フィーチャーチーム入門

by Craig Larman and Bas Vodde

Version 1.1

フィーチャーチームと要求領域は、リーン＆アジャイル開発を大規模開発で実践するためにとても重要です。この２つに関しては、"Scaling Lean & Agile Development" のフィーチャーチームと要求領域の章で、大規模で実践する際のスクラムの考え方と組織的なツール等を徹底的に分析しています。このフィーチャーチーム入門には、いくつかの重要な考え方を集約しました。また、より詳細な内容は "Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum" に記載しました。

フィーチャーチーム概論

図１に示したフィーチャーチームは、”組織の既存の枠やコンポーネント等にとらわれず、多くの顧客のフィーチャーを１つずつ完成させる長寿命のチーム”（1）です。

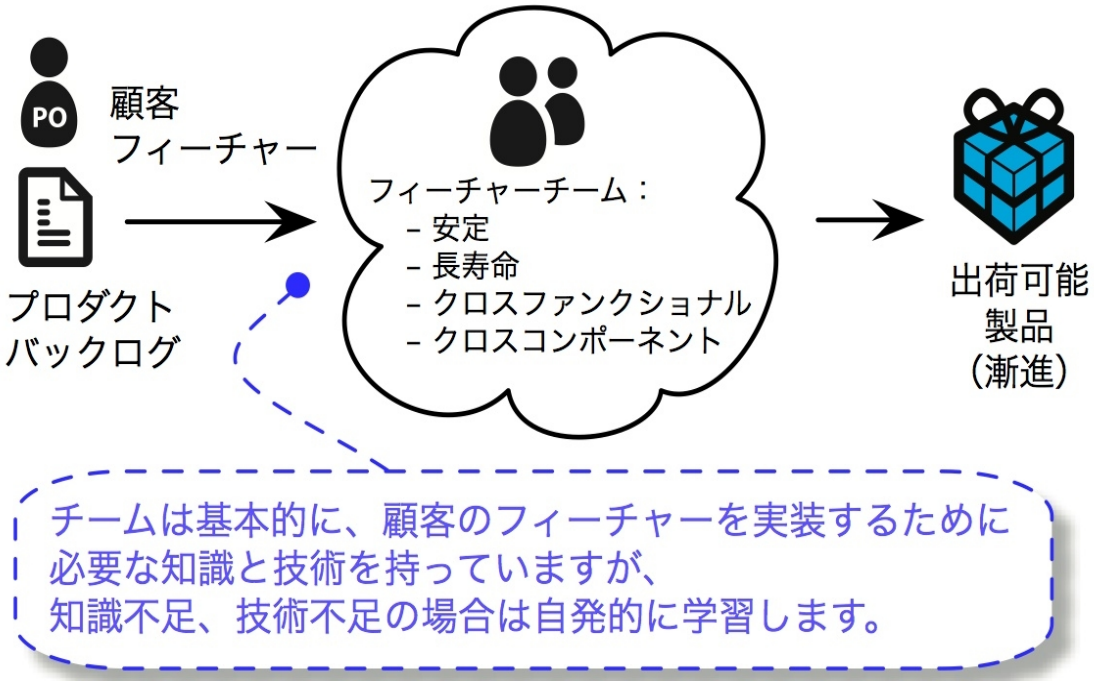


図１ フィーチャーチーム

(1). フィーチャーチームは長期に渡り、常に協働し、共に多くのフィーチャーを実装します。

フィーチャーチームの特徴は以下の通りです。

Feature Team

- チームは長寿命で、より高い性能を "具現化" すべく、常に協働する
  彼らは時間と共に成長し、新機能を提供する
- 組織の既存の枠やシステムのコンポーネント等にとらわれない
- 常に共に働く
- 全てのコンポーネントと全ての領域（解析、プログラミング、テスト等）
  を横断し、顧客中心のフィーチャーを実装するように働く
- （チーム内で）専門作業を平準化するよう、自発的に構成される
- スクラムの場合、通常７±２名で構成される

フィーチャーチームを採用する場合、最新のエンジニアリング技法（特に継続的インテグレーション）を用いることが不可欠です。継続的インテグレーションは、コードの共同所有を容易にし、複数のチームが同時に同じコンポーネントを実装するためには必要不可欠です。

