第6章

適用の際の注意点

「適材適所」の選択がポイント

本章では，各種開発方法論を利用する際のポイントを整理してみます．

Agileの注意点

ここまで見てきたように，Agile開発方法論には数多くの魅力があります．また，Heavyweight開発方法論と比較して，すぐにプロジェクトへ適用できるという印象もあります．少ない投資と準備期間で開発方法論を導入し，効果を得ることができるのは大きな魅力です．

しかし，その反面Agile開発方法論には，明らかに注意点や課題も存在します．“欠点”と書かずに“課題”や“注意点”と書いたのは，Agile開発方法論提唱者たちは，自分たちの方法論を「万能である」とか「すべての開発者に無条件に有効である」とは主張しておらず，方法論適用の事前条件や適応範囲を明確にしているからです．

つまり，事前条件や適応範囲に該当しない場合は，もともと方法論の対象範囲外というとになります．以下では，この事前条件や適応範囲について，確認していきます．

❶経験が重要

Agile開発方法論のチュートリアルや開発事例紹介を，セミナーやカンファレンス等で目にすることがあります．開発事例の考察では，XPを適用した効果が紹介されて，XPの有効性が示されることが多いようです．

XPやその他Agileな開発をすでに実践しているエンジニアの多くは，以前からオブジェクト指向開発などに造詣が深く，ソフトウェア開発経験が豊かな人が多いです．反復型開発の経験もあり，初めての開発分野であっても十分に現実的な開発スケジュールを見積もりするだけのキャリアと技術を備えており，同時にリスクへの対応も心得ています．

このような高い経験値を持つエンジニアにとっては，XPや他のAgile開発方法論が非常に効果的であることは理解できます．Heavyweight開発方法論の詳細なプロセスや技術に従わなくても，自らの経験と知識に基づいて開発プロジェクトがおかれている状況を的確に判断し，適切な作業や対応策を実行できるからです．

しかし，多く一般的なエンジニアに同じことを要求するのは難しいでしょう．

❷技術力（エンジニアリング力）が前提

Agile開発方法論の多くは，伝統的な開発方法論とは違い，具体的な技術事項のノウハウが紹介されているわけではありません．オブジェクト指向やデータベースに関する知識，その他開発に必要なエンジニアリングスキルを身につけていることが前提条件となっています．また，要求定義書

の効果的な作成法，スケジュールの正しい見積もり方法，顧客との交渉力，データベースシステム構築方法などについては，ほとんどのAgile開発方法論では対象外となっています．

このことから，Agile開発方法論が対象としているのは，一定のレベル以上の技術力を備えているものの，より効果的な開発方法やプロジェクト管理方法を知らないために，効率的なソフトウェア開発が行えていないような開発者であるということが伺えます．

開発を通じて技術や知識の吸収が可能とはいっても，それには当然限界があります．たとえば，まともな設計やプログラミングができないエンジニアに，リファクタリングを要求するのはむちゃな話です．

❸無駄なことはしないが，必要なことはする

当然ですが，Agileな開発といっても，ソフトウェア開発にとって必要な作業は行います．意味のない無駄な作業をやめようというのがAgileの趣旨であって，必要な作業までもやめてしまおうというわけではありませんので，この点は注意が必要です．

❹目的はその特定のプロジェクトを成功させること

Agileな開発の目的は，その特定のソフトウェア開発プロジェクトを成功させることです[注1]．つまり，企業や組織全体としての生産性や品質の向上については，当面のスコープから外れています．

この点，たとえばSEI-SW-CMMのモデル[注2]を用いて品質改善を進めていくことになるなら，特定のプロジェクト成功や生産性だけの話では効果はあがりません．したがって，企業あるいは組織としての生産性，品質向上という課題には，Agile開発方法論の持つアプローチとはまったく異なるアプローチが，別途必要になると考えられます．

❺ソフトウェア開発の外注化は未定義

ソフトウェア開発規模が大きくなるにつれて，システムの一部を外注化する場合があります．外注化したソフトウェアの品質は，外注先の企業が責任をもって品質保証すべきですが，発注元にも外注先のソフトウェア開発の品質に関する責任と，品質保証の監視義務は存在します．多くのAgile開発方法論ではこの点に直接言及しておらず，具体的な方法は見えてきません．

