～スクラム開発とは～アジャイル開発の一種について

岡田　隆之

<①基礎編>

◆スクラム開発の内容

スクラム開発とは、アジャイル開発の中の一種で、その中で最も有名なもの。チームを組んで役割やタスクを分散しつつ、コミュニケーションを取りながら行う特徴がある。スポーツとして馴染み深いラグビーのスクラムにちなんで、「スクラム開発」と名付けられた。スクラム開発はチーム全員で開発を進める手法のため、アジャイル開発の中でも、チーム間のコミュニケーションがより大切になる開発手法。

チームとして「何が必要か？」「いつまでに必要か？」「誰が何をやるべきか？」という視点からチームメンバーにタスクを振り分け、それぞれがそのタスクを達成することでプロダクトの完成を目指。それぞれの作業が、他の人の作業を支えている形になるので、チームワークやコミュニケーションが重要になる。

それぞれの人が役割を持って開発に取り組めるため、さまざまな作業を同時進行的に進めることができ、効率化をはかれる。反対に、コミュニケーションが取れなくなると、開発がスムーズに進行しないこともしばしば。また、チーム全員による作業の精査を行うことも可能なので、より充実したプロダクトの制作ができる。複数人でプログラミングやシステム開発を進めるときには、スクラム開発の採用を検討される。

◆スクラム開発に必要な人材（スクラムチーム）

・プロダクトオーナー（PO）：開発プロジェクトにおけるオーナーで意思決定を行う存在。スクラムマスター（SM）やステークホルダーとの連携によってプロジェクトを成功に導くのが役目。そのため、市場やニーズを把握していること、目的についての知識も要求される。
・スクラムマスター（**SM**）：開発プロジェクトにおけるマネージャーのポジション。システム開発の指揮やスケジュール管理などが主な役目。＋トラブルが発生した際に解決役を担うことも多い。
＊スクラム開発においては、開発の管理を担うプロジェクトマネージャーという名称のポジションが存在しない。そこで、スクラムマスターが全体の進行をサポートすることになる。確認作業や、トラブルを解決するためのポジションでもあり、全体を統括する。システム開発の精度を上げるためのポジションでもあり、長期的にチームでのスケジュール管理を担っている。チームのプロセスを考慮して、それぞれの開発エンジニアにアドバイスなどを行うのもスクラムマスター。積極的に改革を行える人材が求められるポジションとされる。
・ステークホルダー：開発プロジェクトに資金提供を行うスポンサーの立場。直接的に開発に携わることは少ないが、POやSMに対する発言権を持つ存在。
・開発エンジニア：開発プロジェクトで実際にシステム開発を行うスタッフ。開発に必要なスキルを備えたエンジニアやデザイナーが採用される。開発エンジニアのスタッフ間の上下関係はなく平等であることがほとんど。開発エンジニアは、ステークホルダーからの要望をヒアリングし、調整、交渉を繰り返すことで、より開発するサービスの精度を上げるという役割も担っている。開発エンジニア自身があまり知識のない分野においては、その分野に精通したエキスパートの指示のもとにプロダクトを開発する場合もある。

◆スクラム開発で使用される用語

スクラム開発で使用される代表的な用語を紹介する。
・スプリント⇒項目に分けてプロジェクトを小単位に区切ったもの。システム開発の進捗や状況などを確認し、状況に応じてスプリントの調整を行う。
・スプリントプランニング⇒名前そのまま、スプリント内での計画のこと。作業の工程や予測を含めた、実現可能なプランを具体的かつ詳細に計画する。
・デイリースクラム⇒スプリントにトラブルがないかどうかを確認するミーティングのこと。プラン通り

に作業が進んでいるかを確認する。
・スプリント・レトロスペクティブ⇒プロダクトが仕上がったら改善点を模索して、より精度の高いプログラムを組む作業。何度か修正を繰り返すこともある。
・プロダクトバックログ⇒プロダクトの作成にあたり、項目を必要順に並べたリストのこと。のちに、項目の追加も可能。
・プロダクトバックログ・リファインメント⇒プロダクトバックログに含まれる項目に対して、詳細の追加、見積もりを行うこと。

