デジタルプラクティス　Vol.11 No.2(Apr. 2020)

特集号招待論文

エンタープライズ領域のアジャイル開発の課題 ―アジャイル開発がもたらす意思決定プロセスの変化―

鈴木雄介[1]

[1]グロース・アーキテクチャ&チームス（株）

エンタープライズ領域においてデジタルトランスフォーメーション（以下DX）と言われるようなビジネスモデルの変化に合わせ，システム開発にアジャイル開発プロセスを導入する取り組みが増えている．しかし，実際に導入しても高いビジネス成果を上げることは容易ではない．本稿ではシステム開発手法を組織の意思決定プロセスと捉え，エンタープライズ領域におけるアジャイル開発の課題を論じる．また事例を挙げて，この解決に向けた取り組みを紹介する．

1. エンタープライズ領域のアジャイル開発

筆者はシステム開発手法を意思決定の仕組みとして考えている．この意思決定という観点からエンタープライズ領域におけるアジャイル開発の課題について論じる．意思決定という点においては，単にシステム開発にとどまらず，企業活動の基本，さらには日本人という背景についても敷衍して理解する必要がある．

1.1 エンタープライズアジャイル

エンタープライズ領域のアジャイル開発を示す言葉として「エンタープライズアジャイル」という用語を定義する．なお，エンタープライズアジャイルについて広く知られた定義は存在しない（コラム参照）．このため，本稿では以下のように定義する．

- ウォーターフォール開発プロセスを主に実施している組織において実施されるアジャイル開発プロセスのこと．

現状，エンタープライズ領域で広く行われているシステム開発プロセスはウォーターフォール開発プロセスと呼ばれている．これは以下のような特徴を持つ．

- プロジェクト開始時に開発対象の全機能を定義し，その全機能の開発完了を目標とする．
- 要件定義→基本設計→詳細設計→実装→テストというようなフェーズを適用し，フェーズごとに成果物の確認を行うことで品質を高める．
- 初期および設計フェーズでの成果物は文書で規定される．

長年にわたってウォーターフォール開発プロセスを実施してきた企業では，その組織および従業員の思考や行動がウォーターフォール開発プロセスに適応してしまっている．さらに企業文化というような暗黙的な判断基準にまで浸透している．

そういった状況でアジャイル開発プロセスを導入しても高い成果を上げることが難しいことがある．これはアジャイル開発プロセスが前提としている思考や行動が，ウォーターフォール開発と大きく乖離しているにもかかわらず，これまでの文化に強い影響を受けてしまうためである．

コラム：エンタープライズアジャイル勉強会

エンタープライズアジャイル勉強会[☆1]は2015年6月に発足したコミュニティであり，筆者も実行委員として参画している．本勉強会には日本を代表するアジャイル開発のメンバが参加しているが，その中でもエンタープライズアジャイルに対する解釈は定まっていない．本勉強会がとりまとめた「エンタープライズアジャイルの可能性と実現への提言―アンチパターンとその克服事例」[☆2]では，エンタープライズアジャイルの3つの可能性として以下を定義している．

- 第1の可能性
  業務の変化に対応し，基幹系やバックオフィス系のシステム（「エンタープライズシステム」）を効果的に開発するためにアジャイル開発を適用する．
- 第2の可能性
  競争力に資する製品やサービス，戦略的システムを開発するためにアジャイル開発や反復開発を大規模に適用する．
- 第3の可能性
  変化により良く対応できる，活力のある組織を実現するために企業や事業部をアジャイル化する．

この中では筆者の考え方は第2の立場と言える．

1.2　開発手法と決定タイミング

では，アジャイル開発が前提とする考え方はどういったものだろうか．

CollabNet VersionOne社が刊行した13th Annual State of Agile Reportはグローバルでのアジャイル開発の実態を理解することができるレポートである[1]．この中でアジャイル開発を採用した企業が挙げている利点の上位3つは以下のようになっている．

- 優先順位の変更を管理する機能　69%
- プロジェクトの可視性　65%
- ビジネスとITの足並みを揃える　64%

まず2点目と3点目は「ビジネスと提供されているシステム機能がリンクしており，かつ，提供済みの機能が明確になっている」ことを示している．そして，この前提の上で1点目は「ビジネスの状況に応じて開発すべき機能の優先順位を決定できる」ことを示す．このように，アジャイル開発プロセスとはビジネスとシステム機能の関係性を管理することが目的となっている．

一方，ウォーターフォール開発プロセスでは「対象の全システム機能を開発する」ことを目的として計画を立案し，実行する．つまり，開発すべきシステム機能の決定は計画時点で確定し，その大きな決定に従った開発を推進することを前提としている．なお，要件定義，基本設計といったフェーズ

は，このような当初の決定に対して適切な品質が実現されるのかを確認するための仕組みであり，当初の決定を変更したり．開発すべき機能の優先順位を変えるといったことには使えない．この点については次章にて詳述する．

システム開発において「どんな機能を作るのか？」という決定タイミングに注目すると，アジャイル開発プロセスは「適宜，状況に対応して決定していく」ということである．そのため，いかに状況とシステム機能を一致させていくのか？ということが目的となる．一方，ウォーターフォール開発プロセスは「最初にすべてを決定する」ということになり，いかに最初の決定を達成するか？ということが目的となっている（**表1**）．

表1　開発手法が前提とする意思決定タイミングと目的

開発手法	システム機能の決定タイミング	プロセスの目的
アジャイル開発プロセス	適宜，状況に対応して決定していく	いかに状況とシステム機能を一致させていくのか？
ウォーターフォール開発プロセス	最初に全てを決定する	いかに最初の決定を達成するか？

