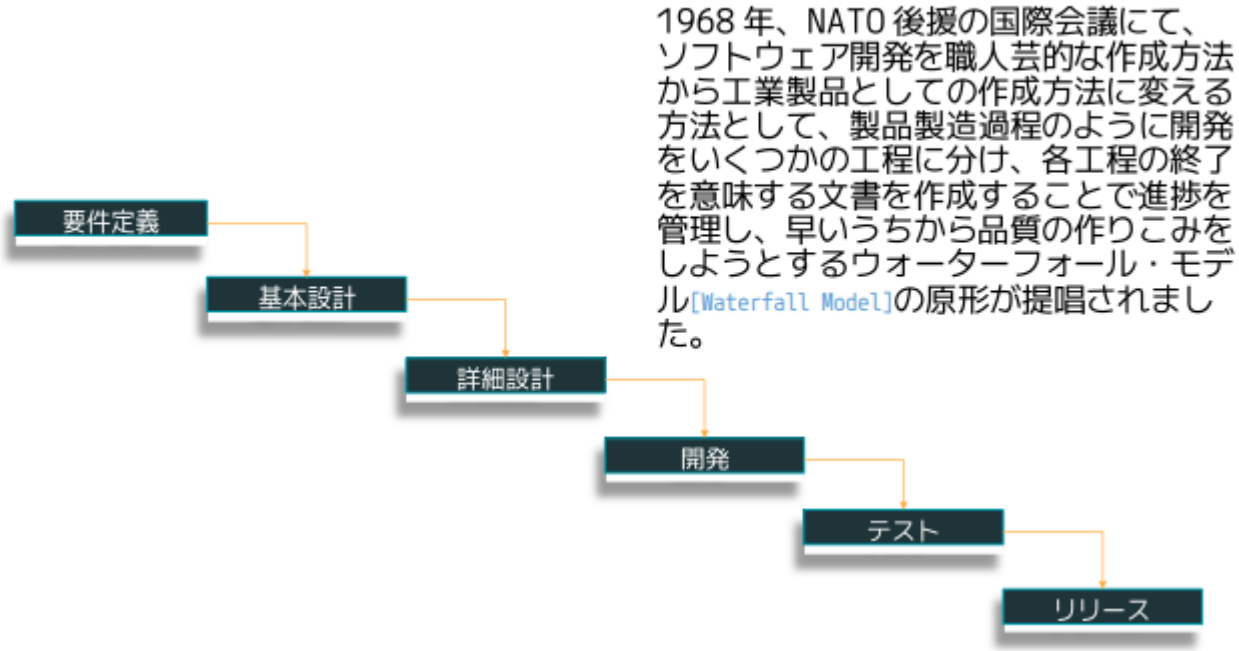
Class 7 -5

システム開発

ウォーターフォール

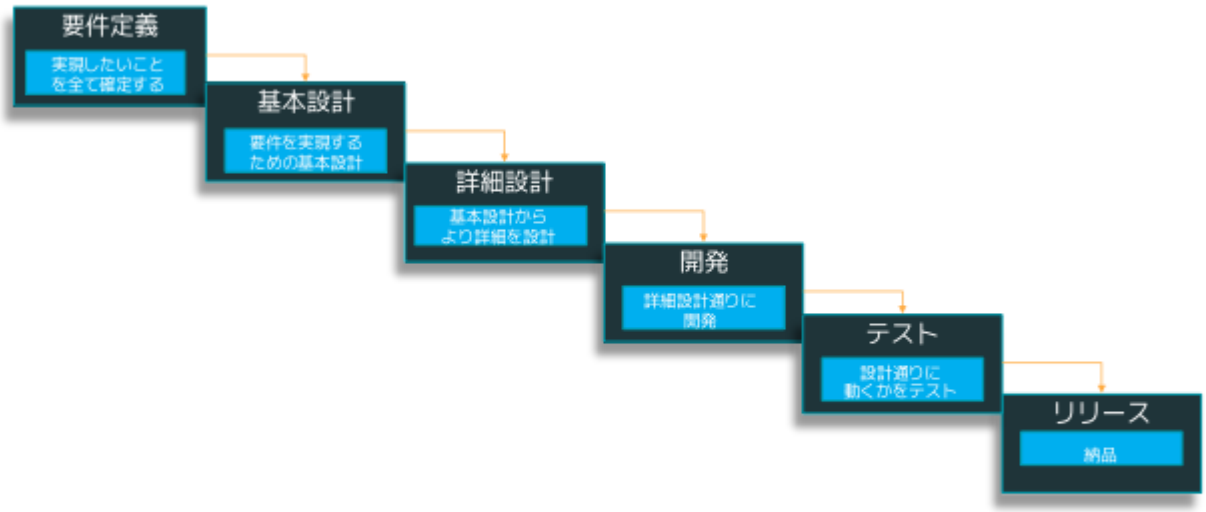


今回は、開発の手法について説明していきたいと思います。

