特集号
招待論文

委託開発におけるハイブリッドアジャイルの適用効果

英 繁雄[†1]　高月 裕二[†1]　東 大介[†1]

[†1] (株) 日立ソリューションズ

日本では，エンタープライズ型のシステム開発は，IT サービス企業へ委託する場合が多い．欧米で多く適用されている迅速な開発手法であるアジャイルプロセスは，委託開発が主流の日本では普及しづらいのが現状である．本稿では，エンタープライズ型のシステム開発にハイブリッドアジャイルを適用し，アジャイルプロセスで採用されるいくつかのプラクティスから適用効果を評価する．適用したプラクティスは，反復，イテレーション計画，テスト駆動開発，継続的インテグレーション，コードレビュー，イテレーションレビュー，バーンダウンチャートである．

1. はじめに

本稿は，企業の人員管理システムをハイブリッドアジャイル[1]で委託開発した事例である．ここでいうハイブリッドアジャイルとは，ウォーターフォールモデルの開発工程のうち，詳細設計から製作（単体テスト）までをアジャイルプロセスで行う方法とした．

人員管理システムなどのエンタープライズ型システムの開発は，リリース時期を確定して開発する場合が多く，段階的な短期リリースを繰り返し行う必要はない．しかし，要件変更に柔軟に対応する開発効率と品質を向上する手法も求められている．

その解決方法として，本プロジェクトでは，アジャイルプロセスで採用されるいくつかのプラクティスから，反復，イテレーション計画，テスト駆動開発，継続的インテグレーション，コードレビュー，イテレーションレビュー，バーンダウンチャートを適用した．

アジャイルプロセスは，市場への早期提供や変更への迅速な対応を求める BtoC（Business to Consumer）型のシステムや製品開発には，相性の良い開発手法である．しかし，エンタープライズ型のシステム開発では，日本の企業は委託開発をする場合が多く，ユーザ企業と IT サービス企業の間に契約が必要となる．そのため，コストの見積りが契約前に必要となり，アジャイルプロセス適用プロジェクトでよく行われるような，システム要件を確定しながら開発を行ったり，要件変更を受け入れたりすることは，委託開発では難しい．

事例プロジェクトでは，基本設計まで行い，その後に発生する要件変更にハイブリッドアジャイルで対応した．プロジェクト結果を品質とコスト，スケジュール，利用者満足度の観点で評価する．

2. 日本のソフトウェア開発の特徴

日本では，エンタープライズ型システムの開発は，委託開発が主流である

2.1 IT 技術者構成の日米比較

図 1 は，米国と日本の IT 技術者の構成を表したものである[2]．米国では，IT 技術者全体の約 70 % がユーザ企業に勤めているが，日本では，ユーザ企業に勤める IT 技術者は約 25 % にすぎない．これは，米国では，ユーザ企業内でシステム開発する内製が主流であり，日本では，ユーザ企業から IT サービス企業へ委託開発することが主流であるといえる．

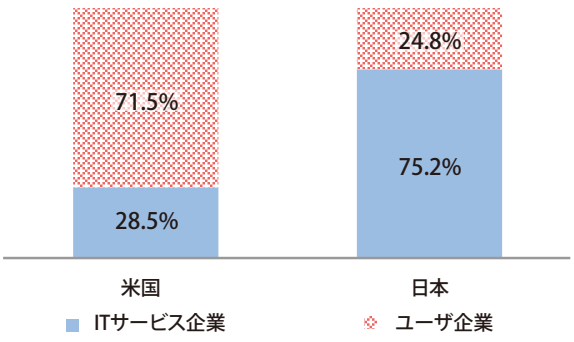

図 1　日米の IT 技術者の分布状況（文献 [2] より筆者らが作成）

2.2　契約における日米比較

2.2.1　日本の契約方法

日本の委託開発の契約方法は，準委任契約と請負契約の2通りがある．

請負契約は，請負人がある仕事の完成を約束し，注文者がその仕事の完成に対して，報酬を支払うことを約束する契約である．一般的には，注文者の検収合格で，請負人から注文者へ引渡しとなる．引渡し後に瑕疵が発覚した場合には，請負人に瑕疵担保責任があり，修補義務や損害賠償義務が生じる．請負代金は仕事の完成後に支払うのが原則である．

