NTT DATA
Trusted Global Innovator

従来型開発に慣れたPM必見！
アジャイルプロジェクトの計画時に抑えるべきポイント

NTTデータ先端技術株式会社

アジャイルプロジェクトを始める流れ

	Step0	Step1	Step2	Step3
検討Step	価値探索フェーズ	アジャイル開発導入可否を判定	アジャイル開発に向けたプロジェクト計画の策定	見積もり実施	契約締結
作業内容	• 対象としているユーザーに対する価値提供機会の発見	• アジャイル開発とするべきかどうかを判定する。	• アジャイル開発を進めるうえでの考慮点を洗い出し、プロジェクト計画を策定する。 • 計画に基づき見積もりを実施する。	
インプット	• 価値探索フェーズに関わる情報すべて	システム化対象案件の下記情報 • 概要案件（全体像） • 概算見積もり • 概算スケジュール	システム化対象案件の下記情報 • 概要案件（全体像） • 概算見積もり • 概算スケジュール • 案件個別チェック結果	
アウトプット	• 価値提供のための仮説（概要案件）	• 案件個別チェック結果	• プロジェクト計画、見積書	

アジャイルプロジェクトを始める流れ

	Step0	Step1	Step2		Step3
検討Step	価値探索フェーズ	アジャイル開発導入可否を判定	アジャイル開発に向けたプロジェクト計画の策定	見積もり実施	契約締結
作業内容	• 対象としているユーザーに対する価値提供機会の発見	• アジャイル開発とするべきかどうかを判定する。	• アジャイル開発を進めるうえでの考慮点を洗い出し、プロジェクト計画を策定する。 • 計画に基づき見積もりを実施する。		
インプット	• 価値探索フェーズに関わる情報すべて	システム化対象案件の下記情報 • 概要案件（全体像） • 概算見積もり • 概算スケジュール	システム化対象案件の下記情報 • 概要案件（全体像） • 概算見積もり • 概算スケジュール • 案件個別チェック結果		
アウトプット	• 価値提供のための仮説（概要案件）	• 案件個別チェック結果	• プロジェクト計画、見積書		

