アジャイル開発における段階的品質の積み上げによる品質保証

Quality Assurance by quality stepwise refinement in Agile Development

ウイングアーク１ｓｔ株式会社　技術本部　ソフトウェアプロセス&品質改善部
WingArc1st Inc. Software Process and Quality Improvement Department

○伊藤　潤平
○Jumpei Ito

Abstract

As agile development has been spreading mainly in Europe and the United States and in domestic vendors, but the quality assurance process in Agile development has not yet been established in the world. Agile development method is to continually provide working software and quickly respond to customer's needs, but it can be said that it is difficult to judge whether the quality is guaranteed in the product released for iterative. In this research, in the quality assurance activities against agile development, I propose a state where we can develop a strategic test plan and provide quality explanation of products released to iteratively. I also propose a framework that can comprehensively guarantee quality by stepwise refinement product quality.

1. はじめに

当社ではソフトウェアパッケージ製品の開発および販売を行っている。製品開発プロジェクトは開発プロセスと品質保証プロセスを区別し、独自ではあるがウォーターフォール開発を行っていた。近年開発プロセスをアジャイルに変えることにより、品質保証プロセスもウォーターフォールからアジャイルにシフトすることに課題を持っていた。

調べてみると 2001 年に「アジャイルソフトウェア開発宣言」が発表されて以降、アジャイル開発は欧米において開発手法のスタンダードになりつつある[1]。また、国内においてもアジャイル開発を導入する企業が増えている傾向にある[2]。「アジャイルソフトウェア開発宣言」では「動くソフトウェア」が重要な尺度といった説明はあるが、特に品質保証に対する定義はない。スクラムによるアジャイル開発で部分的な品質保証活動を説明している例[3][4]もあるが、総合的な品質保証活動に関するフレームワークはまだ世の中に確立されていない。

弊社の品質保証部門はプロダクトの出荷時に市場リリース可能な品質を持ったプロダクトであることをステークホルダーに説明する義務があり、第 3 者が理解可能な品質説明が求められる。そのため、開発プロセスがアジャイルでもウォーターフォールでもリリースするプロダクトに対して品質を保証する必要がある。

本論文では、アジャイル開発を行う際に品質保証プロセスにて、戦略的なテスト計画を策定して徐々に品質を確保することにより、確保した品質が積み上がっていることを確認するためのフレームワークを提案する。

ウイングアーク１ｓｔ株式会社　技術本部　ソフトウェアプロセス＆品質改善部
WingArc1st Inc. Software Process and Quality Improvement Department

新潟県新潟市中央区笹口 1-26-9　大和地所新潟笹口ビル 4F
Tel: 025-241-3108 e-mail: ito.j@wingarc.com
DaiwajisyoNiigataSasaguchiBldg.4F 1-26-9 Sasaguchi, Chuo-Ku, Niigata City 950-0911 Japan

【キーワード：】マスターテストプラン、フェーズテストプラン、アジャイル、スクラム、プロダクトバックログ、テストタイプ、品質特性、品質ゲート、終了基準（Exit Criteria)、ピックアップテスト

2. 提案するフレームワーク

アジャイル開発における品質保証プロセスの理想は、スプリント毎に品質データを提示することによって開発のリスクが明確になり、プロダクトのリリースが可能かどうか判断できることと考える。しかし現状では、非機能を含むシステムテストは機能が実装完了する開発後半で行っているため、スプリント毎に品質データを提示することが難しい。

そのため、本研究ではスプリント毎ではなく、フェーズという単位を用いて、そのフェーズ毎に品質データを提示することで徐々に品質を確保する仕組みを品質保証プロセスのフレームワークとして考案した。本フレームワークは、弊社の品質保証プロセス[5]や、見通しのよいテストの段階的詳細化の手法[6]を参考として、独自にアレンジを加えたものになっている。以降の節に考案したフレームワークの実施概要について述べる。