”フィーチャーチームの全てのメンバーが、システム全体を把握する必要がある” という一般的な誤解がありますが、その必要はありません。なぜなら

- 全ての顧客のフィーチャーを実装するために、個人ではなくチーム全体でテスト、設計、プログラミング等の実用的な技術とコンポーネントに関する知識等が必要であり、多種多様な分野で協働する事が望ましい。しかし、フィーチャーチーム内でも、個々のメンバーは特定分野を専門にしますが、複数の分野を専門にすることが望まれます。
- フィーチャーは、やみくもにフィーチャーチームに配分されません。どのチームが、どのフィーチャーに着手するのかは、チームの現在の知識と技術により決定します。

フィーチャーチームは専門作業が開発の制約になった場合、専門作業への自発的学習を始めます。

フィーチャーチームは要件計画とほぼ同期間で
チームの技術力を向上させるために専門作業を有効に活用する

しかし、現在のチームの技術では要件を計画通りに達成できない場合
"強制された" 組織の既存の枠を超える事を学習する

フィーチャーチームは、専門性と適応性の適度なバランスをとる

表１と図２は、フィーチャーチームと従来のコンポーネントチームとの差異を示します。

表１ フィーチャーチーム VS コンポーネントチーム

feature team	component team
顧客とって最良で価値あるもの を提供するためにチームを最適化 (a)	最良のコード(の量)を提供する ためにチームを最適化
高価値のフィーチャーと システムの生産性に集中 （価値のスループット）	"簡単な" 低価値のフィーチャーを 実装する事による個々の 生産性増加に集中
顧客中心のフィーチャーを 実装することに責任を持つ	顧客中心のフィーチャーの 一部にのみ責任を持つ
"最新" の方法で組織化されたチーム (b) - コンウェイ法則を避ける	従来の方法で組織化されたチーム コンウェイ法則に従う (c)
顧客中心、作業等の可視化 中小規模の組織	"定着した" 仕事を継続し、成長に 非常に長い時間を必要とする組織
適応性向上のため チーム間の依存を最小限にする	チーム間に依存関係があるため 余分な計画が必要
複数の専門作業に集中	１つの専門作業に集中
プロダクトコードを共同所有	コードは、チームの個人が所有
責任はチームに帰属	責任は、明確な個人に帰属
反復型開発	"ウォーターフォール" 開発
適応性を活かし 継続して幅広い学習を行う	既存の専門知識を活かすが 新技術の学習に疎い
熟練したエンジニアリングの慣習 を必要とし、効果が広く現れる	ずさんなエンジニアリングの慣習 で作業し、効果は局所的
維持管理し易いコードとテスト を提供	信念とは異なり、しばしば低品質 なコードで作られたコンポーネントを提供
実践するのが困難	実践するのが容易

(a) 異なる観点で最適化されたフィーチャーチームは、しばしば局所視点から遅く感じられます。
(b) フィーチャーチームは、比較的 "最新" の大規模開発で長期の実績があります。Microsoft や Ericsson がその好例です。
(c) Mel Conway は、1968 年にこの望ましくない構造に気付き、推薦しませんでした。実際は逆に、反対していました。
(d) 計画が追加される度に行われるリリース会議、もしくはリリースまでの経緯に、より多くの管理コストがかかる事は明らかです。

