アジャイル開発講座

目 次

第１章 アジャイル開発の概要

- アジャイル開発の概要
- アジャイル宣言とアジャイル原則
- アジャイルの価値観
- パラダイム・シフト

アジャイル開発は、従来の開発手法と比べてどのような特徴があるのでしょうか？

まずは、親しんでいる方も多いウォータフォール開発の手法と比較しながら、アジャイル開発の特徴を確認していきます。

ウォーターフォール開発の特徴から確認していきましょう。

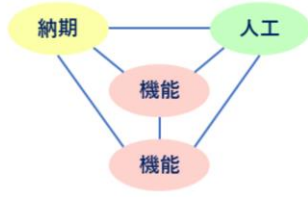
ウォーターフォール開発は、工程ごとに作業を完了し、前工程には立ち戻らないような開発手法を指します。具体的には、企画、要件定義、設計、実装、テストなど、開発工程をいくつかに切り分けます。要件定義が終わったら設計に、設計が終わったら実装に、というように、1つの工程が終わると次の工程に進みます。この手法は、作業の流れが一方通行であり、非常に簡潔でわかりやすいというメリットがあります。また、前工程に立ち戻らないということは、各工程を確実に終わらせて次に進まなくてはいけないような、工程内でのミスが許されないシステム開発に向い

ているといえます。しかしながら、前工程に立ち戻れないために発生するデメリットもあります。開発途中で顧客のニーズに変化があった場合には、途中で仕様を変更することができないため、ニーズに沿った商品を納品することは難しくなります。近年IT技術の進歩は目まぐるしく、顧客からのニーズも多様化しています。刻一刻と変化するニーズに対して、ウォーターフォール開発では対応しきれなくなってきました。

アジャイル開発は、開発するシステムを機能ごとに分割し、分割した機能に優先順位を付けて、優先順位の高い順に開発を進めていく手法です。ウォーターフォール開発が工程ごとに開発を進めることに対して、アジャイル開発は機能ごとに開発を進めます。つまり、アジャイル開発では、各機能の開発ごとに、分析、設計、実装、テストといった工程を繰り返します。開発途中にシステムの仕様に変更が出た場合、ウォーターフォール開発では、影響を及ぼす範囲が大きくなりすぎる懸念がありました。しかしアジャイル開発では、小さな機能が1つ完成した時点でその機能が顧客のニーズにマッチしているか確認することができます。もし、ニーズが変化していた場合でも、小さな機能を1つを変更することで対応が可能となります。これは、顧客のニーズに変化が生じた場合、最小限のシステム変更でニーズに対応することができるということを意味しています。一方で、アジャイル開発にもデメリットは有ります。仕様が変更しやすいといっても、何でも変更できるような状態で開発を進めてしまえば、結果システム自体が完成しないということも在りえます。変更できない前提条件を明確にしておくことが非常に大切です。また、複数のチームに分かれて短期的な開発を繰り返すため、プロジェクト全体の進捗管理が困難であるともいえます。ですから、開発者の経験値に大きく依存してしまい、習得が難しいということがアジャイル開発のデメリットと言えるでしょう。