ウォーターフォール開発でも「状況に応じた変更」をするが，これは，あくまでも進捗遅延などの状況が発生した場合であって，最初に決定された機能の開発を遵守するために行われる．調整手段として当初の決定を変更することもあるが，これは積極的な選択肢ではなく，むしろ，開発の失敗と見なされることも多い．つまり，ウォーターフォール開発では最初の決定が正しいことが前提となる．

1.3　エンタープライズアジャイルの課題

エンタープライズアジャイルとは「最初にすべてを決定する」ことが当たり前となっている組織において「状況に応じて決定する」ことに取り組むことになる．これは，「最初に決定された事項を変更する」ことを意味する．こうした環境では，アジャイル開発プロセスのメリットを活かせないばかりか，アジャイル開発プロセスが既存の組織や従業員が遵守してきた決定を軽視し，毀損しているとすら考えられることがある．

たとえば「AIを使った需要予測」という機能が決定されていたとしよう．この機能について設計をしていったところ「需要予測に必要な精度のデータを入手することができない．現状のデータ精度では，正確な需要予測ができない」といったことが判明したとする．こうした場合，当然ながらビジネス上の価値が得られないと判断されたのだから，状況が変化したと理解すべきである．アジャイル開発プロセスであれば，これは「ビジネスの状況の変化」なので，当初の決定を変更すればよいと考える．一方，ウォーターフォール開発プロセスに慣れた組織や従業員は「当初の決定を遵守することが重要だ」と思考してしまい「精度が低くても機能を提供すべきだ」という行動をとる．

さらに，この行動を強く後押しするのが予算獲得における稟議決裁である．稟議決裁では，開発対象となるものを明示し，それに対して予算が妥当であることを説明する．前述の例であれば「AIを使った需要予測」というのが名目となり，これを実現するための予算を記載し，稟議を上申する．そして，決裁プロセスでは役員を含めて多くのステークホルダの承認が行われる．その結果，「決定の変更」という要素が「開発すべき機能を変更する」だけではなく「企業の意思決定を変更する」というレベルに昇華してしまう．「企業の意思決定を変更する」という行為は稟議決裁にかかわった多くの

ステークホルダとの調整が必要になる．さらに，そのステークホルダが外部関係者との合意に利用するケースもある．上場企業であれば株主へ説明済みであった場合など，調整は困難を極めることになる．

こうした「決定の遵守」に対する批判は簡単である．そもそも，稟議決裁時点では「AIを使った需要予測」は実現されておらず，その実現性には不確定要素が含まれる．よって，決定が変更されるリスクが内在しているのは明らかである．しかしながら，企業活動は将来への目標提示と，それを達成するという信用の積み重ねをベースにしている．銀行が融資をする仕組みも株式市場の仕組みも，将来に対する信用を前提としている．このため自己資金だけで運営されている企業でもない限り「決定の遵守」を覆すことは容易ではない．

「決定の遵守」は，それ自体が悪ではない．しかし，失敗であることが明らかになったあとでも「決定の遵守」にこだわるのは愚かである．こうした傾向は「日本人は」という文脈で繰り返し指摘されている．古典的なものとしては『「空気」の研究』（山本七平）や『失敗の本質』（野中郁次郎ほか）などを挙げることができるが，この詳細は本稿の主論ではないため割愛する．

エンタープライズアジャイルの課題を解決するには組織や従業員に対して「状況に応じて決定する」ということのメリットを理解してもらい，これに応じた思考や行動に変革することが重要である．一方で「決定を遵守する」ということも企業活動としては必要な要素である．これらに折り合いをつけて，最適な開発を実現することがエンタープライズアジャイルに求められることなのである．

2. マネジメントプロセスにおけるアジャイル開発のメリット

アジャイル開発プロセスのメリットを理解するために「状況に応じて決定する」ということが，どういった仕組みによって行われているのかを把握しなくてはならない．そこでプロジェクトマネジメントの観点から開発プロセスの整理を行う．まずはウォーターフォール開発プロセスの特性を理解し，DX推進のためのシステムに適用する場合の課題について説明する．次に，こうしたシステムにおいてアジャイル開発プロセスが，どのように機能し，メリットをもたらすかについて説明する．

2.1 マネジメントプロセス

システム開発の遂行にあたり，そのプロセスについてはプロジェクトマネジメントの観点から考えると理解しやすい．

プロジェクトマネジメントに関するナレッジを集積したPMBOK[2]では，以下のようなプロセス群を定義している．

i. Initilating（立ち上げプロセス）
ii. Planning（計画プロセス）
iii. Executing（実行プロセス）
iv. Controlling（監視・コントロールプロセス）
v. Closing（終結プロセス）

これらのプロセスごとに，マネジメント対象として10個の領域（知識エリア）が設定される．スケジュール，コスト，品質，スコープ，人的資源，コミュニケーション，リスク，調達，ステークホルダ，そして，これらの統合である．

プロセスについては，プロジェクトが遂行状態にある場合，重要になるのはii〜ivのプロセスである．

- Planningプロセスは，その名のとおりプロジェクトの計画を立案する．プロジェクトの目標を定義する．
- Executingプロセスは，計画に従い，実際にプロジェクトを動かしていく．
- Controllingプロセスは，実行結果と計画の差分を確認し，しかるべき調整を行って計画どおりになるように是正していく．調整における主要な対象領域は人的資源（コスト），スケジュール，スコープであろう．なお，PMBOKの日本語訳ではControllingを「監視・コントロール」と表記している．しかし，この言葉の意味が強く，一般的な理解にはそぐわないため，本稿では「確認」と「調整」という言葉を利用する．

