Woodland Academy

7.5 システム開発

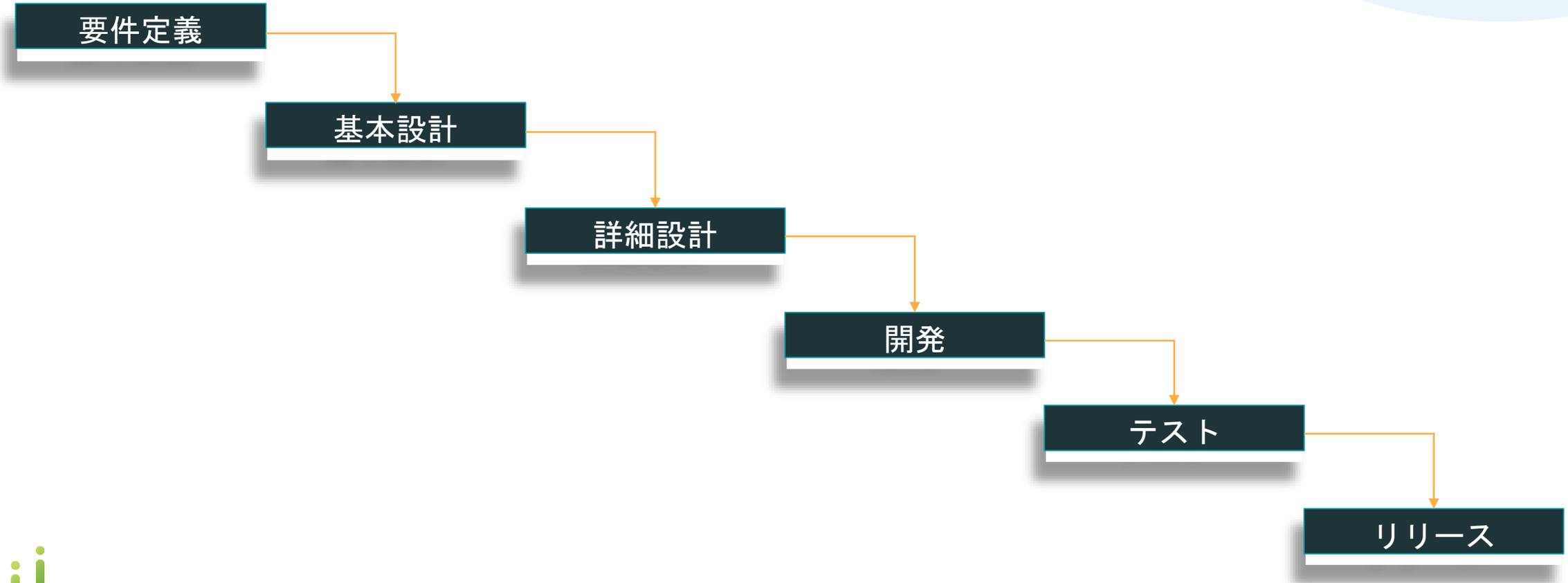
- ウォーターフォール
- アジャイル
- プロジェクト管理ツール

Shape Your Future

目次

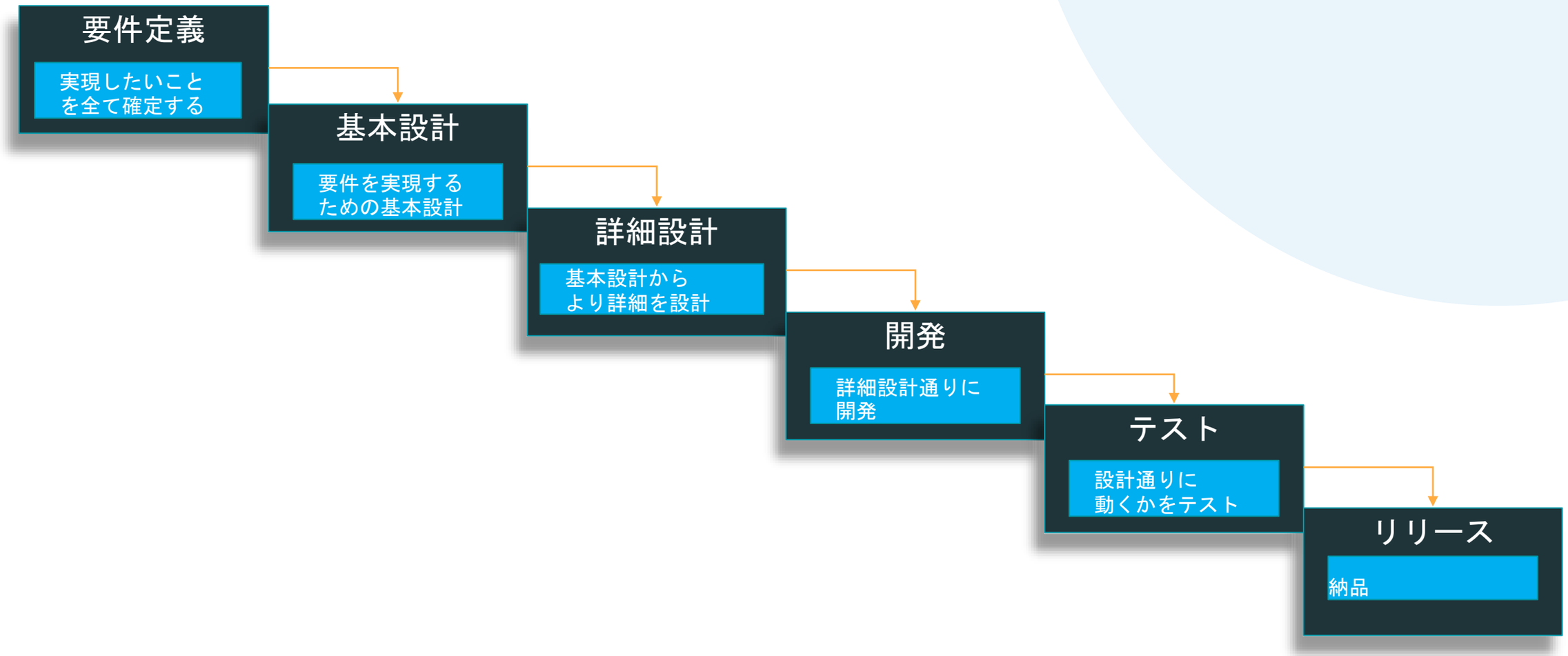
ウォーターフォール

1968 年、NATO 後援の国際会議にて、ソフトウェア開発を職人芸的な作成方法から工業製品としての作成方法に変える方法として、製品製造過程のように開発をいくつかの工程に分け、各工程の終了を意味する文書を作成することで進捗を管理し、早いうちから品質の作りこみをしようとするウォーターフォール・モデル[Waterfall Model]の原形が提唱されました。

