第1章

Agileな開発とは

「臨機応変」を実践する柔軟なアプローチの台頭

はじめに

近年，ソフトウェアシステムの開発方法論，プロジェクト管理プロセスに注目が集まっています．ソフトウェア開発を成功させるには，優れたエンジニアリング技術だけでなく，効果的な開発方法論や管理プロセスも重要であることが，最近になってようやく開発現場に浸透してきたようです．

今の状況を作ったきっかけはいろいろあると思いますが，最近のようにソフトウェアの品質問題が深刻になっていることや，企業としての生き残りをかけた生産性の向上による経費の削減が，その根底にあると考えられます．ソフトウェア開発で生産性向上やソフトウェアの品質向上を目指し，企業が争ってSEIのSW-CMM[注1]を開発現場に導入しているという記事をよく新聞，雑誌で目にしますが，これなどは良い例かもしれません．

本特集では，最近急速に注目を集めている「Agileなソフトウェア開発」を中心に，ソフトウェア開発方法論／管理プロセスの最近動向を紹介し，代表的なソフトウェア開発方法論／管理プロセスを取り上げて解説します．そして，代表的な開発方法論／管理プロセスを比較し，ソフトウェア開発への哲学&プロセスの相違点，各アプローチのメリット／デメリットを比較していきたいと思います．

Agileとは？

最近，“Agile”という単語をキーワードに挙げたソフトウェア開発方法論が数多く登場してきています．キーワードの“Agile”の意味は，「(adj)[動きが]機敏な，頭の回転が早い」であり，発音は「アジャイル」もしくは「アジャル」となります．私の周りでは「アジャイル」と呼ぶ人が多い気がしますが，好みの問題かもしれません．

ちょっと乱暴ですが，一言で“Agileなソフトウェア開発／プロジェクト管理”の特徴を表現すると，**「状況に合わせて臨機応変に，効率よくソフトウェア開発やプロジェクト管理を行うアプローチ」**といえるでしょう．

“Agile”と呼ばれる開発方法論[注2]の多くは，**「効率の良い開発」**を目的としている点は共通しています．しかし，それ以外にもソフトウェア開発のとらえ方，考え方に多くの共通性が見られます．もちろん各方法論，プロセスは，独自の価値観や哲学を同時に持ち合わせているので，各開発方法論ごとにユニークな内容を持っています．

ところで，ある開発プロセスがAgileであるか否かは，誰がどのような基準で決め

注1）カーネギー・メロン大学にあるSEI（Software Engineering Institute）が1989年に発表した，「組織成熟度モデル」です．ソフトウェア開発組織の“能力”を判断する5段階の尺度を持ちます．

レベル1 ー初期レベル（Initial）
レベル2 ー反復可能なレベル（Repeatable）
レベル3 ー定義されたレベル（Defined）
レベル4- 管理されたレベル（Managed）
レベル5- 最適化するレベル（Optimizing）

注2） Agileな手法やプロセスの提唱者たちは，自分の提唱している理論に対し，「方法論」「プロセス」「アプローチ」などの名前を用いています．ただし，中にはプロセス，方法論と呼ぶには適切でないものもあります．また，OMG（Object Management Group）では「方法論」と「プロセス」は定義が異なりますが，今回の記事では便宜的に「Agile開発方法論」で統一して記述することにします．

Agileソフトウェア開発宣言

我々は，自らAgile開発を実践するとともに，
人々がAgile開発を実践するための支援を通じて，
より優れたソフトウェア開発方法を見つけようとしている．

この活動を通じて，我々は

人と人同士の相互作用を，プロセスやツールよりも
動くソフトウェアを，包括的なドキュメントよりも
顧客との協力を，契約交渉よりも
変化に対応することを，計画に従うことよりも

