Agile Japan 2023

形だけのアジャイルから
中身のあるアジャイルへ変わるために
～アジャイルの原点に立ち戻る～

2023年11月17日

NTTコムウェア株式会社
伊山　宗吉

登壇者紹介

いやま　むねよし

伊山 宗吉

NTTコムウェア(株) NTT IT戦略事業本部

2009年～　NTTの通信サービス関連の情報システム開発
2015年～　初めてアジャイル開発を経験
2018年～　数度の実践経験を経て社内推進活動を開始
　　現在に至る

主な業務：

- **社内のアジャイル推進**
- **アジャイルコーチ（案件のプロセス改善）**
- **アジャイルエバンジェリスト（社外へ情報発信）**

認定：

NTTコムウェアの事業

NTTドコモ、NTTグループ、通信を始めとする様々な業界の
お客様のビジネス拡大と業務DX 実現をミッションとしています

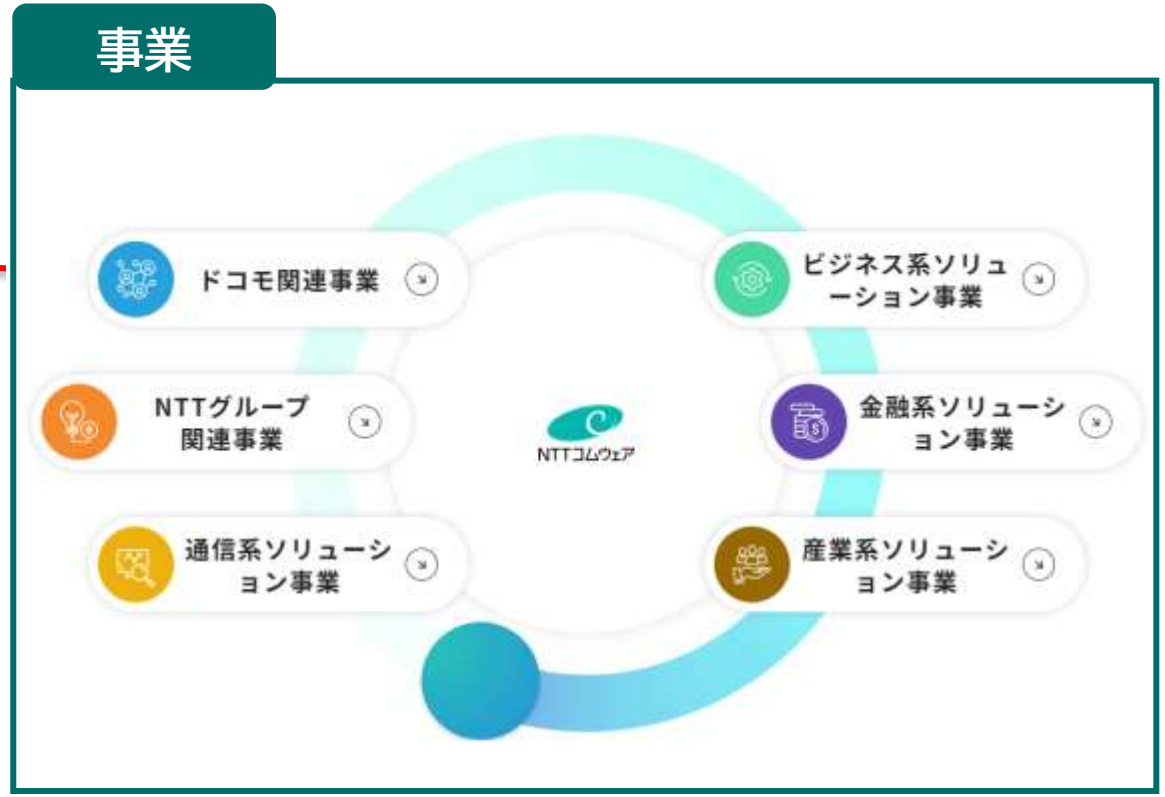
日本電信電話株式会社
NTT

総合ICT事業
docomo
NTT Communications
NTTコムウェア

地域通信事業
NTT東日本
NTT西日本

グローバル・ソリューション事業
NTT DATA

その他（不動産、エネルギー等）
NTTアーバンソリューションズ
NTTアノードエナジー



本日お話しすること

自社のアジャイル開発の推進や改善に取り組む中で多数の案件を見てきた経験をふまえて、アジャイル開発の原点を見つめ直し、改めて得た学びの一部を共有します。

・対象者：
  - アジャイル宣言の価値や原則を落とし込めていない方
  - チームで起きている問題の捉え方に悩んでいる方

・話さないこと
  - 具体的な案件の内容や解決事例

アジェンダ

1．NTTコムウェアでのアジャイル開発を振り返る
2．上手くいかなかったプロジェクトの特徴
3．アジャイルの原点に立ち戻る
4．学びとこれからの取り組み

1．NTTコムウェアでのアジャイル開発を振り返る

NTTコムウェアのアジャイル推進の状況

2013年頃から取り組み始め、
2016年に推進組織が発足した

- 業界動向もあり、案件数は年々増加
- 2020年以降は大きな変化なし

推進活動

アジャイル推進、
アジャイルコーチ

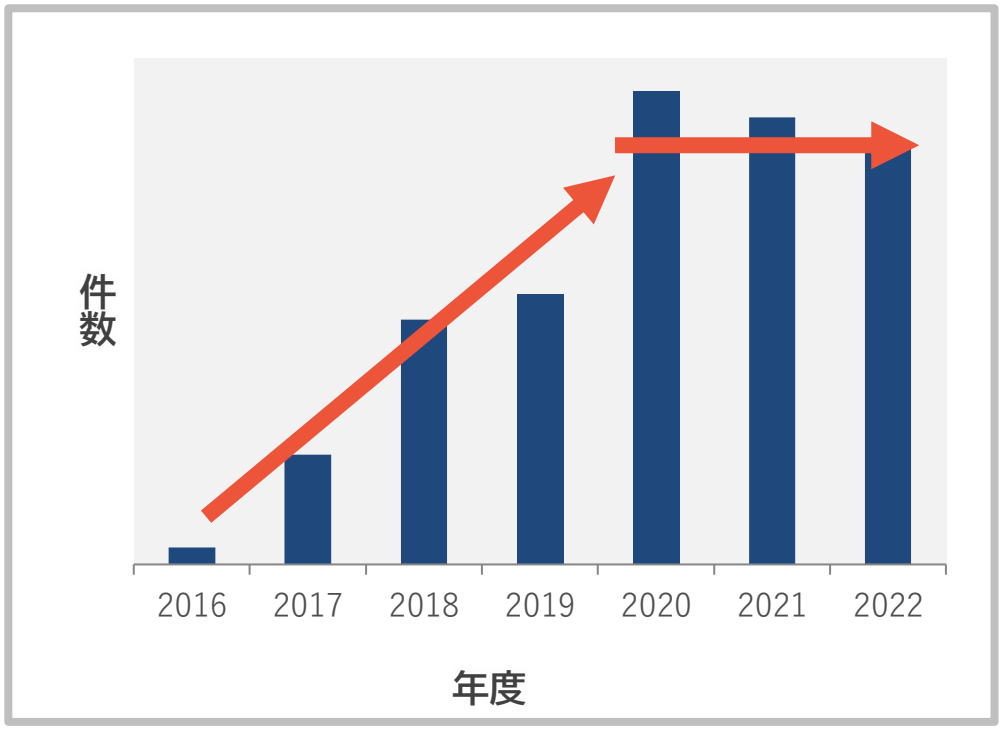
案件支援 → 

研修、
情報発信 → 
スクラムチーム、組織、社内

ルール整備 → 

管理部門、人事、財務、知財 など

案件数



アジャイルに取り組んだチームの状況

成功しているチームもあれば、それ以上に苦戦しているチームも多く存在。
⇒ これらの成否を分けた要因は何だったのだろうか？

２．上手くいかなかったアジャイル開発の特徴

その１：開発前の予兆

開発が始まる前から**怪しい匂いが漂っていた**案件はやはり上手くいかず、