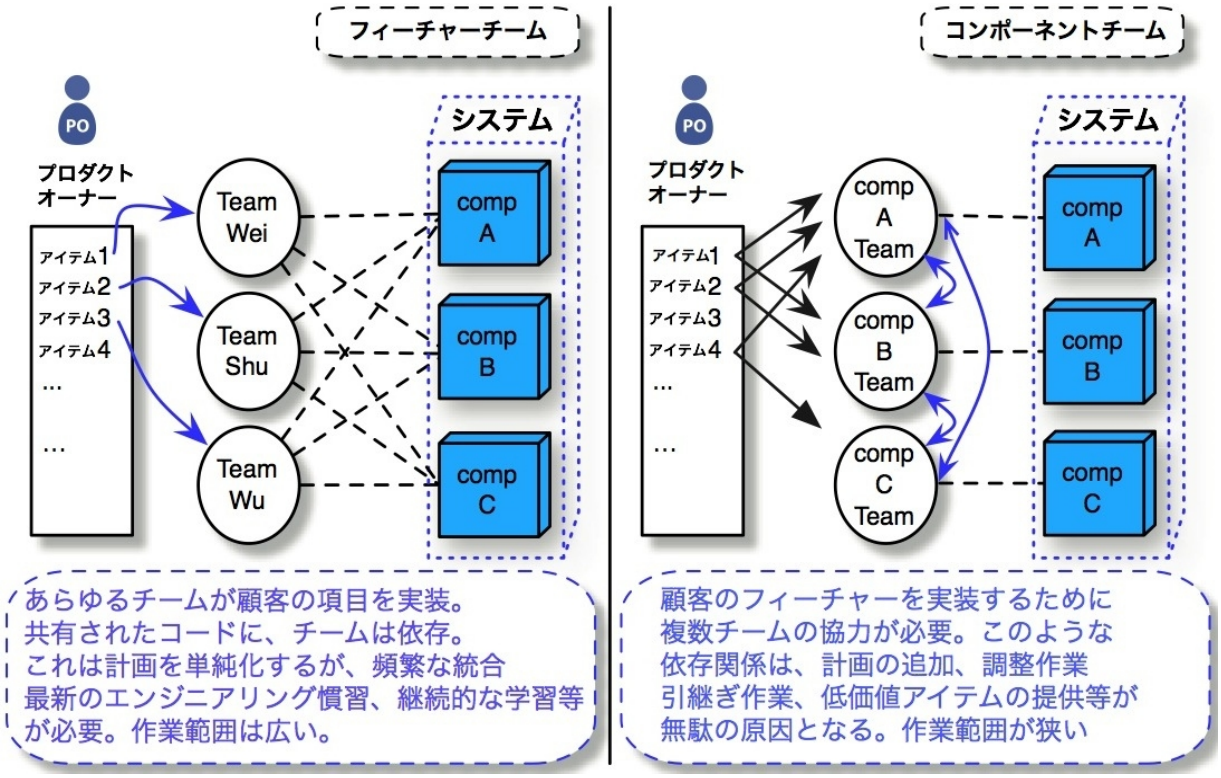


図2 フィーチャーチーム VS コンポーネントチーム

以下の表は、フィーチャーチームと従来のプロジェクトもしくはフィーチャーグループの差異を示します。

表2 フィーチャーチーム VS フィーチャープロジェクト

feature team	feature group or feature project
長期協働し、多くのフィーチャーを実装する安定したチーム	1つのフィーチャーもしくはプロジェクトのために集められた一時的な集団
全ての仕事に関する責任はチームに帰属	専門作業に基づき部分的に責任が分割され "個人" に帰属
自己管理するチーム	プロジェクト管理者が制御
シンプルな組織（マトリクスはない！）	リソースとマトリクス化された組織
チームメンバーは献身的で100%チームの仕事に集中する	メンバーは専門作業しかしないため多くのプロジェクトでパートタイム制

コンポーネントチームの多くの欠点を Scaling Lean & Agile Development の "フィーチャーチーム" の章にまとめました。図3に、これらを示します。