◆スクラム開発の手順

以下が、具体的なスクラム開発の手順:

- バックログを作成する？⇒まず、つくるものをリスト化し書き出す
- スプリントプランニングを行う⇒そのあとで計画を立てる
- スプリントで実装する
- スプリント・バックログで実装、確認する
- デイリースクラムでコミュニケーションをはかる
- ステークホルダーに対するスプリントレビューで性能をフィードバックする

プロジェクト終了後:

- スプリントをチームで振り返る

まずバックログを作成し、プロダクトごとに制作するものをリストアップする。その後、スクラムチームでスプリントプランニングミーティングを行い、工数の見積もりを行う流れ。また、ここでメンバーに作業を振り分けて分担させる。
⇒それぞれのスタッフが指揮や作業に入って以降は、デイリースクラムで都度、情報共有をする。ここで日々のタスクや発生している問題などを解決。
一旦プロダクトが完成したら、ステークホルダーに対してスプリントデビューを行い、スプリントの反省会を行う。このレビューによって、プロダクトの完成度をさらに高めることができる。

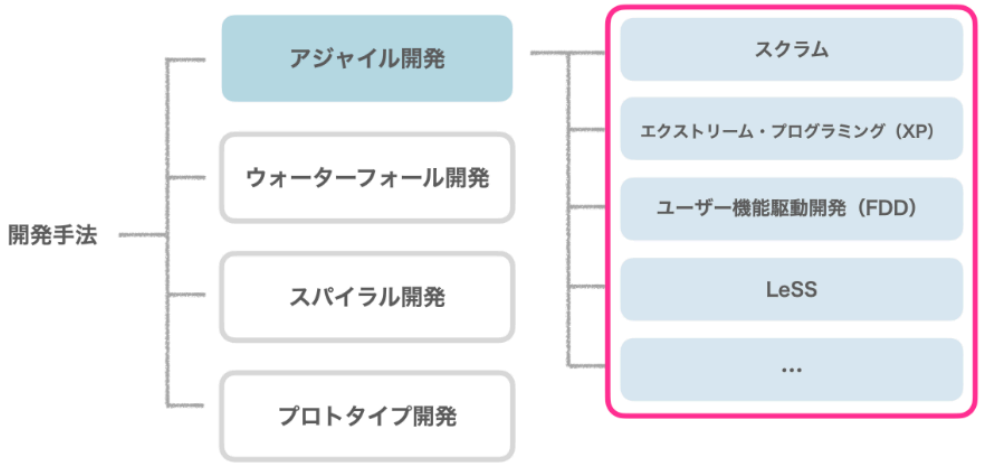
スクラム開発ではこのように、常にチームで知識を共有、問題点を指摘し改善しながら、プランニング、設計、テストを行えることで、⇒プロダクトの完成度を高くできるのが特徴。

◆スクラム開発に適した案件

スクラム開発に適した案件は、後々に更新が必要なシステムといわれる。＝（例を挙げると、）常に更新が必要な運用型の**WEB**サイトやアプリケーション開発、**or**最終的な納期よりも品質を優先したい案件に向いているとされる。チーム間で連携をしつつ、速度と確実性の高い開発を行いたい場合は、スクラム開発を選択することがおすすめ。

しかし、案件によっては他の開発方法を選択した方が良いことも少なくない。例えば、柔軟性のあるアジャイル開発と比較されやすい開発手法としてウォーターフォール型がある。
端的に言えば、ウォーターフォール型は最終的な納期までにプロダクトの完成品を納品するという手法。要件定義から、リリースまで、途中レビューなどをあまり行うことなく（スピーディーに）行う。
⇒このウォーターフォール型での開発は、単調で更新があまり必要ないプロダクト、納期を確定させたい案件が最適。
<②発展編>
下図のように、アジャイル開発はアジャイル開発、ウォーターフォール開発、スパイラル開発、プロトタイプ開発とある中の、開発手法の大分類の**1**つ。機敏・迅速・柔軟なソフトウェア開発の手法の「総称」を指す。