次第に淘汰されていった。

アジャイル開発が目的化？

上司やお客様が
「**アジャイルで**」

提案書に
「アジャイル」
と入れたくて…

誤用？悪用？

要件が決まらなかった
のでアジャイルで…

アジャイルだと
工程毎のクオリティゲートが簡略化
できるし…

経験者不足で大きな開発？

アジャイルの**経験者は居ない**
けど、スコープと納期を
考えると**複数チームで**…

実質、従来開発？

予算、納期、スコープ
の計画は必達です

最初に設計スプリント、
次に製造スプリント、
最後に試験スプリント

その２：開発中に発生した問題

開発中も様々な問題が発生。
継続開発が難しくなったり、従来開発に戻る選択をするチームもあった。

ビジョン・ゴールを合わせられていない

POが多忙で対話不足
- 仕様の認識誤りが発生
- PBIが具体化されず着手できない

POが決められない
- 持ち帰りで待ちが多発
- 偉い人の意見が優先PBIに

リファクタリングや学習が後回しになる

システムが複雑化
- 変更コストが増大
- 不具合が増加

高スキル者に依存
- 不在時に開発が進まない
- 高スキル者が疲弊

従来開発と同じでリスクを終盤まで残している

すぐにはデプロイ／リリースができない
- ステークホルダーの興味が薄れ、レビューに参加しなくなる
- リリース前に1スプリント以上の試験期間が必要になる
- 試験で考慮漏れやバグが見つかり、リリースが危うくなる

ビジネスと開発の間に壁がある

計画を必達として扱う vs バッファを積む
- ビジネス側が初期計画や見積り結果を必達として開発に課す
- 開発側が見積りにバッファを加える、新しい要求や変更を拒む

参考：成功しているチームの特徴

前頁までのような問題は起きていない、
あるいは、問題に向き合い徐々に改善されていった。

共通していた特徴：

- ビジョンが共有されていた。
- 顧客と開発者が頻繁にコミュニケーションを取っていた。
- メンバーのモチベーションも高く、スキルアップにも積極的に取り組んでいた。

何が違ったのか？

上手くいかなかったチームはアジャイルの**形をなぞっただけ**で
大切な**中身が欠けていた**のではないだろうか。

- **中身とは何か？**
- **どうすれば中身を埋められるのだろうか？**

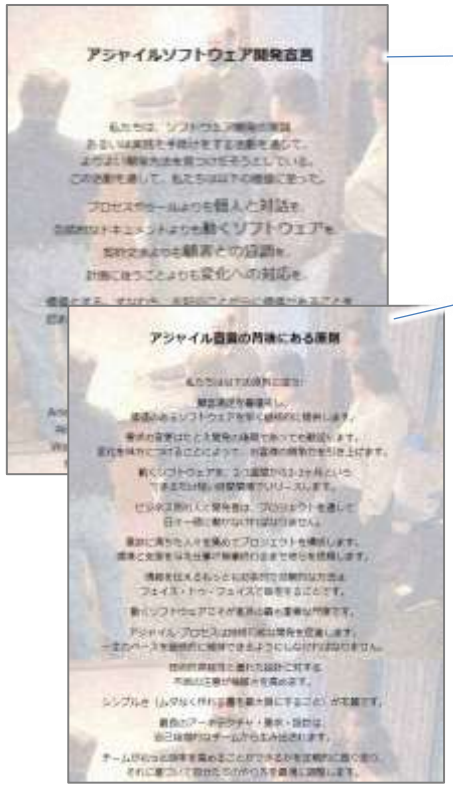
3．アジャイルの原点に立ち戻る

※本書ではアジャイル開発のフレームワークが登場し始めた1990年代以降を対象とします

アジャイルの原点を探る

アジャイル宣言に署名した中でスクラムやXPなどのフレームワークの考案者、実践者の書籍や論文から当時の考えに触れることで、よりアジャイル宣言の価値観や原則の意図を理解できるようになると考えた。