ここまでを整理すると，プロジェクト遂行中におけるマネジメントプロセスは「計画→実行→確認→調整」である．このうち実行については開発手法によって異なる点はない．そのため「計画」「確認」「調整」というプロセスに対してウォーターフォール開発の特性と課題，アジャイル開発の特性と利点について説明する．

2.2　ウォーターフォール開発の特性

1968年に開催されたNATO国際会議において，システム開発の近代化を目的として「フェーズ（工程）」の導入が議論されている[3]．フェーズは開発プロセスにフェーズを設定し，そのフェーズごとに成果物を作ることで段階的に品質管理を行う手法である．たとえば要件定義であれば要件定義書，設計であれば設計書，コーディングであればソースコードといった成果物を作成し，これを確認することによってテスト前に品質を確認することができる．前述のマネジメントプロセスに照らすと，フェーズごとに成果物を利用して「確認」を行い，「調整」を行うということになる．フェーズ導入前のシステム開発では，テスト段階にならないと品質を評価しないことが一般的であったようである．

1970年代になり「フェーズを遡らないトップダウン型のアプローチ」を説明する言葉として「滝（ウォーターフォール）のような」という言葉が使われ，ウォーターフォール開発プロセスという言葉が広まった．フェーズの完了にはフェーズゲート（門）が設けられ，フェーズ内で作成された成果物の品質を確認する．成果物に問題がなければフェーズを完了し，次のフェーズに進む．仮に問題があればフェーズ内の作業を再点検し，成果物作成を完了させるのである．

コラム：ウォーターフォールの起源について

ウォーターフォールの原形について，W. Royceによる1970年の論文「MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS」[4]が知られている．しかし，実際には1968年に開催されたNATO国際会議でフェーズについての議論がされており，Royceの論文は，彼自身が経験した「後戻りのないウォーターフォール型開発」に対して，それでうまくいかないことを説明したものである．よって，彼の論文ではフェーズを遡ったり，繰り返したりすることを推奨している．

2.3　ウォーターフォール開発の課題

ウォーターフォール開発プロセスはフェーズという特性を持っているが，これでは対応しにくいケースがある．特に主要なユーザが一般消費者である場合，開発したシステムが価値を生み出すかどうかは，実際にシステムをリリースしてユーザからフィードバックを得ないと分からない．

計画における課題は，計画精度の向上が困難であることである．ウォーターフォール開発はゴールとなる対象システムの完成にむけて，プロジェクトの開始から終了までの長期間にわたる計画を立案する．しかし，昨今のシステム開発では，開発対象となるシステムが曖昧に定義されることが多い．企画レベルで「このような機能がほしい」という定義があったとしても，その機能にはビジネス的な実現性も技術的な実現性も曖昧なケースがある．こうした場合，ウォーターフォールでは計画精度を下げて計画立案をせざるを得ない．

次に進捗確認における課題は，精度が担当者に依存することである．計画精度が低い場合，当然ながら進捗確認精度が低くなりがちである．さらにウォーターフォール開発では1つ1つの管理項目が大きくなるため，その進捗確認を進捗率というもので表現する．しかし，計画精度が低いプロジェクトでは進捗率が担当者の主観的な判断に依存することが大きい．一見，定量的に見える目標工数／消化工数で表現される進捗率であっても，消化工数によって，目標に対する機能開発が着実に進捗されていることが前提になるが，計画精度が低い場合には，これは使えない．

また，進捗は「十分な品質が確保されていること」が前提となる，初期フェーズで作成される成果物は文書だけになるめ，文書だけで品質評価を正しく行うことは困難である．特にシステム開発経験が浅い人にとっては文書だけでは理解できないことが多い．そのため進捗確認の精度がさらに下がることになる．

最後の調整における課題は，調整力がプロジェクトマネージャの力量に強く依存することである．計画精度が低く，さらに進捗確認も不正確であれば，当然，調整機能が正しく機能しない．しかし，経験が豊富で力量があるプロジェクトマネージャであれば，こういった状況でも適切なリスクマネジメントを行い，プロジェクトを失敗させないことが可能である．そもそも，計画精度を完璧にすることはできないため，計画時点で不確定な要素があることは当然である．そのためプロジェクトマネージャはバッファと呼ばれるような予備工数を適切に管理し，プロジェクト遂行中に発覚した問題に対して対応する．また，バッファの許容を超えそうな場合においても，顧客との交渉において，どのようなパラメータが調整可能であるかを把握し，問題に対して適切な対応を行うことができる．

しかし，実際には，このようなプロジェクトマネージャは稀有な存在である．特に近年のDX案件では，ビジネスや技術的な複雑さや新規性がプロジェクトマネージャ個人の能力を超えることは少なくない．その結果，優秀なプロジェクトマネージャであってもプロジェクトを成功に導くことは容易ではない．

2.4　アジャイル開発の特性

こうしたウォーターフォール開発の課題に対して，アジャイル開発の特性について論じる．

アジャイル開発の特性は以下のように定義できる．アジャイルソフトウェア開発宣言[5]の背後には12の原則が定義されているが，このうちプロジェクトマネジメントの特性として定義されたものを抜粋する．

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

なお，ここでいうチームとは5〜8名を前提とする．これは米AmazonのCEOであるジェフリー・プレストン・ベゾス（Jeffrey Preston Bezos）氏が提供する「2枚のピザ理論」[6]に基づく．

2.5　アジャイル開発での解決

ウォーターフォール開発プロセスで発生した課題を，アジャイル開発プロセスは次のように解決する（**表2**）．

表2　ウォーターフォール開発プロセスの課題とアジャイル開発プロセスでの解決

	ウォーターフォール開発プロセスでの課題	アジャイル開発プロセスでの解決
計画	対象が大きく，計画が長期間になると，計画精度が高めにくい	短期間の計画しか立案しないため精度が高めやすい
進捗確認	進捗確認精度が担当者の主観的な判断に依存しやすい．また，文書だけでは品質を正しく評価できない	関係者全員で動くソフトウェアを確認するため進捗が明確にできる
調整	プロジェクトマネージャーの力量に強く依存する	関係者全員で話し合うため，個人の力量には依存しない