尊重するに至った．

これは，右側にある項目の価値を認めつつも，
左側にある項目の価値をより一層重視する，ということである．

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

図1　Agileソフトウェア開発宣言（原文はhttp://www.agilemanifesto.org）

るのでしょうか．実は，明確な定義は現在までのところ存在していません．しかも，「Agile開発方法論」とか「Agileな開発」というのは，さしあたっての総称的な表現で，特定のプロセスや手法を指しているわけではありません．今のところ，ある開発方法論提唱者が，「自分の開発方法論はAgileだ」と宣言すれば，それをAgileな方法論と位置づけているようです．

Agileアライアンス

それでは，Agileに関する世間の共通認識の拠り所が何も存在しないかというと，そうでもありません．「Agileアライアンス」

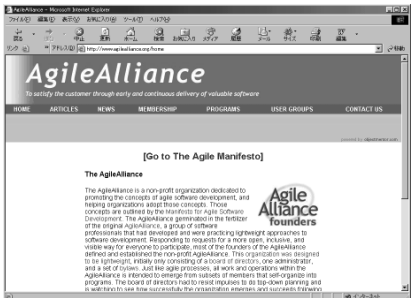
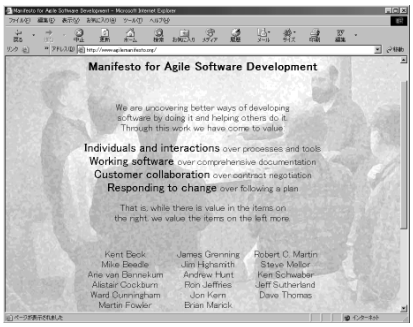


図2　AgileアライアンスのWebサイト
（http://www.agilealliance.org/home）

と呼ばれるAgile開発のアライアンスがアメリカに存在しており，Agileについての「共通理念の宣言」（図1）やAgile開発のための「活動」「メーリングリスト」「記事」「ニュース」などを紹介しています（図2）．



Agileソフトウェア開発宣言のWebサイト（http://www.agilemanifesto.org）

しかし，「Agile開発」のAgileの意味，範囲については，誰も独占的な意見や完全な解釈を与えることができない状況にあります．

Agile開発方法論カテゴリに属する代表的な開発方法論，開発プロセス，管理プロセスの個々の紹介を行う前に，まずはAgileに関することについて整理するところから始めます．そのほうが，種々のAgile開発方法論を知る上で助けになると思います．

Agile出現の経緯と価値体系

Agileな開発を積極的にアピールし，各自の方法論やプロセスを提唱している人々は，従来のソフトウェア開発方法論，プロジェクト管理方法論の生産性や品質効果に対する疑問や，プロセス上の欠陥を感じています．この理由についてはおいおい説明していくことにします．

彼らは，自らの開発経験を通じて，ソフトウェア開発方法論や開発プロセスについて，ある種の考え方（哲学）を持つに至りました．開発方法論やプロセスを導入することは，企業，組織あるいはプロジェクトへ，ある1つの文化を導入することを意味すると考えられますので，方法論やプロセスを選択・導入する際には，導入する組織や個人が，方法論やプロセスが持つ作業への合理性や哲学を正しく理解し，共感していることになります（そのはずです）．このような開発方法論や開発プロセスの持つ哲学や考え方，あるいは方法論やプロセスの理論を下支えするのものの集合を「**価値体系**」と名付け，開発方法論の特徴を分類する方法があります[注3]．

往々にして企業，組織あるいはプロジェクトへ開発方法論や開発プロセスを導入する際に，方法論やプロセスのテクニカルな面ばかりに意識が傾きますが，実際には，まずはじめに方法論やプロセスの基盤となっている考え方に対して理解と共感ができなければ，上手く効果を出せないものです．自分たちの所属する組織とは異なる価値観に基づく方法論を採用した結果，十分な結果を得られなかった，あるいは今ひとつ効果を実感できなかったとしても，それはむしろ当然ともいえるでしょう．