そして、このアジェイル開発の中に、スクラム、**XP**、**FDD**、**LeSS**といった小分類がある。



参照：https://seleck.cc/scrum

そして、アジャイル開発の中にいくつかの開発手法があり、現在最も多くの企業で使われている手法が「スクラム開発」となる。

コミュニケーションを重視した少人数での開発

5～10名程度の少人数で構成されたチームで、メンバー同士が綿密にコミュニケーションをとりながら開発を進める。チームが一体となって開発を進める様子がラグビーのスクラムに似ていることから「スクラム開発」と名付けられた。

スクラム開発のメリット

主に以下のものがある：

- 顧客の細かい要望に対応しやすい
- 正確な見積もりを立てやすい
- メンバーのスキルを最大限に発揮できる
- 属人的な作業を削減できる

大規模なスクラム開発体制「**LeSS**：**Large-Scale Scrum**」について

以下が、日本最大級の実名型グルメサービス「Retty」を展開するRetty株式会社のLeSSの導入事例：　（次ページへ）

2019年時点の同社の開発体制は「検索」「SEO」「ネット予約」といった目的別のチームに分かれ、それぞれで部分最適を進めていた。しかし、、、チームが個別に動くことで、プロダクト全体の優先順位がつけられなくなる、ユーザー体験のズレが発生するなどの課題が生じたという。

そこで同社ではより良い開発体制の構築を目的に、大規模なスクラム開発体制「**LeSS**：**Large-Scale Scrum**」を導入した様子。

組織全体でひとつのスクラムを回すという考え方を持つLeSSでは、メンバーがそれぞれの目的別に動くのではなく、幅広い領域について協力し合いながら開発を進める。

このLeSSの導入によって、同社では「チーム間のコラボレーション意識の向上」「事業全体で優先したいもの集中する」といった変化が生まれたそう。

また結果として、「ユーザーのニーズ」に焦点をあてた議論が増え、ユーザーにより早く価値を届けることができるようにもなったという。

同社のLeSSへの移行が良い影響を与えたプロジェクトとしては、コロナ禍のGo To Eatキャンペーンへの対応が挙がる。全社の最優先事項に位置づけ、みんなでスクラムを回すことで、無事にリリースにまでたどり着くことが出来たという。。。

<③応用編>
以下では、さらに聞きなれないような

- エクストリーム・プログラミング(XP)
- ユーザー機能駆動開発(FDD)

についても紹介する。

@1エクストリーム・プログラミング(XP)

<②発展編>の図で紹介したように、アジャイル開発における開発手法の1つ。略してXPと呼ばれる。
(アジャイル開発とは、最初に綿密な計画を立てず、臨機応変に進める開発方法のこと。設計・実装・テストを短期間で何度も繰り返しながら、顧客の意見や要望を都度取り入れて開発を進める。設計・実装・テストの1サイクルをイテレーションと呼ぶ。)

ただし、闇雲に開発を進めてもうまくいかないので、アジャイル開発の指針となる手法があり、その代表格がスクラム(開発)。エクストリームプログラミングはスクラムほどではないが、アジャイル開発手法として知られている。スクラムが勢いよく製品を作る手法であるのに対し、エクストリームプログラミングは継続的な成長に主眼を置いている。

◆エクストリームプログラミングの5つの重要ポイント
エクストリームプログラミングでは、以下の5つの重視すべきポイントが提唱されている:
i)コミュニケーション
プロジェクトが失敗する原因の多くはコミュニケーション不足。エクストリームプログラミングでは開発チーム内だけでなく、顧客とのコミュニケーションも重視する。
ii)シンプル
最初の設計を極力シンプルにする。基本的な機能だけを盛り込み、そのほかの機能が必要になれば、その都度対応する。⇒**XP**の特徴ポイント。デメリットも
iii)フィードバック
ソフトウェア開発において、不要な機能を盛り込むのは大きな無駄となる。顧客からのフィードバックを得て、必要な機能を洗い出すことが大切。そして無駄を省く!
iv)勇気
最初に綿密な計画を立てないという特性上、途中で大胆な変更が求められる場合がある。
iv)尊重
チームで開発する以上、ほかのメンバーを尊重する姿勢は欠かせない。

