欧米で普及するアジャイル開発

平田 正

IT（情報技術）システムがビジネス上の差別化要因になっている現在、要件が激しく変化する戦略的アプリケーションを開発する手法として、「アジャイル開発」が注目されている。特に欧米では、アジャイル開発はすでに従来のウォーターフォール型開発を凌駕して、最も一般的な開発方法論になっている。日本のシステム開発環境でこのアジャイル開発を成功させるためには、①価値観の転換を同時に図ること、②事前にすべてを計画せず、アーキテクチャーを徐々に成長させていくこと、③契約によって開発成果物を固定しないこと――の3点がポイントとなる。

経営スピードに追い付けない情報システム

企業が情報システムに求める役割は、経営環境に応じて変化する。業務効率化や事務コスト削減のための情報システム投資がほぼ一巡した今、多くの企業が情報システムに求めているのは、新しいサービスや事業を立ち上げ、それを軌道に乗せるための支援ツールという役割である。これは、情報システムがビジネス上の戦略的要素、あるいは他社との差別化要因になっているということの表れである。

このような情報システムの開発においては、俊敏性や柔軟性が従来の情報システムよりもはるかに強く求められる。情報システム開発の遅延やビジネス環境変化への対応のまずさが競争力を失うことに直結するためである。

しかし、このような俊敏で柔軟な情報システム開発が実現されているケースは少ないのが現状である。

柔軟なアイデアのシステムを俊敏に構築したいという経営の要求に対して、従来の情報システム開発の考え方では、まず詳細な要件定義を固めることが求められ、しかもその構築には1年、2年という期間がかかることも珍しくない。その考え方が間違っているわけではないが、少なくとも経営が求めているスピードにマッチしていないことは確かであろう。

スピーディな情報システム開発のために

今、経営環境の変化とシステム開発スピードの乖離を解消することが期待できるさまざまな技術が登場してきている（図1）。

サーバーやネットワークなどシステム基盤の俊敏な構築や柔軟な拡張を可能にするものとしては、クラウドコンピューティングがある。また、ERP（統合基幹業務システム）のように比較的変化が少ないアプリケーションソフト（以下、アプリケーション）は、ソフトウェアパッケージやSaaS（Software as a Service：ソフトウェアの機能をインターネット上のサービスとして利用する仕組み）によって素早く構築するのが一般的になってきた。さらに、機能を再利用可能なサービスとして実装するSOA（Service Oriented Architecture：サービス指向アーキテクチャー）も、サービス間接続やサービスの自由な組み合わせを容易にすることで、システムの俊敏性や柔軟性の向上に貢献する。