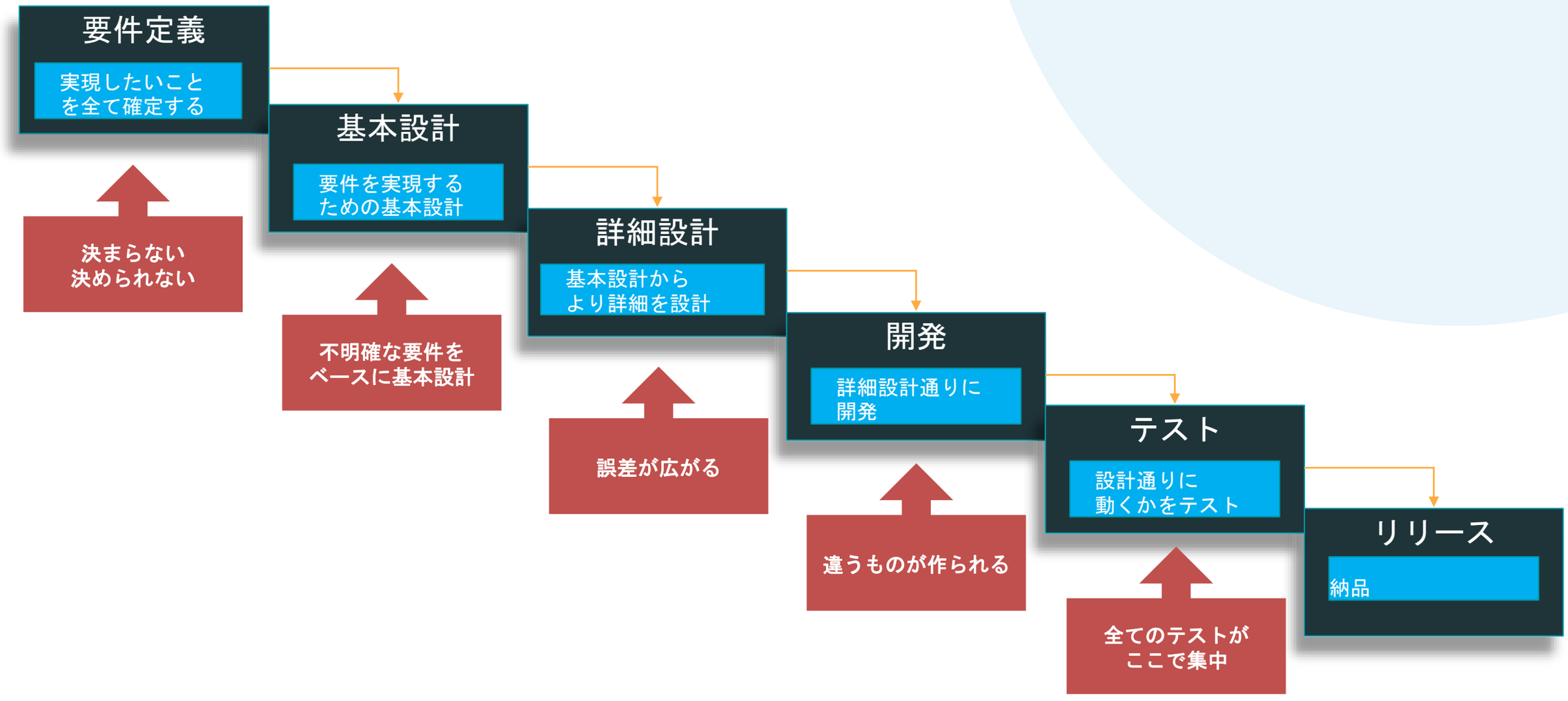


具体的に

- 前の工程が正しいことが前提

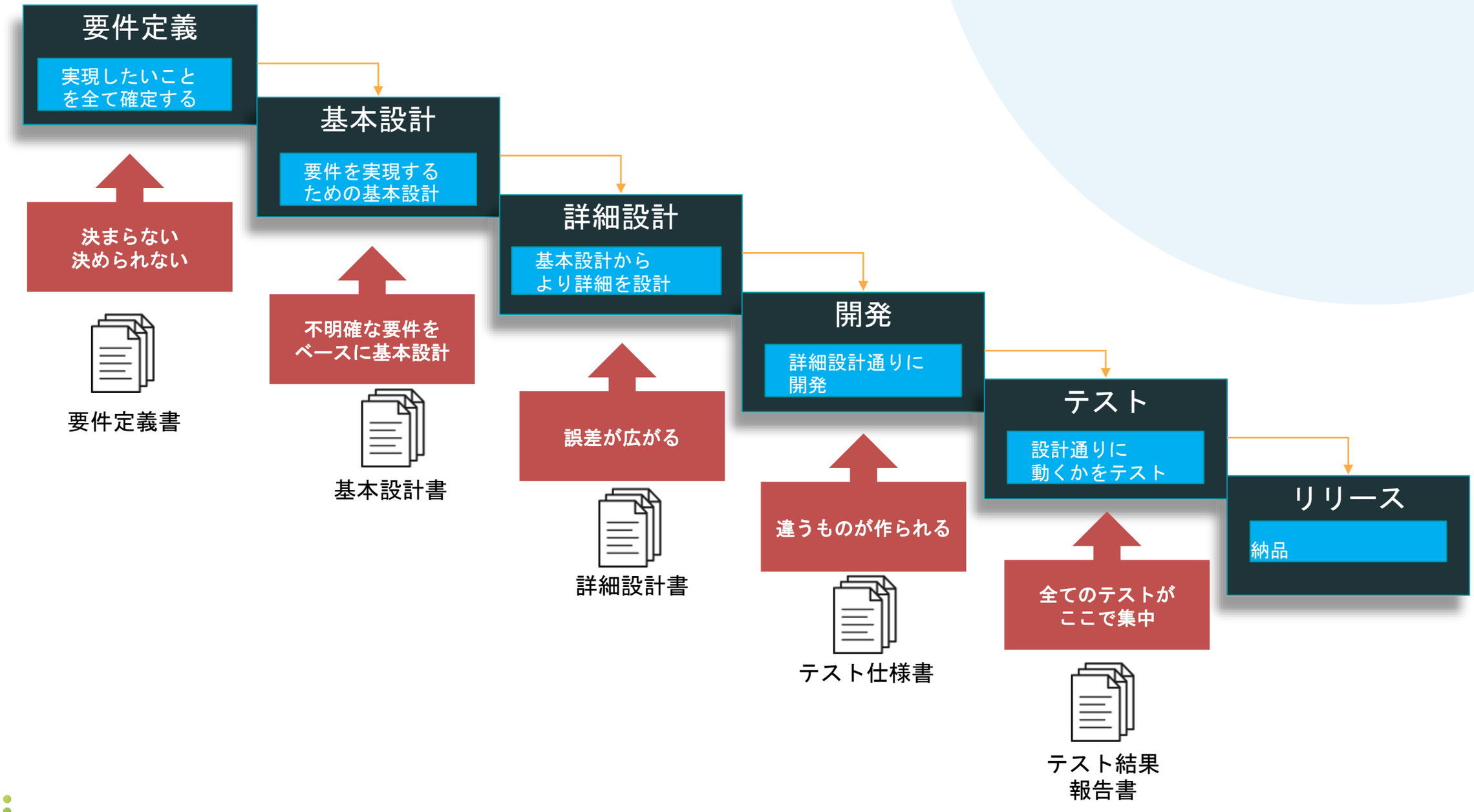


ウォーターフォールの失敗要因

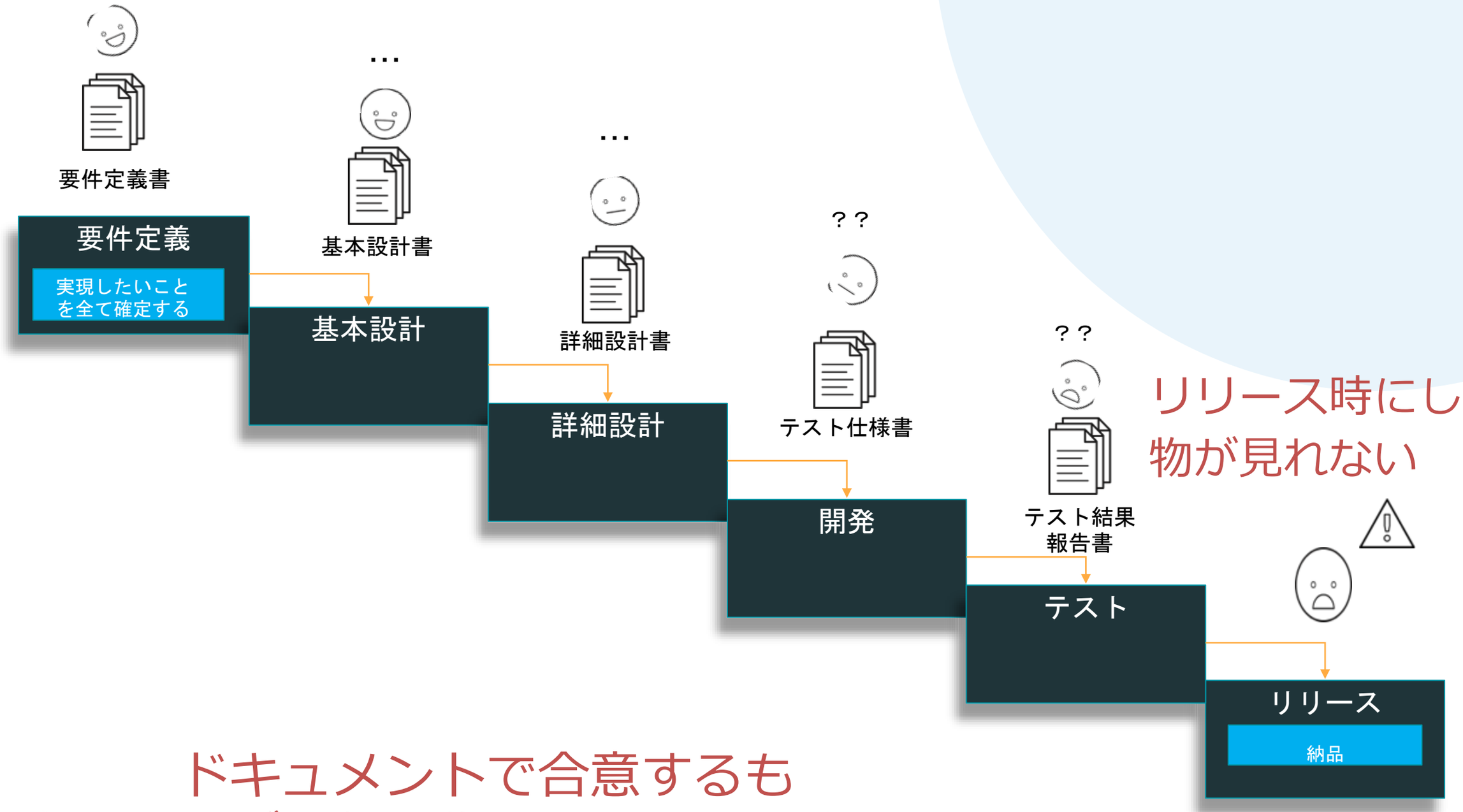


ドキュメントで合意

- 前工程が正しいという証明



ユーザの視点



ドキュメントで合意するも
各フェーズで何を作っているかは分から
ない

ウォーターフォールによる問題

要件定義フェーズに関する問題

- 要件を システム寄りな言葉で 書いてくる・書かせられる
- 見積りの仕方も 各社バラバラ
- ビジネス部門、 リクエスト部門からの 要望を聞ききれない
- 要件が決めきれないが 決めなければ 進まない
- 膨大なドキュメント、 成果物のレビューに 追われる
- 要件の変更が難しい
- 後続工程で発生した 要望に柔軟に 対応できない
- UATで出た要望を対応して もらえない
- 完成時期には 要件が変わっている
- 必要な要件を入れ忘れていた、 または含まれているものと 思い込んでいた
- 価格にリスク分を積 まざる を得ない
- 要件が決まらないと 開発に着手できない
- レビュー待ちに時間が かかって次の工程に 進めない
- 範囲を超えない事、 要件が膨らまないように 進まざるを得ない
- 完成イメージと実物に 差異がある
- 想定外の要件が入っていたが 網羅できていなかった

