアプリ「Yahoo!Map」，「Yahoo!カーナビ」，ブラウザ向けサービス「Yahoo!地図」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開発方法の1つであるLeSS Hugeをベースに複数ソフトウェア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により短期にデリバリを実現した．本稿ではその実践結果から提案方法の有効性を示す．

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さまざまな課題が存在する．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，価値あるソフトウェアプロダクト（以下，プロダクトと略記）を提供し続けるスタートアップがさまざま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開発方法は数多く存在する．その中でもLarge Scale Scrum（以下，LeSSと略記），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開発方法であり，国内における事例も年々増えてきている[10][15]

[17]．しかし， LeSS，LeSS Hugeにおいては1つのプロダクトに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LeSS，LeSS Hugeを組織する必要がある．この場合，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（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）の変化へ柔軟かつ俊敏に対応する必要がある．たとえば，期初に組織の方針変更（内部環境）に対応しつつマーケットおよびユーザのニーズ（外部環境）を考慮しながら複数のプロダクトの開発を調整する必要がある．

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，複数のプロダクトを1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並行して開発し，外部環境，内部環境の変化に俊敏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，その実践の経験と評価を報告する．

2. 研究課題

本稿の課題はエンタープライズアジャイル開発方法を用いて複数のプロダクトを一括開発する方法の提案とその実開発における有効性の評価を示すことである．エンタープライズアジャイルの開発方法としてLeSS Hugeなど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，以下の2点を研究課題（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. 関連研究と実践状況

3.1 アジャイル開発方法と企業におけるアジャイル

3.1.1 スクラム

スクラム（Scrum）は主要なアジャイル開発方法の1つでありアジャイル開発の具体的なルールを定義している[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（スプリントプランニング），見積もり作業（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．スプリントプランニング，リファインメント，スプリントレトロスペクティブなどチームが実施する打合せをスクラムにおいてはセレモニーと定義している．

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を表す情報源になる．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要求仕様の詳細，優先度，数値化された作業量，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述，作業量の数値化，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の一人が所属する組織ではプロジェクト管理ツールJira Software[1]を利用している．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．

また，スクラムを導入，実施していくにあたり各チームではスクラムマスタを任命する．スクラムマスタはスクラムをチームに導入しスクラムでの開発が円滑に進むよう支援をする役割を担う．チームへのスクラムの説明やセレモニーを行うための準備，そして各セレモニーでのファシリテーションなどを行う．特にスクラムを導入したことがないチームや組織においてはスクラムマスタが担うこれらの役割は重要である．

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ある[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企業や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が年々広がってきている．しかし，一般にスクラム等のアジャイル開発方法の導入事例はまだ少なく，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実践している事例は少ない．

3.1.2　DevOpsの行動原理

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社からDevOpsの概念が提案された[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が主張されている．

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る．大企業においては従来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課題の解決をはじめ，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をはじめ欧米の大企業においても当然の文化となり[4]，筆者の一人が所属する組織においてもDevOpsの行動原則は根付いている．

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思想をDevOps行動原則と称することとする．DevOps行動原則は後述するLeSS，LeSS Hugeにおいて顧客価値を中心に開発を行うフィーチャチームが有効に働くための行動原理の1つである．以下，本稿ではDevの側面に焦点を当てて説明する．

3.1.3　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「アジャイル開発方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1つで，実績が多い方法である[13][16]．したがって，本稿の課題はLess, Less Hugeを基礎として，複数プロダクトを並行開発するための，新たなエンタープライズアジャイル開発方法の提案となる．

3.2　大規模アジャイル開発方法

3.2.1　Large Scale Scrum（LeSS）

LeSSはスクラムをベースにした大規模アジャイル開発方法である[3].

LeSSは**図1**に示すように複数のチームから構成される．製品およびサービスに責任を持つ担当者（プロダクトオーナー）が1名（開発者である必要はない），製品に関する作業を集約したプロダクトバックログを1つだけ持つ．このプロダクトバックログはスクラムにおけるそれと同様である．すべての開発チームは同じスプリント期間で作業を行い，リリース可能な製品を開発する．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）などスクラムのプラクティスを行う．

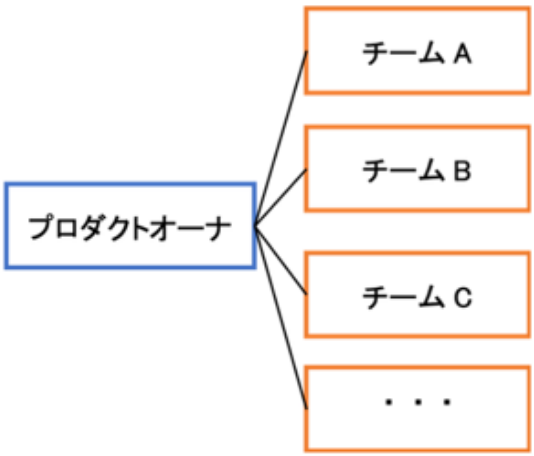

図1　LeSSの組織構造

また，LeSSにおいてはオーバーオールレトロスペクティブ，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というセレモニーを行う．これらのセレモニーは複数のスクラム開発チームが同期して開発するために追加されたセレモニーとなっている．

3.2.2　LeSS Huge

チーム数が8以上の場合を対象とする開発方法がLeSS Hugeである（**図2**）．対象とするチーム数が8以上となっている理由は，これまでの事例から経験的に得られた数値である．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].

