アジャイル開発実践ガイドブック

2021 年（令和３年）3 月 30 日

内閣官房情報通信技術（IT）総合戦略室

〔標準ガイドライン群ＩＤ〕
1022

〔キーワード〕
アジャイル開発の意義、アジャイル開発の向き・不向き、スクラム、利用者目線のシステム開発、協働、対話コミュニケーション、フィードバック、適応、MVP、YAGNI、XP（eXtreme Programming)、準委任契約

〔概要〕
政府情報システムの利用は府省職員だけではなく、自治体及び国民までその範囲とするため、昨今の社会環境の変化や多様化に基づき、様々なニーズに迅速に応えていく必要性が高まっています。この観点から、政府情報システム開発においても、従来の開発スタイルとは別にアジャイル開発という選択肢を設ける必要があります。
本ガイドブックでは、アジャイル開発を理解するためにまず必要とされる基本的な知識を提供します。

改定履歴

改定年月日	改定箇所	改定内容
2021年5月10日	4.1 全般的な事項	・各種ドキュメントについての記載を一部修正
	5.2 プロジェクトマネジメントや運営	・参考情報として「プロジェクトマネジメント知識体系ガイド」第6版を追記
2021年4月7日	5.3 スクラム	・参考文献の書名誤記を修正
2021年3月30日	－	・初版決定

目次

目次 i
１ はじめに 1
１．１ 背景と目的 1
１．２ 適用対象 2
１．３ 位置付け 2
１．４ 用語と概要 2
２ 政府情報システムにおけるアジャイル開発 5
２．１ アジャイル開発とは 5
２．２ アジャイル開発の本質 6
1）アジャイル開発の９つの意義 6
2） ９つの意義を十分に発揮するための前提 9
２．３ アジャイル開発の適用方針 12
1）アジャイル開発に向いている・不向きな領域 12
2）開発方針の検討 12
２．４ 調達時に留意すべきこと 13
1）経験者の参画 13
2）発注者の姿勢 14
3）開発範囲に MVP（Minimum Viable Product）の範囲を用意する 14
4）契約方式を検討する 15
３ アジャイル開発の運営 16
３．１ 運営の概要 16
1）役割 16
2）実施ミーティング（スクラムイベント） 19
3）作成物 21
３．２ 運営の体制 22
３．３ 運営の準備 23
1）アジャイル開発に関する知識の獲得 23
2）事業者との協働 23
3）当該プロジェクトでの開発方針を定める 24
4）全体計画についての認識を合わせる 24
5）チームでワーキング・アグリーメントを決めていく 25
6）チーム内コミュニケーションの一元化 25
３．４ 運営の適応 26

4 各種留意事項 27
4．1 全般的な事項 27
1）各種ドキュメントについて 27
2）要件定義書について 27
3）品質管理について 28
4．2 第三者チェックの有効活用 29
4．3 継続的なチーム体制の確立 29
5 参考情報一覧 31
5．1 アジャイル開発、全般 31
5．2 プロジェクトマネジメントや運営 31
5．3 スクラム 32
5．4 スプリント・レトロスペクティブ（ふりかえり） 32
5．5 チーム開発 32
5．6 プロダクト開発 33

１　はじめに

１．１　背景と目的

本ガイドブックは、政府情報システム開発におけるアジャイル開発の適用を支援するために用意されたものです。

従来のような「システム開発の全体をフェーズで分割定義し、各フェーズが完遂されなければ次のフェーズへと移行できない」という開発スタイルでは、実際の開発着手まで相応の時間を要することになります。こうした開発スタイルでは、原則として、すべての要求を開発着手前に定義し終えることが求められます。

しかし、現実的には開発期間を潤沢に確保することは難しく、要求とその仕様の詳細をすべて記述するのに費やせるほどの時間をあてられないことも少なくありません。また、実際の利用における望ましい挙動を実現するためには、作成されたアウトプットを使ってみて、そのフィードバックを元に洗練させていくアプローチの方が効率的かつ効果的であるケースが往々にしてあります。

特に、政府情報システムの利用は府省職員だけではなく、自治体や国民をもその範囲とするため、昨今の社会環境の変化や多様化に基づく様々なニーズに迅速に応えていく必要性が高まっています。こうした状況下では、システム開発にも変更に対して柔軟に適応することが求められます。

こうした観点から政府情報システム開発においても、従来の開発スタイルとは別にアジャイル開発という選択肢を設ける必要があります。本ガイドブックでは、アジャイル開発を理解するためにまず必要とされる最小限の知識を提供します。アジャイル開発が、少しずつ反復的に情報システムをより良くしていくアプローチであるように、本ガイドブックも本内容をスタートラインにおいて、政府情報システム開発での試行及び実績を踏まえて、継続的に改訂していくことを想定しています。

なお、アジャイル開発の適用にあたっては、本ガイドブックの記述どおり進めれば上手くいくというわけではないことに注意してください。アジャイル開発は、協働という価値観の下に成り立っており、システム開発の関係者がお互いに協力し合う姿勢がその前提となっています。ただアジャイル開発の方法を知っているだけでは有効には機能しません。

協働してアジャイル開発に取り組むためには、取組にあたる関係者全員が、それぞれの立ち位置に基づいてアジャイル開発への理解を深める必要があります。

- 府省職員の理解

職員はアジャイル開発の役割の１つである「プロダクトオーナー」としての振る舞いを理解する必要があります。システム開発全般にわたって、主体的に関与しなければなりません。

・ 事業者の理解

職員及びその支援者だけがアジャイル開発に対応できれば良いわけではありません。実際にシステム開発にあたる事業者の協力が不可欠であり、事業者側も従来の開発スタイルに拘泥することなく、アジャイル開発への適応が求められます。

以上のとおり、アジャイル開発への適応はどちらか一方が担えば良いというわけではありません。関係者全員に、一定の理解と実際の振る舞いが求められるところに、アジャイル開発への適応の難しさがあります。本ガイドブックでは、システム開発の関係者に向けて、アジャイル開発とは何か、その期待される利点と、留意点を説明します。

１．２　適用対象

本ガイドブックは政府情報システムの整備における開発手法としてアジャイル開発を用いるプロジェクトを適用の対象とします。なお、本ガイドブックは、アジャイル開発への理解を深めるための参考文書であり、遵守を求めるものではありません。

１．３　位置付け

本ガイドブックは、標準ガイドライン群の一つとして位置付けられます。

１．４　用語と概要

本ガイドブックにおいて使用する用語は、表　１　用語及び本ガイドブックに別段の定めがある場合を除き、標準ガイドライン群用語集の例によるものとします。その他専門的な用語については、民間の用語定義を参照するものとします。

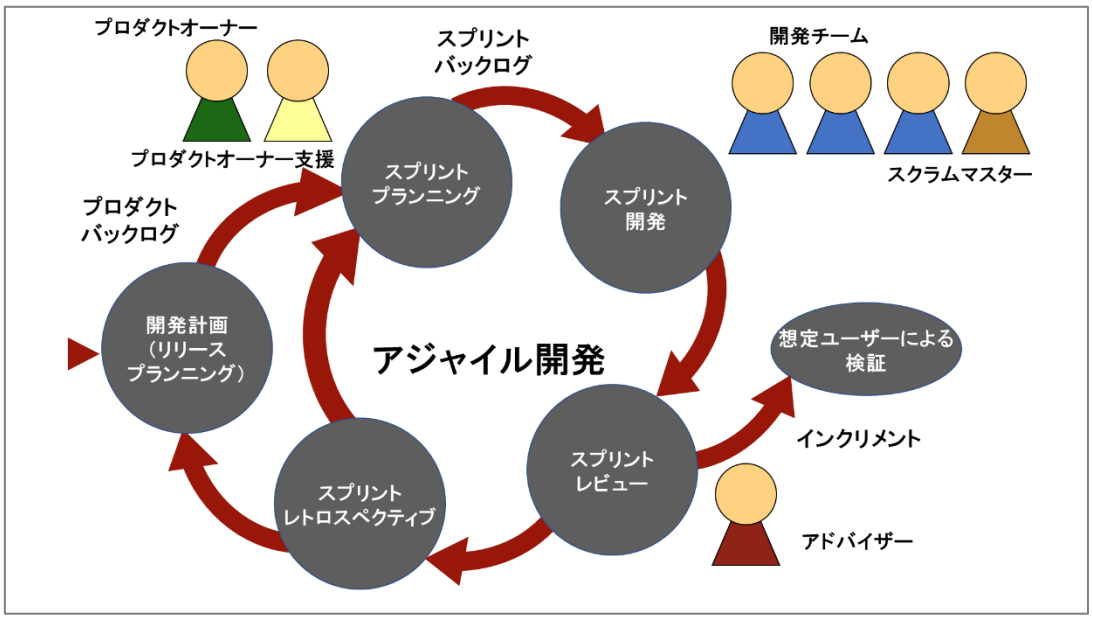
また、図　１　アジャイル開発の概要に本ガイドブックで前提とするアジャイル開発の概要を示しているため、用語とともに内容を確認してください。詳細については第３章「アジャイル開発の運営」において説明します。