今日のアジェンダ

1.要件定義とは違う？バックログができるまで

2.品質の良し悪しはこれで決まる！DONEの定義の作り方

3.スキルが高いだけではダメ？体制構築のポイント

4.場当たり的ではない！アジャイルなスケジュールの立て方

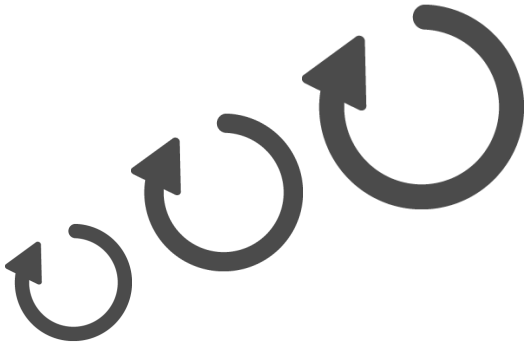
１.
要件定義とは違う？
バックログができるまで

価値探索とは

ユーザーを中心にプロダクトの価値や利用シーンについて企画するフェーズ

価値探索 → 高速開発 → 仮説検証

デザイン思考による技法を用いて、ユーザーを中心にプロダクトの価値や利用シーンを企画する。

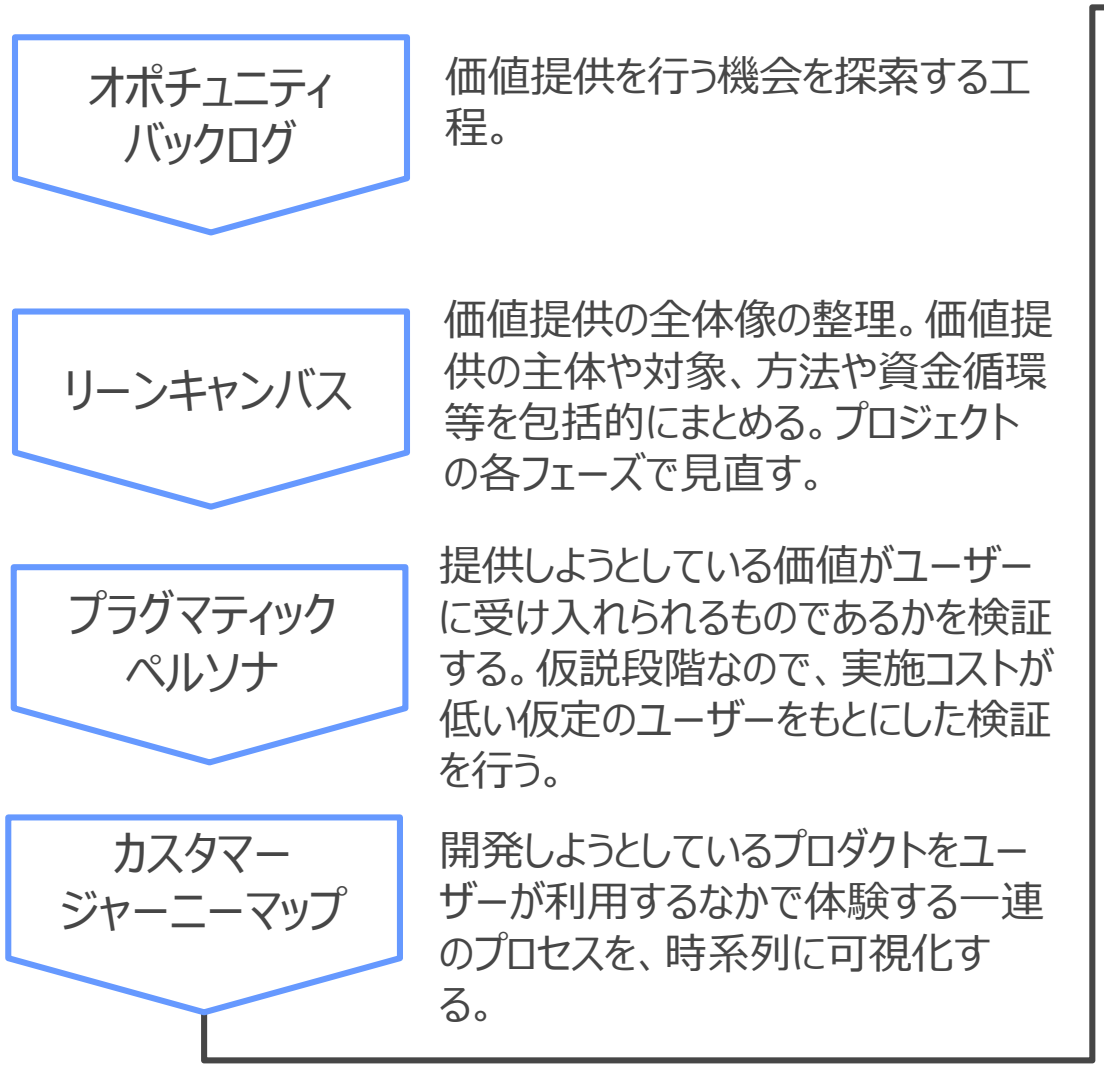


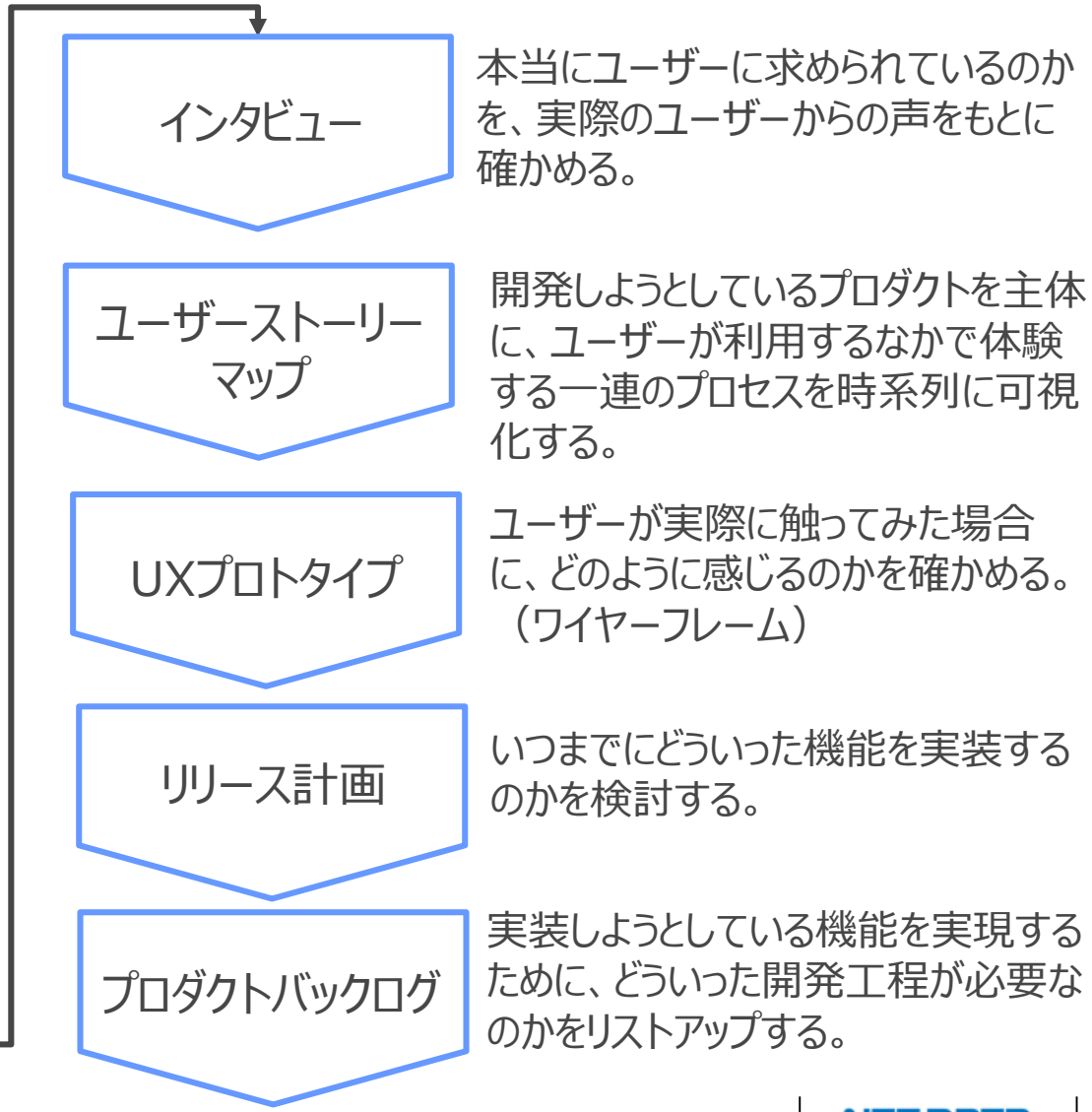
アジャイル開発を行い、デジタルプロダクトの試作品の開発と継続的なアップデートを行う。

企画や試作品を実際のユーザーに見せてのユーザーインタビューや定量テストの結果を見て、プロダクトのピボットやUX改善について検討する。

価値探索フェーズの流れ

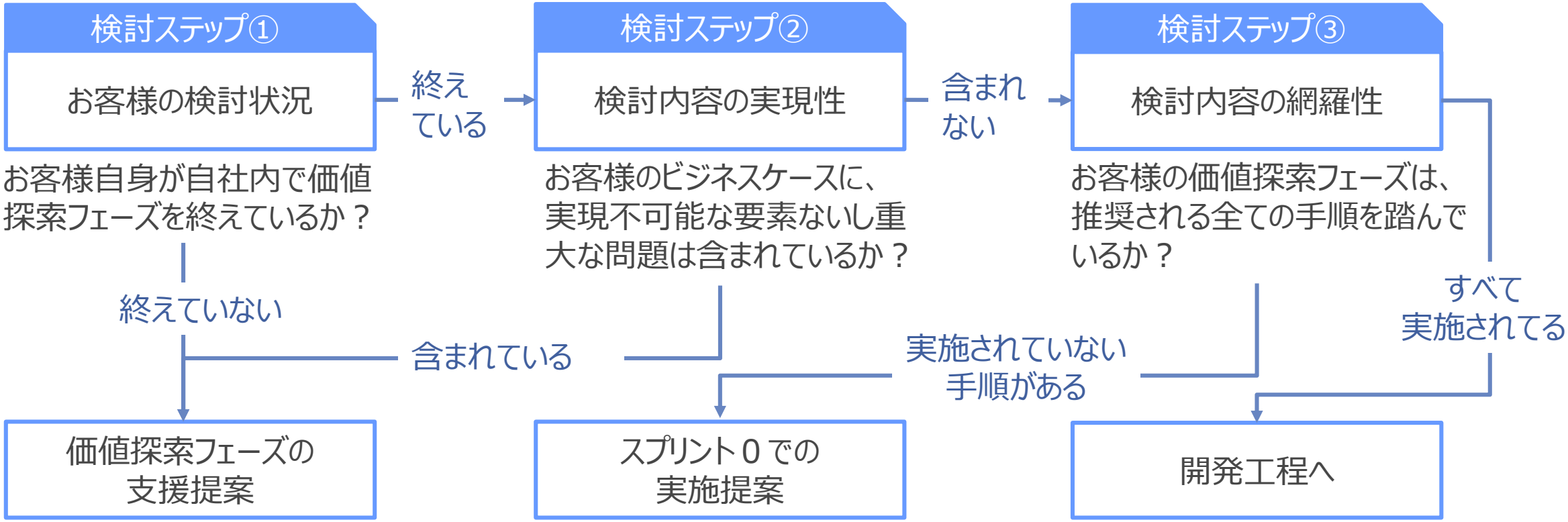
Altemista®における企画工程の流れ





価値探索フェーズの実施検討ステップ

お客様の検討状況にあわせて価値探索フェーズの実施の仕方を変える。



プロダクトオーナー支援が期待されている場合、価値探索フェーズを実施するための支援を提案。また、価値探索の内容に明らかな問題があれば価値探索フェーズをもう一度やり直すこともある。

価値探索フェーズを実施していても、開発フェーズに進むために必要な価値探索フェーズの工程に抜けている箇所があればスプリント０で実施する。
（例：バックログはあるが、開発者としてのバックログがない）

開発工程に進むための要素がすべて揃っていることを確認できたら、開発工程へ進む。

よくある疑問1-①

ウォーターフォールでも同じようなプロセスをやる場合もあるけど？
アジャイルだと何か違いはあるの？

アジャイルでは価値探索は一度やって終わりではありません。
価値探索で作成されたバックログはあくまで仮説のため、検証する（リリースしてユーザのフィードバックを受ける）ことが大切です。
検証して仮説が誤っていた場合は価値探索を繰り返します。