したがって，ソフトウェアの一部あるいは多くの部分を外注企業に委託する場合は，品質保証／管理に関する事項については別途組織やプロジェクトで検討する必要があります．

外注化する場合には，地理的に分散して開発することが多いため，この点もAgile方法論を適用する際の課題です．Agile方法論の中には，地理的に分散して開発を行うことに積極的に取り組んでいるものもありますが，作業を外注に委託する場合について直接言及している方法論はないようです．

❻科学的な品質保証／管理も未定義

多くのAgile開発方法論は，明確な品質保証や管理については言及されていない印象を受けます．

徹底したドキュメント作成の割愛による効率重視と，直接対話による開発者間のコミュニケーションによって開発効率の向上が期待できる反面，構成管理作業や開発規模，作業工数のメトリクスを収集し，組織やプロジェクトでデータを利用しようとする場合には，別途何らかのシステマティック

注1）「次のプロジェクトに備える」というのも目的の1つですが，それは2次的なものです（参考文献12）．

注2）参考文献14．

な手法を導入する必要があります．

この場合，メトリクスの再利用性を重視するなら，ある程度作業手順の定義が必要になるかもしれません．経験による開発規模，作業工数の見積もりを完全に否定はしませんが，誤差や担当者による属人性が大きくなるというリスクもあります．

❼国や企業の文化も関係する

アメリカやヨーロッパで誕生し，効果をあげてきた開発方法論，管理プロセスが，日本で必ずしも効果をあげるとは限りません．開発方法論や管理プロセスの有効性を判断する場合には，国や企業の文化も考慮する必要があります．

Agile開発方法論の多くは，開発者たちのメンタルな部分を非常に重視しています．もちろん日本の文化に共通する部分もありますが，基本的にはその方法論が生まれた国や企業の文化を前提としています．適用を検討する際には，この点も十分考慮する必要があるでしょう．

Heavyweightの注意点

ソフトウェア開発が大規模化・複雑化し，開発期間の一層の短縮が求められるにつれ，開発方法論の重要性が注目されるようになりました．Heavyweight開発方法論は，大規模化・複雑化したソフトウェア開発を管理するために導入されています．

ソフトウェアの規模が小さかった20年以上前は，明確な開発方法論やプロセスが存在せず，専ら開発者個々の経験と勘に基づいた開発が行われていました．その後，ソフトウェアの規模が大きくなるにつれて，開発者間のコミュニケーションの爆発が問題となり，プロセス定義による作業手順の明確化と，ドキュメントによるコミュニケーションでこれを管理しようとしたのです．しかし，近年は逆に，この過度な手順化や，多すぎるドキュメントの作成とメンテナンスによる開発工数の圧迫が問題となっています．

①プロセスの理解に負担がかかる

汎用的な方法論を自分たちのプロジェクト用にカスタマイズするためには，採用する開発方法論について十分理解する必要があります．採用する開発方法論が重量級であればあれるほど，明確な作業定義や汎用性を持っている反面，開発方法論自体の学習コストも大きくなります．

②プロセスのカスタマイズ作業は単純ではない

作業が詳細に決められ，成果物や作業間の関係が明確にされているといっても，汎用的な方法論であるため，それらは一定以上の抽象度で定義されています．そのため，それぞれの組織やプロジェクトに適するように内容を明確に定義するには，それ相当のカスタマイズ作業時間が必要となります．

したがって，利用価値の低いドキュメント作成や，形式だけのMeetingの実施を極力避けなければ，オーバーヘッドが大きくなるという問題があります．

③開発では常に同じ作業を繰り返すとは限らない

カスタマイズは，作業，作業成果物および作業責任を割り当てて，担当者を明確に定め，開発作業は定義作業順序に従うことになりますが，プロジェクトを取り巻くビジネスニーズの変化や突発的なリスクの発生であらかじめ定義していた作業手順を実行することが困難な場合があります．また，反復型開発の場合には，開発ライ