まず計画についてである．ウォーターフォールでは開発対象が曖昧なことで計画精度が低くなることが問題であった．アジャイル開発では実行期間を短くする．これにより，開発対象となる機能の規模は大きくならない．また，期間が短いため，計画立案時点で対象となる機能や仕様がある程度明確になっている必要がある．この結果，計画精度は高めやすい．

次に確認である．ウォーターフォールでは精度が担当者に依存する点を挙げた．アジャイル開発では実行の成果は動くソフトウェアとして提供し，これをステークホルダを含む全員で確認することとしている．進捗は要件を出したステークホルダ自身が「できている」「できていない（これが足りない）」といった形で明確に確認することができる．このため担当者が誰であったとしても正しく進捗確認が行える．このため進捗率のような作業中にタスクの完成度を考えるということが不要になる．また，動くソフトウェアであれば，文書とは異なり，システム開発の経験が浅い人であっても確認が行いやすい．

最後に調整である．ウォーターフォールでは調整力がプロジェクトマネージャの力量に強く依存することを課題とした．アジャイル開発では小さな計画を繰り返すことが前提となっており，次の計画というものが調整として機能する．アジャイルではチーム人数が多くないため，この調整にはチーム全員がかかわって実施される．結果として，個人への依存度が引き下げられ，チーム全員で最適な調整を行うことができる．また，見積もりへの失敗があったとしても，次の計画でリカバリするといった考え方がしやすい．

前章に述べたとおり，アジャイル開発プロセスは「状況とシステム機能を一致させていく」ことを目的にしている．このため調整を前提とした仕組みになっていることが分かる．

2.6　スクラム開発の有用性

スクラム開発[7]は，アジャイル開発プロセスの特性をうまく活かすためのプロセスフレームワークである．

スクラム開発では実行期間をスプリントと呼び，この期間内で計画と確認を繰り返す．また，プロダクトオーナと呼ばれるロールが開発すべき機能を決定する．開発対象機能や作業などを示すものをバックログと呼ぶ．プロダクトオーナは現時点で求められる全機能をバックログとして記載し，プロ

ダクトバックログと呼ばれる一覧にする．プロダクトバックログでは，バックログは優先順位にしたがって並べられており，最初にあるほど優先順位が高い．優先順位はプロダクトオーナがビジネス上の判断によって決定する．

まず計画はスプリントプランニングと呼ばれており，スプリントの最初に実施される．スプリントプランニングでは，全員でプロダクトバックログの上からバックログを確認し，開発チームがスプリントでの作業内容と見積もりを確定する．そして，実行すると決定されると，バックログはスプリントバックログという別の一覧に移動させる．こうすることで「現時点で必要と思われる全機能」と「開発すると決定した機能」を分離して管理する．前者は優先順位のみで見積もりが行われていないためスケジューリングはできない．後者は見積もりがされており，スプリント内で完了するように計画される．

次に確認である．スプリントの実行期間の終わりには全員でスプリントレビューと呼ばれるイベントを実施する．このイベントでは，スプリントで作成された成果物（アーティファクトと呼ばれる）を見て，動かし，計画どおりに完成しているかを確認する．もし，完成しないものがあった場合には，完成していないことを把握し，仮に残作業を行う必要があるなら，新たにバックログを作成し，プロダクトバックログに優先順位をあわせて追加しておく．

最後に調整である．これは次のスプリントプランニングで行われる．ここでもプロダクトバックログの優先順位にあわせて上から計画を実施するだけでいい．全スプリントの残作業の優先順位が高ければ次のスプリントで実行されるだろうし，ビジネスの状況に変化があったなら，優先順位が下げられているため実行されないこともある．

このようにスクラムでは調整が計画（プランニング）と確認（レビュー）というイベントによって明確に実施されるようになっている．調整するタイミングが定期的に訪れ，かつ，そこにはプロダクトオーナも開発者もいるため，全員の話し合いによって調整内容が決定される．

一方，ウォーターフォール開発プロセスでは調整タイミングがプロセス内で明示されていない．確認はレビューの完了時点，あるいは週次や月次で行われているが，それは確認のみに止まり，調整がされるかどうかはプロジェクトマネージャの判断に委ねられる．

また，スクラムでは調整パラメータが「スコープ」しかない点も重要である．ウォーターフォール開発プロセスであれば，人的資源→スケジュール→スコープの順で調整されることが多いであろう．これは計画の決定事項がスコープ→スケジュール→人的資源（≒コスト）という順になるからである．そのため，調整する順は逆になる．アジャイル開発プロセスではチーム人数が固定化されているため人的資源は調整できない．また，スケジュールも実行期間が確定しているため変更もできない．そのためスコープしか調整できないようになっている．調整パラメータが少ないため，プロダクトオーナと開発チームがとれる調整手段は限られる．つまり，機能を減らすか，減らさないのであれば次のスプリントに実行するか，の2択となる．この制約はプロダクトオーナの判断をシンプルにする．

3. エンタープライズアジャイル導入にむけて

では，どのようにエンタープライズアジャイルを導入すればよいだろうか．稟議決裁については企業の基本的な意思決定プロセスであり，これを否定することは現実的ではない．そこで稟議決裁プロセスを前提として，エンタープライズアジャイルがいかに導入されるかについて考えたい．