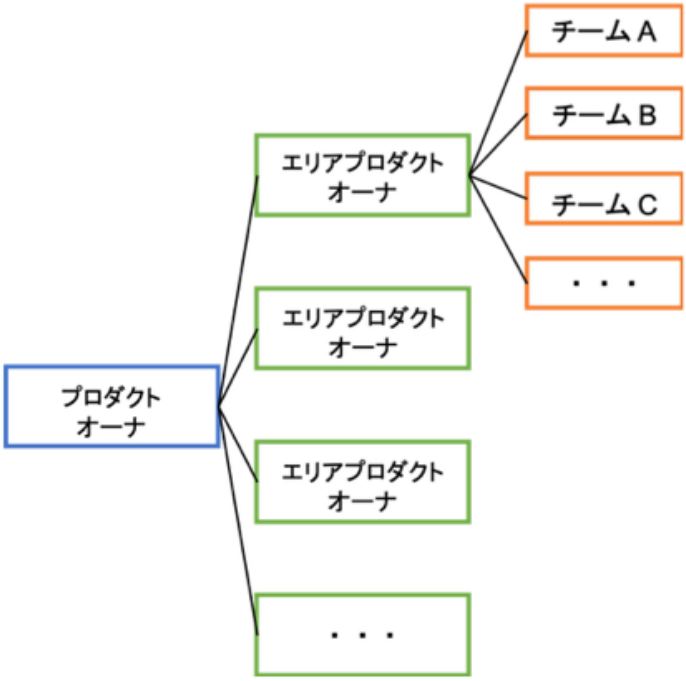

図2　LeSS Huge

LeSS HugeにおいてもLeSSと同様にプロダクトオーナーは1名である．しかし製品のフィーチャ（機能）ごとに責任者（エリアプロダクトオーナー）を設けてプロダクトオーナーの責任と役割を委譲することにより大規模開発を可能としている．ただし，エリアプロダクトオーナーもプロダクトオーナーと同様，開発者である必要はない．

エリアプロダクトオーナーは責任を持つフィーチャのプロジェクト管理等を行う．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．この構造により新たな機能要求に対しても拡張が容易である．

また，リファインメントについては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，エリアプロダクトバックログリファインメント，チームプロダクトバックログリファインメントの3種類がある．各セレモニーの参加者を**表1**に示す．なおチーム代表は各チームのリーダの役割を担うメンバである．LeSS Hugeにおいてはこれらリファイメントを関係者間で実施し，開発を担当するスクラムチームのチームプロダクトバックログリファインメントで作業の詳細化が完了する．

表1　プロダクトバックログリファインメント

セレモニ	参加者
・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	・プロダクトオーナ ・エリアプロダクトオーナ
・エリアプロダクトバックログリファインメント	・エリアプロダクトオーナ ・チーム代表
・チームプロダクトバックログリファインメント	・スクラムチームメンバ

しかしLeSS Hugeは単一のプロダクトを扱うエンタープライズアジャイルの開発方法であるため，異なるプロダクトを扱うためには新たにプロダクトオーナーを設け，異なるLeSS Hugeを構築して，それぞれ個別に開発する必要がある．

大企業においては複数のプロダクトの早期デリバリや変化への俊敏な対応のために，1つの組織で並行して開発，運用を行う必要が高まっている．そのような場合に，複数のLeSS Hugeを連携して開発する方法はこれまで提案されていない．

3.2.3　大規模アジャイル開発におけるプロダクトオーナー

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ーは多くの役割を担う重要な役割である．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．LeSS，LeSS Hugeにおいてもプロダクトオーナーがこれらの役割を担うことが開発を円滑に進めるために重要である．しかし，これらすべての役割を一人が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ーという新たな役割を導入して，プロダクトオーナーの権限を委譲している．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開発方法となっている．

3.3　アジャイル開発の定量的マネジメント

3.3.1　ストーリーポイントによる作業量の定量化

スクラムでは作業量の数値化に任意の作業量を基準とした相対ポイントであるストーリーポイントを用いることが多い．プロダクトバックログで管理される各プロダクトバックログアイテムに各チームがリファインメント時にストーリーポイントを割り当てる．ストーリーポイントを用いた作業量の数値化は特定の技術者が予定作業時間を推測する方法と比べ実作業時間との乖離が少なく精度が高いという評価が示されている[11]．これはストーリーポイントが任意のチームが作業をした場合の相対ポイントを基準に各チームがリファインメントにおいて作業量を定量評価する方法となっていることが要因と考えられる．

