アジャイル開発におけるテスト品質向上を目的とした ウォーターフォール開発のテスト観点活用の提案

A proposal for utilizing the test viewpoints of Waterfall development to improve test quality in Agile development

研究員　：田中邦男（SCSK 株式会社）
丸山泰生（日本ユニシス株式会社）
篠原礼（三菱総研 DCS 株式会社）
小室伊作（岡三情報システム株式会社）
主　査　：永田敦（サイボウズ株式会社）
副 主 査：山口鉄平（freee 株式会社）
副 主 査：荻野恒太郎（株式会社カカクコム）
アドバイザー：細谷泰夫（三菱電機株式会社）

研究概要
アジャイル開発では要件の変更を柔軟に取り込みながら短期間に繰り返しソフトウェアをリリースしていくため,ウォーターフォール開発で行われるような工程毎の品質活動が行われず,リリースする機能単位のテストが品質を担保するために重要な役割を果たしている.そこで本研究ではテスト品質向上のためにテスト内容の確認項目を決める際の拠り所として,汎用的に利用できるアジャイル開発向け「テスト観点」を提案する.実績のあるウォーターフォール開発の各テスト工程のテスト観点を基にアジャイル開発向けに利用できる観点を抽出し,プロジェクトで活用できるチェックリストとして作成した.今後,作成したリストをプロジェクトで活用し,アジャイル開発のテスト品質改善の検証を行う.

Abstract
In this study,a "test viewpoints" for agile development is proposed,which can be used universally as a basis for deciding the test items for agile development to improve the test quality.Based on the test viewpoints of each test process in the proven waterfall development,the test viewpoints were extracted for agile development and created a checklist that can be used in a real project.In the future,the improvement of the test quality of agile development shall be verified by using the checklist in the real project.

1. はじめに

ISO/IEC 25010 [1] では,システム及びソフトウェア品質を利用者視点である利用時の品質および開発者視点である製品品質の二つに分類して定義している.アジャイル開発では動くソフトウェアにより顧客が仕様や動作を確認するため,一般的に利用時の品質はある程度確認される傾向にある [2].一方で,開発者視点の製品品質が十分に確保されていないと,ソフトウェアの市場投入後に製品品質の不良に起因する不具合が発生するリスクが高まることになるため,利用時の品質と同様に製品品質の確認も重要である.

アジャイル開発ではリリースする機能毎のテストが品質担保の手段として重要な役割を果たしている.アジャイル開発における短期間の開発サイクルの中で,いかに開発者視点のテストを効率的かつ効果的に実施して製品品質を確保するかは,アジャイル開発における品質改善のための課題である.

2. 研究課題と活動の目標

アジャイル開発に取り組む組織には，ウォーターフォール開発の経験無しに最初の開発からアジャイル開発を取り入れる組織と，長年に渡りウォーターフォール開発を行ってきた組織がアジャイル開発に取り組む組織の 2 パターンが考えられる．今回の研究員全員が後者のウォーターフォール開発に取り組んできた組織がアジャイル開発に取り組むパターンである事から，各組織が長年に渡り蓄積してきたノウハウを如何に活用しアジャイル開発におけるテストの品質を高めることができるかを研究課題とした．

アジャイル開発のテストでは，顧客の動作確認による利用時品質の観点も含め，開発するシステム全体の製品品質を向上させる必要がある．これを補うためにウォーターフォール開発で実績のあるテスト設計時の観点資産を再利用する．

本研究の目標は，テストの品質を高める方策としてテスト観点を作成し，実際のアジャイル開発を行うチームに適用することによってソフトウェアの市場投入後の不具合発生抑制に効果があることを検証することである．ただし，作成するテスト観点については，情報系システムを対象とする．

3. 研究課題検討

3.1 ウォーターフォール開発とアジャイル開発におけるテスト工程の違い

アジャイル開発では，図 1 に示すようにテスト実施のタイミングが異なる．

一般的にウォーターフォール開発においては，設計および実装が終了したあとに全ての機能について単体テスト・結合テスト・総合テスト・受入れテストが行われ，最後にシステムがリリースされる． 一方アジャイル開発工程においては繰り返し行われる短期間の開発工程の中でテストが繰り返し実行される．システムリリースも機能毎に実施される．