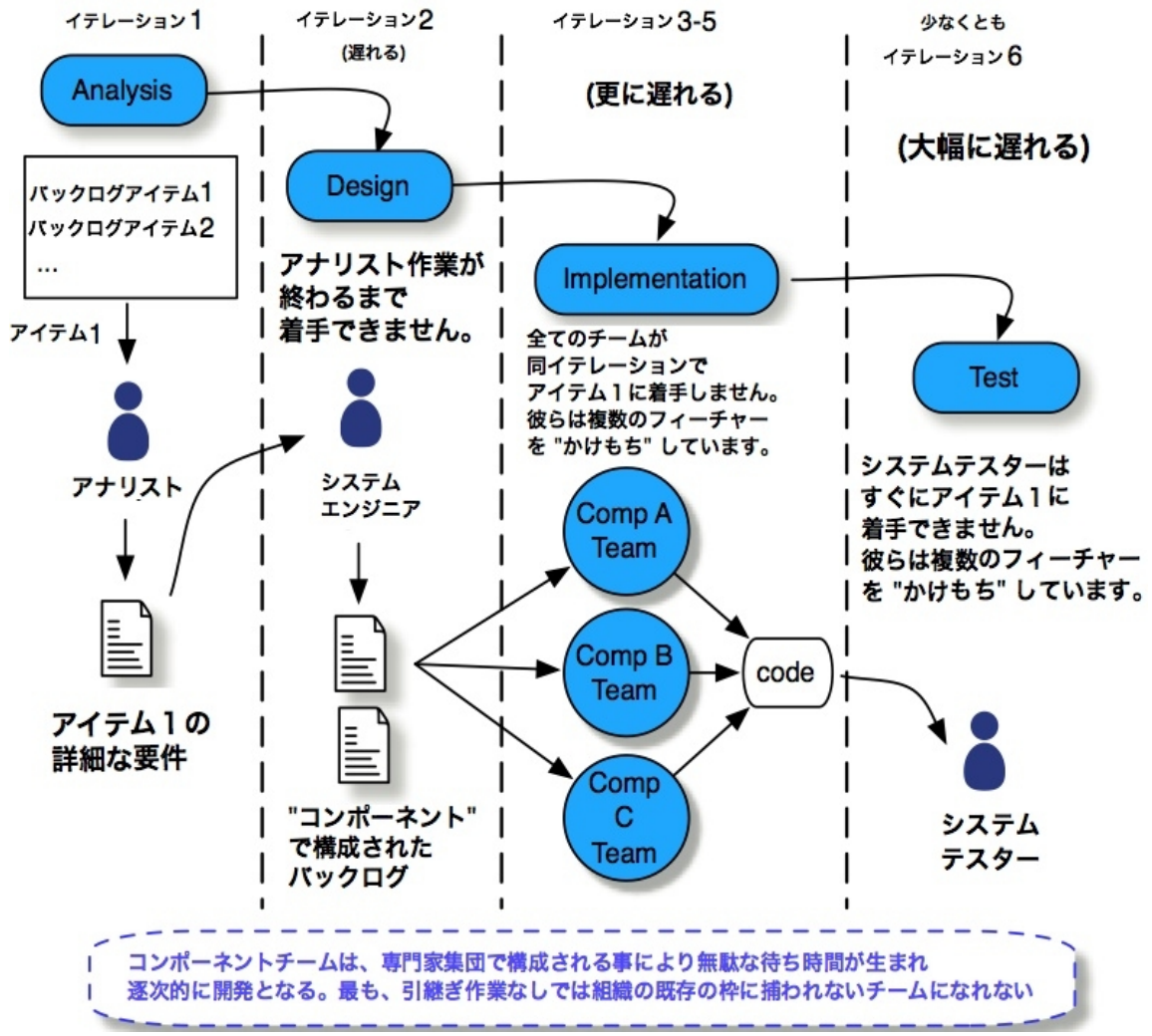


図３ コンポーネントチームの欠点

コンポーネントチームでは作業量に差があり、WIP（Work In Progress）の高レベル、多くの引き継ぎ作業を要し、掛け持ち作業も増加する。これにより部分的な作業配分等になり、ウォーターフォールやＶモデルの様な逐次的開発になる場合がある。

コンポーネントチームか、フィーチャーチームか？

純粋なフィーチャーチームは、価値の提供と組織への適応性の観点から理想的です。価値と適応性は組織の唯一の評価基準ではありませんが、多くの組織は、コンポーネントチームからフィーチャーチームに移行する間は、平行運用します。（注意：ハイブリッドモデルは、両欠点を保有します。）

多くの企業がこの方法を採用したハイブリッド組織になった理由はコンポーネントチームが従来から継続的に行ってきた作業（インフラ構築、再利用可能なコンポーネント構築、もしくは、コードの整理

等）の調整が必要だからです。しかし、コンポーネントチームを作らず、純粋なフィーチャーチームだけで、この調整作業を行うことは可能です。どのように行えばよいのでしょうか。それは、顧客のフィーチャーに関連する "インフラ構築、再利用可能なコンポーネント構築、もしくは、コードの整理等" があれば加え、プロダクトバックログにも追加します。フィーチャーチームは当面（プロダクトオーナーが望む期間）上記のような仕事を行い、終えたら顧客中心のフィーチャーの実装に戻ります。

フィーチャーチームへ移行

コンポーネントチームからフィーチャーチームへ移行する時、各企業ごとに戦略や計画が必要になります。私達は多くの経験から様々な戦略を持っていますが、異なる前提条件や異なる背景でそれを共有しても失敗します。時間はかかりますが比較的安全な戦略は、既存のコンポーネントチームの中から、１つのフィーチャーチームを設けることです。このチームが活躍した後に、第２のフィーチャーチームを設けます。これは、企業が満足する速度で漸進します。これを図４に示します。