3.3.2　ベロシティの計測

ベロシティとはチームの開発速度である．アジャイル開発ではチームが1回のスプリント（イテレーション）で完了したストーリーポイントの合計をベロシティとして用いている[12]． LeSS，LeSS Hugeでは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を測定する方法が一般に適用されている．しかし，測定したベロシティをチーム間で比較をすることは難しい．なぜなら，ポイントの基準となる開発内容の調整やチーム内での開発の熟練度，チームのメンバ数などに影響されるためである．そのため，あるスプリントにおいて，他のチームより完了したストーリーポイント数が多いからといって，そのチームがほかより生産性が高いとは言えない．

4．マルチプロダクトアジャイル開発方法

4.1　エンタープライズアジャイル開発におけるプロダクトオーナー

本稿で提案するマルチプロダクトアジャイル開発方法を**図3**に示す．

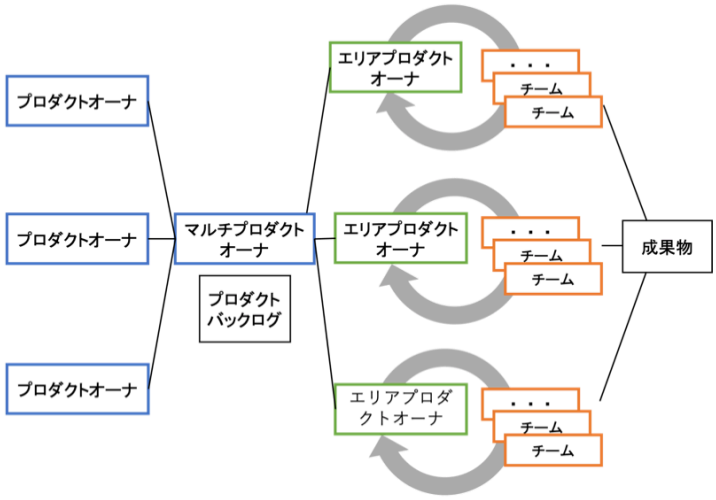

図3　マルチプロダクトアジャイル開発方法のモデル

マルチプロダクトアジャイル開発方法は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ーの役割の追加の2点を拡張した．図3はLeSS Hugeをベースにしマルチプロダクトオーナーの役割の追加をした開発のモデルを示す．

4.2　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においては，扱うプロダクトすべ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調整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1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．この点についてはアジャイル開発を行う組織用に開発されたJIRA Software[1]を利用しプロダクトバックログを作成することで継続的に整理することを可能とした．

図4に作成するプロダクトバックログとその管理との関係を示す．プロダクト，チーム，案件，技術領域などさまざまな粒度で管理できるように設定することで，登録されている数多くのプロダクトバックログアイテムを効率よく管理できる．マルチプロダクトオーナーは全体の状況を確認するため

に利用し，エリアプロダクトオーナーは担当のエリアで開発しているフィーチャおよび案件について確認をする．そしてプロダクトオーナーは自身が担当するプロダクトについて確認を行う．これらはJIRA Softwareの機能であるフィルタやラベル付けの機能を用いることで実現した．

図4　複数のプロダクトを管理するプロダクトバックログ

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

マルチプロダクトアジャイル開発方法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．マルチプロダクトオーナー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．

4.4　プロダクトオーナー（Product Owner）の役割

プロダクトオーナー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責任はプロダクトオーナーにある．そのためプロダクトオーナー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．

マルチプロダクトアジャイル開発方法においてはプロダクトオーナーとマルチプロダクトオーナーが議論することで組織とプロダクトの状況に応じて全体最適化を図る．

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

エリアプロダクトオーナーは担当する機能の開発に責任を持つ．エリアは複数の開発チームから組織され，エリアが担当する機能の開発をエリアプロダクトオーナーが管理する．

マルチプロダクトアジャイル開発方法において，エリアプロダクトオーナーは担当するエリアが開発しているプロダクトをプロダクトオーナーと開発状況について随時共有する．組織全体の状況もマルチプロダクトオーナーと連携することで，担当するエリアのプロダクトオーナーが全体最適されたプロダクトとして開発に集中できるよう管理する．

4.6　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5**に木曜日に始まり，水曜日に終わる1週間のスプリントを実施する例を示す．

