

SCRUM BOOT CAMP

西村 直人、永瀬 美穂、吉羽 龍太郎 著



【プロダクトオーナー】
 プロダクトに対して責任を持ち、
 プロダクトバックログを並べ替える



【開発チーム】
 プロダクトの開発を行う
 成功に向けて最大限の努力を
 約束する

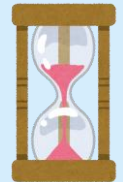
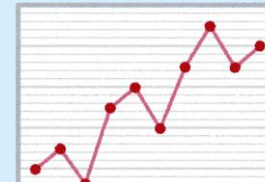


【スクラムマスター】
 スクラムがうまくいくようにする
 外部からチームを守る

【プロダクトバックログ】
 プロダクトオーナーの要求一覧
 プロダクトオーナーが順番を並べ替える
 開発チームが各項目の見積もりを行う

【スプリント計画ミーティング】
 プロダクトバックログを分析、評価
 スプリントで開発するプロダクトバック
 ログを選択
 選択した項目を個々のタスクに分解

【デイリースクラム】
 毎日同じ時間、同じ場所でゴールに向かっていくか
 作業の進捗、問題点がないかを共有



残作業をチャートで追跡



【スプリント】
 最長1ヶ月までのタイムボックス。各スプリントの期間は均一



【スプリントバックログ】
 今回のスプリントで行うタスクリスト

【作業&デイリースクラム】

【スプリントレビュー】
 スプリント中の成果物である
 動作するソフトウェアをデモ

【スプリントレトロスペクティブ】
 もっとうまく仕事をするための振り返り
 アクションプランを作る

【リリース判断可能なプロダクト】

Scrumとは

<プロダクトバックログ>成果物1

実現したい要求をリストにして並び替える
常にメンテナンスして最新に保つ

<プロダクトオーナー>ロール1

プロダクトの責任者
プロダクトの結果責任を取る
プロダクトバックログの管理者で、並び順の最終決定権限
プロジェクトに必ず一人必要
開発チームを活用してプロダクトの価値を最大化する
開発チームに相談できるが干渉はできない

<開発チーム>ロール2

動作するプロダクトを開発する
リリース判断可能なプロダクトをつくる
3~9人で構成する
全員そろえばプロダクトをつくれる
上下関係はない

Scrumとは

＜スプリント＞短く区切って繰り返す

スプリント期間は固定する。(変動してはいけない)

※リズムができて集中できる。プロジェクトのゴールに対する進捗が把握しやすい。リスクに対応しやすい。

製品のリリース判断に必要なすべてのことを行う

短くて1Week、長くて4Weekと週単位で設定することが多い

＜スプリント計画ミーティング＞会議体1

スプリントで開発するためには計画が必要

プロダクトオーナーは何をほしいのか(プロダクトバックログの項目)

開発チームはどれくらいできそうか

開発チームをそれをどうやって実現するか

※スプリントの実績のことをベロシティと呼ぶ

第1部でプロダクトオーナー、開発チーム、スクラムマスターが参加し、そのスプリントで、どのプロダクトバックログの開発するのかを検討し、内容を確認。

第2部でプロダクトバックログの項目を完了させるために必要なすべての作業を開発チームによって洗い出す。これらの作業はタスクと呼ばれタスクの一覧をスプリントバックログと呼ぶ。

個々のタスクは一日以下の単位とする

Scrumとは

<スプリントバックログ> 成果物2

プロダクトバックログを具体的なタスクに分散する
タスクは後から増えることもある

開発チームはスプリント計画で合意した内容を完了させることに全力を尽くす必要はあるものの、計画したことをすべて完了させることを約束しているわけではない。

見積もりが外れたり、難易度が高かったり、不測の事態が発生した場合に開発チームが長時間残業したり、必要な作業を省いたりしてしまうかもしれない。
スプリントバックログの項目の担当を特定の人が決めることはしない。
スプリント計画会議の時点で、すべての項目の担当を決めることもしない
作業に着手するときには作業する人自身がスプリントバックログの項目を選択するようにする。