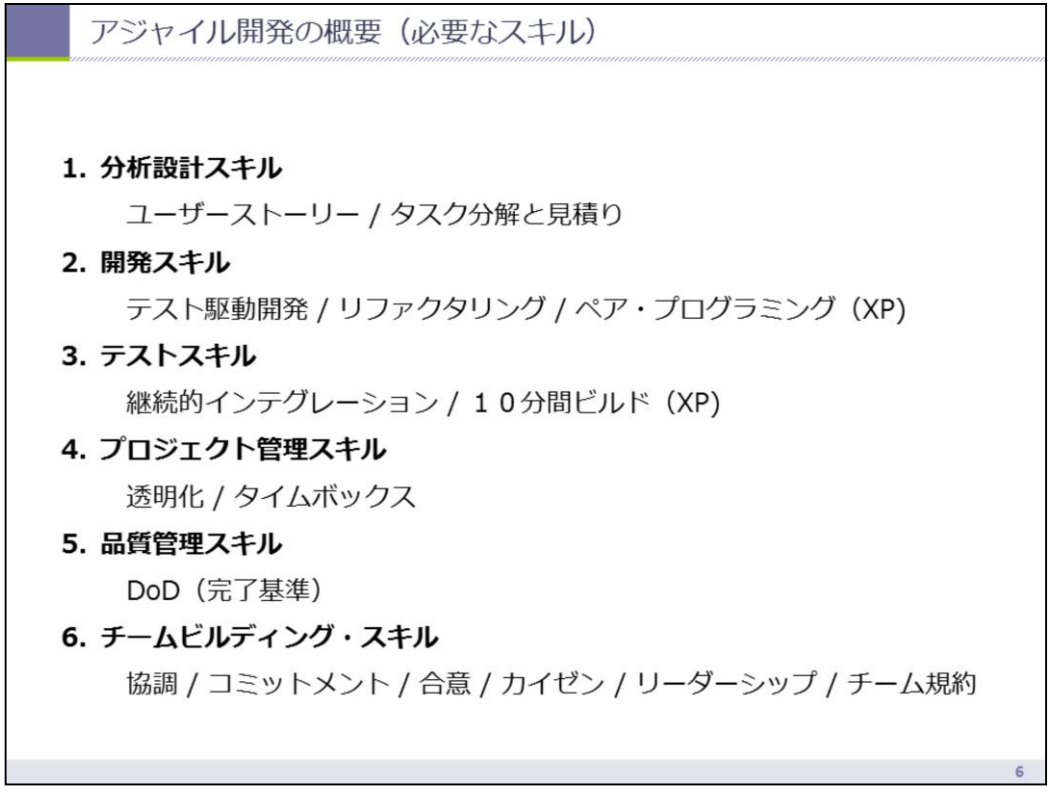
アジャイル開発の概要（特徴）2

◆ **スコープで調整**



◆ **常時リリース**

◆ **適応型開発プロセス**



ここでは、本講座で学んでいくアジャイル開発を実践する上で必要な知識やスキルをご紹介します。

分析設計スキルでは、ユーザーストーリーやタスク分解と見積もりについて学習します。

ユーザーストーリーとは、顧客の要求を簡潔にまとめたものです。つまり、顧客の要求を具体化・詳細化したものと言い換えることもできます。ユーザーストーリーの重要性や使用方法、作成方法について学んでいきます。

アジャイル開発では、短い期間の開発を繰り返し行いますが、その短い期間1つの間にチームがやらなくてはいけないことのリストがタスクです。このタスクのうち一人一人がどこを担当するのか決めるためにタスクを分解しますが、適切に分解するための手法や、そのタスクを適切に見積もる方法についても紹介します。