ウォーターフォールモデルは、開発手法としては、最も古くからあるものです。

用件定義から設計、開発（プログラミング)、テストへと、各工程を順番に進めていきます。

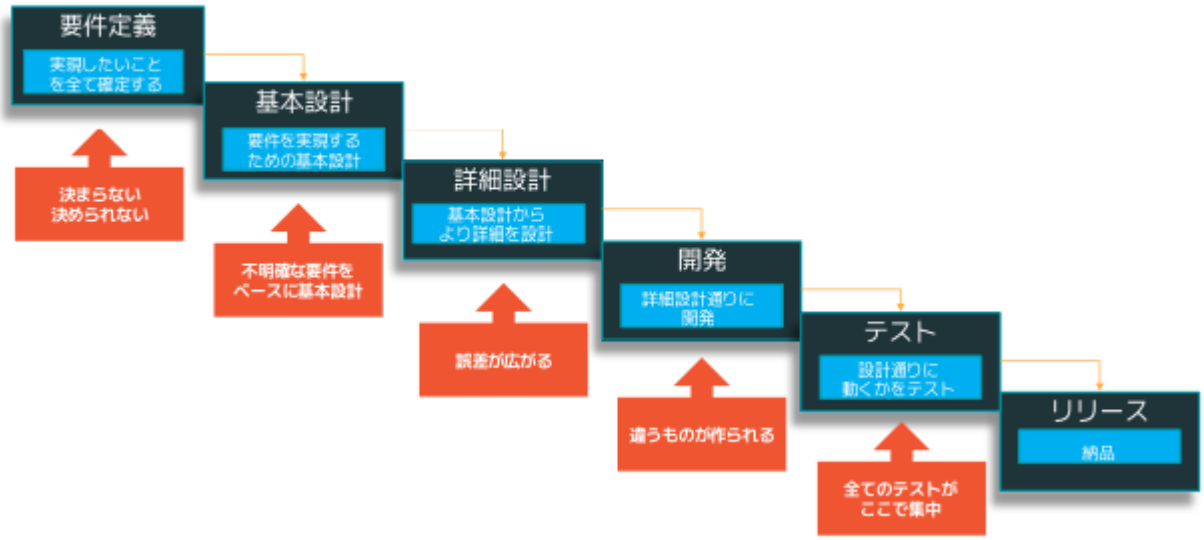
具体的に



