スクラムの拡張による組織づくり

粕谷大輔
Kasuya Daisuke

複数のスクラムチームを Scrum@Scaleで運用する

技術評論社

本書に関する補足情報や正誤情報は、以下の本書サポートページを参照してください。

https://gihyo.jp/book/2023/978-4-297-13661-1

はじめに

筆者がアジャイルソフトウェア開発に興味を持ちはじめた当初、日本のソフトウェア開発の現場でアジャイルな開発をしている会社を探すのはとてもたいへんでした。しかし、転職活動の際に幸運に恵まれ、2012年にXPのプラクティスで開発をしているチームに入ることができました。モバイルゲームを開発する5人ほどのチームでした。これが筆者のアジャイルソフトウェア開発経験のスタートです。

そこから10年近くが経ち、多くの企業でアジャイルソフトウェア開発が試みられるようになってきました。アジャイルを題材にしたカンファレンスには毎回何百人もの人が訪れ、コミュニティも活況です。現在のアジャイル開発の現場ではスクラムが多く用いられており、転職活動の際にスクラムを採用している企業を探すのは、以前ほどは難しくなくなりました。

さまざまな現場で扱われるようになると、多様なニーズが生まれます。そのニーズの中には、「より大規模な開発をスクラムでやるにはどうすればよいか」というものがありました。これは人々にとってとても大きな関心事のようで、「大規模スクラム」をやるための数多くの手法が編み出されます。LeSS、Nexusなどです。日本のアジャイルコミュニティではLeSSの事例を多く見かけますし、日本語書籍も出版されています。

これらの「大規模スクラム」の手法はそれぞれに特徴を持っており、LeSSの導入事例が多いからといってそれが自分たちの現場にマッチするとは限りません。世の中にはさまざまなチームがあり事情はさまざまなため、選択肢はいくつかあったほうがよいでしょう。

本書ではそのような「大規模スクラム」の手法の一つである、Scrum@Scaleを中心に取り上げます。

本書では、はじめにスクラムをスケーリングする意義と難しさを考えていきます。これはスクラムやソフトウェア開発に限った話ではありません。「規模の大きなもの」を扱うことは本質的にとても難しい仕事です。第1章では、スクラムをスケーリングすることを通じて、そのような難しさを考えます。

第2章では、スクラムの用語を復習できるようにしました。スクラムに関する十分な知識を持っている方は読み飛ばしても問題ありません。本書はScrum@Scaleを中心に大規模なスクラムの解説をしていきます。Scrum@Scaleの解説では当たり前のようにスクラムに関する用語が頻出します。そのため、本書ではじめてスクラムに触れる方やスクラムを復習したい方は、第2章を参考にしてください。

続いてScrum@Scaleの解説に入っていきます。第3章ではイメージが湧きやすいように、ちょっとしたストーリー仕立てでScrum@Scaleによる仕事の流れを説明します。

その後、第4章と第5章で、第3章のストーリーの中に登場した用語などを詳しく解説していきます。これらの章の解説は公式ガイドを手がかりとしているため、本書を読むだけでもScrum@Scaleを理解できます。しかし、学習の際には原典にあたるのがとても重要です。公式ガイドを未読の人はぜひ本書を読んだあとに公式ガイドにも目を通してください。

理屈だけを説明されてもなかなか実践には結び付きにくいものです。第6章ではScrum@Scaleを導入する手がかりとして、導入順序の例を紹介します。

最後の第7章では、筆者が実際に所属しているチームでのScrum@Scaleの取り組みを紹介します。これによって、現場での実践方法がイメージしやすくなるのを期待しています。

本書では、公式ガイドだけではわかりにくい、より実践的な考え方などを加えることで、みなさんの現場へ適用する手助けになりたいと考えています。

最後に、本書はたくさんの人のご協力によって完成しました。

吉羽龍太郎さん、大友聡之さん、和田圭介さん、児山直人さん、湯川正洋さん、遠藤良さん、dairappaさん、山根英次さん、中村洋さん。これらの皆さんは本書全体のレビューをしてくださいました。