『アジャイルソフトウェア開発宣言』
『アジャイル宣言の背後にある原則』
（2001）



アジャイルソフトウェア開発宣言（2001）

- 4つの価値
- 12の原則

様々なアジャイル開発フレームワーク

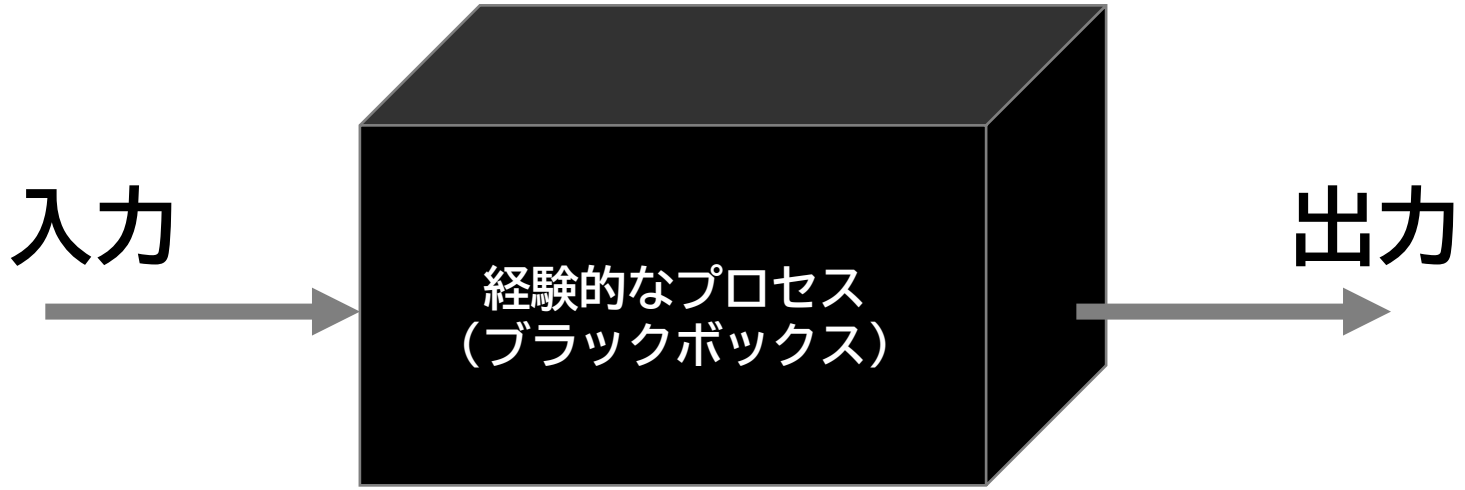
これらは宣言よりも先に登場していた

- スクラム
- XP
- …

様々なアジャイルプラクティス

- …
- バーンダウンチャート
- テスト駆動開発
- …

開発プロセスについて

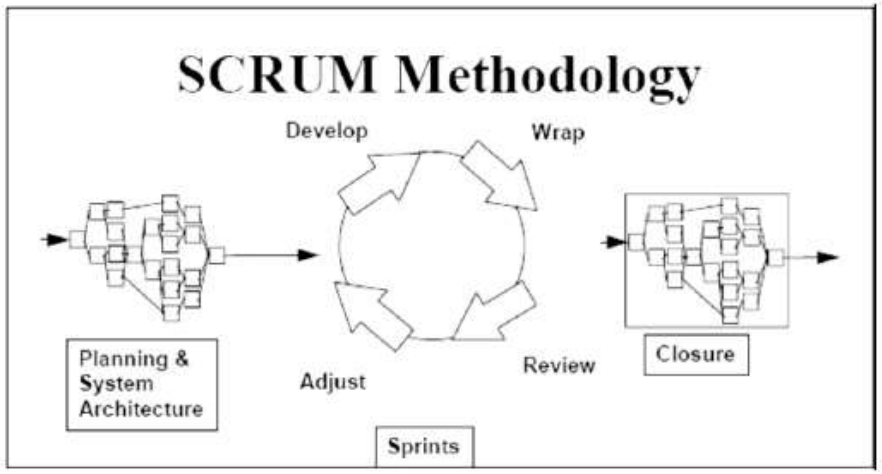


生化学などの**産業プロセス制御**において

- 設計して繰り返し実行することで毎回同じ結果が得られる（**予測可能**な）場合、
  ⇒ **定義されたプロセス**

プロセスが完全には解明されておらず単純計算できない（**予測不可能**な）場合、
⇒ **経験的なプロセス**
ブラックボックスのように扱い、**実験で取得したデータを元に制御**する。
（反復試行で得られた経験的な結果が、制御装置の運転条件に落とし込まれる
実行には頻繁かつ継続的な測定、監視、制御が行われる。）

最初に発表されたスクラムの論文(1995)



スクラムは「システム開発」という経験的プロセスを扱うために考えられたフレームワーク

p.10の図6より転載

出典：『SCRUM Development Process』 Ken Schwaber(著)

(アジャイル宣言の署名者の一人で、スクラム考案者の一人)

(OOPSLA'95 Business Object Design and Implementation Workshop, 1995)

以下、本文より引用(発表者翻訳)

"本稿では、**産業プロセス制御の概念をシステム開発の分野に適用**する。"

"調査の結果、我々はシステム開発プロセスは**経験的なもの**であると断言する"

