アジャイル領域へのスキル変革の指針

アジャイルソフトウェア開発宣言の読みとき方

2018年4月

独立行政法人
情報処理推進機構

はじめに

- **本書の作成経緯・問題意識**

  アジャイルなアプローチで期待される成果を出すための秘訣として、方法論やプロセス、ツールを導入するだけではなく、考え方の規範となるマインドセットや原則を理解し実践することが重要です。

  本書は、その考え方の規範となる「Manifesto for Agile Software Development（アジャイルソフトウェア開発宣言）」を実践者共通のマインドセットと捉え「*Principles Behind the Agile Manifesto*（アジャイル宣言の背後にある原則）」をこれから日本でアジャイルを実践する方がよりよい結果を導き出すための行動指針とし、様々なケースで求められる状況判断のガイドとして役立ててもらうことを目的としています。

  実践経験の有無に関わらず、アジャイルソフトウェア開発宣言の存在が知られていない、一部解釈に関して誤解が生じているという問題意識から本書では、「読みとき方」としてアジャイルソフトウェア開発宣言が示している真意を読者に伝えることができるよう、*4*つの価値とその価値に由来する明確な*12*の原則についてそれぞれの解釈を行っています。

  まずは、受注者、発注者のような立場に関係なく、プロジェクト全員でアジャイルソフトウェア開発宣言の正しい解釈を身に着けることから始めましょう。

- **アジャイルソフトウェア開発宣言のマインドセットはこれからの時代に必要なもの**

  アジャイルソフトウェア開発宣言の最初には「私たちは、ソフトウェア開発の実践あるいは実践を手助けをする活動を通じて、よりよい開発方法を見つけだそうとしている。」と書かれています。

  アジャイルソフトウェア開発宣言はマインドセットですので、この一文の「ソフトウェア開発」と「開発方法」は、製品やサービスまたは働き方と読み替えても十分通じるものでしょう。

  人々の価値観の多様化や国際化によって常に変化し続けるビジネス市場の激流の中では、変化に適応しながら競争力を高めていく能力を育てていくことは大変重要です。

アジャイルソフトウェア開発宣言と アジャイル宣言の背後にある原則

「アジャイルソフトウェア開発宣言」は、従来型のソフトウェア開発のやり方とは異なる手法を実践していた17名のソフトウェア開発者が2001年に一堂に会し、それぞれの主義や手法についての議論を通して生み出されました。そこには、彼らがソフトウェア開発を行ううえで重視している「マインドセット」が書かれています。

その後、このマインドセットは世界中のソフトウェア開発者達に支持され”アジャイル開発” という名で世に広まりました。

一方「アジャイル宣言の背後にある原則」は「アジャイルソフトウェア開発宣言」で表明されているマインドセットを実現するために従うことが望ましい原則、つまり根本的・根源的な取り組み姿勢について書かれています。

本書は、「アジャイルソフトウェア開発宣言」および「アジャイル宣言の背後にある原則」を日本の企業風土、組織文化を考慮し、分かりやすく解釈することで、より多くの方にアジャイル開発を理解していただくことを目的としています。

アジャイルソフトウェア開発宣言

私たちは、ソフトウェア開発の実践
あるいは実践を手助けをする活動を通じて、
よりよい開発方法を見つけだそうとしている。
この活動を通して、私たちは以下の価値に至った。

プロセスやツールよりも個人と対話を、
包括的なドキュメントよりも動くソフトウェアを、
契約交渉よりも顧客との協調を、
計画に従うことよりも変化への対応を、

価値とする。すなわち、**左記のことがらに価値があること**を
認めながらも、私たちは**右記のことがらにより価値をおく**。

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

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

「アジャイルソフトウェア開発宣言」に対する誤解と真意

「アジャイルソフトウェア開発宣言」のうち価値について言及している文は「～よりも」とあることから一見すると左記のことがら「プロセスやツール、ドキュメント、契約、計画」は疎かにしてもよいと解釈されがちです。ここから、アジャイル開発ではドキュメントを作成しなくてもよいとか、計画は考えなくてもよいなどの誤解が生じることがよくあります。ですが、見落とされがちな「左記のことがらにも価値があることを認めながらも」という一文にあるとおり、「プロセスやツール、ドキュメント、契約、計画」にも価値があることを明言しています。よってアジャイル開発でも"価値のある"必要なドキュメントは作成しますし、事前に計画を立てて作業を進めていくことは言うまでもありません。また開発を効率的に進めるためには有用なツールを活用することも重要です。