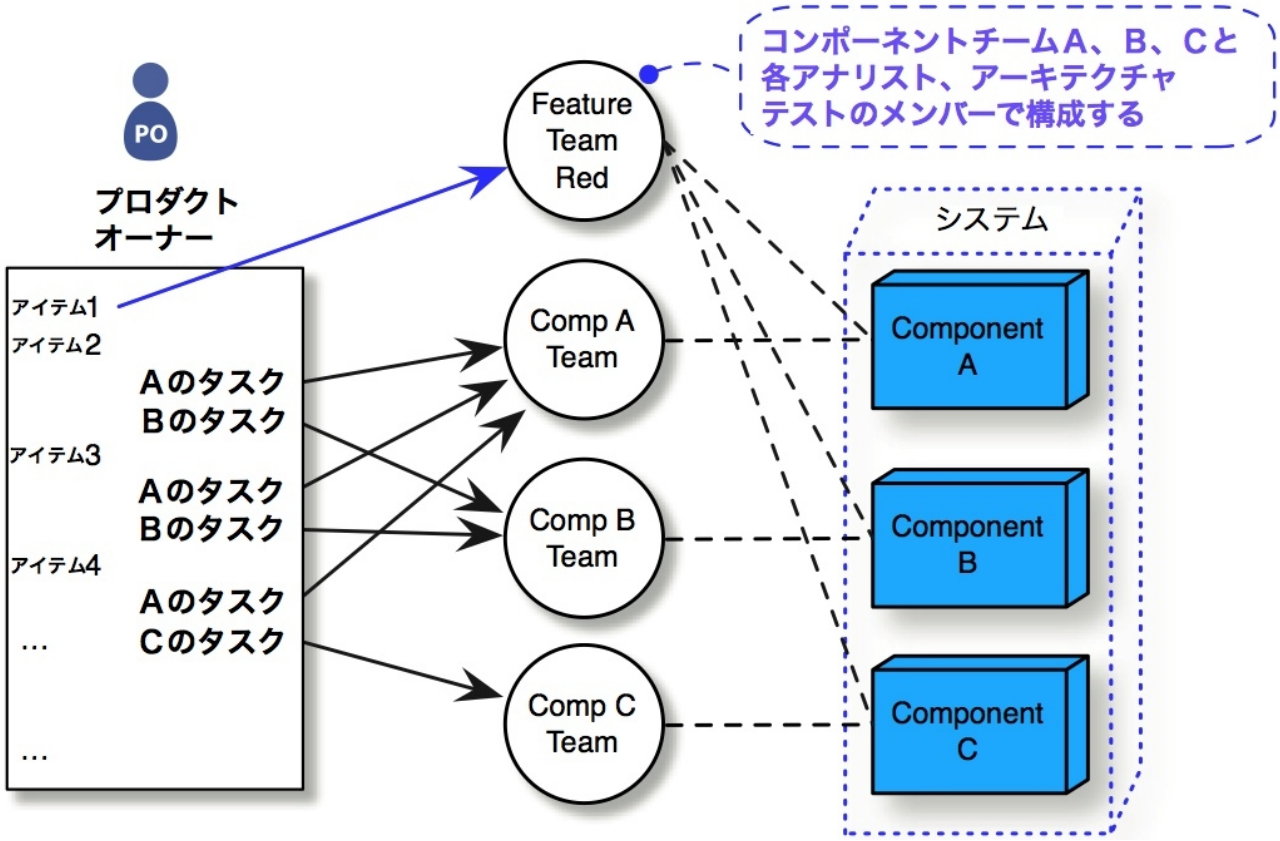


図４ コンポーネントチームからフィーチャーチームへの段階的移行

要件領域概論

フィーチャーチームは上手く規模拡大しますが、１０チームを超え、１００人以上になった場合、組織の状態に合わせて構造を修正する必要があります。要件領域に適切に対応するためには構造修正が必要となります。フィーチャーチームとは別のコンセプトを実現します。この要件を要件領域と呼び、プロダクトバックログとは異なる要件を管理します。

プロダクトオーナーは、要件領域内の特定の要件に関する全てのプロダクトバックログ項目を集約します。次に、プロダクトオーナーは様々な観点から、総合的なプロダクトバックログ（エリアバックログ）へ意見を追加します。エリアバックログは、顧客視点から製品価値を把握できる専門家である "エリアプロダクトオーナー" によって優先順位付けされます。各要求領域は、各エリアバックログに対応するフィーチャーチームがいくつかあります。例を図５に示します。