システムの柔軟性に関する問題・課題

- システムがスパゲッティ状態で 事業部関連系が柔軟に行えない
- 当時の仕様が不明確で 再調査に時間・コストがかかる
- 結合テストのシナリオの 網羅性が少ない

システム堅牢性に関する問題・課題

- 結合テストのシナリオを 実施するのに時間がかかる
- システムリリースが大きなイベントとなり、 1回のリリースで工数が 大きくかかってしまう
- 他のシステムがテストを 行っており、待ち時間が 発生してしまう
- システム停止できる日程に 制限があるため 頻繁な変更が難しい

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

Woodland Academy

Woodland Academy

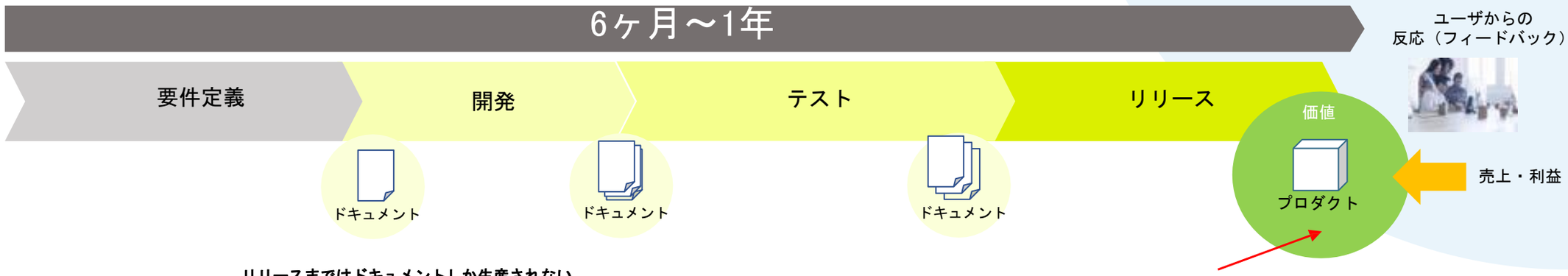
Q&A

目次

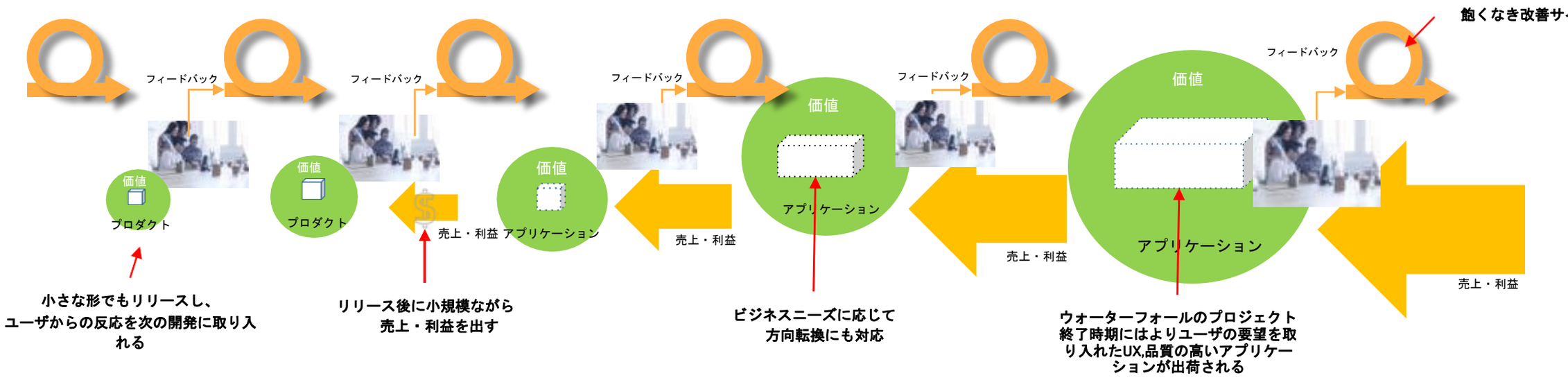
アジャイルとは？

- ウォーターフォール開発と**アジャイル**[Agile]開発の違い：

ウォーターフォール



アジャイル



Woodland Academy

アジャイル開発宣言

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

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。

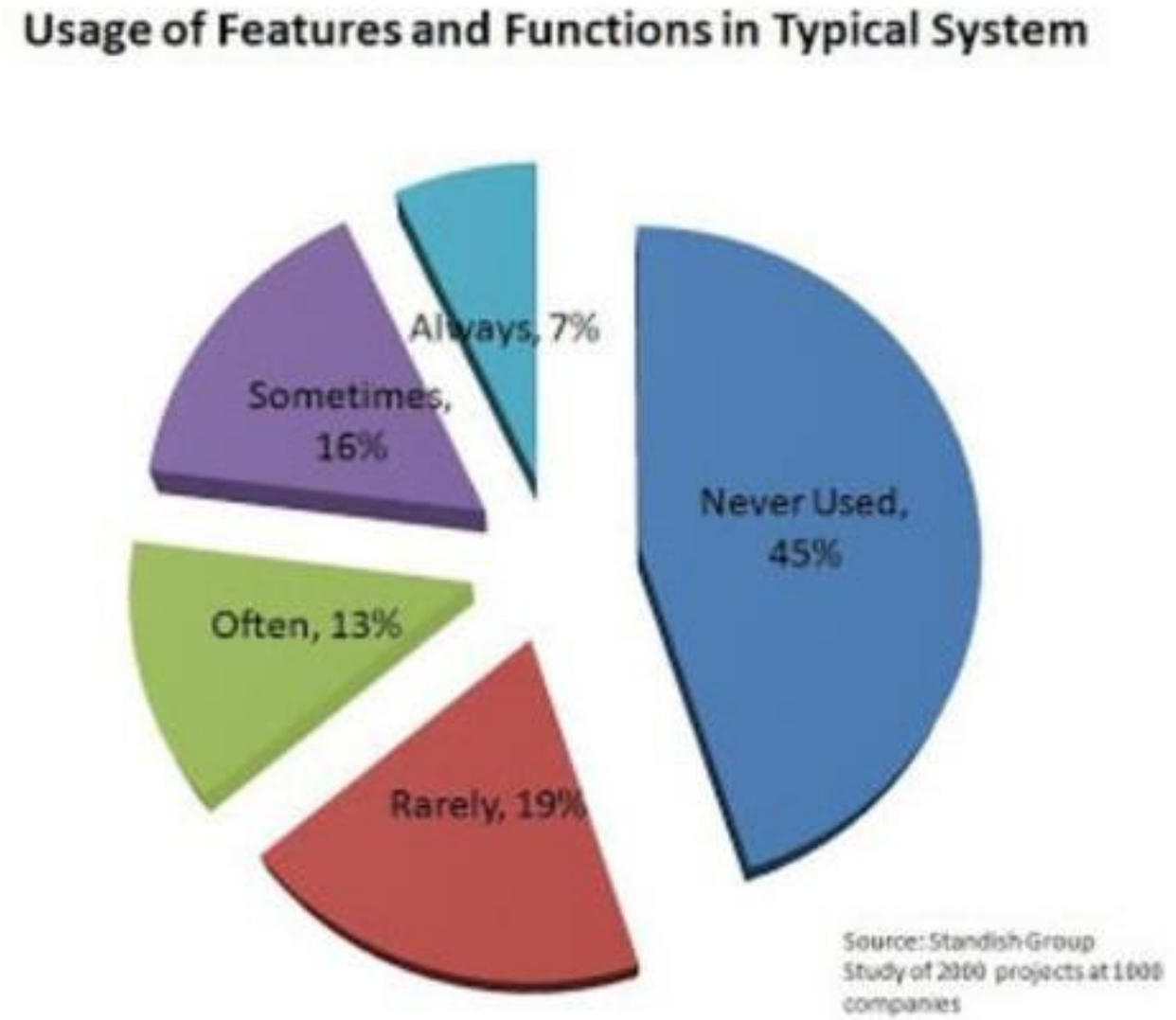
ビジネス側の人と開発者は、
プロジェクトを通して
日々一緒に働かなければなりません。