<リリース判断可能なプロダクト> 成果物3

リリース判断可能な指す内容について共通の基準を持つ必要がある。
これを完了の定義(品質基準)と呼ぶ。

途中で定義を追加することは許されるが、途中で定義を削除するのは要求される品質を達成できなくなるので避けた方がよい

Scrumとは

<デイリースクラム>会議体2

15分間のタイムボックスで行われ延長できない。

各メンバーは下記3点について開発チーム全体にむけて簡潔に報告する。

1. 前回のデイリースクラムからやったこと
2. 次のデイリースクラムまでにやること
3. 困っていること

デイリースクラムは開発チームのための会議であり、特定の誰かにむけての報告会議ではない。問題解決の場でもない。

<スプリントレビュー>会議体3

開発チームの成果物をプロダクトオーナーが確認する。

スプリントレビューで確認するのは実際に動作するプロダクト

以下について報告・議論する

1. 開発チームが完了できなかったプロダクトバックログの項目について説明する
2. プロダクトオーナーがプロダクトの状況やビジネスの環境について説明する
3. プロダクトバックログに追加すべき項目の有無について議論する
4. プロジェクトを進めるうえで問題となる事項について関係者で議論する
5. 現在の進捗を踏まえて、リリース日や完了日を予測する

Scrumとは

<スプリントレトロスペクティブ>会議体4

プロセスやツールなどの観点で今回のスプリントを検査する
うまくいったこと、今後改善すべき点を整理する
今後のアクションプランをつくる

<スクラムマスター>ロール3

このプロセスがうまくまわるようにする
妨害を除外する
支援と奉仕をする
教育、ファシリテート、コーチ、推進役

スクラムマスターがプロダクトオーナーや開発チームに対して行うことの例

1. わかりやすいプロダクトバックログの書き方をプロダクトオーナーや開発チームに教える
2. プロダクトバックログの良い管理方法を探す
3. プロダクトオーナーや開発チームにアジャイル開発やScrumについて説明して理解してもらう
4. スプリント計画ミーティングやスプリントレトロスペクティブなどの会議の進行を必要に応じて行う
5. プロダクトオーナーと開発チームの会話を促す
6. プロダクトオーナーや開発チームの生産性が高くなるように変化を促す

仕事を進めるうえでの妨げになっていることをリスト化して優先順位をつけて解決の方法を検討し、必要に応じてしかるべき人に解決を依頼するといったことも行う。このリストは妨害リストと呼ばれる

プロジェクトのゴールを考える

Scrumはプロジェクトの進め方について焦点を合わせているので、それを補うためにScrumでは決められていないインセプションデッキを紹介

<インセプションデッキ>

プロジェクトを始める前に明らかにしておくべきことを知るための活動

10個の質問という形でまとめられており、それぞれについてスクラムチーム全員で話し合う話し合っって明らかになったことはプレゼン資料のようなスライドにまとめていく。

その中にはゴールとミッションについての質問も含まれている。

今回は10個の質問のうち「エレベーターピッチ」と「我われはなぜここにいるのか」を紹介

エレベーターピッチ・・・今からつくるものに予算がつくほどの大きな期待がかけられている理由を知るための質問

どんな人に向けたもので、どんなことができるものなのか、そしてそれはすでにある他のものと違って、どういう利点があるのか、そしてこれによってどんな期待がかけられているのかを知ることができる

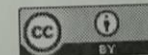
<https://github.com/agile-samurai-ja/support/tree/master/blank-inception-deck>

エレベーターピッチ

ものに予算がつくほどの大きな期待がかけられている理由を知るための質問
どんな人に向けたもので、どんなことができるものなのか、そしてそれはすでにある他のもの
と違って、どういう利点があるのか、そしてこれによってどんな期待がかけられているのかを
知ることができる

エレベーターピッチ

- ・ [最新情報をもとに効率よく営業活動を] したい
- ・ [ずっと外出している弊社の営業マン] 向けの、
- ・ [New営業支援くん] というプロダクトは、
- ・ [営業支援システム] です。
- ・ これは [外でも簡単に操作すること] ができ、
- ・ [既存のシステム] とは違って、
- ・ [外出先でアクセスしやすい方法がいくつか提供され、最新の情報をいつでも参照・更新できる仕組み] が備わっている。



我われはなぜここにいるのか

達成しないとプロジェクトの意味が失われてしまう理由を明らかにする質問
プロジェクトの周辺にいる人たちはそれぞれの思い思いの期待がある。

その期待にすべて応えていくのは大変。期待される内容も重要なものとそうでないものがある。
さまざまな期待のうちどれを達成すればプロジェクトは失敗とみなされないか、を知るために
重要と思われることを3つに絞り、その中でも絶対に死守したいこと最も重要なことを1つ明らかに
にする。