最近のAgile開発方法論の提唱者たちは，ソフトウェア開発経験の中で，伝統的な開発方法論を用いて上手く行かなかったときの反省や経験から，この「価値体系」に対して特別な注意を払うようになりました．この点は，Agile開発方法論が**プロジェクトのメンバのコミュニケーションや作業目的など**，従来の方法論ではあまり取り扱わなかった部分を非常に重視し，数多くの言及をしていることから気がつきます．

たとえば，ソフトウェア開発に参画している関係者の「**人間性**」「**やりがい**」「**メンバ間の協調・競争**」などです．これらはいずれも，伝統的な開発方法論ではあまり言及されていないのに対し，Agile開発方法論では大変重視されており，両者の際立った違いを見せている特徴の1つです．

また，Agile開発方法論は，それぞれが独自の発展経緯を持っていますが，互いに影響も受け合って発展してきています．たとえば，第2章以降で紹介するScrum方法論[注4]，Agile Modeling（AM）[注5]や

注3） SKYVA Internationalのソフトウェア開発者Dirk Riehleが，論文「ASDとXPの価値体系を比較する」（参考文献2『XPエクストリーム・プログラミング検証編』に収録されています．価値体系についてのさらに少し詳しい定義は，こちらの文献を参照してください）で提示した分類方法です．本特集では，このRiehleによる方法に基づいた分類を行っています．
なお，参考文献は第7章の文末にまとめて掲載しています．

注4）参考文献1（『Agile Software Development with Scrum』）．

注5）参考文献5（『Agile Modeling』）．

eXtreme Programming（XP），Adaptive Software Development（ASD）[注6]などです．互いに影響し合えるのは，実はプロセスや方法論が持つ基本的な価値体系が多分に共通しているからです．

方法論によっては，「原則」「価値」「プラクティス」などのような価値体系を前面に押し出しているものもあり，1つの方法論提唱者が他の方法論の価値体系に共感し，自分の方法論にそれらを取り込んでいる例も見受けられます．その一方で，同じ価値体系の方法論やプロセスを補完する意味で用いられる「テクニック」の集合も登場しています．一例を挙げると，Kent BeckのXPと，Scott.W.AmblerのAgile Modelingなどがこれに該当します．

このように，Agile開発方法論を理解するには，この価値体系を最初に理解しておくことが極めて重要であり，近道であると思います．特に日本では，メンタルな部分も重要になる事項に対し，表面的な導入を行う傾向があるので注意が必要です．開発方法論やプロセスを導入する際に，それらが拠って立つ基本理念をきちんと理解せずにテクニカルな部分ばかりを追いかけてしまうと，十分な効果を期待することができないばかりか，かえって混乱を招いてしまう結果になりかねません．

◇ ◇ ◇

なお，Agile開発方法論や管理プロセスについては，「Lightweightプロセス」「適応的開発プロセス」「進化型開発プロセス」など，開発方法論やプロセスを見る視点によって呼び方が異なる場合があり，発言者が議論の焦点に応じて使い分けている場合もあります．ただし，通常は明確な違いを意識しないで用いられることも多いようです．方法論に関する大まかな分類を，表1に示します．

注6）参考文献13（『Adaptive Software Development』）．

開発方法論のモデル

Agile開発方法論は，伝統的な開発方法論とは明らかに異なる価値体系の上に成立しています．Dirk Riehleは，価値体系とは「ソフトウェア開発の根本を支えるものは何か」というテーマに対する信念の体系であると述べています[注7]．これは，**方法論やプロセスの理論を支えている前提事項の集合**と考えてもよいでしょう．

Agile開発方法論の本質は，各方法論ごとの技法を表面的に比較することによってではなく，それぞれの根底に流れる基本理念を整理することによって見えてくると，筆者は考えています．そこで，ここからは価