よくある疑問1-②

価値探索フェーズで実施するアクティビティが多いけど、いきなりバックログを作るのじゃダメなの？

これらのアクティビティは必ずしも全てやる必要はありません。
ただ先述した通り、アジャイルは仮説検証を繰り返すことを前提にしており、仮説が間違っていた場合（想定していたユーザ層が誤っていた、想定していた利用シーンが誤っていた等）にどのアクティビティから再検討すべきかが分かりやすくなります。
いきなりバックログから作成してしまうと、こういったピボットが難しくなってしまうことが考えられます。

2.
品質の良し悪しはこれで決まる！
DONEの定義の作り方

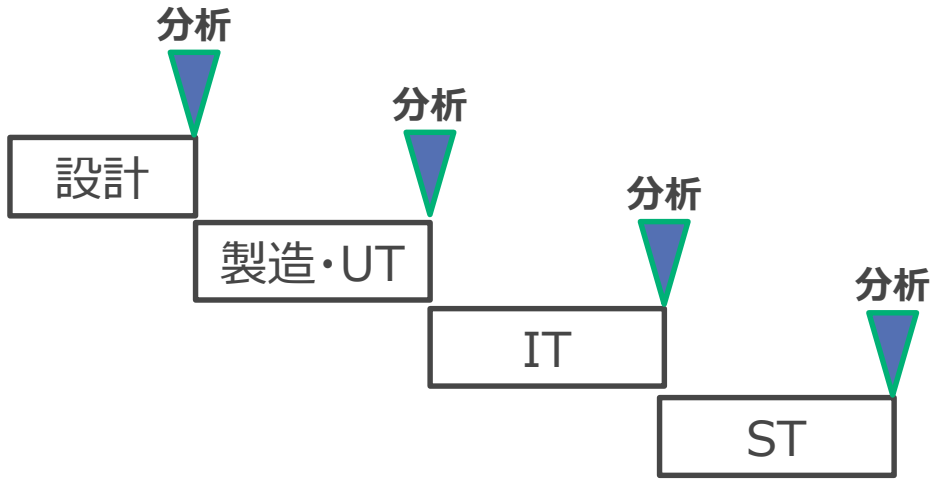
WF型の品質管理はアジャイル開発で適用できるか？

①プロセスの違い

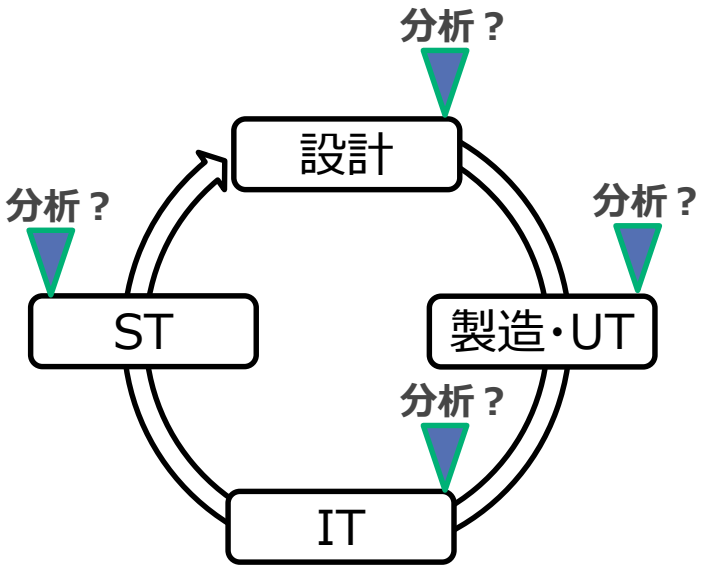
ウォーターフォールでは工程ごとに品質分析するが、アジャイルには工程という概念がない。（曖昧）

設計・実装・テストという段階的な開発を前提とした品質保証は難しい。

ウォーターフォール



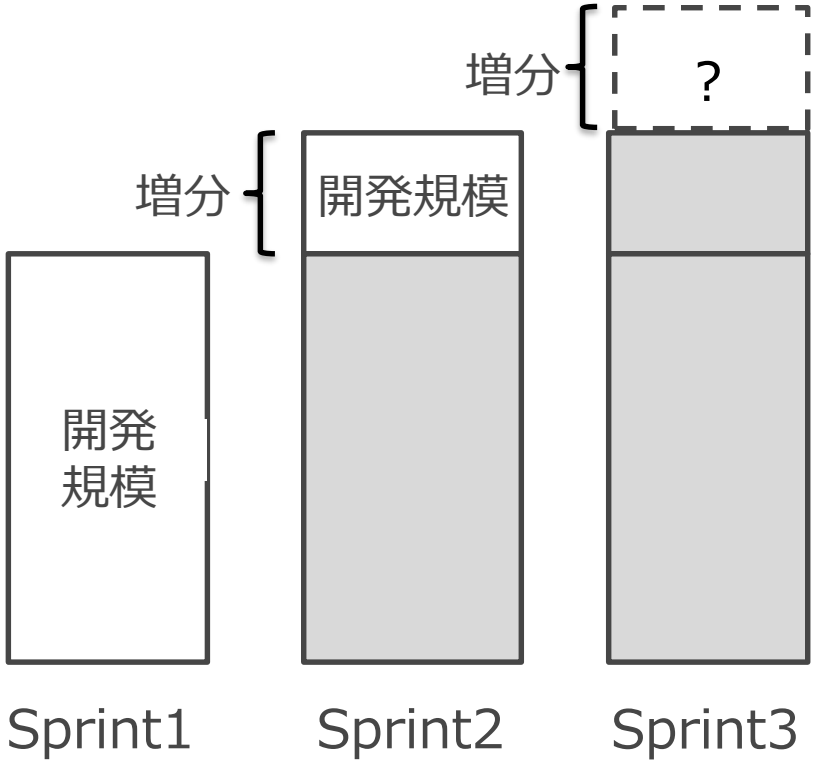
アジャイル



WF型の品質管理はアジャイル開発で適用できるか？

②規模の追跡

アジャイルではひとつの機能を一回作り上げたら終わりではなく、スプリントレビューでのフィードバックや新たなバックログにより常に開発を続けていくことになるため開発規模の追跡が困難。仮に取得できたとしても１スプリント当たりの開発規模は僅かなため、バグ摘出密度を根拠とした品質の判断が困難。



機能Aにおける開発規模

取得できても…
例えば1スプリントのあるモジュールの開発規模が10stepの場合

バグが1件→バグ摘出密度：100[件/ks]（上限越え）
バグが0件→バグ摘出密度：　0[件/ks]（下限越え）

アジャイルでの品質の捉え方

スクラムガイドにある品質に関する記述

確約（コミットメント）：完成の定義

完成の定義とは、プロダクトの品質基準を満たすインクリメントの状態を示した正式な記述である。

プロダクトバックログアイテムが完成の定義を満たしたときにインクリメントが誕生する。

完成の定義により、作業が完了してインクリメントの一部となったことが全員の共通認識となり、透明性が生み出される。プロダクトバックログアイテムが完成の定義を満たしていない場合、リリースすることはできない。ましてやスプリントレビューで提示することもできない。そうした場合、あとで検討できるようにプロダクトバックログに戻しておく。

インクリメントの完成の定義が組織の標準の一部となっている場合、すべてのスクラムチームは最低限それに従う必要がある。組織の標準になっていない場合、そのスクラムチームはプロダクトに適した完成の定義を作成する必要がある。