表 1　用語の定義

用語	意味
スクラム	アジャイル開発の１つとされ、国内外を問わず、適用されていることが多い開発手法。本ガイドブックは、スクラムを基本としてアジャイル開発を説明する。
スプリント	開発に必要な行為やイベントが一通り揃った、スクラムにおける開発期間を指す。スクラムでは、スプリントを単位として反復して開発する。
プロダクトオーナー	開発する機能の仕様策定に関する議論を主導し、開発機能の優先順位や実現方法等に対する意思決定を主体的に行う役割を指す。開発する情報システムの価値を最大化することに責任を負う。
プロダクトオーナー支援	アジャイル開発におけるプロダクトオーナーの振る舞い並びにシステム開発及びプロジェクト運営に対する職員の知見を補完する役割を指す。
スクラムマスター	チームが機能するように、アジャイル開発が前提としている価値観、考え方、実践にあたって必要な振る舞い、方法についてのレクチャーやコーチングなどを行い、支援する役割を指す。
開発チーム	本ガイドブックでは、政府情報システムの開発の現状を踏まえ、開発を行う事業者を想定しているが、発注者（職員）が開発チームとなる可能性もある。
アドバイザー	プロダクトオーナー、スクラムマスター、開発チームを含んだチーム以外の関係者をまとめて指す。システム開発における意思決定には関与せず、チームに対する助言、知見の提供、フィードバック等を行う。
リリースプランニング	プロジェクトの開発範囲の規模から、スプリントがいくつ必要かを見立て、プロジェクト全体の計画を立てるミーティングを指す。本ミーティングはスクラムの規定にはない。
スプリントプランニング	スプリントの開始時に実施する。予めリスト化したプロジェクト全体の開発対象機能（プロダクトバックログ）から、当該スプリントにおける開発対象機能とするものを選択して優先順位で並べ替え（スプリントバックログ）、開発方法などの計画を立てるミーティングを指す。

用語	意味
デイリースクラム	開発チームの状況について共通理解を持ち、スプリントのゴールが達成できそうか確認するために毎日１５分程度の短時間で行うミーティングを指す。
スプリントレビュー	スプリントの成果の確認を行い、ゴールがどの程度達成できたかを判断する。このフィードバックをもとに今後実施するべき事項を検討するミーティングを指す。
スプリント・レトロスペクティブ（ふりかえり）	スプリントの活動を省みて、次のスプリントの活動が効率的、効果的となるよう、継続すべき工夫、取り除くべき問題、そのための施策などを検討するミーティングを指す。
プロダクトバックログ	システム開発に必要となる要求のリストを指す。リストの並び順で要求の優先順位を示す。
スプリントバックログ	当該スプリントで開発対象となった要求及びその要求を実現するために必要な作業を含んだリストを指す。
開発成果物	スプリントによって開発された、動くシステムそのものを指す。スクラムではインクリメントと呼ぶ。

図　１　アジャイル開発の概要



※　本ガイドブックではスクラムをベースとする

２　政府情報システムにおけるアジャイル開発

２．１　アジャイル開発とは

アジャイルとは、元来は特定の開発プロセスを指すものではなく、より良い開発のあり方を追求する態度のことを指しています。実際には、アジャイル開発の１つとされる「スクラム」が国内外を問わず、適用されていることが多いです。本ガイドブックも、スクラムを基本として説明します。

アジャイル開発を方法として端的に表すと、インクリメンタルかつイテレーティブな開発であると表現できます。インクリメンタルとは漸次的に（少しずつ）作り進める様を指し、イテレーティブとは反復的に開発行為を繰り返すことを言います。つまり、少しずつ反復的に作り進める開発です。

こうした開発スタイルによって、作成したアウトプットに基づき、情報システムの挙動がどうあるべきかを検討、判断し、その次に取り掛かる開発の行為を最適化します。なお、反復する期間は、１か月以下（１、２週間、または４週間）を基準として検討することが多く、それ以上の期間にするとウォーターフォール型の開発に近くなり、アジャイルの利点も薄れて行くこととなります。

スクラムでは、短期間のスプリントの中でスプリントプランニングやスプリントレビューといったイベントのタイミングを予め決めて明確にし、それを反復継続していく考え方を大切にします。いつ何を行うのか、それぞれにかける時間はどれくらいかをあらかじめ決定し、関係者がそれを守るスタイルです。これによって、時間調整等にかかるコストを削減するとともに、日常の作業にリズム感が生み出され、円滑に進められるようになります。

また、前述したとおり、アジャイル開発は従来の開発スタイルとは異なり、すべての要求、仕様を言語化し、事前のドキュメントとして整備することなく開発を行うこともできます。ドキュメントで定義しなくとも、短期間のスプリントで得られるアウトプット（インクリメント）が、動くシステムそのものとなり得るためです。ドキュメントの作成にかける手間を最小限にとどめ、情報システムそのもので動作確認を行うことで、要求の確認から設計、開発、テストまで、情報システムの機能追加を短い期間で行うことができます。

開発にあたって不足する情報については、関係者間での対話コミュニケーションで補います。そのため、アジャイル開発においてはより円滑なコミュニケーション、それを実現できる関係性を重視します。

なお、アジャイル開発の歴史的な成り立ちと、アジャイル開発で確認されている価値と原則については第５章「参考情報一覧」を参照してください。

コラム：「混ぜるな危険」

従来の開発スタイル（ウォーターフォール型）に慣れ親しんだ人が、アジャイル開発では逆に足を引っ張ってしまうことがあります。

例えば、開発工程での進捗を確認する方法について、従来の開発スタイルであれば WBS に基づいてクリティカルパスを把握し、週次の報告会でイナズマ線や EVM※に基づいた実績を確認し、そのために様々な資料を事業者が作成することになります。一方で、アジャイル開発ではスプリントのプランニング、開発、レビューというサイクルの中で「動くシステム」を確認しながらプロダクトオーナーを含めた関係者が進捗状況を共有することが中心であり、中間成果物としての報告書等に重きを置きません。

このようなカルチャーの違いを理解せずに、アジャイル開発であるのに進捗を心配して報告書提出等の監視を強めようとすると、アジャイル開発の流れを阻害してしまうことにもつながりかねません。従来型の開発スタイルにもアジャイル開発にもそれぞれメリットとデメリットがありますが、両者の異なる文化を無意識に「混ぜて」しまうことには注意を払いたいところです。

※ EVM：Earned Value Management の略。WBS により詳細化した各作業項目に出来高計画値（PV：Planned Value）を設定し、プロジェクトの進捗を出来高実績値(EV：Earned Value)として定量化して管理すること。詳細は標準ガイドライン実践ガイドブック第３編第７章 Step.３－２－Ｆ「EVM を用いた進捗管理手法を理解する」を参照のこと。

２.２　アジャイル開発の本質

1)　アジャイル開発の９つの意義

アジャイル開発は、具体的にどのような意義を持つのでしょうか。ここでは以下の９つを挙げます。

ア　フィードバックに基づく開発で、目的に適したシステムに近づけていく

漸次的な開発を反復的に行う意図は、早く形作ることでいち早く利用者(もしくはその代弁者たるプロダクトオーナー)からフィードバックを得ることです。

ここでいうフィードバックとは、実際に情報システムを使って得た操作感、理解を情報システムの目的、利用用途に照らし合わせた際の差分です。情報

システムの目的を果たすために必要な機能性、利用用途に適した操作性を発見することを狙います。

コラム：モニタリング改善におけるアジャイル開発の有用性

ある省庁では、給付を行う情報システムの構築にアジャイル開発を採用しました。リリース時点では、すべての機能を作り込むのではなく、直近で必要ない一部の機能はリリース後に追加するロードマップとする一方で、システム構成は将来的な改修を想定したつくりとして、実際の改修は容易に行えるようにしました。

また、この情報システムでは、給付によって得られるアウトプットやアウトカム、給付事務の効率化、受給者の利便性向上等をモニタリング指標としています。これらを適切にモニタリングしていくためには、データを様々な観点から収集・分析することが必要であるため、アーキテクチャをマイクロサービス化し、さまざまな断面でデータを取得できる仕組みを構築しました。