注7）参考文献2．

Heavyweightプロセス	上流から下流に至るまで，手順，成果物，作業担当者が明確にワークフローとして定義されている開発方法論やプロジェクト管理方法論．
Lightweightプロセス	基本的な手順や成果物が定義されているが，詳細かつ明確には定義されてない開発方法論やプロジェクト管理方法論．本質的な作業に注力し，形式的なドキュメントを重視しない傾向がある．
適応的プロセス	Lightweightプロセスと定義は重なるが，その中でも特に「状況に応じて変化に柔軟に対応すること」を重視している開発方法論．
進化型プロセス	繰り返し開発を通じて，ソフトウェアを評価しながら機能を段階的に実現していく開発方法論．発展型開発と違い，フィードバックによって製品の機能が検証され，ダイナミックに実現されていく．

表1　方法論（プロセス）の大まかな定義

値体系に注目して，Agile開発方法論の整理と理解を試みたいと思います．

Agile開発方法論とそれ以前の伝統的な開発方法論，開発プロセスあるいは管理プロセスは，それぞれ異なる価値体系モデルの基盤を持っています．両者の根底にあるこの相違が，結果としてお互いのプロセス定義を大きく隔てる結果となっています．

以後では，ソフトウェア開発の代表的な価値体系モデルを紹介し，その後モデルの内容についてさらに詳しく紹介したいと思います．繰り返しになりますが，Agile開発方法論の真の目的や狙いを理解するうえで，価値体系を知っておくことは大変重要です．

プロセスを中心に考えるモデル

Leon J. Osterweilは，**「ソフトウェア開発プロセスもソフトウェアである」**[注8]と述べています．その趣旨は，「ソフトウェア開発プロセス（モデル）もプログラミング言語で記述できる」という点にあったのですが，Osterweilの本来の意図するところとは異なり，「コンピュータがアプリケーションを実行するように，開発者や開発に関与する人間がコンピュータと同等にプロセスを実行する必要がある」と歪曲されて解釈されることが多かったようです．

このOsterweilの研究の発表以後，急速にソフトウェア開発プロセスのモデル化の研究が加速しました．その結果，伝統的な従来型ソフトウェア開発方法論の多くは，**「目標としている結果を得るためには，作業を計画し，管理することが必要である．そしてプロセスは管理可能であり，だからこそプロセスから生成される成果に対して有効に作用することができる」**という価値体系の上に形成されており，非常に開発プロセスを意識して中心に位置づけていることがわかってきました．

説明を少し補足すると，プロセスを中心に位置付けるこのモデルでは，開発方法論や管理プロセスは，プロセスの内部（作業，成果物，作業間のフロー，成果物間の依存関係）を詳細に定義し，管理することが可能であり，管理しなければならないと考えているということです．

このモデルにおいては，人間には決められた作業フロー，成果物を計画に沿って実践し，成果を出すことが課せられています．

人を重視して中心に置くモデル

一方，プロセスを中心に置くモデルの対極にあるモデルとして，Manny M.Lehmanの見解[注9]があります．**「ソフトウェア開発は，顧客と開発者による共有された学習体験である」「コンピューティング活動そのものが，根本的に人間活動である」**という立場です[注10]．

このモデルでは，ソフトウェア開発に携わる人間の要求や価値観こそが重要であ

注8）Leon Osterweil, "Software Processes are Software Too", Proceedings, 9th International Conference on Software Engineering (ICSE 9), March 1987, ACM Press, ISBN 0-89791-216-0, pp.2-13

注9）M.M.Lehman, "Process Models, Process Programs, Programming Support - Invited Response To A Keynote Address By Lee Osterweil", Proceedings, 9th International Conference on Software Engineering (ICSE 9), March 1987, ACM Press, ISBN 0-89791-216-0, pp.14-16

注10）1971年に出版されたGerald M. Weinbergの"The Psychology of Computer Programming"（日本語版は『プログラミングの心理学』）や，1987年に出版されたTom DeMarcoの"Peopleware"（日本語版は『ピープルウェア』）に見られるような，「ソフトウェア開発を人間の活動ととらえる」見解の系譜に属する価値観です．

