JISA
招待論文

金融系基幹業務へのアジャイル適用事例

花 宜典[†1]　小作 祐史[†1]

[†1]東京海上日動システムズ（株）

東京海上日動あんしん生命（株）（以下，あんしん生命）では，代理店向けにタブレット端末による保険申込システムを開発した．本システム開発では，高い使用性の実現とサービスの早期提供を目指しアジャイル開発を採用した．エンタープライズシステムのアジャイル開発を実践し，特に「計画の策定と見直し」，「効率化の徹底と継続」，そして「適用目的の明確化」の3点が非常に重要な要素であると考えた．本論文では，アジャイル開発を実践したプロジェクトでの具体的なプラクティスを紹介するとともに，エンタープライズシステムのアジャイル開発における，上記3点の重要性を説明する．

1. はじめに

東京海上日動システムズ（株）では，2012年度から「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．

その背景には，システムの価値が多様化するビジネス環境下で，企画段階とサービス開始時点の狙いがずれてしまうリスクを避け，できるだけ早く，そして安くサービスを提供することで，ビジネス競争力を強化していくことが重要との考えがある．これによって，プロジェクトを着実に実行することよりも，プロジェクトで得られる価値を最大化することを重視することになった．

アジャイル開発は，開発の手戻りを極小化し，業務ニーズに適合したシステムを短期間で開発できる手法として普及しつつある．我々は，このようなアジャイル開発をエンタープライズシステムに導入し，システム開発の変革にチャレンジすることにした．

本論文では，第2章でアジャイルを適用したシステムの概要を説明し，第3章で当プロジェクトで実践したプラクティスを紹介する．第4章でまとめとして，アジャイル開発を行う上で特に重要となった点を議論する．

2. システムの概要と構成

生命保険の申込手続きでは，申込書のほかに告知書，口座振替依頼書，意向確認書，重要事項説明資料，本人確認書など，複数の書類を使用する．これらの書類には署名，捺印が必要なものもあり，生命保険に加入するためには，何回も申込者のサインが必要となっていた．また，一口に生命保険と言っても種類はさまざまで，死亡時に保険金が支払われる「死亡保障」，病気やケガで入院した際に入院給付金が支払われる「医療保障」，がんと診断された場合に一時金が支払われる「がん保障」などがあり，保障内容を分かりやすくするために，契約は保険種類ごとに取り交わされるのが一般的となっている．つまり，「死亡保障」，「医療保障」，「がん保障」の3つの保険に加入する場合は，必要となる書類も増え，捺印や署名の回数も増えることになる．

複数の顧客に応対する保険代理店や保険募集人にとっては，申込書類の多さは深刻な課題となっていた．1日に何人もの顧客から大量の申込書類を受け取り，整理，保管し，さらに保険会社へ送付するといった事務作業は煩雑なものであった．また，申込書や告知書には個人情報やセンシティブ情報が含まれており，取り扱いには細心の注意が必要となっていた．

これらの課題を解決するために，我々は生命保険の申込手続きをペーパレスで完結させることにした．今回開発した「らくらく手続き」は，保険申込手続きのペーパレス化によって，保険代理店におけるバックオフィスでの事務作業や，情報漏えいと情報を紛失するリスクを軽減した．

2.1 「らくらく手続き」の概要

「らくらく手続き」は，従来，書面に印字していた情報を，保険募集人が所有するタブレット端末の画面に表示し，保険契約者，被保険者がこれを直接操作することで，生命保険の申込手続きに必要な作業を画面上で完結させることができるシステムである．具体的には，画面は「保障内容確認」「健康状態に関する告知」「重要事項説明」「保険料振替口座の登録」など全部で6つのブロ

ックに分割されており，順次，保険申込者が画面に沿って保険申込み手続きを続けていき，最後の「意向確認」のブロックの画面上で自署することによって，一連の申込手続きを完結させる．

