金融基幹系プロジェクトへの アジャイル適用における課題と解決策

東京海上日動システムズ（株）

■ 執筆者Ｐｒｏｆｉｌｅ ■

小作　祐史

1988 年　東京海上システム開発（株）入社
2002 年　プロジェクト推進部門担当
2006 年　抜本システム開発担当部長
2013 年　現在 生保本部・本部長代理

■ 論文要旨 ■

本論文は、保険ビジネスが非常に複雑かつ急速に変化する中で、最新の「メソドロジー」「フレームワーク」「テクノロジー」を駆使することで、生命保険基幹業務システムを極めて短期間で開発した事例のなかから、特に「メソドロジー」であるアジャイル型開発にフォーカスをあて纏めたものである。

昨今の時代は、金融系システムでもお客さまサービスが始まり、スピードやユーザビリティを最優先とする仕組みが必要とされ、アジャイルのニーズが高まった。

金融業界の中でも、特に保険業界を取り巻く環境は、顧客ニーズの変化や IT 技術の進展、法順守、他社システムの動向など変動面が多い。長期間に亘って開発していては、完成時期にニーズが変わってしまうおそれがある。

本事例で取り上げる「あんしんらくらく手続き」も、従前であればウォーターフォール型でシステムを開発し、数年間かけて普及、展開させていったものと思われるが、業界市場ニーズの変化スピードが加速しているため、システムもこれに応じた。

今回、金融業界の中でもいち早くアジャイルにチャレンジし、実際に開発して成果として得られたこと、課題として明らかになったこと、そして今後の開発にアジャイルを適正かつ有効に活用するためには、どんな取り組みが必要なのかを、本論文で報告する。

■ 論文目次 ■

１．はじめに ……《 4》
１．１　当社の概要
１．２　アジャイルにチャレンジした理由

２．「あんしんらくらく手続き」システムの概要 ……《 4》
２．１　「あんしんらくらく手続き」のねらい
２．２　システムの特徴
２．３　採用した新たな技術

３．アジャイルの適合性 ……《 6》
３．１　環境変化によるアジャイルの有効性
３．２　ビジネスの適合性
３．３　アジャイル実行のポイント

４．アジャイルで得られる効果 ……《 9》
４．１　真に必要な機能を提供する
４．２　変更を大歓迎する
４．３　日々進化して成長する

５．アジャイルにおける現状の課題と今後取り組むべき活動 ……《 13》
５．１　コミュニケーションと体制構築
５．２　品質・生産性向上施策と評価方法
５．３　要求タスクの優先順位づけ
５．４　関係者の理解度向上と人材育成
５．５　開発パートナー契約形態
５．６　思想・背景の継承

６．おわりに ……《 23》

■ 図表一覧 ■

図１　「らくらく手続き」システム構成概要 ・・・・・・・・・・・・・・・・・・・・・・・・・・・《　6》
図２　イテレーション単位とスプリント ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・《　8》
図３　アプリケーションオーナー、エンドユーザー、IT部門の関係 ・・・・・・・・《 10》
図４　スクラムチーム図（ビジネス部門とチームを形成する） ・・・・・・・・・・・・《 11》
図５　適切なレビュータイミング ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・《 18》
図６　開発パートナー会社との契約 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・《 22》
図７　部分的にアジャイルを適用する ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・《 23》

表１　品質評価指標 ・・《 17》
表２　アジャイル適否ガイドライン（抜粋） ・・・・・・・・・・・・・・・・・・・・・・・・・《 20》

１．はじめに

１．１　当社の概要

当社は東京海上グループにおける、情報システムの企画、提案、開発、保守運用、活用支援という IT 戦略を一手に担っている。

2012 年度から 3 カ年に亘る中期経営計画において、「Challenge 50 Start!」と銘打った取り組みを掲げ、「価値」「スピード」「コスト」を重視したシステム開発および働き方の変革に取り組んでいる。

そのコンセプトとは、

① システム開発にかかるスピードを倍増、コスト半減を目指して、プロセスモデル、チームモデル、ビジネスモデルを変革する

② 品質とバーターすることなく、今までと次元の違う新たな QCD バランスを作る

③ 創造と IT（SE）の可能性を掛け合わせ、ビジネス価値をあげる

以上の 3 つである。

１．２　アジャイルにチャレンジした理由

金融系基幹システムにアジャイルを適用した事例はあまり聞いたことがない。

金融系というキッチリとした業界では、システム障害は絶対に許されず、β版の仕組みを世に出せない辛さがあり、お試しでサービスを提供できない環境である。十分にテストを重ねて品質精度を高く保証したうえでリリースするのが常識である。

アジャイルが金融系基幹システムの開発に適していないのではなく、簡単に作って触ってもらって改善し成長させていく「アジャイル」の特性が、従来の金融系システムとマッチしていなかっただけと考える。

今回開発した「あんしんらくらく手続き」システムは、開発初期工程までにすべての要件を洗い出し切ることが難しいと考えていた。ユーザビリティに重点をおいたシステムを、スピーディに完成させるには、システムの改善とレベルアップにタイムリーかつ柔軟に対応しなければならなかった。それゆえ、従来のウォーターフォールではなくアジャイルに挑戦すべきと考えた。

２．「あんしんらくらく手続き」システムの概要

２．１　「あんしんらくらく手続き」のねらい

営業担当といえば「カバンにたくさんの資料を詰め込んで走り回るイメージ」がわたしにはある。当社の営業だけではなく、世の中の営業担当者はそのように感じる。あのカバンには仕事で必要なものがたくさん詰まっているはずである。とくに保険営業、代理店業務となれば、パンフレット、申込書をはじめ、紙の資料がとても多くなり、荷物もかさむ。商品は多様化する時代であり、今後は多くの紙が必要になると予想される。カバンをタブ

レットやノートＰＣに持ち替えて、情報として取り出せればどんなにスマートだろうかと。

生命保険を申し込むには、申込書のほか、告知書、意向確認書など、多くの書類を使用する。署名捺印が必要な書類もある。１日に何人もの顧客に接する保険代理店、募集人は、大量の書類を受け取り、整理し保管し、保険会社へ送付する。事務手続きは煩雑を極め、書類の多さは深刻な問題であった。

東京海上日動あんしん生命社では、「営業、代理店事務プロセスをスマートにして、更に業務に専念できるよう改革する」ことを経営目標として掲げた。その施策のひとつとして、紙中心の保険募集と計上事務を、お客さまの面前で、タブレット端末から情報を入力していただくことで、完全ペーパレスを本格的に実行した。

ペーパレス目標は 80%、紙にかかるコストも相当な額が削減できると見込んだ。これが「あんしんらくらく手続き」システムのねらいである。

２．２　システムの特徴

「らくらく手続き」は、保険申込書に記入する従来のスタイルから、タブレット画面に情報を表示して、お客さまが直接操作して手続きする「生命保険新規加入システム」である。試算、引受査定、告知、署名などの機能を有し、健康状態に関する質問に回答すると、その場で契約引き受け可否確認ができ、これひとつで契約手続きが完結する。