開発者は完成の定義に準拠する必要がある。プロダクトに関わるスクラムチームが複数ある場合、共通の完成の定義を作成して、それに準拠する必要がある。

完成(DONE)の定義の例

プロダクトバックログアイテムの完成の定義

- テストコード含め全てのコードがコミットされている
- プルリクエストによるコードレビューが実施されている
- 全てのユニットテストをパスしている
- 静的解析で必須修正事項がない
- …

スプリントの完成の定義

- 全てのストーリーの完成の定義が満たされている
- 開発ブランチにマージされ開発環境にリリースされている
- アーキテクチャ図が更新されている
- 全てのバグが解決しているか、方針を決めている
- ユニットテストによるコードカバレッジが80%以上である
- …

リリースの完成の定義

- 全てのスプリントの完成の定義が満たされている
- パフォーマンステストが実施されている
- 障害復旧テストが実施されている
- 第三者検証に合格している
- システム構成図、アーキテクチャ図が更新されている
- …

よくある疑問2-①

とは言っても規模に応じたバグ摘出密度を求められることも多いんだけど？

「アジャイル宣言の背後にある原則」では「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。」とあります。
顧客にとっての本当の価値はなんでしょうか？顧客の価値に必要かどうかを改めて考えてみてはどうでしょうか。
その結果、本当に必要との判断でしたらコストをかけてでも（開発量を減らしてでも）やるべきな場合もあります。

よくある疑問2-②

開発するからにはもちろんバグが出ると思うんだけど、いっぱいバグが出ちゃった場合はどう対処するの？

想定以上のバグが出た場合に残業して対処するといったことも考えられますが、アジャイル開発は持続可能であるべきで、一時的なムリが果たして良いのかはチームで話し合うことが必要かと思います。
バグの量を事前に予測する一つの考え方として「不確実性のコーン」があります。開発初期には開発コストのブレ幅が大きく、徐々に収束していくという考えです。これを参考にバッファを設けることは対処の一つでしょう。

3.
スキルが高いだけではダメ？
体制構築のポイント

アジャイルのチーム構成

–スクラムチームは3つのロールで構成される。

プロダクトオーナー

プロダクトの価値最大化に責任を負う。ユーザーのニーズを満たすように、プロダクトバックログの管理を行う。

スクラムマスター

開発者がアジャイルに専念するための環境づくりに責任を負う。プロセスの提示、障害の除去、改善フローの提示等を行う。

開発者

実際のプロダクト開発を行う。プロダクトの価値が高まるように、各開発者は自律的に考えながらプロダクト開発に貢献する。

–スクラムチームは水平・対等

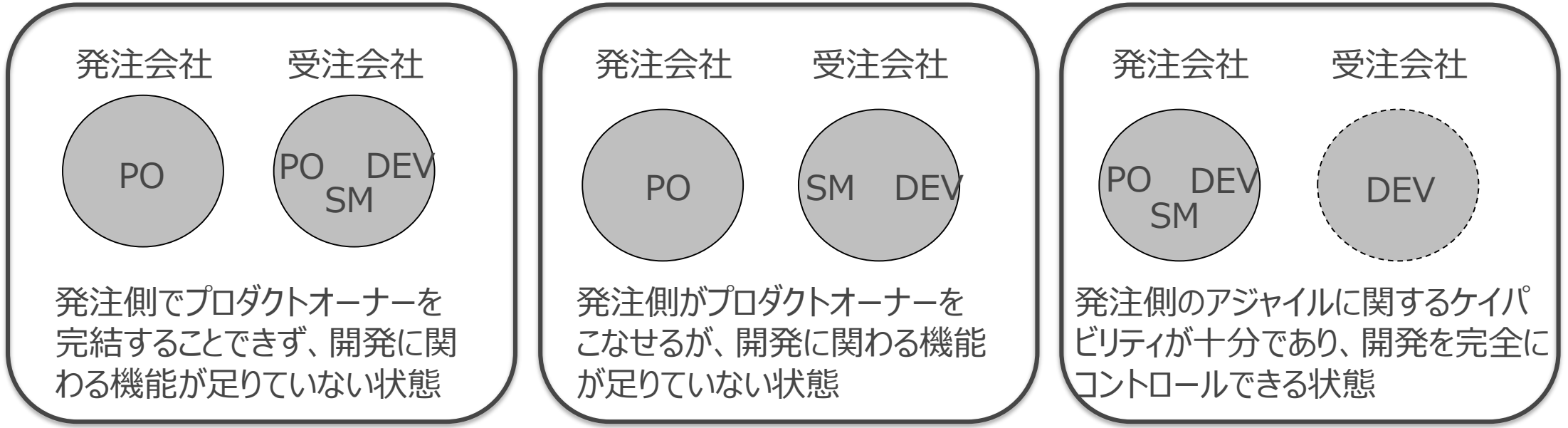
- ✓ **良いソフトウェアを創るために、役割が違う**
- ✓ **従来型のピラミッド構造と大きく異なる**

体制の留意点

発注側のスクラムに対するケイパビリティによって体制や関わり方が変わる。

※PO・・・プロダクトオーナー　SM・・・スクラムマスター　DEV・・・開発者

低い　発注側のケイパビリティ　高い



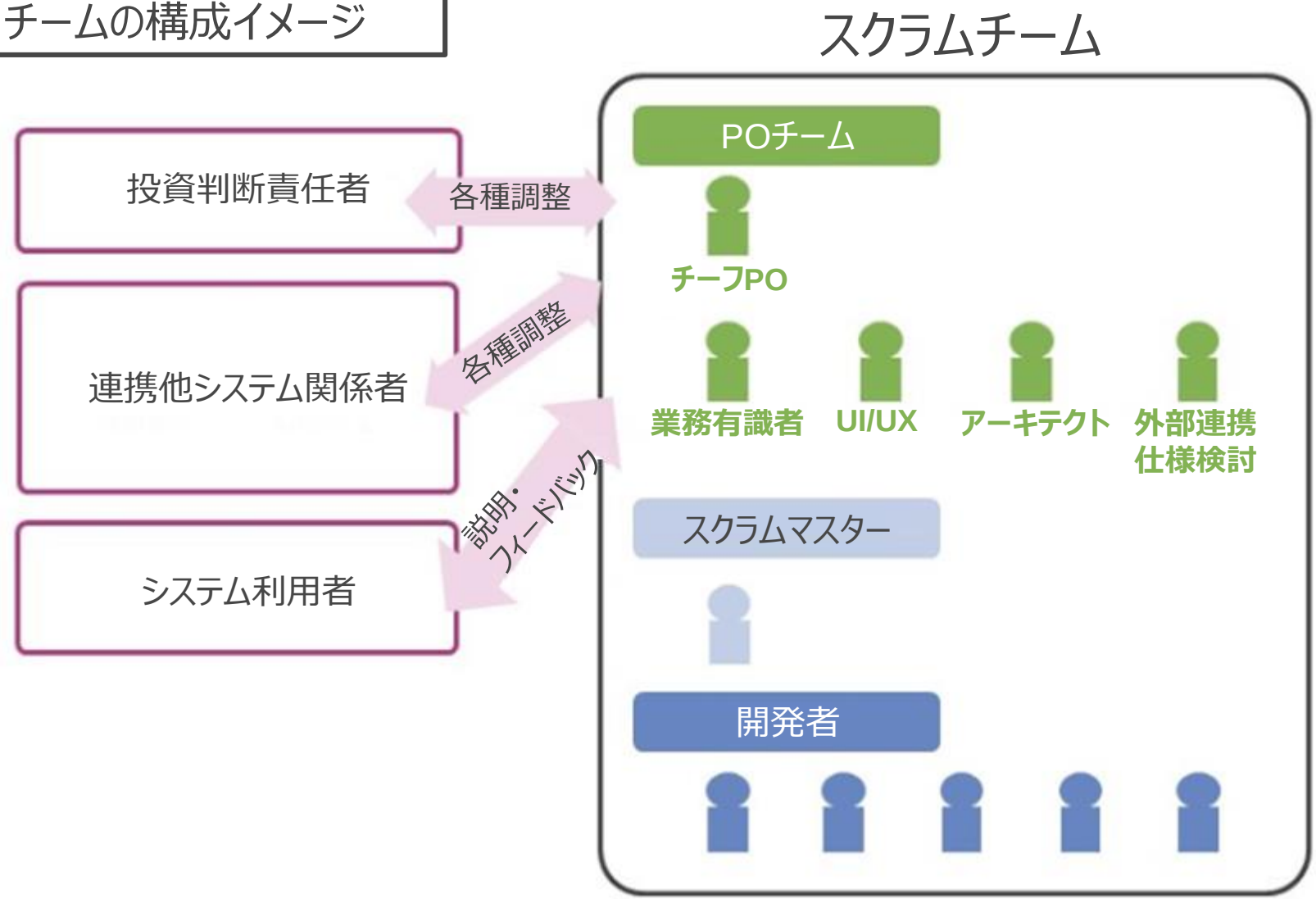
アジャイル開発を行うためのコーチング等の支援をしながら、受注側のスクラムマスター・開発者が開発を進める。（最終的には、発注側がプロダクトオーナーとして自律自走できる状態を目指す。）