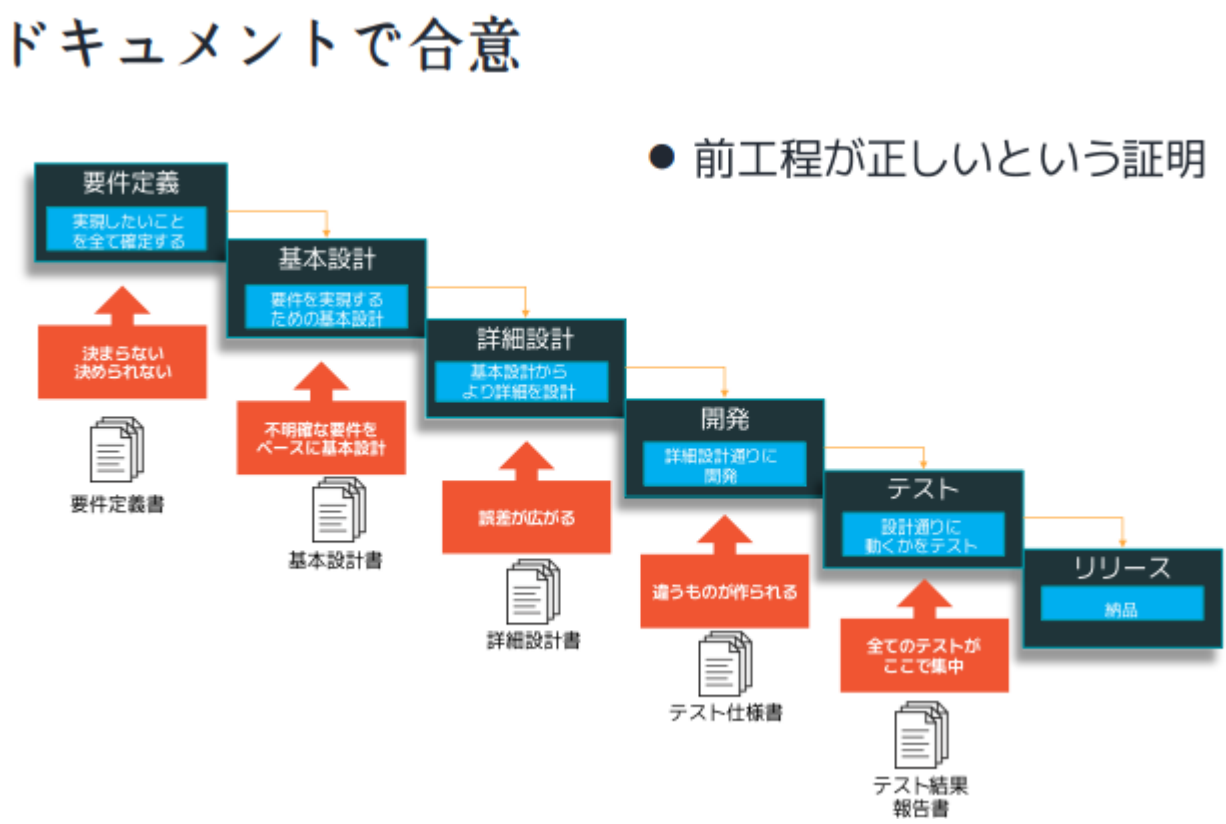
前の工程が正しいことが前提

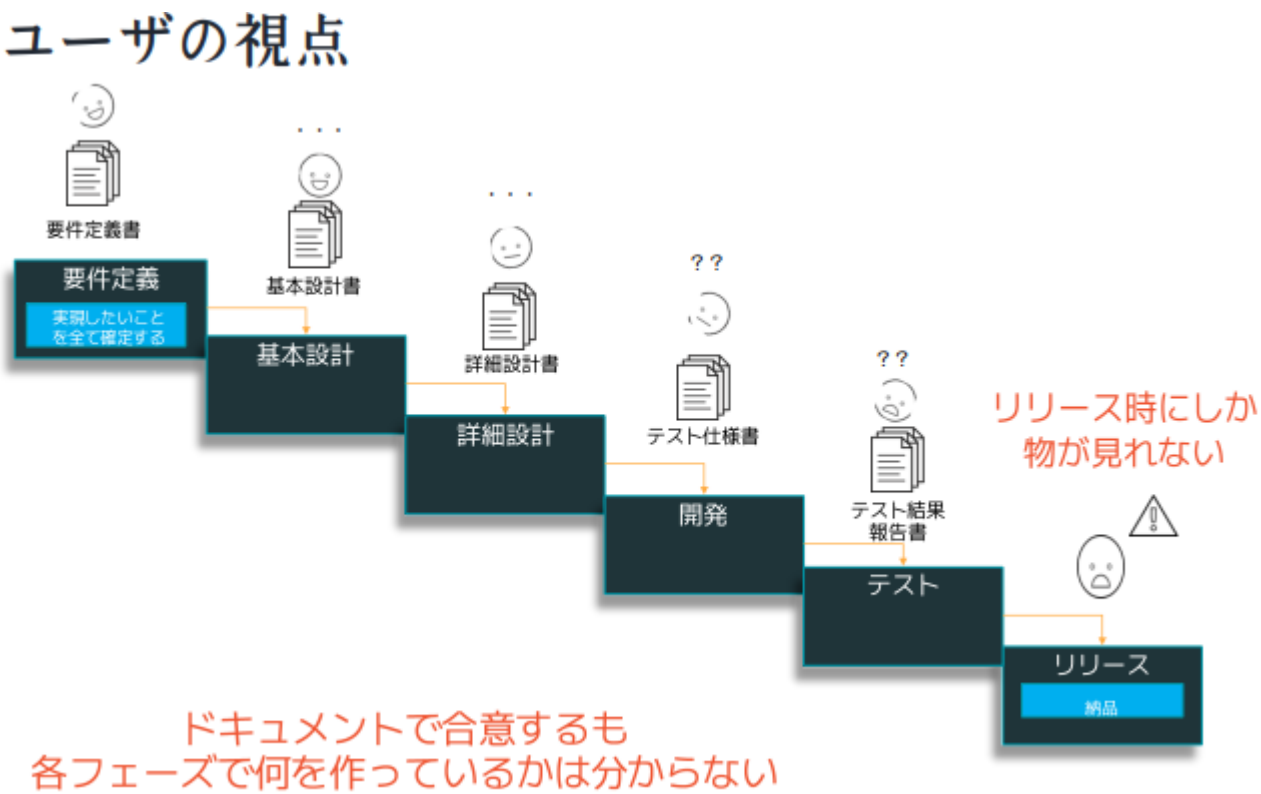
それぞれの工程を完了させてから次に進むので管理がしやすく、大規模開発などで広く使われています。