石毛琴恵さん、増田謙太郎さんは、第3章の架空のチームの活動に対して、ゲーム開発の現場としてよりリアリティが出るようなアドバイスをいただきました。

北濱良章さん、tunemageさんは、本書の骨子として最初に書き終えた時点の第4章の感想をくださり、その後の執筆継続の励みになりました。

藤井善隆さんには同僚として第7章のレビューをしていただきました。

執筆に疲れたときには、open air 湊山醸造所のビールはとても良いリフレッシュになりました。

編集者の池田大樹さんは企画書の作成からすべての工程を伴走してくださり、この人がいなければ本書を作り切ることはできませんでした。

執筆期間中は妻の支えによって、良い環境を維持できました。

皆さん本当にどうもありがとうございました。

2023年7月

粕谷 大輔

目次

スクラムの拡張による組織づくり

複数のスクラムチームをScrum@Scaleで運用する

第6章

現場へどのように導入していくか 121

第7章

Scrum@Scaleで運用される現場
チャットサービスの開発現場の場合 137

第 1 章

スクラムのスケーリングと大規模の難しさ

本書では、Scrum@Scaleというフレームワークを軸に、スクラムをスケーリングするためのやり方や考え方を説明します。具体的な内容へ入る前に、スクラムのスケールとはどのようなものなのかを考えてみます。また、Scrum@Scale以外のスケーリングの手法もいくつか簡単に紹介します。

スクラムをスケールするとはどういうことか

スクラムは本来、10人以下で形成される1つの職能横断チームの動き方をガイドするために考えられた、軽量なフレームワークです。スクラムの定義が書かれている「スクラムガイド」[注1]には、複数チーム間の連携に関する説明はありません。スクラムを採用しているチームが2つ以上あり、それらが相互に協調する必要があるときには、その協調のためのしくみが別途必要です。そのため、複数のスクラムチームがどのように協調すればよいか、いろいろなやり方が考え出されてきました。本書で紹介するScrum@Scaleもその一つです。

では、スクラムを複数チームで採用し、それらが協調したい場合とはどのような場合でしょうか。大きく分けて「1つのスクラムチームから増やしていく場合」と「大規模な組織に新しくスクラムを適用する場合」の2つがあります。それぞれのケースを見ていきましょう。

ここからの説明では、「プロダクトバックログ」「インクリメント」などのスクラムに由来する用語をいくつか使っています。スクラムの解説はこのあとの第2章にありますので、用語に馴染みのない方は第2章を見ながら読み進めてください。

注1　https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Japanese.pdf

1つのスクラムチームから増やしていく場合

まずは「1つのスクラムチームから増やしていく場合」について、ある架空のチームを思い描きながら考えていきます。

●――チームを増やしたくなる動機

新しくソフトウェアサービスの開発がスタートすることになり、チームが作られました。そのチームにはプロダクトオーナー、スクラムマスター、エンジニアやデザイナーがいます。一般的にスクラムを始める場合は、小さなソフトウェアから仮説検証を繰り返しながら探索的に開発をしていくので、チームも小さいサイズで十分です。プロダクトオーナーとスクラムマスターを除いた開発者の数は4〜5人程度が適切です。

さて、このチームによって開発が進んでいき、サービスが順調にユーザーに受け入れられると、やらないといけないことが少しずつ増えていきます。日々の機能開発に加えて、ユーザーから報告された不具合の修正や、カスタマーサポートの対応などにも手が取られるようになってきました。ビジネスサイドは競合に対抗するため、いくつかの機能開発を望んでいます。しかしその前に期日が明確に定められている法改正への対応をしなければなりません。このようにしてプロダクトバックログアイテムが積み上がっていきます。チームも、スプリントごとに改善を繰り返してどんどん効率良くインクリメントが生み出せるように成長していますが、生産性がある日突然2倍や3倍にはなりません。スプリントごとに安定してプロダクトバックログを消化していても、あとからどんどんやるべきことが追加されます。こうして積み上がるプロダクトバックログアイテムを目の前にして、メンバーの1人の頭にふとこのような思いがよぎるのです。「もっとたくさんの人がいれば、一度にたくさんの仕事ができるのに……」。