開発スキルでは、アジャイル開発の中でもXPと呼ばれる手法を例に取り、テスト駆動開発、リファクタリング、ペア

プログラミングについて学習します。

テスト駆動開発やリファクタリングの流れを確認するとともに、ペアプログラミングではどのようにペアを組むべきなのか、なんのためのペアなのか等を紹介します。

テストスキルでは、継続的インテグレーションと10分間ビルドについて学習します。

継続的インテグレーションと10分間ビルドが、アジャイル開発においてどのように有益なのかを紹介します。

プロジェクト管理スキルでは、見える化とタイムボックスについて学習します。

見える化実現のポイントやその有用性、タイムボックスという概念の重要性について紹介します。

品質管理スキルでは、DoDについて学習します。

DoDとは、各作業をどこまで行えば完了なのかという基準のことです。品質管理の基本ともいえる重要な項目ですから、基本的なDoDの考え方を学んでいきます。

チームビルディング・スキルでは、協調、合意、リーダシップ、コミットメント、カイゼン、チーム規約というような、良質なチームを形成するための要素について学習します。

一見当たり前のことばかりかもしれませんが、チームをいかに上手に形成するかは、プロジェクト成功に向けて最も重要な要素の1つですから、しっかり学んで行きます。

アジャイル開発概論

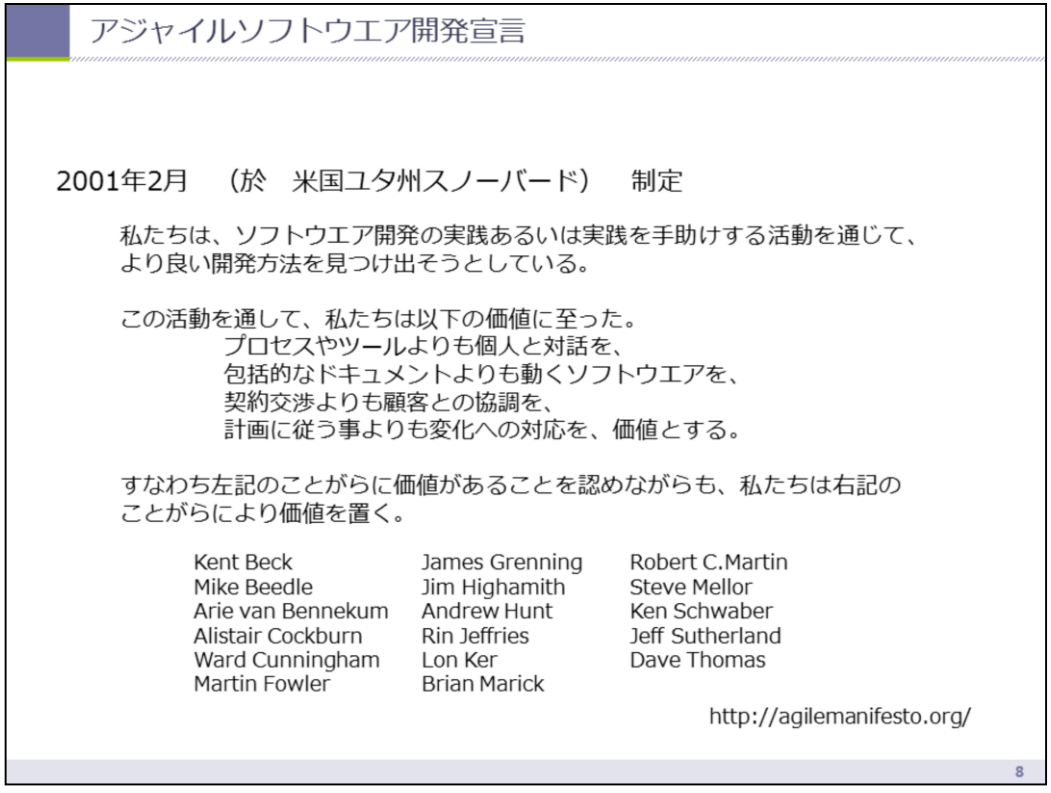
アジャイル開発とは、高品質で要求に柔軟なソフトウェア作成を行うため、
協調性、自律性を重視した反復的な開発方法である。

- ダイナミックシステムズ開発技法
  Dynamic Systems Development Method (Dane Faulkner ほか)
- アダプティブソフトウェア開発
  Adaptive Software Development (Jim Highsmith)
- クリスタルメソッド
  Crystal Methods (Alistair Cockburn)
- スクラム
  Scrum （Ken Schwaber, Jeff Sutherland)
- XP / エクストリームプログラミング
  (Kent Beck, Eric Gamma ほか)
- リーンソフトウェア開発
  Lean Software Development （Tom and Mary Poppendieck)
- フィーチャ駆動開発
  Feature-Driven Development (Peter Code, Jeff DeLuca)
- アジャイル統一プロセス
  Agile Unified Process (Scott Ambler)