ウォーターフォールの失敗要因



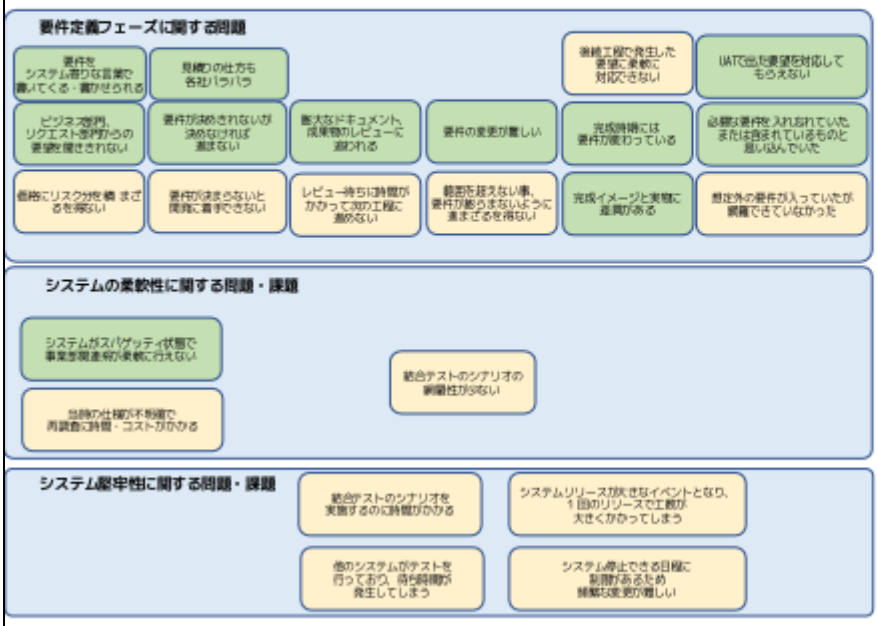
ただし、必然的に利用者がシステムを確認できるのは、最終段階に入ってからです。しかも、前工程に戻って作業すること。このことを「手戻り」というのですが、その手戻りを想定していないので、いざ動かしてみて、「この仕様は想定していたものとは違う」という話になると、とんでもなく大変なことになります。