筆者の経験上，エンタープライズアジャイルの導入パターンには大きく3つある．

- 半島型パターン
- 孤島型パターン
- 出島型パターン

これは既存のウォーターフォール開発プロセスが適用された既存領域を大陸と捉え，アジャイル開発プロセスを実施する新領域を，どのように配置するのかを示している（**表3**）．

表3　エンタープライズアジャイルの導入パターン

	既存領域との関係	発生する課題	課題への対応
半島型パターン	強く影響を受ける	「初期の決定を変更しない」ことに注力してしまい、ビジネスの状況に対応できず成果をあげられてない	関係者全員がアジャイル開発の特性を理解する
孤島型パターン	影響を受けない	既存領域の資産が使えないため、成果が低くなる	既存領域に理解者を発見し、つなげていく
出島型パターン	適切な関係性にある	ー	ー

3.1　半島型パターン

半島型パターンは，新領域が既存領域に強く影響を受ける状態を示す．特に重要なのが「決定の変更」に対する許容である．既存領域では「最初にすべてを決定し，それを遵守する」が常識になっている．その常識が新領域についても適用される．こうした状況でエンタープライズアジャイルの導入はきわめて困難である．

半島型パターンにおいて，新領域のアジャイルチームが直面する状況は以下のとおりである．

- 稟議決裁にて機能一覧，スケジュール，コストが決定されている．
- 稟議決裁には「アジャイル開発で実施する」との記載があるが「機能は試行錯誤の結果，変わることもある」との記載はない．
- 機能について精緻な実現性の検証が行われていない．
- 新領域に割り当てられた要員は，アジャイル開発プロセスの教育を受けたことはあるが実践したことはない．また，コーチなどのサポートもない．
- 仕様の決定は企画部門が行い，開発されるシステムを使って成果を上げるべき事業部門とのコミュニケーションに距離がある．
- 既存領域の企画部門と開発部門の関係性が持ち込まれ，定期的なミーティングや文書でないとコミュニケーションできない．
- システムのインフラ，リリースプロセスなどは既存領域に従うことが決定されている．
- アジャイル開発プロセスを望んでいるのは開発部門だけ．

半島型パターンの最大の問題は，無意識のうちに当てはまっており，経営層をはじめとして，組織全体，さらにはアジャイルチームのメンバすらも「初期の決定は変更しない」と思っている点である．

特に開発部門だけがアジャイル開発プロセスの採用に前向きな場合は注意が必要である．アジャイル開発プロセスは「ビジネスの状況に合わせて，どんな機能を開発するかを決める」ことができる．ビジネスの状況を正確に把握できるのは事業部門である．そのため事業部門がアジャイル開発プロセスの特性を理解している必要がある．仮に，これが困難である場合は，少なくとも企画部門が理解し

ている必要がある．事業部門や企画部門が既存領域の中にいて，開発部門のメンバだけが半島側にいるような場合，アジャイル開発プロセスは正しく機能しない．前述のとおり，ビジネスの状況に対応する能力があったとしても，ビジネスの状況にあわせた判断を行う人が欠如するからである．

さらにプロジェクトが進捗する中で以下のようなことが判明すると大きなねじれが生じる．

- 決定済みの機能に実現性がない，あるいはビジネス上の価値が低い．
- 決定済みの機能を実現するためには当初以上の工数が必要になる．
- 決定済みの機能にビジネス的な価値が少ない．

アジャイルチームは，定期的な確認と計画を繰り返しているため，上記のような状況を把握し，対応が必要なことを明確に理解する．これに対して企画部門が当初の決定にこだわり続ける場合，状況の変化に対応する判断を行わない．そのためチームは「価値がないと分かっているものを作る」という状況に陥る．そして，実質的にウォーターフォール開発プロセスと変わらない状態が求められる．しかし，アジャイル開発プロセスの導入を行っていると，全体計画が明確になっていないため，プロジェクトの進捗の把握が困難になる．こうした状況に陥るとアジャイルチームが正常化することはきわめて困難である．

半島型パターンになっている場合，やるべきことは経営層，事業部門，企画部門，開発部門を横断して，アジャイル開発プロセスの特性を理解することである．その上で，このプロジェクトにアジャイル開発プロセスを採用する必要があるのか，判断しなくてはならない．

3.2　孤島型パターン

孤島型パターンは，既存領域と新領域が完全に切り離された状態を示す．このパターンであればアジャイル開発プロセスを運営することは困難ではない．ただし，既存領域に事業部門や企画部門が残っている場合，アジャイルチームは高い成果を上げることが困難になる．

孤島型パターンにおいて，新領域のアジャイルチームが直面する状況は以下のとおりである．

- アジャイルチームの創設が決定しており「新規DXサービスの立ち上げ」といったような曖昧な活動定義がある．
- 予算はついており，1年程度の活動費は確保されている．
- 新領域に割り当てられた要員は，アジャイル開発プロセスの教育を受けたことはあるが実践したことはない．ただし，コーチなどのサポートをつけることは可能．
- 従来領域の企画部門はかかわらないが，新領域に企画を担当する専門メンバがいる
- 事業部門がかかわることにはなっているが，新領域の重要性について合意がなく，あまり労力を割いてくれない．
- システムのインフラ，リリースプロセスなどは既存領域に従わない．逆に既存領域のシステムとサーバ間連携が許可されていない．

孤島型パターンの最大の課題は，事業上の成果を出せないことである．DX領域は既存の製品やサービスの変革が大きなテーマとなっている．しかし，「従来領域のルールがあってはうまくいかない」という判断のもと，その制約を避けることに注力した結果，従来領域との関係性が希薄になり，その既存資産を活かすことができなくなる．