このようにして、私たちは「チームを増やしたい」という考えに至るのです。

●――人が増えることでコストは大きくなる

多くの人がいれば一度にたくさんの仕事ができると考えるのは自然です。しかしソフトウェア開発においては、この考え方を安易に取り入れるべき

ではありません。これは「人月の神話」の罠に陥っているからです。

ソフトウェア開発の場合、タスクを担当する人どうしでの密なコミュニケーションが必要です。そのため、それに携わる人の数が増えるにつれて負担は急激に増加します。この負担の増加をわかりやすく説明したのが「ブルックスの法則」です。提唱者のFrederick Brooksは、著書の『人月の神話』の中でこのように書いています。

コミュニケーションを図ることで増える負担は、教育・訓練および相互コミュニケーションの2つの部分からなる。（中略）これらの教育は分けることができないから、負担の増加分は要員数に合わせて線形に変化する。相互コミュニケーションとなるとさらにひどい。仕事の各部分がそれ以外の部分と個別に調整されなければならないから、そのための労力は、人がn人いれば、n(n − 1)/2に比例する。要員が3人なら、それぞれの相互コミュニケーションが2人のときの3倍、4人なら6倍必要になる。さらに、もしその要員が共同で問題解決にあたるための会議が必要となると、事態は一層ひどくなる。

——Frederick P Brooks, Jr.著、滝沢徹／牧野祐子／富澤昇訳『人月の神話【新装版】』丸善出版、2010年、p.17

1つのスクラムチームの人数を増やして、たとえば1チーム10人とか15人で形成されるような形になるとします。そうするとブルックスの法則によってコミュニケーションにかかるコストは増大します。特にスクラムチームはメンバー間のコミュニケーションを緊密にやりとりしながら、チームによる自己管理を促進するための活動をしています。そこにたくさんの人数を押し込めるとその活動は阻害されてしまいます。

スクラムチームの適切なサイズは、開発者が4〜5人ほどです。そうするとチームが2つあれば、チームの健全な状態を維持したまま全体の人数は8〜10人にできるのではないか、と考えたくなります。しかしいくらチームが分かれていても、人数が増えることでコミュニケーション量が大きくなるのは同じです。個人間のコミュニケーションだけでもたいへんなのに、それに加えて複数のチームが絶えず同期を取りながら仕事をするのはさらにたいへんです。本書では、どのようにして複数チームで同期を取りながら仕事をすればよいのかを全体を通して解説します。ですが前提として「関係する人の数

が増えるとコストは大きくなる」のは忘れないようにしてください。

●――スクラムをスケールしない方法を考える

チームのスケールを考える前に、まずは1つのチームを維持したままうまくやる方法は本当にないのか、というのをギリギリまで考え抜いてください。

顧客からさまざまなフィードバックを得たり、ビジネスサイドから多くの機能開発の要望を受けたりしていると、やることはどんどん増えていきます。このようにして肥大化したプロダクトバックログであっても、優れたプロダクトオーナーは適切に扱っていくことができます。少ない労力で、プロダクトの価値を最大化するために本当に必要なものは何か。こうしたことに真剣にフォーカスしていくと、案外本当にやらないといけないことはそれほど多くないと気が付きます。やるべきことにしっかり焦点を絞って、そうでないものはスコープから外すなど、やることを少しでも小さくできないかを検討すべきです。

また、開発チームのキャパシティをさらに増やすことはできないかも検討材料となります。たとえば、繰り返しの作業を自動化する、フロー効率を高めて顧客に価値が届くまでのリードタイムを短縮する、などです。

