ISV アプリケーションの受入テストにおける探索的テストの導入効果

Effectiveness of adapting Exploratory Software Testing to Acceptance Test for ISV Applications

NEC パーソナルコンピュータ株式会社　第一商品開発本部　ソリューション開発部
NEC Personal Computers, Ltd., 1st Product Development Division, Solution Development Dept.
○森 一郎　荒木 伸昌 1)　東 正浩 2)
○Ichiro Mori　Nobumasa Araki1)　Masahiro Higashi2)

Abstract We bundle about 50 ISV (Independent Software Vender) applications to NEC's personal computer products. Our objective is to deliver products to our customers without any defects. In order to achieve this, our QA division conducts an audit during manufacturing to check whether there is no defect as a system. And before QA division's audit, our development division conducts a test called "Acceptance Test" to find defects of each application. ISVs provide us with their applications after their QA, but some applications don't meet our strict quality criteria. So we list-up all functions of the applications, create a check list of the functions, and confirm all the functions work well in Acceptance Test. The problem is that some errors are detected in our QA division's audit, although we have completed all function tests.

To solve this problem, we use Exploratory Software Testing method to expand test items. After a trial, we adapted this test method to actual products and increased man-power ratio of Exploratory Test gradually. As a result, both the number of detected errors and the efficiency of error detection increased in proportion to increase of man-power ratio of Exploratory Test.

In this paper, we introduce how to define Exploratory Test Process, how to organize Exploratory Test Team, how to make Test Viewpoint, how to keep man-power and the result of adaptation to actual products.

1. はじめに

当社は、PC 商品に自社製アプリケーションだけでなく、約 50 本の ISV（独立系ソフトウェアベンダー）製アプリケーションをプリインストールして出荷している。ISV による QA をパスしてから納品されたアプリケーションは、当社の受入部門による受入テスト、テスト部門によるシステムテストを経て、QA 部門による量産品監査を受け、重大な不具合がないことが保証され、出荷される。

受入部門の任務は受入テストの段階で、すべての不具合を検出し、ISV に修正を依頼し、量産品監査では不具合が検出されないようにすることにある。つまり QA 部門で行う品質保証よりも、不具合の検出と修正に重きを置いている。しかし、実際には量産品監査で不具合が検出され、ISV への緊急修正依頼、マニュアルへの注記、修正モジュールの Web 公開など、予定外の不具合対応に追われることが多い。受入テストに追加リソースを投入し、テスト工数を増やすことで不具合検出数を上げる方法は、開発費増につながり得策ではない。受入テストでいかにして効率的に多くの不具合を検出できるかが課題である。

神奈川県横浜市西区みなとみらい 3-6-1 みなとみらいセンタービル
Minatomirai Center Building, 3-6-1, Minatomirai, Nishi-ku, Yokohama-shi, Kanagawa Japan
Tel: 080-3579-3187　e-mail:imori@necp.co.jp
第一商品開発本部　ソリューション開発部　マネージャー
1) NEC パーソナルコンピュータ株式会社　第一商品開発本部　ソリューション開発部
NEC Personal Computers, Ltd., 1st Product Development Division, Solution Development Dept.
2) NEC パーソナルコンピュータ株式会社　第一商品開発本部　ソリューション開発部
NEC Personal Computers, Ltd., 1st Product Development Division, Solution Development Dept.

ISV アプリケーションの場合、各社の企業秘密である機能仕様書を入手できることはごく稀である。そのため当社の受入テストでは、納品されたアプリケーションを動作させて機能を洗い出し、テスト項目を作成し、テスト担当者がそのテスト項目に沿ってテストを実行するブラックボックステストの技法[1]を用いてきた。

一方、量産品監査ではテスト設計は行わず、QA 部門の勘と経験に基づきテストを実行するアドホックテストの技法[1]を用いている。量産品監査で不具合が検出されるということは、受入テストにおけるテスト観点が不足していることを示唆している。テスト観点拡充の方法として、不具合データベースを分析する方法[2]、担当者の経験に基づくノウハウを教訓集にまとめる方法[3]などが報告されている。当社では、Exploratory Software Testing[4]という文献で提唱されているテスト方法から想起されるテスト観点に基づき、探索的テストを実施することで、受入テストにおける不具合検出効率向上に成功したので、結果を報告する。

2 章では当社で従来から実施しているブラックボックステストの手法および課題を述べ、3 章で今回導入した探索的テストの手法と有効性検証結果について報告する。4 章では探索的テストを実際の商品開発に適用した結果を報告し、5 章で考察を加え、6 章で結論を述べる。