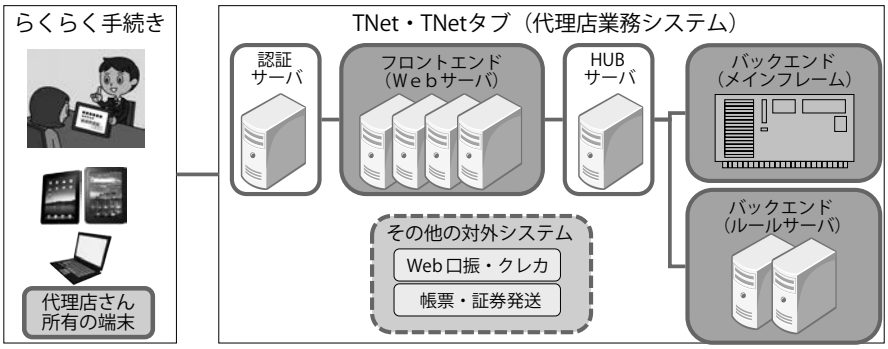

図 1　システム構成

「らくらく手続き」は，iOS，Android，Windowsの各OSで利用可能である．そのため，特定の機種を用意する必要がなく，上記のOSが稼働する端末を保険代理店が所有していれば，すぐに「らくらく手続き」を導入することが可能である[☆1]．最近は，生命保険を取り扱う保険代理店が，複数の保険会社の商品を取り扱うケースが増えており，保険会社ごとに特定の端末を用意することは，保険代理店にとって大きな負担となっていた．「らくらく手続き」のシステム開発では，端末やOSによる制約を可能な限り取り払うことで，システムを普及させることを目指した．また，HTML5を使用することで，1つのソースで各OSへ対応できるようにした．これによって，サービス開始後の状況変化への柔軟性も確保できた．

生命保険に加入するためには，被保険者の健康状態によって加入可否，加入条件等を審査する必要がある．従来の審査では，アンダーライタと呼ばれる医学的知識を有する社員が，送付された個々の申込書類に対して，告知書面を見て査定していたため，判定結果報告までに数日を要することもあった．「らくらく手続き」では，被保険者自身が，保険申込時に画面から必要事項を入力したあとに，加入判断結果を提示できる「自動査定システム」を導入した．これによって，契約成立期間の大幅な短縮と，保険代理店，および保険会社の事務負荷の軽減を実現した．この「自動査定システム」の構築では新たにBRMS（Business Rule Management System）[1]の活用に挑戦した．その結果，アンダーライタがノウハウとして有するルールを「見える化」してBRMSに登録することで，自動査定エンジンをコーディングレスでつくることができるようになった．

2.2　システム構成・開発規模

システムの構成を図1に示す．システムはクライアント側にアプリケーションを配置せずに，Webオンラインの稼働構成とした．この構成では，フロントエンドにはサーバシステムを，バックエンドにはメインフレームとサーバシステムを配置したハイブリット構成となっている．開発規模は約240Kstepであり，開発期間は約1年半であった．

システムリリース後，10カ月間で約7,800代理店に利用され，累計7万件の保険契約が成立している．

3.　アジャイルプラクティス

3.1　アジャイルの採用理由

開発プロセスにアジャイルを採用したのは，以下3つの主な理由による．

(1) 高い使用性の追求

我々はこれまで，主に保険会社ならびに保険代理店向けのシステムを開発してきた．このようなシステムの開発では，その利用者が生命保険に関する知識を有していることを前提としたシステム設計を行うことができた．しかし，「らくらく手続き」は，保険契約者や被保険者など，一般のコンシューマが操作するシステムである．そのため，複雑な生命保険の申込手続きをスムーズに行えるよう，マニュアルがなくても操作しやすい，デザイン性を追求した画面設計にする必要があった．このような設計を行うためには，出来上がった画面を実際に使って，その動きを見ながら，使用性を高める必要があった．

(2) 変動リスクへの対処

生命保険業界でも，業務ニーズの変化や他社との競争が激しくなっている．こうしたビジネス環境において，１年を超えるウォーターフォール型のシステム開発を行っていては，当初決定した要件が陳腐化してしまうリスクがある．また，日々変化する業務ニーズに対して柔軟に対応していけるシステムを構築する必要もあった．変化に柔軟に対応できるシステムは，高い業務適合性を維持し続けることが可能であり，その結果「使われる」システムとなる．

☆1　ただし，各OSやブラウザのバージョンについては一定の基準を設定している．

さらに，法的なリスクも無視できなかった．「らくらく手続き」は，電子署名や生命保険引受可否の自動判定など，開発着手当時，ほかの保険会社が導入していなかった新しい機能を取り込む予定となっていた．しかも，これらの機能が法的に認可されないことも予想された．その場合は，代替手段を検討し，開発内容を変えざるを得ない．システムの柔軟性は，このようなリスクに対処するための要件であった．