たとえばシステム間連携なども実現できないため，手作業でデータを連携するといった事態になる．これでは機能上の制約が大きくなり，もちろん，ビジネス的な価値も低くなる．場合によっては，既存領域の事業に対する侵食であると見なされ，データ提供を拒否されることもある．

こうした状況に陥ったチームは，自分たちが作っているものが使われないことから，時間が経つにつれてモチベーションが下がっていく．アジャイル開発プロセスに参加する面白さはビジネス状況の変化に応じて，新しい機能を開発し，それがユーザに価値をもたらすことである．いくら新しい機能を開発しても，ユーザに価値をもたらさないと，たとえ，技術的に先進性があったとしても，いつかは興味を失っていく．

孤島型パターンになっている場合，必要なのは既存領域側の協力者を発見し，新領域をつなげていくことである．これが次に挙げる出島型パターンである．

3.3　出島型パターン

出島型パターンは，既存領域と新領域を適切な関係性でつなぎ，事業価値を高め，アジャイル開発プロセスが機能している状態を示す．このパターンこそが，エンタープライズアジャイルにとって，もっとも望ましい．

出島型パターンにおいて，新領域のアジャイルチームは以下のような状況にある．

- 達成すべきビジネス課題が明確であり，1年間は活動できる予算がある．
- ビジネス課題に直接かかわる事業部門がアジャイルチームに深く連携している．
- 新領域に割り当てられた要員は，アジャイル開発プロセスの教育を受けたことはあるが実践したことはない．ただし，コーチなどのサポートをつけることは可能．
- 既存領域の企画部門はかかわらない．
- システムのインフラ，リリースプロセスなどは部分的に既存領域にかかわり，既存領域のシステムとサーバ間連携が許可されている．

こうした状況であれば，アジャイル開発プロセスは成果を生むことができる．

3.4　導入成功にむけた確認ポイント

エンタープライズアジャイルをうまく機能させるためには出島型パターンになっている必要がある．そのためには以下の3つのポイントを達成することが重要である．これらが高いレベルで実現できている場合，エンタープライズアジャイルの導入が成功する可能性が高くなる．

なお，ここでは，一般的なアジャイル開発を成功させるための要因については記述しない．

（1）ビジネス成果の事前検証

システム開発が開始される前に，開発システムで実現しようとするアイディアのビジネス成果が検証されている必要がある．

システム開発の稟議決裁時点で説明されたビジネス成果に対し，実際にシステム開発を進める中でビジネス成果が著しく低くなることが判明することがある．これはシステム開発の問題ではなく，そもそものアイディアの問題である．アジャイル開発は，システムがより高いビジネス成果を生み出すための手法であり，アイディアの価値を高めるものではない．むしろ，システム開発が開始された後にアイディアの信憑性が疑われる場合，システム開発を即座に停止すべきである．しかし，これは容易なことではない．そうであれば，本格的にシステム開発される前に，アイディア段階でビジネス成果の確実性について検証を行うべきである．

事前検証では実データを利用し，実際にビジネス成果を生み出せるかどうかを手動で検証する必要がある．AI導入などであれば，実際のデータを利用したプロトタイピングを行い，実ユーザからのフィードバックを得る必要がある．ここで成果が得られないと分かった場合，システム開発を行うべき

ではない．

ただし，事前検証を行ったとしても，その結果を正しく評価できないケースがある．この点についてはレビュアが「ビジネス成果を望めるのか？」という観点で厳しく検証結果を確認する必要がある．

（2）シンプルな意思決定プロセス

システム開発中もビジネス成果の向上にむけた意思決定プロセスがシンプルで明確になっている必要がある．

システム開発を推進する中で，より高いビジネス成果を目指した方針変更や決定がスムーズに行えるようになっている必要がある．この意思決定は組織において尊重され，別部門の協力が得られるようなものでなければならない．たとえばチーム内での意思決定がスムーズであったとしても，その後，複数人のステークホルダーの確認が必要であったりする場合，意思決定プロセスとしてはシンプルではないといえる．

（3）技術的な独立性とシステム連携

技術選定の独立性が担保される一方で，既存システムとの自動連携が可能になっている必要性がある．

システム開発や運用において従来ルールに従わなくてはならない場合，これは大きな制約となる．近年のシステム開発において開発スピードを高めるために重要なのは「既存のツール，サービスを利用し，開発しない部分を多くする」ことである．新たなツールやサービスの導入にあたり，従来ルールによって，これが阻害されることは時間やコストのロスになる．このためシステム開発の技術選定に独立性が担保されている必要がある．

一方で，エンタープライズ領域でのビジネス成果を高める場合，既存システムや既存データのの利活用は必須である．前述で前提されたサービスあるいは構築されたシステムに対して既存システムとの自動連携によるデータ授受が達成されることでビジネス成果を上げていくことができる．

4. 事例

筆者のかかわった案件で出島型パターンになっており，適切にアジャイル開発プロセスが導入できたプロジェクトを紹介する．いずれも前述の3つの確認ポイントを高いレベルでクリアしており，アジャイル開発プロセスが正常に機能した．この結果，高いビジネス成果を上げることができた．

4.1 Your FIT 365

三越伊勢丹は，三越伊勢丹グループの子会社であり，関東圏を中心に百貨店および化粧品などの小型店を運営している．売上は6,000億円を超えており，グループ売上は1.2兆円となる．

三越伊勢丹ではパンプスやヒールなどの婦人靴を対象にしたパーソナルフィッティングサービスとして「Your FIT 365」を提供している．このサービスは店頭にて足を3Dスキャンで測定し，そのデータをもとにフィットする靴を提案する．その上で，シューフィッターが候補となる靴から，最適な1足を見つけ出すことができるようになっている．