設計書などをユーザーに見せて、合意してもらいますが、実際にユーザーさんは、システムを公開（リリース）したときしかものを見ることができません。

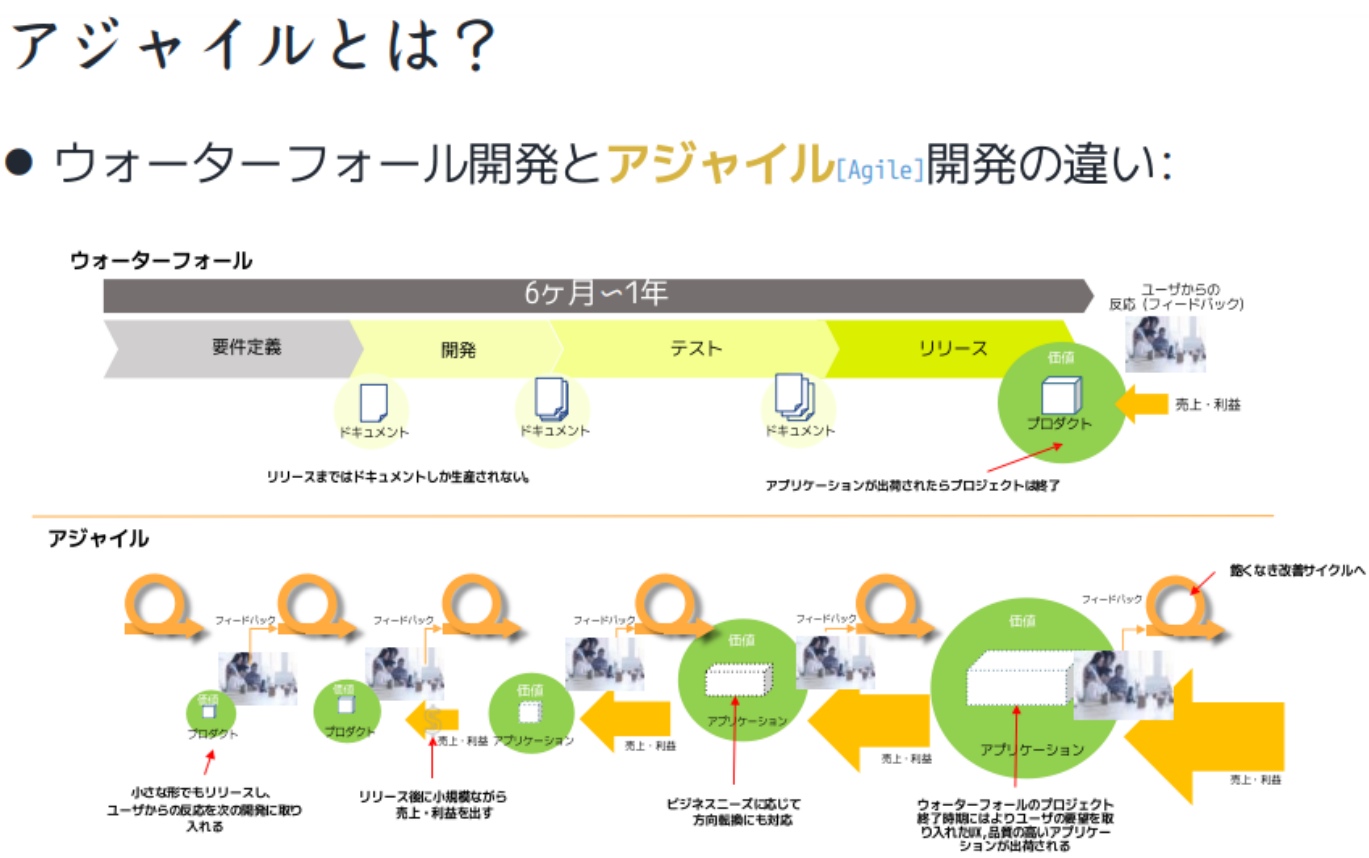
ウォーターフォールによる問題



1 ウォーターフォール開発の問題点
- 前の工程に問題がない事が前提
- 各工程の完全性はドキュメントにより証明
- 要件の変更は前提が崩れるため、困難
- 「完成」は最後の工程終了後

2 アーキテクチャの問題点
- 複雑化、スパゲティ状態
- 変更の影響範囲が大きい・見えない
- 過去のプロジェクトで実装した仕様はもはや誰も知らない
- 今後も頻繁な変更は難しいため、できるだけ長く使えるように要件を詰め込んで作る

3 計画的リリースの問題点
- システムの停止調整
- リリースのためのリハーサル、リリーステストの計画、実施
- 本番環境へのリリース承認
- リリーススタッフの確保