別の省庁の情報システムでは、職員自身が BI ツールを使いこなしてデータ分析を行えるようになっており、データの抽出などを運用事業者に依頼する必要がありません。この手法によって、社会環境の変化を捉えたデータを、法律改正の根拠などに活用できるようになっています。

従来の情報システムでは、運用事業者に依頼しなければデータ抽出も困難でしたが、近年はアジャイル開発や BI ツールなどの技術を活用することで、実際のデータを見ながら試行錯誤を経て分析方法を検討できるようになりました。このような新しい技術を駆使して、社会環境の変化を迅速に捉えつつ、証拠に基づく政策立案（ＥＢＰＭ）を推進していくことが重要です。

イ　形にすることで、関係者の認識を早期に揃えられる

情報システムを形作り、早期に確認、試すことで、作るべきものへの関係者の認識を引き出すことができます。ドキュメントや会話によるコミュニケーションで理解できることも多分にありますが、実際に使ってみることで得られる認識もあります。関係者それぞれの認識を表出させることで、その間でのズレを解消する機会を生み出すことができます。

ウ　システム、プロセス、チームに関する問題に早く気付ける

最短１週間から、長くとも１か月以下という短期間でシステムをアウトプットしていくことによって、作るべきものに対する認識相違や誤謬、開発上

の準備不足、開発フローの不備、開発チーム内でのコミュニケーション齟齬や不足等に気付くことを早めることができます。

エ　チームの学習効果が高い

アジャイル開発では、フェーズという概念を置いてフェーズごとにチームを切り替えるということを行いません。単一のチームで、何を作るべきかを探索し、開発し、試行し、さらに調整するということを繰り返し続けていきます。そのため、フェーズ間での知識の受け渡しという行為が発生せず、すべての行為の経験とナレッジとがチームに蓄積されていくことになります。

また、技術領域によっても担当者を分けず、例えば、１人のエンジニアがインフラ（サーバー、ネットワーク、ミドルウェア等を含むシステム基盤）構築からアプリケーション開発、運用まで幅広く担当したりサポートしたりする体制を組むこともしばしばあります。この点では、インフラ構築を専任のインフラ・エンジニアでなくアプリケーション・エンジニアが行うことも多いクラウドとの相性が良いと言えます。こうしたフォーメーションが組めるようになると、チーム内での役割を特定の個人に固定する必要がなくなり、開発が効率よく進むようになります。

オ　早く開発を始められる

すべての要求を仕様化するまで開発を始められない従来の開発と比較すると、アジャイル開発は１か月以下の開発期間（この期間のことを特にスクラムではスプリントと呼びます）ごとに何を作るか選択しながら進めるため、少なくとも最初の数スプリント分の要求が仕様化されていれば開発を始めることができます。

カ　システムの機能同士の結合リスクを早期に解消できる

機能一つ一つを単独で作り、結合テストフェーズで機能間の整合性を検証する方法だと、機能間の認識齟齬について検知できるのがかなり後になってしまうリスクがあります。アジャイル開発では、基本的にスプリントごとに開発した機能を結合し、テスト環境や本番環境に展開し、動く状態にします。つまり、結合時の問題について早期に検知できる可能性が高くなります。

キ　利用開始までの期間を短くできる

すべてのフェーズが終了しきらないと利用が開始できない開発スタイルに比べると、アジャイル開発の場合は、すべての機能を開発しなくても利用が試せる利点があります。一部の要件しか満たさない情報システムであっても

当該システムの利用が緊急に必要な場合や、一部機能であっても利用開始に十分と判断される場合には、その時点で当該システムを利用開始することができます。すべての機能を備えるために長期間開発を続けてからリリースする場合と比べ、開発期間中に社会情勢や環境の変化によって要件が変わったり、情報システムがリリース前に陳腐化してしまったりといったリスクを避けることができます。

なお、実際に本番運用を始めるレベルとしての利用開始なのか、あくまで試行レベルでの利用開始かによっても、そのために必要な準備作業が異なってきます。プロジェクトごとに利用開始の位置づけやスケジュールを決める必要があります。

ク　開発のリズムが整えられる

スプリントの中では、開発に必要な行為やイベントをひと揃いで行います。あるスプリントではプランニングを行うが、あるスプリントではプランニングを行わない、ということはありません。

そのため、スプリントを繰り返していくと進め方に安定性が生まれます。このような開発のリズムが定着することで、開発の効率も高めることができます。

ケ　協働を育み、チームの機能性を高める

スプリントごとに成果を出すために、進め方やコミュニケーションのあり方についてより良くしていく機運を高めやすいという利点があります。協働によって成果が上がるというサイクルを作ることで、よりチームの機能性を高めていくことが期待できます。

2)　９つの意義を十分に発揮するための前提

以上の９つの意義を十分に発揮するためには、以下の前提をチーム及び関係者間で確認する必要があります。

ア　常にカイゼンを指向すること

アジャイル開発とは、スプリント単位での開発に必要な「準備」、「実施」及び「適応」の繰り返しです。適応とは、実施した結果から情報システムやプロセス自体をより良くする働きかけであり、具体的には「ふりかえり（スプリント・レトロスペクティブ）」を定期的に行うことです。こうした適応行為がなければ、ただ定められたタスクを繰り返し実施するだけになってしまいます。

イ　対話コミュニケーションの重視

短い期間でアウトプットを行う開発では、ドキュメントに過度に依存したコミュニケーションではなく、対話による共通認識作りが重要となります。そのため、対話がしやすいチーム体制（過度な階層構造を避ける）の構築及び定期的なコミュニケーションの場の設定と、その遵守がチーム全員に求められることになります。

ただし、それは必ずしも同じ会議室に一同に会するということではありません。コミュニケーションツールを用いて、オンラインで打ち合わせや短時間のコミュニケーションを行う方法も、積極的に検討・導入し、コミュニケーションしやすい環境を作ります。

また、ドキュメントの共同作成・利用も重要です。オンラインでドキュメントを共有して同時編集することができるサービスを用いると、オンラインの打ち合わせの最中にもリアルタイムで同じドキュメントを参照し、その場で決定・変更した内容を反映して共通認識とすることができます。打ち合わせ後に議事録と変更後のドキュメントをメールで送るといった旧来の方式と比べて円滑に作業を進められることが期待できます。

ウ　情報システムの変更容易性を確保し続ける

スプリントを繰り返す中で、開発した機能がより良い内容となるよう変更を加えていくことになります。そのため、プロセスだけではなく、そもそも変更が可能なシステム設計が前提となります。どのようにして変更容易性を確保するのか、設計上の不断の注意が必要です。

変更容易性を確保するには、構成を柔軟に変更できるクラウドやコンテナ技術、CI/CD や DevOps の考え方が有効です。システム間を REST API によって疎結合とする方式や、データ構成を柔軟に変更できる非構造型の DB（No SQL）の活用、クラウドデザインパターンと呼ばれるシステム構成を利用することも選択肢の一つです。なお、変更を容易にするためのマイクロサービスアーキテクチャは、その利点とともにデメリットや副作用もありますので、導入する場合にはレベル感（サービスを分割するだけなのか、サービスメッシュまで導入するのか等）を含めて慎重な見極めが必要です。

エ　利用者目線で開発を進める

フィードバックを反映できる開発手法を採用し、それができるチームを作り上げたとしても、利用者目線の適切なフィードバックができなければ意味がありません。

このような利用者目線を踏まえたシステム開発を行うための心構えと視点として「サービス設計１２箇条」があり、特にアジャイル開発と親和性が高い以下の７箇条を念頭において進めることが重要です。実践ガイドブック「第３編第４章　サービス・業務企画」で詳しく説明しているのでこちらも参照してください。

- 第１条　利用者のニーズから出発する
- 第４条　全ての関係者に気を配る
- 第７条　利用者の日常体験に溶け込む
- 第９条　オープンにサービスを作る
- 第１０条　何度も繰り返す
- 第１１条　一遍にやらず、一貫してやる
- 第１２条　システムではなくサービスを作る

コラム：サービス設計１２箇条を意識したWebサイトの構築

ある Web サイトでは、開発手法としてアジャイル開発を採用しました。加えて、利用者目線で構築を行わなければ利用されなくなるという危機感の下に、利用者目線に沿ったユーザインタフェースの実現、継続して使われるサイトの実現を目指して、当初からサービス設計１２箇条、特に以下の３箇条を意識して検討を進めました。