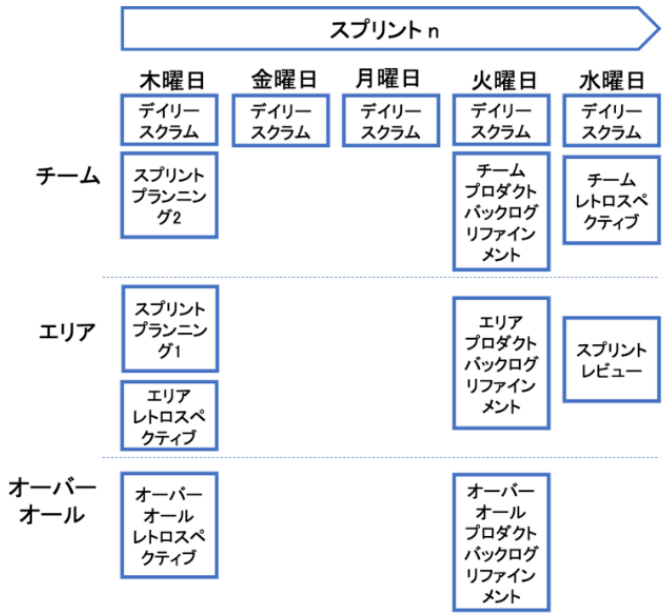

図5　スプリント

各チームのセレモニー実施時間はチームにより個別に設定することも可能である．リファインメントについては**表2**に示す通りとなる．

表2　マルチプロダクトアジャイル開発方法におけるリファインメント

セレモニ	参加者
・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	・マルチプロダクトオーナ ・エリアプロダクトオーナ
・エリアプロダクトバックログリファインメント	・エリアプロダクトオーナ ・チーム代表
・チームプロダクトバックログリファインメント	・スクラムチームメンバ

図5においてはまとまった開発作業時間を確保するために，月曜日，金曜日にはセレモニーの時間を短くして，開発効率を下げない工夫を行っている．

4.7　マルチプロダクトオーナー，プロダクトオーナー，エリアプロダクトオーナー間のコミュニケーション

マルチプロダクトアジャイル開発方法ではマルチプロダクトオーナー，プロダクトオーナー，エリアプロダクトオーナーのコミュニケーションを円滑に進めることが鍵になる．

マルチプロダクトオーナーはプロダクトの開発責任を負ってはいないが，プロダクトオーナーと連携し，組織および外部環境の状況や変化に対応して優先度を把握し，調整する．その優先度に基づき，マルチプロダクトオーナーはプロダクトバックログの開発を調整する．詳細な開発計画はエリアプロダクトオーナーと連携して調整する．このとき，エリアプロダクトオーナーが担当するフィーチャは優先度に応じて変化する．そのため随時プロダクトオーナーとも直接コミュニケーションをする必要がある．当面の予定が決まっているエリアにおいては直接プロダクトオーナーと連携することも必要となる．マルチプロダクトオーナー，プロダクトオーナー，エリアプロダクトオーナーは相互に密に連携することでプロダクトバックログを変化に応じて継続的に調整する．

4.8　担当するプロダクトとエリアのパターン分類による柔軟な開発の実現

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1つのプロダクトバックログで管理している．そのため各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が担当するプロダクト，フィーチャに変更が発生する．このとき，各スプリントにおいて各エリアと担当するプロダクトとの関係を次の3つのパターンに分類して開発を進めた．

（1）パターン1: 単一プロダクトの優先開発

図6にパターン1の構造を示す．図に示すように，任意のスプリントにおいてはプロダクト Aのフィーチャの開発に集中し，すべてのチームがプロダクト 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開発する．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．プロダクト B，Cのように開発を担当するエリアが割り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プロダクトオーナー，マルチプロダクトオーナー，エリアプロダクトオーナーを中心に準備を進める．

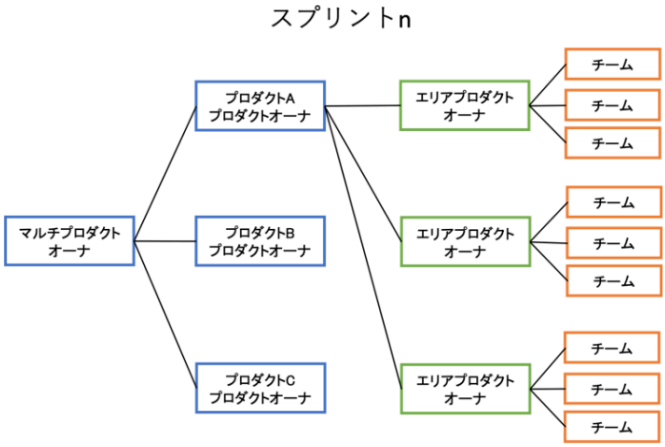

図6　パターン1

（2）パターン2: 一部の複数プロダクトの優先開発

図7にパターン2の構造を示す．図に示すように，任意のスプリントにおいてはプロダクト A，プロダクト Bの開発に集中しており各エリアのエリアプロダクトオーナーは担当するフィーチャのプロダクトオーナーと連携し開発を進める．パターン1と同様に開発を担当するエリアが割り当てられていないプロダクトCにおいてはリファインメントを進める．