ペーパレスだけでなく、物流レス、不備レス、点検レス、照会応答レスと５つのレスを実現させ、利便性向上、事務品質を確保、お客さま情報を保護、災害対策強化、営業、代理店業務専念体制を強化する効果もねらった。そして何より、代理店とお客さまが友好な関係が築けるよう顧客接点強化を狙った。

iOS、Android、Windows の各 OS で利用可能なマルチディバイスに対応したことで、専用端末は不要で、市販されているＰＣ、タブレットが使える。

ＰＣ、タブレットにアプリケーションを保有せず、WEB オンラインで稼働する。プレゼンテーションはオープン系サーバーシステムを、そして既存のビジネスロジックを活用するために、保険料計算をはじめとする基幹機能はメインフレームを使用した。システム構成概要を図１に示す。

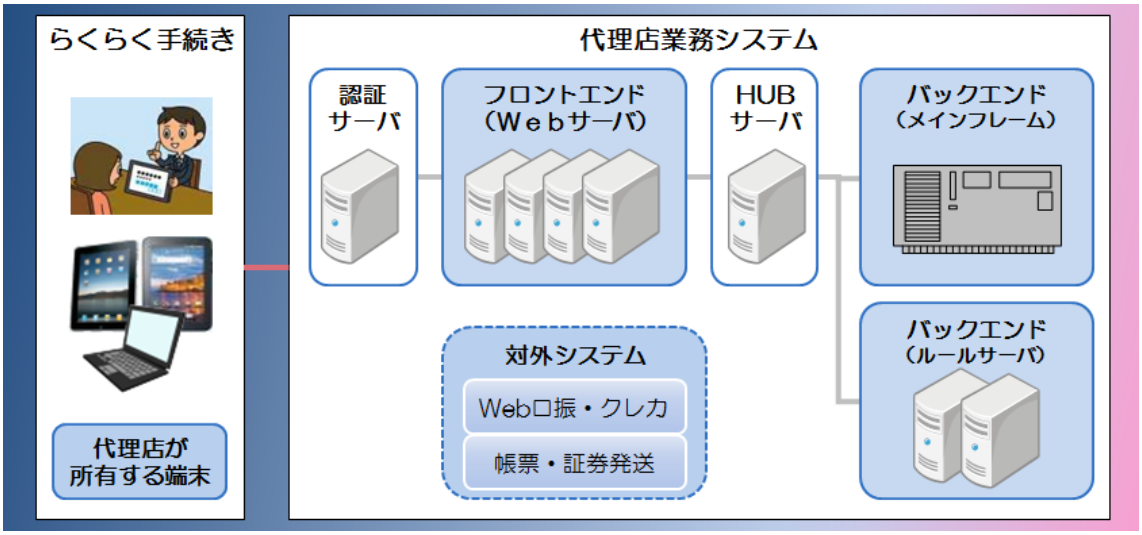

図１ 「らくらく手続き」システム構成概要

２．３　採用した新たな技術

日々進化し続けるテクノロジーと、効果的なメソドロジー、そして効率よく進めるためのフレームワークを自らのものにして、実践しながら開発することをプロジェクト憲章として掲げた。

テクノロジーには HTML5 を採用し、ワンソースでシステムを完成させたことで、マルチディバイスでありながら、柔軟で迅速なシステムメンテナンスを可能とした。

メソドロジーにはアジャイルを、フレームワークにはルールエンジンと継続的インテグレーションツールを取り入れた。ルールエンジンは引き受け可否判断に使用するアンダーライティング定義に適用し、保守性を確保しながら、開発の柔軟性とスピードアップを実現させた。継続的インテグレーションツールは要員効率を上げるために採用して、リグレッションテストを自動化させた。

３．アジャイルの適合性

３．１　環境変化によるアジャイルの有効性

タブレット、スマートフォンが世の中に普及し生活必需品となり、誰もが使いこなす時代になった。ここで動くアプリケーション、いわゆるモバイルアプリは、使い勝手が最優先でなければならないと考える。コンテンツそのものはもちろん重要であるが、マニュアルが無くても簡単かつスムースに操作ができないとすぐに使われなくなる。

迷わずに次に進められ、わかりやすく使い勝手がよいアプリを作るには、従来のウォーターフォール型開発手法では難しい。モバイルの操作性を考えると、アジャイルで作ることが最適である。

「らくらく手続き」は一般の方に使ってもらう、しかも、これからお客さまになっていただく方を対象にしているので、わかりやすい操作だけでなく、見た目もイメージにも印

象的で、デザインを重視したユーザビリティに優れたものを提供しなければいけなかった。

３．２　ビジネスの適合性

厳しい環境変化と価値が多様化する時代には、安価で開発してサービスをスピーディに提供することでビジネス競争力を強化できる。それには多機能製品を作るよりも、開発規模は小さく、機能を限定しても、製品として大きな成果のあるシステムを素早く作り提供することを優先に考えるのが適している。[1]

もしも時代のニーズが変わったとしても、柔軟にシステム変更できるようにしないといけない。柔軟性と俊敏性を高めるとともに、価値・コスト・スピードそれぞれを重視することがアジャイルであり、ビジネスそのものを開発することになる。

３．３　アジャイル実行のポイント

(1) プロジェクト計画、グランドデザインの重要性

開発プロジェクトにおいては、開発の主旨と目的をはじめ、システムが担う役割、体制、予算、抱えるリスク、全体方針を定めて成功裡に導くためのプロジェクト憲章を掲げて、開発担当者はもちろんのこと、上席含めた関係者と共有しプロジェクト計画を立てる。

アジャイルも例外ではなく、計画を立てることはとても重要である。むしろ常に計画を見直して何度も設定しながら実行することが求められる。

「らくらく手続き」においても、プロジェクト立ち上げ当初にグランドデザイン工程を設け、十分に時間をかけて計画した。各イテレーションごとに予算やゴールまでのシナリオ、スケジュールを策定してチーム内で共有した。開発工程に突入してからも、グランドデザイン工程で設定した計画を日々軌道修正しながら進めた。

(2) イテレーション計画と構成

プロジェクト計画と併せてイテレーションも計画する。各イテレーションは比較的短い期間に区切り、設計、製造（開発）、テスト、調査、改善のサイクルを複数回し、内在するリスクと課題を発見し改善させ進めていく。

イテレーション順序は、開発の基盤となるフレームワークを作ること、そして不確定要素が大きなものや変化によるリスクが高いものを、まず最初に着手するとよい。方針転換や再度計画する時間を確保したいからである。早い時期にシステムが動くので、ビジネスユーザーだけでなく、エンドユーザーにも見てもらうことが可能となり、利用者ニーズに近づけることができる。[2]

つぎに、そのシステムのベースとなる正常ケースを計画して、基本機能だけのシステムが稼働できるようにしておく。動くシステムを見せることでアジャイルの特性が活かされる。