(3) スケジュール遅延リスクの軽減

稼働開始時期を遅らせることは，あんしん生命の経営方針に大きな影響を与える可能性があった．特に，プロジェクトの後半で認識齟齬や致命的な課題が発見されることは，稼働開始時期の遅延に直結するリスクファクタである．そのため，早い段階で動くものを作り，内部リリースを繰り返しながら「段階的に育てていく」方式を採用し，致命的な問題や課題を早期に発見して，遅延リスクをできるだけ早い段階で軽減させることを目指した．

3.2 アジャイル開発プロセスの実際

図2に開発スケジュールを示した．プロジェクトは，全体で約1年半であった．このうちはじめの約5カ月間をグランドデザイン工程とし，この間に開発計画を立てた．このグランドデザイン工程の詳細は後述する．グランドデザイン以降の実開発は7.5カ月として計画された．実開発では，ビジネスユーザ，設計者，プログラマから成るスクラム[2]チームを構成し，スクラムやXP(Extreme Programming)[2]等の複数のアジャイル開発プロセスの中から，自分たちの開発内容にフィットするプロセスを組み合わせることを模索しながら開発を進めた．その結果，この間に約30回の内部リリースを行うことができた．

製造工程終了後には約6カ月をかけてサイクルテストや性能・運用テストを実施した．グランドデザイン工程で基本となる要件を抽出した後にイテレーション[3]を開始した点や，サイクルテスト，性能・運用テストを終盤で実施した点はウォーターフォール型の開発概念を取り入れた部分である．

3.2.1 イテレーション計画と実施

設計，製造，テストなど製造工程は，1.5カ月ごとのイテレーションに区切り（図3），これを合計5回実施することで，フレームワーク作りから最後の他システム接続まで，段階的にサポートできるバリエーションを増やしながらシステムを「少しずつ成長」させた．

(1) 反復振り返り[2]

各イテレーション開始前には，スクラムチーム全員で前イテレーションを振り返った．また，ムダな作業を取り除き，効率化することを目的に，スプリント開始時に前スプリントの振り返りを行うことを徹底した．この様子を図4に示した．振り返りを継続することで，スクラムチームメンバ全員が効率化への高い意識を維持することができた．

(2) 開発プロセス

振り返りの後に次のイテレーション計画を実施し，イテレーション完了時にシステムがどのような状態になるのかをイテレーションのゴールとして定義した．ゴール達成に向けた作業計画は開発メンバ全員で共有した．

図3 イテレーションの単位とゴール

図2 開発スケジュール

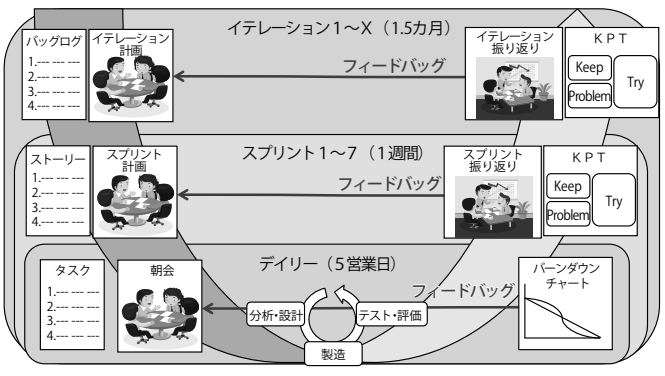

図4 イテレーション，スプリント振り返り

アジャイル開発の事例として，イテレーションを機能単位に分割し，最後に結合するという手法も見られるが，我々は，はじめに「最も基本的なケースの処理が完結できる状態」を作り上げ，そのあとで正常ケース，複雑ケース，エラーケース，他システム連携ケースと段階的に，必要となる機能を肉付けして，システムの信頼性を高めていく方式とした．各イテレーションでは短期間で動くものを作る必要があるために，プログラマはドキュメント抜きでコーディングを実施した．このとき，品質が低下しないようにXPのプラクティスであるペアプログラミング[4]を取り入れた．

(3) リスク管理

イテレーション1では「最も基本的なケースの処理が完結できる状態」を構築すべく，HTML5を使用した電子署名機能など，業務要件として必須であるが技術的なリスクの大きい機能を開発した．これにより，プロジェクトの早い段階で技術リスクを解消させることができ，かつ，最低限の機能が結合され，「動く状態」を維持させていくための基礎を完成させることができた．