本サービスのシステム開発は2019年4月に開始され，2019年8月よりサービスが提供されている．スクラム開発プロセスを採用し，プロダクトオーナ1名，プロダクトオーナ代理1名，開発チーム5名（時期によって変動あり）によって実施された．

システム開発の推進にあたってはコンサルティング企業にトレーニングおよびコーチング，さらに技術支援が依頼された[8]．

（1）ビジネス成果の事前検証

本サービスの肝となるのは3Dスキャンの測定結果の正確さと，そのデータをもとにしたフィットする靴の検索精度にある．この部分には足と靴のフィッティング情報を提供する外部サービスを利用している．

このサービスの選定は婦人靴売場が行った．システム開発の約2年前より，複数の計測機を試用し，シューフィッターが実用性を確認していた．その結果，今回採用にあたったサービスが実用的であると判断をしていた．

また，この外部サービスには簡易的なフィッティング提案機能が付属しているため，この機能を利用して特定ブランドでのビジネス検証を行った．結果として，売上向上が見込めることを確認していた．

この時点で売場のスタッフがサービス企画書を作成し，事業部門長の確認を経て，情報システム部門に開発を依頼している．

なお，実サービスにおいては，業務プロセスの中でビジネス効果を高めるために，外部サービスのフィッティング提案機能は利用せず，独自に提案機能を構築している．

（2）シンプルな意思決定プロセス

本サービスにおいてビジネス成果について的確に意思決定できるのは靴売場である．そこで前述のサービス企画書を作成したスタッフを情報システム部門に異動させ，プロダクトオーナとした．その際，サービスの機能の仕様決定はプロダクトオーナに一任された．また，年間の活動予算は開発チームを維持することを目的として決定され，その際はビジネス上のKPI（売上，計測人数）のみが指定され，機能一覧は概要レベルでしか示されなかった．

プロジェクト開始後，靴売場の責任者，商品仕入担当，シューフィッターなどを集めたワークショップを開催し，本サービスの意図や業務プロセスについて整理を行った．この結果，プロダクトオーナ，靴売場のメンバ，開発チームは高いレベルで目的を共有することができた．その後もプロダクトオーナは売場スタッフと綿密な連携をとりながら業務プロセス設計，画面設計，システム機能設計の判断を行った．プロダクトオーナはサービス開始後も売場に通い，意見を収集し，機能改善を行っている．

（3）技術的な独立性とシステム連携

本サービスの立ち上げにあたって，基本的には技術選定はチームに任された．一方で，要件として会員情報，商品情報など基幹システムとの連携が強く求められていた．

従来より，三越伊勢丹グループでは基幹システムとDX関連システムを連携するための取り組みが実施されていた．この結果，基幹システムの機能をAPI（Application Programming Interface）化して公開するAPI基盤が存在した．本取り組みにおいても，この基盤を利用し，既存APIの利用，および，APIの追加開発を実施した．

また，本サービスがAPIを利用するには，いくつかのセキュリティ要件をクリアする必要があった．ただ，本サービスの採用した技術に起因し，従来の手法が適用できない部分があった．これに対して，API基盤を担当するメンバが中心となって，基幹システム連携を可能にするための調整を積極的に行い，新たな手法の整備を行うことで短期間でセキュリティ要件をクリアすることができた．

この結果，技術的な独立性を保ちながら，基幹システムとの自動連携を実現することができた．

4.2 マーケティングシステム

A社は出版，オンラインメディア，セミナーを運営するマルチメディア企業である．A社では歴史的経緯から事業を追加するたびにシステムを増築してきたため，事業単位でシステムが分割されていた．このため，ある顧客がA社とどういった契約を結んでいるのかを横断的に把握するのが難しい状況にあった．そこで既存システムから顧客に関する情報を収集し，その結果を用いてマーケティングを行うためのデータ分析システムの開発を行うこととなった．

（1）ビジネス成果の事前検証

システム開発にあたり，開発業者を選定するためにRFP（Request for Proposal）作成を実施した．このRFP作成にあたって，外部コンサルが3カ月間をかけて既存システムの棚卸しを行い，収集すべき顧客データについての整理，さらに，データ分析後のマーケティングプロセスについての方針を作成した．

この過程において，データ分析で成果を出せるという確信が得られたものの，一部の顧客データについてはシステム間で紐づけを行うキーが不足しているなどの状況が把握されていた．また，連携対象システムが多く，データ連携するためには既存システム側に追加開発が必要であることも分かった．ただ，そのための投資額やタイミングを明確にすることも困難であった．

そこでRFPの調達要件としてアジャイル開発プロセスの採用を前提にした．開発チームは段階的にデータ収集を進め，早期に，その時点で集まったデータで有効なマーケティング施策を行うことと定めた．

結果として，開発チームの規模を増減させながら数年間にわたって開発を継続している．なお，開発開始から3年間の開発費用は，当初予算の半分以下となった．

（2）シンプルな意思決定プロセス

RFPではシステム開発において月1回担当役員への報告会を実施することを定めていた．この報告会では開発チームは1カ月間の活動成果を発表し，その次に取り組むべきことを明確にする必要があった．このためチームは早期からビジネス成果を意識して活動した．

プロダクトオーナであるA社メンバは，開発チームが収集できた顧客データを理解し，そのデータを利用して成果が出ると思われる事業部門に働きかけをし，案件を作り出して実施した．その結果，開発開始から3カ月目には，ある案件で従来手法よりも高い成果を上げることができた．なお，この際は自動連携などは実現できないため，手作業で連携し，データ分析ツールで処理をしていた．

このようにチームの活動目的を「システム開発」とはせず「事業部門が成果を出すこと」と定義したことによって，担当役員は，本システムがもたらす成果を1カ月単位で把握することが可能になった．このため，担当役員も月1回の報告会の中でも事業部門や既存システムチームへの指示をその場で判断し，社内での本システムの位置付けを広げていった．