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。

Woodland Academy

ソフトウェアの無駄



- 常に使っている機能は 7%
- 使ったことがない機能は 45%

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 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。

前へ

- 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**

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。

次へ

前へ

- **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。

次へ

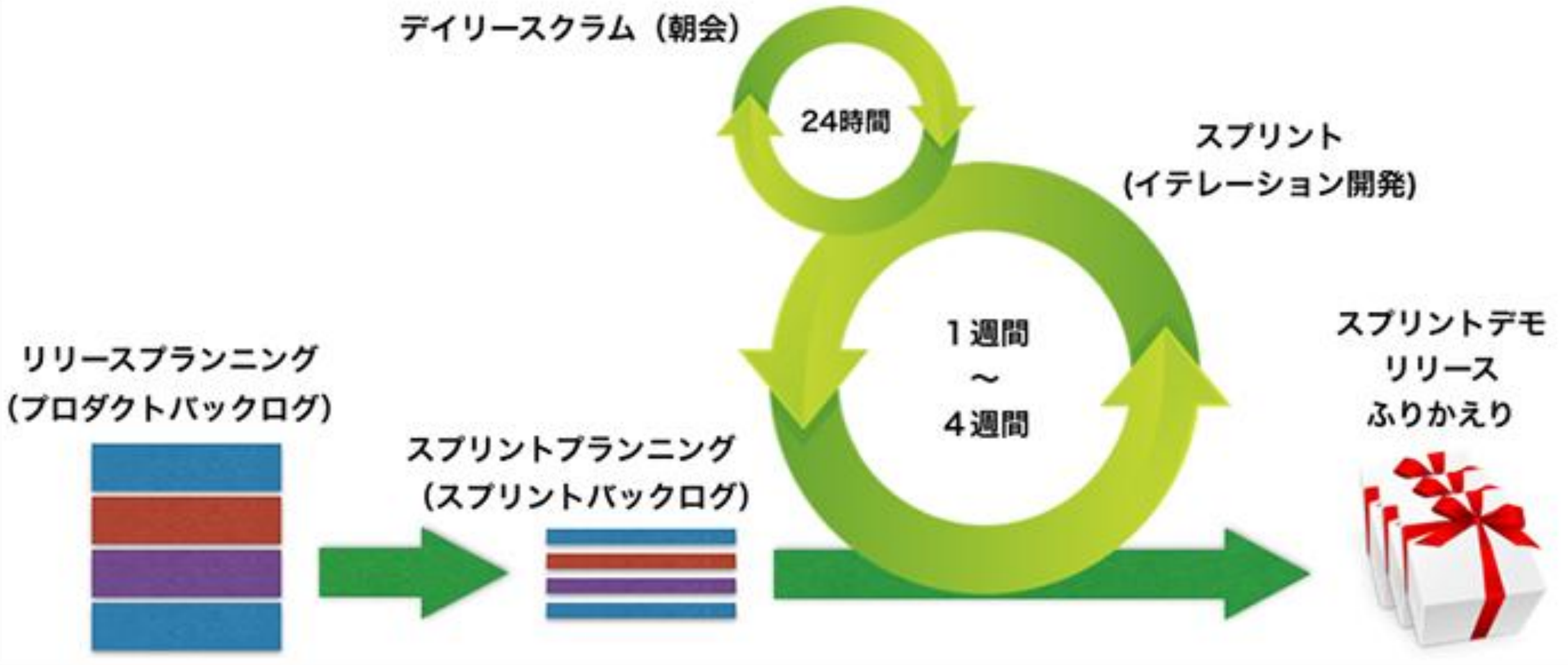
Woodland Academy

前へ

- **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。

スクラムの流れ



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。

次へ

- **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

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。

Woodland Academy

Q&A

目次

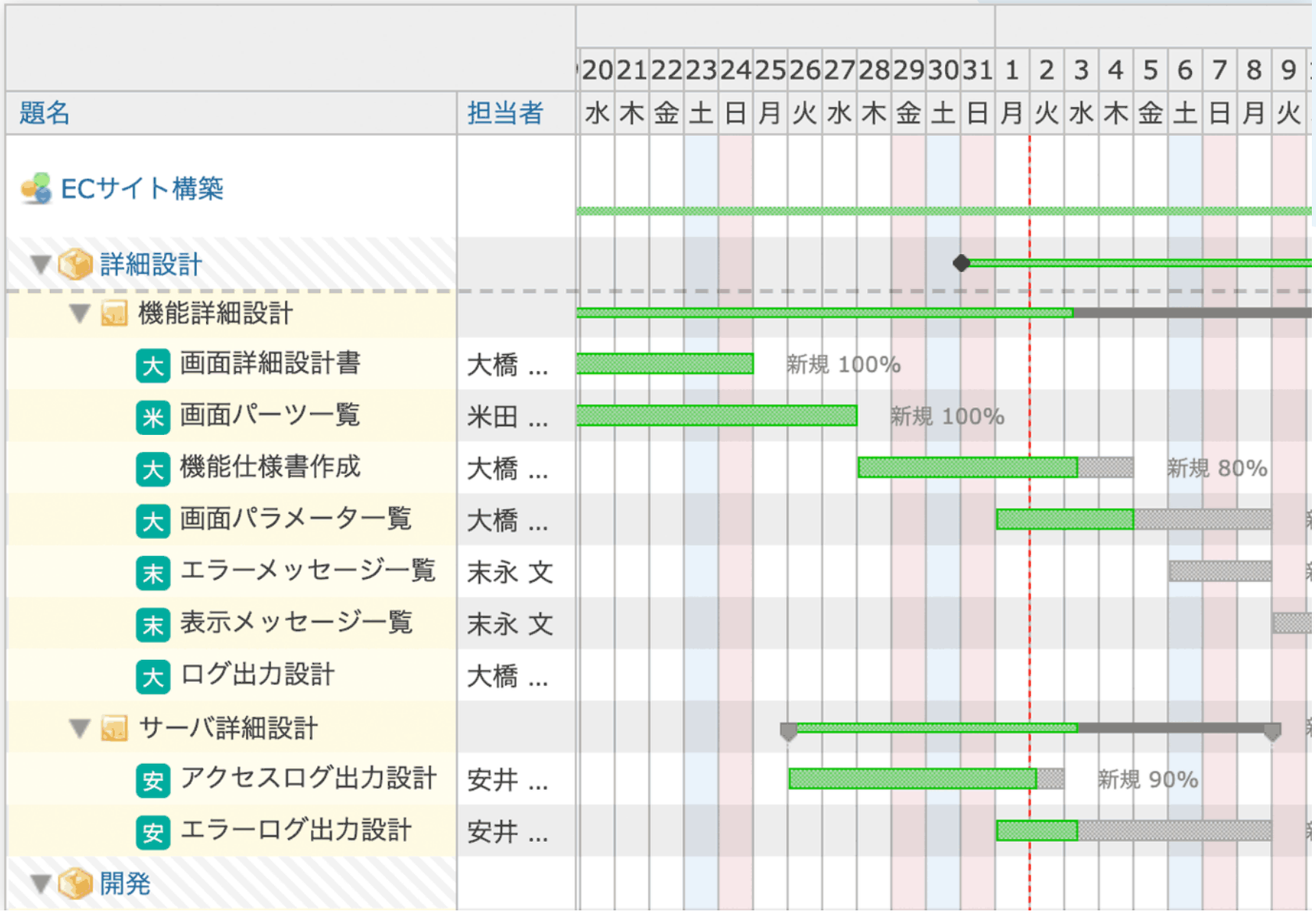
カンバンボード

- **カンバンボード**[Kanban]：課題が書かれている「カード」をドラッグ＆ドロップすることで直感的に課題の状態が変更できます。変更はリアルタイムに反映されます。

WEB-4	プロジェクトメンバーのアサイン		星島 賢人	完了	2018/03/08
WEB-5	キックオフミーティングの開催と議事録の作成		土屋 典子	処理中	2018/03/09
WEB-6	既存サイトのコンテンツの洗い出しと精査		星島 賢人	処理済み	2018/03/20
WEB-7	サイトマップの作成		峯山 マリナ	未対応	2018/03/23
WEB-8	ワイヤーフレーム作成		峯山 マリナ	未対応	2018/03/30
WEB-9	ステージングサーバーの構築		天木 亮	処理中	2018/03/26
WEB-10	CMSの検討と調査		神園 裕康	処理中	2018/03/30
WEB-11	トップページ	(0/2)	草光 ケン	処理中	2018/04/10
WEB-12	トップページデザイン		峯山 マリナ	処理中	2018/04/05
WEB-13	トップページコーディング		駒川 堅太	未対応	2018/04/10
WEB-14	導入事例	(0/5)	草光 ケン	未対応	2018/04/16

ガントチャート

- **ガントチャート**[Gantt Chart]：ガントチャートはプロジェクトの進捗を見える化します。担当者や期限を明確にして、計画通りに作業が進んでいるか確認します。