イメージ図 1　提案するフレームワーク

2.1 開発プロセスと品質保証プロセスの定義

弊社では開発チームと品質保証チームで組織が分かれており、製品開発プロセスがウォーターフォールかアジャイルかに関わらず、プロダクトの品質保証をする上では、品質保証プロセスを定義している。

開発プロセスはモノづくりにおける活動に重点を置く。一方で品質保証プロセスは品質の計測および分析の活動に重点を置き、製品の生産活動でどのように品質が確保されるのかを確認するためのプロセスと位置付けている。

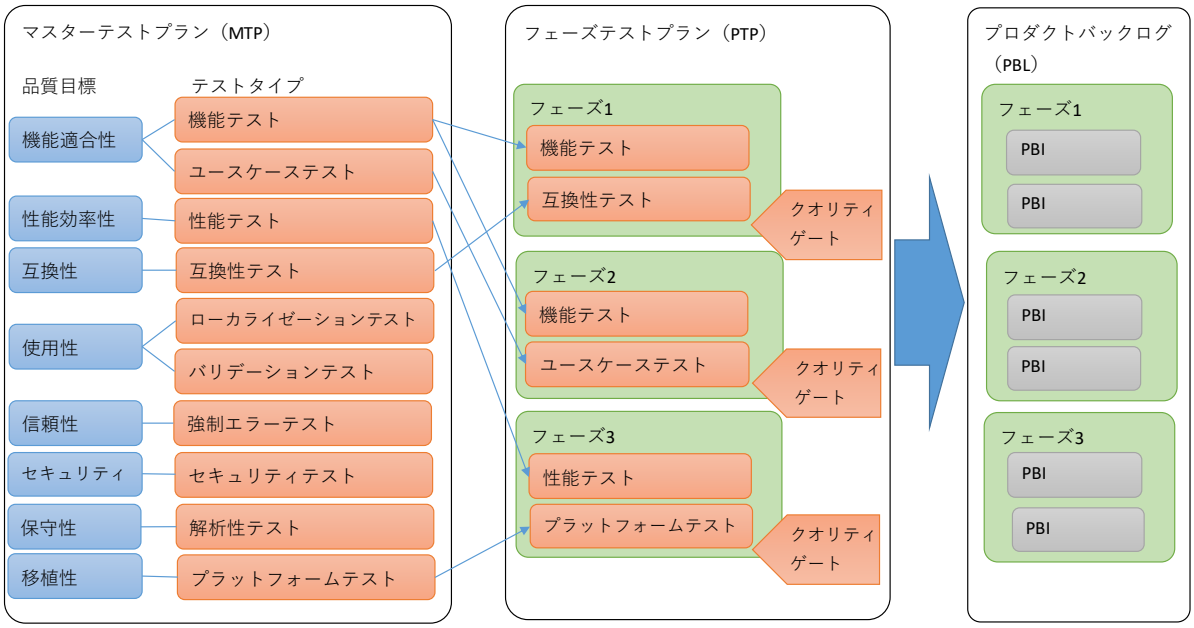
開発プロセスをウォーターフォールからアジャイルに変えることにより、同じ期間にレビューやテスト等、開発と品質保証の両方にまたがる活動があるため、プロセスを明確に分けることは困難になってはいるが、組織が分かれていることもあり、敢えてプロセスの定義を明確にして、役割を分けている。

2.2 戦略的テスト計画の作成

プロジェクト開始時に、JIS X 25010:2013 のシステム/ソフトウェア製品品質の品質特性[7]を利用し、マスターテストプラン（MTP）を作成する。MTP では、品質特性の副特性毎にプロダクトの品質目標を定義する。

次に、フェーズテストプラン（PTP）で各フェーズの単位でテストプランを作成し、フェーズ毎に達成すべき品質目標を細分化してテストタイプを定義する。これにより、品質要求として MTP で定義された品質目標をフェーズ毎に段階を経て実現するための戦略的なテスト計画ができる。