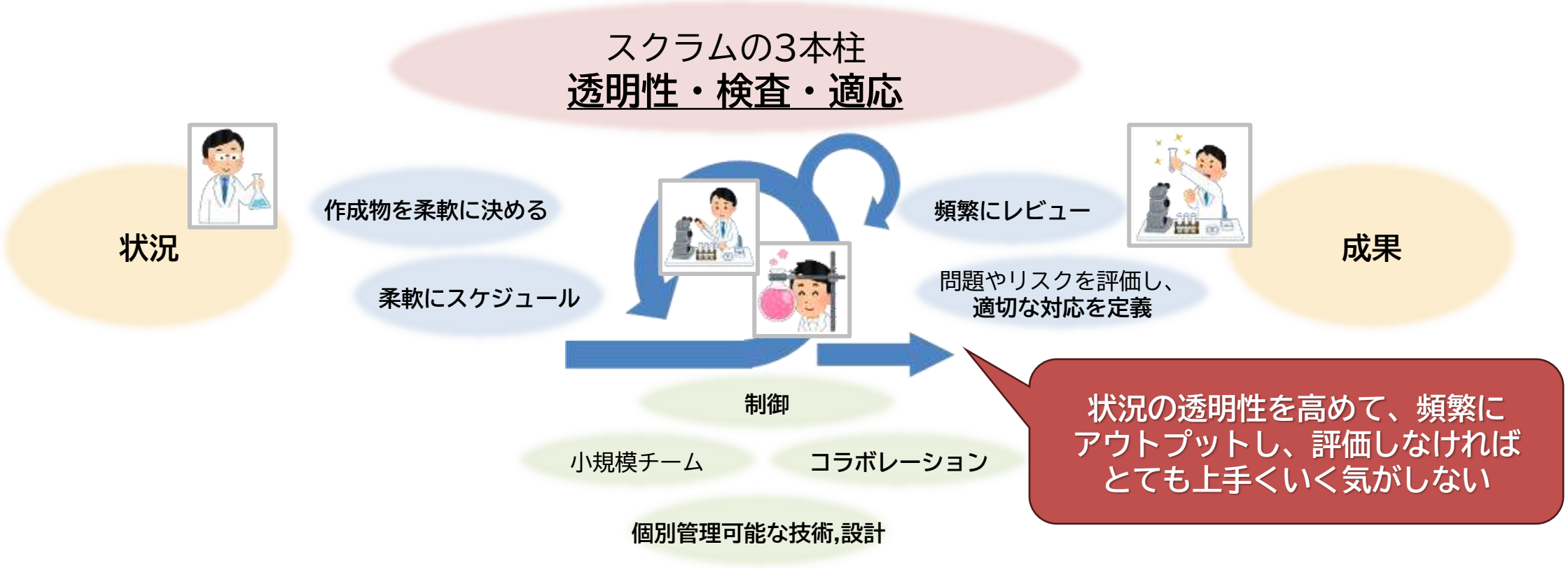
"スクラムは、システム開発プロセスが**予測不可能で複雑**なプロセスであり、
全体的な進行として大まかにしか説明できないことを前提としている"

"スクラムでは、これらのシステム開発プロセスを**制御されたブラックボックス**として扱う"

"予測不可能性を管理し、リスクを制御するために制御メカニズムが使用される。
その結果、柔軟性、応答性、信頼性が得られる。"

システム開発プロセスの制御（スクラムからの学び）

システム開発プロセスそのものが**生物学や化学の分野と同様に複雑で予測不可能なため、経験的なアプローチを必要とする**ことを改めて認識したことで、私の中でアジャイルおよびスクラムの価値や原則、フレームワークへの理解がより深まった。



（再考）

その１：開発前の予兆

開発が始まる前から**怪しい匂いが漂っていた**案件はやはり上手くいかず、
次第に淘汰されていった。

アジャイル開発が目的化？

上司やお客様が「アジャイルで」

提案書に「アジャイル」と入れたくて…

誤用？悪用？

要件が決まらなかったのでアジャイルで…

アジャイルだと**工程毎のクオリティゲートが簡略化**できるし…

経験者不足で大きな開発？

アジャイルの経験者は居ないけど、スコープと納期を考えると**複数チーム**で…

実質、従来開発？

予算、納期、スコープの計画は必達です

最初に設計スプリント、次に製造スプリント、最後に試験スプリント

生化学的手法を用いて特定の目的に沿った機能や特性を持つ物質を新しく合成しようとした場合、このような体制・計画を立てて上手くいくだろうか？

とてもリスクの高い選択をしていることを理解する必要がある

（再考）

その２：開発中に発生した問題

開発中も様々な問題が発生。
継続開発が難しくなったり、従来開発に戻る選択をするチームもあった。

ビジョン・ゴールを合わせられていない

POが多忙で対話不足
- 仕様の認識誤りが発生
- PBIが具体化されず着手できない

POが決められない
- 持ち帰りで待ちが多発
- 偉い人の意見が優先PBIに

リファクタリングや学習が後回しになる

システムが複雑化
- 変更コストが増大
- 不具合が増加

高スキル者に依存
- 不在時に開発が進まない
- 高スキル者が疲弊

従来開発と同じでリスクを終盤まで残している

すぐにはデプロイ／リリースができない
- ステークホルダーの興味が薄れ、レビューに参加しなくなる
- リリース前に1スプリント以上の試験期間が必要になる
- 試験で考慮漏れやバグが見つかり、リリースが危うくなる

ビジネスと開発の間に壁がある

- ビ
  結
- 開発側が見積りにバッファを加える、新しい要求や変更を拒む

10

認識齟齬から期待と異なるものが出来上がるリスクが高い

セットで重要

期待とのギャップに気づけず、適切な制御へのフィードバックも行えないまま進んでしまう

計画、品質について

ステークホルダーの期待

関係者の期待はどれもまっとうなものに見えるし、私もこれに応えたいと思う。

出典：『Clean Agile 基本に立ち戻れ』
（Robert C.Martin(著);
角征典, 角谷信太郎(訳)、ドワンゴ、2020）

（著者はアジャイル宣言の署名者の一人）

（以下、第2章より引用）

見積りと計画に関する期待

- **“あなたの見積りが誠実であることを期待する”**
- “ビジネス側の観点から機能が動作しているように見えれば、それはすでに完成していることを期待するだろう。そこから安定化のために、**QA作業で1か月待たされることは期待していない。**”
- “品質が高く、欠陥が少ないシステムを提供して欲しい”
- “新しく**リリースされるものは徹底的にテストされていることを期待する**。資金や時間が足りないことを理由に、開発チームがテストを省略することは期待していない。”
- “ソフトウェアは変更しやすいことを期待している。”
- “**ソフトウェアチームの速度が時間とともに遅くなることは期待していない**。2週間かかった機能とよく似たものを作ろうとしたときに、たとえ1年後であっても、同じく2週間かかることを期待する。”

アジャイルと計画（XP）

出典：『Clean Agile 基本に立ち戻れ』
（Robert C.Martin(著);
角征典, 角谷信太郎(訳)、ドワンゴ、2020）

（以下、第1章 より引用）

- “マネージャーがチームに「進捗どうですか？」と聞くと、希望があるので「順調です！」と答えてしまう。**希望はソフトウェアプロジェクトをマネジメントするのに最悪の方法**だ。**アジャイルは早い段階から希望を殺し、継続的に冷たくて厳しい現実を提供する**。”
- “**アジャイルは速く進むことだと思っている人もいる。だが、そうではない。**”
“**アジャイルとはどれだけうまくいっていないかをできるだけ早く知る**ことだ。できるだけ早く知りたいのは、**そうすれば状況をマネジメントできる**からだ。”