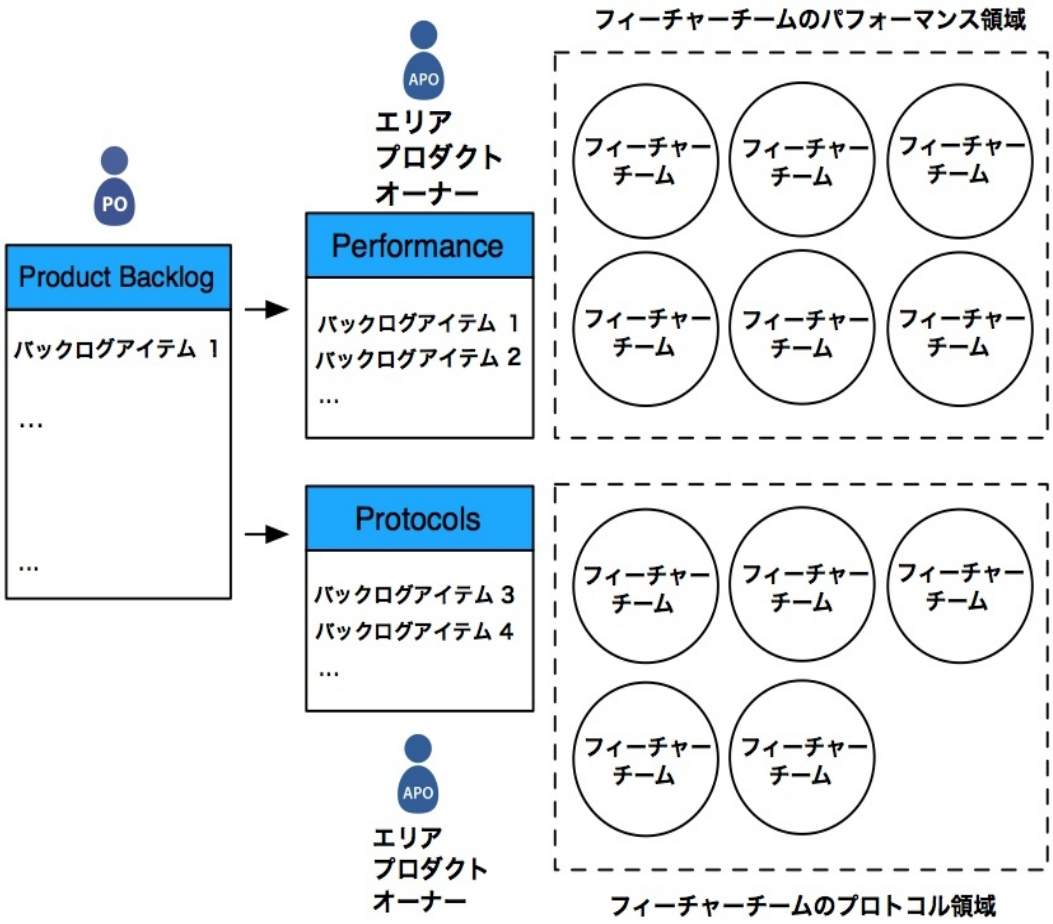


図５ 要求領域

要求領域は、フィーチャーチームを大規模に適応したものです。製品構造に従い、チームを構造化することによって大規模に適応します。これは開発領域と言われます。表３に要求領域と開発領域の差異を示します。

表３ 要求領域 VS 開発領域

Requirement Area	Development Area
顧客観点の要件が中心	製品アーキテクチャが中心
サブシステムのコード所有権はない	サブシステム毎にコード所有権がある
製品のライフサイクルには 必ず変化があるが、それは一時的で 常に変化する必要はない	製品のライフサイクルを終えても 修正する傾向がある
顧客の言葉を用いて 顧客観点で協働する	技術者の言葉を用いて アーキテクチャの観点で協働する

最終的に、エリアプロダクトオーナーはプロダクトオーナーをサポートしながら、全体を調整するよう働きます。エリアプロダクトオーナーは、プロダクトオーナーとは異なる事柄に焦点を合わせ、異なる責任を持ちます。故に、特定の１プロダクトではなく複数のチーム（少なくとも４チーム以上）と共に協働します。こうすることにより、１プロダクトチームによる組織の部分最適化を避けます。