開発プロセスがスクラムの場合、プロダクトバックログ（PBL）には優先度順でプロダクトバックログアイテム（PBI）が並べられている。PTP の作成において PBI の内容を参照するが、優先順位が低い PBI の内容は粒度が荒い状態のため、プロジェクト後半のフェーズに対しては、PTP で定義する品質目標があいまいな状態になる。このような場合、PBI の内容が明確になった時点で PTP も明確化していく。



イメージ図２　戦略的なテスト計画のイメージ

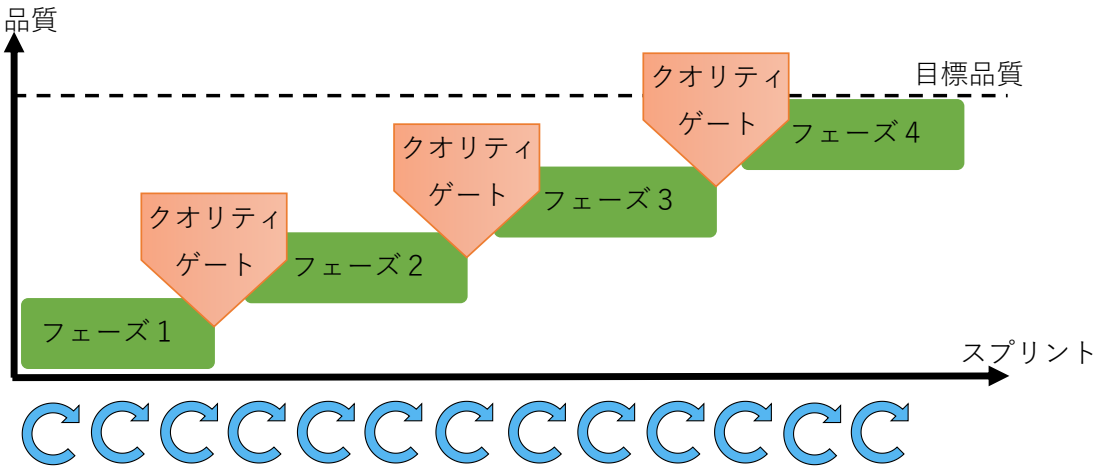
2.3　フェーズとクオリティゲートの設置による段階的品質の積み上げ

各フェーズにはクオリティゲートを設置し、終了基準（Exit Criteria）としてフェーズ内のテスト完了を目的に、フェーズ内で実施されたテストからサンプリングとしてある程度のテストケースを抽出して実行する。テスト実施後に定性分析や定量分析を行い、他に致命的な不具合がないか、類似の不具合が存在しないかを確認し、問題がなければクオリティゲート通過の判断とする。

このクオリティゲートをパスすることにより、少なくとも PTP で計画した品質特性を網羅する品質目標が実現可能であることが明確になる。またパスしなかったとしても条件付き合格のような基準を決めた上で、条件面だけを次のフェーズでテストするような運用をすれば、リスクは存在するものの、ある程度の品質が実現できることが明確になる。

各フェーズで設定した品質目標をクリアすることにより、最終的に MTP で計画した総合的な品質目標が確保されることが明確になり、プロダクトの品質が段階的に積み上がることがわかる。

また、2.2 節で述べたように、プロジェクト後半のフェーズに対する PTP はプロジェクト開始時には明確に定義できない場合がある。PTP が明確でないと、Exit Criteria の定義もあいまいになるため、PTP が明確になった時点で、Exit Criteria も明確化していく。



イメージ図３　フェーズとクオリティゲートの設置による段階的品質の積み上げイメージ

3. プロジェクトへの適用

3.1 背景

弊社のパッケージソフトウェア製品の一つである「SPA」の開発プロジェクトに適用した事例を紹介する。このプロジェクトは初期版をリリースした後の派生開発プロジェクトである。この製品は初期版リリース時に開発プロセスと品質保証プロセスを分けて実施しており、開発チームはウォーターフォールから初めてアジャイルに変更し、QA チームは従来のウォーターフォールで取り組んだ。結果的に既存の手法ではプロジェクト後半に品質が安定していないことが判明したため、品質保証プロセスを開始出来ず、リリース遅延のリスクが発生した。そのため、次期バージョンでは QA チームの品質保証プロセスをよりアジャイル開発に近づけ、プロジェクト開始時から品質を判断できる本手法を適用した。