その後、異常ケース、複雑なケース、他システムとの連動パターンというように、段階的にシステムを育てるように計画するとよい。

「らくらく手続き」では、図 2 のとおり、1 回のイテレーションを 1.5 カ月、計 5 回計

画した。不確定要素が大きいものとして、開発経験が無かった HTML5 と、ペンタブレットを含めた電子署名機能を、イテレーション初期段階に計画した。加えて、電子署名においては法的に認可されないことも予想され変化によるリスクが高かったので、同様に初期に計画した。

図 2 イテレーション単位とスプリント

各イテレーションでは１週間ごとにスプリントを設定し、各スプリントで目標とするゴールを定めて開発を進めた。

開発した「動くシステム」をテスト環境に毎週反映し、ビジネスユーザーにシステムに触れてもらい要望をあげてもらうことを繰り返した。優先度が高いものから順に、以降のスプリントで改善していった。

(3) アジャイルに適したマインド

アジャイルはチームを作り進める手法である。黙々と机に向かって仕事するスタイルの人間ではうまくいかない。常に会話しながら進めていくので対人関係が苦手な人や、柔軟に考えられない頑固者は、正直向かないかもしれない。

「らくらく手続き」ではアジャイルを進めるうえで、特に大切であると考えた３つのマインドがある。

●真に必要な機能を提供すること
●変更を歓迎すること
●日々進化、成長すること

これらについては、アジャイルで得られる効果として次章にまとめた。

４．アジャイルで得られる効果

アジャイルで得られる効果を最大にするために、「らくらく手続き」の開発では以下をポイントとして進めた。

４．１　真に必要な機能を提供する

(1) コミュニケーションとスクラムチーム結成

アジャイルのプラクティスに「スクラム」がある。数ある中でも多く適用されるメジャーなプラクティスである。スクラムチームを形成するねらいは、確実な意思疎通と迅速な意思決定、そしてスピード感ある反復開発である。[3]

スクラムチームは、ビジネス部門、IT 部門のメンバーで構成され、それぞれに役割がある。

ビジネス部門を統括する立場で、システム化する機能や追加要望を判断する役割を持つアプリケーションオーナー、プロジェクトを統括するプロジェクトマネジャー、打ち合わせをはじめとするスクラムチームの活動を運営するスクラムマスター、開発担当者を指揮するプロジェクトリーダーとその開発担当者でチームを形成する。これがスクラムチームの基本スタイルである。

プロジェクトによっては、品質保証部門やシステム基盤を担当するメンバーをチームメンバーに参加させておくと、組織間における制約に時間をとられることなくスムーズに進めることができる。システムの完成度を評価する品質保証担当者をチームに参加させれば、反復開発時から評価が可能であり、最終局面だけで判断することもなくなる。またシステム基盤を構築する場合や、データ移行などシステム処理が必要な場合には、IT 部門の中でも、それぞれを担当する部門からメンバーを選出してチームの一員にすると話が早い。

(2) アプリケーションオーナーの役割

ビジネス部門が打ち出す戦略や施策において、IT が不可欠になりつつあった 2000 年ごろに、当社では「アプリケーションオーナー制度」を制定した。システム開発とサービス開始後の運営と活用をビジネス部門が責任を持ち、IT 部門だけの仕事ではなく、それぞれの役割を果たすことで、システムの品質と精度を確保していった。

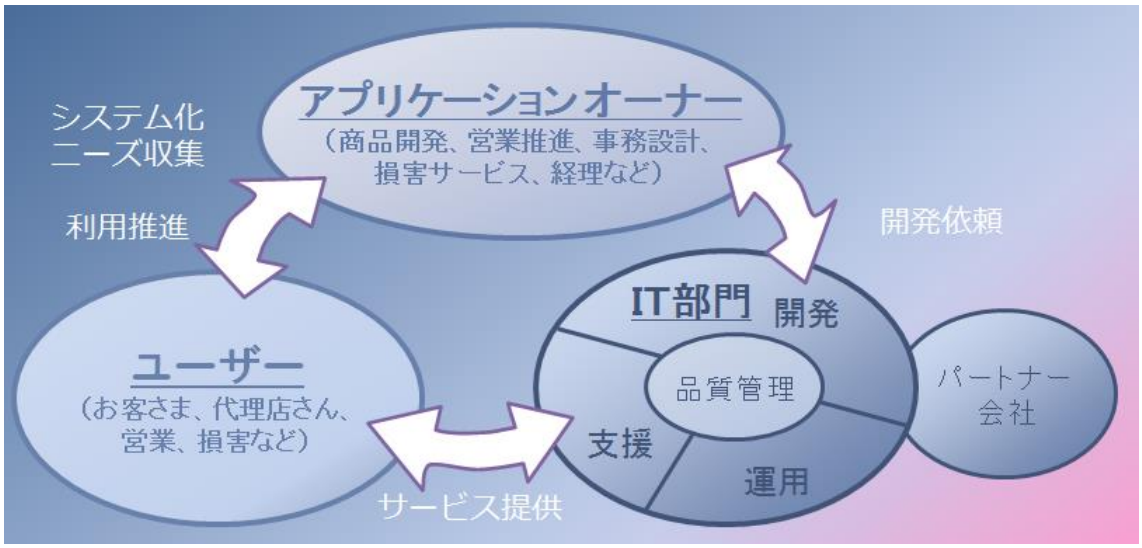

図 3 アプリケーションオーナー、エンドユーザー、IT 部門の関係

アプリケーションオーナー、エンドユーザー、IT 部門との関係は図 3 に示したとおりで、開発プロジェクトは三者間でバランスをとり進める。アプリケーションオーナーの主な役割は以下のとおりである。

○ エンドユーザーの状況、情報リテラシーの現状などを踏まえたシステム化計画を立案する
○ 要件を詳細に確定する
○ 要求仕様どおりか確認（テスト）する
○ システム開発における各工程終了を確認する
○ 利用促進・効果検証測定　など

「らくらく手続き」を検討し始めたころは、このアプリケーションオーナー制度が定着していたので、各人の役割に対する理解者が多く、スクラムチームを結成しやすかった。チーム内の役割も改めて確認することなく定位置についた。

「らくらく手続き」の開発体制（図 4）は、アプリケーションオーナーが 4 名、IT 部門はスクラムマスター2 名をはじめ、設計者、開発者含めて約 40 名でチームを形成した。チームメンバーはそれまで別々の場所で勤務し席があったが、開発にするにあたり、効率よく進めるために専用の開発部屋を用意して常駐スクラムチームを結成した。作ったものを評価して変更を加えるサイクルを如何に速く回していくかがポイントであり、膝詰めで仕事ができる環境は非常に高い効果があった。