3.2.2　ビジネスユーザの協力

図5に，各イテレーションを1週間のスプリントという期間に区切った様子を示した，この1週間の間に，作られたシステムへの改善要望をタイムリーに得るために，スプリントの後にシステムを内部リリースし，ビジネスユーザに操作してもらった．改善要望はシステム開発工数をすぐに見積もった上で，残務リストに書き込み，一覧化した．スプリントを開始する前には，スプリント計画会議を開催し，スプリントで開発すべき内容を，前スプリントで出た改善要望とともに，優先順位付けを行った．この優先順位に従って開発対象をピックアップし，開発計画に組み込んだ．このようなスプリントを繰り返したことで短期的な軌道修正を積み重ねることができた．このようなビジネスユーザの協力を得たことも，業務適合性の高いシステムに育てることができたひとつの要因であると考えられる．

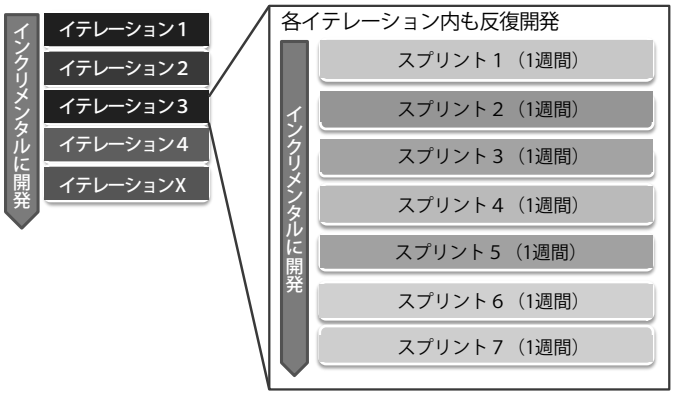

図5　スプリントによる反復開発

3.3　開発のポイント

アジャイル開発はウォーターフォール開発との対比の中で「計画は立てない」と誤解されることもあるが，ここまで述べてきたとおり，アジャイル開発でも計画を立てることは非常に重要であった．アジャイル開発では，常に計画を立てながら実行していくことが求められるのではないだろうか．

(1) グランドデザイン

「らくらく手続き」の開発においても，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，各イテレーションごとの大枠の予算やゴールまでのシナリオ，スケジュールを策定し，ビジネスユーザとも共有した．実際の開発工程に入ってからも，このグランドデザイン工程で策定した計画を日々軌道修正しつつ，常に現在位置を確認しながら開発を進めた．

(2) イテレーション前の準備

スケジュール，予算のほかにも，開発ベンダとの契約方式の合意，前述の各イテレーションの作業範囲や考え方，イテレーション中の日々の過ごし方や最低限のルール，作るシステムの出来姿のイメージをスクラムチームのメンバ全員と共有するなど，実際のイテレーションに入る前に十分な準備を実施した．これによって，開発生産性の著しい低下を防止し，後に大幅な変更が必要となるような作り込みを防ぐ効果があった．このグランドデザイン工程で，プロジェクトの骨格をしっかりと作ったことが，アジャイル開発を成功させる上で非常に大きなファクタであったと言える．

(3) 開発計画の作成と見直し

我々は，状況やニーズの変化にあわせて開発内容を柔軟に変えていったが，計画がなければ，変更要望を制御できず，サービスを開始できなくなってしまっていたかもしれない．

ただし，我々が立てたのは緻密な計画ではなく，ゴール達成に向けたシナリオである．このシナリオをスクラムチーム全員で共有し，短いサイクルでシナリオに沿った計画を見直しながら進めた．我々のアジャイル開発における計画策定は「あとで変更になる」ことを前提に，作り込み過ぎないことがポイントであった．

(4) 計画の実績と見直し

アジャイル開発では，状況の変化にいかに迅速に適応し，軌道修正していくかが成功の鍵となる．常に現

在の状態を把握し，外部要因の変化にも柔軟に対応していかなければならない．我々はこのような開発の現状をリアルタイムに把握するために，バーンダウンチャート（図6）[4]を活用した．

ここでは，イテレーションの計画時に，実施したい作業を時間単位に見積り，生産性（ベロシティ[3]）から，当イテレーション（期間）で開発可能な作業量を割り出し，イテレーション計画を立てた．