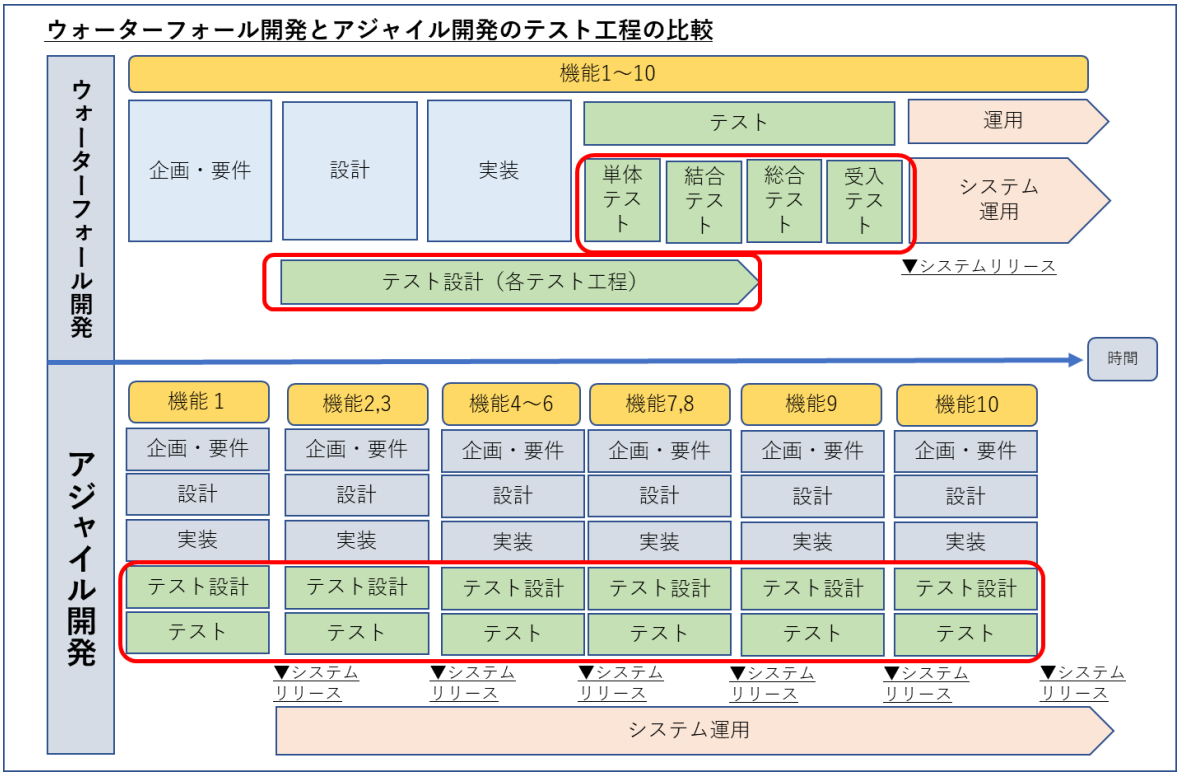

図 1　ウォーターフォール開発とアジャイル開発のテスト工程の比較

3.2　アジャイル開発におけるテストの特徴

テストの特徴として以下のような点を挙げることができる.

(1) ウォーターフォール開発におけるテストと異なり単体テスト,結合テスト,総合テストといったテスト工程が明確に細分化されておらず短期間のスプリント毎に繰り返しテストを行う.

(2) 事前にテスト設計を行わずテスト実行に伴いテスト設計を行う探索的テストを主たるテストとして実行するケースがある,またテストベースとしてユーザストーリーをベースに実施する.

(3) 顧客との仕様確認のための総合的な動作テストを先に行い,単体テストや結合テストのような機能内テストをその後に行うなどテスト項目の実施順序の自由度が高く,開発機能とその開発タイミングに応じて機能テスト内のテスト項目や非機能テストのテスト項目に優先順位をつけテストを実施していくことになる[3].