準委任契約は，委任者が一定の業務の処理を委託し，受任者がこれを承諾する契約である．受任者は，善良な管理者の注意をもって当該業務を処理する義務（善管注意義務）を負うものである．

2.2.2　米国の契約方法

米国の契約方法は，FAR（Federal Acquisition Regulation）[3]に記載されている．Fixed-Price Contracts（固定価格契約）やCost-Reimbursement Contracts（実費償還契約）などの各契約の中に，物価変動調整やインセンティブ付きなどのバリエーション契約も規定されている．

日本の委託開発の契約は，米国のFixed-Price ContractsやCost-Reimbursement Contractsに相当する．しかし，米国では，開発範囲のスコープ変更や貢献度などに対して，対価を追加できる契約方法が規定されている．そのため，内製が主流であることだけがアジャイルプロセスの普及の理由ではなく，委託開発であっても変更に対応するアジャイルプロセスを適用する契約を締結しやすいことも普及の理由となっている．

2.2.3　日本と米国の契約方法の比較

日本の請負契約は米国のFixed-Price Contractsに近く，派遣契約は米国のTime-and-materialsに近い．準委任契約は米国のCost-Reimbursement ContractsやTime-and-materialsにも似ているが，日本固有の契約方法である．

米国のCost-Reimbursement Contractsは，費やした実費は，必ず確保できるため，ITサービス企業にはコスト面でのリスクはない．Time-and-materialsは，実作業時間と必要な経費で契約するが，日本の準委任契約は，実作業時間で契約する場合だけでなく，工数見積りから価格を決めることもあるため，少し意味が異なる．準委任契約の工数見積りは，ITサービス企業の生産性データから算出された工数人月とその他の必要費用から価格を決めることも多い．

3.　委託開発における課題

委託開発には，アジャイルプロセスは適用が難しい

日本で委託開発する場合は，柔軟な価格設定が難しいため，早期リリースや変更への柔軟な対応，開発コスト削減を目的にしたアジャイルプロセスの適用は，敷居が高いといえる．

しかし，ウォーターフォールモデルのままでは，現状の開発プロセスの課題は改善できない．ウォーターフォールモデルの課題とアジャイルプロセスを委託開発に適用する場合の課題を以降に挙げる．

3.1　ウォーターフォールモデルの課題

3.1.1　コストの課題

要件変更が発生し，開発コストに影響がある場合は，契約の見直しが発生する．ウォーターフォールモデルの場合は，ユーザ企業側も追加予算を確保するためには，多大な努力が必要となる．

3.1.2　スケジュールの課題

要件定義の段階で，開発する機能を決定する必要があり，要件確定の遅延によりスケジュールが遅延するリスクがある．基本的には，開発が完了するまでリリースを行わないため，ユーザ側のテスト工程以降に仕様齟齬による手戻り作業が発生するリスクもある．

3.1.3　利用者満足度の課題

リリース後に利用者の不満が多く挙がってくる場合があり，ユーザビリティやアクセシビリティなど利用者満足度が得られないリスクがある．

3.2　委託開発時のアジャイルプロセスの課題

ウォーターフォールモデルの課題を解決するために，アジャイルプロセスの適用を検討するが，エンタープライズ型システムの委託開発では，アジャイルプロセスを適用する上での課題がある．

3.2.1　契約に対する課題

アジャイルプロセスでは，変更への柔軟な対応を求められる場合が多く，そのためコストとスケジュールに変更が必要となる．委託開発時は，事前にコスト見積りを行い契約するため，変更への追加契約の対応が課題である．

3.2.2　品質に対する課題

アジャイルプロセスでは，設計書を省略または簡略化し，作業効率を上げる場合が多い．また，ウォーターフォールモデルのように，テスト工程も明確に定義されていないため，テスト不十分となり品質劣化となるリスクがある．