2. 従来ブラックボックステスト概要

2.1. テスト設計

前述の通り、ISV アプリケーションの場合、各社の企業秘密である機能仕様書を入手できることはごく稀である。そのため、OEM 向けの製品紹介資料、ヘルプやマニュアル等のユーザドキュメント、そして納品されたアプリケーションそのものを頼りにテスト設計を行うことになる。入手した資料やユーザドキュメントを見ながら、実際にアプリケーションを動作させて全機能を洗い出し、テスト項目を作成する。すべてのメニュー項目やボタンをテストできるようにするため、テスト項目はアプリケーション 1 本あたり平均で約 1,000 項目、多機能なアプリケーションでは約 4,000 項目に及ぶ。

テスト項目にはユーザ視点での期待結果も記入する。アプリケーションの動作確認において、期待結果が得られない場合には、開発元の ISV に対して“仕様確認”を行い、確認できた仕様に基づいて期待結果を修正する。

2.2. テスト実行

テスト項目作成後、テスト担当者がそのテスト項目に沿ってテストを実行する。納品されたアプリケーションで期待結果が得られない不具合現象に遭遇した場合は、直ちに Excel のスプレッドシートによる不具合リストに記録し、ISV に送付し、修正依頼を行う。

2.3. テスト設計における課題

全機能を確認しながらテスト項目を作成しただけでは、機能網羅性は高いものの、メニュー項目やボタンの単体動作確認に偏りがちとなる。例えば、複数の機能を連続的に使用する、複数の機能を並列で動作させるといった複雑なテスト項目は作られにくい。その一方で、多くのテスト工数を費やして実施したメニュー項目やボタンの単体動作確認で不具合に遭遇する可能性は低い。これはあくまで ISV での QA をパスして納品されたアプリケーションの受入であるから、単体で動作しない機能は少ないのである。

そこでこれまで当社受入部門のテスト品質改善の取り組みにおいて、複数機能によるテスト項目の追加、異常系テスト項目の追加、他のアプリケーションとの同時動作テスト項目の追加ということを地道に積み上げてきた。しかし、テスト項目は肥大化する一方であり、事業環境の変化により費用削減も進める必要が出てきたことから、これ以上のテスト項目追加はできなくなってきた。

2.4. テスト設計の改良

あらためて、これまで量産品監査で検出された不具合をリストアップし、その傾向を分析した。その結果、ユーザには実行可能な操作だが、全機能確認からでは想定できていなかった操作で不具合検出される傾向が見られた。例えば、「処理中に外部デバイスを取り外すとフリーズ」、「特定のボタンを連打するとエラー」、「アプリケーションが二重起動できてしまう」といった不具合である。これは、当社 QA 部門の勘と経験によるテスト観点が、受入テストには足りないためと考えられる。そこで、全機能確認ではなく、ユースケースの想定によりテスト観点を拡充すれば、受入テストにおける不具合検出数が増える、という仮説を立てた。

まずは直接的に、当社 QA 部門におけるテスト観点を取り込むべく、QA 部門のテスターにヒアリングを行った。しかし、テスト観点がテスター個人や実施時期によって異なり、明文化もされていないため、そのまま受入テストに持ってくることはできなかった。そこで当社受入部門のテスト品質改善活動の一環で、独自に、テスト観点の拡充を行うこととした。

テスト項目を追加するのではなく、テスト観点を拡充する。その具体的手法として、当社から ISV アプリケーションの受入テストを委託している中国のテストチームの発案で、文献[4]で提唱されている探索的テストを導入することとした。テスト担当者にとっては未知のアプリケーションをユーザ視点で操作しながら、不具合を検出していく探索的テストの手法は、ISV アプリケーションの受入テストには最適と考えたからである。

3. 探索的テスト概要

3.1. 探索的テストとは

探索的テストとは、1983 年に Cem Caner 博士によって提唱されたテスト手法の１つで、ソフトウェアの理解とテスト設計とテスト実行を同時に行うテストである[5]。テスト項目を作成しないことが特長であり、短時間ででき、テスト効率が高いとされているが、テスト担当者に高度なスキルが要求される手法である。また、どのような探索アプローチでテストを進めていくのか、どのような終了基準でテストを完了するのかを予め決めておくことがポイントである。