(4) 繰り返し機能の開発を行なうためリグレッションテストなど同様のテストケースを利用できる部分については,テストを自動化しているケースが多い.

3.3　ウォーターフォール開発のテスト資産の再利用

テスト資産の再利用について以下の仮定を立てた.

(1) ウォーターフォール開発もアジャイル開発も開発プロセスは異なるが同じソフトウェア開発である.

(2) ウォーターフォール開発の各工程で作成したテスト設計書のチェックリストをアジャイル開発のテスト設計時に活用すれば,網羅的なテストを行うことが可能となり市場投入後の不具合抑制につなげることができる.

(3) ウォーターフォール開発の各工程のテスト資産はそのままではアジャイル開発のテスト資産には利用しづらい面があるため,品質特性で分類してから活用を検討することにした.

以上のアジャイル開発におけるテストの特徴を踏まえ上記仮定に基づき,アジャイル開発のテスト設計を行う際のよりどころとなるテスト観点について検討することとした.再利用の対象については,情報系システムのテスト資産を再利用した.したがって組み込み系システムなどは対象外とした.

4.　テスト観点表の作成

4.1　テスト観点表作成手順

ウォーターフォール開発で実績のあるプロジェクトから収集した各工程のテスト設計のチェック観点をベースとして,アジャイル開発のテスト設計に活用できるテスト観点表を以下の手順で作成する.

(1) ウォーターフォール開発向けに作成された単体テスト,結合テスト,総合テスト,受入テスト,非機能テストにおけるそれぞれのテスト設計のチェック項目を収集する.

(2) 特定の組織に偏ったテスト観点表とならないように複数組織から収集したチェック項目の同類項目を整理して記載内容の汎用化を行なう.

(3) ウォーターフォール開発では品質特性とテストレベルで分類していた項目を品質特性に合わせて分類しアジャイル開発に活用できる観点として整理した.

(4) 3.2(3)のアジャイル開発のテスト実施順の自由度が高いことに対応するためアジャイル開発のテスト実施時に優先順位や必要な観点の選択ができるようにする.

(5) 観点表は各観点項目の確認ポイント例を記載することで,より具体的なテスト設計に活用できるようにする.

(6) 初期段階は，複数システムに共通する観点を対象としているため，システム固有のテスト観点は対象外とした．個別観点については，7. 今後の課題と展開で記載する．