一方、差別化要因となるアプリケーションや、要件が変化し続ける戦略的アプリケーションの開発としては、俊敏な開発手法である「アジャイル開発」が注目されて

いる。

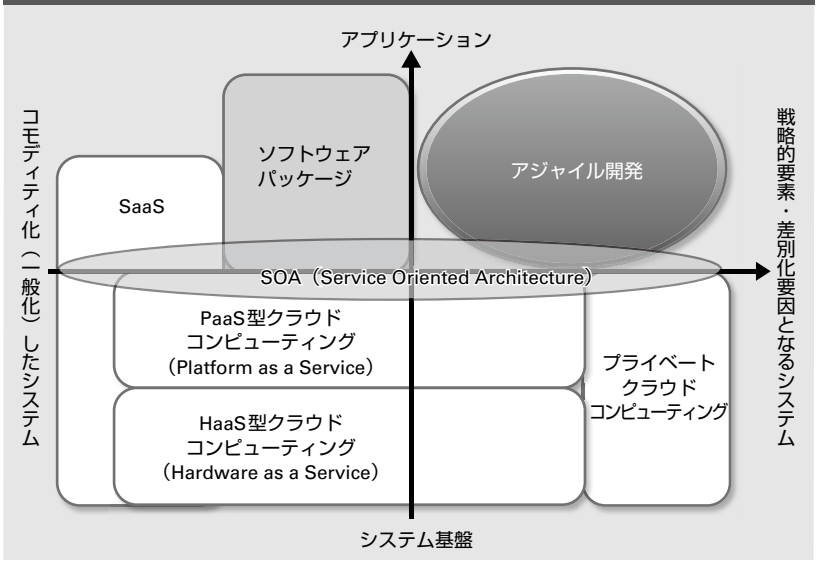
米国で普及するアジャイル開発

アジャイル開発とは、小規模な開発とリリースを繰り返し、フィードバック（評価）と変化を積極的に受け入れる反復型開発手法の一つである。頻繁なリリースにより、情報システム開発全体の進捗状況を可視化できるだけでなく、ビジネス環境の変化に俊敏かつ柔軟に対応できる。

欧米で情報システムがビジネス上の戦略的要素といわれるようになり、変化への適応が問題とされたのは1990年代後半である。2001年ごろにはアジャイル開発がブームとなり始め、05年ごろから、最新ITの採用に積極的であった金融業界を中心に普及していった。

2010年に米国のフォレスターリサーチ（Forrester Research）が同国の情報システム開発プロジェクトで採用されている開発手法を調査したところ、アジャイル系が35%、その他の反復型が21%、ウォーターフォール系（一連の工程を順次完成させていく従来型の手法）が13%であった（Dave West, Jeffrey S. Hammond「The Forrester Wave™：Agile Development Management Tools, Q2 2010」May 5, 2010）。米国の調査会社ガートナーの予測によると、「2012年までにソフトウェア開発プロジェクト全体の80%でアジャイル開発方法論が利用される」という（出典：Gartner「2010年の展望：アジャイルとクラウドがアプリケーション開発に及ぼす影響」T. Murphy他共著、2010年4月15日）。

図1　システム開発スピードを改善する技術



これに対して日本では、アジャイル開発はまだそれほど普及していない。野村総合研究所（NRI）が2009年に実施した「ユーザー企業のIT活用実態調査」では、アジャイル開発を「ほぼ実施」、または「ある程度実施」していると答えた企業は、合わせて13.6%にとどまった。また、情報処理推進機構（IPA）の『ソフトウェア開発データ白書2009』（日経BP出版センター、2009年）によれば、反復型開発を採用したプロジェクトは、全体のわずか2.8%とされている。

アジャイル開発の要点

日本ではなぜ、アジャイル開発があまり採用されていないのだろうか。以下では3つの観点からその理由を考えるとともに、併せてアジャイル開発の要点を解説する。

①価値観の転換

日本の開発現場ではアジャイル

開発を、「ドキュメントを書かない」や「仕様を頻繁に変更する」といった表面的な特徴で捉え、その特徴だけを模倣して失敗しているケースが見受けられる。確かにそれらはアジャイル開発の特徴の一つではあるが、その特徴を活かしつつも、品質が低下しないための考え方や方法を備えているのがアジャイル開発である。そういったアジャイル開発の本質を、関係者が十分に理解しないまま表面的なやり方だけを真似てプロジェクトを進めれば、プロジェクトは混乱するばかりである。

重要なのは、アジャイル開発には情報システムについての価値観の転換があるということである。アジャイル開発では、個々人の問題解決能力を信頼し、可動するソフトウェアを重視し、顧客との信頼関係を築き、変化を柔軟に受け入れることについて、関係者全員が理解し同意する必要がある。

もちろん価値観の転換は簡単ではない。1つの機能を少人数で俊敏に開発するので、組織体制もオフィスのフロアレイアウトもそれに合わせる必要がある。マネジメント手法や人事評価方法、業務プロセス、品質保証プロセス、開発標準などの変更も必要になる。そして何よりも、「アジャイル開発に取り組む」という経営の強い意思も大切である。

②成長するアーキテクチャー

アーキテクチャーに関する固定的な考え方も、アジャイル開発に踏み切れない要因の1つである。

従来のウォーターフォール型開発では、システムのライフサイクルすべてをカバーする堅固なアーキテクチャーを構築することに労力を注ぐ。たとえば「当初はマスター更新は1日1回だが将来のリアルタイム更新に対応できるようにする」、あるいは「サービス時間は9時～17時だが将来に備えて24時間365日のシステムにする」といった話はよく聞かれる。そのこと自体が悪いわけではないが、変化の激しいビジネスアプリケーションに過剰な投資をしているケースは少なくない。