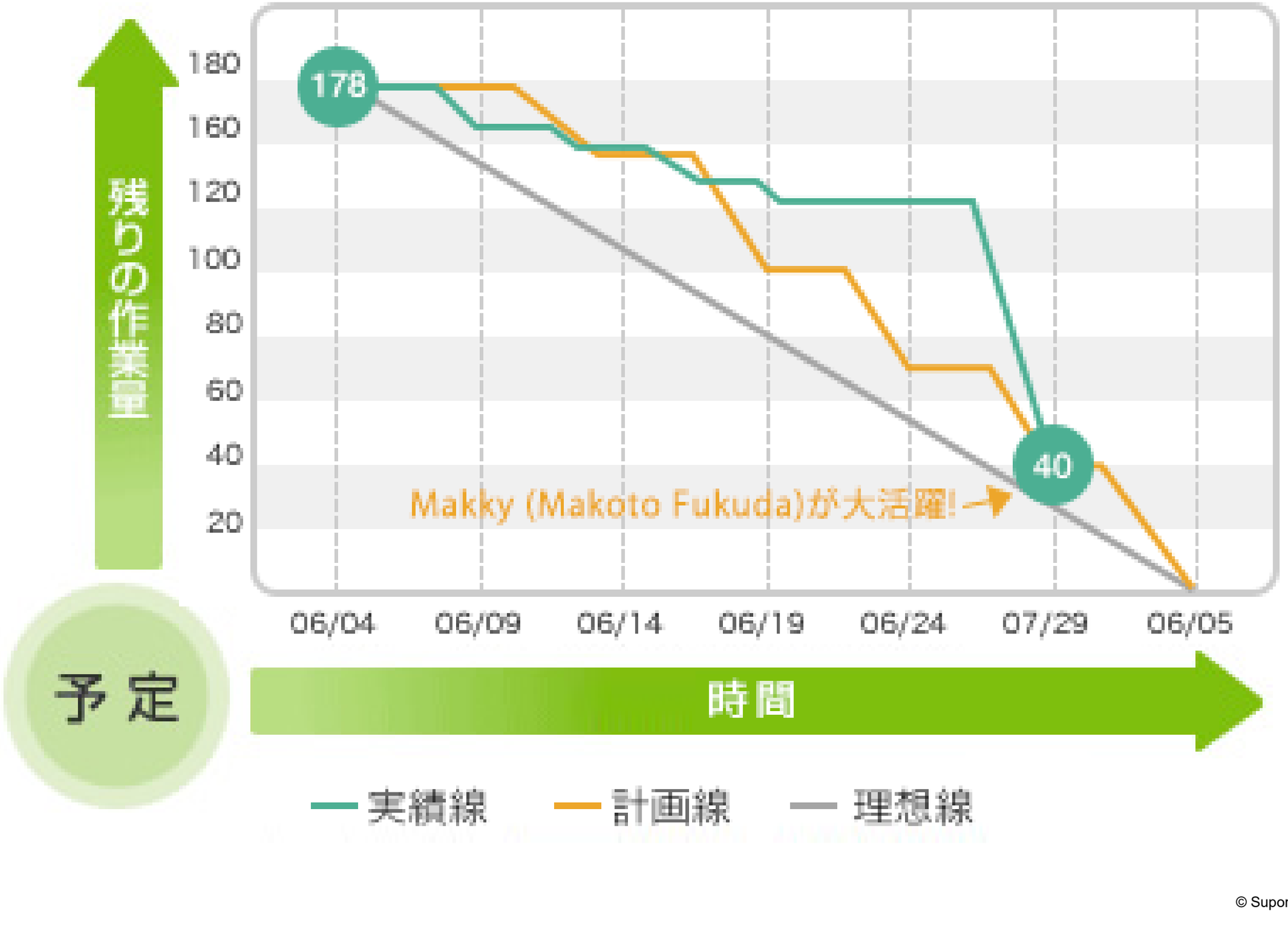
プロジェクト管理

バーンダウンチャート[Burndown Chart]：バーンダウンチャートは「期限までに全ての作業を消化できるのか？」ということが一目で分かるグラフです。バーンダウンチャートでは縦軸に「作業量」、横軸に「時間」を割り当てて残りの作業量がグラフ表示されます。時間が進むと残りの作業量が減っていくので、右肩下がりのグラフになります。バーンダウンチャートには指標となる 3 本の線が表示されます。

- **実績線**：実績を表す線で、その日時点での残りの作業量を表します。課題が完了したら設定された「予定時間」ぶん減少していきます。
- **計画線**：計画を表す線で、課題に設定された「期限日」と「予定時間」を元に表示されます。予定通りに計画が進めば実績線は計画線にほぼ重なります。
- **理想線**：全ての作業量合計を期間で平均した理想の線です。計画線と比較して、計画が妥当かどうかの目安となります。