品質確保のための施策が不十分なプロジェクトでは，次のようなケースが品質劣化の要因として考えられる．

- 設計書を省略・簡略化することにより，テストケースが不十分となる．
- 設計書を省略・簡略化することにより，実装されたシステムと注文者が求める仕様とに齟齬が発生する．
- 変更を受け入れることにより，デグレードが発生する可能性が高くなる．
- 変更を受け入れることにより，再テスト工数が膨らむ．
- 品質管理方法やテスト計画が不十分なことにより，問題発見が遅れる．

3.2.3　コストとスケジュールの課題

アジャイルプロセスでは，仕様が確定しない状態で開発を進めたり，要件変更を受け入れたりすることを前提として開発を行うため，スコープ変更が発生する．そのため，開発コストの膨張や何度もスケジュールの変更となるリスクがある．

4.　委託開発における課題解決

委託開発には，ハイブリッドアジャイルは相性が良い

委託開発の課題を解決するため，ハイブリッドアジャイルを適用し，契約の課題解決と品質，コスト，スケジュール，利用者満足度の改善を目指す．

図2は，ウォーターフォールモデルの工程の詳細設計から単体テストを含む製作工程までを，アジャイルプロセスでイテレーションを数回繰り返し行う一例である．ハイブリッドアジャイルとして特別に定義されたものはない．

4.1　契約の対策

委託開発する場合は，基本設計から総合テスト（ベンダ確認）までの工程を1つの契約で締結する一括請負契約とする場合もある．しかし，作業量見積りが難しい工程を準委任契約とする分割契約が採用されることも多い．たとえば，ユーザ企業主体で行う要件定義（場合によっては基本設計まで）と総合テスト（ユーザ確認）を準委任契約で行い，基本設計から総合テスト（ベンダ確認）までを請負契約のように行う場合である．

図3は，要件変更を受け入れる対策として考えた契約方法である．まずは基本設計までを準委任契約で実施し，当初計画分として基本設計終了時点の要件で基本契約を結ぶ．要件変更に対する追加分は，当初計画分とは別に追加契約とする．ここで重要なのは，要件変更量を予測し，ユーザ企業側は要件変更量を含めた予算をあらかじめ確保し，ITサービス企業と要件変更分に対するコストのゆとりを共有しておくことである．イテレーションごとに対応する要件変更をユーザ企業とITサービス企業の間で，コストのゆとりの予算内で合意する．

結合テストは，アジャイルプロセスの適用範囲外としたが，詳細設計から結合テストまでを請負または準委任契約とした例である．

- 基本契約：変化しない部分の契約

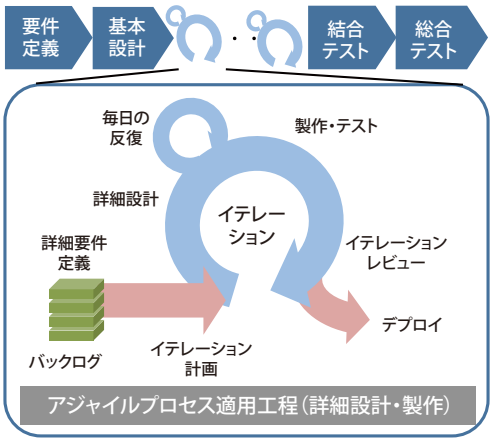

図2　ハイブリッドアジャイルの作業

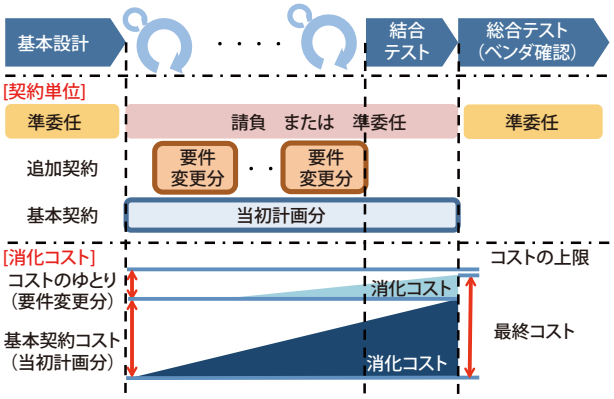

図3　ハイブリッドアジャイルの契約例

- 追加契約：イテレーションごとに追加となる具体的な成果物や変化する部分に対する契約

4.2　品質の対策

アジャイルプロセスでは，開発効率を上げるために作業の省略と簡略を行うが，品質向上のための作業はしっかりと行う．作業項目が増えてもテストの自動化などで作業スピードを上げることを考える．