前章で紹介したウォーターフォールモデルは、「伝統的」といっても良い旧来からある存在です。

開発の基本的な流れを抑えるときには、今も無視できない開発手法です。

しかしながら、コンピュータの利用が多岐にわたり、ネット上のサービスも多種多様に生まれては、

消えていく現在では、「より早くコンパクトに」「より少人数で」等、開発現場には前にもましてス

ピード感が求められています。そうすると、当然開発手法の側も、それに応じた変化が求められま

すよね。その一例が「アジャイル」です。

アジャイル開発は、『計画→設計→実装→テスト』といった開発工程を**機能単位の小さいサイクルで繰り返すのが最大の特徴**です。

優先度の高い要件から順に開発を進めていき、開発した各機能の集合体として 1 つの大きなシス

テムを形成。「プロジェクトに変化はつきもの」という前提で進められるので**仕様変更に強く、プロダクトの価値を最大化することに重点を置いた開発手法**です。

アジャイル開発宣言

https://agilemanifesto.org/iso/ja/manifesto.html

アジャイル開発という概念が生まれたのは、2001 年。アメリカ・ユタ州に集まった 17 名の技術者・プログラマーによって提唱されたのが始まりです。

彼らがより良い開発手法について議論するなかでまとめられたのが「アジャイルソフトウェア開発宣言」です。

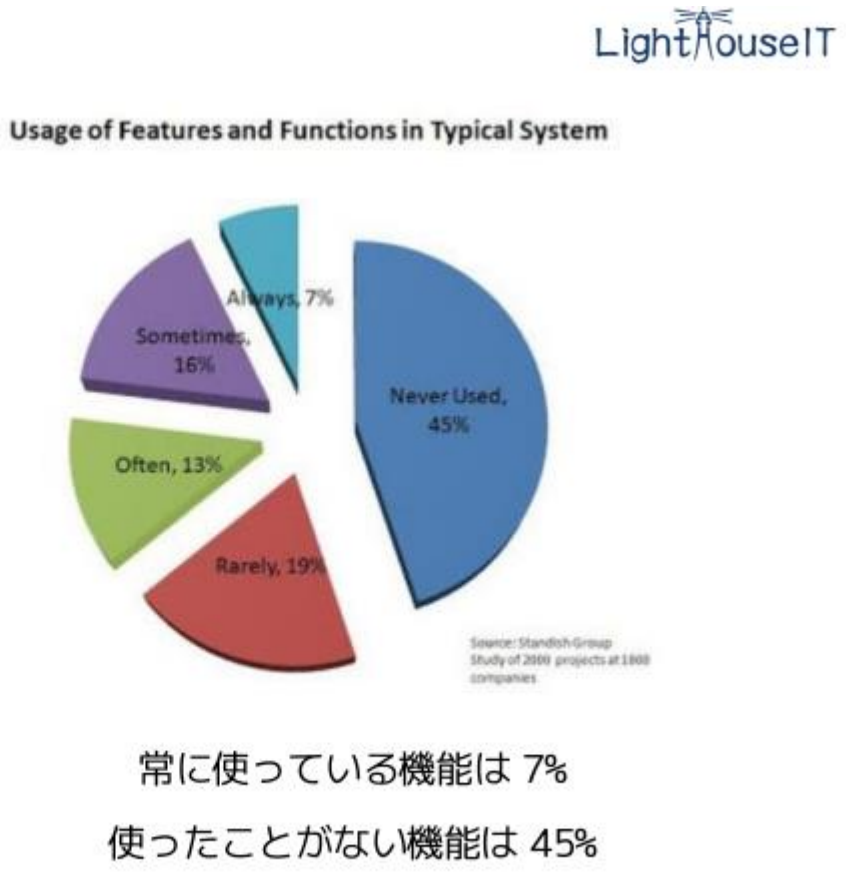
一部文字が大きくなっていますが、これは**「ソフトウェア開発において、どこに重きをおくか」という価値観を定義したもの**。

アジャイルマニフェストの背後にある 12 の原則

顧客満足を最優先し、 価値のあるソフトウェアを 早く継続的に提供します。	意欲に満ちた人々を集めてプロジェクトを 構成します。環境と支援を与え仕事が 無事終わるまで彼らを信頼します。	技術的卓越性と優れた設計に対する 継続的な注目がアジリティを高めます。
変更要求を歓迎します、 たとえ開発の終盤であっても。 アジャイルプロセスはお客様の競合優位性の ための変化を制します（意訳）。	情報を伝えるもっとも効率的で効果的な方法は フェイス・トゥ・フェイスで話をすることで す。	シンプルさ （ムダなく作れる量を最大限にすること）が 本質です。
動くソフトウェアを数週間から数ヶ月など、 より短い期間を選択して 定期的にデリバリーします。	動くソフトウェアこそが 進捗の最も重要な尺度です。	最良のアーキテクチャ・要求・設計は、 自己組織的なチームから生み出されます。
ビジネス側の人と開発者は、 プロジェクトを通して 日々一緒に働かなければなりません。	アジャイル・プロセスは 持続可能な開発を促進します。 一定のペースを継続的に維持できるように しなければなりません。	チームがもっと効率を高めることができるかを 定期的に振り返り、それに基づいて 自分たちのやり方を最適に調整します。

アジャイルソフトウェア開発における 12 の原則「アジャイル宣言の背後にある原則」というのが、

アジャイル開発についてさらに詳しく解説されています。



無駄ということで、実際使っている機能は 7 %で、使ったことがない機能は 45%と驚きですよね

アジャイル開発の手法

- アジャイル開発では、様々な開発手法が利用されます。ここでは代表的な 3 つの手法を紹介します:

1. **スクラム**
   スクラムは、アジャイル開発の中でも有名な手法で、開発を進めるためのフレームワークを指します。スクラムとはラグビーで肩を組んでチーム一丸となってぶつかり合うフォーメーションのことで、その名の通り、チーム間のコミュニケーションを重視している点が特徴です。

2. **エクストリーム・プログラミング**
   エクストリーム・プログラミング[Extreme Programming]は **XP** とも略され、事前に立てた計画よりも途中変更などの柔軟性を重視する手法です。