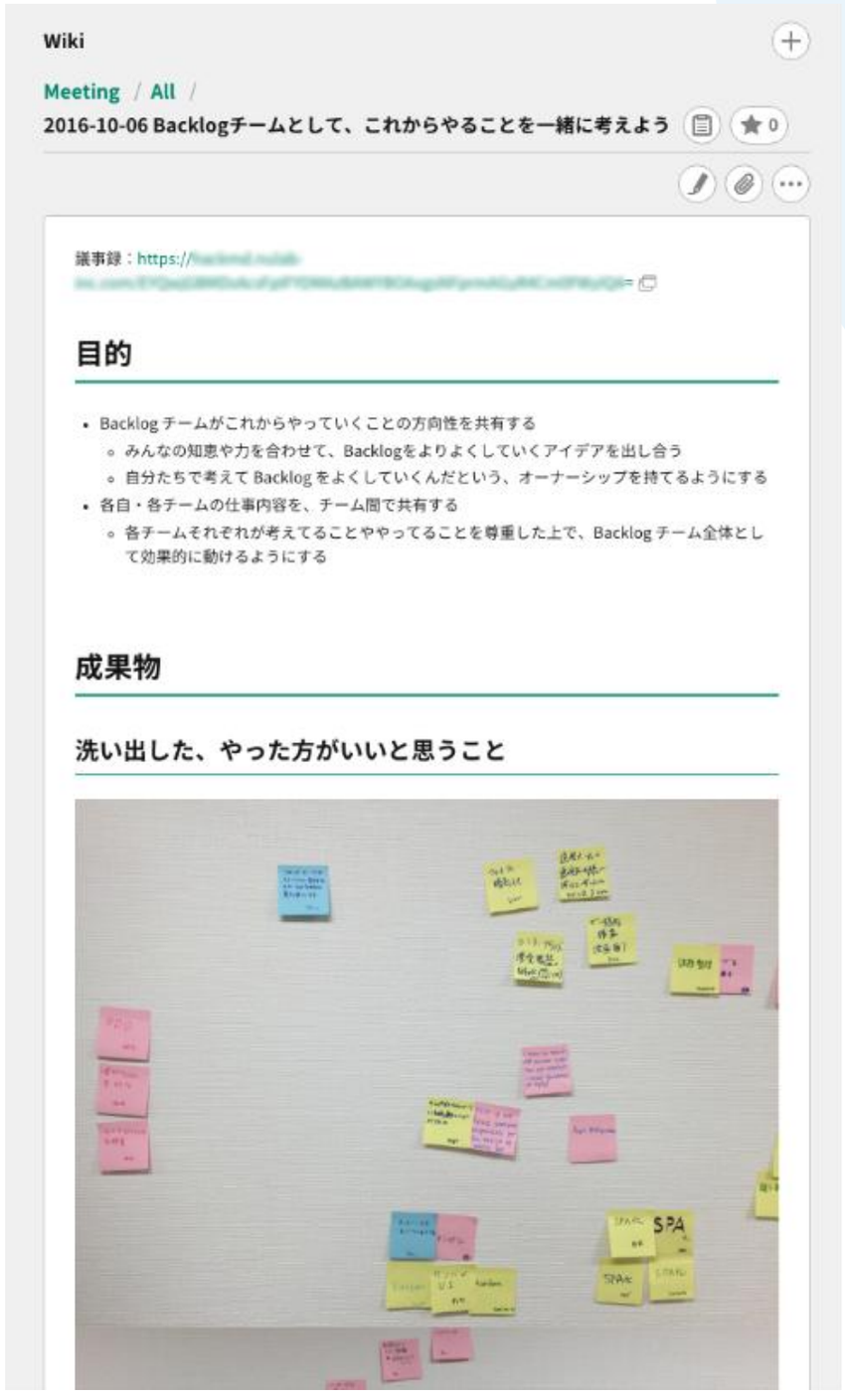
Woodland Academy

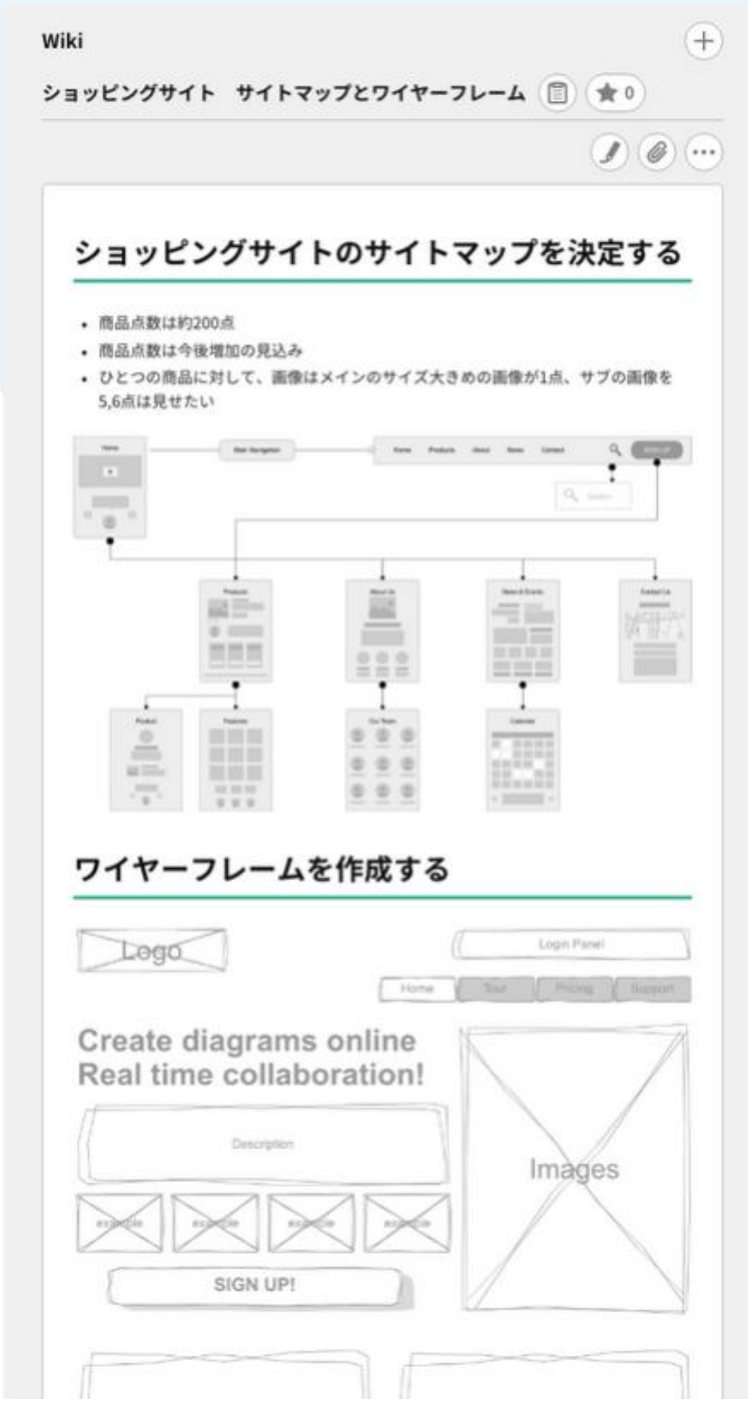
前へ



チームコラボレーションツール

- **Wiki**：プロジェクトや課題に関するメモ、会議の議事録、作業マニュアル、仕様書などチームメンバーに向けた情報を文書で管理します。





バージョン管理

- **バージョン管理システム**[Version Control System]とは、ファイルの変更を管理するものです。**いつ**ファイルが作られたのか、**どこ**が**誰**によって変更されたのか、その履歴を管理することはソフトウェア開発において、とても重要なことです。

- ソフトウェアに何か重要な問題が発生したときに、過去の状態に戻して確認が必要なこともある。そういったときに過去の状態に戻す作業はバージョン管理システムがなければ大変です。変更を行った本人でさえも、当時のソースコードを再現することは難しいかもしれません。

- また、同じソースコードに対して複数人が同時に作業を行ってしまうと、最新版がどれなのか分からなくなってしまう問題も発生します。

バージョン管理：Git

- **Git** はもともと Linux カーネルの開発者の間で作られた**分散バージョン管理システム**です。

- Git に限らずですが、分散バージョン管理システムで重要なのは「リポジトリ」という概念です。このリポジトリに全てのファイルの変更履歴が管理されています。

次へ