3.2 戦略的なテスト計画の作成

プロジェクト計画時には戦略的なテスト計画として MTP を作成し、品質特性をベースとして品質目標を定義した。また、定義した品質目標を確保するためのテストタイプをマッピングすることにより、全体的なプロダクト品質が確保される仕組みを見える化した。表 1 ではわかりやすく主要な品質特性である機能適合性、性能効率性、信頼性を表現している。

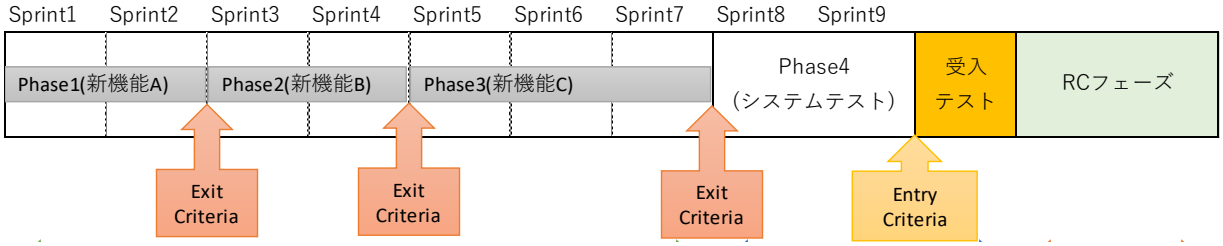
表 1. マスターテストプランの品質目標の定義とテストタイプのマッピング

品質特性	*副特性*	*品質目標*	*テストタイプ*
機能適合性	機能完全性	顧客要望が実装され機能する	要件確認テスト リリーステスト
		修正予定となっている潜在不具合がすべて修正されている	最終リグレッションテスト
	機能正確性	実装された機能が正しく動作する	新機能テスト
		母体機能が正しく動作する	既存機能テスト
		新機能と母体機能を組み合わせた動作が正しく動作する	新機能テスト 既存機能テスト
	機能適切性	想定するユーザーシナリオが満たされる	シナリオテスト
性能効率性	時間効率性	既存機能の処理時間が初期版と比較し劣化が発生していない	性能テスト
	資源効率性	新機能を含んだ一連の操作の断続的な利用で不正なハードウェアリソースの使用が発生しない	ロードテスト
信頼性	成熟性	初期版リリース以降に修正された不具合が全て修正されておりデグレードが発生していない	リグレッションテスト 最終リグレッションテスト
	可用性	新機能を含めてシステムに高い負荷が長期に渡り続いてもシステム全体が安定稼働する	ロードテスト
	障害許容性	機能追加した箇所で障害発生時にアプリケーションが致命的な状態に陥らず安全に停止する	強制エラー／リカバリーテスト
	回復性	機能追加した箇所で障害復旧時にユーザーの特別な操作なくシステムがリカバリーする	強制エラー／リカバリーテスト
	責任追跡性	新機能の操作がアクセスログに記録され操作をトレースできる	セキュリティテスト