フサイクルの時期によって作業のウエイトや作業内容が異なります．

④プロセスに柔軟性が失われるリスクがある

カスタマイズした方法論といっても，実際に開発に適用してみないことには要／不要な作業が見えてこない場合も少なくありません．最初から最適なプロセスを定義することは難しいからです．

また，開発作業はもともと流動的なスケジュールになるのが普通で，事前に定義した作業どおりに進まないことも珍しくありません．事前に詳細なプロセスを定義する方法では，状況の変化に柔軟に対応するのが困難になるというリスクがあります．

Agile開発方法論を提唱する人々の多くが，この問題を指摘しています．

⑤即効性よりも組織成熟度向上に向く

RUPなどの新しいものも含めたHeavyweight開発方法論やCMMの狙いは，ある特定のプロジェクトの成功だけではなく，企業，組織などを含めた全体としての生産性や品質の向上にあります．

したがって，１つのプロジェクトへ導入する場合の即効性だけで，方法論の優劣を判断すべきではありません．１つのプロジェクトだけで効果を期待するのではなく，ある程度のタイムスコープで生産性や品質を考えていく必要があります．

選択の基準

ここからは，何を基準に開発方法論を選択すべきかについて検討します．

プロジェクトの種類や規模に合った選択が重要

あるプロジェクトにとっては，Heavyweight開発方法論の明確な手順と成果物を定めたプロセスは，組織やプロジェクトに一定の基準に従った「秩序」をもたらします．また，定義したプロセスによる開発事例が増えれば，手順や成果物に見直しをかけ，プロセスをより最適化することが可能となります．ソフトウェアの規模や開発工数の見積もりの精度も向上するでしょう．実際に，Heavyweight開発方法論に準じた方法論やプロセスを導入して，効果をあげている企業も少なくありません（40ページのコラム参照）．

一方，Agile開発方法論のような徹底したスピードと効率重視の開発も，変化が早く期間が短い現在のソフトウェア開発では不可欠な考え方です．

Agileの提唱者たちは，従来の開発でHeavyweightプロセスのような予見的開発方法論を利用して効果をあまり実感できなかった経験から，結局そういったHeavyweightなプロセスでは，自分たちの行っている開発はうまくいかないという結論に至りました．したがって，AgileとHeavyweightという2つの方法論は，一見対極にあるような印象を受けます．しかし，単純に2つを比較し，方法論の特徴や優劣をつけることはできないでしょう．

周知の通り，ソフトウェア開発の種類や規模は千差万別で，結果的にソフトウェア開発形態は多岐に渡っています．この多岐に渡るソフトウェア開発の状況が，開発方法論や管理方法論の決定打が存在しない理由でしょう．特定の開発分野や条件を考慮した方法論が存在する一方で，すべてのソフトウェア開発の分野をカバーしようとしている方法論も存在します．

そのため，開発方法論を比較検討して採用する場合には，ソフトウェア開発がお

かれている種類・規模や，課せられている条件等がそれぞれ異なるものであることを考慮しなければ，方法論の特徴や優劣を議論する意味自体があまりないといえます．

基準（1）価値体系による選択

あるプロジェクトに開発方法論を導入しようとする場合の選択基準の1つとして，価値体系に注意を払うのが効果的と考えています．企業，組織，プロジェクトが持っている文化，社風，開発対象システムの特徴に照らし合わせて，同様な価値観に基づく方法論を採用するということが，選択基準の１つになると思います．どのような開発方法論を採用するとしても，まずはその方法論が持つ基本理念に対して理解と共感ができるか，自問してみるとよいでしょう．

代表的なAgile/Heavyweight方法論の価値体系の比較を**表1**にまとめましたので，参考にしてみてください．

基準（2）目的による選択

AgileとHeavyweightでは，そもそも方法論が目指している目的自体がそれぞれ異なっています．したがって，あるソフトウェア開発プロジェクトに方法論を導入する場合，その方法論の目的あるいはゴールを十分考慮して判断する必要があります．