図4は，品質を確保するためにウォーターフォールモデルで実施していた現行手順に対して，アジャイルのプラクティスを適用した今回手順である．

詳細設計と製作・テストに該当する作業は，毎日の反復の中で，テスト駆動開発と継続的インテグレーション，コードレビューのプラクティスを適用して同時に行う．

今回手順の作業内容は次のとおりである．

(1)　イテレーション計画

イテレーション計画とは，各イテレーションの始めに，今回のイテレーション中に開発する機能を決定する場のことである．

開発チームは，イテレーション計画の時間内で，開発する機能を作業レベルのタスクに分割し，作業に必要な工数を見積る．工数を見積るために，開発する機能の仕様を理解する必要があり，全員で見積りのための議論をする．こうすることにより，開発チームメンバ全員が，このチームで開発する機能の仕様を理解することができる．

(2)　テスト駆動開発

テスト駆動開発は，プログラミング時に，テストコードの作成と，それをクリアする処理の実装を少しずつ交互に行いプログラムを完成させていく方法である．テストコードを先に作成することで，開発者は余計な処理の実装を抑制したり，テスト実施漏れを防いだりする効果がある．

作成したテストコードは，後に修正作業によってデグレードが発生していないことを確認する場合にも再利用できる．図5は，テスト駆動開発の手順である．数分の短いサイクルで，①から④までの作業を繰り返し，期待する動作のプログラムを完成させる．

リファクタリングとは，プログラムの振舞いは変えずにコードを綺麗にすることである．

コードレビューは，ペアプログラミングで行っていれば，コード作成と同時に行う．

(3)　継続的インテグレーション

継続的インテグレーションとは，個々の開発者が完成したプログラムを開発チームの共用サーバに常時結合し，設定した任意のジョブを自動実行するための専用環境である．

プログラムコードのビルド，テスト実行，コード解析などを自動的に行うように設定する．継続的インテグレーションでは，ソフトウェアを動作する状態を維持しながら，再テストを頻繁に行うため，バグを即時に発見でき，再確認作業の効率化と品質確保に効果がある．

(4)　コードレビュー

コードレビューは，プログラムした本人以外のものが実施する．

アジャイルプロセスでよく用いられるペアプログラミングは，2人で1台のパソコンに向かって，一緒に開発する方法である．1人がナビゲータで，もう1人がドライバの役割となる．ドライバ役の開発者がコーディングすると同時に，ナビゲータ役がコードレビューをする．テスト駆動開発のテストコードとプログラムコードの作成時にコードレビューを行う．