・第４条　全ての関係者に気を配る

アドバイザーには、他府省の関係者、民間の事業者や団体を加え、議事の結果をインターネットで公開するなど、検討過程をオープンにしています。

・第８条　自分で作りすぎない

情報を活用するアプリケーションは、アドバイザーとして集まった民間事業者が任意に構築することとし、当該 Web サイトではデータを利用しやすい形で提供するにとどめることとしました。

・第１０条　何度も繰り返す

リリース後の運営方針でも、将来的なあるべき姿に向かってサービス設計１２箇条を意識して改善を続けることを明示しています。

※ 本ガイドブックにおけるアドバイザーは、一般的なアジャイル開発

における「ステークホルダー」に相当する役割を想定しています。

２．３　アジャイル開発の適用方針

1)　アジャイル開発に向いている・不向きな領域

アジャイル開発には、前述のような意義がありますが、どのような場合でもアジャイル開発を採用すれば良いというものではありません。アジャイル開発が向いている領域や不向きな領域について以下に示します。

ア　向いている領域

開発対象についてある程度の方向性はあるものの、全容が明らかになっておらず、開発を進めながら詳細化していく必要があるケース。あらかじめ詳細を決めることができない、あるいは決めにくい領域（例えば、利用者の体験デザインから検討が必要な情報システム)。

イ　不向きな領域

あらかじめ対象範囲や実現するべき詳細が定められており、明らかになっているケース。業務内容が明らかになっており、作って確認するという余地が少ない領域。

ただし、業務内容が明らかになっている領域であっても、具体的な UI（ユーザインタフェース）の部分は、利用者からのフィードバックを得ながら構築するアジャイル開発が向いている場合があります。

ウ　慎重な判断が必要な領域

大規模な情報システム、業務内容等が極めて複雑、あるいはミッションクリティカルな(業務、サービス提供上ほぼ一切の障害や誤作動が許されない)ケース。

このような場合は、どこまでをあらかじめ詳細化するか、どの部分をアジャイルに開発するか、また、どのように品質を確保し、継続的に高めていくかといった判断が必要となります。

2)開発方針の検討

アジャイル開発の向き・不向きを踏まえ、どのような開発方針を採用するか検討しましょう。開発方針の決定は、プロジェクト全体に大きな影響を及ぼすため、調達仕様書を作成する際に、有識者と十分な協議を行い判断する

ことを推奨します。

また、調達段階で決定した方針を一方的に押し通したところで事業者が対応できなければ意味がありません。もちろん、開発方針に対応可能と考えられる事業者を選定することが前提となりますが、実際にプロジェクトを進める事業者とも十分なすり合わせを行いましょう。

以下では、どのような開発方針があるのか、２つの軸をもとに分類します。

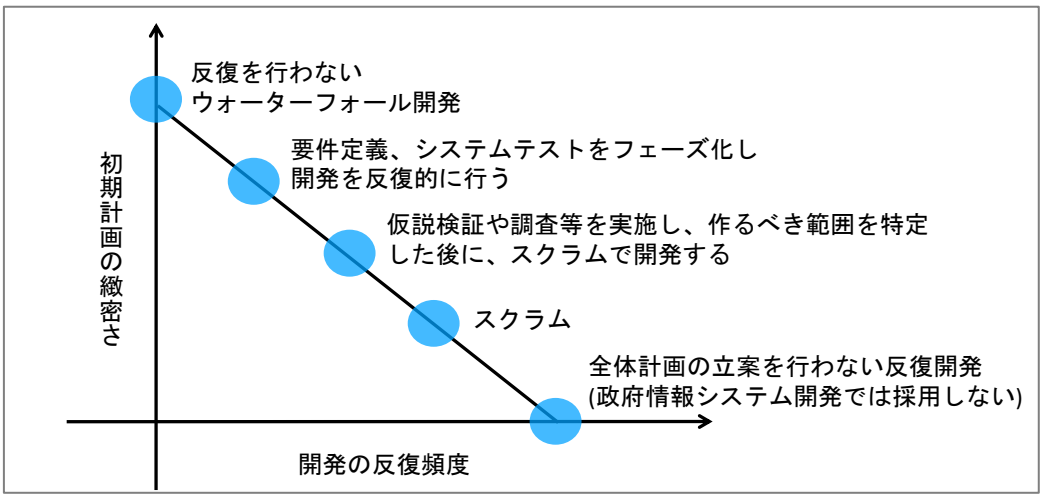
ア　プロジェクトの初期計画の緻密さ

プロジェクトの初期で緻密に計画を立て、大きな変更は認めず、計画どおりに開発を進めていくケースや、まずは全体計画を大まかに決めておき、詳細化は次に開発に取り掛かる範囲だけを対象とするケースがあります。

イ　開発の反復頻度

反復を行わずに要件定義、基本設計などのフェーズを順に進めていくケースや、これまでに完了した作業を踏まえて次にとりかかる開発範囲の計画を立て、開発し、振り返るという一覧の手順を反復して開発するケースがあります。

図　２　２軸による開発方針の分類



２．４　調達時に留意すべきこと

アジャイル開発の調達を行うにあたって、以下の点に留意しましょう。

1)　経験者の参画

開発の大部分を担う事業者の選定にあたっては、事業者側からアジャイル

開発の経験者がプロジェクトに参画することを前提とします。参画者がどのようなシステム開発（領域、規模）において、どのような役割を果たしたのか確認し、参画者の経験と調達対象の領域や規模とがかけ離れている場合は、その差分を解消する工夫が必要となります。アジャイル開発に関する有資格者は一定の知識があるとは判断できますが、それだけでアジャイル開発を実践できるかまでは判断できません。

また、参画者が当該プロジェクトでどのような役割を果たすのか事業者と認識を合わせましょう。責任者やマネージャーとして参画するだけでは十分に機能しない可能性があるため、現場活動への関与者として経験者が存在することを確認しましょう。特に、開発チームの中で重要な役割を果たすスクラムマスターとなる予定の方の見識、経験は重要です。

なお、事業者からさらに外部委託が検討されている場合は、その委託先のアジャイル開発経験を確認する必要があります。事業者や府省職員の経験が不足している場合は、別途外部支援者の確保を検討しましょう。

2) 発注者の姿勢

アジャイル開発を採用する場合、調達仕様書に「開発手法はアジャイル開発を採用すること」と記載すれば、あとは事業者がうまく進めてくれると考えてはいけません。発注者（プロダクトオーナー）は、仕様を決定するためにアドバイザーやエンドユーザーを含む関係者に日々確認や調整を行ったり、事業者と検討や議論を行うための日次ないし週数回以上の打ち合わせの時間を確保したりする必要があります。

これらの十分な時間と、より良いプロダクトのための不断の努力ができる環境を準備できない場合、アジャイル開発でのプロジェクトは成功確率が大きく下がります。

3) 開発範囲にMVP（Minimum Viable Product）の範囲を用意する

MVP とは、実用的で最小限の範囲で動くプロダクトを意味します。具体的には、予定されている開発範囲に含まれる、プロジェクトの課題を解決するために重要でかつ極力範囲を絞った領域を指します。つまり「その時点で必須もしくは解決すべき優先度の高い開発範囲 (MVP)」と「開発対象としたいが実現範囲と内容は調整可能」という少なくとも２つの領域を設けて、調達仕様書に示しましょう。

MVP に関連して「YAGNI」と呼ばれる考え方があります。これは「You Ain't Gonna Need It」もしくは「You Aren't Going to Need It」（あなたはそれを必要とはしない）の略で、XP（eXtreme Programming）が提唱するアプローチの

1つです。XPは、ペアプログラミングやテスト駆動開発、ソースコードを個人ではなくチームで共同して責任を持つことなど、現在のアジャイル開発の基本となる考え方を説いた手法の1つです。多くのソフトウェアで、開発した機能の半分以上がほとんど使われないという統計的データをもとに、今本当に必要なものだけを作ろう、という考え方です。使われない機能であっても、開発すればテストや保守、マニュアル作成等のコストもかかりますし、ソフトウェアの複雑度も増加します。

MVP については確実に実現するようにプロジェクトを進め、そのほかの領域については開発を進めながら実現範囲と内容を決めていく、という方針を置くことで、アジャイル開発を適用したために実現するべき開発範囲が揃わなかったという事態を防ぐようにしましょう。ある情報システムでは、初期に優先度がそれほど高くない機能に工数を使い、後々優先度の高い機能の実現に支障が生じてしまいました。MVP で重要な機能を明らかにしていても、常に残りのスプリント数を注視し、管理しなければ、本当に重要な開発範囲を実現できなくなるおそれがあります。