スケールを検討する前に、できる工夫を最大限試しましょう。1つのチームが最大のパフォーマンスを発揮できるのは、チームの仕事がすべて自分たちで完結できるときです。

●――疎結合なスケールを検討する

人数が増えた場合に問題となるのはコミュニケーションコストであると先に説明しました。つまりチームが複数あったとしても、それらが常に同期的にコミュニケーションをする必要がなければ、大きなコストにはなりません。それぞれが独立して動くことができれば、スケールによる難度は大きく低減できます。

扱っているプロダクトのアーキテクチャに少し手を入れて、各チームが扱う範囲をそれぞれが独立して開発できるようにならないか、扱うドメインを分割できないかなどを検討するのもよいでしょう。この考え方は本書で解説

しているScrum@Scaleを組織に適用する場合にも役に立つ考え方です。

大規模な組織に新しくスクラムを適用する場合

ここまで述べてきたような、1つのスクラムチームを起点にチームが増えていく場合は、事前にスケールするための準備ができるので、比較的簡単です。

しかし、スクラムをスケーリングしたいと考えている組織は、すでにある程度の規模の開発組織が存在している場合が多いのではないでしょうか。つまり、最初から大規模な組織があって、そこにあとからスケーリングスクラムのフレームワークを当てはめて、組織全体をマネジメントしたい場合です。長い間運用されてきた組織構造がすでに存在している場合は、そこからの変革は困難を伴います。このケースでは、既存の組織構造、そこからの変革に対する抵抗勢力、すでに運用されているシステムのアーキテクチャなどが障壁になります。

●――大規模組織であっても最初は小さく始める

先に述べた「ブルックスの法則」に示されるように、コミュニケーションの経路をできる限りコンパクトに保つのが大規模な組織で開発をうまくやるポイントです。

仮に100人の開発組織があったとします。それが10人ずつの10チームに分かれており、それぞれが完全に独立して仕事ができる状態を考えてみましょう。組織をまたいだコミュニケーションがそれほど必要ないのであれば、やりとりはチームの中だけで完結します。この場合、組織のコミュニケーション経路の最大数は10人の開発組織とほぼ同じです。

大規模な組織にスケーリングスクラムを導入する場合にも、これを念頭に置きます。

すでにモノリスなシステム[注2]が稼働しているのであれば、コンポーネント

注2　複数の機能やサービスを1つのアプリケーションやモジュールで構成しているシステムをモノリスなシステムと呼びます。対して、機能やサービスごとにアプリケーションやモジュールを分割し、それらを連動して一連の機能を提供している構成をマイクロサービスと呼びます。

に分割して小さなチームが個々に独立して運用できるようなアークテクチャを目指します。最初はスクラムチームを1つだけ立ち上げ、そこに少数の職能横断的な人材を集めます。そして、スプリントを繰り返しながら、既存のシステムからコンポーネントを切り出します。これを繰り返しながら2つ3つとスクラムチームを立ち上げていき、それらにスケーリングスクラムのフレームワークを当てはめていきます。こうして、コンポーネントごとにチームを組成し、コミュニケーション経路をコンパクトに保ちます。

ただし、小さなチームを大きくするよりも、大きなチームを小さく分割するほうが当然複雑さは増します。原則としては、まだ組織や扱うソフトウェアのサイズが小さいうちから準備をするのが理想です。新規事業に取り組むチームなど、既存の組織構造とある程度切り離して独立して始められるような仕事があればやりやすくなります。まずはそれを行う専門チームだけをターゲットにしてスクラムを始める、というのもよくあるセオリーです。

スクラムのスケールは安易に選択すべきではない

ここまで、1つのスクラムチームから始めるにせよ、最初から大きな組織にスクラムを当てはめるにせよ、いずれにしろそれは難しいことである、と説明しました。

どのような場合であれ、規模の大きなものを扱うのは難しいです。まずはこれを念頭に置いたうえでスクラムのスケールに取り組んでください。規模の大きなものを大きなまま扱うのが難しいのであれば、規模の大きなものをできる限りコンパクトな要素に分割して扱っていくとよいです。