ペアプログラミングを行わなくても，開発者の技術的リーダかほかのプログラマが，プログラムの完成後に各開発者のプログラムをコードレビューすることもある．

(5)　イテレーションレビュー

イテレーションレビューとは，アジャイル開発などで各イテレーションの最後日に，仕様に意見を述べること

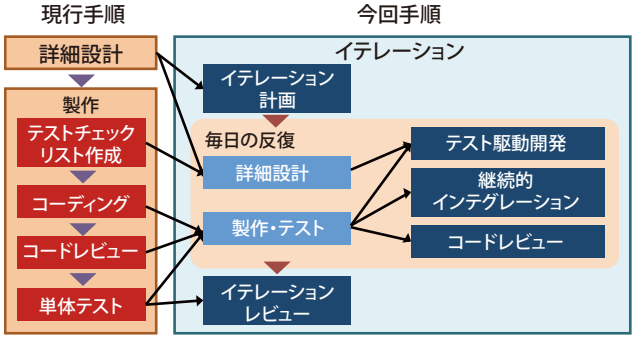

図4　品質確保のために実施した作業例

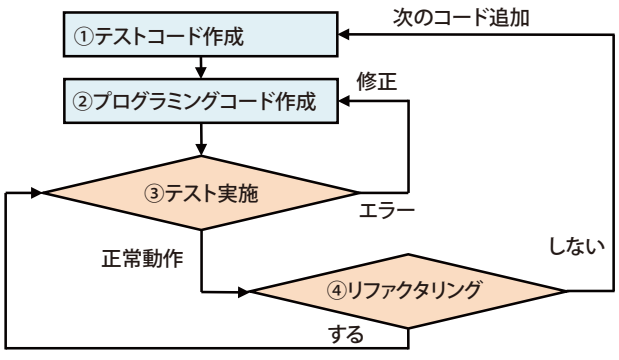

図5　テスト駆動開発の作業手順

ができる数名のステークホルダに参加してもらい，実際に動くアプリケーションで仕様を確認してもらう場である．これにより，実装漏れや仕様齟齬の早期発見，利用者観点から見た操作性の確認を行うことができる．

イテレーションレビューは，開発中に定期的に実際に動作するもので確認することで，問題を早期に発見することができ，品質向上と手戻り作業を最小限に抑えることもできる．

4.3　コストの対策

(1) 設計書の削減

詳細設計書に相当するドキュメントは，できる限り省略し，設計イメージをホワイトボードや紙に手書きでメモ書き程度にする．設計書を省略・簡略化することで，作業量の削減だけではなく変更による設計書の修正コストを削減する．

(2) 作業スピードの向上

管理や会議体の見直しや効率化を行い作業量の削減を行う．また，有効な開発ツールを導入し，作業スピードの向上も行う．特にテスト作業は，継続的インテグレーション環境で自動化することで効率化できる．

4.4　スケジュールの対策

イテレーション単位の反復で開発することにより，実際に動作するアプリケーションを定期的に仕様齟齬と操作性を確認しながら開発することができる．そのため，早い段階で修正対応ができ，手戻り作業の影響を低減することができる．

1～4のイテレーションの進捗をチームごとに管理することで，管理対象が小さくなりコントロールしやすくなる．図6は，チームのイテレーション内の進捗を管理するバーンダウンチャートのイメージ図である．チームの開発スピードを日々監視しながら進めることができる．

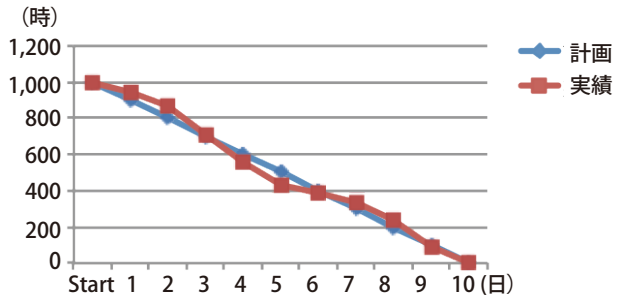

図 6　バーンダウンチャートの実施例

4.5　利用者満足度の対策

イテレーションレビューで，実際に動作するアプリケーションを確認してもらうことにより，操作上の問題を早期に確認することができる．

開発期間中に問題点の対策を行うため，利用者満足度の高いシステムを，初期リリースから提供することができる．

5.　ハイブリッドアジャイルの評価

ハイブリッドアジャイルは，
リスク防止効果が高い

実際に，委託開発でハイブリッドアジャイルを適用した人員管理システム開発プロジェクトの結果で評価する．

5.1　契約における改善

要件変更分に対する追加コストを，基本契約分に対して，追加契約を20%として目標コストを設定した．

追加分のコストをイテレーションごとに調整しながら契約する方法で実施した．次のイテレーションとの間の短期間に，ユーザ企業とITサービス企業間で，対応する要件変更の合意を得ながら進める．

要件変更分を予測して，目標コストを設定する契約方法は，日本国内におけるソフトウェア開発契約では，事例が見当たらない．

コスト増加とスケジュール遅延を抑止するために，コストの上限値をユーザ企業とITサービス企業間で目標コストとして共有することとし，それにより追加契約時の作業負担を低減させることができた．