発注側のプロダクトオーナーと連携して受注側のスクラムマスター・開発者が開発を進める。（スクラムマスターは、開発要件が現実的な水準になるようプロダクトオーナーと調整する。）

発注側にプロダクト開発のケイパリビリティが十分にある場合、受注側は開発者の派遣のみを行う。受注側チームは発注側チームに入る前提のため、発注側の開発ルールに従う必要がある。

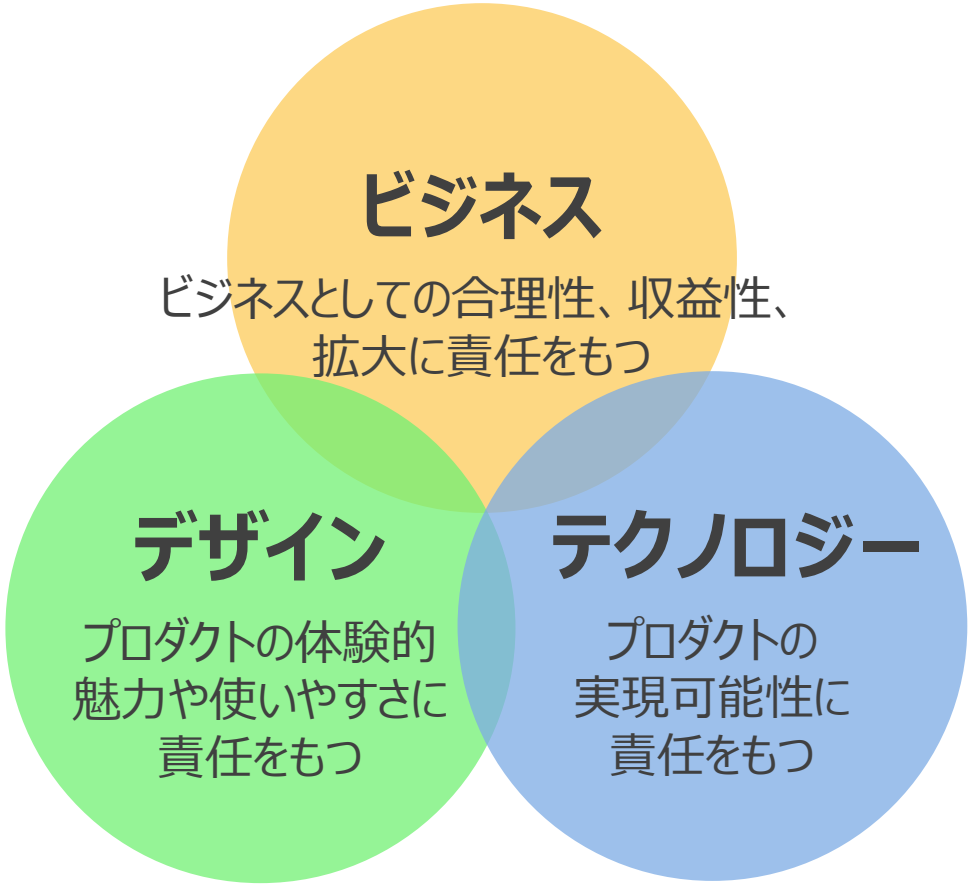
プロダクトオーナーチームの構成

チームの構成イメージ



プロジェクトに必要となる人財確保

プロジェクト遂行に必要なスキルセットを満たすよう要員を構成し、社内で必要な人財が揃わない場合には外部ベンダーの協力も視野に入れる。



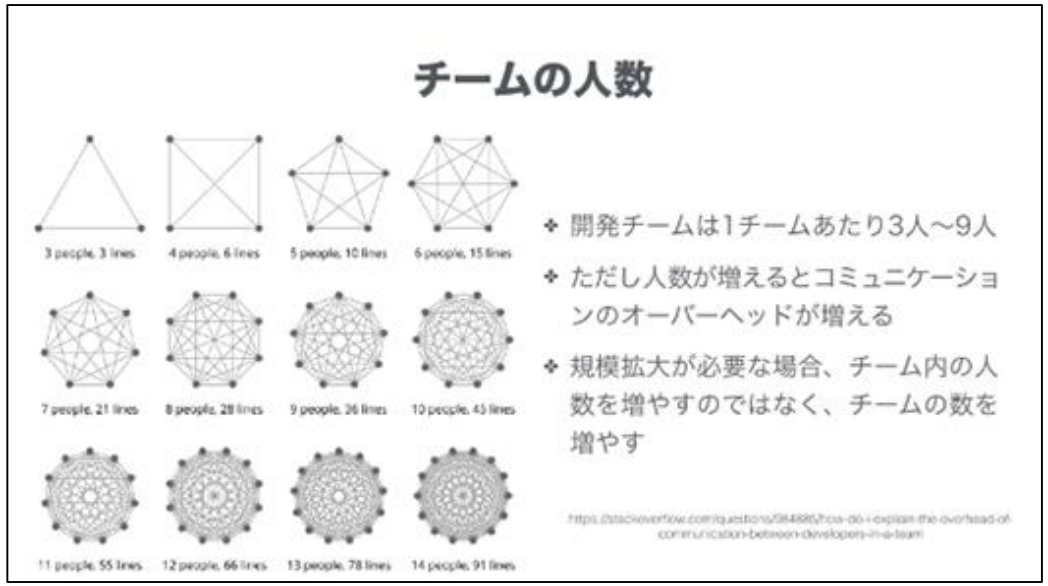
- ✓ **多様な参加者：**
  主要な関係者を集めたプロダクトオーナーチーム
  が全体をリード
- ✓ **ビジネス、テクノロジー、デザインの3点を押さえることが重要**
- ✓ **１人が兼任することもあるが、この役割が全部揃うことが大事**