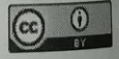
できる。

我われはなぜここにいるのか

- ・全支店で最新情報を入力してもらえらるように
外出先でまともに使えるシステムにする
- ・半年後に開発人員が他で必要なので、それま
でに終わらせる
- ・社内でScrumでの開発実績をつくる

外でまともに使えるシステムにする

「我われはなぜここにいるのか」



プロダクトバックログ

プロダクトバックログは実現したいことがすべて書かれている表

単なるToDoリストのようだけど、計画についてのさまざまなことを把握するための重要な表
Scrumでは機能・要求・要望・修正事項などを全部書くことは決めているが
詳細なフォーマットは決めていない

<プロダクトバックログの例>

欲しい機能を列挙したもの

機能	目的	詳細	見積
営業日報入力機能	最新の情報をもとに 営業戦略を考えたい	日単位で訪問先、日時、 担当者、案件情報を 入力する	
ログイン機能	機密情報なので 利用者を制限したい	全社員が社員番号と パスワードで認証する	
取引先検索機能	事前情報をもった状態で 有利にやり取りしたい	業種、会社名、会社規模、 重要度などで取引先を検索する	
....			

ユーザーストーリーという形式で書いたもの

ストーリー	デモ手順	見積
外出先の営業マンとして、毎日訪問 先の状況を記録したい。それは最新 の状況をもとに営業部として戦略的な 営業活動をしたいからだ。	XX社の記録ページを表示して、訪問日時と訪問者、 商談状況、報告内容を入力して記録ボタンを押す。 確認画面にユーザー名が..	
利用者を制限したい。それは機密情 報を正社員のみに表示したいからだ。	未ログイン状態でアクセスするとログイン画面が表示 される。 Xxさんの社員番号とパスワードを入力して..	
営業マンとして、取引先についてさま ざまな視点で探して、詳しい内容を知 りたい。それは取引先とのやり取りを 優位に進めたいからだ。	トップページから検索タブを押すと検索画面が表示さ れる。検索条件として会社名、業種、資本金、住所..	
....		

プロダクトバックログ

プロジェクトに深刻な影響を与えるような重要な項目があとからどんどん追加されていくとどうしようもなくなるため、実現したいことはおおきな漏れがないようにしておく。

スクラムチームみんなでプロダクトバックログに書いたほうがいいと思う項目を付箋などで書き出していく。

プロジェクトのゴールや概要についてわかる資料を持ち寄る。

いまから実現していくもののイメージを明確にするのに役立つものならなんでも持ってくる

→プロジェクトのゴールを達成するために必要だと思うものや、やっておいたほうがいいと感じるものをとにかく書き出していく。
ここで大事な量は量。プロジェクトの期間内ではやらないかもしれないという先入観は取り払う

その1つ1つに順序をつけていく。

ステークホルダーに決めてもらいたいと思うかもしれないが、うまく決められなかったり、決めた順序をスクラムチームが理解できなかったりする。自分たちが大丈夫と思えるような見通しを立てるために自分たちで順序をつけることで自分たちが大丈夫だと思えるものにする。情報が不足しているなら、まずは情報集めから始める。

プロダクトバックログを見れば全体の大まかな流れがわかる

スクラムチーム全員がプロダクトバックログについて知っておかないといけない

プロダクトバックログは最初に作ったら終わりじゃない。

少しでもゴールに近づくためにこれから先もずっと更新していく

プロジェクトの見通しを立てるのは最初だけでなくずっとやり続けていく

プロダクトバックログの各項目の見積もり

実際に作業する人が最終的な見積もりをする

作業の量を見積もることが、自分たちの作業について考える良い機会にもなる

作業の量を数値化する

相対見積もりというやり方で基準となる数値を決めて、それとの比較で作業の量を考えていく

基準となるのはプロダクトバックログの項目のどれか。必要な作業量が具体的にイメージできるものにする

基準の作業量は真ん中ぐらいの項目

比較したときにほかが10倍ぐらいに収まるような基準になっているのがいい

相対見積もりはフィボナッチ数を使って見積もる

フィボナッチ数・・・1、2、3、5、8、13、21、34、55、89、...