以下では，Heavyweight/Agileそれぞれの方法論の目的について確認します．

●Heavyweight開発方法論の目的

Heavyweightな予見型開発方法論は，なぜ多くの手間と時間をかけて，プロセスをカスタマイズし，企業，組織やプロジェクトに適したプロセスを作り上げようとしているのでしょうか．

それには，企業が「高い生産性」や「優れた品質」を達成するには，１つのプロジェクトだけで良い結果を出せても仕方なく，（理想的には）すべてのプロジェクトで高い生産性，高品質が達成されなければならないという意図が前提にあります．

こうした目的を達成するには，何らかのシステマティックな方法を利用する必要があります．そうでなければ，たとえ１つのプロジェクトで良い結果を出せたとしても，他のプロジェクトで同様の効果を期待することはできません．たまたまうまく行っただけかもしれないからです．

加えて，成功したプロジェクトのノウハウやメトリクスを，他のプロジェクトで活かしづらいということもあります．企業が高い生産性，高品質を達成するには，単発のプロジェクトだけで品質や生産性を検討するのではなく，「企業」「事業所」「部」等の組織としての取り組みが不可欠と考えることは，理にかなっています．

予見型開発方法論は，プロジェクト単位でなく，企業，事業所，部等の“組織”レベルでのベストプラクティスを蓄え，組織としてプロセス化することが目的にあります．これは，見積もりのメトリクスや計画の制度を向上させるには不可欠なアプローチです．組織として定められたプロセスは，実績結果を反映して，より効果的なプロセスとして最適化されていきます．

このため，プロセスや結果を評価する時間軸が，Agileな適応型開発方法論のアプローチとまったく異なります．当然，ある特定のプロジェクトだけで効果を判断するのであれば，Heavyweightな予見的開発方法論はAgileなアプローチに比べて高くつく場合もあるでしょう．

しかし，実際にHeavyweightプロセスを導入して，生産性と品質を向上させて

方法論 / 価値体系の軸	RUP	RAD方法論	XP	Scrum
開発者の役割	重視するがプロセス中心．人は作業の担当と責任が明確化されている．	重視する．人は作業の担当と責任が明確化されている．	非常に重視する．人が中心．プロセスや技法は手段に過ぎない．基本的に分析～実装，テストまで同じ人が担当．	非常に重視する．人やチームが中心．プロセスや技法は手段に過ぎない．
エンジニアリングプロセスの定義	詳細に定義されている．	詳細に定義されている．	特に決められていないが，基本的な作業の流れは存在する．	特に決められていない．
プロジェクト管理プロセスの定義	ワークフローとして手順，技法が詳細に定義されている．	タイムボックス手法として手順，技法が定義されている．	具体的なプロジェクト管理はあまり明確になってない．	基本的な管理作業ワークフローが存在する．Scrum Meeting，Product Planning Meetingなど．
開発者間のコミュニケーションの位置づけ	重視する．	重視する．	口頭によるコミュニケーションを非常に重視する．	口頭によるコミュニケーションを非常に重視する．
ドキュメントなど成果物の位置づけ	重視する．モデルドリブン開発を推奨．	重視する．モデルドリブン開発を推奨．	可能な限り作成しない．成果物は作業のオーバーヘッドを生む．動作するシステムとソースコードが重要．	可能な限り作成しない．成果物は作業のオーバーヘッドを生む．動作するシステムが重要．
技法の位置づけ	作業ワークフローの中に定義されている．	作業ワークフローの中に定義されている．	ペアプログラミング，リファクタリング，テストファースト，計画ゲームなど．	定義していない．状況に応じて選択し，利用する．
再利用性の考慮	重視する．	重視する．	考慮しない．	考慮しない．
開発規模	中規模～大規模．	特に限定していないが，小規模～中規模規模が中心．	小規模．	小規模．但し，中規模以上への適用方があり，実績もある．
対応分野	汎用的．	基本はIT分野，DBが重要となるシステム．他の分野にも適用は可能	汎用的．ただしミッションクリティカルな分野はあまり意識してない．	ITビジネスが中心．ただし他の開発分野でも利用は可能．
Toolの位置づけ	非常に重視する．方法論をサポートするToolの使用を推奨．	非常に重視する．方法論をサポートするToolの使用を推奨．	特に規定しない．ただしテストToolやリファクタリングToolを利用する．	特に規定しない．
変化に対する適応性	考慮している．	考慮している．	非常に意識しており，変化に強い．	非常に意識しており，変化に強い．
方法論のタイムスコープ	１つのプロジェクトから組織の成熟度までカバー．	基本的にプロジェクト単位に注力するが，プロセスをテンプレート化して組織の成熟度までカバー．	基本的にプロジェクト単位．	基本的にプロジェクト単位．
方法論の目的	最適化された手順とメトリクスを用いた生産性と品質向上．	ソフトウェアによるビジネスリエンジニアリングと短期間によるシステム開発．	質の高いシステムを効率良く開発する．システムを期日までに確実に完成させる．	費用対効果に見合う意思決定をして，システムを効率良く開発する．システムを期日までに確実に完成させる．