適切な開発チームの人数は？

アジャイルでの開発チームの最大は9人
スクラムでやる場合は5人くらいがちょうど良い。

コミュニケーションパスの問題

5人の時のパスの数は10
10のパスを満たせば、全員の共通理解を得られる。
同じ情報を持っていれば同じ判断を下せるので、意思決定の速さに繋がる。



出所「スクラムを組織全体へスケールさせていくフレームワーク「Scrum@Scale」入門（前編）」
https://www.publickey1.jp/blog/19/scrumscaledevelopers_summit_2019_1.html

強いチームを形成するまでの5段階

「タックマンモデル」を理解することで、理想的な立ち回りとは何かを把握しチームを導くために必要な要素を知る。

①形成期(Forming)
招集されたばかりのチームで、特にお互いを知らないメンバーが多い時期

②混乱期(Storming)
チームとして目的や責任、役割が定着していない、理解しきれていない時期

同じ目的、ゴールに向かって進んでいくために、ゴールはどこかを早い段階で明確に示す。

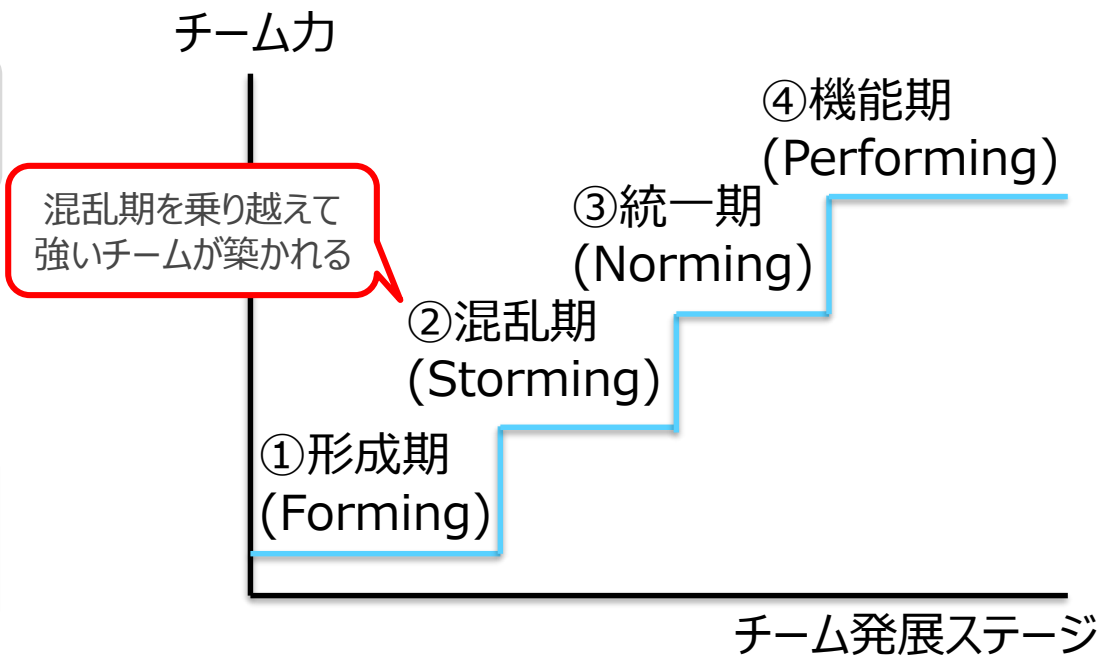
③統一期(Norming)
チームとして目指す方向、意見統一。お互いに存在価値を見出し、リスペクトできている時期

メンバーたちが混乱期に逆戻りしてしまっているようなことがないか注意すること。「チームとしてうまく業務が遂行できているか」細かいところをチェックする。

④機能期(Performing)
チームとして、成熟して理想的な状態

メンバーの自発性を促す方向にシフト

⑤散会期(Adjourning)



よくある疑問3-①

水平・対等とあるが、上司/部下、発注元/発注先、の関係があるとそうはいかないのでは？どういう工夫をするの？

水平・対等になるためには、心理的安全性を高めることが重要です。心理的安全性とは自分が発言することを恥じたり、拒絶したり、罰をあたえるようなことをしない、という確信を持っている状態を指します。例えば相手を尊重する、ミスをしても責めず人ではなく問題にフォーカスする、といったことを続けることで心理的安全性は高まっていきます。
また、プロジェクト開始前にアジャイル研修を活用して、全員のマインドをそろえることは良いことでしょう。