スクラムをスケーリングする際は、どうすれば単純さを可能な限り維持し続けられるか。それを考え続けながら取り組むとうまくいく可能性は高くなるでしょう。

最も機能しやすいスクラムチームは、1つのスクラムチームが機動的に動いている状態です。チームが増えるにつれ、複雑さは増していきます。これを十分理解したうえで、スクラムのスケーリングに取り組んでいきましょう。

さまざまなスケーリングスクラムのやり方

本書はScrum@Scaleを中心にスケーリングスクラムの解説をしますが、参考のためにいくつかほかのフレームワークも簡単に紹介します。それぞれの詳細は、公式ガイドや専門に書かれた書籍などを参照してください。

ここではLeSS、Nexusを取り上げます。また、スクラムのスケーリングではありませんが、大規模な組織にアジャイルを適用するフレームワークとしてSAFeも簡単に触れておきます。最後にScrum@Scaleの簡単な特徴も紹介します。

LeSS　1人のプロダクトオーナーと1つのプロダクトバックログ

LeSSは「Large-Scale Scrum」の略です。

LeSSは最大8チームまでの規模を想定しており、それを超える場合は「LeSS Huge」という形式に形を変えます。

「LeSSはスクラムである」と公式サイトにも謳われているとおり、通常のスクラムをそのまま拡張したシンプルなフレームワークです。プロダクトバックログはプロダクト全体に対して1つであり、プロダクトオーナーも1人です。単一のプロダクトバックログに対して複数のチームで取り組みます。

プロダクトバックログは1つですが、当然スプリントバックログはチームの数だけ必要です。まず、スプリントプランニング1ですべてのチームが一緒になって、どのプロダクトバックログアイテムをどのチームが担当するかを決めます。続いて、スプリントプランニング2では、チームごとに通常のスプリントプランニングを行います。

スクラムとしての各チームの活動は、スクラムガイドが示す普通のスクラムを行うだけです。ですがプロダクトバックログリファインメントはチームごとに行われるものとは別に、チームの代表者が集まって全体で行うものがあります。これをオーバーオールプロダクトバックログリファイン

メントと呼びます。

また、レトロスペクティブも各チームで開催するもののほかに、チームの代表者が集まる全体開催のものがあり、こちらはオーバーオールレトロスペクティブと呼びます。

LeSSは日本語の書籍[注3]も出版されていますし、公式サイト[注4]も日本語化されており、日本国内での実践者も数多くいます。そのため、日本語で発表されている事例なども比較的豊富です。日本のアジャイルコミュニティのイベントなどに参加すると、LeSSを取り上げたセッションを多く目にします。

Nexus 統合チームが統合の責任を持つ

NexusもLeSSと同様、プロダクトバックログは全体で1つです。

Nexusにおける大きな特徴は、「Nexus統合チーム」というロールが定義されているところです。これは、各チームで制作したインクリメントを統合し、少なくともスプリントごとに統合インクリメントを作成する責任を持ちます。

Nexus統合チームは、プロダクトオーナー、スクラムマスター、1人以上の統合チームのメンバーから形成されます。Nexus統合チームのメンバーであることは、個別のスクラムチームのメンバーであることよりも優先されます。したがって、Nexus統合チームと個別のスクラムチームのメンバーを兼務している場合は、Nexus統合チームとしてのタスクが常に優先されます。

スプリントプランニングやスプリントレビューなどのスクラムイベントは、Nexus全体としてそれぞれ実施され、一定のリズムでスプリントのサイクルが回ります。

公式サイト[注5]から日本語版のガイドを入手できますので、詳しくはそち

注3 Craig Larman／Bas Vodde著、榎本明仁監訳、荒瀬中人／木村卓央／高江洲睦／水野正隆／守田憲司訳『大規模スクラム Large-Scale Scrum(LeSS)―― アジャイルとスクラムを大規模に実装する方法』丸善出版、2019年