- **反復（同期）的(Iterative)**
  --- 定期的なリリース
- **漸進的(Incremental)**
  --- 徐々に機能を増加
- **適応主義(Adaptive)**
  --- 変化に対応（即応）
- **自律的(Self-Organized)**
  --- 学習する組織
- **多能工(Cell Production)**
  --- 一人多役
  （ＳＥ、プログラマー、テスター）

7

ここでは、アジャイル開発の概論と、その技法の種類を紹介します。

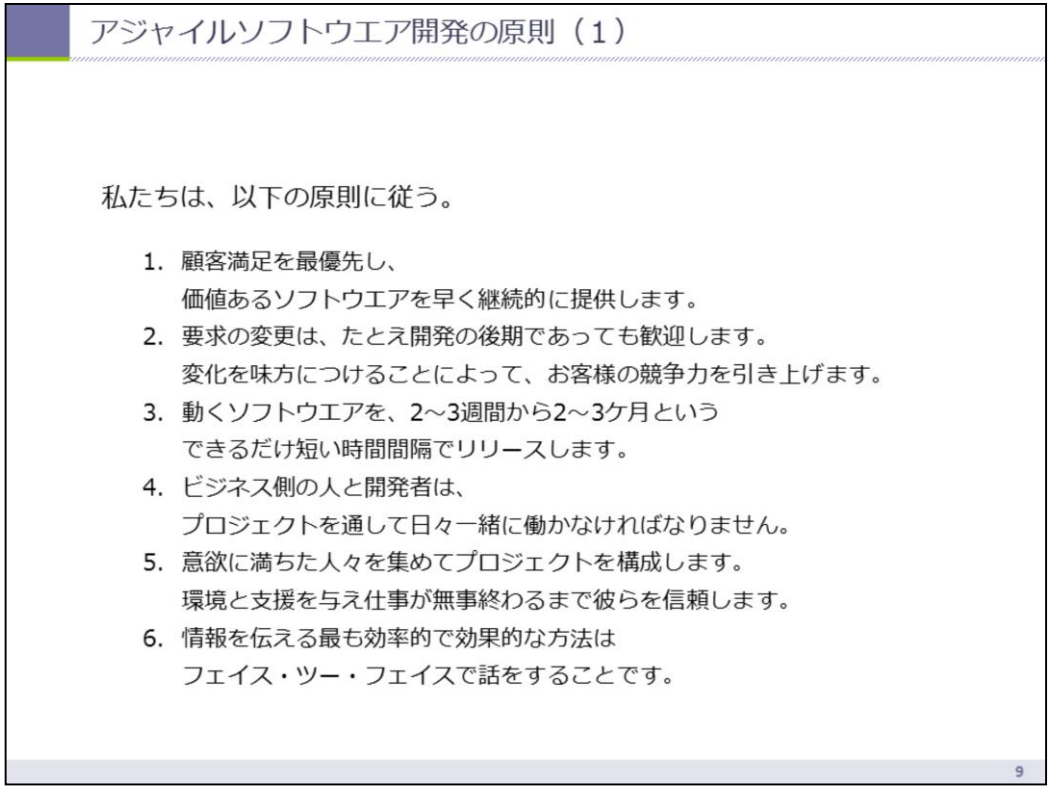


アジャイルソフトウェア開発宣言は、2001年2月、従来広く使われてきたソフトウェア開発手法とは違う開発手法を実践していた17名の開発者が集まり、それぞれの手法について議論が行われた結果生まれた宣言です。アジャイルソフトウェア開発宣言に記されているのは、具体的な手法ではなく、重視すべきマインドセットです。アジャイル開発を正しく身につけるためには、アジャイルソフトウェア開発宣言の解釈を誤り無く理解することが重要です。