出典：『エクストリームプログラミング』
（Kent Beck, Cynthia Andres(著);
角征典(訳)、オーム社、2015）

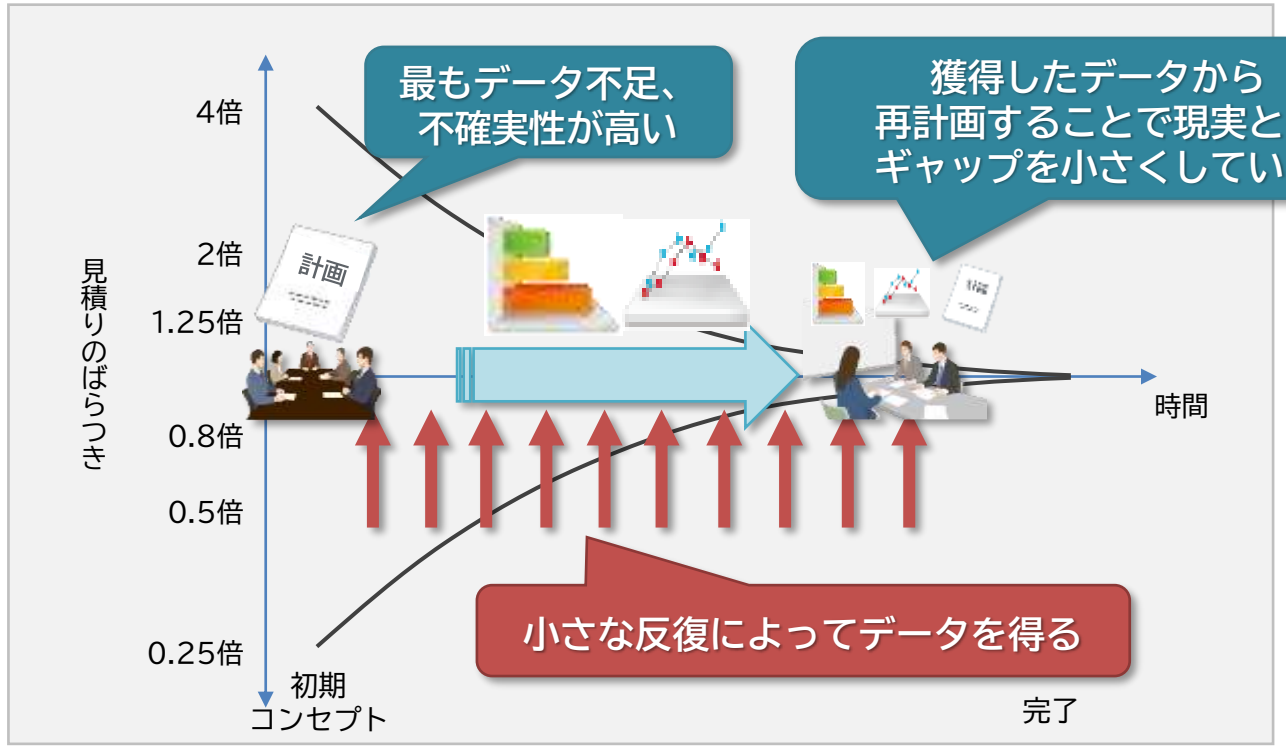
（著者はアジャイル宣言の署名者の一人で、XP(エクストリームプログラミング)の考案者）

（以下、第10章 より引用）

- “XPの計画づくりは活動であり、フェーズではない。プロジェクトマネージャーには、**計画と現実を同期し続ける**責任がある。”

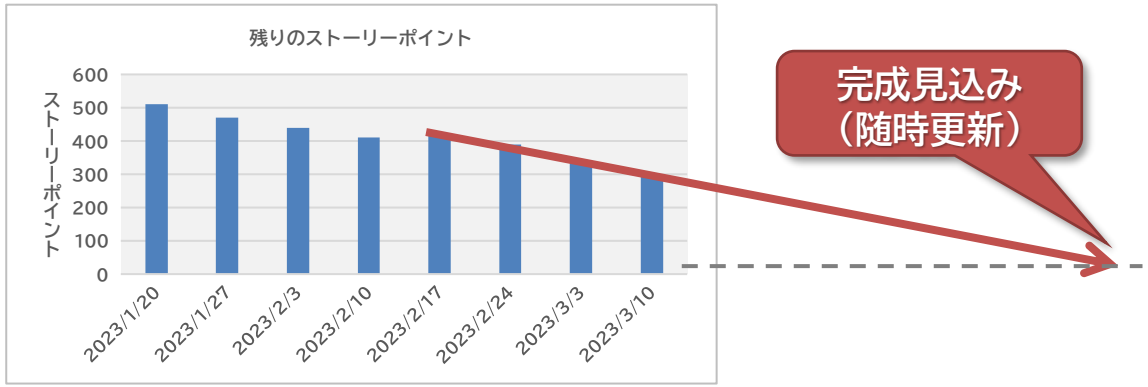
アジャイルはデータを提供する

- 小さな反復は「チームが関係者と共に現実を認識し、現実に適応するために再計画する」。そのためのデータを得る活動でもあることが改めて認識できた。
- この「現実に適応」する考えは、プロダクトの成長にも適用できる。



不確実性のコーン

出展「ソフトウェア見積り 人月の暗黙知を解き明かす」 1部4章 図4-1を参考に発表者が作成
（Steve McConell(著); 田沢恵、溝口真理子(訳), 久手堅憲之(監修)、日経BP社、2006年）