注4 https://less.works/

注5 https://www.scrum.org/resources/nexus-guide

らを参考にしてください。

SAFe　エンタープライズ向けビジネスフレームワーク

SAFeは「Scaled Agile Framework」の略です。

これまで紹介してきたLeSSやNexus、本書で取り上げるScrum@Scaleなどは、「スクラム」を大規模向けに拡張するフレームワークです。SAFeはそれらと違い、名前が示すとおり、スクラムに限らずさまざまな要素を取り扱います。リーンやアジャイルを企業の既存の組織構造に適用しつつ大規模に実践するための、ナレッジベースのフレームワークです。スクラムやXP、システム思考、DevOpsなど非常に広範なナレッジを扱います。

企業が持つ既存のマネジメントのしくみを活かしながらアジャイルを導入する方針として優れていますが、それゆえやや複雑なフレームワークになっています。プロダクトオーナーやスクラムマスターのスケーリングなど、Scrum@Scaleとの共通点も多いです。

SAFeは、3つのレベル(チーム、プログラム、ポートフォリオ)で構成されています。チームレベルでは、アジャイルな開発手法を使用して、高品質のソフトウェアを迅速かつ効率的に開発します。プログラムレベルでは、複数のチームが協力して大規模な機能を開発し、一緒にリリースします。ポートフォリオレベルでは、組織のビジョンや目標に基づいて、投資計画やリリース戦略を策定します。

SAFeは、組織がアジャイルな文化を受け入れ、適切な方法でアジャイル開発を導入するための包括的な方法論です。日本国内でも大企業などでいくつか採用事例を見かけます。

Scrum@Scale　プロダクトオーナーをスケールする

本書では一冊を通じてScrum@Scaleを中心に取り上げます。Scrum@Scaleでは、開発現場のHowの部分を担うスクラムマスターサイクルと、Whatの部分を担うプロダクトオーナーサイクルの2つが定義されています。

スクラムマスターサイクルは、開発チームが複数で連携するためのやり

方を定義しています。プロダクトオーナーサイクルは、複数の開発チームが連携するために、それぞれのチームに所属するプロダクトオーナーたちの仕事のやり方を定義しています。

このようにScrum@Scaleは、開発チームのスケールだけではなく、プロダクトオーナーのスケールも扱っているのが大きな特徴です。

ここで紹介したほかのスケーリングスクラムは、1つのプロダクトバックログを複数チームで扱います[注6]。対して、Scrum@Scaleは協調するスクラムチームごとにプロダクトバックログを持ちます。そのため、組織全体で複数のプロダクトを扱っているような場合でも、その全体を1つのスケールされたスクラムとして動かしていくことが可能です。

大規模スクラムの導入と組織文化

大規模スクラムの導入は、組織の広い範囲を巻き込まないといけないために骨が折れます。一方で、社内のある1つのチームが独自にスクラムを導入したいと考えた場合は、チームの中だけで多くを完結できます。

5〜6人ほどのチームに所属しているメンバーの1人が、参加したカンファレンスで大きな感銘を受けてスクラムを導入したいと考えました。そのメンバーは、独自に書籍やコミュニティなどでスクラムマスターとしての仕事を学習します。

さらにその人は、チームメンバーに対してスクラムを導入したい熱意を伝えます。そして自らがスクラムマスターを買って出て、スクラムガイドにしたがってチームの活動を整えていきます。

その結果チームがスプリントごとに各スクラムイベントをこなしながら

注6 LeSSをより大規模に拡張するLeSS Hugeには「エリアプロダクトオーナー」という複数のプロダクトオーナーが登場し、複数のプロダクトバックログを扱います。しかしLeSSを採用する多くの現場では、1つのプロダクトバックログを複数チームで扱う事例が多いため、ここでは「ほかのスケーリングスクラムは、1つのプロダクトバックログを複数チームで扱います」としました。