MTP 作成後は PTP を作成し、各フェーズにて達成すべき品質とその品質を確保するためのテス

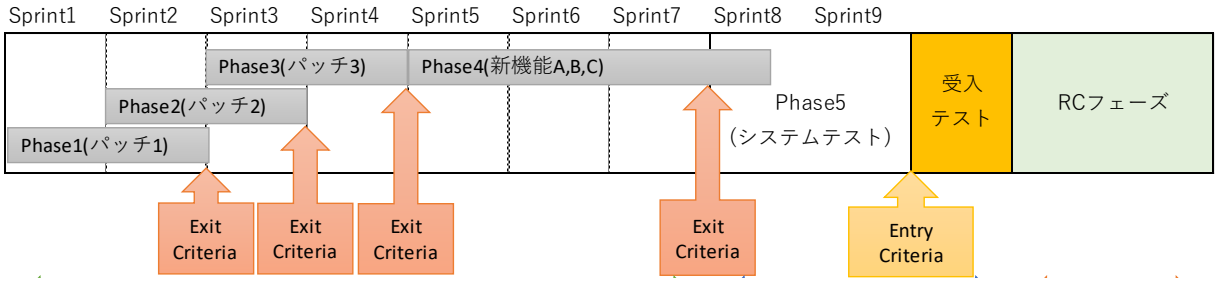
トタイプをマッピングする。テスト十分性については、全ての副特性に関するテストを該当するフェーズで必ず実行するように計画した。

ただし、プロジェクト開始時に全ての PTP の作成は困難であった。アジャイル開発は要求仕様がユーザーストーリーとして記述され、市場投入の価値の高いものから優先順位付けされる。そのため、プロジェクト計画時にはどの機能がいつ実装され、どのフェーズで対象となる機能がテスト可能になるかの判断ができない。以下のイメージ図 4 はプロジェクト計画時の PTP に反映されたスケジュールである。プロジェクトが予定通りに進行した場合は 3 つの新機能がフェーズ毎に実装され、リリース可能な状態になる。



イメージ図 4　プロジェクト計画時の PTP イメージ

実際に本プロジェクトではいくつかの既知不具合修正が割り込みとして入り、プロジェクト前半から開始していた新機能実装中に既知不具合を修正する計画がされていた。新機能実装はある程度進んではいるものの、プロジェクトが進むにつれて振り返りや、割り込みの不具合修正が多くなり、計画は常に変更されている状態になった。最終的に既存顧客からの要望である既知不具合対応は優先順位が高くなり、新機能実装は優先順位が低くなった。そのため Phase1～3 は不具合修正を対応してパッチリリースのタイミングでフェーズ分けし、Phase4 の中で全ての新機能をテスト可能な状態にし、テストを集中的に行った。以下はそのスケジュールのイメージ図である。



イメージ図 5　PTP の最終的なスケジュールイメージ

このように PTP はプロジェクト計画時は直近のテスト計画以外はあいまいな状態であり、プロジェクトが進行するにつれて明確化していく。これはスクラムの PBL と同じで、プロジェクトの進行と共に PBI の優先順位の明確化とテストフェーズが明確になる状態であることが分かった。

フェーズ毎の移行については、フェーズをクリアする条件としてクオリティゲートを設置し、Exit Criteria として「ピックアップテスト」と称するサンプリング検証を実施して問題がないことをクオリティゲート通過の判断としている。

ピックアップテスト実施のタイミングはフェーズの最終日もしくは翌日にフェーズ内で予定されたテストが全て完了した時点で開始される。ピックアップテスト中は開発は並行して次のフェーズのタスクに着手するが、もし不合格になった場合は不合格部分を最優先 PBI として次フェーズ初期段階で対応する。また、合格するまで再ピックアップテストは実施される。もし条件付き合格になった場合は、条件部分を PBI として妥当な優先順位で対応する。