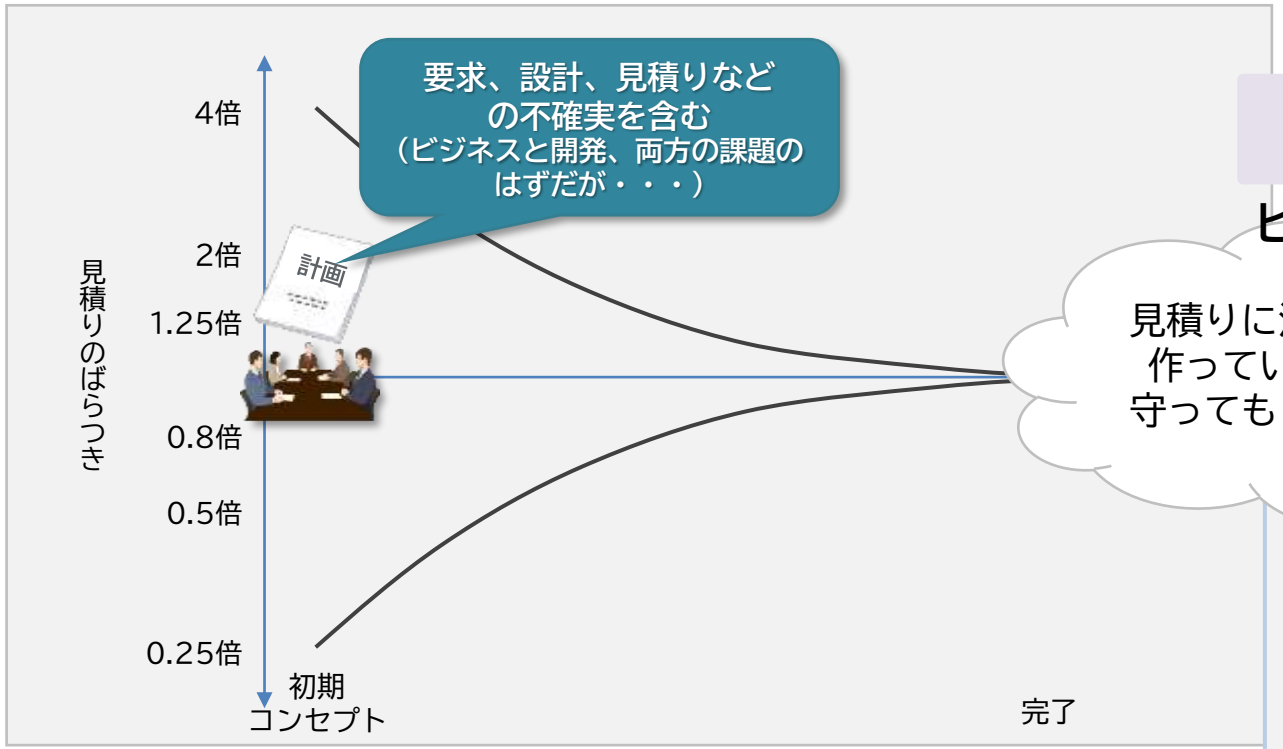


リリースバーンダウンチャート

出典：「Clean Agile 基本に立ち戻れ」の図1-3を参考に発表者が作成
（Robert C.Martin(著)； 角征典, 角谷信太郎(訳)、ドワンゴ、2020）

不確実な計画を必達として扱うと

- ビジネス側が計画を必達として扱うと、開発側は**新しい要求の追加や変更を拒む**ようになる。
- 開発側は変更や遅れへの対処として**バッファを設ける**心理が働いてしまう。
- ビジネスと開発の間の壁によって**プロダクトの価値向上の機会を失ってしまう。**



不確実性のコーン
出展「ソフトウェア見積り 人月の暗黙知を解き明かす」 1部4章 図4-1を参考に発表者が作成
（Steve McConell(著); 田沢恵、溝口真理子(訳), 久手堅憲之(監修)、日経BP社、2006年）

計画を必達として扱う vs バッファを積む

ビジネス

見積りに沿って計画を
作っているんだから
守ってもらわないと…

開発

必達になるなら仕様凍結して
もらわないと…
変更・遅延のリスクもあるし
最後までバッファは残したい…

アジャイル開発でなぜ動作するシステムが重要なのか

- **予測の限界**：ビジネス要求や開発の設計、見積りを事前に完璧にすることは困難である。
- 動作するシステムの利点：
  - フィードバック：期待と動作のギャップから、**実際の問題を特定することができる**。
  - 再計画：**実態に即したデータが得られる**ため、**現実的な計画への調整**が可能になる。

出典：『The New Methodology』（MartinFowler.com）
（**Martin Fowler (著)**、2005/12/13、参照 2023/11/16）
https://www.martinfowler.com/articles/newMethodology.html

（著者はアジャイル宣言の署名者の一人）

（以下、本文より引用（発表者翻訳））

“あらゆるプロジェクトに現実を突きつけるためには、
テスト済みの統合されたシステムほど優れたものはない。
文書にはあらゆる種類の欠陥が隠れる可能性がある。
テストされていないコードには多くの欠陥が隠れている可能性がある。
しかし、**人々が実際にシステムの前に座ってそれを操作するとき、**
バグの観点 と 誤解された要件の観点 の両方で欠陥が真に明らかになる。”

（再考）

その２：開発中に発生した問題

開発中も様々な問題が発生。
継続開発が難しくなったり、従来開発に戻る選択をするチームもあった。

ビジョン・ゴールを合わせられていない

POが多忙で対話不足
- 仕様の認識誤りが発生
- PBIが具体化されず着手できない

POが決められない
- 持ち帰りで待ちが多発
- 偉い人の意見が優先PBIに

リファクタリングや学習が後回しになる

システムが複雑化
- 変更コストが増大
- 不具合が増加

高スキル者に依存
- 不在時に開発が進まない
- 高スキル者が疲弊

従来開発と同じでリスクを終盤まで残している

すぐにはデプロイ／リリースができない
- ステークホルダーの興味が薄れ、レビューに参加しなくなる
- リリース前に1スプリント以上の試験期間が必要になる
- 試験で考慮漏れやバグが見つかり、リリースが危うくなる

ビジネスと開発の間に壁がある

計画を必達として扱う vs バッファを積む
- ビジネス側が初期計画や見結果を必達として開発に課す
- 開発側が見積りにバッファを加える、新しい要求や変更を拒む

不確実性の高い初期計画を前提に対立してしまい、プロダクトを成長させられずにチームの時間と労力を費やすことを**大きな損失**だとお互いが理解し、**現実に適応**することを一緒に考えるべきではないか。

（再考）

参考：成功しているチームの特徴

前頁までのような問題は起きていない、
あるいは、問題に向き合い徐々に改善されていった。

共通していた特徴：

- ビジョンが共有されていた。
- 顧客と開発者が頻繁にコミュニケーションを取っていた。
- メンバーのモチベーションも高く、スキルアップにも積極的に取り組んでいた。