図 2 は，ウォーターフォール開発プロジェクトのテスト観点収集からアジャイル開発に活用できる観点作成の流れを示す．

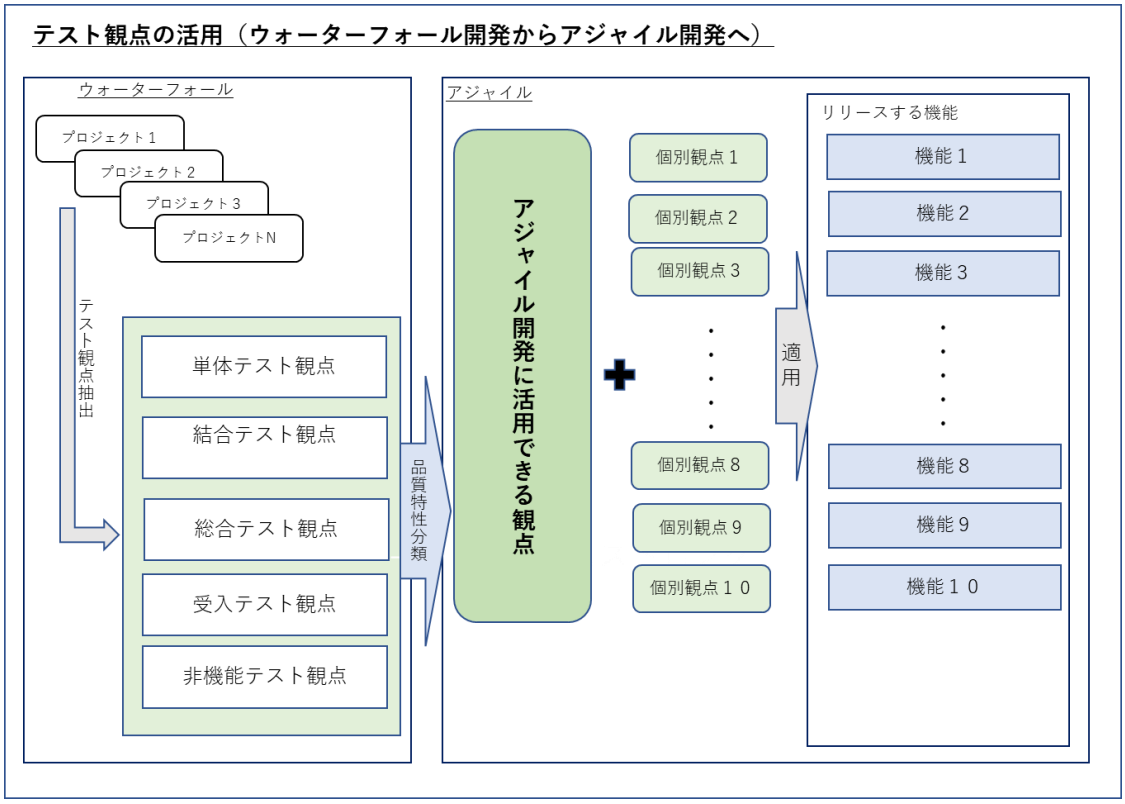

図２　ウォーターフォール開発のテスト観点のアジャイル開発への適用

4.2　テスト観点表の内容

表１は，テスト観点の大項目を開発視点での品質特性別[1]に分類したものである．テスト観点は機能観点だけでなく，非機能観点についても記載した．また情報システムで一般的なテスト項目を網羅する形とした．

表１　テスト観点

NO.	テスト観点項目	テスト観点概要	品質特性
1	機能	機能単位の詳細,要求項目の実現,画面/帳票の操作性,帳票の出力先 内部/サブシステム間/システム間のＩ／Ｆ,排他制御など	機能適合性 保守性
2	業務	業務サイクル,業務運用など	機能適合性
3	セキュリティ	アクセス制御,表示制御など	セキュリティ
4	性能・負荷	オンラインレスポンス,バッチ処理時間,負荷,総合性能など	性能効率性 信頼性
5	移行	データ移行,業務移行,システム移行など	互換性 移植性
6	障害回復	障害時処理,システム間連携障害,障害時運用など	信頼性
7	システム運用	ジョブ・ジョブネット,本番運用手順など	信頼性
8	システム環境	本番環境の確認など	信頼性
9	ユーザビリティ	利用手順,画面の構成,分かりやすさ エラーの防止と処理,ヘルプなど	機能適合性 使用性

表２は,プロジェクトで使用することを想定して作成した観点一覧（一部抜粋）である.大項目は 9 項目,中項目は 32 項目,小項目は 127 項目あり,小項目毎に確認ポイント例を記載してイメージしやすい構成になっている.