◆エクストリームプログラミングの慣習(=プラクティス)
エクストリームプログラミングにおいては、以下のプラクティスが存在します。
i)共同プラクティス
エクストリームプログラミングに関わる全員を対象としたプラクティス。以下の4つの行動を実行する:
・反復
⇒1つのイテレーション(設計・実装・テスト)が約1~2週間で終わるような開発とし、これを何度も繰り返しながら開発を進める。
・共通の用語
⇒用語集を作成する。これにより、コミュニケーションの齟齬を防止する。
・開けた作業空間
⇒開発チームと顧客間で密なコミュニケーションをとりやすく、作業に集中できる広い環境を構築する。
・回顧

⇒ミスが再発しないよう、作業状況を明らかにし、過去のフィードバックを活かす。

ii)開発プラクティス

開発プラクティスは、プログラマーなどの開発チームを対象としたプラクティス。大きな括りで分類して紹介する（細かくは19つある様子）。

-テスト後に実装を行う「テスト駆動開発」

・・プログラムの実装よりもテストコードを先に作成すること。

⇒それにより、求められる機能が洗い出され、シンプルな設計が実現する。◎

テストの通過に絞って開発を行えば、仕様変更などによる開発途中のブレを最小限に抑えられる。なお、テストはユニットテストと受け入れテストからなり、どちらのテストも自動化が望ましい。

-2人1組で行う「ペアプログラミング」

・・2人1組でプログラミングを行うこと。1人がコードを記述し、もう1人はそれを確認・補佐する。

記述しながら確認を行うと、細々とした問題をその場で解決できるメリットがある。また、記述されたコードを把握している人物が2人いるので、その後の問題発生時にも迅速な対応が可能になるかも。

-内部構造を整える「リファクタリング」

・・完成したコードをわかりやすく書き換えること。外部の動作を変えずに、内部構造だけを変更する。同じ動作をするコードでも、わかりやすいものに変換されるので、メンテナンス性の向上や不具合の発生頻度の低下が期待できる。

-必要なコードだけを記述する「**YAGNI**」

・・YANGIは「You Aren't Going to Need It」＝「今必要なことだけをする」という意味。つまり、必要なコードのみを記述することを意味する。

開発時には、後に必要になりそうな機能を考慮して、あれこれと盛り込みたくなるかもしれない。しかし、そうした思惑は外れる場合が多く、結果として無駄が出る。今必要とされるコードの記述に注力することが大切という考え方。

iii)管理者プラクティス

・・管理者を対象としたプラクティスです。開発が適切に進行しているのか把握し、チームで共有する必要がある。特に、チームの負荷が大きすぎないか常に確認することが求められます。エクストリームプログラミングは短期間で集中して作業を行う方法**(1~2**週間**)**であるため、過度な労働は避けなければならない。

iv)顧客プラクティス

・・その名のとおり顧客を対象としたプラクティス。

エクストリームプログラミングでは、顧客も開発に携わるチームメンバーとして扱う。ビジネス的な視点から必要な機能を考え、開発の優先順位を付けるのが顧客の役割。

具体的には、要求機能のコンセプトを短い文章にしたストーリーの作成やリリース計画を立案する。また、イテレーションごとに開発チームとともに受け入れテストを行い、求めた機能が実現しているか確認する。

（以上、XPのプラクティス）

◆エクストリームプログラミング**(XP)**まとめ

まずプログラミングから入るということで、動作を確認できながら無駄なく必要な部分だけ早く実現できる強みがあると感じた。また、顧客を開発メンバーと考えて積極的に通信したり、ペアプログラミングも慣行（プラクティス）に挙げるなど、特徴的な点が多い手本だったと感じた。採用の機会があればいかがだろうか？