見積もっていく対象が不確実な前提で見積もりをする

基準より大きな数値になる場合は不確実なことがたくさん含まれていることがわかる

使える数字が限定されるために細かな誤差を気にせずに素早く見積もっていける

見積もりは推測である

見積もるときにちょっとした工夫で不確実なことを減らせるならやっておく

見積もりは推測にすぎないことは忘れてはいけない

プロダクトバックログの項目を詳細にするのは直近の数スプリント分ぐらいでよい

推測と実際の結果を分析する

実際の結果と大きくずれた原因がわかる。自分たちの推測がどの程度正しいのかもわかる

これらはとても重要な情報

周りの期待を満たせそうかも判断できる

プロジェクトの開始前の見積もり

開発チーム全員で見積もりポーカーで見積もる

見積もる項目を1つ選んで、ひとりひとりがどの数字かを考える

合図に合わせて一斉に公開する。

同じ数字にならないときはその数字を出した理由を簡単に話し合う

話し合いに時間がかかる場合なら一番小さい数字と大きい数字を出した人だけ発言するとかにする

開発チーム全員の知恵を持ち寄ってさまざまな視点で意見を出し合うのが大切

実際に作業を進めていく人達が対話する

エース級のひとばかりが発言して対話にならないのは良くない兆候

エースといえども何か見落としているかもしれないし、みんなの認識もズレたまま

その場合には一旦外れてもらってアドバイザーになってもらい、みんなが見落としている点をあとでアドバイスをもらう

ベロシティ

プロジェクトの先のことを整理、計画する

プロダクトバックログの項目を見積もれば、プロジェクトの先のことを考えることができる

プロダクトバックログには実現したいこととそれがどれくらいの作業量になりそうかがポイントという単位で見積もられている
スクラムチームがスプリントごとにどれくらいの作業をこなすことがわかれば様々なことが見えてくる

スプリントごとに終わらせられるポイント数がベロシティ

- 絶対に必要な項目の見積もりの合計 ÷ ベロシティ = 必要なスプリント数(期間)
- ベロシティ × 期間内に実施できるスプリント数 = 実現できるポイント(どこまで実現できそうか)

ベロシティは決めるものではなく、測るもの

ベロシティをスプリント毎に計測していくとより確実な数字になる

ベロシティはプロジェクトの期間中、多少は変動するが作業スピードを根拠もなく想像するよりはずっと確実な数字

ベロシティだけではわかりにくいリリースに関する作業はリリーススプリントで別に準備

リリースに必要な準備の作業は普段の作業とは違うことが多い。

(リリースできるかを判断する部署や組織への申請、セキュリティやパフォーマンスに関するテスト、マニュアル作成など)

スプリント計画ミーティング

これから始まるスプリントの計画をつくるための活動

<第1部>

プロダクトオーナーと開発チームで今回のスプリントでどこまで終わらせられそうか検討をつける

<第2部>

開発チームが中心となって実際の開発作業を洗い出し、より現実的で具体的な計画づくりをする
今回のスプリントで達成するプロダクトバックログの項目を決定する

第1部

プロダクトオーナーから開発チームにそれぞれのプロダクトバックログの項目を説明

よりうまく伝えられるように資料を用意する、ホワイトボードに書くなどをするとよい

開発チームは自分たちに理解が正しいか確認したり、わからないこと/不安に思っていることをどんどん聞いていく。

「終わった」ことをどうやって確認するかを決めておく

それが達成できそうかをプロダクトオーナーと開発チームで相談する

第2部

スプリントをどういう流れで進めていくかをイメージして作業(タスク)の洗い出しをする

タスクを洗い出したら、時間での見積もりを実施する。

時間の見積もりを合計すればスプリント期間内に収まるかどうか判断できる

会議に参加したり、メールを見たりとプロジェクト以外に使う時間もあるので一日で使える時間は5~6時間で見積もる

開発チームがスプリントの期間で実現できると判断したらプロダクトオーナーに連絡して終了

少しできそうにない場合や、難しいと判断した場合はプロダクトバックログの項目をもう一度プロダクトオーナーと調整する

第2部で洗い出したタスクと見積もりをまとめる

スプリントバックログと呼ばれる成果物で日々の進捗の確認に使ったり、問題が起こったときの原因を見つけるために使う

スプリント計画ミーティング

スプリント計画は確実に終わらせられる計画にする

確実に終わるかわからない計画でスプリントを進めても、その結果はすぐに判断される
その結果がよくなければ次はスプリント計画で確実に判断できる項目だけにするという対策になる
怖いのはよくない結果が隠されてしまうこと。