表２　テスト観点一覧（一部抜粋）

確認観点								テスト優先順	テスト対象
大項目 Ⓐ		中項目 Ⓑ		小項目 Ⓒ		確認ポイント例 Ⓓ		Ⓔ	Ⓕ
1	機能	1-1	基本的な機能	1-1-1	画面・帳票の基本的な機能	画面、帳票等ユーザーインターフェース機能の基本的な処理を検証			
				1-1-2	バッチ処理の基本的な機能	バッチジョブの基本的な処理を検証			
		1-2	機能単位の詳細	1-2-1	プログラムロジック	ホワイトボックステストによるC1カバレージ（全ての分岐条件の「真」と「偽」を網羅）検証、または、C2カバレージ（分岐が複合条件の場合、各々の条件の「真」、「偽」を網羅）の検証			
				1-2-2	画面、帳票の表示内容	レイアウトどおりか、項目は編集仕様どおりか、制定帳票・オーバーレイとの位置関係は適切か、等			
				1-2-7	限界値入力	フル桁入力、最大値・最小値入力、0値入力等、項目定義上の限界値及びその近辺の値を設定した単機能の検証			
				1-2-8	仕様上限値	仕様上限値が、システム全体で問題なく処理されるか			
				1-2-14	バックアップ、リストア	物理バックアップとリストア処理の検証			
		1-3	画面／帳票の操作性	1-4-1	見栄え、入力操作	全画面／帳票の操作性の検証（デザイン、見栄え、入力方式、入力順、エラーチェックタイミング、etc）			
				1-4-4	エンドユーザー操作性	表示や操作性を、エンドユーザーの視点での検証			
		1-4	帳票の出力先	1-5-1	印刷媒体	定められた出力媒体で印刷できることの検証			
				1-5-2	帳票出力先	各帳票がそれぞれの所定の出力先に正しく出力されるか			
		1-5	内部I／F	1-6-1	モジュール間I／F	モジュール呼び出し、及び戻りインターフェース項目バリエーションの検証			
				1-6-5	ジョブネット制御	ジョブスケジューラ登録によるJOB起動、ジョブネット起動条件の検証			
		1-6	サブシステム間I／F	1-7-1	モジュール間I／F（サブシステム間）	他サブシステムモジュール呼び出し、及び戻りインターフェース項目バリエーションの検証 （他サブシステムの部品を共有する場合で、契約範囲の制約、スケジュールの制約がある場合等）			
				1-7-5	データ項目バリエーション	送り側、受け側のデータ項目バリエーション検証			
		1-7	システム間I／F	1-8-1	システム間画面遷移	画面遷移図どおりか（システム間）			
				1-8-8	運用連携	システム間I／Fにより、相互の業務運用、システム運用がスムーズに連携するか			
		1-8	企業間I／F	1-9-1	転送方式	送信側、受信側それぞれの設定と転送ツール動作の検証			
				1-9-5	運用連携	企業間I／Fにより、相互の業務運用、システム運用がスムーズに連携するか			
		1-9	排他制御	1-10-1	オンライン排他	同一オンライン機能同士、異なるオンライン機能同士の排他制御の検証 Ａ．サブシステム間での排他の検証			
				1-10-4	バッチ非列処理	非列処理時に競合が発生しないか			
2	業務仕様	2-1	業務サイクル	2-1-1	定時の業務サイクル	日次、週次、各締日、月次、四半期、半期、期末（年次）、等、定時の業務サイクル処理の検証			
				2-1-3	特定日付での処理を交えた業務サイクル	年末・年始、特定日の重なった場合の処理の検証			
		2-2	業務運用	2-2-1	業務フロー	業務フロー通りに業務が実現できるか Ａ．業務運用テストが契約範囲の工程に含まれない場合			
				2-2-7	類似業務の運用	導入先類似での業務全般（内部統制の観点含む）			
3	セキュリティ	3-1	アクセス制御	3-1-1	画面利用権限	業務ポータル設定内容、業務メニュー、及び各画面機能内のセキュリティ動作の検証			
				3-1-3	データセキュリティ	データ参照が、権限にて許容された範囲に制限されているか			
		3-2	表示制御	3-2-1	個人情報非表示	個人情報が必要以外の画面や帳票に表示されていないか			
				3-2-2	帳号化	帳号化の必要な項目が、DBやファイル格納時帳号化されているか			
		3-3	総合	3-3-1	セキュリティポリシー	導入類似におけるセキュリティポリシー全般に関する検証			
4	性能・負荷	4-1	オンラインレスポンス	4-1-1	単一画面レスポンス	主要画面のレスポンス検証			