3. **ユーザー機能駆動開発**
   ユーザー機能駆動開発[Feature Driven Development, FDD]は、実際に動作するソフトウェアを適切な間隔で繰り返す手法で、顧客にとっての機能価値（Feature）という観点で開発が進められているのが特徴です。

スライドそのまま読み上げる

スクラム

- スクラム[Scrum]は、竹内弘高氏、野中郁二郎氏が 1986 年にハーバードビジネスレビュー誌にて発表した『*The New New Product Development Game*』という論文を元に、Jeff Sutherland 氏らが 1993 年に考案、適用したアジャイル型開発手法の 1 つです。

- この開発技法は、アジャイル型開発技法の中でもチーム一体となってプロジェクトを遂行して行くことに重点を置いている、という特徴があります。この開発技法の名前からも、顧客も巻き込んでスクラムを組んで進んでいく、という光景が目に浮かんできます。

スライドそのまま読み上げる

- メンバーが自分たちで計画を立案し、イテレーションごとに開発の進行に問題がないか、制作物は正しい動きをしているのかを精査します。そのため、メンバー間でのコニュニケーションが重要で、コミュニケーションが不十分になると、イテレーションの制作物としてリリースができなかったり、リリースした機能が正常な動きをしなかったりするといった問題が生じる可能性があります。スクラムを組むように、チーム全員が協力して開発を進めることが大切です。

スライドをそのまま読み上げる

スクラムのプロセス

- スクラムでは、チーム一体となって活動するために、以下の様なプロセスが定義されています。
  - **デイリースクラム（朝会）**
  - **リリースプランニング（プロダクトバックログ）**
  - **スプリントプランニング（スプリントバックログ）**
  - **スプリント（イテレーション開発）**
  - **スプリントレビュー（デモ）**
  - **ふりかえり**

で、そんなスクラムのプロセスなのですが、スライドの通りなんですけど、正直、わからない用語がたくさんでているので、一つずつ解説をしていきますね。

プロセスの定義

- **デイリースクラム**:
  毎朝チームメンバーで集まり、15 分など短い時間を区切り、昨日やったこと、今日やること、障害となっていることを一人一人報告・共有します。このミーティングにより、チーム全員の状況、障害や問題の共有、今日どこまで完了するかの宣言を行い、プロジェクト全体の見える化を行います。

- **リリースプランニング**:
  プロジェクト立ち上げ時に、どのようなプロダクトをどのような機能優先順で、どのくらいの期間で実施するかをチーム全員でプランニングします。万が一実態とのずれが大きくなってきた場合には、随時見直しを行います。プランニングを実施した結果は、『プロダクトバックログ』という名前の、機能優先順で並べた機能一覧により管理します。

- **スプリントプランニング**:
  ひとつのイテレーション期間（1 ~ 4 週間程度）で、全体のプロダクトバックログの中からどの範囲の機能を実現するかをチーム全員でプランニングします。プランニングした結果は、『スプリントバックログ』という名前の、機能優先順で並べた機能一覧により管理します。チームが毎スプリント内でどのくらいのタスクを消化できるかは、毎回計測し、精度を高めて行きます。

- **スプリント**:
  実際のひとつのイテレーション期間の開発フェーズです。チームメンバーは、宣言通りにスプリント内で安定したプロダクトがリリースできるよう、最善を尽くします。基本的に、スプリント内の変更（機能の追加や変更、削除）は認められません。

※イテレーションとは：

一連の工程を短い期間にまとめ何度も繰り返すことで次第に完成度を高めていくアプローチが取られることがあるが、この**繰り返しの単位となるサイクルのこと**をイテレーションということがある

- **スプリントレビュー**:
  ひとつのイテレーション期間で完成したプロダクトを、ステークホルダを集めデモを行います。このフェーズでは、チームメンバー達の作ったプロダクトが安定して動くことをアピールすると共に、要求の伝達ミスや漏れがないことを必ずチェックし、チームメンバー全員が正しい方向を向いてイテレーション開発を進めているかを確認します。

- **ふりかえり**:
  基本的に毎スプリント終了時に行います。ふりかえりには *KPT* 法などが用いられます。今回のスプリントの良かったこと、問題点、挑戦したいことをメンバー全員で出し合い、次のスプリントでさらにチームメンバーが高い価値を生み出せるように、メンバー同士話し合いを行い、確認し合います。