終わっていないのに終わったことにしてしまう

本当はやらなければいけないテストをさぼってしまう

プロジェクトの最後に深刻な問題になって返ってくる

ベロシティが信頼できなくなる。ベロシティはよくないことを知るための重要な手がかり。早い段階なら対策もしやすい
ベロシティが信用できなければプロジェクトの先のことがまったくわからないのと同じ

スプリント計画ミーティングの目的

プロジェクトのゴールに確実に近づくための活動。

リリース日や必要なものを全部揃えるというゴールを守るための計画にすることが目的ではない。

スプリントでは確実にこれだけは終わらせられると確信を持てる計画をつくることが重要

ひとりではなく、スクラムチーム全員の力が必要。

数週間分のことを確実に終わらせていくことでプロジェクトの先のことが守れるようになる

デイリースクラム

同じ時間に同じ場所で毎日開催する

目標が達成できるように毎日15分だけデイリースクラムを実施する

参加するのは開発チームだけでスクラムマスターは必要や要望があれば司会をする。

進め方は開発チームが一人ずつ質問に答えていく

前回のデイリースクラムからやっていたこと

次回のデイリースクラムまでにやること

問題点

スプリントのゴールを守るために毎日検査する

デイリースクラムが誰かへの進捗報告になっている場合がある。

報告に意識が向いてしまうと問題を見つけるのは難しい。そのときは質問を工夫する

「なんで僕のほうを向いて報告するの？」→デイリースクラムの目的を思い出してもらう

「その作業はあとどれくらいで終わるの？」「スプリントレビューの準備は順調？」→問題に気づきやすくする

デイリースクラムで問題を見つけたらすぐに対策

デイリースクラムのあとに必要な人だけが残って話し合って対策する

同じ時間、同じ場所で毎日開催するのは一日分の予想が正しいかを検査する

見つけた問題はすぐに対処する

こうした作業を積み重ねてプロジェクトをうまく進めて行く

スプリントレビュー

スプリント計画ミーティングで決めたプロジェクトバックログをどう実現したかをPOにお披露目
開発チームがそれぞれ実現したものを説明しながらデモ
プロダクトオーナーはそれについて気になったことを質問して問題がないかを判断
問題があれば次のスプリント以降で実現するかどうかから考えていく

スプリントレビューは毎回必ず実施する

プロダクトバックログの項目を実現して確認すると、考えていたことが間違っていることもある
確認が済んだことが大事。確認が済めばそこからいろいろなことがわかってくる。それを元にプロジェクトの方向を修正

実際に出来上がったデモを実際に目で見て判断する

報告は人によって認識がバラバラだから終わったという報告を鵜呑みにするわけにはいかない。
デモをする本当の目的は、実際に動くものを触ってみて本当はどうすべきかを知ること

完了の定義で判断する

完了の定義はスクラムチームで合意する
スプリントレビューはそれをリリースしたときに周りの期待に応えられるかどうかを判断するのが重要
スプリントレビューでは2つの視点で考える
 プロダクトオーナーの視点
 開発チームの視点

問題になる前にみつける

問題になりそうなことならスクラムチーム全員で対処する

問題になりそうなことは、個人でどうにかしようとしがちだが、思っていたよりうまくいかないことぐらいある。作業がうまくいっていないことを人前と言うのはなんとなくイヤだし、周りも自分自身で解決してほしいと思っていたりする。作業はスクラムチームが達成すべきことのためにやっているのだからスクラムチーム全員に影響するかもしれない。スプリント期間は思ったよりも短いのでどんな些細なことでも見逃さないようにしてスクラムチーム全員で対処する。いち早く見つければ作業しているひとにちょっとしたアドバイスをするぐらいで解決する。