り，それは決して機械やコンピュータのような資源と見なすことはできないと考えています[注11]．

このモデルに基づく開発方法論の多くは，詳細に手順ワークフロー，作成成果物を定義することに懐疑的です．**ソフトウェア開発を取り巻く状況は非常に複雑なため，あらかじめ詳細に手順や開発スケジュールを決めることは不可能**と考えています．**それゆえに，状況の変化に合わせた柔軟な対応が可能なプロセス**が必要であると主張します．

このモデルにおいては，プロセス中心の見解とは異なり，人間が状況を判断し，協力して作業を進めていく重要性が唱われています．さらに「ソフトウェア開発者も人である」という視点から，仕事への「やり甲斐」や「個人のプライド」を重視して，それらを方法論に取り込んでいる点も，このモデルの特徴です．

対照的な2つのモデル

以上の2つのモデルでは，**「ソフトウェア開発に携わる人間をどのようにとらえるか」**についての見解が対極にあります．したがって，どちらのモデルを支持するか，どちらのモデルに重点を置くかによって，定義される開発方法論やプロセスは，まったく違ったものになります．

Agile方法論の価値観

人と作業フローに関する2つのモデルを確認したことで，Agile開発方法論の本質が少し見えてきたと思います．以下では，価値体系が持つ他の各項目（「軸」と呼びます[注12]）をもう少し明確にして，各種Agile開発方法論に共通する基本的な価値観理解を試みます．実際には，下記以外にも軸がありますが，ここでは省略します．

注意点は，現段階で検討されている軸が，価値体系モデル間の比較としてまだまだ今後の研究課題にあるという点です．軸の粒度や軸間の相関関係を含めた整理が，今後行われていくでしょう．

また，Agile開発方法論にとって，すべての軸に平等な価値観の重みが割り当てられるとは限りません．ここでの目的は，Agile開発方法論の比較ではなく，多くの**共通価値観を俯瞰**することにあります．そのため，詳細な問題には立ち入らないことにします．

表2に価値体系の軸をいくつかピックアップしていますが，ここでは軸の紹介と，軸に対してAgile開発方法論に見られる価値観を簡単に記述します．

開発方法論の動機

なぜその方法論が必要なのかの理由（誕生の経緯）は，個々のAgile開発方法論ごとに異なります．しかし，伝統的な開発方法論／プロセスが自分たちの開発経験から

注11，注12）参考文献2.

価値体系の軸の一例		
方法論の動機	開発者の役割	プロジェクト管理者の役割
顧客の役割	人間関係の位置づけ	協調関係の位置づけ
競争の位置づけ	コミュニケーションの位置づけ	技法（技術）の位置づけ
開発Tool	開発の目的	満足のいく仕事
製品出荷への考え方	開発規模	人と技術の関係

表2 価値体系の軸の例

上手く機能しない，望む結果が得られないと結論づけている点は共通しています．

その理由については多くのAgile開発方法論が，**「現在のソフトウェア開発を取り巻く市場のニーズは複雑であり，その変化のスピードは予測不可能だから」**と考えています．

それにもかかわらず，伝統的な開発方法論やプロセスが，基本的に市場のニーズの複雑性や変化のスピードを「予測可能」と仮定していることに，Agile開発方法論を提唱あるいは支持する人々は納得できません．伝統的な開発方法論やプロセスのように，**事前にあらかじめプロセス内部の作業を詳細に定義することにどれだけの価値があるのだろうか**，と考えています．

なお一般には，このようにあらかじめプロセス内部の作業を詳細に定義し，開発はその手順に沿って進めるという開発方法論を，「予見的開発プロセス」と呼びます．作業フローや作成する成果物を事前に定義しているからです．

ITビジネスに代表される，過激なほど状況が変化するビジネスでは，“最適化[注13]されたシステム”を目指すことより[注14]，状況に合わせて適応的な開発を目指すことが重要であると考えています．