表 1 に一般的なテストケースベーステストと探索的テストの比較を示す。当社の従来テストはテスト項目を作成する手法であるから、このテストケースベーステストに該当する。表 1 を参照すると、当社で探索的テストの利点を生かして導入できれば、テスト効率の向上が期待できる。

表１　テストケースベーステストと探索的テストの比較（文献[5]を参考に筆者が作成）

	テストケースベーステスト	探索的テスト
プロセス	ソフトウェアを理解した後、テスト設計を行い、テスト項目を作成後、テスト項目に沿ってテストを実行	ソフトウェアの理解、テスト設計、テスト実行を同時に行う
テスト項目	作成する	作成しない
テスト工数	大	小
テスト効率	低	高
テスターの要求スキル	低	高
ドキュメンテーション量	大	小
ドキュメンテーションの例	テスト計画書、テスト項目、テスト項目の実行結果、不具合リスト、テスト報告書	タスク記述、探索アプローチガイド、終了基準、テスト結果、不具合リスト

3.2. 探索的テストプロセスの定義

当社の探索的テストは、ISV アプリケーションの受入テストを委託している中国のテストチームにおいて導入した。導入にあたり、まず、探索的テストプロセスを定義した。大きく分けて、(1)理解 (2)区分 (3)計画 (4)実行 (5)報告の 5 ステップで探索的テストを行う（表 2)。

表 2　探索的テストプロセス

ステップ	プロセス名	内容
(1)	理解	テスト担当者は、テスト対象となるアプリケーションの情報（マニュアル、類似したアプリ等）を入手し、対象アプリケーションの機能を理解する。
(2)	区分	テスト担当者は、理解した機能に応じて、各アプリケーションに文献[4]に基づく探索アプローチ（表 3）を割り当てる。1つのアプリケーションに、複数の探索アプローチを割り当てることもある。
(3)	計画	割り当てた探索アプローチに基づき、各アプリケーションのテスト時間を決定し、その計画が適切かレビューする。
(4)	実行	計画に沿ったテストをし、実行結果を記録する。
(5)	報告	検出された不具合を ISV へ報告し、修正依頼する。

表 2 を参照すると、ステップ(1)と(5)は従来テストと同様である。従来テストと異なるステップ(2)～(4)のうち、最も特長的なのはステップ(2)である。文献[4]には旅行者のメタファーを用いて表現された 33 種類の探索アプローチが書かれている。旅行者のメタファーには、例えばガイドブックツアー、博物館ツアー、徹夜ツアー、収集家のツアーなどがあるが、当社ではこの中から PC 向けアプリケーションのテスト観点を想起しやすい 10 種類の探索アプローチを採用した。ステップ(2)では、理解したアプリケーションの機能に応じて、どの探索アプローチを適用するかを決める。ステップ(4)では、テスト項目がないため、テスト担当者はテスト観点のみに基づき、テストを行う。

表 1 の定義と異なり、当社の探索的テストプロセスは、ソフトウェアの理解とテスト設計とテスト実行の同時性を必要条件としていない。これはスキルレベルの異なる複数のテスターから構成される中国テストチームでの導入を容易にするための措置である。

3.3. 探索的テスト体制

当社の探索的テストプロセスにおいて、テスト項目は作成しない。テスト担当者は、各アプリケーションの機能に基づいて割り当てた探索アプローチから想起されるテスト観点に基づき、アプリケーションを操作し、不具合を検出していく。従ってテスト担当者には、テスト観点のみに基づいてテストを実行する高度なスキルが要求される。

しかし、20 代の若いエンジニアで構成された中国テストチームのテスト担当者が全員このスキルを保有しているわけではない。そこでまず、スキル保有者による少数精鋭のマスターチームを編成し、それ以外のテスト担当者は、マスターチームの指導の下で探索的テストを実施できる体制を構築することとした。中国テストチームの中から、過去に探索的テストに近いテスト技法を経験した者 4 名を選抜し、マスターチームを編成した。

マスターチームは、探索的テスト理論の説明、テスト観点の洗い出し、ドキュメンテーション、テスト計画、テスト担当者への指導を行いながら、自らテスト担当者としてテスト実行も行う。

3.4. 探索的テスト観点

探索的テスト導入にあたり、マスターチームは、文献[4]から採用した 10 種類の探索アプローチごとにテスト観点を洗い出し、明文化した（表 3）。テスト観点は過去の受入テストにおける不具合事例や、マスターチームの勘と経験に基づいて想起し、具体的に表現した。