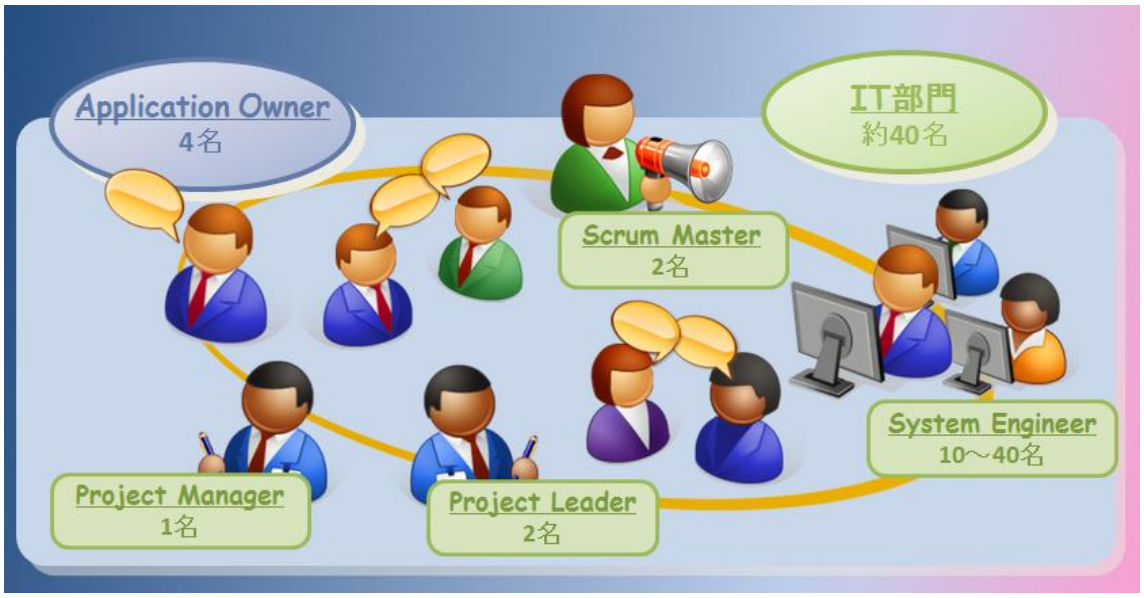

図 4 スクラムチーム図（ビジネス部門とチームを形成する）

(3) タイムリーでやりやすいレビュー

レビューはプロジェクトを成功に導く手段のひとつである。アプリケーション開発におけるレビューは、開発工程の区切りとして実施するケースが多い。プロジェクトの目的、進捗状況やスケジュールを確認して、そのときに考えられる課題とリスクを報告し、有識者から指摘やアドバイスされる場である。

スクラムチームを組み、同じ場所で開発しているのだから、レビューは随時、好きな時にできる。ペアプログラミングなどはプラクティスそのものにレビュー効果がある。

レビューに説明用資料を別途作成するのが通常であるが、アジャイルでは製造が始まってしまえば動くもので評価ができるので、資料を作らないでもレビューができる。

そのものが見られるので、レビュアが誤解することもなく、的確なアドバイスが受けられる。

レビュアだけでなく上席への報告も同じで、方針伺いや進捗状況など定期的に報告する場合も、そのものを見てもらうだけで済む。上司用の報告資料も不要で、作業ロード削減に繋がる。

同様に紙を使うものに、設計書、指示書があり、ものづくりには必要だ。口頭指示や議事録を作成しないなど極力少なくさせ、原則禁じ手と考えられているものも効率化のためなら取り込んでもいいだろう。もしもうまくいかなければ見直せばいいだけである。従来のやり方や概念にとらわれずに合理化しながら進めることがアジャイルでは大事である。

4．2　変更を大歓迎する

(1) 変更を歓迎しないウオーターフォールと歓迎するアジャイル

計画、設計、製造、テストと工程をしっかりと分けて順番に進めていくウオーターフォ

ール型開発の場合、前工程に変更が生じるとその地点に戻らなければいけない。避けるためには変更は取り入れずに決めたことをしっかりと守り進めるのが前提である。

アジャイルは反復させるため、変更が生じる地点を複数回とおる。そのタイミングで変更を反映させればいいので、柔軟に対応できるフレームワークである。

更に計画や設計段階では、要件を大まかに決めているため、変更を加えることでシステムが成長する。変更はシステムを成長させる源泉であり大いに歓迎したい。

ウオーターフォールを主に開発してきたメンバーにはマインドチェンジが必要である。

(2) 徹底的に自動化させ作らないで作る環境を作る

要望を多く取り入れることは、システムも多く変更するということである。如何に手数を少なくして迅速に取り込むかがポイントとなる。

また、システムに変更を加えると、変更した箇所以外で予期しない不具合が出てしまうことは周知の事実である。変更を取り込むこと、テストしてシステムを検証すること、自動化できる作業は徹底して自動化させる取り組みが大事である。

4．3　日々進化して成長する

(1) カイゼン活動によって仕組みが成長する

今ではアジャイルで使われるさまざまなプラクティスが公開されている。しかしながら、これをやれば必ずうまく行くという確証されたものも正解もない。「らくらく手続き」の開発でもいろいろなものを模索しながら進めた。中でもいわゆる「カイゼン」は必須の活動だった。

開発メンバーはこれまでにアジャイルは未経験だった。グランドデザインで進め方を共有し、作業を効率化してムダを排除し、コミュニケーションを活性化させる。チームでカイゼンにカイゼンを重ねることで自分たちの進め方を作りあげた。

各イテレーション、スプリントの完了時に開催する振り返り会議では、問題が多く提起された。課題解決策はチームメンバー全員で論議して決めた。本当に効果があるか疑わしいものもあったが、「とにかく一度やってみよう」、「効果がなければ別な方法を考えよう」とチャレンジ精神を大切にした。

(2) アジャイルによって人が成長する

カイゼン活動によってチームメンバーは考える習慣が身についた。失敗を怖がらずにチャレンジする精神も身についた。解決策が適切でなく効果が得られなかった場合でも、だれかを責めることなく、チーム全員で反省し次の策を考えようと、全員がポジティブ思考に変わっていった。

アジャイルを経験すると、話が上手になることも事実としてあげておく。毎朝のスタンドアップミーティングや、開発時のペアプログラミングによって、会話する機会が多くなったことによるいい影響だろう。人と話しながら即断即決するスタイルだからこそ、単刀直入に自分の意見を発言することが習慣化され、物事を明確に説明できるようになる。その人物キャラクター自体も明るくさせるアジャイルは、適用しなければ明らかに損である。

５．アジャイルにおける現状の課題と今後取り組むべき活動

５．１　コミュニケーションと体制構築

コミュニケーションは人が生きていくうえでとても重要であり、ビジネスともなればなおさらである。アジャイルでも例外ではない。スクラムチームを組み、繰り返しカイゼンを重ねて、システムを育成させていく。メンバーが目指すもの、志を統一できれば更にいいものが出来上がるであろう。製作物以外にも得られることは間違いなく多いはずである。