@2ユーザー機能駆動開発(FDD)

◆ユーザー機能駆動開発（**FDD**）とは

ユーザー機能駆動開発（**Feature Driven Development**：**FDD**）とは、ユーザー目線で価値のある機能を中心に開発を進める、アジャイル開発手法の一つです。顧客にとって価値のある小さな単位の機能（＝**feature**）という観点から開発を行い、稼働するシステムを**2**週間未満の短い期間で作成し顧客に提供する。

◆ユーザー機能駆動開発の特徴
ユーザー機能駆動開発の特徴は「**5**つの基本活動」と呼ばれる開発工程と、
＋ソフトウェア工学における「ベストプラクティス」を中心に構成されている点。
それぞれどのようなものなのか、詳しく見ていく。
i)ユーザー機能駆動開発の「5つの基本活動」
ユーザー機能駆動開発はモデル駆動型の開発（＝モデル図と呼ばれる仕様を基に、プログラミングを行いシステム開発を行う開発）の工程で、2週間未満の短い開発期間を繰り返して開発を行う。
その時にユーザー機能駆動開発で以下の5つの基本活動を行う。
1.全体モデル開発
プロジェクトは、システム全体の範囲と内容についてのウォークスルー（＝プロジェクトのメンバーが仕様や構成の問題点を探したり解決策を討論したりする作業）から開始される。
次に部門ごとのウォークスルーを行い、より専門的にモデルの作成を行う。こうして出来上がった各部門の複数のモデルをまとめたものが全体モデルとなる。

2.フィーチャーリストの構築
全体モデル開発にて収集された知識を基に、顧客の考える重要度ごとに分類されたフィーチャー（**feature:**機能）のリストアップを行う。
このリストは<action>、<result>、<object>という様式で記述され、利用者の操作（action）、その操作の結果（result）、実装するオブジェクト（object）を指定して記述される。フィーチャーとは2週間以内で開発できる小さな機能のことで、**2**週間以上かかる場合はそのフィーチャーを更に小さく分割し、**2**週間以内で開発できる粒度まで小さくする。
3.フィーチャーごとの計画
フィーチャーリストの作成が完了したら次はその開発計画を立案する。
フィーチャーの依存関係や開発チームの負荷状況、フィーチャーの複雑さなどを考慮し、着手するフィーチャーの順番、フィーチャーを実装先のクラスを担当する開発者を割り当てる。各クラスにはそのクラスの開発責任を負うプログラマー（オーナー）を決定する。
4.フィーチャーごとの設計
2週間で開発が可能なフィーチャーをいくつか選択し、それらの設計パッケージを作成する。
フィーチャーに関連するクラスのオーナーと連携し、フィーチャーごとの詳細なシーケンス図を作成し、全体モデルを更新する（シーケンス図とは、クラスやオブジェクト間のやりとりを時間軸に沿って表現する図のこと）。その後開発者はクラスとメソッドの概要を書き、設計インスペクションと呼ばれる第三者による「設計の不具合や問題」の確認を行う。
5.フィーチャーごとの構築
フィーチャーごとの設計が完了すると、フィーチャーを実装するためのコーディングを行う。
コーディングが終わったら単体テストを行い、コードインスペクションと呼ばれる第三者によるコードの確認を行う。
＋ユーザー機能駆動開発の「ベストプラクティス」とは
ユーザー機能駆動開発ではソフトウェア工学に基づく「ベストプラクティス」を採用した。ソフトウェア工学とは、有用な（有名な）ソフトウェアが持つ特性や構造を探り、それらの構築や管理に有用なプロセスを見出す学問。
ユーザー機能駆動開発を構成するベストプラクティスは以下になる：
・ドメイン・オブジェクト・モデリング
⇒解決すべき問題の領域を探査し、説明する。
・フィーチャー毎の開発
⇒**2**週間以内に実装できない機能は更に小さな部分機能に分割され、最終的にフィーチャー（feature）と呼ばれる小さな機能単位になる。
・クラス毎のオーナーシップ
⇒オーナーは、クラスの一貫性、性能、概念的完全性に責任を負う。
・フィーチャーチーム
⇒小規模で動的に結成されるチーム。動的にチームを結成することで、個人や固定的なチームで陥り易い独善的な開発が防げる。
・インスペクション
⇒インスペクションは、第三者による検証によって不具合や問題を見つけること。不具合を検出することで設計やコードの品質を保証する。
・構成管理
⇒全フィーチャーのソースコードについて、進捗を管理し、フィーチャーチームによる改変の履歴を保守する。
・定期ビルド
⇒定期的にビルドを行うことで、顧客に対してデモンストレーション可能な最新システムを提供することができる。また、ソースコード結合時のエラーを早期に発見できる。
・進捗と成果の可視化
⇒プロジェクト内外に頻繁で適切で正確な進捗報告をすることで、マネージャーがプロジェクトを適切に運営する補