5.2　品質における改善

4.2節の品質の対策で実施した結果で評価する．図7は，

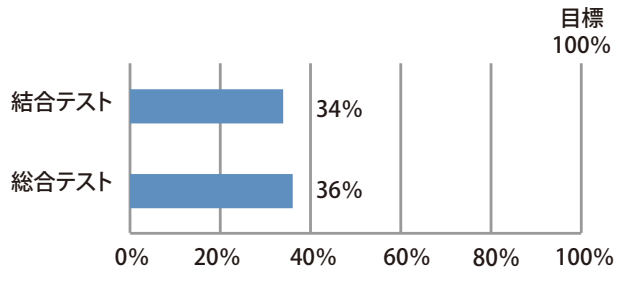

図 7　バグ摘出目標値に対するバグ抽出実績比率

ウォーターフォールモデルで標準としているバグ摘出目標値（コードステップあたりのバグ摘出件数）を100％とし，それに対する実績件数の比率を示したものである．結合テストと総合テストで，それぞれ34％と36％という結果が得られた．テストカバレッジは，C0（命令網羅）とC1（分岐網羅）で100％としており，テスト実行が困難なテストケースはソースコードの目視確認とした．稼動後も安定していることから，コーディング段階から高い品質が維持されていたといえる．

アジャイルプロセスでよく用いられるプラクティスは，品質向上に効果的なものも多くあり，作業の簡略化だけでなく，それを補う開発テクニックを備えることが，アジャイルプロセスで品質を劣化させないためには必要である．

5.3 コストにおける改善

図8は，事例プロジェクトから得たハイブリッドアジャイルでのコスト削減率である．ただし，この値がほかのプロジェクトにも常に適用できるものではないことに注意していただきたい．

工数比の平均値は，『ソフトウェア開発データ白書2014-2015』[4]のウォーターフォールモデルの工数比を参考にした．ウォーターフォールモデルの各工程の工数比と，事例プロジェクトで得られたデータから算出したハイブリッドアジャイルの工数比の比較である．

アプリケーションの新規開発時の基本設計から総合テスト（ベンダ確認）までの全体工数を100％とし，各工程比率に対してハイブリッドアジャイルでの工数削減率を適用してみると，全体で89.1％に工数を削減できた．

各工程の工数比は，次のように算出した．

(1) 詳細設計と製作工程

表1は，プロジェクトの実績データである．詳細設計から製作（単体テスト）までの工程範囲で，設計書の省略・簡略化と自動化の効果で約90％の工数に削減できた．

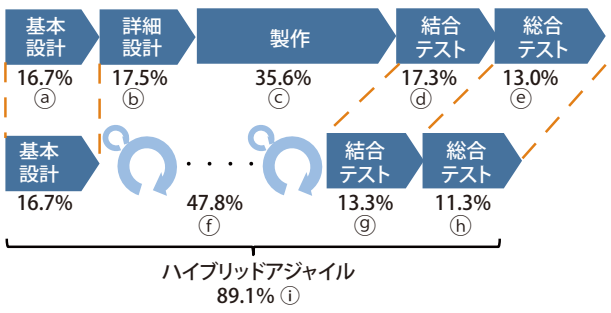

図8　コスト削減率

- ウォーターフォールモデルの詳細設計と製作の合計工数比率
  17.5％ⓑ＋35.6％ⓒ＝53.1％ⓠ
- ウォーターフォールモデルに対するハイブリッドアジャイルの詳細設計と製作の合計工数比率
  17.5％ⓑ×61％ⓙ＝10.6％ⓡ
  35.6％ⓒ×98％ⓚ＝34.8％ⓢ
  10.6％ⓡ＋34.8％ⓢ＝45.4％ⓣ
  45.4％ⓣ／53.1％ⓠ≒85％ⓜ
  85％ⓜ＋5％ⓝ＝90％ⓟ
- ハイブリッドアジャイルの詳細設計と製作の合計工数比率
  53.1％ⓠ×90％ⓟ＝47.8％ⓕ

(2) テスト工程

テスト工程では，品質の対策を実施したことでバグ摘出件数が，過去の類似プロジェクトの実績から設定した目標値に対し減少した．バグ摘出件数の目標と実績の差からバグ修正時間減少をコスト削減効果とし，テスト工程の工数を削減することができた．