表 3　探索アプローチとテスト観点

No	探索アプローチ	テスト観点（例）
1	ガイドブックツアー	ヘルプのリンク先が正しく表示されるか
2	隣人ツアー	アプリの実行中に、イベントビューア、CPU 負荷、メモリ使用量を見て、問題ないか確認する
3	前バージョンツアー	過去バージョンの公開情報から、既知不具合事例を確認する
4	複数目的地ツアー	長いテストパスを確認する
5	スーパーモデルツアー	アプリのボタン状態（ボタン押下の状態、表示変化の状態）を確認する
6	わき役ツアー	二重起動しないか
7	奥深い路地ツアー	対象アプリと他のアプリが文書を共有するとき、正常に共有できるか
8	非社交的ツアー	アプリの実行中に、外部メディア（USB メモリ、SD カード等）を取り出しても問題ないか
9	間違った順序ツアー	インストール時、順番を変えて操作しても問題ないか
10	強迫観念にとらわれたツアー	各ボタンを連打しても問題ないか

従来テストのテスト項目（大項目）と、探索的テストのテスト観点を比較すると表 4 のような違いがある。テスト観点には単独の操作ではなく、可能な限り複合的な操作を記載するようにしている。また、開発元が想定する手順通りの操作ではなく、ユーザがやりそうな操作を記載する。

表 4　従来テストのテスト項目と探索的テストのテスト観点

従来テストのテスト項目（大項目）（例）	探索的テストのテスト観点（例）
外部デバイスにアクセスできるか	[非社交的ツアー] アプリが外部デバイスにアクセス中に、外部デバイスを取り外すと適切なエラーメッセージが出るか
インストールが成功するか	[間違った順序ツアー] アプリのインストール中に、ユーザが画面を回転させても、アプリのインストールが成功するか
ユーザがボタンを押すと、機能が切り替わるか	[強迫観念にとらわれたツアー] ユーザがボタンを連打しても、エラーなく、機能が切り替わるか

3.5. パイロットランによる有効性検証

探索的テストを実際の商品開発に適用する前に、まずその有効性を検証した。15 年春商品のうち 7 本の ISV アプリケーションを選定し、探索的テストのパイロットランを実施した。その結果、テスト工数あたりの不具合検出数を表す不具合検出効率は、14 年秋冬商品では 0.11 件/人日であったのと比べ、表 5 に示す通りパイロットランでは 3.9 件/人日となり、大きな効果が見込めることがわかった。

表 5　パイロットランでの不具合検出数

アプリケーション番号	種別	テスト担当者	マスターチーム担当者	テスト工数（人日）	不具合検出数（件）	不具合検出効率（件/人日）
001	新規採用	A	ー	1	6	6
002	新規採用	B	ー	1	9	9
003	過去に不具合発生	C	ー	1	5	5
004	過去に不具合発生	E	A	1	4	4
005	過去に不具合発生	F	C	1	3	3
006	過去に不具合発生	G	A	1	2	2
007	過去に不具合発生	H	D	1	2	2
					平均	3.9

4. 探索的テスト実施結果

4.1. 実施対象

探索的テストは従来テストでは不足しがちなテスト観点を補完し、不具合検出効率を上げる目的で実施する。従って、受入テストに割り当てられた総テスト工数は増やさずに、その一部を探索的テストに置き換える必要があった。そこで、従来テストのテスト項目のうち過去に一度も不具合を検出していないテスト項目の削除、前商戦期から変更を加えていないアプリケーションのテストサイクル削減（例えばβ版テストを省略）などの工夫により、従来テスト工数の一部を段階的に探索的テスト工数に振り向けることとした。