５．１．１ 現状の課題

(1) ビジネス部門がチーム参加する意義

スクラムチーム内にはそれぞれ役割はあるものの、お互いの立場に踏み込んで論議することが成功への近道である。ポイントはビジネス部門を組織丸ごと本気にさせることであり、理解度を上げるために、トップダウン、ボトムアップ、それぞれのアプローチを実行すべきである。

チームメンバーは制度、理解、有効性を随時共有しないといけない。特にビジネス部門の代表であるアプリケーションオーナーは役割が大きく、スクラムチームに必要不可欠なキーパーソンである。

(2) アプリケーションオーナーの負担感と重み

スクラムチームではコミュニケーション効率を上げるため、意思を統一し、即断、即決しなければいけない。中でもアプリケーションオーナーは、追加要望をコントロールしながら、スピード感もって即断しなければいけないので責任は重い。

(3) スクラムマスターのリーダーシップ

アジャイルでは、プログラミングにかかる時間は短くなるものの、打ち合わせにかける時間は多くなりがちである。打ち合わせ時間を短くさせ、スピーディに効率よく進めるには、スクラムマスターの重要な役割である。

繰り返しながら進めるアジャイルでは、改善はシステムだけでなく、進め方や手段も適宜見直すことがポイントである。

スクラムマスターは状況を的確に判断して、最適なチーム運営ができるようにコーディネイトとファシリテイトしなければならない。

(4) ファシリティによって効率が変わる

「らくらく手続き」では専用部屋を準備してコミュニケーション効果を最大限発揮できる環境とした。

しかし、アジャイルを開始しようとした場合に、人材と同様に専用場所を確保するのは簡単ではない。ましてや継続して利用するのも調整が必要であろう。複数案件を併行して

進める場合や、長期に亘る開発であればなおさら非現実的なものである。スクラムチーム全員が、Face to Face で作業できるファシリティを確保し続けることが難しくなる。

５．１．２ 今後取り組むべき活動

(1) 離れた場所で開発する

離れた場所でアジャイルをすることは、現時点では経験とノウハウが少ないので、何が足りないのか実際に確認してみるのがよいと考えている。どのレベルまでコミュニケーションが可能なのか、コミュニケーションロスがどこから生まれてしまうのか、そして補充できるツールはどんなものが有効なのか、実務から判断したい。

離れた場所でアジャイルができるのであれば、会議室をシェアすることが容易となり、ファシリティ問題は少なからず解決できる。比較的簡単な開発からトライし経験を積めば、将来、オフショア開発、ニアショア開発も展望できるはずである。

離れた場所でのスクラムチームを想像すると、迅速に的確な判断をすべきアプリケーションオーナーにいつでも連絡がとれるような状況を保つことができれば、それほど制約はないと思われる。また最新の進捗と情報が常に確認できるボードを共有できることができれば、１か所に集まらなくてもアジャイルは可能だと考える。ただし、ボードを共有するにはテレビ会議システムが必須となる。コミュニケーションロスを減らすためにも、連絡手段は電話ではなく、姿が見えるテレビ会議システムは無くてはならないはずである。テレビ会議システムに類似した WEB カメラでも効果はありそうである。そうなればソーシャルネットワークを利用して自席でアジャイルも現実なものになるかもしれない。

(2) 提案力向上のための情報収集とツール・サービスを有効利用する

アジャイルに適したプロジェクトに対して、得意とするプラクティスを適用することで十分な効果が期待される。

アジャイル以外にもたくさんの開発手法があり、ツールやサービスも世の中に多く存在する。それぞれが得意としているもの、そして適したプロジェクトを適切に組み合わせることで効果が数倍にもなるであろう。逆に組み合わせを誤ると失敗プロジェクトに近づけてしまう。

これを防止するためにも、現存するツール、サービスの情報収集作業を怠らずに、常に情報、知識として有しておくべきである。

(3) スターターキットでアジャイルを簡単に始める

スクラムチームや体制構築など、アジャイルに取り組むにはハードルが高そうな話をしてしまっているが、個人的には軽くスタートさせて、どんどん肉付けしていけばいいと思っている。スタートさせるために必要最低限なものを洗い出しスターターキットとして整理して、その手順に沿って進めれば、経験者がいなくても誰でもアジャイルに着手できる。まずはスタート台に立って飛び込む勇気をメンバーに与えたい。

５．２　品質・生産性向上施策と評価方法

５．２．１ 現状の課題

(1) システム品質を確保するためのＣＩツール

反復開発によってシステムは日々成長進化していく。検証済みのシステムに悪影響を及ぼしていないか、継続的インテグレーションツール（CI（continuous integration）ツール）を利用して自動的に保証する。自動化しないとすべて手作業となるため、度重ねてシステム改修するアジャイルではとても大きなロードとなってしまう。

「らくらく手続き」では、ビルドから単体テスト、結合テストまでの実行とエラー検出を、jenkins、selenium を組み合わせて自動化させた。毎日夜間に自動でテスト実行し、翌朝にテスト結果をチェックして問題があれば午後に改修する一日のサイクルを作り開発した。

自動化する環境を構築するにも時間を要したことや、自動化によってシステムに負荷がかかり開発環境のレスポンスが悪くなることもあり、ツールを利用するのは簡単ではなかったが、幸いにもプロジェクトの初期段階から準備したのでスムーズにテストを進めることができた。今後アジャイルを推進するには汎用的に常に使えるテスト環境を準備しておくべきであろう。開発環境にツールを常設することで、初期設定ロードも削減できる。

またツールそのものを使うスキルを有する担当者が必要なこと、その人材を増やすことが今後の課題である。

(2) レビュータイミングと質

ウォーターフォール型開発では各開発工程の変わり目で、有識者によるレビューを開催して、成果物となる製品の品質を向上させることが多い。当社ではアプリオーナー制度導入と併せて、レビュー制度も制定し、すべての開発案件を制度に則って進めている。

開発工程が曖昧になりがちなアジャイルは、どのタイミングでレビューすると効果的なのか時間とのバランスから判断して明らかにしないといけない。

同時にレビューの質を高めないといけない。アジャイル独特のレビュー方法があるはずである。カイゼンしながら進めていくアジャイルは、レビュー自体も進化するはずである。

またレビュアの育成も必要であろう。アジャイルを理解しないとレビュアとして機能しない部分もあるであろう。レビュアが開発スピードにブレーキをかけてしまっては本末転倒である。

(3) アジャイルで生産性は向上するのか

開発プロジェクトには目的とねらいがある。作り上げるシステムに対して、何にこだわるのか、何を優先するのか、目的によって手段は変わるはずである。アジャイルはプロジェクトを遂行するための手段であり開発手法のひとつであり、生産性向上のための手法ではない。結果的に生産性が上がったと評価できる場合もあるだろうが、アジャイルを適用したことが主な要因ではないと考える。