（3）技術的な独立性とシステム連携

RFPにおいて，本システムの技術選定は開発業者に任された．この結果，既存システムでは利用していなかったオープンソース製品や商用製品を組み合わせた構成が選定された．これらの製品について開発業者はノウハウを持っており，開発効率は高かった．

また，本システムは既存システム群からデータを受領することを前提にしていた．ただし，既存連携を行うためにはデータ管理についての規約をクリアする必要があった．これについてはデータ暗号化などの手法によって段階的に解決していった．

4.3 考察

いずれの事例においても，3つの観点に対する取り組みは異なるが，出島型パターンになっている．

まず，ビジネス成果の事前検証である．Your FIT 365では，システム開発の2年前から事業部門側が測定サービスの検証や，そのサービスを用いたビジネス効果検証を行っていた．一方，マーケティングシステムにおいてはシステム開発の3カ月前からシステムやデータの整理を行い，見込まれるビジネス成果について検証を行っていた．いずれもシステム開発が開始される時点で，どのようにしたらビジネス成果が生まれることが分かっていたため，数カ月で成果を示すことができた．

次にシンプルな意思決定プロセスである．Your FIT 365では現場への権限移譲が徹底され，プロダクトオーナと売場が直接コミュニケーションをとり，判断を行った．一方で，マーケティングシステムでは役員を含めた意思決定プロセスが整備され，月1回，判断が行えるようにした．マーケティングシステムは利害関係者が多く，現場レベルだけでは判断が行いにくいことから，役員判断が入ることで無駄な調整が省かれた．いずれにしても重要なのは，役員はビジネス成果を重視し，システム開発の中身には口を出していない点である．

最後に技術的な独立性とシステム連携である．上記のようにビジネス成果を目標にしていても，それがスピーディーに実現されなくては意味がない．いずれに事例においても採用すべき技術は開発チームが選定し，高い生産性を実現している．一方で，既存システムとの連携においては，既存システム側に働きかけをできる人物がいて，連携を実現した．Your FIT 365の場合はAPI基盤の担当メンバであり，後者は担当役員である．

5. おわりに

冒頭に述べたように，アジャイル開発のメリットはビジネスの状況に応じて，どのような機能を開発するのか，という意思決定が適切に行えることである．そのため，開発プロセス内で，開発すべき機能の調整を行うために，定期的な計画立案とレビューを繰り返す．

エンタープライズアジャイルにおいて問題になるのは，稟議決裁時点で決定された機能が目標となり，せっかくの調整機能を活かせない状況に陥ることである．

逆に言えば，稟議決裁時点では機能まで細かく定めず，むしろ，システム開発にあわせて調整を行っていくための意思決定プロセスを整備することが重要になる．また，その前提としてビジネス成果の事前検証が行えていることが重要になる．さらに，調整によって決定された機能を素早く実現するためにチームが技術選定を行うべきであるが，一方で，既存システム連携が実現されなくてはならない．こうした点が実現できれば，エンタープライズアジャイルであっても高い成果を上げることができる．

残念ながら，日本の労働生産性は高くない[9]．これらを改善していくためにビジネス成果を目的としたIT活用は必須である．アジャイル開発プロセスは，そのための大きな助力となる．本稿が真にエンタープライズアジャイルが有効に機能するための参考となれば幸いである．

参考文献

1）CollabNet VersionOne編：13th Annual State of Agile Report.
2）Project Management Institute：A Guide to the Project Management Body of Knowledge PMBOK GUIDE - Six Edition.
3）小椋俊秀：ウォーターフォールモデルの起源に関する考察　ウォーターフォールに関する誤解を解く，商学討究，第64巻1号，小樽商科大学，pp.105-135, ISSN 0474-8638 (2013).
4）Royce, W.：Managing the Development of Large Software Systems, Technical Papers of Western Electronic Show and Convention (WesCon) (Aug. 25-28, 1970).
5）アジャイルソフトウェア開発宣言：https://agilemanifesto.org/iso/ja/manifesto.html
6）RACHEL GILLETT：Productivity Hack of The Week, The Two Pizza Approach To Productive Teamwork, Fast Company & Inc. (2014-10-24)
https://www.fastcompany.com/3037542/productivity-hack-of-the-week-the-two-pizza-approach-to-productive-teamwork
7）Rising, L. and Janoff, N. S.：The Scrum Software Development Process for Small Teams (2000).
8）鈴木雄介：三越伊勢丹のデジタルサービスにおけるアジャイル開発の取り組み，エンタープライズアジャイル勉強会資料（2019年11月7日）https://easg.smartcore.jp/C23/file_list
9）日本生産性本部：労働生産性の国際比較 https://www.jpc-net.jp/intl_comparison/

脚注

☆1　https://easg.smartcore.jp/
☆2　http://www.impressrd.jp/news/190419/NP

鈴木雄介（非会員）y.suzuki@graat.co.jp
流通系システム子会社，オンラインマーケティング企業，フリーランス，ソフトウェアハウスなどを経て，ITアーキテクトとして活躍．マイクロサービスとアジャイルのコンサルティングサービスを提供するグロース・アーキテクチャ＆チームス（株） 代表取締役社長．（株）アイムデジタルラボ 取締役，日本Javaユーザーグループ CCC運営委員長．著書にCloud First Architecture 設計ガイド（2016），監訳にソフトウェアアーキテクトが知るべき97のこと（2009）など．
Twitter @yusuke_arclamp
ブログ http://arclamp.hatenablog.com/

採録決定：2020年2月25日
編集担当：藤瀬哲朗（所属）