結果的に、本格導入を開始した 15 年夏では受入テスト総工数の 14%、15 年秋冬では 25%を探索的テスト工数に振り向けることができた（図 1）。

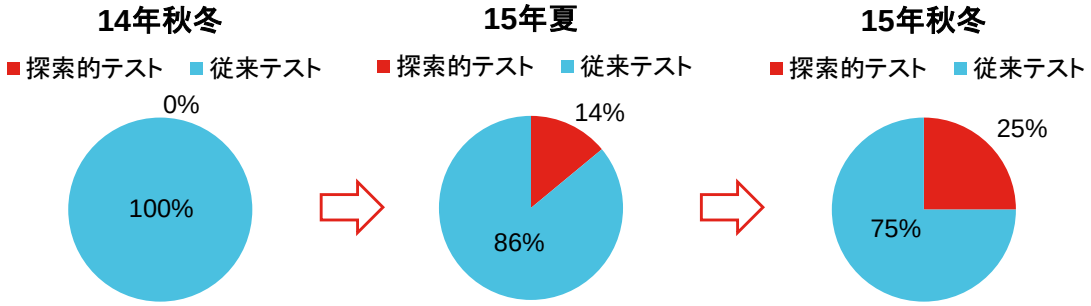

図 1　受入テスト総工数のうち探索的テストの工数割合

4.2. 不具合検出数

探索的テスト導入による不具合検出数の推移を図 2 に示す。図 2 を参照すると、探索的テスト初導入の 15 年夏では 185 件、15 年秋冬では 249 件と、不具合検出数が増加した。従って、探索的テストの工数割合を増やすことで、より効率的な不具合検出ができていることがわかる。

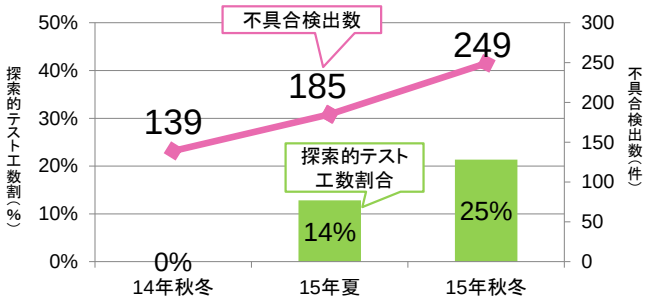

図 2　探索的テスト工数割合と総不具合検出数

4.3. 不具合検出効率

テスト工数(人日)あたりの不具合検出数(件)を、不具合検出効率(件/人日)と定義し、その値を計測した結果を図 3 に示す。

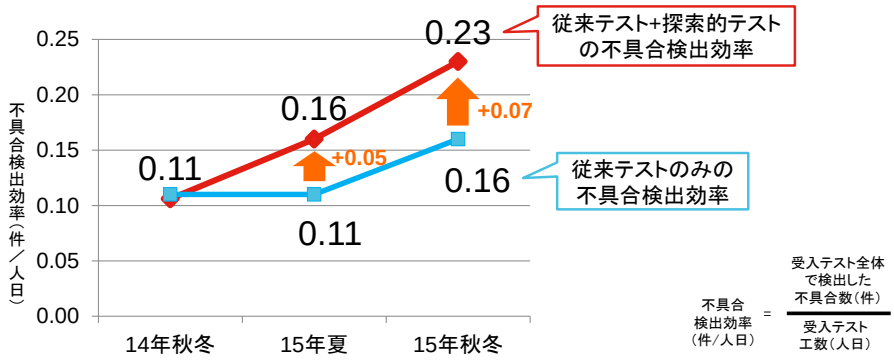

図 3　不具合検出効率の推移と比較

図 3 を参照すると、従来テストのみよりも、従来テストに探索的テストを組み合わせた方が、不具合検出効率が高い。また、従来テストのみで算出した不具合検出効率は落ちていない（0.11～0.16 件/人日）ことから、従来テスト工数の削減による逆効果はないと考える。15 年秋冬は OS のメジャーバージョンアップという不具合が混入しやすい時期であったため、従来テストのみでも不具合検出効率がやや上昇（0.16 件/人日）しているが、従来テストに探索的テストを組み合わせた値（0.23 件/人日）を見ると、その上昇分を上回る効果が得られたことがわかる。

5. 考察

5.1. 探索的テストで検出された不具合

従来テストに探索的テストを組み合わせることにより、不具合検出数が増加したので、実際に探索的テストで検出された不具合の内容を精査した。その結果、大多数は探索的テストのテスト観点によって検出された不具合であることがわかったが、探索的テスト実行中のテスト担当者が所定のテスト観点とは異なるテスト観点で検出した不具合も含まれていた。表 6 にその例を示す。これは、テスト対象となるアプリケーションを“探索”する過程で“新たなテスト観点”が生まれる可能性を示唆するものであり、探索的テストの利点の一つと考えられる。この点は今後さらにデータ収集し、テスト観点拡充のためフィードバックに活用したい。