ではコスト削減はどうだろうか。これも生産性と同じだと考える。アジャイルを適用したことで開発期間が短くなり、その分だけ委託費が減ることもあるが、同様に主要因では

ない。

生産性を最優先にするのであれば、作るものを詳細まで予め決めておき、指示書を作成したのち一気に作り上げるのが効率的であろう。コスト削減を最優先にするのであれば、実装する機能を減らし、とことんシンプルな仕組みとして開発規模と工数を減らせばいい。

使い勝手やユーザビリティを最優先とするなら、生産性を犠牲にしてでも、多少コストをかけてでも、見た目や操作感にこだわるべきである。

生産性を捨てろ、浪費せよと述べているのではない。生産性は後からついてくるものである。いいシステムが完成すれば、開発にかかった費用もすぐに回収できるはずである。

(4) 第三者による評価は必要か

品質においては、スプリント開発ごとに改善した機能を、動くものを見て検証し保証している。加えて、度重ねてシステムを改善しても、前述した継続的インテグレーションツールを利用して検証し続けているので、機能改善した部分以外も保証できていると考える。

このことから、スクラムチームメンバーは常に品質が保たれていると考えている。そして論理的にも品質保証はできているはずである。

しかし、従来のフレームワークに則り、第三者による評価や具体的な指標から判断しようとすると、このやり方だけでは不十分であり評価判断はできない。

５．２．２ 今後取り組むべき活動

(1) システム品質を適正に評価し判断する

ウオーターフォール型開発の評価スキームとして、信頼度成長曲線をはじめとした数々の評価手法が存在する。永年培った経験から成熟度は高い。

一方、アジャイルは始まったばかりで評価スキームは整備されていないであろう。世の中にも評価するベストプラクティスは少ないと思われる。開発手法が違うので判断基準も違っていて然るべきである。

整備するには実績値が必要となる。アジャイル用評価スキームを今後制定したいが、基準に必要な数値を収集するにはそれなりに時間がかかり、すぐには確立できない。

よって当面は、従来使っていた情報を利用して、アジャイルによる数値のズレを加味すればよいと考える。アジャイルの経験値を上げ、指標となる数値をたくさん収集しないと精度は上げられない。

(2) 第三者が評価できる指標を設定する

当社で開発したシステムは、表１のとおり、８つの項目から品質評価している。

	品質評価項目	評価内容
①	テスト密度	開発規模に対するテストケース数
②	テストケース消化	テスト消化状況と見通し
③	障害密度	開発規模に対する障害発生数
④	障害率	テストケースに対する障害発生数
⑤	収束率	障害摘出目標数と摘出数比較
⑥	仕様変更収束	追加要望数に対する変更対応数
⑦	未解決バグ数	未解決障害対応残数
⑧	ペンディング件数	未解決懸案事案数

表１ 品質評価指標

各項目ともに、過去に開発した実績数値から指標を設けて、客観的に評価できるようにしている。開発はウォーターフォールが中心であり、指標もウォーターフォール型開発の実績から算出して設定している。

評価の対象となる工程が異なること、また障害検知タイミングが異なることから、アジャイル実績を積んで数値を収集し、信頼度が高い指標を設定しなければいけない。

(3) アジャイル版のレビュースキームを制定する

ウオーターフォールで制定したスキームを十分活かした、アジャイルに適したレビューのやり方を確立するとよいと考える。

システムを作り上げるという大目標はウオーターフォールと変わりないので、開発に伴うリスクや品質向上にむけた指摘は従来どおりで構わない。

しかし、プロジェクトの進め方が違うので、レビュータイミングは考慮すべきである（図 5）。併せて、スピードを活かした手法なので、レビュー形式にこだわらずに、インスペクションという形で常に指摘していただくことでも、得られる成果は同じと考える。

むしろタイミングよくタイムリーに実行することで、今まで以上の成果があるはずである。

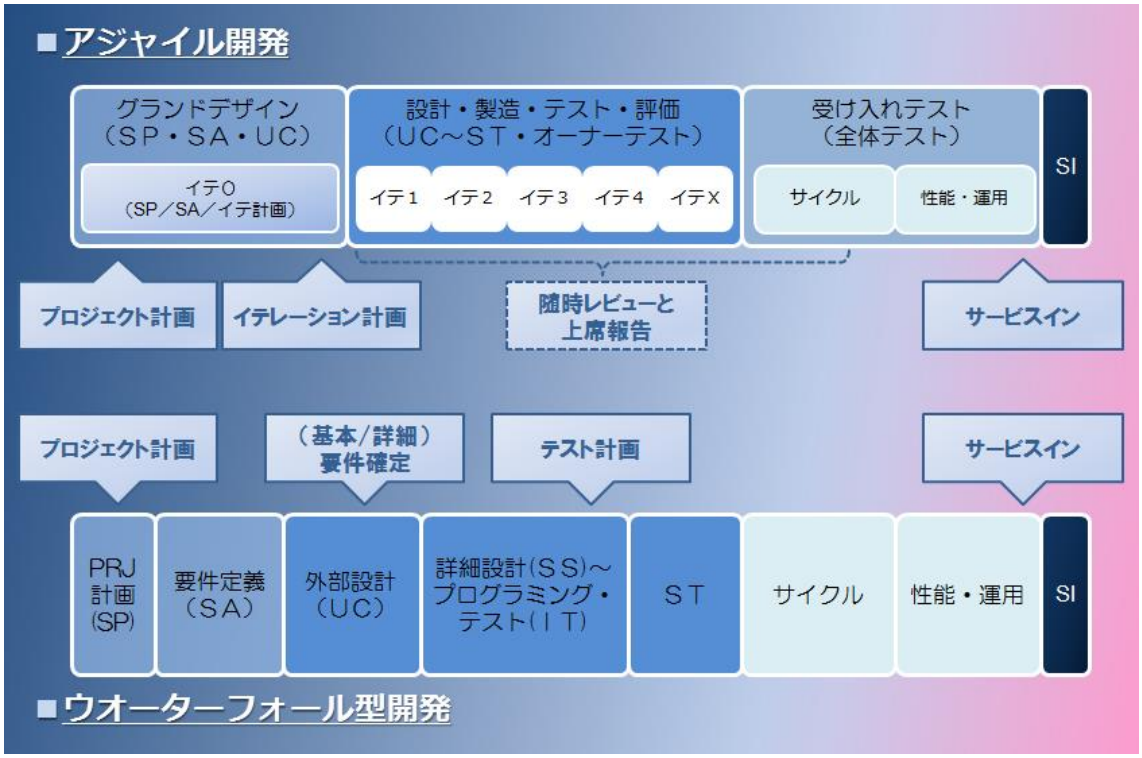

図 5 適切なレビュータイミング

5．3　要求タスクの優先順位づけ

5．3．1 現状の課題

システム開発要件を決める際には、要求を一覧にして優先順位をつけたうえで、開発予算とリソースを考慮して、対象となる機能を決めていく。

アジャイルでは、詳細なシステム仕様と改善要望を開発途中に判断することになる。[4]