アジャイルソフトウェア開発宣言で伝えようとしていることは、まずマインドセットがあって、そのうえで「プロセスやツール、ドキュメント、契約、計画」を考えるべきである、ということなのです。このマインドセットは「個人と対話、動くソフトウェア、顧客との協調、変化への対応」として簡潔に表現されていますが、後述の「原則」を踏まえてもう少し具体的に表すと次のようなことを示しています。

- 「対面コミュニケーション」

  個人同士の対話を通じて相互理解を深めることで、よりよいチームが作られる。

- 「実働検証」

  動くソフトウェアを使って繰り返し素早く仮説検証をし、その結果から学ぶことがよりよい成果を生み出す。

- 「顧客とのWin-Win関係」

  お互いの立場を超えて協働することにより、よりよい成果と仕事のやり方を作ることができる。

- 「変化を味方に」

  顧客ニーズやビジネス市場の変化は事前計画を狂わす脅威ではなく、よりよい成果を生み出す機会と捉える。

「アジャイル宣言の背後にある原則」の読みとき方

特に、次ページに示す「アジャイル宣言の背後にある原則」については、重要なポイントが一読して分かるタイトルをつけたうえで基本的な考え方と行動規範を加えて原則の読みとき方を記述しています。

基本的な考え方

各原則で重要となる「基本的な考え方」について説明しています。基本的な考え方には、その原則の本質的な解釈のほか、その原則がある理由、その原則を実現するための手段などについて記載しています。アジャイルに馴染みのない方でも理解できたり実感できるよう、平易な言葉で説明をしていますので、誰もが容易に基本的な考え方を理解できるようになっています。

行動規範

「行動規範」では、各原則を実践するうえで、どのような行動が望ましいか、またどのような行動が望ましくないか説明しています。各原則の行動規範を通じて、今までの行動を振り返り、今後はどのように行動すればよいか考えるきっかけになれば幸いです。

アジャイル宣言の背後にある原則

私たちは以下の原則に従う:

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。

要求の変更はたとえ開発の後期であっても歓迎します。
変化を味方につけることによって、お客様の競争力を引き上げます。

動くソフトウェアを、2-3週間から2-3ヶ月という
できるだけ短い時間間隔でリリースします。

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。

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。

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。

動くソフトウェアこそが進捗の最も重要な尺度です。

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。

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。

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。

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。

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。

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

原則01 顧客の満足を求め続ける

オリジナル

「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。」

“Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”

基本的な考え方

開発者にとって最も重要な活動は、価値のあるソフトウェアを素早く継続的に提供して、顧客に（QCDの達成ではなく）ビジネスゴール達成の観点で満足してもらうことです。

日本の多くの企業は近年の劇的な市場の変化への対応に迫られており、ビジネスを支えるITシステムの構築は、企業の存続を左右するほど重要な要素になっています。よって、開発者は、ビジネスの実現を最優先に考えてITシステムを構築する必要があります。

ただし、ビジネスの成否はビジネスごとに異なります。よって、開発者は、常に顧客と同じ目線で、顧客の満足を高める価値の創出について考える必要があります。

まず、顧客満足とは何かを定義することからはじめ、ビジネスの観点で十分に評価可能な動くソフトウェアを素早く継続的に提供し、顧客の満足度がどのような状態なのか、明らかにします。

行動規範

- 顧客は、QCDの達成ではなく、ビジネスゴールであることを、まっさきに開発者へ伝えましょう。
- 顧客と開発者は、プロジェクトのゴールについて初期の時点で徹底的に話し合いましょう。
- 顧客と開発者は、プロジェクトのゴールに向かっているかどうかを常に確認し、方向がずれていたら素早く修正しましょう。
- 開発者は、ソフトウェアで実現すべき要件を顧客と一緒に考えましょう。顧客が必ずしも正解を持っているわけではありません。
- 開発者は、顧客が価値を確認しやすい単位に分割しましょう。分割した単位を、素早く継続的に、顧客に提供し続けましょう。

原則02 要求の本質を見抜き、変更を前向きに

オリジナル

「要求の変更はたとえ開発の後期であっても歓迎します。変化を味方につけることによって、お客様の競争力を引き上げます。」

“Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.”