スプリントが順調に進んでいるかを確認する

未着手(ToDo)、着手(Doing)、完了(Done)といったタスクの状態を列を分けたタスクボードを貼りだす。スプリント内のタスクがどれくらい未着手なのか、どれが進んでいないのか一目瞭然。縦軸にタスクの見積もり時間の合計、横軸にスプリントの営業日を書いて最終日に0になる理想線を引き、毎日決まった時間に残タスクの見積もり時間の合計を記入したプロットするスプリントバーンダウンチャートを貼りだす。スプリントの進み具合が順調なのか確認する。

プロジェクトをうまくすすめるためにうまくいっていないことを透明化する

プロジェクトの途中でなにかがうまくいっていないと感じたら、スクラムチームが見落としていることがあるかもしれない。まずはそれを透明にすることから始めてみる。

タイムボックスは譲れない

- スプリントの期間は1ヶ月以内
- デイリースクラムは15分以内
- スプリント計画ミーティングは4時間まで(スプリント期間が2週の場合)
※スプリント期間が1ヶ月の場合は8時間まで
- スプリントレビューは2時間まで(スプリント期間が2週の場合)
※スプリント期間が1ヶ月の場合は4時間まで
- スプリントレトロスペクティブは2時間まで(スプリント期間が2週の場合)
※スプリント期間が1ヶ月の場合は4時間まで

制限時間を設けて、その中で必要なことをやり、
時間内に終わらなかったものは次のタイムボックスに回す
(やることをすべて終わらせるまでにどのぐらいの時間が必要かという考え方とは逆)

スプリントの期間を一定にして、実際にどこまでできたかを計測する。(延長は一切なし)
すべての作業を終わらせるのに、どれぐらいのスプリントが必要かを予測する。

スプリント以外のタイムボックスによってスクラムチームの実力がわかる
15分でデイリースクラムが終わらない→検査のイベントとして成立しない
スプリント計画ミーティングが終わらない→実力以上のことを計画しようとしている
スクラムイベントについて理解が不足した状態でプロジェクトを進めているかもしれない

次にやることは誰もが知っている

スプリントの予定よりも早く終わったら、次になにをするのかはプロダクトバックログを見ればわかる。

プロダクトバックログの順序は、作業を円滑に進めていくためにも大切。
普段から順序を見直しておく

次に着手する項目でちょうどいい分量(ポイント)のものがない場合は
項目を分割して着手するものを調整する。
分割するときは、実際に動いているものを触って、確認できる状態で分割しないといけない。

プロダクトバックログの項目追加は誰が実施してもいい。
プロジェクトで実現してほしいことや、もっとこうしたい方がいいということはあるべく多く集めないといけない。
プロダクトバックログはそうしたことがすべて記載されている一覧にしないといけない。

何を本当にやるべきかを最終的に判断して責任を持つのはプロダクトオーナー

自分たちの責務を理解しないとうまくいかない

スクラムイベントだけを形だけやってもうまくいかない

それぞれのスクラムイベントやロールは何のためにあるのだろうか？

なにをすべきだろうか？

スクラムチーム全員で考えていくことで求められていることに応えていける

守り続けるための規律は誰も与えてくれない

規律は自分たちでつくり、それに従い行動し、守れないときは何のための規律が必要なのかを考える

自律したスクラムチームになるためには何度も何度も考えていかなければならない

少しでも理想に近づけていく

問題とは、プロジェクトのゴールを直接脅かしてくるもの

バグが多い

ほかの部署に頼んだ仕事が遅れている

プロダクトオーナーの要求が二転三転する

技術的負債が蓄積している

問題を把握するために、デイリースクラムやスプリントレビューがある

プロジェクトが混乱しているときにはうまくできない

問題をタスクボードのように貼りだして、スクラムチーム全員が問題を知って、それぞれの状況を知る

問題が起きてしまったらなるべく早く対処する

問題は放置すればするほどプロジェクトに与える影響が大きくなる

問題の解決も日々の作業として取り組んでいく

スクラムマスターが妨害リスト作り、管理していく

スクラムチームを理想に近づけていくためのToDoリスト

プロダクトバックログのように順序を付けた一覧にしておく

妨害リストはスクラムチームが見えるところに貼っておき、スクラムチームを巻き込んでいく

妨害を取り除くのは簡単じゃないが、妨害はいつかは大きな問題を生んでしまうのですこしずつでもよくしていく

手戻りを防ぐ

手戻りが起きやすいのは、何を実現したいのかがあいまいなとき
何を実現すればプロジェクトで達成すべきことを満たせそうかを
スプリントで実際に着手する前に確認しておく