				4-1-2	ピーク時、高トラフィック時	主要画面のレスポンス検証、最大の同時アクセスユーザ数でのレスポンス検証			
		4-2	バッチ処理時間	4-2-1	単一ジョブ	バッチ単独での処理時間の検証			
				4-2-4	データ伝送時間	システム間データ伝送時間の検証			
		4-3	負荷	4-3-1	ボリュームテスト	大容量データ処理の検証			
				4-3-2	高頻度テスト	一定時間、処理能力が最大になる程度に同時に多数の処理を実行時の検証			
		4-4	総合性能	4-4-1	複合処理のレスポンス	オンライン、バッチ複合処理時の性能検証			
				4-4-4	本番性能テスト	導入先類似で要求性能が実現されるか			
5	障害回復	5-1	障害時処理	5-1-1	オンライン異常終了	オンライン異常終了時の対応手順の検証（アプリレベル）			
				5-1-3	機器障害	回線障害、サーバ障害、クライアント障害、通信機器障害時の手順の検証			
		5-2	システム間連携障害	5-2-1	送信相手先システム停止	相手先システム停止時の処理と、起動再開後の処理の検証			
				5-2-2	受信時自システム停止	自サシステム停止時の処理と、起動再開後の処理の検証			
		5-3	障害時運用	5-3-1	障害時の縮退運用	障害発生時に、稼動業務範囲を絞り込んだ縮退運用のの検証			
				5-3-2	障害回復後運用	障害回復後の運用再開、縮退運用から通常運用への戻しの検証			
6	システム運用	6-1	ジョブ・ジョブネット	6-1-1	シェル機能	指定されたプログラムが正常に起動されるか			
				6-1-5	外部接続先との運用連携	システム間I／Fにより、相互の業務運用がスムーズに連携するか			
		6-2	本番運用手順	6-2-1	オペレーション	運用担当者が問題なく運用できるか			
				6-2-5	構成管理手順	アプリケーションのリリース申請からリリース、稼動確認までの手順の検証			
7	システム環境	7-1	本番環境の確認	7-1-1	本番用モジュールの稼動	正しいバージョンのモジュールが本番環境にインストールされているか			
				7-1-4	本番環境での稼動	導入先類似での本番環境での検証			
8	移行	8-1	データ移行	8-1-1	変換機能	移行データのフォーマット、項目属性、コード変換は正しいか			
				8-1-5	移行処理時間	移行データの作成、転送、変換、格納の処理時間、手作業変更、登録の時間が所定の範囲内で完了するか			
		8-2	業務移行	8-2-1	業務切り替え手順	現行業務手順の停止と新業務手順の開始タイミングの検証			
				8-2-3	未完結データの処理	未完結データが既存システムに残る場合の業務処理の検証			
		8-3	システム移行	8-3-1	システム切替手順	既存システムの停止、I／F切替え、再開の手順の検証			
				8-3-3	システム切り戻し手順	切り替えできなかった場合に旧システムに切り戻す手順の検証			
9	ユーザビリティ	9-1	利用手順	9-1-1	業務の流れ	使用者が想定する業務の流れに沿った手順になっているか			
				9-1-4	業務達成	利用者が行いたい作業を確実に達成できるようになっているか			
		9-2	画面の構成	9-2-1	次の行動の理解	見てすぐに何をすれば良いのか利用者が気付けるか（利用者視点でのメニュー分類、目的の業務に紐付けられるボタン・メニューになっているか、など）			
				9-2-4	無駄の排除	無駄な情報やデザイン、機能を排除し、シンプルで分かりやすい画面になっているか			
		9-3	分かりやすさ	9-3-1	画面・帳票の用語・文章	操作の指示や説明、メニュー、項目名等は、利用者が理解できる用語になっているか			
				9-3-3	システムからのフィードバック	システムが処理している内容を利用者がすぐ分かるようになっているか（処理時間が長い場合は予めメッセージを出す、処理結果をフィードバックする、など）			
		9-4	エラーの防止と処理	9-4-1	入力項目の説明	入力に制限（必須、文字種、等）がある場合や、何を入力すべきか分かりにくい項目には、入力欄のそばに説明や入力例を記載してあるか			
				9-4-6	エラー発生時	エラーが発生したとき、エラーの発生原因だけではなく、対処の方法（戻るボタンを押す等）が分かるようになっているか、また、対処の方法が用意されているか			
		9-5	ヘルプ	9-5-1	ヘルプ・マニュアル	利用者が必要とする時に、業務の手順に合わせたヘルプ情報やマニュアル等を利用できるようになっているか			