基本的な考え方

改善に繋がる要求は新しい価値を見つけた証拠。勇気を持って受け入れることで、変化に強くなり、企業の競争力を高めることに貢献します。

従来は、定型業務の処理や記録を扱う基幹系システムなどのビジネスが多く、最初から作るものが明確なシステムが大半でした。しかし、昨今、人とモノの繋がりを重要とする業務改善や変革を目的としたビジネスは、最初から要求に明確な答えがないため、作るものが不明確な場合があります。このような特性が異なるビジネスのITシステム開発を従来と同じソフトウェア開発プロセスで実施しても、成功確率が上がりません。

そもそも顧客にとっての価値のないシステムを作っても意味がないので、たとえ開発の後期であっても要求の本質を見抜き、与えられたコスト・納期・スコープのバランスと、要求の優先順位から変更の受け入れを判断する必要があります。

行動規範

- 関係者は全員、変化を「新しい価値」の発見の機会ととらえ前向きに受け入れましょう。
- 関係者は全員、改善に繋がる要求を生み出すことに努めましょう。
- 開発者は、価値があるのであれば積極的に変更を受け入れましょう。
  - ただし顧客に言われるがままに変更を受け入れることはやめましょう。
- 開発チームは、間違いについてもオープンに議論しましょう。
  - 例えば数分で修正できる小さな間違いが、後までそのままにしておくことによりリリース日を数週間遅らせる原因になります。
- 開発チームは、よいフィードバックを生み出すようなリリースをすることに努めましょう。

原則03 成果物を2-3週間で、リリースし続ける

オリジナル

「動くソフトウェアを、2-3週間から2-3ヶ月というできるだけ短い時間間隔でリリースします。」

“Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.”

基本的な考え方

顧客が積極的にビジネスの舵取りができるよう、開発チームは2-3週間の短い時間間隔で、成果物を定期的にリリースし続けます。

リリースするというと、比較的長期間(例えば半年から数年など)の開発の後に顧客に納品することを思い浮かべるかもしれません。しかし、長い開発期間中に当初顧客が望んでいたゴールが変わり、顧客が本当に欲しかった成果を得られていない可能性もあります。

この原則の短い時間間隔とは、2-3週間から2-3ヶ月です。顧客のビジネスゴールが変わって要求が変更になることが想定される開発の場合は仮説検証型で進める必要があります。成果物を頻繁にリリースすることで、顧客からフィードバックを得ることができ、結果として顧客が本当に欲しいものを作ることができます。また、たとえ後戻りがあったとしても小さくすませることができます。

行動規範

- 関係者は、要求は不確実であり変化することが前提であることを共有しましょう。
- 開発者は、価値に結びつかない中間成果物ではなく、ビジネス観点で評価できるものを提供しましょう。
- 開発者ではなく顧客が、成果物のリリース間隔をビジネスの観点から決めましょう。
- 顧客は、市場からのフィードバックも得て開発者と共有しましょう。

原則04 全員で共通の目標に向かおう

オリジナル

「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。」

“Business people and developers must work together daily throughout the project.”

基本的な考え方

ビジネス側の人と開発チームを含めた関係者全員が共通目標に向かって一緒に働くことで、一連のリリースおよびフィードバックのサイクルが円滑になります。

不確実性の高いプロジェクトでは、状況が刻々と変化するため、共通の目標が見えないまま進めると認識の齟齬などが発生する可能性があります。

ビジネスの価値を最大化するために、作るソフトウェアの価値やプロジェクトの状況について、常に共通目標に向かってベクトルを合わせながら働きましょう。そのために、常にあるべき姿を関係者全員で共有し、必要に応じて改善し続ける必要があります。

行動規範

- 関係者全員が一緒の場所で働くことを理想としましょう。
  - 一緒の場所で働けない場合は、工夫して、方向性を共有しましょう。
- 関係者全員でビジネスを成功に導くという共通目標を、日々確認しましょう。
- 関係者全員で双方向のコミュニケーションが図れるようにしましょう。
  - ビジネス側の人は、要求の出しっぱなしはやめましょう。
  - 開発チームは、フィードバックをもらうために、頻繁にリリースをすることから始めましょう。
- 関係者全員が、共通認識を持つために最低限必要なドキュメントは書きましょう。

原則05 人の意欲は信頼から

オリジナル

「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。」

“Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.”