宣言内にある「左記のことがら」とは、プロセスやツール、包括的なドキュメント、契約交渉、計画に従う事、を指しています。アジャイルソフトウェア開発宣言を一見すると、左記のことがらは重要視されていないように誤解されがちですが、「左記のことがらに価値があることを認めながらも」とあるように、決してアジャイル開発には不要なものであるということが言いたいわけではありません。より価値を置くとされている「右記のことがら」、個人との対話、動くソフトウェア、顧客との協調、変化への対応、というマインドセットがしっかりあることを前提に、左記

のことがらを考えていかなくてはいけないことを示しています。

この宣言は、次ページで紹介する「アジャイル宣言の背後にある原則」と合わせて自由にコピーできますが、この注意書きを含めて、全文を掲載しなければなりません。



アジャイルソフトウエア開発宣言には、更にこのような原則が付加さ

れています。この原則はソフトウエア開発で何を重視すべきかが謡わ

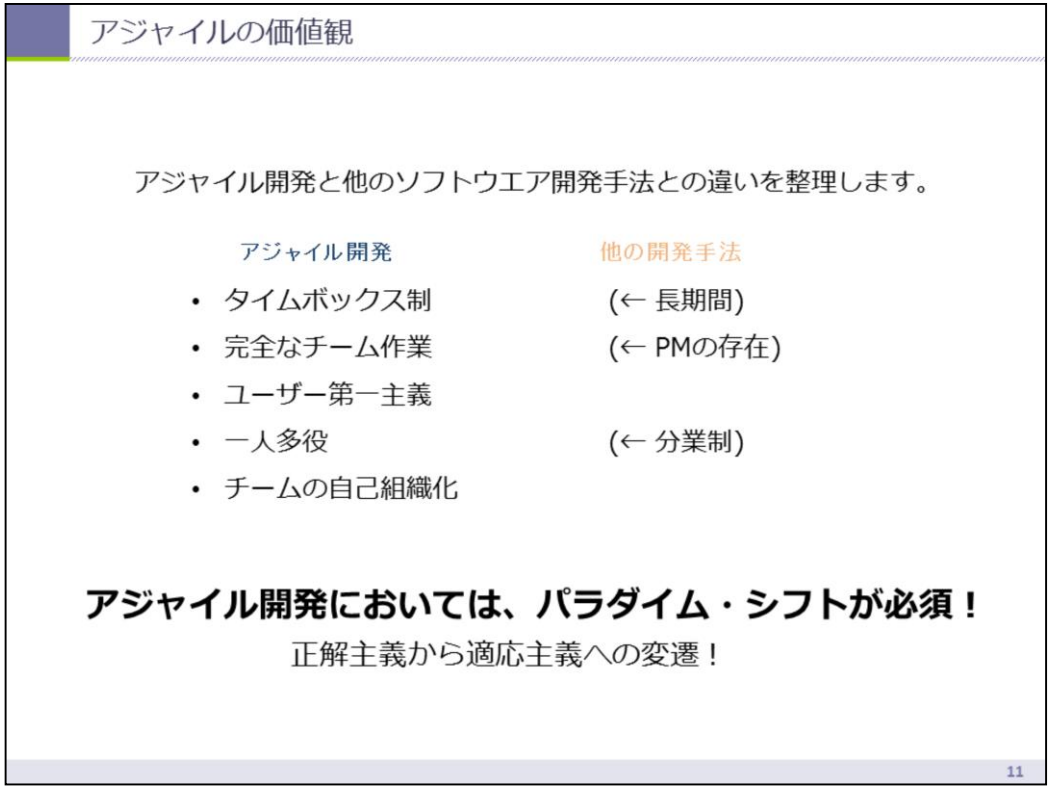
れています。これはソフトウエアの価値を最大化するための具体的な

行動指針です。

アジャイルソフトウエア開発の原則（2）

7. 動くソフトウエアこそが進捗の最も重要な尺度です。
8. 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。
9. 技術的卓越性と優れた設計に対する不断の注意が
   機敏さを高めます。
10. シンプルさが本質です。
11. 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。
12. チームが最も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。