- ブランチとは、変更履歴の流れを枝のように分岐して記録していくためのものです。この仕組みによって、リリース版に対して行われた変更の影響を受けることなくバグ修正に取り組むことができます。逆もまた然りで、バグ修正の影響がリリース版に及ぶ前（マージ前）にテストを行うことができます。
- デメリットとしては、マージ作業の手間がかかる場合があるのと、マージさせたときにソースの競合が発生するとその解決に時間がかかるという点でしょう。

バージョン管理：SVN

- Apache Subversion（SVN）は**集中バージョン管理システム**に分類されます。リポジトリが 1 つだけあり、それに対して複数の開発者がアクセスします。

- コミットするたびに新しいリビジョン番号（バージョン）が振られるため、直感的にバージョン管理しやすい一方、複数で 1 つのリポジトリにアクセスするため、競合が発生しやすいというデメリットもあります。誰かがソースコードをチェックアウトしているときは他のメンバーは編集することができません。

次へ

- Subversion のメリットとしては導入コストの低さです。開発プロジェクトメンバーの中でもスキルの格差が大きいことはよくあります。分散バージョン管理システムの理解が難しいメンバーに教育するコストも必要ありません。

- デメリットとしては、1 人のミスが開発メンバー全員に影響します。

Woodland Academy

Q&A

まとめ

Sum Up

1.ウォーターフォール。

2.アジャイル開発：
① スクラム、
② XP、
③ FDD。

3.プロジェクト管理ツール：
① チャートやボード、Wiki。
② バージョン管理。

Woodland Academy

Thank you!

From Seeds to Woodland — Shape Your Future.

Shape Your Future