具体的には以下のような計算式により，結合テスト，総合テストの工数比率を，それぞれ76.7％，87.3％と算出した．

［テスト工程の工数削減効果の計算式］

a：実績テストケース消化時間
b：平均バグ修正時間
c：バグ摘出実績件数
d：バグ摘出目標件数
工数比率＝a＋b×（c／d）

テストケース消化時間は，ウォーターフォールモデルでもハイブリッドアジャイルでも同等に行うため，変わらない．

表1　ハイブリッドアジャイルの実績工数比率例

工程	作業概要	工数比☆	
		工程単位	詳細設計＋製作
詳細設計	設計書の省略と簡略化を行い効率を上げる	61％ⓙ	85％ⓜ
製作	テスト駆動開発によるテストコード作成（テストコード分のチェックリストは，作成しない）と継続的インテグレーションによる作業効率化をする	98％ⓚ	
	メモ書きなどから，保守用ドキュメントして必要なものを清書する	－	+5％ⓝ
合計		90％ⓟ	

☆ウォーターフォールからアジャイルにした場合の工数比率

ハイブリッドアジャイルでは，バグ摘出件数が減少したことにより，バグ修正時間が減少する．

テストケース消化時間とバグ修正時間を加えたものが，テスト工程の時間となる．

- 結合テスト工程の工数比率 ＝ 76.7％
  17.3％ⓓ × 76.7％ ＝ 13.3％ⓖ
- 総合テスト工程の工数比率 ＝ 87.3％
  13.0％ⓔ × 87.3％ ＝ 11.3％ⓗ

5.4　スケジュールにおける改善

図9は，事例プロジェクトの各イテレーションのバーンダウンチャートである．

イテレーション1では，途中作業遅延が発生したが，早期に状況を把握できたことで，イテレーション1の期間内に，遅延個所をチームメンバが協力して対策をしたり，メンバの得意作業に担当作業を割り当て直したりして遅延から回復した．

イテレーション2以降は，チームの開発スピードも安定し，遅延なく開発が進んだ．このようにチームの開発スピードを，短期間のイテレーションで繰り返し管理する方法は，管理対象を小さくすることができる．そのため，チームのスケジュールをコントロールしやすかった．

5.5　利用者満足度における改善

図10は，イテレーションレビューで指摘された要望件数の比率である．

グラフの操作性向上の39％と誤操作防止の31％との合計70％は，システムの操作性と解りやすさのことであり，利用者満足度につながる指摘である．

利用者からの操作手順が複雑で間違えやすいとか意味が分かりづらい記述などの不満は，ウォーターフォールモデルではリリース後に挙がるが，ハイブリッドアジャイルでは，リリース時には対策できていた．

6.　おわりに

エンタープライズ型システムの委託開発の場合でも，ハイブリッドアジャイルであれば，アジャイルプロセスの契約の課題を解決することができる．ハイブリッドアジャイルでは，アジャイルプロセスに大きく期待する早期リリースや大幅なコスト削減はできないが，アジャイルプロセスのプラクティスを採用することで，開発プロジェクトのリスク低減や品質，コスト，スケジュール，利用者満足度の向上を図ることができる．

アジャイルのプラクティスは，ウォーターフォールモデルでも適用可能なものが多い．しかし，ウォーターフォールモデルのままで今回のプラクティスを適用しても，生産性向上は期待できない．

開発するシステムの開発体制や要求されるリリース時期，品質の重要度などのプロジェクト特性に合わせて，Scrumなどの一般的なアジャイルプロセスとハイブリッドアジャイルを使い分けていくのがよい．