一方スクラムチームのメンバは各自に与えられた作業を完了させるために必要な「残り時間」を日々更新することとした．これらの情報からバーンダウンチャートを作成し，常にスクラムチームのメンバ全員に開発の進み具合を見える状態とした．毎日このバーンダウンチャートを見ながら，予定よりも作業が遅れていれば，計画の見直しを検討し，見直しが必要となれば，イテレーションで構築する機能を減らす，または対応内容を変更するようにビジネスユーザと調整した．また逆に予定よりも早く作業が完了しそうな場合は，ビジネスユーザに働きかけ，対応リストから優先度の高い要望の対応を追加することも実施した．

作業が遅延しているからといって，残業でカバーしたり，期間延長したり，品質を犠牲にはしなかった．対応内容，つまり開発のスコープを変えることで作業量を調整することを原則とした．こうした調整ができるように，プロジェクトの状態を包み隠さずビジネスユーザにも常に開示し，お互いに信頼関係を高めておいたことが，我々のアジャイル開発プロジェクトが成功した理由のひとつであると考えている．

(5) CIツールによるテストの自動化

アジャイル開発を進めていくと，システムは日々成長，進化するので，日々のリグレッションテストが重要となる．今回のプロジェクトではこれをCIツール（継続的インテグレーション[3]ツール）を導入することで自動化した． たとえば，CIツールであるjenkins[5]，selenium[6]を組み合わせることで，ビルドから単体テスト，結合テストまでの実行およびエラー検知を自動化した．

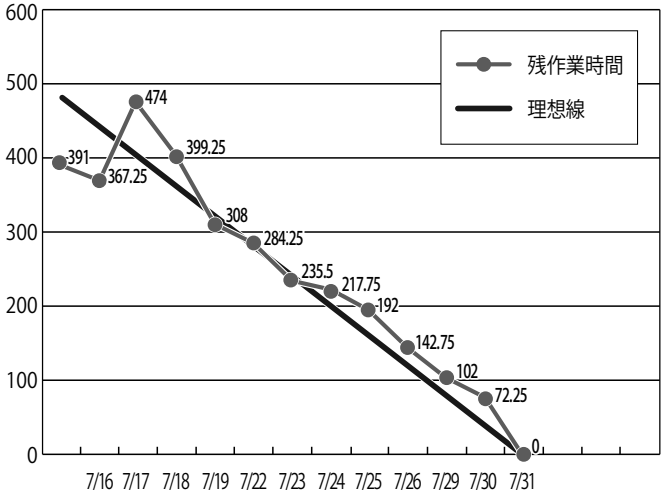

図6　バーンダウンチャート

プロジェクトの後半になればなるほどシステムには機能が増え，最終的には日々のリグレッションテストだけでも約12,000ケースにのぼった．人による打鍵をしていてはリグレッションテストを行うだけで1日が終わってしまい，新規機能を開発する時間はなくなってしまっていたであろう．このような状況になることを避けるために，プロジェクトの初期段階からリグレッションテストの自動化を準備した．

しかし，自動化の仕組みを作っただけでは課題は完全にはクリアできなかった．テストの自動化自体，各種機器に大きな負荷がかかり，日中に実行すると開発環境のレスポンスが悪化するという問題が発生した．そのため，夜間に自動テストを実行し，翌朝，テストの結果を検証し，午後に開発を進めるという1日の作業サイクルを導入する必要があった．

(6) カイゼン活動

過去の実績，経験からさまざまなプラクティスがアジャイル開発で編み出され，公開されている．しかし「こうすれば必ず成功する」といったやり方が確立されているわけではなく，現状では「こうすべき」という正解もない．我々は本プロジェクトを遂行するにあたり，さまざまなやり方を模索しながら開発を進めた．このような開発では，「カイゼン」は必須の活動であった．

当社でも過去に数回アジャイル開発を実践していたが，今回のプロジェクトメンバはいずれもアジャイル開発が未経験だった．先に述べたとおり，グランドデザイン工程で，全員で大枠の進め方を共有したものの，作業の効率化，ムダの排除，コミュニケーションの活性化等，さまざまな面でカイゼンにカイゼンを重ねて，自分たちの進め方を構築することができた．

カイゼン活動は各イテレーション，各スプリントの完了ごとの振り返り会議の問題提起が起因になることが多かった．会議ではKPT法[7]を使用し，問題（Problem）を抽出し，抽出した問題を解決するため解決策（Try）を全員で議論した．