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



アジャイル開発と他のソフトウェア開発手法との大まかな特徴の違いは前述しましたが、ここでは、もう少し具体的に違いを見ていきましょう。

アジャイル開発は、時間の使い方に特徴があります。アジャイル開発における、区切られた短い開発期間のことをイテレーションといい、イテレーションの中でやるべきことのリストをタスクといいます。イテレーションの開始時にゴールを設定し、ゴールから逆算してタスクを設定します。このような時間の設定方法をタイムボックスといいます。つまり、固定の時間（タイムボックス）内でやるべきことを明確にするということです。タイムボックスもできるだけ短い時間で区切るようにして、ダラダラと作業することを避けます。

チームでの開発にも重きを置いています。少数精鋭のチームを編成し、チーム内でも一人一人が自分の責任を常に意識しながら開発に取り組んでいくことで、結果、自己組織的なチームとして開発を有利に進められます。

要求の変化に対応できるのがアジャイル開発の特徴の1つでした。つまり、顧客の要求の変化にいかに答えていくかが重要視さ

れています。イテレーションの最後に顧客から受けたフィードバックを、積極的にプロジェクトに反映させていくことが大切です。そして、正解を求めて開発を進めるのではなく、変化への適応を受け入れながら開発を進めるのが、アジャイルの特徴です。