アジャイル開発では、ある程度のアーキテクチャー設計を行えば、プロジェクトの開発期間を通じてアーキテクチャーを徐々に成長させていけばよいと考える。アーキテクチャー設計は、「システム構成の決定」と「非機能要件（性能・信頼性・セキュリティなど機能以外の要件）」の実現に大きく分けられる。

システム構成の決定とは、階層分け（プレゼンテーション、ビジネスロジック、データストアなどの階層とそのインターフェースの決定）のことである。一般的には、一緒に変化すると思われるものを1つにまとめ、ほかから隠すことで変化への対応を容易にする。層間のインターフェースは、ビジネスや技術の変更への適応コストを低く保つようにする。

各階層内に配置されるコンポーネント（プログラムの部品）は、可能なかぎり独立に開発可能で、それぞれの実装の仕方に依存しないことが大切である。この特性によって自動単体テストによる品質保証が可能になる。

このほか、アーキテクチャーとしては、コンポーネントの物理配置、アプリケーション機能の割り当て、永続データの構成（データベースのテーブル構造）などを規定する必要がある。これらは、事前のアーキテクチャー設計で決定することもあれば、アーキテクチャーを成長させていく過程で決定することもある。どこまでを事前のアーキテクチャーとして準備しておくかは、柔軟性や安定性、俊敏性と規律のバランスを考慮して

判断する。

アーキテクチャーのもう1つの要素である非機能要件は、アプリケーション開発上の大きな制約条件である。非機能要件による制約を意識せずに開発されたアプリケーションに対して、後から性能や信頼性を与えることは困難である。したがって、非機能要件は事前のアーキテクチャー設計に含めておく必要がある。

一方で、非機能要件の実現に当たっては、ビジネス環境、プロジェクトの規模や条件、変化の頻度、採用技術の成熟度、システムの耐用年数などを考慮する。これはウォーターフォール型開発のような過剰投資を避けるために必要である。

③契約形態

契約の形態も、日本でアジャイル開発が普及しにくい大きな理由の1つである。日本でよく行われる受託開発は、費用と期間と成果を最初に約束する請負契約が一般的である。しかし、成果を固定した契約形態は、変化を柔軟に受け入れるアジャイル開発には向いていない。請負契約では、受注側が開発に伴うリスクをすべて引き受けることになるため、受注側は事前に詳細な要件定義を要求する。それに加えて、リスクを回避するために費用も期間も過大な見積もりをしがちである。その結果、情報システムは柔軟性を失い、肥大化することになる。

これを防ぐ方法は、成果物ではなく作業に対して支払う「準委任契約」で開発することである。準委任契約は、発注側にとって「最終的に何ができるのかわからない」というリスクがある。しかしこのリスクはアジャイル開発では、早期にかつ頻繁に、実際に動作する成果物が提供されるために軽減される。すべてのリスクを受注側に押し付けてしまう請負契約に対し、準委任契約ではリスクの一部を発注側が引き受けることになるので発注側のリスクコントロールが働く。そのため、より少ないコストでより良い結果を引き出しやすいのである。

請負契約の場合でも、「要件を同等規模の要件で入れ替え可能」というように、約束される成果を固定的にしないための付帯条件を設けることでアジャイル開発を行うことは可能である。

日本ではアジャイル開発が欧米ほど普及していないと述べたが、情報システムをビジネス上の戦略的要素・差別化要因と位置づける先進的な企業を中心に、アジャイル開発への取り組みが徐々に進んできた。NRIでも2004年からアジャイル開発の研究・開発を進めており、アジャイル開発に取り組むユーザー企業を支援している。

『ITソリューションフロンティア』
2010年11月号より転載

平田 正（ひらたただし）
ITアーキテクチャーコンサルティング部上級テクニカルエンジニア