よくある疑問3-②

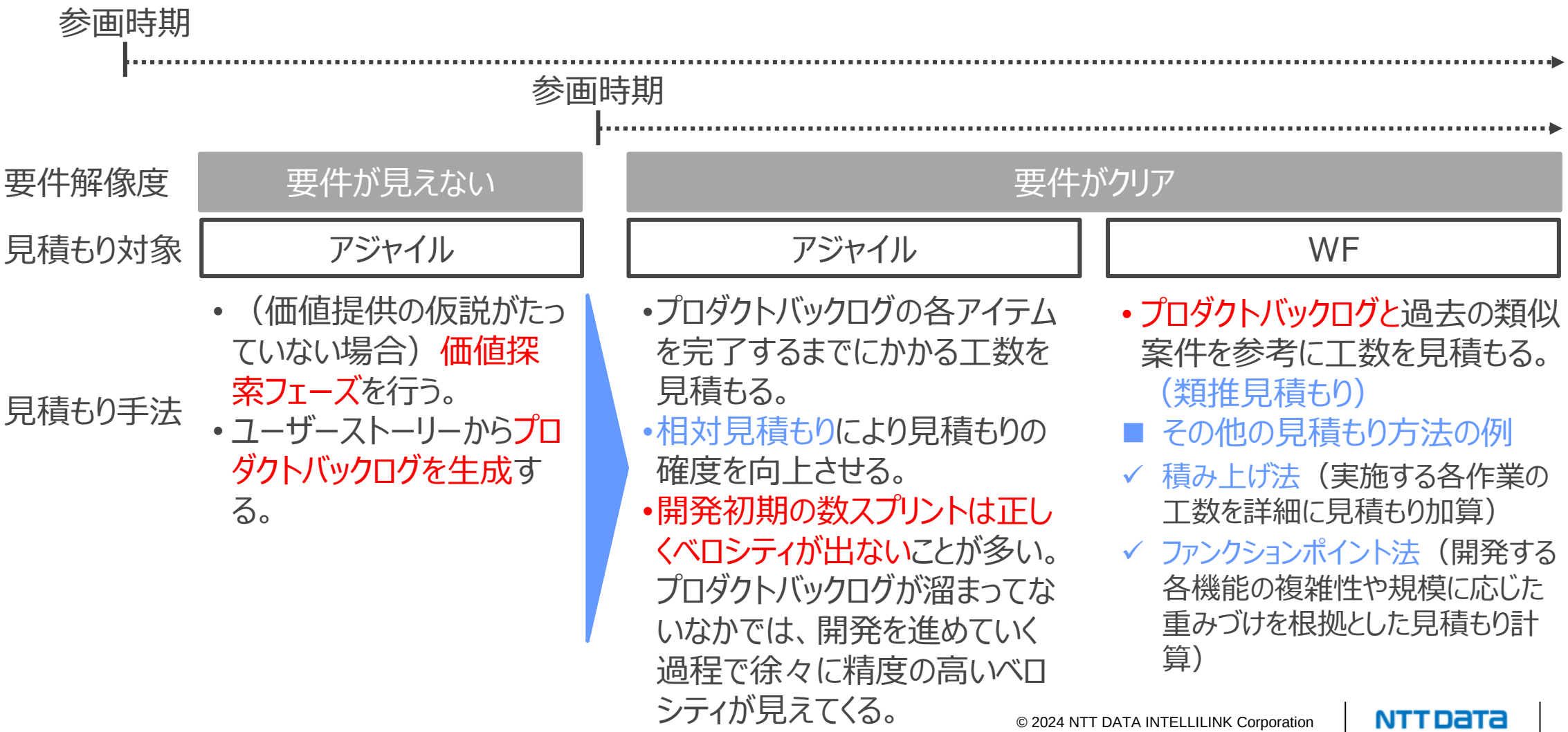
途中で人を増やしたい/減らしたい/変えたい場合はどうしたらいい？

プロジェクトによっては様々な状況によって人員の入れ替えが必要になるケースはあるでしょうが、アジャイルでは一般的にリスクが高いと言われています。
人を減らす場合は機能横断が崩れる可能性があります。人を増やす場合や入れ替える場合はそれまで培ってきたチームの文化（経験主義）が崩れ、イチからチームの組成が必要になると言われています。そのためこういったリスクを鑑みながら、それでも本当に必要かを判断するべきと言えます。

4.
場当たり的ではない！
アジャイルなスケジュールの立て方

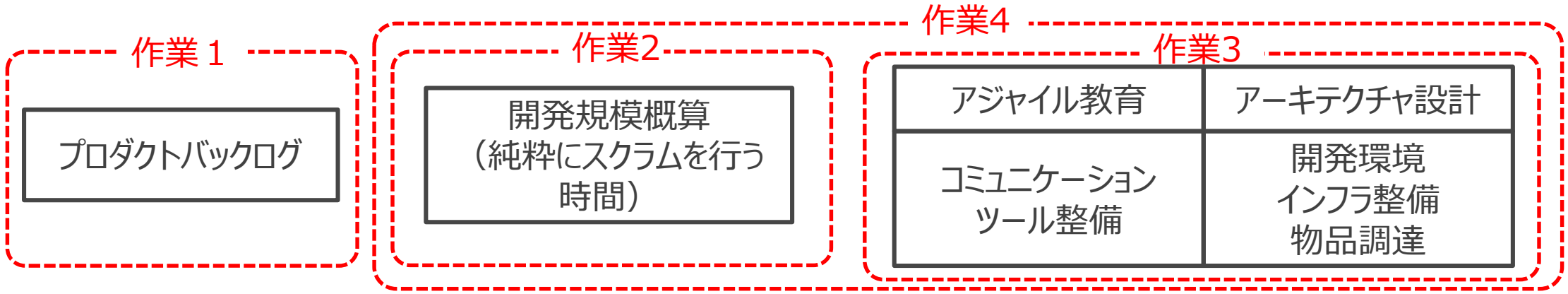
見積もり

アジャイル開発の作業見積もりはプロダクトバックログをベースに算出する。
WF・アジャイル両方の見積もりを行なって乖離がないか確認する。



何を見積もるのか

開発工数だけではなく周辺の工数も見積もる。



	作業１ プロダクトバックログの詳細化	作業２ 開発規模概算見積もり	作業３ 関連作業見積もり	作業４ プロジェクト全体の見積もり
作業全体	✓ 全社	✓ IT部門	✓ IT部門	✓ IT部門
インプット	✓ ユーザーストーリー（またはバックログ）	✓ プロダクトバックログ ✓ 要員計画案、調達計画案	✓ プロダクトバックログ ✓ 要員計画案、調達計画案	✓ 作業１～３のアウトプット
アウトプット	✓ プロダクトバックログ	✓ 開発規模概算見積もり ✓ 開発スケジュール	✓ 関連作業見積もり ✓ 関連作業対応スケジュール	✓ プロジェクト全体の見積もり
作業内容	✓ 作成されたユーザーストーリーないし粒度の粗いバックログを見積もり可能な粒度まで分割する。（要件の大きさの偏りを分割して整える。）	✓ 相対見積もりで概算規模を見積もる。 ✓ 見積もり工数とアジャイル開発者の規模から完成可能な開発スケジュールを試算する。	✓ 純粋な開発作業以外に必要な関連作業の工数と期間・費用を見積もる。	✓ 作業２・３の見積もりを合算し、全体の工数と期間を見積もる。 ✓ 従来型見積もりや過去実績を比較して正確さを担保する。