- 実現したいことは先に深く理解しておく
- 決めるべき仕様を決めておく
- 技術的にどういうふうを実現するといったかを確認しておく

プロダクトオーナーは開発チームが何を実現するのかを深く
理解できるように理解を助けるための準備をしておく。
(ペーパープロトタイピング、画面のスケッチなどイメージできる材料)

スプリントの準備はスプリント計画ミーティングで洗い出した作業と
並行して進めるために、準備のためのイベントを用意する

スクラムチームで定期的集まって、プロダクトオーナーは次回以降のスプリントで実現し
たいことを伝え、みんなですべてについて話し合う

着手するために必要なことを洗い出して、日々の作業に組み込みスクラムチームで手分け
して、次のスプリントまでに片付ける

いつもゴールに向かう

ゴールに近づくための方法

1. スクラムチームの仕事の進め方をよくする

すぐに効果がでるかはわからない

少しずつ取り組んでゴールに着実に近づいてのに有効

2. 何かを調整してゴールに近づいていく

多少の痛みを伴うが確実な方法

プロジェクトで調整できるものは次の4つ

<品質>

品質は調整が最も難しい

品質をどうするかはプロジェクトをはじめたときに決まっている

プロジェクトの状況が良くないからといって品質を犠牲にしてはいけない

<予算>

追加の予算があれば即効性のある対策を打つのに有効

今より早い開発マシンを支給してもらう

作業により集中できる作業スペースを与える

人員を増やすのに使っても即効性のある対策にはならない

新しく来た人がスクラムに慣れたり、一緒に協力して進められるようになるのに時間がかかる

<期間>

期間の調整は1度限りならできるけれど大きく調整するのは困難

<スコープ>※一番調整しやすい

プロダクトバックログを何度も見て、単なる思いつきや、なくなってもどうにかなるものが見つかるはず

さっさとあきらめるか、優先順位を下げて余力があれば実現することにしてしまう

どう実現するかに関係をつけて調整する

協力して乗り越える

開発チームは自己組織化されていて、機能横断的であることが求められている
実現したいことには多少のバラつきがあってもそれらをうまく整理してタイムボックスまでに終わらせる
作業中に何か困ったことがあっても自分たちで解決する
この作業をしなさいと指示がなくても大丈夫で必要なスキルもすべて備えている仕事のできる集団

自己組織化とは自分たちで状況に応じて役割を決めていくこと
そのときどきの状況で最も力を発揮できる人が、リーダーシップを発揮して開発チームを引っ張っていく

開発チームがプロジェクトを進められるだけのスキルがあるのかスキルマップを書いてみる
プロジェクトに必要なだと思うスキルや知識を列挙
(プログラミング言語、ミドルウェア、開発環境、要件定義、設計の経験、プロジェクトの状況説明、社内の調整など)
それぞれの開発メンバーは「得意」「経験有」「未経験」と書き込んでいく
開発チームの得意なこと、苦手なことが見えてくる
必要なスキルが足りない場合や大きく偏っている場合はプロジェクトの関係者と相談しなければならない

開発チームがどういうスキルを持っているかを理解しておく
下記のようなことを話し合っておく
これまでどんなプロジェクトにいたか
自分が仕事をするうえで大切にしていること
自分に期待されていることは何だと思うか

一人ひとりが得意な作業だけをやっていてもダメ
スクラムが求めているのは「私はここまでしか担当しません」ではなく、みんなで協力して作業を進めていくような開発チーム
プロジェクトを進めていけば難しい問題に何度も遭遇するが安定して作業は進めなければいけない。
一人ひとりが優秀であることより大切なことは誰かが困っていたらほかの人が助ける
どんな状況でもすぐに相談できて、それが良くない状況ならすぐに協力して乗り越えていく

自己組織化はタスクから

自己組織化の最初の一步は、
開発チームがどのタスクに取り組むかを自発的に決めていくこと

失敗から学んでいこう

Scrumに代表されるアジャイル開発では、コミットメントという言葉を大事にしている

コミットメントとは責任を伴う約束のこと

開発チームはスプリントのゴールを守るために全力を尽くす義務がある

コミットメントを表明するのは開発チームだけなので他の人が口をはさんではいけない

実際に作業に関係のない人の発言は参考意見に過ぎない

プロジェクトでは、さまざまな判断をしていく必要がある、そして

それらは実際にプロジェクトを進めるスクラムチームでないとできない

それらの判断は現場のさまざまな情報をもとにしないと的確にできないから

(ビジネスの状況、プロジェクトの周辺の状況、要件や仕様の決まり具合、メンバーの状況)