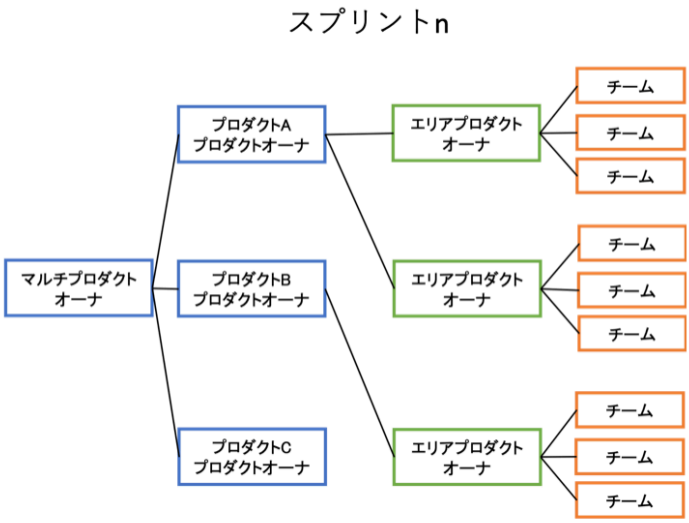

図7　パターン2

（3）パターン3: 全プロダクトの開発並行

図8にパターン3の構造を示す．図に示す任意のスプリントにおいては各プロダクトにそれぞれエリアが割り当てられ，担当するフィーチャを並行して開発できる．

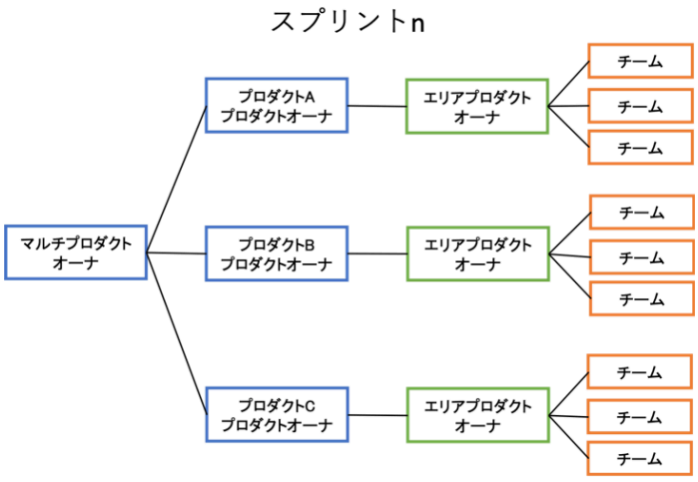

図8　パターン3

5. 提案開発方法の実践

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実践した経験を報告する．

5.1　アジャイル開発への移行

5.1.1 移行前の開発形態

提案開発方法を実践する以前は各プロダクト内で案件ごとにプロジェクト管理を行う開発方法を採用しており，スクラムやLeSSの体制を構築していなかった．移行前の組織構造を**図9**に示す．プロダクトごとにプロダクトの責任者（プロダクトオーナー）がおり，担当する技術領域ごとに開発チームが構成されていた．

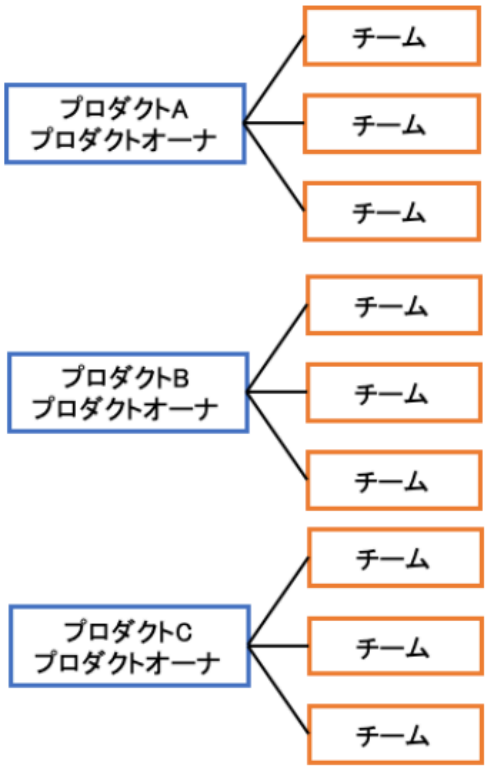

図9　移行前の体制

また，この開発形態での開発プロセスを**図10**に示す．

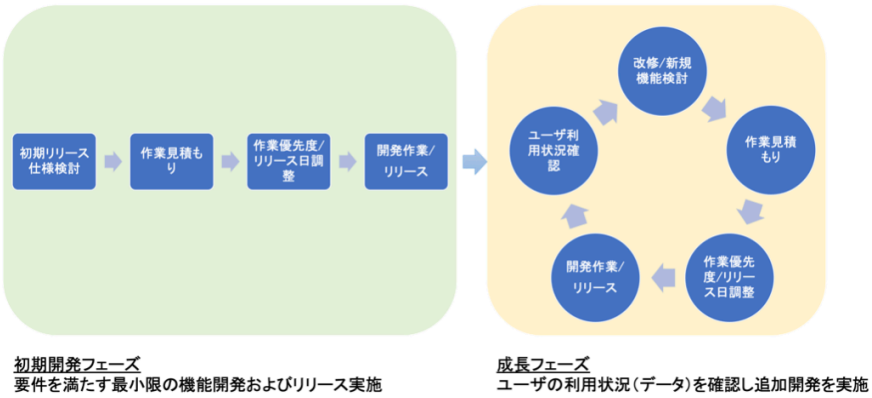

図10　移行前の開発プロセス

初期開発フェーズとして最小限の機能開発とリリースを実施し，初回の機能リリース後にはユーザの利用状況をデータで確認しつつ追加で必要な機能および改修を繰り返すこと（成長フェーズ）でプロダクトを成長させることを重視していた．