11

生き残ったチームも、**顧客(ビジネス側)との対話**に必要なデータを集め、
実績値にもとづいて**スコープの再計画**の
調整をし始めたことが改善のきっかけ
だったように思う。

ステークホルダーの期待（再掲）

出典：『Clean Agile 基本に立ち戻れ』
（Robert C.Martin(著);
角征典, 角谷信太郎(訳)、ドワンゴ、2020）

（以下、第2章より引用）

品質と生産性に関する期待

- “あなたの見積りが誠実であることを期待す
- “ビジネス側の観点から機能が動作しているように見えれば、それはすでに完成していることを期待するだろう。そこから安定化のために、**QA作業で1か月待たされることは期待していない。**”
- “品質が高く、欠陥が少ないシステムを提供して欲しい”
- “新しく**リリースされるものは徹底的にテストされていることを期待する。**資金や時間が足りないことを理由に、開発チームがテストを省略することは期待していない。”
- “ソフトウェアは変更しやすいことを期待している。”
- **“ソフトウェアチームの速度が時間とともに遅くなることは期待していない。**2週間かかった機能とよく似たものを作ろうとしたときに、たとえ1年後であっても、同じく2週間かかることを期待する。”

リリース可能な品質の確保（XP）

出典：『Clean Agile 基本に立ち戻れ』
（Robert C.Martin(著);
角征典, 角谷信太郎(訳)、ドワンゴ、2020）

（以下、第2章 より引用）

- **“各イテレーションの終了時点でシステムが技術的にはデプロイ可能**な状態になっていれば、デプロイするかどうかはビジネス判断となり、技術的な判断ではなくなる。”
- “この状態の**コードはクリーン**で、**テストも全てパスしている**。”
- “毎週や隔週という頻度で達成できるだろうか？もちろんだ。イテレーションが終了するまでにデプロイに必要なすべてのタスクを完了できるように、ストーリーの量をチームで調整すればいい。テストも大部分を自動化しておいたほうがいいだろう。”

出典：『エクストリームプログラミング』
（Kent Beck, Cynthia Andres(著);
角征典(訳)、オーム社、2015）

（以下、第5章 より引用）

- “大きな変更が失敗してチームが無駄に後退するよりも、小さなステップのオーバーヘッドの方が小さい。ベイビーステップは**テストファーストプログラミング**や**継続的インテグレーション**などのプラクティスで表現されている。”

継続的インテグレーションの重要性（XP）

- 技術プラクティスのようだが **チームのプラクティス**と言われており、**継続的なフィードバックを実現する**。
- 経験的アプローチを用いて頻繁かつ継続的に状況を確認し、プロセス制御へフィードバックするためにも非常に重要な要素であると改めて認識できた。

出典：『Clean Agile 基本に立ち戻れ』
（Robert C.Martin(著);
角征典, 角谷信太郎(訳)、ドワンゴ、2020）

（以下、第1章 より引用）

“継続的インテグレーションは、**チームが現在地を常に把握できるように、フィードバックループを何度も閉じることにフォーカス**するプラクティスである。”

- 他者の変更に気づく、
- コンフリクトに気づく、
- 自動テストの失敗に気づく、
- など

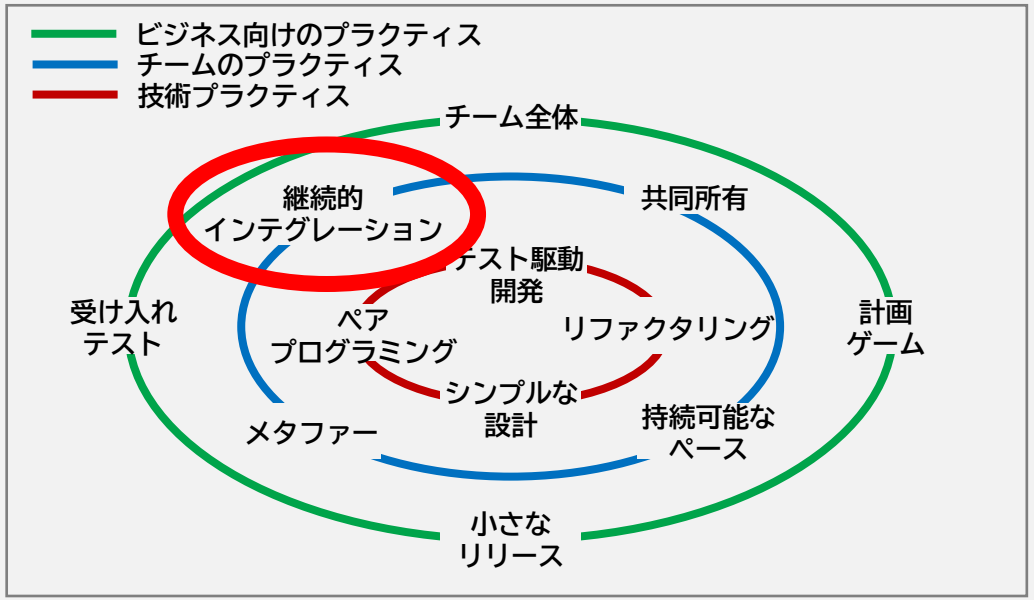


図1-8を参考に発表者が作成

テクニカルプラクティスの重要性（XP）

継続的インテグレーションのフィードバックループを機能させるために、
技術プラクティスが不可欠であることを改めて気づかされた。

出典：『Clean Agile 基本に立ち戻れ』
（Robert C.Martin(著);
角征典, 角谷信太郎(訳)、ドワンゴ、2020）

（以下、第1章 より引用）

“**これらのプラクティスこそがアジャイルの核心**だからだ。
「TDD」「リファクタリング」「シンプルな設計」そしてもちろん
「ペアプログラミング」。
**これらがなければ、アジャイルは本来意図されたものではない、
骨抜きにされた役立たずなものになってしまうだろう。**”