解決策は本当に効果がでるかどうか疑わしい場合もあった．しかし，「とにかく一度やってみて効果がでなければ別の方法を考える」という精神を大切にした．プロジェクトの終盤では，細かい改善を含めると全量が把握しきれなくなるほど，各スクラムチームが自主的にカイ

ゼンを実施する状況にすることもできた．

4. まとめ

ここまで，我々のアジャイル開発で実践したプラクティスについて紹介をしてきたが．今回の開発を通じて，特に，①計画の策定と見直し，②効率化の徹底と継続，そして③適用目的の明確化がアジャイル開発でプロジェクトを成功させるために特に重要な要素と考える．

4.1 計画の策定と見直し

繰り返しになるが，アジャイル開発でも計画作成は非常に重要であった．イテレーションに着手する前に，プロジェクトの進め方とゴール，各イテレーションの考え方，ベンダとの契約方法等，計画と骨格作りをしっかりと行った．また，ウォーターフォール型と異なり，作った計画を遵守する必要はなかったが，常に計画の見直しが求められた．この意味では，アジャイル開発における計画はウォーターフォール型における計画の重要性に匹敵した．

一見遠回りに思えたグランドデザイン工程に十分な期間を確保し，しっかりと計画と準備を行った上で実開発に入れたことが，トータルでは近道となったと考える．

4.2 効率化の徹底と継続

開発工程ではカイゼン活動を継続し，徹底的に効率化し続けること重要であった．アジャイル開発では，ウォーターフォール型以上に，日々変化する状況に，スクラムチームが迅速，柔軟適応することを追求した．状況の変化に迅速，柔軟に適応するためには，従来の開発にあったムダを排除したり，効率化を行ったりすることでチームの俊敏性を維持する必要があった．日々，「自分たちの作業にムダがないか？」「もっと効率化できることはないか？」を見つめなおし，効率化を進めることで，開発チーム自体も変化に強いチームに進化し続けていったことが重要であった．

4.3 適用目的の明確化

今回の実践で，プロジェクトが迅速に変化に適応することが求められているか否かという「プロジェクト特性」により，アジャイル開発による効果が得られやすいケースと，そうでないケースがあると感じた．今後，我々がアジャイル開発を適用するときには，このことを理解して，適用することで得られる効果，狙いを明確化して，適用すべきか否かを見極めなければならないと考える．

一般的にアジャイルは大規模開発よりは小規模開発，メインフレームよりはサーバ開発，大人数よりは少人数による開発に向くとされているが，メインフレームだからアジャイルができないというわけではないことは，今回の我々のプロジェクトで明らかにできた．本プロジェクトは，大規模であり，エンタープライズシステムであった．

今回のプロジェクトでは，立ち上げ時点でビジネスユーザも自らが必要としているシステムを明確にイメージできていなかった．また，保険契約者が使用するシステムであったことから，ニーズの変化が激しく，初期段階で決定した業務要件がシステムリリース時には不要な機能となってしまう恐れがあった．したがって，もし仮にウォーターフォール型で開発していたとしたら，はじめの要件検討で多くの時間が割かれ，ようやく決定した要件も，次々と変更要望があがってしまい，大幅なスケジュール遅延が発生する状況となっていたに違いない．さらに，せっかく苦労して作り上げたシステムも十分な業務ニーズを満たせず，使われないシステムとなっていた可能性もある．

今回のプロジェクトは，アジャイル開発を適用したことで，変化する業務ニーズに柔軟に対応しながら，今までにない画期的なシステムを構築することができたと考えている．

参考文献

1) http://www.ogis-ri.co.jp/rad/webmaga/rwm20121202.html
2) 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).
3) 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).
4) 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).
5) 末広尚義，竹内一成，太田健一郎，西川茂伸: 入門 Jenkins - 実践「継続的インテグレーション」，秀和システム (2012).
6) http://oss.infoscience.co.jp/seleniumhq/docs/01_introducing_selenium.html
7) 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).

花 宜典（非会員）yoshinori.hana@grp.tmnf.jp
2005 年東京海上日動システムズ（株）入社．現在，生保本部あんしん生命システムデザイン一部デザイナ．

小作 祐史（非会員）yuuji.ozaku@grp.tmnf.jp
1988 年東京海上システム開発（株）入社．現在，東京海上日動システムズ（株）生保本部長代理．

採録決定：2014 年 8 月 14 日
編集担当：中谷多哉子（筑波大学）