プロジェクト最終時期に更新された PTP と、各フェーズで計画されたテストタイプ及び Exit Criteria は以下の表 2 を参考のこと。

表 2. フェーズテストプランとクライテリア

テストフェーズ	*期間*	*テストタイプ*	*クライテリア*
Phase1	Sprint1 Sprint2	9.3.4.1 パッチ検証 ● 要件確認テスト ● 既存機能テスト ● リグレッションテスト ● 強制エラー／リカバリーテスト ● ログ妥当性テスト ● パッチ適用テスト	ピックアップテスト 各テストタイプから約 5～20%程度を抽出して実施 すべてのテストタイプで不具合 0 件
Phase2	Sprint2 Sprint3	9.3.4.2 パッチ検証 ● 要件確認テスト ● 新機能テスト ● ローカライゼーションテスト ● バリデーションテスト ● リグレッションテスト ● セキュリティテスト ● ログ妥当性テスト ● パッチ適用テスト	ピックアップテスト 各テストタイプから約 5～20%程度を抽出して実施 すべてのテストタイプで不具合 0 件
Phase3	Sprint3 Sprint4	9.3.4.4 パッチ検証 ● 既存機能テスト ● リグレッションテスト ● パッチ適用テスト	ピックアップテスト 各テストタイプから約 5～20%程度を抽出して実施 すべてのテストタイプで不具合 0 件
Phase4	Sprint5 Sprint6 Sprint7	機能テスト ● 要件確認テスト ● 新機能テスト ● シナリオテスト ● ローカライゼーションテスト ● ユーザーエラー防止性テスト ● バリデーションテスト ● セキュリティテスト ● 解析性テスト ● プラットフォームテスト	・ピックアップテスト 各テストタイプから約 5～20%程度を抽出して実施 すべてのテストタイプで不具合 0 件 ・リグレッションテスト Phase1～3 で実施したテストについて機能完全性レベルのテストケースを抽出して実施 すべてのテストケースで不具合 0 件

3.3 フェーズ毎のクオリティゲート通過結果

フェーズテストの結果として、Phase 1～3 は不具合の修正に対するテストのため順調にクオリティゲートを合格し品質の積み上げに成功した。新機能に関してはプロジェクト前半から着手してはいたものの、本来 7 スプリントで完了する計画だったが、最終的にプロジェクト後半の 3 スプリントで集中的にテスト可能状態にし、Phase4 として予定されたテストを完了させた。Phase4 の PTP では新機能に対するテストが実行されるため、テスト範囲を明確にし、1 つのテストケースで複数機能を網羅できるようなテストタイプの作成を実現し、効率の良いテスト消化を目指した。また、性能テストについては開発者の実装環境で単体テストとして性能に影響がないことを確認し、システムテスト内で性能テストを実施することにより効率化を目指した。

結果的にクオリティゲートでの確認時にはレビュー不足やコミュニケーション不足があり、クオリティゲートで不合格が続いた結果、4 回もピックアップテストを実施している。イメージ図

5 で表現されている通り、Phase4 でピックアップテストを 4 回実施した遅延部分はシステムテストフェーズ内で予定されているテストと同時並行で実施された。システムテストフェーズ内ではテストタスクの優先順位を変更して新機能に影響しないテストを先に実施することで、新機能部分のテストを後半に実施できた。結果的にシステムテストフェーズ内で遅延部分のリカバリーが出来、全体的なプロダクトリスクが発生しなかったため、スケジュール遅延のリスクは解消された。

表 3. テストフェーズ毎のクオリティゲート通過結果

テストフェーズ		*結果*
Phase1		合格 ピックアップテスト（37.0%を抽出）を実施し、不具合検出がないことを確認した。※テスト実施者の工数的余裕があったため抽出率が基準より多い。
Phase2		合格 ピックアップテスト（36.4%を抽出）を実施し、不具合検出がないことを確認した。※テスト実施者の工数的余裕があったため抽出率が基準より多い。
Phase3		合格 ピックアップテスト（25.4%を抽出）を実施し、不具合検出がないことを確認した。※テスト実施者の工数的余裕があったため抽出率が基準より多い。
Phase 4	1 回目	不合格 ピックアップテストを実施し、2 件の不具合を検出した。6 件の追加検証を実施し問題ないことを確認した。
	2 回目	不合格 2 回目のピックアップテストを実施中に開発内で 1 件の不具合が検出された。1 回目のピックアップテスト後のレビューで QA の指摘内容が反映されていないことが原因だった。16 件の追加検証を実施し問題ないことを確認した。
	3 回目	不合格 3 回目のピックアップテストの結果、不具合を 1 件検出した。テスト設計者と実施者間で意思疎通が取れていないことにより、意図した検証となっていない問題であった。不具合の事象自体は限定的ではあったが、セキュリティ上の問題であるため修正が行われている。42 件の追加検証を実施し問題ないことを確認した。
	4 回目	合格 4 回目のピックアップテストを実施し、追加の不具合検出がないことを確認した。 テストケースを再抽出し、追加されたテストケースについては全 175 件を対象とした。また、Phase1 から 3 までの機能実装を確認するリグレッションテスト（テストケース数：9 件）でも不具合がないことから、今まで積み上げてきた品質に問題ないことが確認できた。 以上から、すべての Exit Criteria を満たす結果となり Phase 4 を通過及びシステムテストフェーズを開始できる品質であると判断している。