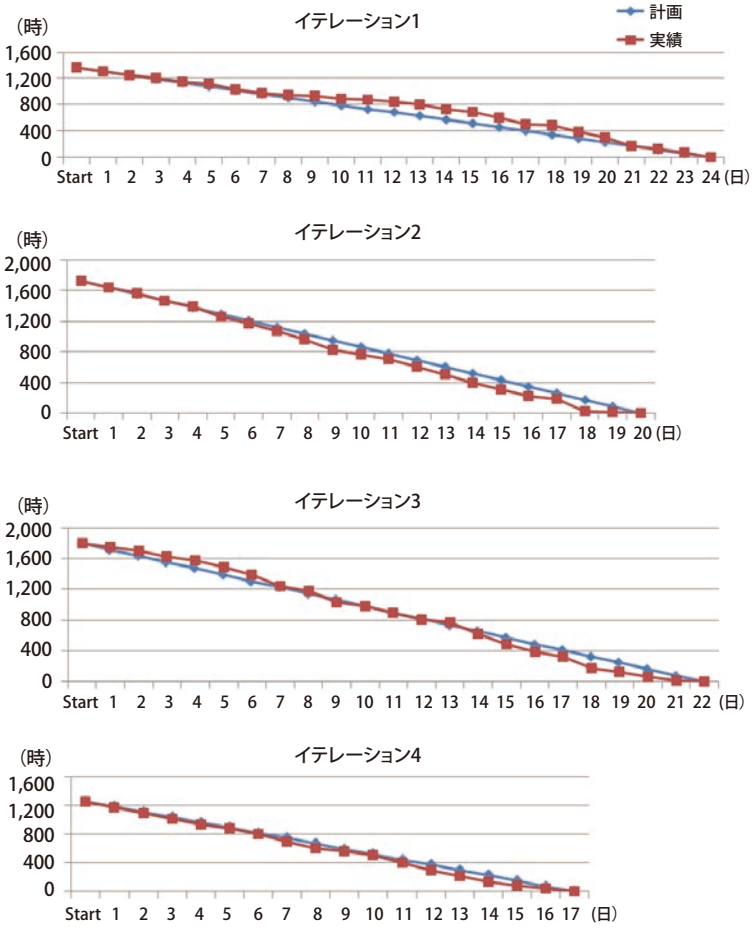

図9　バーンダウンチャートによるスケジュール管理

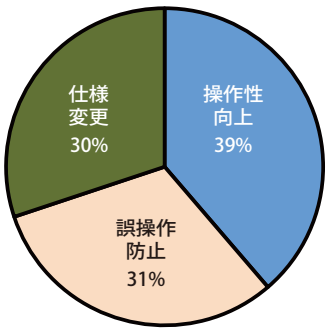

図10　イテレーション中の変更件数比率

参考文献

1) 英　繁雄 他（著），長瀬嘉秀（監修）：ハイブリッドアジャイルの実践，（株）リックテレコム（2013）．
2) （独）情報処理推進機構：IT 人材白書 調査報告書 グローバル化を支える IT 人材確保・育成施策に関する調査，p.44, 図表 3-14, http://www.ipa.go.jp/files/000010609.pdf（2016 年 4 月 11 日現在）
3) FAR（Federal Acquisition Regulation），https://www.acquisition.gov/?q=browsefar（2016 年 4 月 11 日現在）
4) （独）情報処理推進機構ソフトウェア・エンジニアリング・センター：ソフトウェア開発データ白書 2014-2015, p.227, 図表 8-1-13,（2014）．

英 繁雄（正会員）shigeo.hanabusa.zg@hitachi-solutions.com
1985 年，現在の（株）日立ソリューションズ入社，アジャイルプロセスには，2002 年から取り組み，アジャイルプロセス適用支援を行っている．著書に「ハイブリッドアジャイルの実践」（（株）リックテレコム）がある．

高月 裕二（非会員）yuji.takatsuki.kk@hitachi-solutions.com
1992 年，現在の（株）日立ソリューションズに入社，2012 年に DevOps やアジャイル開発ソリューション整備に取り組み，ユーザ企業へコンサルティングを行っている．

東 大介（非会員）daisuke.azuma.cj@hitachi-solutions.com
2005 年，現在の（株）日立ソリューションズ入社，開発部に所属し，組込みソフトウェア開発に従事．2012 年以降は，アジャイル開発ソリューションに携わり，ユーザ企業へのアジャイルプロセス導入支援を行っている．

採録決定：2016 年 4 月 11 日
編集担当：斎藤　忍（日本電信電話（株））