優先順位が低いものはそもそも見送るか、以降のイテレーションで反映するか判断する。

優先順位をつけるのはとても重要なものの、チームメンバーの感覚で決めていくので、アナログ的である。人によって判断も異なり、同じ人でもその日の気分や体調によって左右する場合も考えられる。

スクラムチームには、要求をたくさん取り込みたいビジネス部門と、仕組みや機能をたくさん作りたい職人気質の IT 部門がいる。意外にも意気投合するケースが多い。無駄な機能を作りこまず、如何に抑制できるか、どんな基準で判断するか課題として残る。

5．3．2 今後取り組むべき活動

(1) 優先順位を数値化して見える化する

反復開発するゆえに、改善要望は次から次へとあげられる。何を優先すべきか正しく判

断するために基準を作っておくとよい。チームの誰もが評価判断できるように数値化しておくと明確である。

開発が進むと状況が変わってくるので判断基準も見直すことも考えたい。イテレーションごとに設定してもいいだろう。これもアジャイルならではである。

(2) 利用者のニーズをとことん追求する

開発しているシステムを効果的に使ってもらうために、初期段階に設定した要件を適宜見直したい。開発中であろうと利用者が何を望んでいるのか、ニーズが変動していないか、定期的にキャッチして要求タスクに追加すると、真に使える仕組みに成長させることができる。ビジネス部門メンバーだけでなく、エンドユーザーとの関係を継続的にいい状態に保てるよう定期的にコミュニケーションがとれるとよい。

(3) 作りすぎ防止にむけた対策を考える

予算や開発キャパシティ、そしてリソースには限りがあるので、何をどのレベルで限度とするか、それぞれにルールを設定して担当者間で合意しておくとよい。要求タスクの優先順位を数値化した場合には、検討対象とする最下限値を設定するもの効果的である。改善要望がオーバーしてしまった場合は、諦める覚悟も持ってほしい。

イテレーションを残した状態で改善要望が無くなることも想定される。別な要因も考えられるが、思い切って開発を終了させリリースを早める選択肢もあると考える。

５．４　関係者の理解度向上と人材育成

５．４．１ 現状の課題

「らくらく手続き」の実践経験から、変化に迅速に対応できる特性を感じるとともに、効果が得られやすいケースと得にくいケースがあると感じた。システム開発案件には、アジャイルに適したものと適さないないものがある。

関係者に向き不向きを理解してもらうと、今後アジャイルを適用するのが容易になる。それには多くの成功プロジェクトを経験し成果として残していくしかない。

アジャイルに取り組むスクラムチームメンバーの育成も重要な課題である。優柔不断なアプリケーションオーナー、エラーをおそれてチャレンジしないプロジェクトリーダー、ユーザーニーズや世間動向が感じられないプロジェクトマネジャーは、アジャイルの重荷になるだけである。重要なのは、開発プロセス、さまざまなプラクティスだけでなく、人材と言っても過言ではない。人を育成していかないとアジャイルの将来は明るいものではない。

５．４．２ 今後取り組むべき活動

(1)アジャイルを適用すべきか正しく判定する

正しい判断ができるように、基準を設定しておくと失敗は減少する。アジャイルの向き

不向きを整理したガイドライン（表 2）を設定して、誰でも同じ判定ができるようにしておくと、論理的な説明ができる。

		← 向き　不向き →
外的要因	市場ニーズ変革スピードが速い	●
	市場競争が激しい	●
	法的認可により左右する	●
システム	画面イメージや遷移が想像できる	●
	ユーザビリティが優先である	●
ステークホルダー	要求決定者が特定できる	●
	関係者が少数である	●
	専任できる担当者がいる	●
	信頼できるビジネスユーザーがいる	●
	アジャイル理解やチャレンジ精神がある	●

表 2 アジャイル適否ガイドライン（抜粋）

一般的には大規模より小規模な開発、メインフレームよりオープン系、大人数より少人数に向くとされているが、決してそうではないことは「らくらく手続き」から明らかである。

逆にアジャイルには向かないプロジェクトに適用させないことも大事であり、向き不向き両方のアプローチで整理するとよいと考える。「画面イメージや遷移が想像できる」のであれば、製造に着手する前に画面設計をして、ウォーターフォール型開発で進めた方が早く完成する。

(2)アジャイルで必要な人材を育成する

スクラムチームメンバーはそれぞれ重要な役割を担っているが、特にスクラムマスターはアジャイルを進めるうえで必要不可欠な人材である。スクラムマスターにはそれなりの資質が必要と思われるが、どんな人材が適しているのかを見極め、育成スキームを考えたい。

５．５　開発パートナー契約形態

５．５．１ 現状の課題

アジャイルで開発する場合の委託先との契約形態は大きな課題である。現時点では契約の折り合いがつかずに障壁となり、アジャイルが進まないとも言われている。

課題は以下のとおりである。

① ユーザー企業メンバーとパートナーメンバーでスクラムチームを形成し緊密に協力

体制を組んで進めるため、パートナーメンバーの責任範囲が不明確になりがちである。

② システム化する要件が契約時点では未確定で、評価すべき成果物が決まっていない。見積もりも何を基準にすればいいのか不明確である。

③ 開発途中で要望が出されて柔軟に対応するので、決定した事項も変わる。最終的に使われないものも作ってしまう。

基幹システムを新規に開発するとなると、長期間に及んでしまう。一括で請負契約を結ぶには受注者にリスクが生じる。[5]

発注者はいかにコストをかけずに開発できるか、受注者も如何に効率よく進められるか。アジャイルによる恩恵を双方が受けられないと今後の展望が開けない。

５．５．２ 今後取り組むべき活動

システム開発における契約形態は請負と準委任の２つが主流である。準委任では責任範囲が不明確となりリスクが高い。一方で請負では成果物が不明となりこちらもリスクがある。この２つのメリットとデメリットを考慮し、うまく組み合わせて適用するのが現時点ではよいと考える。

開発パートナー会社との契約は、いくつかの工程に分けて契約するのがいいと考える。グランドデザインや、イテレーションをどう進めるかを検討する計画工程と、テスト工程後半からサービスインまでの終盤の工程は準委任契約が望ましい。

「らくらく手続き」では、イテレーション毎に請負契約を結び、ほかは準委任契約とした（図 6）。パートナーの想いと熱意があったことと、アジャイルを推進したい気持ちが当社の考えや意向と一致したことで、契約に関する不都合は少なく、とてもスムーズに進み、システム開発に集中できた。いいパートナーと仕事ができるのは、普段から真摯にお付き合いをしてきたからだと考えている。