判断を的確にやっていくのは簡単じゃない

プロジェクトの大事な判断をしていくには責任を持って取り組まないといけない

プロジェクトの大事な判断ができるようにコミットメントを表明する機会を数多く用意

約束をすることで責任感を芽生えやすくする

コミットメントには良くない面もある、約束を守ろうとして無理をしてしまう

約束を守ることばかり意識して技術的負債を溜め込んだり、スクラムチームが疲弊

無理やり約束させられてしまうこともある

コミットメントは自分たちがやるべき作業にベストを尽くして取り組んでいくためにある

約束を守っていけるようになるには

まずは、責任を持って約束できるようになろう

自分たちが自信を持ってやれると判断できるようになっていく
失敗しても構わないから、自分たちで判断してみる。
その判断が間違っていたら、なぜ間違ったのかを考えればいい
そうすれば、次の機会には自信を持って判断できるようになる

大切なのは失敗から学ぶこと

失敗は絶対に許さないなんて言われたら、失敗しないことばかり考えてしまう
失敗をしなければ何も学べないし、スクラムチームは成長しない

プロジェクトの最初と最後でスクラムチームの実力が同じではいけない
プロジェクトはどんどん大変になっていく。仕様もコードもどんどん増えていく
失敗を許さずにチームが成長しないというのはとても危ない

スクラムチームが自信を持ってくると、プロジェクトに良い影響を与える
仕事にもっと前向きに取り組める
出来上がるものも良くなっていく

Scrumで求めているのは、コミットメントを必ず果たしてくれるスクラムチームではなく
責任を持って取り組んでくれるスクラムチーム

Scrumで大人数による開発を行うには？

Scrumでは開発チームは10人未満

必要な分の開発チームとスクラムマスターを用意する

意思決定を明確にするためにプロダクトオーナーは一人でないといけない

プロダクトバックログもひとつ

(プロダクトオーナーを支える専門の体制を用意することが多い)

スクラム・オブ・スクラムというイベントを用意

開発チームごとにデイリースクラムを実施したあと代表者が集まって、全体のことについてなにかうまくいっていないかを検査する

大人数での開発は、少人数での開発と比べて扱う問題の大きさも量も桁外れ

全体がうまくいっているかどうか頻繁に確認する必要がある

それぞれの開発チームがScrumのやりかたに十分慣れている前提

少なくとも開発チーム内の問題は自分たちだけで解決できるべき

開発チームと全体の問題が大量に押し寄せてプロジェクトはうまく進まない

そんな開発チームがたくさんないなら、大人数での開発はやめた方が無難

それでもスクラムを取り入れていきたいなら、慣れたやり方で途中まで進めて、ある程度まで作ってしまう。

そしてその期間にScrumのやり方を少しでも取り入れて問題を解決していく練習をする

Scrumをやったからってうまくいくわけじゃない

Scrumで成果を出せるかどうかは、実際に取り組んでいく現場の人たち次第
プロジェクトで期待されていることを達成するために、自分たちが確実にできることを少しずつやっていく
その結果がどうなったかを受けて、次を考えていく

Scrumはうまくいっていないことを見逃さず、対処しやすいように以下のことを提供している

- ・どこがうまくいっていないかを特定しやすい
- ・実際にうまくいっていないことを解消する機会を与えている
- ・うまくやるためのやり方に変えられる余地がある
- ・やり方を多少変えても影響が少ないようになっている

現場の人たちが中心となって問題になりそうなことを見つけて、
仕事の進め方から見直して解決していく活動

→カイゼン活動

日本の製造業で生まれた現場の作業者が中心となって現場を良くし続けていく活動は
Scrumの中でも重要な要素になっている

プロジェクトに常に問題があったら、その対処に翻弄されて、作るものをよくしていくことができない
プロジェクトが常に問題がない状態にするのが不可欠

Scrumの本質は体験して学んでいくための仕組み