

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

2020年2月

IPA 独立行政法人  
情報処理推進機構

# はじめに

## • 本書のターゲット

本書は、アジャイルソフトウェア開発にこれから取り組もうとする企業をターゲットに作成しています。そのような企業の多くは、未だに受注者・発注者の関係に悩んだり、アジャイルソフトウェア開発を誤解して解釈しがちです。そうした企業の皆さんに、本書が少しでも悩みの解決に役に立てれば幸いです。

## • 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つ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冒頭に、「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が、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、それぞれの主義や手法についての議論を行い、2001年に公開されました。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」が書かれていました。

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、「アジャイルソフトウェア開発」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」について、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ソフトウェア開発を理解していただくことを、目的としています。

右図は、本書で示している人物の関係図を表しています。



# 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

© 2001, 上記の著者たち

この宣言は、この注意書きも含めた形で全文を含めることを条件に自由にコピーしてよい。

<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、アジャイルソフトウェア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。よってアジャイルソフトウェア開発でも“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」  
顧客の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、今までの行動をふりかえり、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。

# 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

ビジネス側の人と開発者にとって、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、顧客に（QCDの達成ではなく）ビジネスゴールの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います。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。ビジネス側の人と開発者は、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。したがってビジネス側の人と開発者は、常に顧客と同じ目線で、顧客の満足度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、顧客の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

- ビジネス側の人と開発者は、ビジネスゴールについて初期の時点で徹底的に話し合しましょう。
- ビジネス側の人と開発者は、QCDの達成ではなく、ビジネスゴールの達成を目指しましょう。
- ビジネス側の人と開発者は、ビジネスゴールに向かっていく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。
- ビジネス側の人と開発者は、ソフトウェアで実現すべき要件を顧客の立場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。
- ビジネス側の人と開発者は、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。



## 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です。勇気を持って変更を受け入れることで、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ことが多く、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、成功確率は上がりません。

そもそも、顧客にとって価値のないシステムを作っても、意味はありません。そこ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

- ビジネス側の人と開発者は、変化を「新しい価値」の発見の機会ととらえましょう。
- ビジネス側の人と開発者は、改善に繋がる要求を生み出すことに努めましょう。
- ビジネス側の人と開発者は、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。
- ビジネス側の人と開発者は、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。
- ビジネス側の人と開発者は、より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。

## 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

ビジネス側の人々が積極的にビジネスの舵取りができるよう、ビジネス側の人と開発者は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なくなっている可能性もあります。

この原則03という短い時間間隔とは、2-3週間から2-3ヶ月です。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

- ビジネス側の人と開発者は、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。
- ビジネス側の人と開発者は、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。
- ビジネス側の人と開発者は、市場からのフィードバックを共有しましょう。

# 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。」

“Businesspeople 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者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、常にあるべき姿をビジネス側の人と開発者が共有し、必要に応じて改善し続ける必要があります。

## 行動規範

- ビジネス側の人と開発者は、全員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。
- ビジネス側の人と開発者は、一緒に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。
- ビジネス側の人と開発者は、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。
  - ビジネス側の方は、要求の出っぱなしはやめましょう。
  - 開発者は、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。
- ビジネス側の人と開発者は、全員が共通認識を持つために、必要最低限のドキュメントを書きましょう。

# 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、開発者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、ビジネス側の人と開発者、そして組織内の風通しがよくなり、ビジネス側の人と開発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

- ビジネス側の人、開発者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。
- ビジネス側の人と開発者は、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。
- ビジネス側の人と開発者は、自分の作業だけでなく、顧客の価値創造のために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、顧客からのフィードバックを受けましょう。
- ビジネス側の人と開発者は、お互いが協調し、集中して作業できる環境を作りましょう。

# コミュニケーションは直接対話で

## 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事です。」

“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。

ビジネス側の人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。

ビジネス側の人と開発者が、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

- ビジネス側の人と開発者は、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一方通行はやめましょう。
- ビジネス側の人と開発者は、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。
- ビジネス側の人と開発者は、何か気になることがあったら、直接会話をしましょう。
- ビジネス側の人と開発者は、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。
- ビジネス側の人と開発者は、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。

## 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も測る進捗では、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。
- ビジネス側の人と開発者は、顧客の期待に合っているか判断します。そのための受入基準をビジネス側の人と開発者で作りましょう。
- 開発者は、受入基準を満たすソフトウェアを作りましょう。
- ビジネス側の人と開発者は、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、実物に対する受入基準の合格/不合格で測りましょう。
- ビジネス側の人と開発者は、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。

# 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、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。

開発者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、ビジネス側の人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

- ビジネス側の人材は、機能を小さくし、短い期間で価値を提供できるようにしましょう。
- 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開発することを、繰り返しましょう。
- ビジネス側の人材と開発者は、定期的に繰り返す「リズム」を大切にしましょう。
- ビジネス側の人材と開発者は、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、開発者ごとに異なります。
- ビジネス側の人材と開発者は、コミュニケーションを密にし、お互いの考えを共有して助け合いましょう。
- ビジネス側の人材と開発者は、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。

## 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、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。開発者は技術力を磨き、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。
- ビジネス側の人と開発者は、全員上記の行動に価値があることを認めましょう。



# 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

ビジネス側の人と開発者は、顧客が求める価値が何かを考え、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、アジャイルソフトウェア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

- ビジネス側の人と開発者は、顧客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。
  - 顧客にとって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。
- ムダについて、真剣に話し合しましょう。
  - 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。

# 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、メンバー一人ひとりが自らの行動、作業に責任を持つとともに、お互いの連携により、チームとしての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。

優秀なメンバーが集まったチームだとしても、お互いに連携や協調がなければ、チームの能力は、個々のメンバーの能力の和でしかありません。あるいは、個々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ます。

一方自律型のチームでは、メンバー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する取り組みが継続されます。そこには相乗効果が生まれ、チームとしてメンバー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

- 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。
- チームは、自分たちで規律・ルールを決めましょう。
- メンバーは、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。
- メンバーは、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。
- メンバーは、お互いの作業状況をオープンにし、共有しましょう。
- メンバーは、困っているメンバーがいたら、助け合いましょう。

# 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

チームが成長し続けるためには、定期的に(長くて2~3週間、オススメは毎週)ふりかえり、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。しかしたいいていの場合、次のプロジェクトは周りの環境もチームも異なることが多く、“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

- チームとして、主体的にふりかえりましょう。
- “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。
  - だらだら続けることはやめましょう。
- “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。
  - 個人の責任を追究してはいけません。
- 必要に応じて周りに協力をあおいで解決しましょう。
- 拳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～

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

コミュニケーションは直接対話で

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ソフトウェア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、アジャイルソフトウェア開発へのシフトがうまく進んでおらず、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。

アジャイルソフトウェア開発へのシフトの阻害要因の一つとして、アジャイルソフトウェア開発の原点である「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げられます。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、アジャイルソフトウェア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。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、アジャイルソフトウェア開発とは、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。