4) 契約方式を検討する

アジャイル開発では、実現する機能の優先度を決め、プロジェクトの状況によっては、一部の機能の実現を見送ったり、要件を変更したりします。このような場合には、あらかじめ内容が特定された成果物を予定どおりに完成させることに対価を払う請負契約よりも、業務を受託した事業者が専門家としての注意義務を果たしながら業務を遂行することに対価を支払う準委任契約の方が馴染みやすいという考え方もあります。プロジェクトの開発方針を踏まえて、適切な契約方式を検討する必要があります。この点については、実践ガイドブック「第3編第6章 Step.2-1-E.参考：アジャイル開発を行う場合の契約方式」も参照して検討を進めてください。

3　アジャイル開発の運営

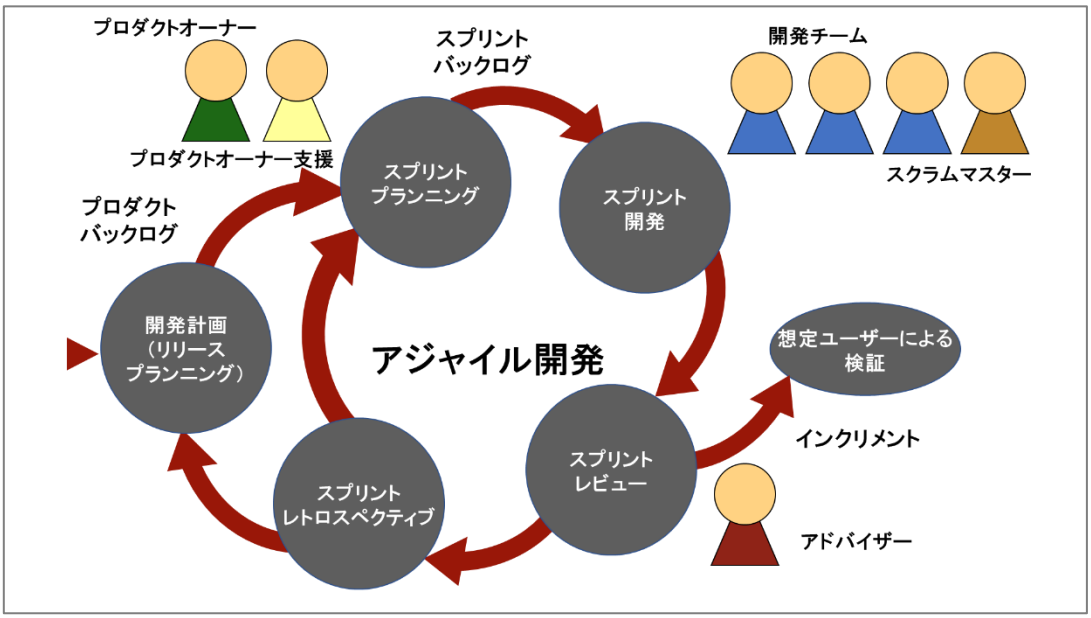
3．1　運営の概要

アジャイル開発の実際の運営に際して、以下の内容を把握しておきましょう。

- 役割
- 実施ミーティング（スクラムイベント）
- 作成物

なお、本ガイドブックでの説明は最小限の内容に留めています。実践にあたってはより広く、深い知識が問われるため、第５章「参考情報一覧」において提示する文献等を参考に知識拡充に努めましょう。

図　１　アジャイル開発の概要（再掲）



1）　役割

アジャイル開発（スクラム）における体制には４つの役割があります。

ア　プロダクトオーナー

開発する情報システムの価値を最大化することに責任を負います。具体的には、開発する機能の仕様策定に関する議論を主導する役割を担い、「プロジェクトとして何を作るべきか」という判断に最も影響を与えます。また、

開発する機能の順序についても、「何から形にしていくべきか」という判断を下す際の中心的な役割を果たします。

政府情報システム開発においては、原則としてPJMOのうち課長級の職員がプロダクトオーナーを務めることが一般的ではありますが、プロジェクトとしての判断を適切に実施できる職員であれば、責任者から当該職員に必要な権限を委譲した上で、当該職員をプロダクトオーナーとすることもできます。

なお、意思決定が混乱しないように、１つのプロジェクトに設置するプロダクトオーナーは原則として１名です。他の職員は、当該プロダクトオーナーが行う意思決定の支援にあたります。

コラム：プロダクトオーナーが果たす役割

あるプロジェクトでは、調達仕様書作成段階からPMOが関与し、PJMOと何度も打合せを実施して仕様を固めました。

開発が始まってからは、そのままPMOが数名のチームで事業者に対応しましたが、思ったほどの開発スピードが出ませんでした。事業者側はワンチームになっていましたが、PMOは分野ごとに担当を分け、それぞれのテーマで個別に打合せしながら開発を進めており、両者でチーム構成が整合していなかったからです。このような構成の違いが、全体として話をまとめあげていくのに足かせになっていました。

そこで、事業者からの要請もあり、このチームでは分野ごとではなく全体での意思決定をできる「プロダクトオーナー」を置くことにしました。一人のプロダクトオーナーに意思決定を一元化したわけです。進め方としては、PMOとPJMOの間で仕様をすり合わせ、それをプロダクトオーナーを含めたPMOチーム内で咀嚼し、プロダクトオーナーが事業者と調整するという流れを作りました。

その結果、開発機能における優先順位や実現方法の意思決定がスムーズになりました。また、事業者側のリーダーを拘束する打合せ時間が削減され、さらには担当間のポテンヒットがなくなる効果もありました。

このようなフォーメーションを組む上では、プロダクトオーナーが事業者やPMOメンバーから信頼されている必要があります。打合せの中で逐一宿題として持ち帰って、意思決定を遅らせてしまうのではなく、自らその場で意思決定でき、しかも後から大きくブレることがないということへの信頼感です。

このプロジェクトでは３ヶ月で、PMOからPJMOにプロジェクトを「大政奉還」しました。このようにプロダクトオーナーの役割を「代行」するのは一

時的には有効ですが、その後もシステム開発を運営していく上では、やはり本来の役割であるPJMOがオーナーシップを発揮すべきだからです。

イ　プロダクトオーナー支援

アジャイル開発におけるプロダクトオーナーの振る舞い並びにシステム開発及びプロジェクト運営に対する職員の知見を補完する役割を担います。特に、プロダクトオーナーが行う意思決定に適宜助言を行い、またプロダクトオーナーの意図を開発チームに適切に伝えるなどの補完的な役割を期待します。

プロダクトオーナー支援は、支援事業者又は政府CIO補佐官が担います。

ウ　開発チーム

情報システムを構築する役割を担い、情報システム構築にあたって必要となる様々な専門性をチームメンバー間で分担します。専門性に基づく分担は各プロジェクトで定義します。

政府情報システム開発においては、事業者がその役割にあたります。本来は発注者も開発チームとして携わる可能性がありますが、政府の情報システム開発においては事例が少ないことを考慮し、本ガイドブックでは事業者の役割としています。

開発チームとして十分にアジャイル開発の練度があり、各メンバーが自律的である場合を除いて、開発チームを取りまとめる役割（リーダー）の設置を検討しましょう。なお、１つの開発チームにおけるチームメンバーは７、８名を上限とし、それ以上となる場合は開発チームの分割を検討しましょう。

エ　スクラムマスター

チーム（プロダクトオーナーを含めた全体を指す）が機能するようにコーチングする役割を担います。具体的には、アジャイル開発が前提としている価値観、考え方、また実践にあたって必要な振る舞い、方法について適宜レクチャーや寄り添った支援を行います。

また、スクラムマスターの重要な役割の一つに「プロジェクト進行の阻害要因の除去」があります。阻害要因とは、例えば、要件や仕様が決まっていないこと、開発環境の不備、仕様に対する不必要な開発チーム外部からの干渉といったものがあります。これらの阻害要因に対し、時には（ウォーターフォールでの）PM（プロジェクトマネージャー）のように、時にはアーキテクトのように振る舞い、臨機応変に阻害要因を取り除きます。ただし、外部

からの干渉をただ排除してはより良いプロダクトにならないため、要不要や重要性を適切に見極めて協働の精神で対応します。

なお、事業者には、スクラムマスターは従来のPMと誤解されることが多い役割です。PMは事業者側の「プロジェクトの責任者」であり、「指揮者・指示者」として上位者の立ち位置から様々なアクションを取ります。それに対して、スクラムマスターは、システムや開発の専門家によって自律的に運営される開発チームに対して対等であるとのスタンスを取り、指示はしませんから、PMとは全く異なります。PMと同じ認識でスクラムマスターを担当するとプロジェクトの進行に混乱を来すので、注意が必要です。