基本的な考え方

プロジェクトを成功させるため、開発チームには意欲のある前向きな人々を集めて、彼らが働きやすい環境を整えることが求められます。

個人が十分に高い能力を備えていたとしても、お互いの関係がぎくしゃくしているなど、働きにくい環境では十分に能力を発揮できません。もともと日本には「和」を重んじる文化、精神があり、経営やものづくりの局面にも活かされてきました。「対立」を避け、相互にリスペクトし信頼し合うことで、チームや組織内の風通しがよくなり、関係者全員が意欲的に活動できるようになるのです。

行動規範

- 関係者は、開発チームを信頼して仕事を任せ、気持ちよく働けるよう配慮しましょう。
  - 細かな作業指示は開発者の自律的な行動を阻害します。
- 開発者は、顧客の価値創造のために、自らが考え、自らの行動と判断に責任を持ちましょう。
- 開発者は、自分の作業だけでなく、開発チームとして何ができるかを考えましょう。
- 開発者は、積極的にデモを行い、ビジネス側の人からのフィードバックを受けましょう。
- 関係者は、開発者が協調し、集中して作業できる環境を提供しましょう。

原則06 顧客も開発チームも直接対話で

オリジナル

「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。」

“The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.”

基本的な考え方

お互いの本音をフェイス・トゥ・フェイスで汲み取り、直接対話することが大切です。

顧客が開発者に伝えたい情報や、開発者間で伝え合う情報の中には、仕様とは異なり、目的、希望、夢、心配ごとなど、正確に表現できないものもあります。そのような情報は、言葉では伝わりにくかったり、伝える際に齟齬や誤解を起こしやすかったりします。

関係者が全員同じ空間に集まり、対面で双方向に対話することで、お互いの本音を汲み取りやすくなります。もし齟齬が起きたとしても、対話を通してすぐに気づくことができます。

行動規範

- 顧客も開発者も、必要な人たちが集まり、双方向の議論をすることを心がけましょう。
  - 決まった仕様を伝えるだけのような一方通行はやめましょう。
- 顧客も開発者も、他の役割の人と話すことで自分の範囲を広げ、これまで気づいていなかったことにも気付けるようにしましょう。
  - 自分の役割を、狭く、固定しすぎないようにしましょう。
- 顧客も開発者も、何か気になることがあったら、直接会話をしましょう。
- 顧客も開発者も、メールだけの伝達は対話より非効率であることを共通認識としましょう。
- 顧客も開発者も、しぐさや表情から、言葉にならない本心（心配ごと、違和感）を感じ取りましょう。

原則07 進捗も品質も現物で

オリジナル

「動くソフトウェアこそが進捗の最も重要な尺度です。」

“Working software is the primary measure of progress.”

基本的な考え方

進捗状況を正確に把握するためには、動くソフトウェアを確認することが最も信用できるやり方です。 なお、この場合の進捗状況とは、求められている価値をどれだけ実現できているか、ということです。 また、この場合の動くソフトウェアは、計画通りに進んでいるかということではなく、市場にリリースできる程度の品質を満たしている必要があります。

設計書の完成度など、動くソフトウェア以外のもので測る進捗では、実際に動かしてみるまでは問題に気づけないことがあるため、どうしても表面上の進捗状況把握に留まってしまいます。

不確実性の高いプロジェクトでは、進捗状況を常に正確に把握し、適切な判断を行うことが重要です。

行動規範

- 開発者は、動くソフトウェアを早いタイミングで見せられるように、小さな単位でモノを作る計画を立てましょう。
- 顧客は、期待に合っているか判断するために評価できる受入基準を作りましょう。
- 開発者は、顧客の作った受入基準を満たすソフトウェアを作りましょう。
- 顧客も開発者も、動くソフトウェアに求められる必要な品質が担保されているべきと考えましょう。
- 開発者は、開発中の仕掛りを極力少なくしましょう。
- 開発者はパーセントで進捗を測るのではなく、実物に対するテストの合格/不合格で測りましょう。
- 顧客も開発者も、報告書に時間をかけることはせず、実物を触って進捗を測りましょう。

原則08 一定のペースでプロジェクトにリズムを

オリジナル

「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。」

“Agile process promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.”