このように各フェーズでクオリティゲートを設置し、Exit Criteria を満たした上でゲートの通過を判断すれば次のフェーズへと進むが、不合格の場合はクオリティゲートを通過せず、Exit Criteria を満たすまで検証が行われる。ただし、Exit Criteria はあくまでも品質判断の材料として使われるため、クオリティゲート通過の判断をする際に定性分析を行い、条件が満たされていない場合でもリスクが明確になっており、次のフェーズで十分評価することが出来ると QA チームが判断した際は「条件付き合格」として、条件を提示した上でクオリティゲートを通過する

こともある。例えば Phase4 では新機能の限定的な箇所の不具合であるというリスクの特定ができた。そのため、テストの優先順位を変えてリスク以外の個所のテストをサンプル評価したところ、問題がなかったため、クオリティゲートを通過させてシステムテストフェーズを開始する判断をした。また条件部分はシステムテストと同時並行で実施され、プロダクトリスクが発生しないことを確認することで最終的に問題ないことを判断している。

ただし、役割として QA チームは品質データを提供し、クオリティゲート通過の判断をある程度行うが、最終的な判断は PO に委ねられる。

4. 今後の課題

本研究では品質特性に基づいてプロダクト品質のゴールを定義して戦略的なテスト計画を作成することにより、フェーズという単位で段階的に品質が積み上がることを可能にし、最終的にアジャイル開発で作成されたプロダクトの品質を保証するフレームワークを考案したものである。

ただし適用した事例は前述したパッケージソフトウェアに限定しており、汎用性の検証はできていない。今後は最もアジャイル開発が適するであろうサービスの分野で、本プロセスの有効性や改善点、課題などを確認したい。 また、本事例では担当者間でのコミュニケーション不足を起因とするクオリティゲートの不合格で追加テストが発生していることから、品質コストの削減についてのプロセス改善も課題としていきたい。

参考文献

[1] VERSIONONE、 State of Agile Survey 12th annual、 https://explore。versionone。com/state-of-agile/versionone-12th-annual-state-of-agile-report （2019 年 1 月 16 日現在）

[2] 独立行政法人情報処理推進機構 技術本部 ソフトウェア・エンジニアリング・センター、 非ウォーターフォール型開発の普及要因と適用領域の拡大に関する調査報告書（非ウォーターフォール型開発の海外における普及要因編）、 http://www。 ipa 。 go 。 jp/sec/softwareengineering/reports/20120611。html#L1 （2019 年 1 月 16 日現在）

[3] SQuBOK V3 研究チーム、 “アジャイル品質保証の動向“、 ソフトウェア品質シンポジウム 2016、 2016

[4] 永田敦、 “なんたって DevQA アジャイル開発と QA の合体が改善を生む“、 アジャイル冬の陣 2017、 2017

[5] 伊藤潤平、 “クオリティゲートの通過判断として品質特性を利用した受入テストの導入と効果“、ソフトウェア品質シンポジウム 2017、 2017

[6] 吉岡克浩、水野昇幸、西康晴、 “見通しのよいテストの段階的詳細化の手法 ーテストの網羅性確保の提案ー“、ソフトウェアテストシンポジウム 2013、 2013

[7] 日本工業標準調査会、 “ソフトウェア製品の品質要求及び評価（SQuaRE）-システム及びソフトウェア品質モデル”、 JIS X 25010、 2013