政府情報システム開発においては、設計・開発事業者、支援事業者（工程管理事業者)、政府 CIO 補佐官、PJMO 職員等のいずれかがスクラムマスターとしての役割を担います。１つのプロジェクトに設置するスクラムマスターは１名です。

なお、プロダクトオーナーとスクラムマスターを同じ人が兼任することは避けてください。

オ　アドバイザー

上記４つの役割に該当しない関係者をまとめて指します。システム開発における意思決定には関与せず、あくまでチームに対する助言、フィードバック等を行います。チームが有しない専門性や知見の提供に努め、決してシステム開発の進行を妨げるようなことがあってはいけません。

政府情報システム開発においては、プロジェクト外の府省職員、政府 CIO 補佐官などがあたります。

※ 本ガイドブックにおけるアドバイザーは、一般的なアジャイル開発における「ステークホルダー」に相当する役割を想定しています。一方で、標準ガイドラインでは、直接的又は間接的に影響を受ける外部関係者や内部関係者のことを「ステークホルダー」と定義しているため、便宜上、異なる用語を用いています。

2)　実施ミーティング（スクラムイベント）

実施ミーティングは以下の５つを基本とします。これら以外の会議体はプロジェクトごとに検討し、定義します。

ア　リリースプランニング

リリースプランニングでは、プロジェクト全体の計画作りを行います（本ミーティングはスクラムの規定にはありません)。具体的には、プロジェク

トの開発範囲の規模から、スプリントがいくつ必要かを見立てます。この見立ては、開発の進行具合やフィードバックを取り込む分量によって変わっていくため、プロジェクト開始後も継続的に実施する必要があります。プロジェクト予算からスプリントの数が足りなくなることが予測された場合は、ただちにその対策を練ります。

まず、プロジェクトを始める最初の段階で実施します。その後は、数スプリントに１回の頻度か、１か月に１回は実施します。

イ　スプリントプランニング

スプリントプランニングでは、これから始めるスプリントの計画を作るために、２つの活動を行います。１つは、プロダクトバックログの中で当該スプリントではどれを開発対象とするか（＝スプリントバックログ）を決定することです。開発対象の決定にあたっては、当該スプリントで何を実現するべきなのかというゴールについて、プロダクトオーナーを含めたチーム全体で確認し、そのゴールを実現するのに必要な開発対象を選定します。

もう１つは、開発対象となった機能をどのようにして完成させるかについての議論です。開発に必要な作業などを挙げて、やるべきことの不明点を明らかにしたり、必要に応じて詳細化したりします。

スプリントプランニングは、スプリントの最初に行い、１つのスプリントにつき１回実施します。所要時間は、１か月スプリントの場合であれば最大８時間を目安とします。実際の運用にあたっては、２週間スプリントであれば４時間、１週間スプリントであれば２時間など、各プロジェクトの実情を踏まえて調整しましょう。

ウ　デイリースクラム

デイリースクラムは、スプリントのゴールが達成できそうかを日々確認するために行います。基本的には、開発チームの状況を共通理解とするべく以下の３点を確認します。いずれもスプリントのゴールを達成できるのかを判断するための情報提供にあたります。

- 昨日実施したこと
- 本日実施すること
- 直面している問題、懸念点

毎日決められた時間に実施し、１回の所要時間の目安は１５分程度です。議論が発生し、それ以上の時間となる場合は、デイリースクラム後に別途ミーティングを持って必要な参加者を集めるようにします。

通常は開発チームのみで実施するものですが、この拡張版として、プロダ

クトオーナーなどを交えて仕様を確認・決定する協働の場としてのミーティングを、日次や隔日で定例化したものも開発効率化の観点で有効な場合があります。そこで決定されたものが「受け入れ条件（アクセプタンス・クライテリア)」となり、開発チームに開発作業の指針やゴールとして示すことができます。

エ　スプリントレビュー

スプリントレビューは、スプリントの成果を確認し、今後（特に次のスプリントで）何を行うべきかを決定するために行います。チーム、アドバイザーが参集して、スプリントの成果を確認し、ゴールがどの程度達成できたかを判断します。また、確認したスプリントの成果を基に、今後何を実施するべきか議論を行います。このレビューで寄せられる情報システムへのフィードバックは一旦チームで受け止めた後、次のプランニングで適用可否や実施順序を検討します。

スプリントの終わりに実施し、１つのスプリントにつき１回実施します。所要時間は、１か月スプリントの場合で最大４時間です。

オ　スプリント・レトロスペクティブ（ふりかえり）

スプリント・レトロスペクティブは、スプリントの活動を省みてプロダクトオーナーを含めたチーム全体としてのカイゼンを計画するために行います。次のスプリントの活動が効率的、効果的となるよう、継続すべき工夫、取り除くべき問題、そのための施策などを検討します。

スプリント・レトロスペクティブは、スプリントレビューの次に行うため、スプリントの最終ミーティングにあたります。所要時間は、１か月スプリントの場合で最大３時間です。

3)　作成物

スプリントでは３つの作成物があります。

ア　プロダクトバックログ

プロダクトバックログとは、システム開発に必要となる要求のリストです。リストの並び順で取り組むべき順序を表現します。プロダクトバックログの一つ一つについて、それ 1 つで完結する機能を目安として記述しますが、どの粒度でどこまでの記述を行うかはプロジェクト内で決めます。少なくとも開発チームが規模を見積もれる程度の情報が必要です。

リリースプランニングでは、その時点でのプロダクトバックログを俯瞰し、

必要なスプリント数を概算するために使います。スプリントプランニングでは、当該スプリントで開発対象とする範囲を決めるために使います。

スプリントの途中や、スプリントレビューで必要と考えられたシステム要求については適宜プロダクトバックログへの追加を行います。プロダクトバックログの順序については、スプリントプランニングやスプリントレビューで確認し、適宜並び替えを行います。

なお、同様にプロジェクトで取り組むべきものを列挙したリストとして作業又は成果物を階層的に要素分解した WBS がありますが、これは下位レベルの要素を足し合わせると上位レベルと一致する「100%ルール」に従って作成されるため、トップダウン的なアプローチとなります。一方、プロダクトバックログは個々の「やるべきことリスト」であり、発生の都度バックログは追加されるため、目下の関心事を中心としたボトムアップ的なアプローチとなります。

イ　スプリントバックログ

スプリントバックログとは、当該スプリントで開発対象となった要求のリストであり、その要求を実現するために必要な作業を含んでいます。スプリントバックログは開発チームによる、開発チームのための計画です。デイリースクラムで状況を把握できる程度に詳細にしておきます。

スプリントプランニングでは、当該スプリントで開発対象とする範囲として特定されます。デイリースクラムでは、何が終わっていて、これからどれを始めるのかなどの確認の対象となります。

ウ　開発成果物（情報システムの機能）

開発成果物（スクラムではインクリメントと呼ぶ）は、機能する情報システムそのものであり、スプリントによって完成された機能性が追加されていきます。開発成果物はスプリントごとに常に検査され、完成された状態を保つ必要があります。なお、「完成」の定義はプロジェクトごとに決定します。何をもって完成したと言えるのかは、開発チームで共通の理解とします。

スプリントレビューで、スプリントの成果として具体的な確認の対象となります。

３．２　運営の体制

政府情報システム開発において、アジャイル開発の運営は３．１の「1）役割」で示す役割を担うPJMO職員、設計・開発事業者、支援事業者、政府CIO補佐官などによって行われます。必要に応じて、体制を補完する役割として、

「プロダクトオーナー支援」の設置を検討します。
ここまで説明してきたとおり、アジャイル開発の運営は各者の協働を前提とし、全体で１つのチームとして運営にあたります。

３.３　運営の準備

アジャイル開発の運営を始めるにあたり、以下の点を準備として取り組みましょう。

1)　アジャイル開発に関する知識の獲得

本ガイドブックを手がかりにアジャイル開発に関する知識の獲得を進めましょう。また、各府省の PMO、政府 CIO 補佐官、情報通信技術（IT）総合戦略室等にアジャイル開発の勉強会やレクチャー会の開催を相談することも選択肢の一つです。

上記の場を利用し、運営に関する疑問の解消及びプロジェクトの実際の状況や課題を想定したシミュレーションを行うことも良いと考えられます。これまで実施を終了しているプロジェクトを題材に、アジャイル開発として運営した場合の利点や課題について関係者内で意見交換を行いましょう。