結論

フィーチャーチームは、完全な顧客のフィーチャーを実装する安定したチームです。フィーチャーチームは、コンポーネントチームが引き起こした部分最適化と無駄なオーバーヘッドを解決します。しかし、フィーチャーチームは自ら知識、技術等の探求へ挑戦し続けなければなりません。

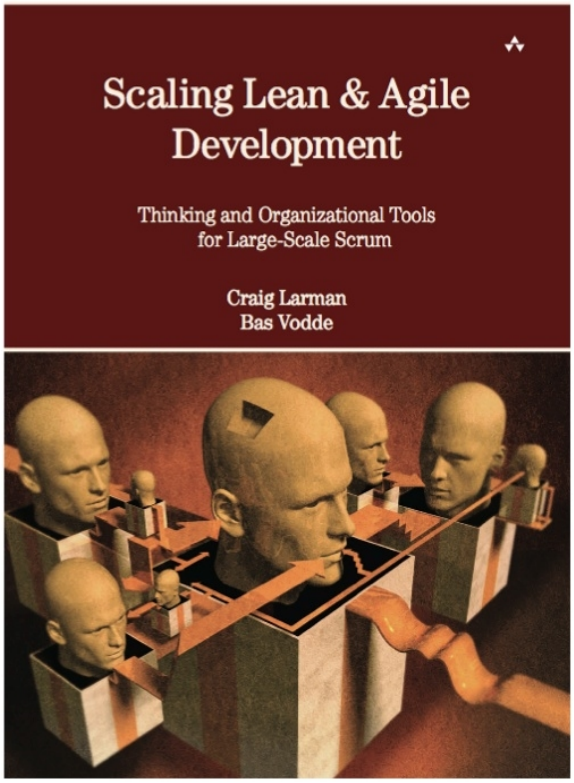
要件領域は、総合的な顧客のフィーチャーを用いてエリアプロダクトバックログを作ることで、フィーチャーチームのコンセプトを拡大します。その結果、フィーチャーチームは、どのような規模にも拡大しても（組織等を）構造化できます。

翻訳者より

フィーチャー入門をお読み頂き、ありがとうございます。ご意見、フィードバック、お気付きの点等がございましたら、私（Kazumasa EBATA : ebacky）ebacky AT odd-e.com までご連絡下さい。また、日本には “すくすくスクラム (http://www.sukusuku-scrum.jp/)” というスクラムのコミュニティがあります。よろしければ、一度、ご参加下さい。

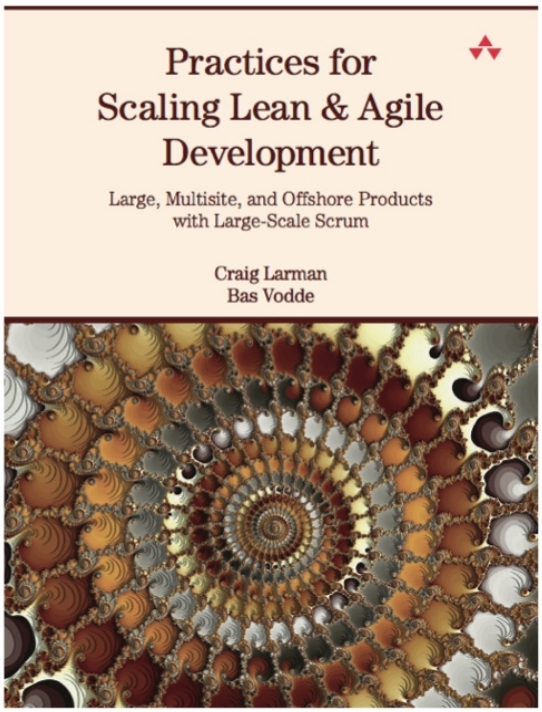
最後に、本資料が何かの形で、皆様のお役に立てれば幸いです。

参考文献



Chapters:

- Introduction
- Systems Thinking
- Lean
- Queueing Theory
- False Dichotomies
- Be Agile
- Featurea Teams
- Teams
- Requirement Areas
- Organization
- Large-Scale Scrum



Chapters:

- Large-Scale Scrum
- Test
- Product Management
- Planning
- Coordination
- Requirements
- Design
- Legacy Code
- Continuous Integration
- Inspect & Adapt
- Multisite
- Offshort
- Contracts