表 6　探索的テストで検出された不具合の例

時期	不具合現象	探索的テスト中に実行したテスト
15 年夏	他のアプリケーションのウィンドウが最前面に来ない	他のアプリケーションのウィンドウをクリックし、前面に持ってくる
	アプリケーションが不正終了	住所欄に’a’を入力し続ける
	同じメッセージが 2 回表示	期限切れのアカウントを使用
15 年秋冬	ビデオの時刻が不正	ビデオ録画後、そのビデオをアップロード
	反応なし	検索条件を追加後、その条件を削除
	説明文の改行位置が不適切	データ復元画面を開く

5.2. 量産品監査に流出した不具合

不具合検出効率は高まったものの、量産品監査での不具合検出はゼロにはならなかった。15 年夏で 4 件、15 年秋冬で 5 件の不具合検出があった。これら 9 件の原因を分類すると、セキュリティソフトの影響 4 件、ヘルプ誤記 3 件、初回起動時の処理誤り 1 件、サーバからの更新通知誤り 1 件となる（表 7）。探索的テストでは、特定条件下で再現する低頻度の不具合を必ずしも検出できるわけではない。また受入テスト完了後に発生するようになった不具合は検出不可である。これらを考慮すると、探索的テストで検出可能であった不具合は 5 件であり、それらはセキュリティソフトの影響 2 件とヘルプ誤記 3 件である。

当社ではこの不具合分析結果から不足していたテスト観点を受入テストに追加し、2016 年以降

の商品開発で適用中である。

表 7　量産品監査で検出された不具合

時期	番号	不具合現象	原因	探索的テストでの検出可能性
15 年夏	1	動作停止	初回起動時の処理誤り	×（低頻度）
	2	インストール失敗	セキュリティソフトの影響	○
	3	起動不可	セキュリティソフトの影響	○
	4	表示不正	セキュリティソフトの影響	×（低頻度）
15 年秋冬	5	ヘルプと実機説明が異なる	ヘルプ誤記	○
	6	ヘルプと実機動作が異なる	ヘルプ誤記	○
	7	ヘルプと実機ボタン名が異なる	ヘルプ誤記	○
	8	インストール失敗	セキュリティソフトの影響	×（低頻度）
	9	更新通知のバージョンが古い	サーバからの更新通知誤り	×（受入テスト完了後に発生）

5.3. 探索的テストの課題 1

探索的テストの定性的効果を見るため、実際に探索的テストを実施したテスト担当者へヒアリングしたところ、探索的テストは新規採用アプリケーション、および、バージョンアップしたアプリケーションの受入テストの初回サイクル時には適しているが、継続採用アプリケーションや 2 回目以降のサイクルでは不具合を検出しづらい、との意見があった。これは、当社の探索的テストプロセスにおいて 2 回目以降のサイクルでテスト観点を変えていないことや、探索的テストが低頻度問題の検出に向いていないことに起因するものと推測している。この点は今後、定量的検証を加え、探索的テストと従来テストの最適な配分の定義につなげていきたい。

5.4. 探索的テストの課題 2

探索的テストにおいて、一度は不具合現象に遭遇したにもかかわらず、再現条件や再現手順がわからず、不具合調査が長期化したケースがあった。当社の探索的テストでは、不具合現象に遭遇した時点で、テスト手順を思い出してメモする方法をとっているが、テスト担当者の記憶に頼った方法と言える。これは予めテスト手順を決めない本手法の本質的な課題と考えられる。この課題については今後、画面キャプチャツール、ビデオ撮影などテスト手順記録方法の改善を検討し、有効性を検証予定である。

6. 結論

ISV アプリケーションの受入テストにおいて、従来のテスト技法に、“Exploratory Software Testing”のテスト方法から想起されたテスト観点に基づく探索的テストを組み合わせることにより、テスト工数あたりの不具合検出数を向上させることに成功した。

今後はテスト観点の拡充、テスト手順記録の改善を行い、QA 部門の量産品監査での不具合検出ゼロを実現できる ISV アプリケーションの受入テストプロセスを確立したい。

参考文献

[1] SQuBOK 策定部会編,“ソフトウェア品質知識体系ガイド”,2007
[2] 金子 昌永, “不具合と開発現場の実態に基づくテスト分析手法の提案”, ソフトウェア品質シンポジウム 2015
[3] 星名 卓郎, “製品検査における仕様書記載外の問題点摘出強化に向けたテスト観点拡充方法の検討と実践”, ソフトウェア品質シンポジウム 2015
[4] James A. Whittaker, “Exploratory Software Testing”, Addison-Wesley Professional, 2009
[5] 高橋 寿一, “知識ゼロから学ぶソフトウェアテスト（改訂版）”, 翔泳社, 2013