2)　事業者との協働

アジャイル開発を成功させるためには、プロダクトオーナーである発注者と、開発チームである事業者との協働が極めて重要です。発注者（プロダクトオーナー）は、システムの専門家・有識者である事業者と共に考え、議論し、事業者との共通認識として仕様を固める作業に多くの時間を費やします。情報システムが提供するサービスはどうあるべきかという、情報システムがその仕様によって実現する根幹部分だけでなく、場合によっては UX（ユーザー体験）／UI を通じたシステムの振る舞いの細かな部分までがその対象となります。もちろん、事業者はそういった観点でシステムの専門家・有識者たる技量や経験が必要ですし、事業者はその議論や決定のために日々技術調査や検証を行ったり、プロトタイプやモックの作成を行ったりしつつ、発注者とより良いプロダクト作りに注力することが求められます。

協働を促進するために、事業者を決定した後は、速やかにチームビルドのためのセッションを持つようにしましょう。セッションでは、プロジェクトの目的から、開発する情報システムの特徴、チームに参画するメンバーの役割とコミットメント、プロジェクト内における優先基準、リスク、想定スケジュールなどを確認します。

こうした内容を確認するワークショップとして「インセプションデッキ」

作りをチーム、関係者で行うことを推奨します。インセプションデッキについては、第５章「参考情報一覧」の「５．１ アジャイル開発全般」に記載の『アジャイルサムライ』や『カイゼン・ジャーニー』を参考にしてください。

3) 当該プロジェクトでの開発方針を定める

「２．３　アジャイル開発の向き、不向き」で示したとおり、「アジャイル開発」にも進め方や考え方の幅があります。これから始めるプロジェクトにおける「アジャイル開発」とは何なのか、言葉にして、チーム及び関係者と認識を合わせておくようにしましょう。アジャイル開発である狙いや課題、具体的な進め方などは、「本ガイドブックのとおりとする」という周知ではなく、プロジェクトごとに定義しましょう。

チームにおいてアジャイル開発の経験が不足している場合、要件定義フェーズを設置するなど、新たな取組に対する難易度を下げる処置を取るようにしましょう。

4) 全体計画についての認識を合わせる

アジャイル開発だからといってプロジェクト全体の計画が不要なわけではありません。想定スケジュールについての確認や認識合わせを、チーム全体で必ず行うようにしましょう。

計画を確認する際は、マイルストーンの確認を行うようにしましょう。プロジェクトで実施するイベントや報告ごとに(あるいは月単位で)、プロジェクトの断面としてどのような状態になっているのが望ましいのか、チーム全体で確認しましょう。マイルストーンの定義がない場合はこれを作ることから始めましょう。

マイルストーンを設ける上で、「早めに小さく失敗しておく」ということも意識します。アジャイルでは、要件や仕様を詳細化しながら開発するため、MVP として初期開発したものが目的に合致しないことや、合致していても将来に向けて技術的な「負の遺産」となると想定されること等があります。その場合、MVP そのものを一度破棄して再度作り直すことがあり得ます。そのため、作り直しが発生する可能性を織り込んだマイルストーンを設定し、リスクを低減します。

なお、全体計画の段階で、従来型の開発に慣れ親しんだ方が「最初に決めたものをすべて作る」という姿勢を貫こうとしてしまうことがあります。アジャイルでは、MVP から始めて機能を順次追加して実際に動くものを見て考えや発想を進化させていくので、そのことを前提として、マイルストーンを設定していきましょう。

そして、リリースプランニングにおいて、想定する開発規模に対して必要となるスプリント数を見立てましょう。

5) チームでワーキング・アグリーメントを決めていく

準備を経て開発を始めてからも、進め方について認識齟齬は生まれるものと考えましょう。チームの活動を行うにあたって、必要なルールや守るべき原則をチームの中で決めても構いません。

ワーキング・アグリーメントとは、チーム活動を円滑なものにするためのチームメンバー同士の約束事です。コミュニケーション上お互いに守りたいことなどを定義し、折に触れて確認し、意識できるようにしていきましょう。

6) チーム内コミュニケーションの一元化

アジャイル開発では、動くシステムを基に、開発者が利用者（又はプロダクトオーナー）とコミュニケーションを行い、フィードバックを得て開発を進めていきます。このため、チーム内で円滑にコミュニケーションを行えるよう配慮が必要です。チームの組織構造が複雑化して開発チームのリーダー層とプロダクトオーナー支援とのやり取りが増え、開発者とプロダクトオーナーが直接コミュニケーションを取れなかったり、開発を事業者へ丸投げしてコミュニケーションが発生しなかったりといった状況は避けなければなりません。

コラム：アジャイル開発における Web 会議を中心としたミーティング

テレワーク環境の急速な普及に伴い、アジャイル開発においても Web 会議を中心としながらミーティングを行うことが多くなっています。

あるプロジェクトでは、プロダクトオーナー等の職員側と開発事業者の主要メンバーでデイリースクラムを実施する様子を、地方にある開発ルームの真ん中に大型ディスプレイを接続して映し出し、コーディング等を行うエンジニアも確認できるようにしました。プロダクトオーナーの指示が開発メンバー全員にブロードキャストされたわけです。

その結果、デイリースクラムで主要メンバーと検討している最中にもその後ろでエンジニアが修正し、１０分前に議論したことがすぐに画面に反映されて確認できるといった形で、スピーディに議論を進めることが可能となりました。

３．４　運営の適応

アジャイル開発の本質は、経験したことに基づいて、その次の活動を最適化していくところ（適応）にあります。この適応は、情報システム及びチームそのものにおいて行います。情報システムを開発し動かしてから可視化されることで判明することや改善点を次のスプリントプランニングに反映していきます。また、チームとしても、スプリント・レトロスペクティブ（ふりかえり）を通じて、開発の進め方についてのカイゼンを常に行いましょう。これらの適応を両輪として、プロジェクトを進めていきましょう。

また、全体の計画や理解（インセプションデッキで表現する内容）についても、開発を進めていく上で適宜ふりかえり、目的が果たせているのかを確認しましょう。ふりかえった結果、計画や理解について修整を行う必要がある場合はこれを行い、変更点に対してチーム全体で認識合わせを行うようにしましょう。

４　各種留意事項

４．１　全般的な事項

1)　各種ドキュメントについて

アジャイル開発では包括的なドキュメントよりも動くシステムを重視するからと言って、全くドキュメントを作成しなくてよい、というのは必ずしも正しい認識ではありません。ドキュメント作成が重視されがちなウォーターフォールと比較して、不必要なドキュメント（保守工程で参照される機会があまりなく、ソースコードでも代替できるプログラムの詳細設計書等）は省くという姿勢はありますが、例えばシステム全体の概要を認識するための構成図や基本設計、ERD やエンティティの定義、システム境界のインターフェース等をドキュメント化することはしばしば運用・保守や改修に役立ちます。

また、ウォーターフォールでは、情報システムを開発する前にドキュメントを作成して、要求や仕様を言語化しますが、アジャイル開発では、動くシステムによって要件を調整したり変更したりするため、ドキュメントの作成は主要な機能の開発後や運用開始前に行うことも多くなります。ドキュメントを作成しながら開発を進めることを妨げることはありませんが、構築した情報システムを基にドキュメントを作成する方が効率的なことも多く、開発の前や開発と並行して作成するドキュメントと、プロジェクトの後半又は終盤に作成するドキュメントとを見極め、後者のためにプロジェクトの後半又は終盤に作成期間をまとめて設けることが望ましいです。

2)　要件定義書について

アジャイル開発であっても、限られた工数の中で際限なく変更を取り込めるわけではなく、従来の開発スタイルと同様に、あらかじめ作るべき機能やデータモデルなどを把握しておくことが重要です。特に、データモデル（ERD やエンティティ定義等）とシステム境界のインターフェースの定義は後から変更すると影響が大きいこともあり、予め整理・整頓しておくことが望ましいものの１つです。

プロジェクトの制約が厳しいほど、開発範囲と優先度を明確にし、スムーズにプロジェクトを運営することが必要となります。特に開発期間が短いプロジェクトでは、スプリント期間中にも要件が決まらず開発が滞ったり、関係者への根回し不足によって「ちゃぶ台返し」が発生したりしないように注意が必要です。

コラム：要件定義書における工夫

あるアジャイル開発を前提としたプロジェクトでは、発注者側の要件や思いを事業者に分かりやすく伝えて短期間での開発を円滑化するため、要件定義書で次のような工夫を行いました。