Agile開発方法論のような，状況に合わせた適応的な開発であっても，プロセス中に開発手順や成果物をまったく定義しないというわけではありません．開発を取り巻く環境や変化の状況に合わせて，開発手順や成果物を柔軟かつ適切に，開発作業の中で対応できるようにしています．

状況に合わせた適応的な開発を目指す理由は，最適化されたシステムの開発が難しいからということばかりでなく，費用対効果の面（Good-Enoughなシステム開発）の考慮や，“収益逓増の理論”[注15]からみてビジネス戦略視点から得策だから，と主張するAgile開発方法論も存在します[注16]．

いずれにしても，Agileの世界は“予見的開発プロセス”の効果に懐疑的です．市場のニーズの複雑性や変化のスピードが速く，ソフトウェア開発を取り巻く環境は非常に流動的だから，事前に作業フローは決められないし，状況に応じて柔軟な作業を実施する必要があると考えています．これを端的に物語っているのが，要求仕様策定です．

「ソフトウェア開発で要求が定まらない」または「頻繁に変更になる」のは今や常識であり，市場のニーズの複雑性や変化に対応することは絶対必要であるとまで述べています．要求が明確でない部分が存在し，頻繁に変更になるならば，そもそも開発計画を詳細に決めることは困難だし，計画したところで“砂上の楼閣”になるという意見です．

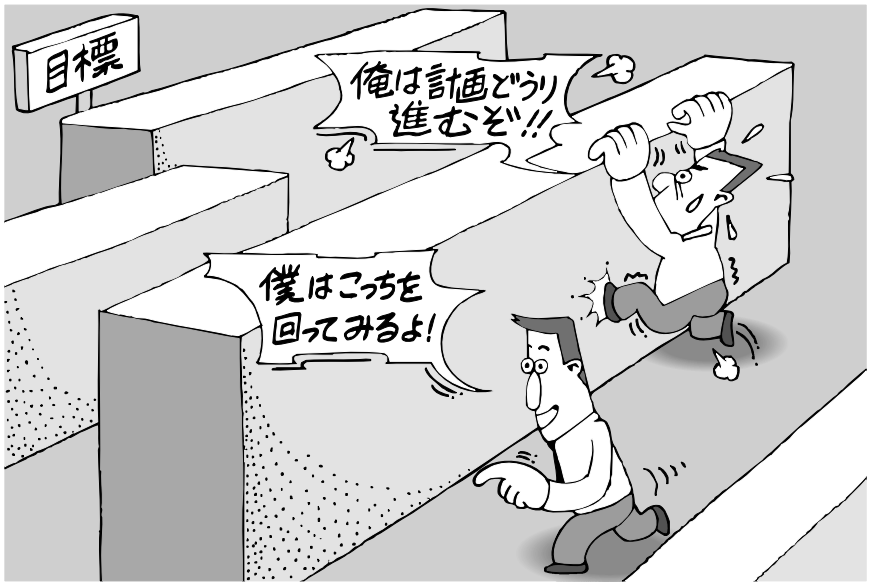
Agile開発方法論を提唱する人々の中には，「要求仕様の変更は，ビジネス変化への対応であり，要求は変更されるべきであ

注13)ここでいう「最適化」とは，「ソフトウェア開発に求められるすべての要求事項に理想的な成果を出すこと」というくらいに考えてよいでしょう．

注14)それを実現するのは，実際にはかなり困難です．

注15)最初は収益がそれほど増加しませんが，「クリティカル・マス」と呼ばれる一定の閾値を超えると，収益が急激に増えることです．収益逓増の法則が成り立つ世界では，「一人勝ち」か「完敗」のどちらかといわれています．