表1　方法論の価値体系の比較

いる企業も存在します．Agile開発方法論はITビジネスを意識しているものが多く，Agile提唱者の中には，「政府請負型の航空宇宙などのミッションクリティカルな分野では，まったく違ったアプローチが必要になるだろう」と述べている人も存在します．実際に，政府請負型の航空宇宙などの大規模な開発では，Agileとは価値体系が異なるHeavyweightプロセスが多く採用されており，大きな効果をあげています（コラム参照）．

●Agile開発方法論の目的

Agileのような適応型開発方法論は，システムを期日までに完成させることを第一の目的としています．これは，ITビジネスのような極めて変化の早いビジネスでのソフトウェア開発を意識しているものです．

このようなビジネスでのソフトウェア開発では，政府請負型開発に比べて，ソフトウェアへの要求は変更しやすく，開発期間も極めて短くなっています．急激なビジネス変化に対応するためには，開発に時間をかけられないからです．また，ソフトウェア開発会社間の競争も激しく，開発期間やコストを巡る，激しい競争にさらされます．

確かに，こうした状況の中の開発では，Heavyweightな予見型開発方法論とはまったく異なるアプローチが必要でしょう．ITに代表される変化の早いビジネスでは，開発したシステムの陳腐化も早いものです．ビジネスの急激な変化により，まったく新しいサービスをサポートするシステムが必要となることもありますし，システム自体の拡張も求められるからです．このような開発の場合，航空宇宙や防衛産業のような政府請負型開発とは，システムおよびシステム開発の性質が著しく異なってきます．

Agileアライアンスの創設者の一人であるKent Beckは，「今現在のシステム要求に加えて，将来の拡張や再利用を意図してアーキテクチャに多くの初期投資を試みても，ビジネスの変化によっては，実際にはまったく違った拡張やシステムになりうる．つまり，当初かけた初期投資を回収できない状況が多い．ならば，現在要求されていることに集中し，後で必要になることは後で開発すればよい」と述べています． ■

コラム：Heavyweightの成功事例

ここでは，Heavyweight方法論を導入して成功した事例を2つ紹介します．

まず，米国Raytheon（レイソン・エレクトロニック・システムズ（RES）の1987年～1995年の活動報告です．

RESは，対象航空管制，船舶航行管理，ミサイル，水面下の戦闘などのシステムに搭載するソフトウェア開発で，約1200人のソフトウェア技術者を対象としてCMMを導入した品質改善を実施しました．CMMを参考にして，詳細な開発プロセスと管理プロセスを独自に定義したのです．すると，評価2年間で手戻りコストが半減し，8年間で1人あたりの生産性は1.9倍に向上し，欠陥密度は1/4に減少しました（参考文献20）．

もう1つのの事例は，米国Motorolaの1989年～1994年の活動報告です．

政府電子機器部門（GDE）の電子システムの設計および開発で，約1500人（内約350人がソフトウェア技術者）の技術者を対象に，CMMを参考にして，詳細な開発プロセスと管理プロセスを独自に定義しました．

その結果，評価レベル4を達成するとともに，生産性は2.8倍以上に向上し，システムの欠陥は1/8に減少しました（参考文献21）． ■