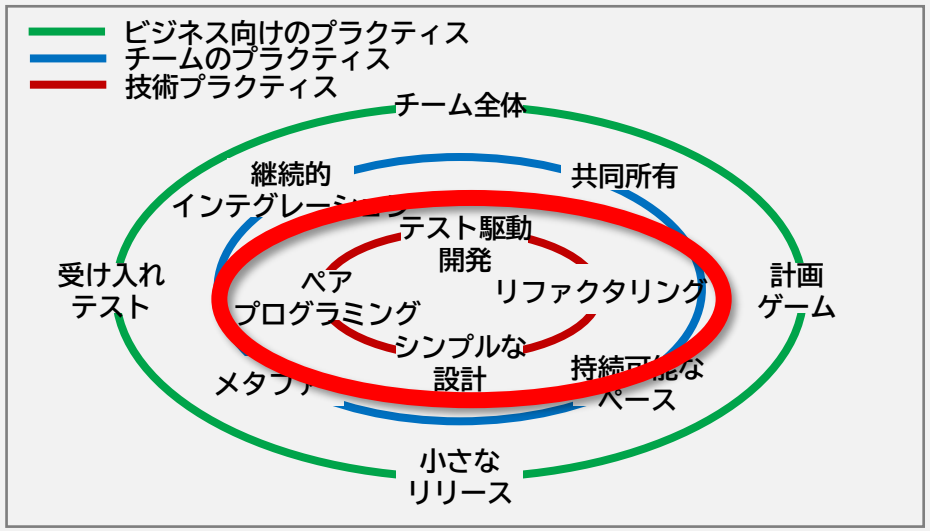


図1-8を参考に発表者が作成

（再考）

その１：開発前の予兆

開発が始まる前から**怪しい匂いが漂っていた**案件はやはり上手くいかず、
次第に淘汰されていった。

アジャイル開発が目的化？

上司やお客様が「**アジャイルで**」

提案書に「アジャイル」と入れたくて…

誤用？悪用？

要件が決まらなかったのでアジャイルで…

アジャイルだと**工程毎のクオリティゲートが簡略化**できるし…

経験者不足で大きな開発？

アジャイルの**経験者は居ない**けど、スコープと納期を考えると**複数チーム**で…

実質、従来開発？

予算、納期、スコープ

自動化されたテスト、
継続的インテグレーション、
完成の定義、
などがクオリティゲートであり、
絶えず検証されていなければならない。

（再考）

その２：開発中に発生した問題

開発中も様々な問題が発生。
継続開発が難しくなったり、従来開発に戻る選択をするチームもあった。

ビジョン・ゴールを合わせられていない

POが多忙で対話不足
- 仕様の認識誤りが発生
- PBIが具体化されず着手できない

POが決められない
- 持ち帰りで待ちが多発
- 偉い人の意見が優先PBIに

リファクタリングや学習が後回しになる

システムが複雑化
- 変更コストが増大
- 不具合が増加

高スキル者に依存
- 不在時に開発が進まない
- 高スキル者が疲弊

従来開発と同じでリスクを終盤まで残している

すぐにはデプロイ／リリースができない
- ステークホルダーのレビューに参加
- リリース
- 試験で考慮漏れやバグが見つかり、リリースが危うくなる

Copyright © NTT COMWARE CORPORATION 2023

10

関係者の期待に応えるためにも
チームでクリーンなコードとテストを保つことの必要性を関係者全員と合意しておきたい。

４．学びとこれからの取り組み

原点から得た学び

- **システム開発プロセスそのものが複雑で予測不可能**なため経験的アプローチが必要
- **頻繁かつ継続的にフィードバック**ループを閉じ、現在地を知ることから始まる。
  - 開発プロセスの制御へフィードバックする
  - リリース計画や成長計画を現実に適応し続ける
- **フィードバックループの核**となるのが**継続的インテグレーション**であり、
  それを支えるのが**技術プラクティス**である。

上記に加えて以下も重要

- **ビジョン、ゴールの共有**
- ビジネスと開発の**頻繁なコミュニケーション**
- メンバーの**モチベーション**と**スキルアップ**

形だけのアジャイルから中身のあるアジャイルへ

学びを活かして今後取り組んでいきたいこと

アジャイル開発の価値・原則の理解促進

強化したい

自社のマネージャー向けに研修を定期的に
実施しているが、**自社の幹部クラス、
顧客(ビジネス側)のPOやマネージャー、幹部クラス
に対しても働きかけていきたい。**

今回の学びを活かし、アジャイルの価値、原則が腹落ち
し、複雑な開発プロセスを制御するための活動を理解・
合意してもらえるようにしたい。

アジャイルチーム、プロジェクトの評価

強化したい

現在はスクラムガイドに体制、イベント等が
沿っていればアジャイル開発案件として社内の
クオリティゲートを省略できる仕組みを設けているが、

今回の学びから**最初の計画時だけでなく、反復を通じて
適切にフィードバック獲得と再計画が行われているか**も
判断に含めることを検討したい。

アジャイル開発を実践できる人材育成

取り組み中

各レベルに応じた人材育成

1. **初級**：3日間の研修（新入社員は必須）

実践

2. **中級**：3か月間の実践トレーニング

アジャイル開発の原則：アジャイル開発の原則に基づき活動
PBL(Project-based Leaning)：チームで成果を出す活動

-アジャイルマインドの修得
-開発スキルの修得
-実務能力の修得

更なるスキルの高度化

3. **上級**：道場(領域を特化して、得意技を磨く)

クラウド　フロント　バック

ご清聴ありがとうございました

参考文献

- 『アジャイルソフトウェア開発宣言』（2001）
  https://agilemanifesto.org/iso/ja/manifesto.html
- 『アジャイル宣言の背後にある原則』（2001）
  https://agilemanifesto.org/iso/ja/principles.html
- 『SCRUM Development Process』（Ken Schwaber(著), OOPSLA’95Business Object Design and Implementation Workshop, 1995）
  https://jeffsutherland.com/oopsla/schwapub.pdf
- 『Clean Agile 基本に立ち戻れ 基本に立ち戻れ』（Robert C.Martin(著)；角征典，角谷信太郎(訳)、ドワンゴ、2020）
- 『エクストリームプログラミング』（Kent Beck，Cynthia Andres(著)；角征典(訳)、オーム社、2015）
- 『ソフトウェア見積り 人月の暗黙知を解き明かす』（Steve McConell(著)；田沢恵、溝口真理子(訳)，久手堅憲之(監修)、日経BP社、2006年）
- 『The New Methodology』（Martin Fowler（著)、2005/12/13、参照 2023/11/16）
  https://www.martinfowler.com/articles/newMethodology.html