これは１つのケースに過ぎず、いくつかの契約パターンを準備しておき、プロジェクトケースによって使い分けるガイドラインがあれば判断に迷わないであろう。

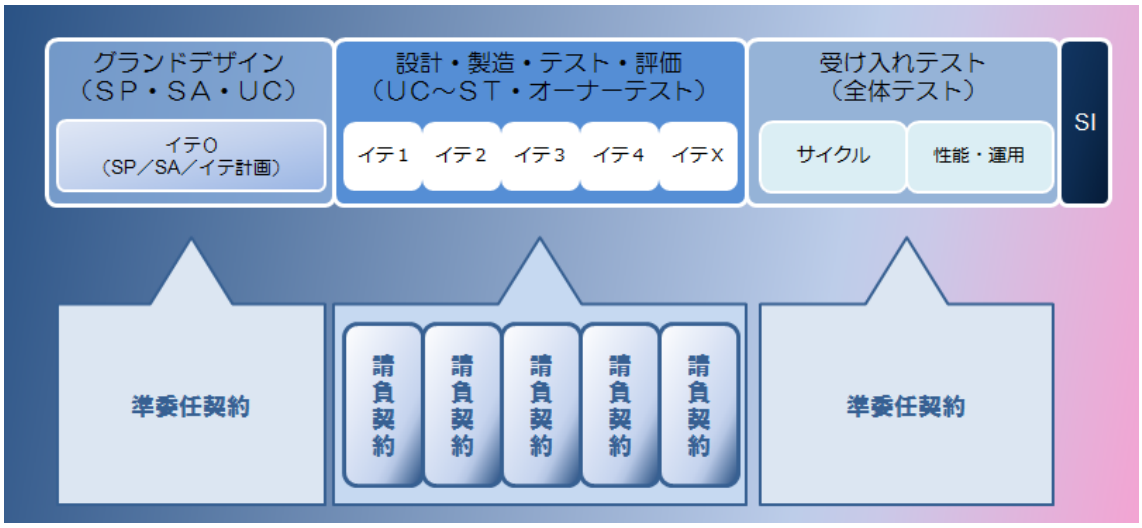

図 6 開発パートナー会社との契約

5．6　思想・背景の継承

5．6．1 現状の課題

反復的に開発していく中で方針や思想が変わったり、新たに生まれたりするのが、アジャイルである。

ウォーターフォール型開発では要件変更は極力取り入れないのが前提である。なぜなら機能が変わることによって、手戻りの要因となり納期が遅れる原因になるからである。加えて変更する内容によっては、システムの基本コンセプトまで変わるおそれがあり、それまでに作り上げたものが無駄になることもあり得る。

一方、アジャイル型開発においては、むしろ「変更を歓迎する」くらいのマインドが必要である。改善要望こそが完成したシステムの価値を上げるものであると考える。改善要望をたくさん出してもらい、その中から真に価値があるものを優先してシステムに反映することが、アジャイルの効果を最大限に発揮させるポイントである。[6]

5．6．2 今後取り組むべき活動

(1) ウォーターフォール型開発との融合

アジャイルは現世代に適した開発手段であると考える。しかしながら日本国内では名前は普及したものの実践する企業はまだまだ少なく、アジャイルに適した開発プロセスやスキームは整備が必要な状態で成熟度はまだまだ低い。ウォーターフォール型開発で長年培い築き上げたフレームワークと比較すると、検討し導入すべきものが多くあり、現時点では不十分であることは明確であり、アジャイルが普及しない理由のひとつにあげられる。

これからアジャイルを導入するもしくは継続推進する企業やチームにおいては、十分に洗練されたウォーターフォール型開発のフレームワークを活かさない手はないと考える。既存の仕組みを十分に活かして、アジャイル適合へのスピードをあげたい。

それには部分的にアジャイルを適用するのがよいと考えている（図 7）。ウオーターフ

ォールとアジャイルのメリットを融合する形である。

図 7 部分的にアジャイルを適用する

(2) 日本式アジャイルとしてスタートさせる

チームプレイは日本文化が得意としているものであり、アジャイル開発はむしろ日本向きであるとも言える。海外から届いた新しい開発手法という考えは捨てたほうがいい。日本独自のスタイルを築くのがよいと考える。

またソフトウエア開発だけでなく幅広い分野で使うことができるであろう。

これからは当社だけの活動ではなく、IT 分野が牽引する形で、日本国内への普及を目指して活動したい。

６．おわりに

「らくらく手続き」はアジャイルで作り上げたからこそ完成したと皆が口を揃える。

逆に言えば、ウォーターフォール型開発ではこの期間では完成できなかった。完成したとしても、形が変わってしまい、使い物にならないものが出来上がったのではないかと思う。

開発中にあげられた改善要望（スプリント開発結果によるフィードバック）は、843 件にものぼり、そのうち 764 件をシステムに反映させた。やり過ぎとも思えるが、アジャイルだからここまでやれたものであり、ビジネスニーズにより相応しいシステムが完成したと考えている。

担当したメンバーが「開発中に心掛けていたこと」、「開発完了後に良かったと感じたこと、うれしかったこと」を抜粋すると次のとおりである。

アジャイルで心掛けていたこと

- ◆ チームメンバーを信じて本音で論議する
- ◆ ネガティブな思考を排除する
- ◆ 使う人の気持ちになりユーザー目線を忘れない
- ◆ 本当に使われる機能だけをシンプルにつくる

開発を終えた感想・よかったこと

- ◆ お互いの考えが理解でき、相手の立場にたった最善策を導き出せた
- ◆ 信頼感が増した、無理難題でもできる自信がついた
- ◆ 「使いやすい」「画面が親しみやすい」と代理店さんから言われた
- ◆ アプリオーナーに「本当に必要なものだけやろう」と言わせることができた

何れもアジャイルを進めるうえで必要なものだが、担当する前からこの意気込みがあったわけではない。アジャイルを進めながら、メンバーが考え、改善し、志も抱いていったものである。これらはすべて入社２～４年目のメンバーが語ったことを加えておく。

最後まで読んでいただきおわかりいただいたであろう。アジャイルはものを作るための有効な手段だけではない。人を作り上げる育成フレームワークであると言っても過言ではない。

今後はアジャイルに携わるメンバーを増やし、アジャイルを推進する活動に力を注ぎたい。

参考文献

[1]Leffingwell D.（著），玉川憲，橘高陸夫，畑英明，藤井智弘，大澤浩二(訳)：“アジャイル開発の本質とスケールアップ”，翔泳社（2010).

[2]Rasmusson J.（著），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).

[3]平鍋健児，野中郁次郎： “アジャイル開発とスクラム 顧客・技術・経営をつなぐ協調的ソフトウェア開発マネジメント”，翔泳社（2013).

[4]David J. Anderson（著），長瀬 嘉秀、永田 渉（訳）： “カンバン ソフトウェア開発の変革”，リックテレコム（2014).

[5]梅本 大祐 ：“アジャイル開発向けモデル契約案について”， IPA SoftwereEngineeringCenter 2012.5.23

[6]英繁雄，奈加健次，平岡嗣晃，前川祐介 ：“ハイブリッドアジャイルの実践”，リックテレコム（2013）.