基本的な考え方

無理のない一定の開発ペースを保つことは、ビジネスの成功につながります。

往々にして開発者は、ゴールを目指して過負荷なパフォーマンスを求められることがあります。無理をした状態では創意工夫をしたり、改善をしたりという意欲が起きません。また、無理をして一時的に生産性を上げたとしても、その無理が祟ってその後の生産性が落ちて結果的にトータルパフォーマンスは低下してしまいます。

開発チームの生産性を最大限に引き出し、価値あるものを提供し続けるためには、開発者だけでなく、スポンサー、顧客も心身ともにベストな状態を保ち、無理のない一定のペースを維持することが重要です。

行動規範

- 開発者は、タスクを小さく均一化し、一定期間内に優先順位の高いものからの開発を繰り返しましょう。
- 開発チームと関係者は、定期的に繰り返す「リズム」を大切にしましょう。
- 開発チームは、一定のペースを保つために、生産性を見える化し、適切に計画を見直しましょう。
  - 持続可能なペースは開発チームごとに異なります。
- 開発チーム内のコミュニケーションを密にし、開発者同士の考えを共有して助け合いましょう。
- 開発チームと関係者は、無理が生じている箇所（ボトルネック）を見つけ、助け合って改善しましょう。

原則09 よい技術、よい設計、よい品質の追求

オリジナル

「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。」

“Continuous attention to technical excellence and good design enhances agility.”

基本的な考え方

新しいビジネスを支えるソフトウェアは、状況に応じて、いつでも素早く変更できることが求められています。そのためには、優れた技術力に支えられた優れた設計になっている必要があります。よい品質には、高い保守性が欠かせません。

もし、設計やソースコードの可読性などについての技術的な問題が蓄積されると、ソフトウェアの保守性が低下し、素早く変更できない状態に陥ります。すると市場の変化への対応が遅くなり、ビジネスを成長させる際の大きな阻害要因となってしまいます。

よりよい技術を追い求め、効果的にその技術を活用することで、素早く、品質よく、変化に対応し続けることができます。技術は常に進化しています。チームは技術力を磨き、プロダクトへの適用技術と設計に対して、常に注意を払うことが重要です。

行動規範

- 開発者は、常に最新の技術を学び、効果的な技術はプロダクトへ適用しましょう。
- 開発者は、ソースコードの状態を目視やツールで常に診断し、常に変更して高い可読性を維持しましょう。
  - 可読性を高めると、多くの場合、コードの行数は減ります。
- 開発者は、自動化されたテストを作成して常に実行し、問題をすぐに発見しましょう。
- 関係者は全員、上記の行動に価値があることを認めましょう。

原則10 ムダ＝価値を生まない、を探してヤメる

オリジナル

「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。」

“Simplicity--the art of maximizing the amount of work not done--is essential.”

基本的な考え方

顧客が求める価値が何かを全員で考え、その価値に直結しない作業や機能をムダと定義し、そのムダを減らすためにたゆまぬ努力をしましょう。これは、アジャイル開発における本質の活動です。

ムダの中には、組織にとって現在必要なものも含まれるので、見逃しがちです。これまでの常識や慣習にとらわれず、新鮮な目で現状を直視しましょう。

日々の作業の中から、全員がムダを見つけ、削減することは、身軽になり機敏性を高めるだけでなく、コスト削減と生産性向上にも貢献します。

行動規範

- ムダについて真剣に話し合いましょう。
  - 例えば、ある作業を100倍して、その価値も100倍になっているか考えてみましょう。
- 顧客に価値をもたらさない会議や報告をなくしましょう。
  - 手始めに、進捗会議は短くしましょう。報告資料はシンプルにしましょう。
- 開発者は、顧客からの要求を鵜呑みにせず、「その要求は本当に必要ですか？」と聞きましょう。
  - 顧客が本当に必要か分からない機能を、先に作るのは「つくりすぎのムダ」です。

原則11 よいモノはよいチームから

オリジナル

「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。」

“The best architectures, requirements, and designs emerge from self-organizing teams.”

基本的な考え方

最良のアーキテクチャ・要求・設計は、最良のチーム(=自律型のチーム)から生み出されます。自律型のチームとは、開発者一人ひとりが自らの行動、作業に責任を持つとともに、開発者同士の連携により、チームとして成果を最大限に発揮することができるチームのことです。

チームが優秀な開発者の集まりだったとしても、チーム内に連携、協調がなければ、チームの能力は、個々の開発者の能力の和でしかありません。あるいは、個々の開発者の作業が重複していたり、相反する作業をしていたりすれば、チームとしての成果に悪影響を与える可能性もあります。

一方自律型のチームでは、開発者一人ひとりが常に改善の意識を持ち、お互いの能力を高めながら共通の目的や目標に対しての取り組みが継続されます。そこには相乗効果が生まれ、チームとして開発者個々の能力以上の成果を発揮することが可能になります。その結果、自律型のチームから、ベストな成果が生み出されることになるのです。

行動規範

- 開発チームは、プロジェクトの達成すべき目的や目標とそれを支える規律・ルールを共有しましょう。
- 開発チームは、自分たちで規律・ルールを決めましょう。
- 開発者は、自ら率先して作業の改善や課題の解決に取り組みましょう。
- 開発者は、自分の不得意な作業、領域についてもチャレンジして取り組みましょう。
- 開発者は、お互いの作業状況をオープンにし、共有しましょう。
- 開発者は、困っているメンバーがいたら、助け合いましょう。

原則12 自分たちのやり方を毎週、調整する

オリジナル 

「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。」

“At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.”