活動する型ができれば、そのチームは「スクラムを導入した」と言える状態になるでしょう。

ひょっとしたら、プロダクトオーナーをスクラムチームのメンバーとしてしっかり巻き込んでいくのは少し骨が折れるかもしれません。それでも1つのチームがスクラムを導入するレベルであれば、このようにチームの内部だけのボトムアップな活動でもなんとかなります。

一方、複数チームにスクラムを導入する場合はこのように簡単ではありません。

大規模スクラムの導入は組織的な支援が必要

1つのチームだけにスクラムを導入するケースと異なり、複数チームが大規模スクラムのフレームワークにのっとって活動するためにはボトムアップだけでは不十分です。5人とか10人といった少数の人たちの意識を変えるには、草の根活動でもなんとかなるでしょう。しかし大規模スクラムを動かすためには、場合によっては何十人という人が足並みをそろえていく必要があります。

特に、これから紹介していくScrum@Scaleでは、CEO（*Chief Executive Officer*、最高経営責任者）やCTO（*Chief Technology Officer*、最高技術責任者）、人事の責任者といったエグゼクティブの協力が不可欠です。

まずは組織内にあるどれか1つのチームにスクラムを導入してみよう、というように最初に小さく始めるのはかまいません。しかし、大きな規模で組織に対して何かを導入する場合は、どこかの段階でトップダウン的なアプローチが必要になってきます。

大規模スクラムの導入を成功させるための大きなポイントとして、トップダウンなアプローチができる人を必ず巻き込むべきです。そして、組織全体の文化を変えていかなければなりません。

大規模スクラムを成功させる「動機付け」

スクラムに限らず組織づくりのために重要なのは、メンバーのモチベー

ションをどう喚起していくかです。

Daniel Pinkの『モチベーション3.0』[注7]では、「外発的動機付け」と「内発的動機付け」が解説されています。

「外発的動機付け」とは、メンバー個人の外側に由来するものを動機としてモチベーションを引き上げることを言います。報酬であったり、評価や称賛であったりなど、これらはメンバーが所属する組織におけるルールづくりなどによって引き起こせます。

「内発的動機付け」は、次の3点の要素によって引き起こされます。「自律性」「熟達」「目的」です。

「自律性」は自分に裁量がある、という感覚。「熟達」は、難しい仕事を達成できる能力。「目的」は自分が成し遂げようとする事柄です。

「内発的動機付け」はメンバー個人の内側から沸き起こるものですが、それと同時に組織的な支援が必要です。

「自律性」を実感するためには、メンバーに仕事に対する裁量を持たせる必要があり、マイクロマネジメントな現場では実現できません。「目的」も、単なる個人的な目標にとどまっていてはいけません。所属するチームのゴールや、その先にある会社としてのゴールと地続きの目標となっていることで、より強力に組織に貢献できる力となっていきます。

このように、外発的動機付けや、内発的動機付けをうまく組織の目指す方向とそろえていくと、組織としての目的達成に近付いていけます。

第4章で詳しく説明しますが、Scrum@ScaleにはEATやEMSというリーダーシップグループの定義があります。特にEATは、組織全体の変革の起点となります。意思決定の中核を担うEATが明らかにした変革のビジョンに従い、Scrum@Scaleで定義された組織構造を通して変革のモチベーションが各チームへ伝播(でんぱ)します。このようにScrum@Scaleの中にはトップダウンで変革を推進していくための構造が備わっています。

このような観点からも、大規模スクラムの導入や、そこに起因する組織づくりにおいてトップダウン的なアプローチが欠かせないことがわかります。

注7 Daniel Pink著、大前研一訳『モチベーション3.0――持続する「やる気！」をいかに引き出すか』講談社、2010年

まとめ

本章では、Scrum@Scaleの解説へ入る前提として、スクラムのスケールそのものに焦点を当てました。

人が増えることによりコミュニケーション経路が複雑化することで、チーム活動のコストが上がってしまうことに問題があります。また、大きな物事を扱うのは本質的に難しいものです。

そのため、なるべくコミュニケーション経路が少ない状態を保ち、スクラムを導入する場合はできるだけ小さく始めることが重要です。