4.3　テスト観点表の利用

テスト観点一覧の利用方法は,アジャイル開発でリリースする機能単位に必要な観点を選択できるようになっている.以下の(1)～(3)の手順で利用する.

(1)リリースする機能単位に必要なテスト観点を表２のⒶ→Ⓑ→Ⓒの順で確認し,Ⓕでフィルタリングして絞り込む.

(2)(1)で絞り込んだ観点についてⒺでテスト優先順を設定する.

(3)Ⓓの確認ポイント例を参考にテスト設計を行う.

上記がプロジェクトで活用する場合の利用方法になる.他にも品質保証担当者が第三者として,テスト品質の良否を確認するための観点として利用できる.

5. テスト観点の必要性の検証

本研究で作成したテスト観点表についてその必要性を検証するため，アジャイル開発を実際に行う開発メンバー，品質保証メンバー，アジャイル開発を支援するメンバーにヒアリングを実施した．

5.1 ヒアリング実施内容

(1) 目的

アジャイル開発のテストに活用するテスト観点の必要性の確認

(2) ヒアリング内容

①アジャイル開発の関わり度合や保有する知識有無の確認
②テスト設計実施時期
③テスト観点の利用状況
④テスト観点の必要性

上記内容について立場の異なる 10 名にヒアリングを実施した．

5.2 ヒアリング結果

(1) ヒアリング集計結果

表 3 で示すように「アジャイル開発を実施している」や「間接的に関わっている」と答えた人が 2 名と少なく，ヒアリング結果としては傾向を把握することはできなかった．テスト観点の利用については，1 名のみであったが，全員がテスト観点は必要と答えた．

表 3　ヒアリング結果

アンケート項目	分類	回答
アジャイル開発の関わり度合について	現在、開発を行っている	1
	間接的に開発に関わっている	1
	アジャイル開発に関わってないが知識あり	6
	アジャイル開発の知識なし	2
テスト設計の実施時期について	各スプリント開始後に実施	1
	テスト実施と並行して実施	1
	実施なし/わからない	8
テスト観点の利用状況について	観点利用あり	1
	観点利用なし	9
テスト観点の必要性/有効性について	必要である/有効である	10
	不要である	0

(2) テスト観点についてのコメント

テスト観点の必要性に関連するコメントがあった．

①アジャイル開発の共通観点はシステムによって異なるので統一できないのではないか．いくつかのパターンがあれば応用できるかもしれない．
②アジャイルだから気を付けること，システム規模や特性で選択できるといいと思う．
③どんな時に，どうやって確認するかの解説があると使いやすいと思う．
④多くの会社にあるウォーターフォール開発の観点をアジャイル用に転用できればアジャイル開発の敷居も低くなると感じる．

5.3　必要性の検証結果

表３のヒアリング結果が示すように，ヒアリングではアジャイル開発に対するかかわり具合や知識の有無，テスト設計時期とテスト設計時のテスト観点の利用状況については一定の傾向を示す情報は収集できなかったが，テスト観点の必要性については 10 名全員が必要と回答した．

6. 研究成果および考察

情報系システムのウォーターフォール開発で実績のある複数プロジェクトから収集したテスト設計のチェック観点をベースとして，共通項目抽出，記載内容汎用化，および品質特性別の分類を実施することで，アジャイル開発のテスト設計に活用できるテスト観点表を作成した．ヒアリング結果から，この観点はアジャイル開発に関わる組織に有効に利用されるものと考えられる．

本テスト観点表は，アジャイル開発で実行される探索的テストやユーザストーリーをベースとしたテストを行う際に活用することでテストの網羅性や信頼性を高める事ができる可能性がある．また，本テスト観点表は開発メンバーだけではなく品質保証担当者が第三者的にソフトウェアの品質を評価する際の観点として利用することもできる．