開発期間については仮設定したリリース日から逆算し，現在のリソースと要求を満たす最小限の機能開発を優先度付けすることで最終的に開発期間が設定される．スクラムのような一定のスプリント期間を設けてはいないが，優先度を設定し短期間で小さくリリースを繰り返す．この方法を用いることで各アプリケーションに対して平均して毎月2〜3回のリリースできる状態であった．

開発チームの作業はGitHubが提供するIssue機能，Jira Software，ホワイトボード等に付箋紙を用いて作業管理をするチームもあり，各チームがそれぞれ適していると考える方法で開発を管理していた．チームの自主性を尊重した自由な開発ができる一方で，組織全体の視点からは開発進捗状況の把握が難しいという課題があった．

5.1.2　Lessによるアジャイル開発への移行

2017年10月から2カ月ほどの期間を開発体制移行期間とし期間中にプロダクトバックログの準備や組織全体への説明を進めた．組織への説明を含め開発体制移行はLeSSによる開発経験があるメンバを中心に進めた．体制の移行には大きな問題はなかった．その理由としては5.1.1項で述べたように移行前にアジャイル開発を取り入れていたことにあると考える．短期間で小さくリリースを積み重ねるというアジャイル開発方法への理解が浸透していたことは新しい開発体制の理解を助けた．

5.1.3　開発対象プロダクト群

開発対象のスマートフォン向けアプリ「Yahoo! Map」，「Yahoo!カーナビ」とブラウザ向けサービス「Yahoo!地図」は地図を構築するデータの一部を共有している．しかし，ユーザ層が異なることもあり提供している機能は大きく異なる（**表3**）．そのためソースコードも個別に管理され，それぞれ独立したプロダクトである．

表3　プロダクト詳細

プロダクト	提供機能
Yahoo! Map	地図機能，近隣の店舗検索機能，おすすめ店舗情報，任意の地点への徒歩案内機能
Yahoo!カーナビ	カーナビゲーション機能，駐車場情報，渋滞情報の地図表示，道路規制情報の地図面表示
Yahoo!地図	PC ブラウザでの地図利用，スマートフォンブラウザでの地図利用

5.2　適用結果と評価

組織変更により変動もあったがチーム数は常時10チーム前後，各チームは5名前後であった．各チームでスクラムマスタの役割を担うメンバは1名であった．

以下，適用結果の評価を示す．

5.2.1　ベロシティによる開発速度と生産性の評価

提案開発方法の有効性をベロシティを用いて評価する．**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践した直近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週間である．

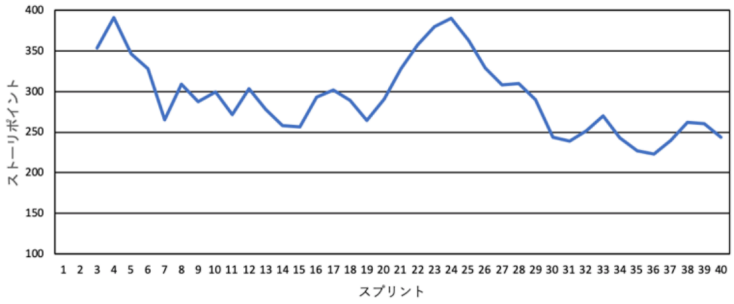

図11　直近3スプリントの移動平均ベロシティ

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尺度として，その影響を軽減した．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を調整し，認識のずれを軽減した．

ベロシティによって組織におけるスプリント単位での速度，すなわち，開発生産性を可視化できる．さらに，ベロシティとその変動，すなわち開発の加減速はスプリント単位を超えた中期的な開発とそのリリース配分の計画やリスク管理に有用であると言える．

5.2.2　組織変更による開発への影響への予測と対応

提案方法を適用した組織では毎年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（水）に終わるスプリントであった．スプリント16中にも実際に組織変更の通知があり，開発組織が再編された．

組織変更により開発チームのメンバ構成，チーム数，そしてマルチプロダクトオーナー，エリアプロダクトオーナーの変更があった．この期間については移行期間として開発業務より業務の引き継ぎに時間を充てた結果，この前後でベロシティは低くなっている． このことから，アジャイル開発の構造とベロシティというスプリント単位での評価尺度を用いることにより，組織変更時におけるスプリント単位でのベロシティ低下の予測が可能となり，組織内での変化への俊敏な対応が可能となる．

5.2.3　大規模開発期の生産性の評価

スプリント20からスプリント29の期間において直近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7と図8に示した体制を開発状況に応じて柔軟に変更した．メンバ数，チーム数に変更がなかったことからこの期間においての生産性は向上していたと考えられる．

5.2.4　開発の達成尺度の推移

組織全体の開発の達成度の尺度として，コミット率を式（1）で定義する．2017年11月30日（木）から2018年10月10日（水）の間のコミット率の推移を**図12**に示す．図12中には期間中のコミット率の回帰直線を破線で示す．