助となる。

◆ユーザー機能駆動開発のメリット、デメリット（結局何がいいのか）
◎ユーザー機能駆動開発のメリット◎
・機能重視のシンプルなシステムが開発できる
ユーザー機能駆動開発では、顧客が必要な機能とする機能を重要度ごとにまとめたフィーチャーリストを作成し、そのリストを基に開発を進める。フィーチャーリストには不必要な機能はなく、必要な機能のみを実装する開発を行うため、完成したシステムがシンプルになりやすい。
・大規模なプロジェクトにも対応しやすい
ユーザー機能駆動開発では、最初に全体モデルを作成し、フィーチャーごとにチームを分けて開発を行う。⇒フィーチャーごとにチームが独立しているため、並行して別の機能の開発を行うことができ、開発機能が多い大規模なプロジェクトにも対応が可能。

xユーザー機能駆動開発のデメリットx
・コミュニケーションコストが発生する
ユーザー機能駆動開発では、ユーザーが必要とする機能を中心に開発する開発する開発手法。そのため、ユーザー自身がどのようなシステムを求めているのかを明確に把握し、プログラマーに共有する必要がある。しかし、ユーザーは漠然と欲しい機能は決まっているがどのようなシステムが欲しい、という明確な方針を持っているとは限らない。
そのため、開発者はユーザーが真に求める機能をユーザー目線で考え、システムに落とし込むためのコミュニケーションを行う必要があるため、コミュニケーションコストが多く発生する。

（続く）
◆ユーザー機能駆動開発**(FDD)**まとめ
ユーザー機能駆動開発は、以上のように先にユーザーの求めているものをまとめ上げてから開発に取り組むので、コミュニケーションコストが高くつき、ウォーターフォールに似た部分があるといえるかもしれない。
しかし、各作業単位フィーチャーが2週間未満で終わるようにすることを徹底していたり、定期的に、第三者による検証やプロジェクト内外に頻繁で適切で正確な進捗報告をすることを盛り込んでいたりするので、アジャイル的な方向転換も可能である。
この**FDD**はスクラムに比べてあまり見かけないが、
部分的に参照したり併用したりすることで、アジャイル開発の性能向上につなげられるかもしれない。

以上、アジャイル開発の中の一手法スクラム（開発）、そして同じくアジャイル開発に属する**LeSS(**<②発展編>**)**、**XP**、**FDD(**ともに<③応用編>で記載**)**についても取り上げて紹介した。
こちらの資料が何かの開発で役に立てば幸いである。

（参考資料のURL）
https://hnavi.co.jp/knowledge/blog/scrum/ ⇒基礎編
https://seleck.cc/scrum (再掲)⇒発展編
応用編：
https://it-trend.jp/development_tools/article/32-0023 ⇒エクストリームプログラミング開発
https://www.alobridge.com/blog/1130/ ⇒ユーザー機能駆動開発(FDD)

（資料終わり）