- 要件定義書の別添として「システムイメージ（案）」を作成。利用者、関係者間での業務や情報のフローを図解するとともに、利用者向け機能（スマートフォンを想定）や業務担当者向け機能（PC を想定）の画面イメージ、画面遷移、入力・表示項目、統計データ分析のアウトプットイメージ等を、あくまで調達時点での案としながらも具体的に記載。
- 管理するデータについても、Raw データ、正規化データ、統計データ等に大きく分類した上で、主要なデータについてデータ項目やリレーション等を案として記載。

また、運用フェーズの品質確保と効率化のため、事業者から提案を求める事項として次のような工夫を行いました。

- システム運用に関する事項を「稼働監視、セキュリティ監視、リソース監視等を適切に行う仕組みとして、基本的にはクラウドのマネージドサービスを活用することを想定しているが、必要な場合は理由を添えて想定するツールを提案すること」として、事業者からの提案を積極的に依頼。
- 運用フェーズでの CI（継続的インテグレーション）について、インフラ層もコード化（Infrastracture as Code）してテスト環境や検証環境を容易に準備できることを要件とした上で、ツールや運用方法の提案を依頼。
- ミドルウェア等のバージョンアップやアプリケーションの機能改修に伴う検証工数を削減するために、デプロイ方法（カナリアリリース後のローリングデプロイやブルー/グリーンデプロイ等）によってコストを下げながらバランスよく対応する方法の提案を依頼。

3)品質管理について

アジャイル開発であっても従来の開発スタイルと同様に、品質の管理が重要です。そのために、発注者と受注者が協力して開発に取り組み、開発された情報システムの品質の良し悪しを発注者が判断する必要があります。

また、アジャイル開発は変更に柔軟に対応するのが特徴ですが、変更を際限なく取り込めるわけではありません。特に短納期のプロジェクトでは、プ

ロダクトバックログやスプリントバックログの作成にあたって、品質を確保するために、残りのスプリント数などを考慮して要件に優先度をつけるなど現実的な判断を行うことが必要です。

ある省庁では、品質確保の工夫として、機能や実現方式等を動くシステムを用いて先行して決定する一方で、品質を確保するために専門のチームを設置し、これらを並行して進めました。

４．２　第三者チェックの有効活用

まだ政府内ではアジャイル開発の事例は多くありません。アジャイル開発に初めて取り組む場合は、上手くいっているのか、このまま進めて大丈夫なのかといった不安が生じるだけでなく、当事者では気付けないリスクが潜在している可能性もあります。

そこで、開発手法としてアジャイル開発を採用する場合には、専門知識を有する第三者（CIO 補佐官、外部の支援事業者など）による状況判断の機会を設けるようにしましょう。ただし、せっかくそのような機会を設けても、形式的なセレモニーとなっては意味がありません。プロジェクトの運営状況や課題について説明し、専門知識を有する第三者の理解と適切な助言を得られるようにしましょう。定期的に状況判断の機会を設けるだけでは不安な場合には、支援事業者に伴走型で支援してもらうことも選択肢の一つです。このような対策をしっかり行うことが、重要な機能が実現できなかった、利用者の求めるものと違う情報システムになってしまったなどといった問題を未然に防ぐことに繋がります。

状況判断の機会を、全体プランニングのタイミングでマイルストーンとして設定することも重要です。

４．３　継続的なチーム体制の確立

システム開発プロジェクトは構築した情報システムをリリースして終わりではなく、その後も継続していくものです。情報システムがリリースされてから実際に情報システムが利用者に利用された結果、使い勝手の良し悪しなどがフィードバックされ、それを踏まえて改修していくことを考えると、むしろ、稼働を開始してからが始まりであるとさえいえます。

設計・開発の工程をふりかえり、保守開発において必要なチームの構想や工夫を整理しましょう。例えば、国民など、組織外の利用者のニーズが把握しきれず、仮説を立てて構築した機能については、実際の利用状況を分析したり、利用者のニーズを探索したりすることとなるため、それらのスキルを持ったメンバーを新たに迎えることを検討する必要があります。

また、情報システムを開発する段階で、ユーザーのログイン数やアクセス数、登録・変更等の処理件数といった利用状況を収集・可視化することができる仕組みを（情報の取り扱いやセキュリティに配慮しつつ）予め組み込んでおくと、分析に役立ちます。

５　参考情報一覧

本ガイドブックを作成するにあたり、以下の情報を参考にしています。

５．１　アジャイル開発、全般

- 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「アジャイルソフトウェア開発宣言」(2001 年)
  https://agilemanifesto.org/iso/ja/manifesto.html
- 「アジャイルソフトウェアの 12 の原則」(2001 年)
  https://agilemanifesto.org/iso/ja/principles.html
- 市谷聡啓・新井剛・小田中育生『いちばんやさしいアジャイル開発の教本 人気講師が教える DX を支える開発手法』インプレス、2020 年
- Jonathan Rasmusson 著、西村直人・角谷信太郎監訳、近藤修平・角掛拓未訳『アジャイルサムライ 達人開発者への道』オーム社、2011 年
- 市谷聡啓・新井剛『カイゼン・ジャーニー たった 1 人からはじめて、「越境」するチームをつくるまで』翔泳社、2018 年
- 宇田川元一『他者と働く 「わかりあえなさ」から始める組織論』NewsPicks Publishing、2019 年
- Robert C. Martin 著、角征典・角谷信太郎訳『Clean Agile 基本に立ち戻れ』ドワンゴ、2020 年
- Steve McConnell 著、長沢智治監訳、クイープ訳『More Effective Agile “ソフトウェアリーダー”になるための 28 の道標』日経 BP、2020 年
- サイモン・シネック著、栗木さつき訳『WHY から始めよ！』日経 BP、2012 年

５．２　プロジェクトマネジメントや運営

- Mike Cohn 著、安井力・角谷信太郎訳『アジャイルな見積りと計画づくり 価値あるソフトウェアを育てる概念と技法』マイナビ出版、2009 年
- エリヤフ・ゴールドラット著、三本木亮訳『クリティカルチェーン なぜ、プロジェクトは予定通りに進まないのか？』ダイヤモンド社、2003 年
- Henrik Kniberg 著、角谷信太郎監訳、市谷聡啓・藤原大共訳『リーン開発の現場 カンバンによる大規模プロジェクトの運営』オーム社、2013

年
・ Marcus Hammarberg・Joakim Sundén 著、原田騎郎・安井力・吉羽龍太郎・角征典・高木正弘訳『カンバン仕事術 チームではじめる見える化と改善』オライリー・ジャパン、2016 年
・ メアリー・ポッペンディーク・トム・ポッペンディーク著、平鍋健児・高嶋優子・佐野建樹訳『リーンソフトウェア開発　アジャイル開発を実践する 22 の方法』日経 BP、2004 年
・ Project Management Institute 編、「プロジェクトマネジメント知識体系ガイド」第 6 版

５.３　スクラム

・ Ken Schwaber・Jeff Sutherland 著「スクラムガイド スクラム公式ガイド：ゲームのルール」(2020 年 11 月)
https://www.scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Japanese.pdf
・ Kenneth S. Rubin 著、岡澤裕二・角征典・高木正弘・和智右桂訳『エッセンシャルスクラム アジャイル開発に関わるすべての人のための完全攻略ガイド』翔泳社、2014 年
・ 西村直人・永瀬美穂・吉羽龍太郎『SCRUM BOOT CAMP THE BOOK スクラムチームではじめるアジャイル開発 増補改訂版』翔泳社、2020 年
・ Mitch Lacey 著、安井力・近藤寛喜・原田騎郎訳『スクラム現場ガイド スクラムを始めてみたけどうまくいかない時に読む本』マイナビ出版、2016 年

５.４　スプリント・レトロスペクティブ（ふりかえり）

・ 天野勝『これだけ！KPT あらゆるプロセスを成果につなげる最強のカイゼンフレームワーク』すばる舎リンケージ、2013 年
・ Esther Derby・Diana Larsen 著、角征典訳『アジャイルレトロスペクティブズ 強いチームを育てる「ふりかえり」の手引き』オーム社、2007 年

５.５　チーム開発

・ 市谷聡啓『チーム・ジャーニー 逆境を越える、変化に強いチームをつくりあげるまで』翔泳社、2020 年
・ エイミー・C. エドモンドソン著、野津智子訳『チームが機能するとはどういうことか 「学習力」と「実行力」を高める実践アプローチ』英治出

版、2014 年

５．６　プロダクト開発

- エリック・リース著、井口耕二訳『リーン・スタートアップ ムダのない起業プロセスでイノベーションを生みだす』日経 BP、2012 年
- 市谷聡啓『正しいものを正しくつくる プロダクトをつくるとはどういうことなのか、あるいはアジャイルのその先について』ビー・エヌ・エヌ新社、2019 年