基本的な考え方 

常にチームが成長し続けるためには、定期的に(長くても2～3週間、オススメは毎週)振り返り、自分たちのやり方を調整し続けることが大事です。

“ふりかえり”というとプロジェクトの最後の反省会を思い浮かべるかもしれません。次のプロジェクトでは、周りの環境もチームも違うので“ふりかえり”の結果を活かすことは難しいでしょう。

この原則で言う“ふりかえり”とは、プロジェクトの進行中に定期的に、短い時間で行うことです。“ふりかえり”を繰り返すことで、よいやり方はどんどんよくなり、もし、失敗したとしても(うまくいかなかったとしても)影響を最小限に留めることができます。定期的な“ふりかえり”の場があることで、チームメンバーは、プロジェクトにとって有用ではあるが言いにくいことを言えるようになります。

行動規範 

- 開発チームだけでなく、ビジネス側も交えてチーム全体で振り返りましょう。
- “ふりかえり” にかける時間は、毎週の場合、30分～1時間程度を目安にしましょう。
  - だらだら続けるのはやめましょう。
- “ふりかえり” のルールはチーム全体で決めましょう。
- チームで問題に立ち向かいましょう。
  - 個人の責任を追及してはいけません。
- 挙がった課題をすべて解決しなくても気にしないようにしましょう。
  - 失敗は許容しましょう。すぐ改善すればよいのです。

アジャイル宣言の背後にある原則　～一覧～

IPA

- 原則01 顧客の満足を求め続ける
- 原則02 要求の本質を見抜き、変更を前向きに
- 原則03 成果物を2-3週間で、リリースし続ける
- 原則04 全員で共通の目標に向かおう
- 原則05 人の意欲は信頼から
- 原則06 顧客も開発チームも直接対話で
- 原則07 進捗も品質も現物で
- 原則08 一定のペースでプロジェクトにリズムを
- 原則09 よい技術、よい設計、よい品質の追求
- 原則10 ムダ＝価値を生まない、を探してヤメる
- 原則11 よいモノはよいチームから
- 原則12 自分たちのやり方を毎週、調整する

おわりに

変化とスピードが要求されるこれからのビジネスに対応するためには、変化とスピードがプロセスにあらかじめ組み込まれているアジャイル開発にシフトしていく必要があります。これまで、日本では諸外国に比べてウォーターフォール開発プロセスをうまく工夫しながら適応してきました。そのためアジャイル開発へのシフトがうまく進んでおらず、本書初版発行時点においては、欧米はおろかアジア（中国、インド、ベトナム等）の中でも取り残されつつあります。

アジャイル開発へのシフトの阻害要因の一つとして、アジャイル開発の原点である「アジャイルソフトウェア開発宣言」および「アジャイル宣言の背後にある原則」が日本においては、十分に伝わっていないことがあります。また、伝わっていたとしても正しく伝わっていないケースが少なからず見受けられます。

そこで、本書では「アジャイルソフトウェア開発宣言」および「アジャイル宣言の背後にある原則」を日本の企業風土、組織文化を考慮し、分かりやすく解釈することを試みました。

価値を追い求めるために常に考え続け、体現し続けていく姿勢こそが、アジャイル開発に一番大事なマインドセットです。その姿勢を維持し続けるためには、自分たちに合った方法を常に試していく必要があります。 当然その途上では失敗も経験するでしょう。しかし、その失敗の中からより最善の策を見出し、その時に一番よい方法を見つけ出していきましょう。失敗を効率よく成功の糧としていくのです。よいと思ったらまずやってみて、うまくいかなければ短いサイクルで「改善」を繰り返していけばよいという考え方です。 そう、アジャイル開発とは日本の得意な「改善」をたくさん試せる手法なのです。

本書がその具体的な行動に踏み出すための一助になれば幸いです。