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

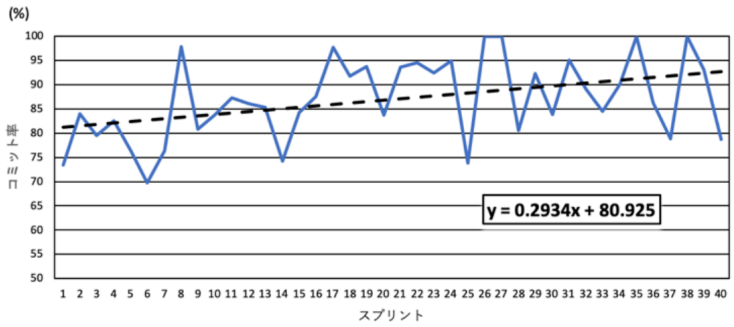

図12　コミット率

コミット率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合意した達成可能な目標であるため，移動平均とする必要はないと考えた．

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おおむね80%のコミット率を維持できている．最大値で100%となり，最小値のスプリント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では優先度が低く先送りとなっていた，いわゆるUndoneワーク[3]を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．このことから，Undoneワークを解消する周期は，開発規模やスプリント単位を超えた組織活動の変化などに応じて調整が必要であると言える．

6. 評価

6.1　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は1年間の実践において組織変更という変化に実際に対応できたことがベロシティ，コミット率の推移より明らかになった．このことから**図13**に示すように開発体制をスケールアウトすることが可能と考える．

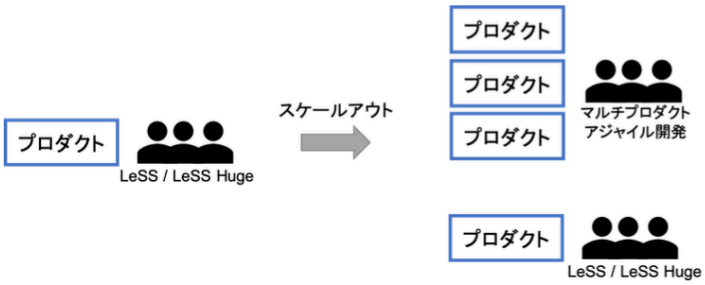

図13　スケールアウト

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するのが一般的である．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，図13に示すように，マルチプロダクトアジャイル開発方法を用いた開発組織とプロダクトの柔軟な配置ができることが必要と考える．

6.2　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．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，提案した開発方法は実践において有効であると評価できる．

7. 考察

7.1　マルチプロダクトオーナーとプロダクトオーナーの役割

提案方法ではLeSS Hugeにおけるプロダクトオーナーの役割の一部をマルチプロダクトオーナーへ権限委譲することで複数のプロダクトを並行して開発できる組織構造となっている．そのため，プロダクトオーナーの役割は本来のLeSS，LeSS Hugeで示される役割から変化している．プロダクトオーナーはエリアプロダクトオーナーとも連携はするが，提案方法ではマルチプロダクトオーナーと密に連携することが多くなった．そのためプロダクトオーナー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．このためマルチプロダクトオーナーおよびプロダクトオーナーの役割を再確認する必要がある．

7.2　フィーチャ間の依存関係による影響アプローチにおけるプロダクトオーナーの役割

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）を受け他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問題が起こり得る．この問題は，提案方法においても発生した．これは，LeSS Hugeなどのエンタープライズアジャイル開発の実践における共通問題であり，プロダクトオーナーが機能しないことによる問題ではないと考える．

対策としては，マルチプロダクトオーナー，プロダクトオーナー，エリアプロダクトオーナー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ーの分割を行うなど他のストーリーに着手することを準備する方法がある．

7.3　パターン切り替えと提案開発方法の適用可能条件

本事例においては3つのプロダクトを扱い任意のスプリントにおいて図7に示したように1つのプロダクトに集中して開発を進めるという方法をとった．経験したことがないプロダクトの開発にかかわるチームもあったが図12で示したコミット率の推移から長期的には組織全体として成果が出ていたことが分かる．短期的な視点で見るとパターン切り替え時には各チーム内で学習コストの増加による負荷はあったが，全プロダクトに共通する地図機能があることがこの学習コストの軽減に寄与したと考える．各エリアはさまざまなフィーチャを担当することを要求されるが，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いことから，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く，ドメイン知識が共有されていた．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担当した場合でも地図というドメイン知識があるため，学習コストが抑えられ，提案方法の実践に寄与していたと考えられる．

異なるドメインを持つプロダクトを扱う場合にも提案方法が有効かどうか検討する必要がある．

7.4　アジャイル開発組織のスケールアウト

表4にスクラム，LeSS，LeSS Huge，提案方法であるマルチプロダクトアジャイル開発方法についてチーム数と対象とするプロダクト数についてまとめた．提案方法の実践ではLeSS Hugeを拡張していることからチーム数は8以上とした．