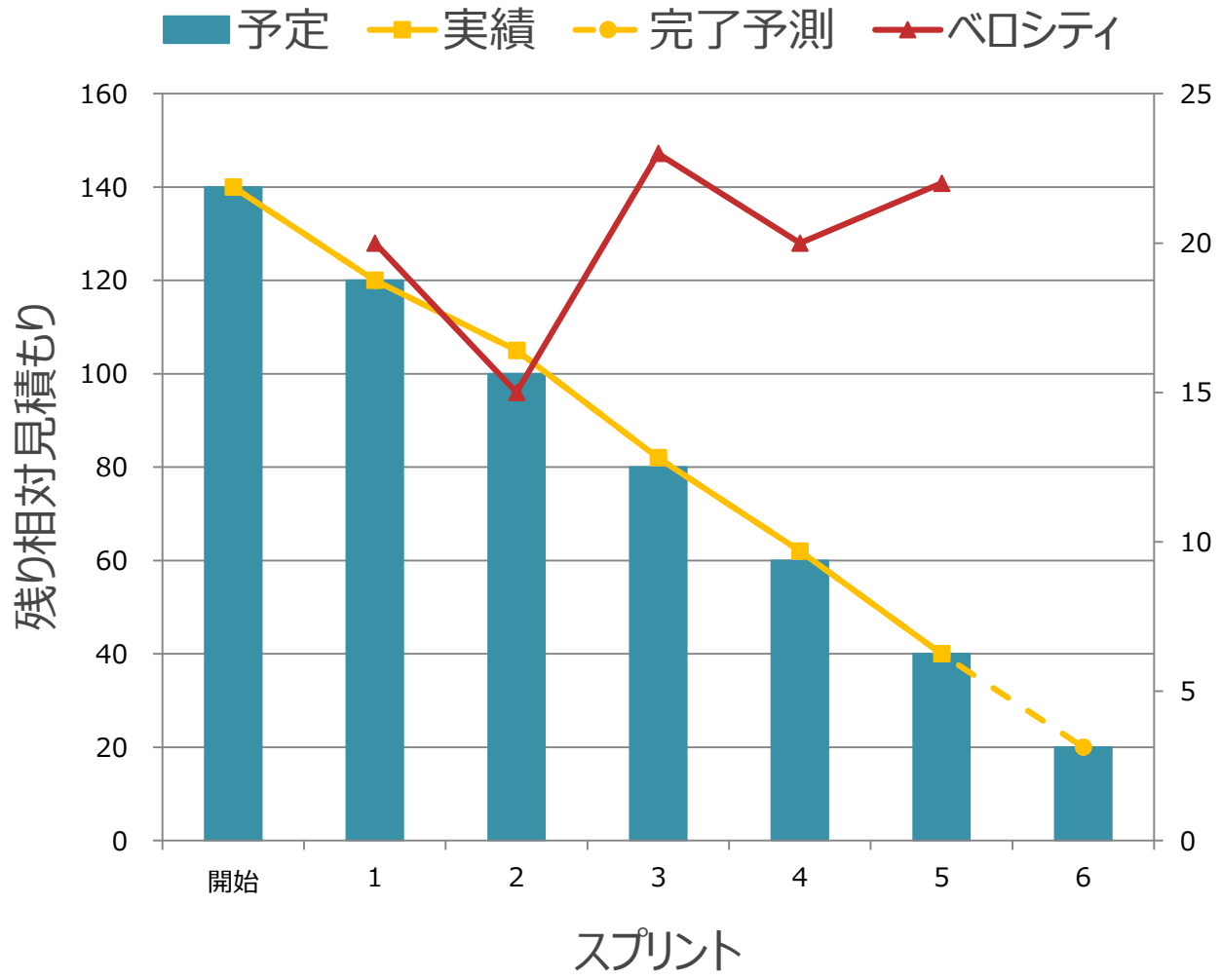
Sprint0

開発者のスムーズな立ち上げのための地ならし

	タスク	詳細
スプリント0	キックオフ	
	インセプションデッキ作成	何故このプロジェクトに取り組むのか、プロダクトのコアとなる価値、ユーザーに届けたい価値は何か等について、10個のトピックを通してチームの共通認識を得る。自己組織化されたチームを目指す。
	（初回の）プロダクトバックログ（PBL）の作成	直近で開発するものの詳細化。実現可能性の検証も兼ねて実施する。（初回は普段の運用と少し異なる）言語仕様の限界を見切る。
	（初回の） 開発アウトプット決定	開発において何をどの程度までアウトプットするのかを決定する。可能であればドキュメント等にまとめる。（完成の定義ならびに初回スプリントにおけるインクリメントの定義）
	開発者ルール決め	開発に関わる全ステークホルダーの役割の確認。レビューはレビュワー〇人に見てもらう、朝会は〇〇~〇〇で実施する等のルールを決める。
	コーディング規約作成	コーディングのスタイルを定義する。支援先と決定する。
	アーキ・フォルダ構造の検討	アーキテクトとフォルダ構造について定義する。支援先と決定する。
	開発環境の構築	Gitlab等のライセンスの取得、プロジェクトの作成を行う。その他、開発にあたり必要となるハードウェアやソフトウェアの設定を行う。
	テスト自動化検討	単体、結合、総合テストをどこまで自動化するかを検討する。（PMは、CI/CDパイプラインの有識者として対応する。）

スケジュール管理

リリースバーンダウンチャート



左図では、見積もられていたプロダクトバックログアイテムがどれぐらい残っているのか、予測とどれぐらい乖離しているのか、ベロシティの増減はどのようになっているのかを把握する。

ベロシティは時間の経過とともに上昇するのが通常なので、ベロシティが下がるということがあれば、その理由について調査を行って対応する。

よくある疑問4-①

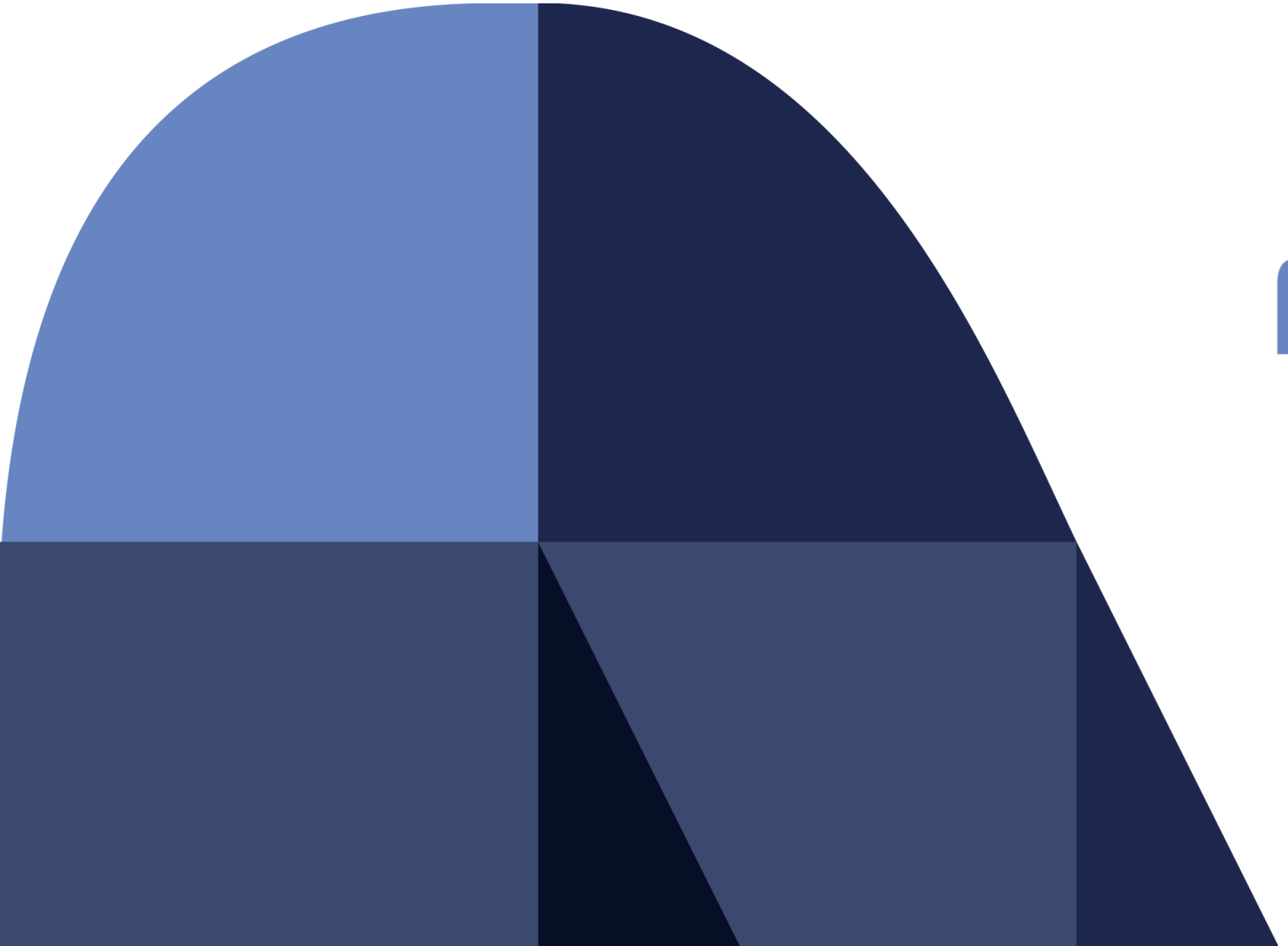
アジャイルだと途中で追加の要件（バックログ）が入ってくると思うのだが、
それをどうスケジュールに反映していくの？

アジャイルの計画は一度やって終わりではありません。
追加されたバックログは見積もられてリリースバーンダウンチャートが更新されます。
スプリントレビュー等でリリースバーンダウンチャートを元に今後のスケジュールについて議論し、必要ならばリリース日を変更することを検討します。

よくある疑問4-②

アジャイルはスコープ変動を前提にしているけど、納期が決まっている場合はどうやるの？

リリースに必要なMVP（価値を提供できる最低限のプロダクト）は何かを検討し、スコープ調整をします。
スプリントレビューで実際に動くプロダクトを見て、新しいアイデアをプロダクトバックログとして追加することは良いことです。ただ納期が決まっている場合は新しいアイデアを採用する代わりに他のバックログの優先順位を下げてスコープアウトすることを検討することが必要です。



NTT DATA

Trusted Global Innovator