スライド通り読み上げる

メンバーの役割

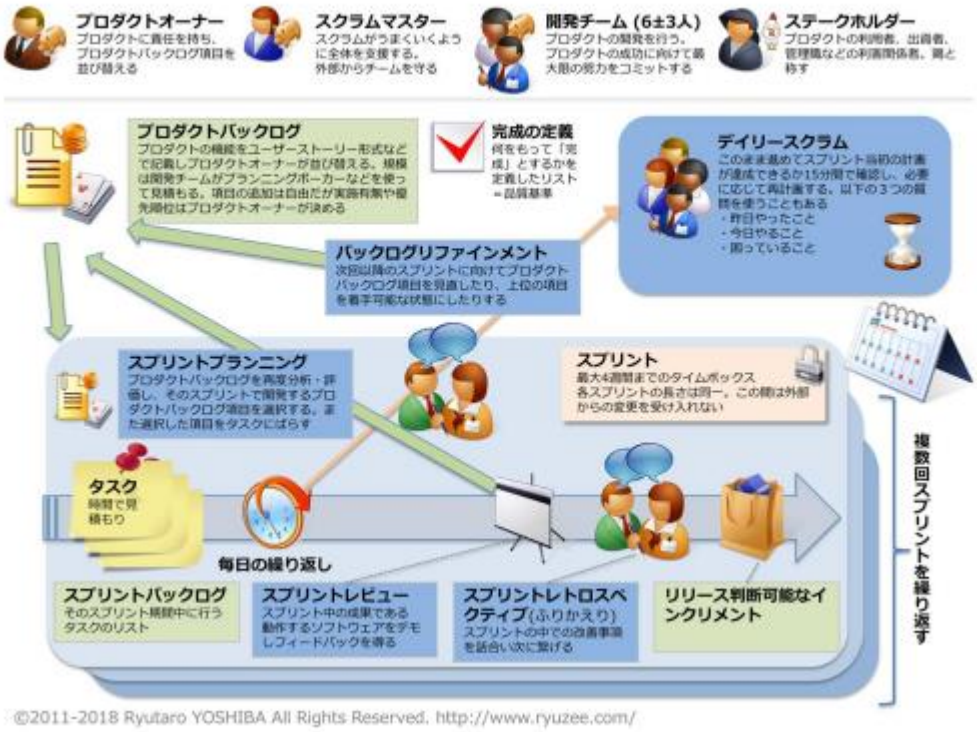
- **プロダクトオーナー**:
  作成するプロダクトに対する、最終決定権と責任を持つ人です。プロダクト全体の機能優先順である『プロダクトバックログ』を常に最新の状態に管理する責任があります。同時にプロダクトに対する情熱を持ち、スクラムチームと綿密な議論を交わし、プロダクトの価値を最大限にしようと常に努力します。

- **スクラムマスター**:
  スクラムの理解と実践を推進し、プロジェクトを円滑に進めることに責任を持つ人です。スクラムをスクラムチーム全員が正しく理解した上で開発を実践していることを常にチェックします。また、チームの自律的な行動を引き出し、成果を最大限に引き出すことに注力します。

スライド通りに読み上げる

- **チーム**:
  ウォーターフォール型開発では、概要設計者・詳細設計者・実装者・テスターなど役割が決まっていますが、スクラム開発では全ての開発に関わる人を『チーム』と呼びます。役割として名前を分けず、『チーム』と呼ぶのは、チーム一丸となって最大の価値を提供する、ということに重点を置くためです。

役割の具体例



最後に全体像を見ておきましょう

他の手法

- **エクストリーム・プログラミング**（XP）：
  開発チームでは「コミュニケーション」「シンプル」「フィードバック」「勇気」の 4 つの価値を共有することを推進しており、中でも「勇気」は、開発途中の仕様変更や設計の変更に立ち向かう勇気を指しています。初期の計画よりも技術面を重視しているため、プログラマー中心の開発手法といえます。

- **ユーザー機能駆動開発**（FDD）：
  実際に動作する機能を開発するには、ユーザー側のビジネスの見える化を行います。そのため、事前にビジネスモデリングを実施する必要があります。

スライド通り読み上げる

最後管理については、git 以外は、スライドを読み上げて簡単な紹介で問題ない。

ただ、リポジトリとかブランチとか、聞いただけだとイメージがわかない単語が出てくるのでそこだけ要注意！

リポジトリ（英：repository）とは

ファイルなり、プログラムなり、設定情報なり、何らかの「保管場所」をカッコ付けて言った表現

マージ（英：merge）とは

何かと何かを「結合する」とか「統合する」とか「合体する」とか、そんな感じの意味

です。

コミット（commit)とは

「ここからここまでワンセット」な処理（トランザクション処理）において、あれやこれやとやった結果を確定させること