7. 今後の課題と展開

本研究では，アジャイル開発で活用するテスト観点表を作成できたが，その観点表を実際のプロジェクトに適用するところまでは実施できなかったため，その有効性や効率性の検証ができていないことが大きな課題である．さらに今回抽出した各観点項目の妥当性を確認していくことも今後の課題である．

図３はこれら課題を解消しテスト観点表をさらにより良いものにするための検証と改善のプロセスを示したものである．

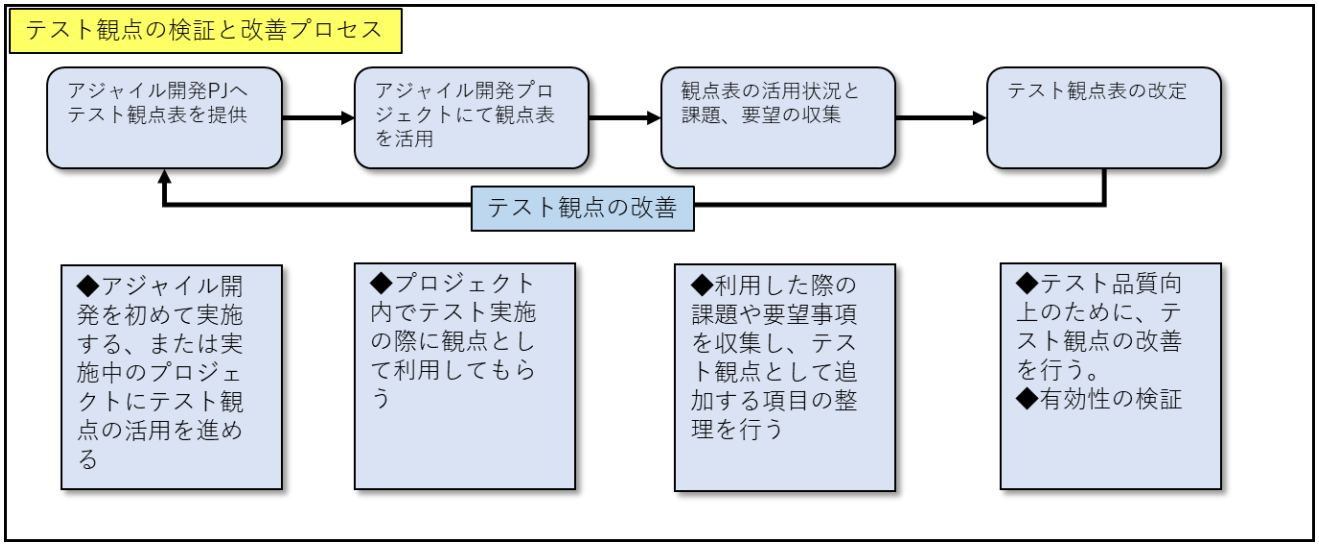

図３　テスト観点検証・改善プロセス

改善プロセスは以下の(1)～(4)のプロセス＋(5)～(7)の施策を順次実施していく．

(1)今回作成した汎用観点表をアジャイル開発プロジェクトに提供し活用してもらう．

(2)観点についての使い勝手や有効性に関する要望，課題を収集し蓄積する．

(3)蓄積したデータに基づき改善点を整理し観点表を改定し PJ にフィードックする．

(4)観点表の有効性の検証を実施する．

(5)使い勝手の改善や観点の選択漏れなどを防止するためのツール化をする.
(6)各プロジェクトよりプロジェクト固有の個別観点を収集する.
(7)今回作成したテスト観点にプロジェクト固有の個別観点を加えることで網羅性を向上させる.

上記の取り組みでアジャイル開発のテスト品質を向上させ,リリース後の障害発生の抑止に貢献できればと考えている.

8.参考文献

[1] ISO/IEC 25010:2011 Systems and software engineering -Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models
[2] SQuBOK 策定部会 SQuBOK Review 2016 Vol.1 4 4-1
[3] ISTQB Foundation Level Extension SyllabusAgile Tester Version 2014