第２章 スクラム

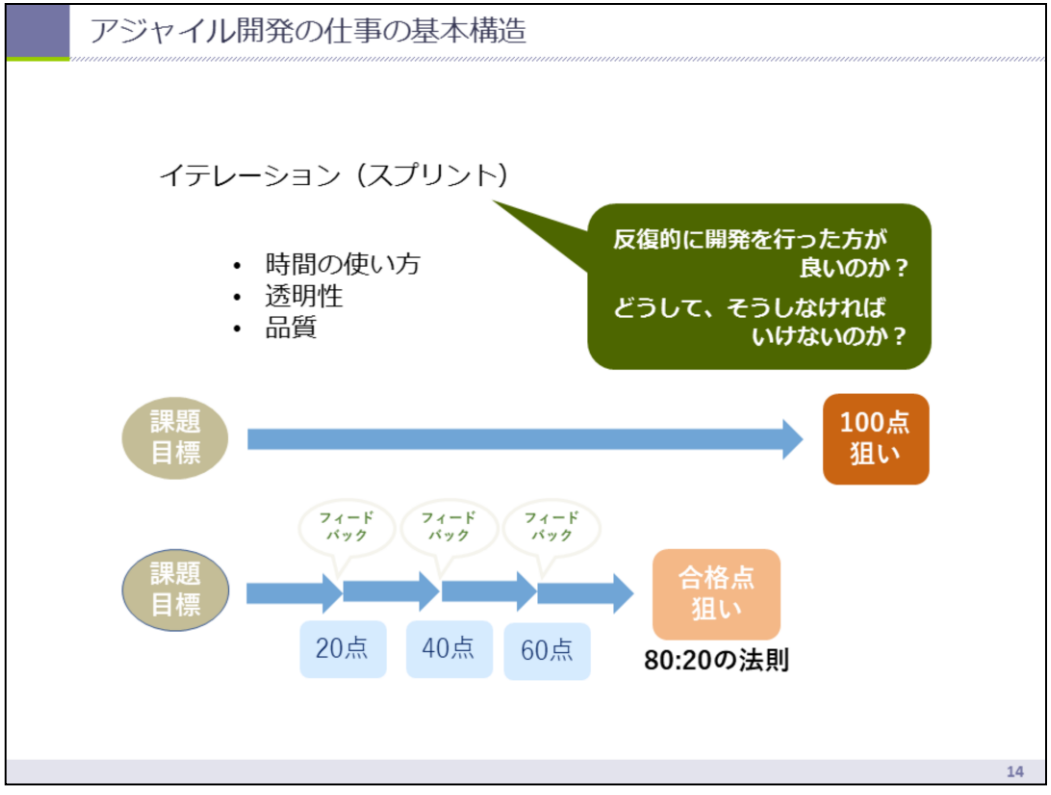
- スクラムの価値観（働き方）
- パラダイム・シフト

スクラムガイド（https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Japanese.pdf）によると、「スクラムは、経験的プロセス制御の理論（経験主義）を基本にしている。経験主義とは、実際の経験と既知に基づく判断によって知識が獲得できるというものである。スクラムでは、反復的かつ漸進的な手法を用いて、予測可能性の最適化とリスクの管理を行う。」とあります。つまり、ウォーターフォール開発のように予測から成るプロセスではなく、短い期間で何度も経験して成果に近づいていくプロセスであるといえます。また、スクラムガイドには、「経験的プロセス制御の実現は、透明性・検査・適応の 3 本柱に支えられている。」ともあります。経験を繰り返して開発を進めるために重要なのが、透明性・検査・適応の３つということです。

透明性とは、チーム内においてチームの現状（進捗状況や問題点等）が見える化されており、チーム全員が共通の認識を持つことができているかということです。チームの状況が明確にわかり全員がそれを共有していれば、自ずと次

にするべきことは見えてくるはずです。透明性が保たれることで、次の行動につながっていきます。

検査とは、成果物の出来やプロジェクトの進捗に問題がないか検査し、問題点を洗い出すことを指します。検査はその頻度設定が重要で、多いほど好ましいといえますが作業の妨げになるほど増やしてはいけません。