表4　開発方法の比較

開発方法	チーム数	対象プロダクト数
スクラム	1	1
LeSS	2～8	1
LeSS Huge	8～	1
マルチプロダクト アジャイル開発方法	8～	2～

複数のプロダクトを1つの組織で一括してアジャイル開発を実践できるマルチプロダクトアジャイル開発方法という選択肢が提案できたことで，より柔軟にアジャイル開発組織の構築とスケールアウトが可能と言える．

8. 今後の課題

8.1　マルチプロダクトオーナーの役割の明確化とアジャイル開発組織のスケールアウト

マルチプロダクトオーナー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と実践の積み重ねによる検証を行う必要がある．マルチプロダクトオーナーは各プロダクトに対して責任を負うのではなく内部環境，外部環境などさまざまな要素を統合的に考えた上での優先度付けに責任

を負っている．プロダクトオーナーが担当する役割と明確に差別化した責務を明示し実践を積み重ねることで，提案方法がより機能すると考える．

8.2　マルチプロダクトオーナー，プロダクトオーナー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セレモニーを実施している．しかし，提案方法において追加しているマルチプロダクトオーナーの役割にはプロダクトオーナーとの連携方法を定義する必要がある．特に，マルチプロダクトオーナー，プロダクトオーナー間の連携は提案方法が機能するために重要であり，スプリントごとに同期可能なセレモニーを新たに定義し，実践を積み重ねる必要がある．

9. 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の開発に実践し，その効果を確認した．これにより，外部環境および内部環境の変化に俊敏に対応できる開発を実現した．提案方法によって，LeSS, LeSS Hugeのプロダクト中心の組織拡張によらず，組織の状況に応じた柔軟で俊敏なエンタープライズアジャイル開発が可能となる．

参考文献

1）Atlassian, Jira Software, https://ja.atlassian.com/software/jira (参照 2020-06-22).
2）Schlatmann, B.: 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 2020-06-22).
3）Larman, C. and Vodde, B.: Large-Scale Scrum More with LeSS，Addison-Wesley (2016).
4）Kim, G. et al.: The DevOps Handbook, IT Revolution Press (2016), 榊原　彰（監修）: The DevOps ハンドブック, 日経BP社 (2017).
5）Allspaw, J.: 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 2020-06-22).
6）Bass, J. M. and Haxby, A.: Tailoring Product Ownership in Large-Scale Agile Projects, IEEE Software, Vol.36, No.2, pp.58-63 (Mar./Apr. 2019).
7）Smed, J. et al.: DevOps: A Definition and Perceived Adoption Impediments, Proc. XP 2015, LNBIP Vol.212, Springer, pp.166-177 (May 2015).
8） Schwaber, K. and Sutherland, J.: The Scrum Guide (Nov. 2017), https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf (参照 2020-06-22).
9）川口恭伸, 他：楽天でのエンタープライズアジャイルとDevOps, デジタルプラクティス, Vol.7, No.3, pp.243-251 (July 2016).
10）貝瀬岳志, 他：スクラム実践入門, 技術評論社 (2015).
11）Williams, L. et al.: Scrum + Engineering Practices: Experiences of Three Microsoft Teams, Proc. of ESEM 2011, IEEE Computer Society, pp.463-471 (Sep. 2011).
12）Cohn, M.: 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）Kalenda, M. et al.: Scaling Agile in Large Organizations: Practices, Challenges, and Success Factors, J. of Software: Evolution and Practice, Vol.30, No.10, pp.1-24 (Oct. 2018).
14）Scrum Alliance, Scaling Works to Fit Your Needs, https://www.scrumalliance.org/get-certified/scaling (参照 2020-06-22).
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 Japan2017 (Apr. 2017), https://www.agilejapan.org/2017/img/session/document/A-5.pdf (参照 2020-06-22).
16）Dingsøyr, T. et al.: Agile Development at Scale: The Next Frontier, IEEE Software, Vol.36, No.2, pp.31-36 (Mar./Apr. 2019).
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 2020-06-22)

田中優之（正会員）masaytan@yahoo-corp.jp
ヤフー（株）にてスマホ向けナビゲーションアプリ開発を担当．2010年同志社大学工学部卒業．2017年より現職．2019年より南山大学大学院理工学研究科ソフトウェア工学専攻所属．大規模アジャイル開発，機械学習工学に関する研究に取り組んでいる．

青山幹雄（正会員）mikio.aoyama@nifty.com
1980年岡山大学大学院工学研究科修士課程修了．同年，富士通（株）入社．通信ソフトウェアの開発とその管理に従事．2001年より南山大学数理情報学部情報通信学科教授，2009年より情報理工学部ソフトウェア工学科教授．博士（工学）．Webソフトウェア，自動車組込みソフトウェアなどを対象として，要求工学，ソフトウェアアーキテクチャ，機械学習ソフトウェア工学に関する実践的課題の研究と教育に取り組んでいる．

投稿受付：2019年8月5日
採録決定：2020年6月16日
編集担当：倉林利行（NTT）

本論文は特集「DXを推進する俊敏なシステム開発・運用―アジャイルにつなぐビジネスとICT―」への投稿論文です．