注16)参考文献2．



る」と考える人もいます．そうでなければ，開発されたシステムは，ビジネスニーズに合わず利用されないまま放置されるから，というのがその理由です．

技法（技術）への姿勢

技術に対する姿勢は，Agile開発方法論によって多少違いが存在するようです．Kent BeckのXP（eXtreme Programming）は，「ペアプログラミング」「テストファースト」「リファクタリング」などの技法（技術）を駆使して開発を進めることを意図しています．一方，技法（技術）に関してはほとんど言及していない方法論も存在し，状況に合わせて適切なものを利用すればよいという考えのものもあります．

Scott.W.AmblerのAgile Modeling（AM）は，開発方法論やプロセスではありませんが[注17]，Agile開発方法論，プロセスで利用可能なモデリングテクニックなどの技法（技術）を紹介しています．AMの基盤には，Agile開発方法論の価値体系モデルが成立しています．

開発人数&開発規模の範囲

開発人数については，少人数向きとしているAgile開発方法論が多いです．特に少人数であること自体の積極的な理由はありませんが，開発メンバが“同じ仕事場”で作業し，“十分なコミュニケーション”が不可欠であるとするAgile開発方法論が多く，必然的にこの不可欠要素が今のところ開発規模を少人数に限定していると考えてよいでしょう．

ただし，少人数といっても，その範囲には方法論によってある程度の開きがあります．XPは10人くらいが最も適切なようですし，Scrumは1チームが3人以上7人までで，8人以上の場合はチームを複数作成することを推奨しています．その一方で，50人位で実践されているAgile開発方法論も存在します．

注17）Ambler本人も，Agile Modlingについて，開発方法論やプロセスであるとは説明していません．

なお，現在は多くのAgile開発方法論を，大規模に，かつ分散させた開発条件が積極的に試みられています[注18]．

人の役割／人と技術の関係

ソフトウェア開発における，人間の役割は何でしょうか？

Agileの価値体系では，“人”に開発上の役割や単なるマンパワー以上の価値を置いています．つまり，**人が開発の中心にある**と考えているのです．人には，興味ある仕事や，実りある成果が存在する仕事を与えられるべきであると考えています．**プロセスや技術は人が利用するものであり，プロセスや技術が人を利用するわけではない**のです．

しかし，伝統的な開発方法論の多くは，ソフトウェア開発を人間の活動とはとらえていないのが普通です．開発手順や成果物がまず先にあり，開発者はあたかもそれらを担当する機械のような存在であるかのごとく位置づけられている，という印象さえ受ける場合もあります．

実際にはそこまで極端ではありませんが，いずれにしても，ソフトウェア開発に対する基本的なとらえ方が，Agileのそれとは異なっています．

ソフトウェア開発の目的

ソフトウェア開発を人間的活動と位置づけ，コミュニティやコミュニケーションを重視するAgile方法論の中には，ソフトウェア開発の目的について，単にビジネス的な成功だけではないとしているものも少なくありません．このあたりの価値観から，Agile開発方法論が“人を中心”に位置づける，最も特徴的な一面が伺えます．

たとえばXPは，「ビジネスの目的を達成することを重視」している一方で，「開発者のやり甲斐」なども非常に重視しています．後述するASD（Adaptive Software Development）は，「ビジネスゴールを達成すること」だけでなく，「組織の存続と発展」を目的に入れています．

競争の位置づけ

ソフトウェア開発は，チームで作業します．複数の人が集まって作業をすれば，そこには協調関係が生まれるのと同時に，“競争”も生まれます．この競争をどのように取り扱うか，利用するか，避けるかは，開発方法論によって異なります．

伝統的な開発方法論の多くは，人を中心に位置づけておらず，コミュニティやコミュニケーションを重視していないため，“協調”“競争”などを取り扱っていません．

これに対し，Agile方法論では，プログラミング作業などにおける人間の心理的要素を大変重視しています．ソフトウェア開発といえども人の活動である以上，開発者の心理的要素が与える影響は大きいと考えているからです．■

注18）参考文献2．