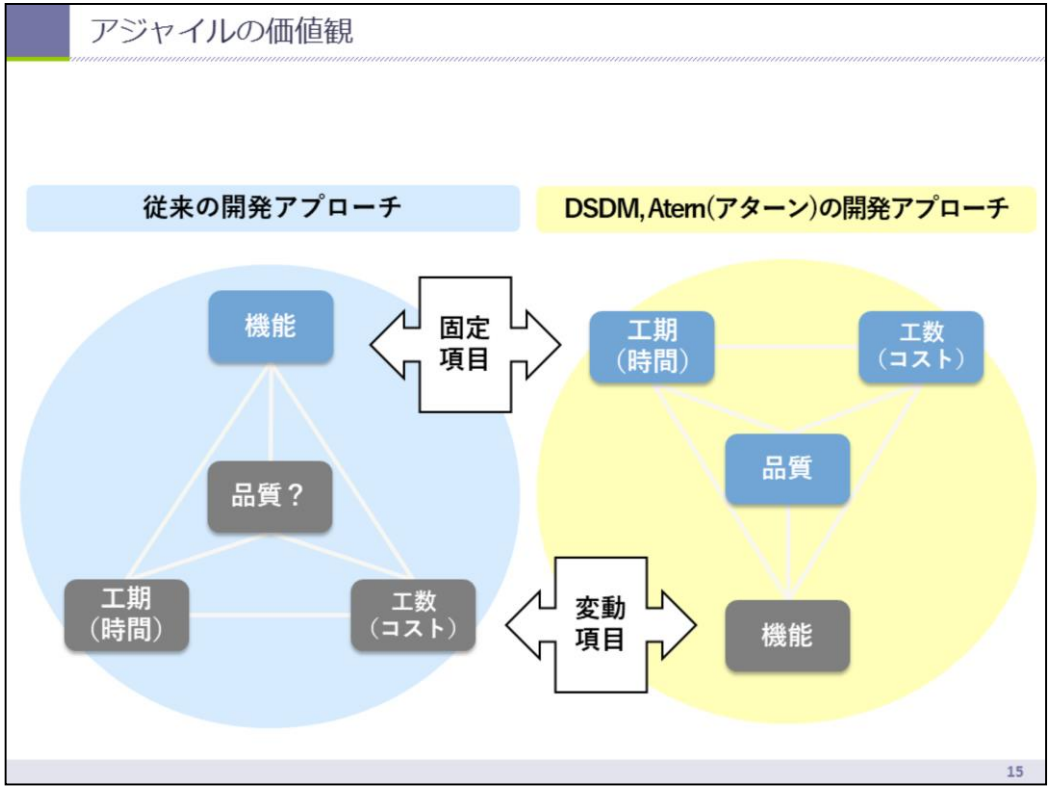


アジャイル開発は、短い期間での開発を反復的に何度も行うことでゴールに近づいていきます。その短い一回の開発期間の事をイテレーションといいます。一般的なアジャイル開発ではイテレーションといいますが、スクラムではイテレーションのことをスプリントといいます。

では、なぜ反復的に開発を行うのでしょうか？

反復的に開発を行う利点として、要求の変化に対応できること、優先順位の高い順に開発を進めるため動くソフトウェアを早い段階から確認できること、短期的であるため計画の見通しが立ちやすいこと、検査も繰り返し行うため問題の検知が早いこと、などが挙げられます。

そもそもアジャイル開発は、従来のウォーターフォール開発のような開発手法と比べて、柔軟性に富んで一線を画すという発想から成り立っています。この発想を現実化するためには、あえて短期的な計画しか明確にせず変化を受け入れる余地を残しておかなければいけません。そうなると、小分けに開発を進めていく手法が最も効率的なのです。



アジャイル開発の基本的な考え方を紹介します。従来の考え方、アプローチからは大きなパラダイムシフトになります。

従来は、まず開発すべき機能（要求）が大前提で、これだけの機能を実装（開発）する為には、これだけの期間（工期）と何人の体制（コスト）が掛かりますというプロジェクト計画の作成方法です。

アジャイルでは、逆の発想で、現在与えられているリソース、すなわち与えられた時間（工期）と予算（コスト）の範囲内で実現できる機能（要求）はどれだけか？というプロジェクト計画になります。

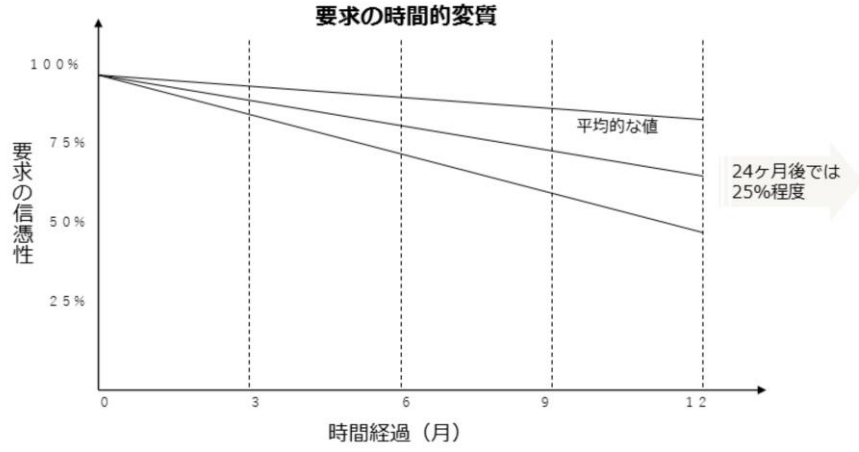
したがって従来の方法では、決められた機能を100%実装するために、工期、工数が不足してくると、品質を犠牲にするアプローチではないでしょうか？

アジャイルでは、工期、工数、品質を満たす事が前提で、実現される機能で調整を行う事になります。

要求の変質

迷信：要求を明確に定義できれば良い

現実：要求は時間と共に変化



16

仮に要求仕様をしっかりと定義できてもその要求自体が変質しています。

　開発初期における要求定義

　従来の、開発初期に明確な要求定義をすれば、良いシステムが

　できあがるという考え方は、データから見ても迷信である。要求の

　信憑性は時間の経過とともに下がるため、初期の要求を変更せ

　ず開発を進めることは品質の向上に繋がりにくいことを説明する。